Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Service-oriented design of energy management systems for microgrids

Amimul Hossain

Course of Study:	Computer Science - STE
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Dr. Ilche Georgievski
Commenced:	September 16, 2019
Completed:	March 16, 2020

Abstract

Microgrids are low-voltage, reliable, economic, renewable energy resources that help to reduce the increasing demand of energy. Microgrids are environmentally friendly because of their renewable energy resources. The installations of various renewable energy resources are increasing day by day to produce vast amounts of energy. The management of produced energy is operated by a central software system, called Energy Management System (EMS). According to our literature review of microgrid EMSs, we notice that the capabilities of traditional EMSs are limited, and these are tightly coupled with the EMSs. This is considered as an obstacle for sustainable development of microgrid EMSs. We need EMSs whose capabilities have to be flexible, reusable, and interoperable against the traditional EMSs that ensure the sustainability of EMSs. Based on our literature review, we find that the Service-Oriented Computing (SOC) is a computing paradigm that helps to develop a flexible, reusable, and interoperable EMS. In this thesis, we propose to design EMS based on the design principles of SOC. We design five categories of RESTful Web services to develop service-oriented EMSs for microgrids. By using these RESTful Web services a sustainable micorgrid EMS can be developed. We design and implement a small service-oriented microgrid EMS by using our designed Web services that simulate distributed energy resources (wind turbine, phovoltaic) and distributed energy storage (battery). We use real energy consumption data with the simulated data to make the system operational.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

Microgrids are becoming more popular because of the integration of renewable energy resources. Renewable energy resources do not create any major adverse effect on the environment for producing electricity [40, 46, 55]. Usually renewable resources emit a small amount of carbon for power generation in contrast to the traditional power grids [40, 46, 55]. Due to the improvement of technologies, the climate change issues, and the demand of economic electricity, the traditional power grids are being reshaped day by day [16]. Currently, microgrids are considered as a promising technology for their reliability and economic supply of electricity [40, 55]. Generally, a microgrid is constructed based on energy storage, controllable and uncontrollable loads, and various renewable energy resources [2, 28, 52]. An Energy Management System (EMS) operates a microgrid, which is a software system that controls and monitors the loads, supply and demand of the microgrid [52]. An EMS is also responsible for making decisions on the economic power distribution of a microgrid. A microgrid EMS is needed to be scalable, reusable, and interoperable to support its enormous heterogeneous end-users.

According to the previous studies of microgrid EMSs, such as [1, 2, 5, 29, 46], most of the EMSs are tightly coupled with the microgrids. The entire EMS fully depends on its microgrids. A small modification of a microgrid directly affects the associated EMS and its energy consumers. A single EMS of a microgrid has to be developed multiple times depending on the executing environments. To design and develop a platform-independent microgrid EMS, it is essential to ensure scalability, reusability, and interoperability. Service-Oriented Computing (SOC) is a paradigm that utilizes services as the basic constructs to design and develop flexible, reusable, and loosely coupled systems [21, 36, 37]. SOC has been suggested as a suitable paradigm for the smart grid [35].

The aim of this thesis is to look at the functionality of microgrid EMSs from the service orientation point of view and design a service-oriented EMS for microgrids.

The organization of the thesis is as follows: Chapter 2 describes the related works of the thesis. Chapter 3 describes three basic parts as background knowledge such as microgrids, energy management systems (EMSs), and service-oriented computing. In Chapter 4, we provide a conceptual model of a microgrid EMS by considering the functional and non-functional requirements of microgrid EMSs. We also design Web services for the microgrid EMSs. Chapter 5 presents the implementation of the designed microgrid Web services. Finally, in Chapter 6, we discuss some future works and conclude the thesis.

2 Related Work

In this section, we discuss the papers related to service orientation of EMSs for microgrids. For each of the papers, we provide a summary, and try to find the missing parts that are important for our work. We mention very briefly our contribution against the missing parts of each of the papers.

Su et al. [52] describe an overall picture of EMSs in microgrid operation. They have mentioned different aspects related to the microgrid EMS operation. They identify several components of a microgrid EMS that are very much essential, such as distributed generator, distributed energy storage, controllable loads, critical loads and point of common coupling. The authors also mentioned some real-world examples of the centralized and decentralized control of microgrid EMSs. The most important fact is that the authors figure out four very important functional requirements of microgrid EMS. The functional requirements are optimization, forecast, human–machine interface, and data analysis. This paper gives the idea of what should be considered for a microgrid EMS but it does not clarify how to do that. For example, the paper just mentions the human–machine interface functionality, but it does not point out any graphical instance or what are the common properties of the human–machine interface. We design a conceptual model for microgrid EMSs where we describe each of the basic functional requirements separately.

Lee et al. [28] propose a microgrid platform for efficient advanced microgrid EMS operations. To design and implement the proposed microgrid platform, the authors focused on two design issues: first, the basic microgrid EMS functionalities, such as optimization, forecast, human–machine interface, and data analysis, and second, the microgrid engineering challenges, such as interoperability, extensibility, and flexibility. Energy Service Interface (ESI) is an essential component of their microgrid platform which works as a medium of energy services from both the facility perspective (Smart appliances, EV) and the grid perspective (Automated Demand Response, Market information). It is mentioned that the microgrid platform delivers some basic services, such as historical energy data, forecast energy data, information related to consumption, generation, and storage etc. But there are no hints about the interfaces of their services. To develop a service-oriented microgrid EMS, we design five categories of Web services and each of the services has multiple interfaces with accessible methods, resources, and a brief description.

Shin et al. [49] designed Application Programming Interfaces (APIs) for Web services of microgrids. The Web services are implemented using the RESTful mechanism together with an XML-based data transfer format. Among the designed microgrid Web services are the historical data service, on-demand data service, statistic data service, data analysis service, environment data service, distributed energy resource (DER) schedules and controls services. The authors provided the idea of what a microgrid could be at a service level, but not what it could represent at a system level, i.e., how to compose an energy management system out of such Web services. The authors do not explicitly mention the Web service design principles, such as reusability, loose coupling etc. of their designed APIs. To design the microgrid Web services, we apply some design principles (autonomy, reusability, loose coupling) of the Service-Oriented Computing (SOC).

Eger et al. [13] describe the functional architecture for microgrid as part of the FINSENY project. In that paper, they fully describe the functional, information, communication, and component layer of microgrids. In the functional layer, they describe a lot of functions, such as forecasting, balancing supply & demand, planning, DER monitoring. They also provide a graphical model for each of the functions that helps to understand the overall concept of the functions. They also describe the component as well as the corresponding use case for each of the functions. These kind of functions give us an idea to determine the functional building blocks of a microgrid. In that paper, they provide a long list of microgrid functionalities from which we choose the functionalities that are essential for a small-scale microgrid EMS, and construct our conceptual model of microgrid EMS.

Chen at al. [8] propose a conceptual Web services infrastructure for power systems. The aim of this Web services infrastructure is to develop an open, flexible and scalable framework that ensures better cooperation and integration capability among various users, applications and utilities. The backbone of the proposed Web services infrastructure is based on the Service-Oriented Architecture (SOA) and Web services technologies. Web Services Description Language (WSDL) is used for the service description, Universal Description Discovery and Integration (UDDI) is used for service discovery and identification, Simple Object Access Protocol (SOAP) is used as an access protocol, and Extensible Markup Language (XML) is used as a data exchange format. This infrastructure is compatible for operation-oriented services such as Energy Management System (EMS), Distribution Management System (DMS) and Supervisory Control and Data Acquisition (SCADA), business-oriented services such as enterprise resource management, trading and maintenance-oriented services. Actually, this Web service infrastructure is a one kind of general structure for information integration in power systems. In their proposed model, there is lack of concrete specification that can be especially applied in microgrid EMSs. In that model, they do not clearly mention the functionalities of the power systems that can be applied in microgrid EMSs. As microgrid EMSs power management are different from the conventional power systems, so, the functionalities of microgrid EMSs are not similar to the conventional power systems. We discuss the main functionalities of microgrid EMSs as well as we design some Web services that can be used to achieve those functionalities.

3 Background Knowledge

In this chapter, we discuss three parts closely related to our thesis topic: microgrid system, microgrid energy management system (EMS), and service-oriented computing (SOC). In the microgrid system part, we discuss the components, characteristics, and mode of operation of microgrids. In the microgrid EMS part, we discuss the control, functionalities, and challenges of a microgrid EMS. Finally, in the SOC part, we discuss the service design principles, REST-based Web service, and RESTful Service Description Language (RSDL).

3.1 Microgrid System

A microgrid is a low-voltage distribution network that is generally formed by combining a collection of loads, distributed energy storage (DES) and distributed energy resources (DERs). The definition of microgird is not unified yet. The microgrid definition varies depending on various loads, storages and resources.

Wencong Su et al. define a microgrid as a distribution network of low-voltage energy resources which is placed in a distribution substation by using a point of common coupling (PCC) [52]. Eun-Kyu Lee et al. define a microgrid as a low-voltage distribution network [28]. It is formed by using different types of components such as controllable energy loads and DERs [28]. Amjad Ali et al. define a microgrid as a network of different DERs, energy storage system (ESS), loads, supervisory control, protection and energy management systems [2]. For the U.S Department of Energy, a micrgrid forms electrical boundaries based on interconnected various loads and a collection of distributed energy resources and it is considered as a single controllable unit from the perspective of main grid [15].

The common parts in all definitions above are:

1. **Distributed energy resource.**
2. **Energy storage capacity.**
3. **Loads (household, commercial etc.)**

Figure 3.1 represents a basic conceptual design of a microgrid considering the most common parts.

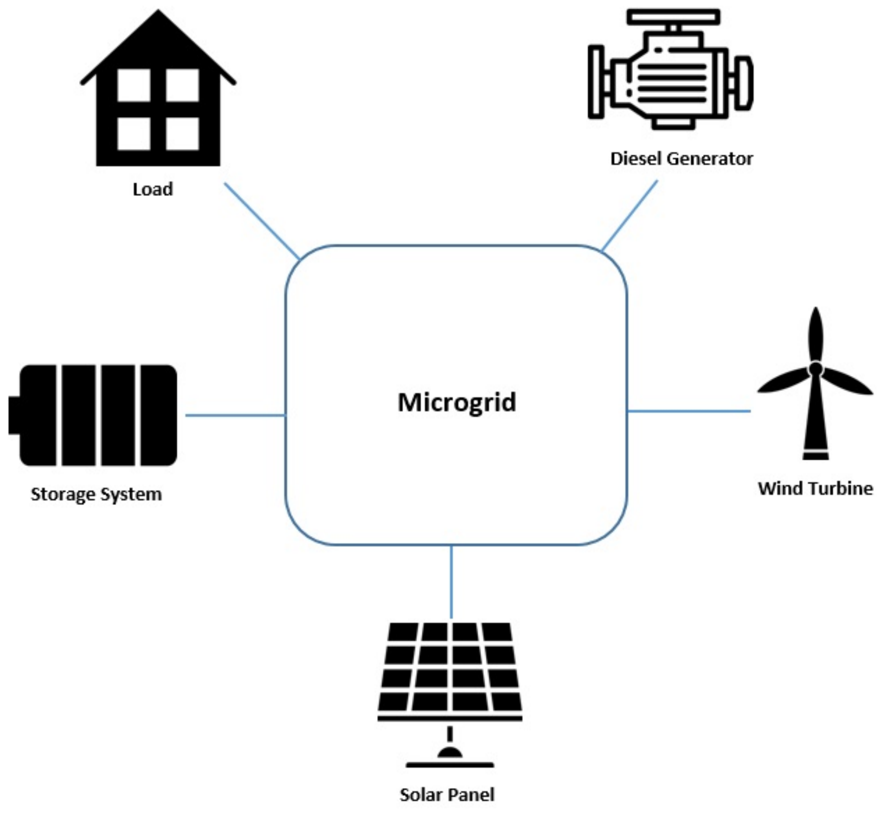


Figure 3.1: A conceptual design of a microgrid

3.1.1 Components of Microgrid

Depending on the capabilities and environmental or geographical conditions, the infrastructure varies from one microgrid to another. As the infrastructure of a microgrid is not unified, components of microgrid are not also globally unified [52]. Nonetheless, there a few basic components that are usually used in each microgrid. Figure 3.2 shows a basic infrastructure of a microgrid. Each of the components are discussed in the following [30, 52].

Distributed Energy Resources (DERs)

Distributed energy resources are the energy supplier of a microgrid system. They usually produce 200kW of power [39]. Actually, it is a low-voltage energy source for a microgrid. Different kinds of renewable and non-renewable energy sources are used to produce energy. These are listed in Table 3.1. Normally, DERs are installed near to the end-user in contrast to the traditional power grid. Recently, the renewable energy sources are the main concern to generate power to emit less CO_2 to the environment. So, the massive production of power from the renewable energy resources will create a positive impact on the environment. To generate more power from the renewable energy sources, always consider few challenges that are as follows [30]:

- Geographical location.

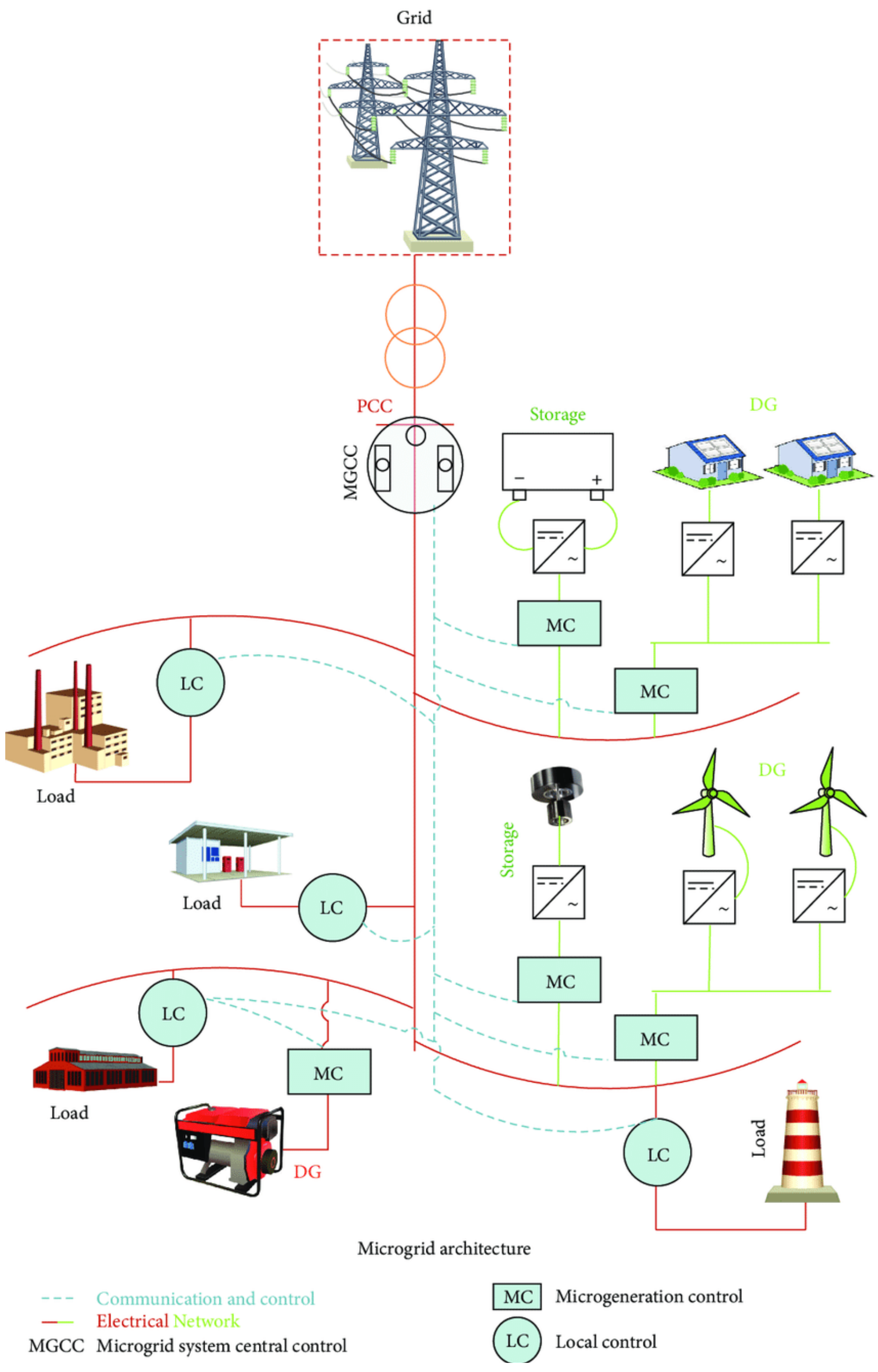


Figure 3.2: A simple microgrid infrastructure [30]

- Controlling of generated power.
- Power conversion (Alternating current to Direct current or Direct current to Alternating current).

Combined Heat and Power (CHP) technologies provide a remarkable characteristic in microgrid systems [39]. It can parallelly produce electric power and usable heat which carries a profound advantage for micrgrids. The efficiency of a CHP system is around 80 to 85% in contrast to conventional generators which are efficient only up to 35%. According to Wencong Su el al., CHP will be the heart of microgrid economics [52]. A few DERs such as diesel, fuel cells generate power based on fuel. These fuel based sources are controllable according to consumer needs but these are mostly depend on operating costs (fuel cost). Optimal scheduling is a vital aspect for DERs to produce the required power based on geographical conditions and operating costs.

Table 3.1: Renewable and Non-renewable DERs [20, 30, 52]

Renewable	Non-renewable
Wind turbine (WT)	Coal
Solar panel or photovoltaic (PV)	Diesel
Micro-Hydropower	Furnace oil
Geothermal	Natural gas
Biomass	Fuel cells

Distributed Energy Storage (DES)

The main functionality of DESs is to maintain the reliability of microgrids. In off-pick time, it will store excessive energy and in peak time, it will provide energy to balance the demand and supply of microgrids. They also play an important role in saving electric energy cost optimally. The operating behaviour of DESs depends on the operational mode of microgrids. For example, in islanded mode (Chapter 3, Section 3.1.3), DESs are able to give energy to the microgrid to support consumers, and in interconnect mode (Chapter 3, Section 3.1.3), DESs store low-cost electricity as well as maintain stability of DERs [52]. The energy storage devices used in micogrids are the following [30, 39]:

- Battery,
- Ultracapacitor,
- Supercapacitor,
- Flywheels or Superconducting Magnetic Energy Storage (SMES),
- Compressed air energy storage.

The installation of energy storage devices depend on a microgrid's capacity, loads and other factors. The installation of suitable storage devices could provide a better management of microgrid energy. An ultracapacitor is a suitable option if huge amount of energy needs to be charged or discharged within a very short period of time. Flywheels could be another choice to store energy if the production of renewable energy is greater than the required load and long period of discharging time is not a concerning issue. For a commercial flywheel, the discharging time normally varies between 15 minutes to few hours [39]. However, its charging efficiency is 90% [30].

Listing 3.1 Type of loads [52]

1. Controllable load
   - Heating, ventilation, and air conditioning (HVAC) system
   - Plug-in hybrid electric vehicle (PHEV)
   - Plug-in electric vehicle (PEV)
   - Household
2. Critical (uncontrollable) load
   - School
   - Hospital
   - Commercial, industrial etc.

For an effective distributed energy storage, the charging and discharging rate is not only a single factor but some other factors, such as energy price, service life, maintenance etc. have to be considered. For example, a Lithium-ion (Li-ion) battery could be a good option for electric vehicles but not for microgrids because of its high energy cost. Its cost is around 500-1500$/kWh [39]. If we consider the service life of an energy storage device, the flywheels are the best choice when compared with batteries and supercapacitors. A flywheel's service life is 20 years, and 5 and 10 years for a battery and supercapacitor, respectively. In terms of maintenance, a supercapacitor is the perfect choice because it does not need any maintenance.

Loads

In microgrid systems loads are treated as demand of energy. The loads of a microgrid system can be divided into two types, such as controllable load, Critical (uncontrollable) load. Listing 3.1 represents the types of loads that are generally used in microgrid systems. Critical loads are usually static load, their priority is more important than the one of the listed controllable loads. To ensure the reliability of microgird power supply, some of controllable (non-critical) loads can be disconnected [39].

Controllable loads are adjustable from the consumer side. In microgrid systems, the consumers can set the priority of controllable loads in contrast to the traditional power system. For example, a household lighting system can be controlled by the consumer to save energy cost [18]. Plug-in hybrid electric vehicle (PVEV) and Plug-in electric vehicle (PEV) are potential examples for controllable loads. Vehicle to gird (V2G) technologies provide reverse flow of energy, where PVEVs/PEVs can give energy back to distribution network in peak periods of load [52].

Point of Common Coupling (PCC)

Point of common coupling is a meeting place for the distribution network, customer interface and power production [52]. PCC plays an important role for the microgrid's mode of operation. In the grid-connected (interconnected) mode, PCC is attached with the microgrid but in islanded (autonomous) mode, PCC is detached from the microgird. Figure 3.3a and Figure 3.3b represent the two cases of PPC.

Communication System

Communication system is an essential part of micorgrids. In a microgrid, the heterogeneous components communicate through various methods and protocols. What types of communication method and protocol will be used depends on the infrastructure of the microgrid and supporting technology of the microgrid's components. A robust communication system is also important for the energy management of any microgird. The different methods of communication are listed in Table 3.2.

Table 3.2: Microgrid communication methods [39]

Wired	Wireless	Network & Protocol
Power Line Communication (PLC)	WiFi (5–10Mbps)	LAN
Ethernet	WiMAX (75Mbps)	WAN
Broadband over power line	ZigBee	TCP/IP
Leased telephone line	Wireless radio communication	
Optic fiber		

3.1.2 Characteristics of Microgrids

Microgrid is an economic power supplier that assists consumers to save the power cost. The infrastructural design and components of a microgrid are different from those of a traditional power provider. Microgrid have a few unique characteristics in contrast to the traditional grid. According to [1, 25, 52] and [20], some of microgrid characteristics are as follows:

- Low-voltage distribution networks.
- Power generation, consumption and storage follow a hierarchical control process.
- PCC is used to connect or disconnect the main grid.
- Control of electrical components to ensure reliability of electrical power.
- Management of controllable loads based on demand of energy.
- Two basic modes of operation
  - Islanded mode.
  - Interconnected mode.

- In interconnect or islanded mode of operation, controlling energy supply optimally.
- Provide **plug and play** features.
- Provide **peer to peer** functionality.
- Follow some essential protection procedure.
- Considered as a single controllable unit from the perspective of the main grid.
- Depending on electric current, it can be modeled in one of the following three types
  - Alternative Current (AC) microgrid.
  - Direct Current (AC) microgrid.
  - Hybrid (AC-DC) microgrid.
- Provide electrical energy and thermal energy.
- Ability to handle unpredictable situations in case of faults or other incidents.
- Support two types of communication network technology
  - Wired: Ethernet, Power Line Communication (PLC).
  - Wireless: Wifi, ZigBee.

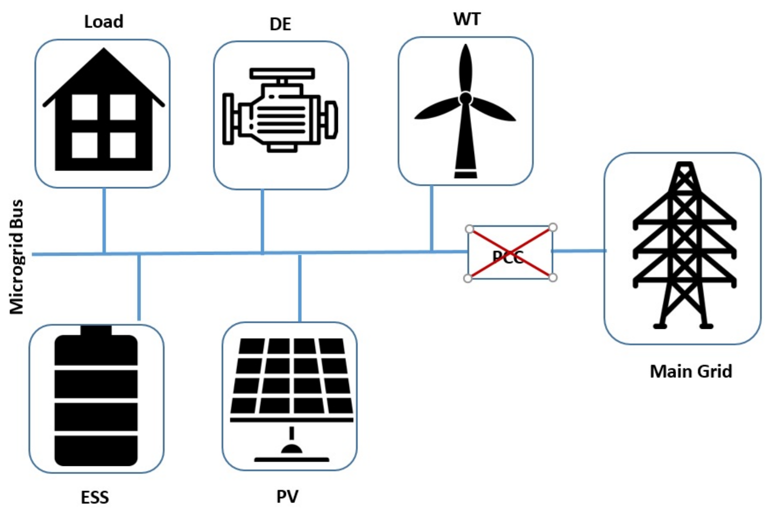
3.1.3 Microgrid Mode of Operation

As we know from Section 3.1.1, the microgrid structure is different from the conventional grid system. It is mostly composed of various renewable energy sources, such as wind turbines, solar panels which are fully dependent on the environmental conditions (temperature, wind speed, solar irradiation) [48]. According to the behaviour of environment, there is a possibility to increase or decrease the energy production of microgrids. Distribute energy resources (DERs) produce variable amount of energy while loads are eventually constant. So to optimally support the requested demand of energy as well as to balance the supply and demand of energy with the help of DESs, the microgrid system has to operate in different modes namely, islanded mode and interconnected mode.

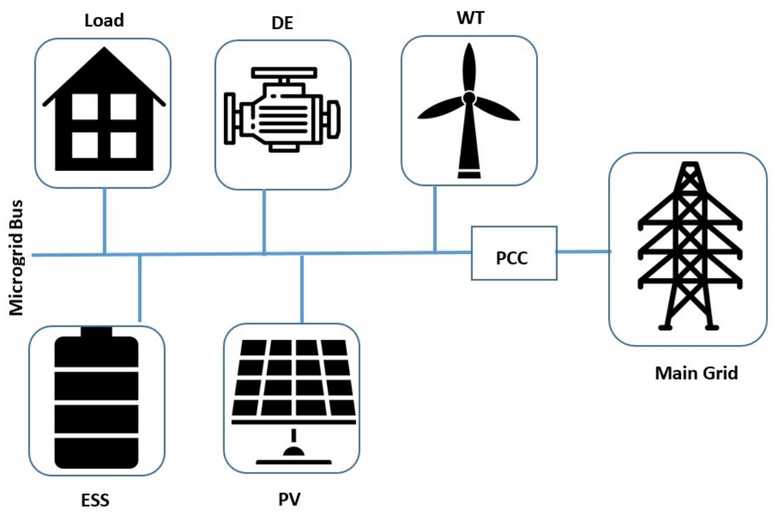
Islanded Mode

A microgrid is in islanded or standalone mode if it is disconnected from the main grid. It also named as autonomous mode [52]. Typically, the islanded mode of operation is not a good practice for a microgrid. As we know, the production of energy in a microgird is not static, the handling of various loads in different times of the day becomes problematic. On the other hand, because of being independent from the main grid, it is always functional if there is any major disturbance happening or any unpredictable blackout incident occurring in the main grid. To maintain the power balance, the islanded mode of microgrid can perform two actions [25]:

1. Supply surplus: Charge action of DES and/or decreasing generation.



(a) Islanded mode of microgrid operation [50]



(b) Interconnected mode of microgrid operation [15]

Figure 3.3: The modes of microgrid operation

2. Supply shortage: Discharge action of DES and/or load-shedding.

Figure 3.3a represents the islanded mode of the microgrid operation.

Interconnected Mode

A microgrid in interconnected mode is also called a grid-connected microgrid. The microgrid is actually connected with the main grid through the substation transformer. A microgrid can take advantage of being in the interconnected mode to main its demand and response of energy. If the energy production of distributed energy resources is grater than the demanded load, the energy storages will be charged at first and, if there is still some excess of energy, it will be transported to corresponding connected grid. The opposite phase is applicable if the demanded loads are grater than the produced energy of DERs [29]. Figure 3.3b represents the interconnected mode of a microgrid operation.

3.1.4 Issues of Microgrids

There are few issues that should be considered for each microgrid system. The important issues are listed below:

- Voltage and frequency control of microgrid [44],
- Selection of energy storage systems by considering cost, service life, maintenance, charging and discharging rate etc. [30],
- Design of microgrid based on geographical conditions [50],
- Protection of microgrids [44, 50],
- Instability of controllable loads [52],
- Islanding of microgrid [40, 44, 50].

3.2 Microgrid Energy Management System (EMS)

To manage the loads and real-time operating conditions of microgrids, a central software system is used, called Energy Management System (EMS). An EMS performs multiple tasks simultaneously, such as collecting and forecasting energy production, storing and demanding information from DERs, DESs and loads, respectively, maintaining an equilibrium level of energy based on demand and supply of energy, maintaining an interaction with the heterogeneous users. The short-term power balancing and long-term energy management requirements are maintained by the EMS. Listing 3.2 represents those requirements.

3.2.1 Supervisory Control of Microgrid EMS

Efficient control strategy is obviously essential for any kind of management systems. It helps to achieve the desired goal of a system. An EMS is a heterogeneous component-based energy management system, whose control strategy can be of two types [40]:

1. Centralized control
2. Decentralized control

Both centralized or decentralized control follow a simple hierarchical structure. This structure is divided in three levels as follows [24]:

1. Distribution Network Operator (DNO) and Market Operator (MO),
2. Microgrid Central Controller (MGCC),
3. Local controllers (LCs).

Listing 3.2 Short-term and long-term energy management requirements [24]

1. Short-term power balancing
    - Capability to maintain the power quality constraints of critical loads.
    - Being able to share real power between DERs that ensure appropriate control of voltage and frequency as well as load-following capability.
    - Capability to maintain real-time response and frequency or voltage restoration at the time of power variation.
2. Long-term energy management
    - Concentration for each DER requirements or limitations, such as generation cost, maintenance cost, unit type, service life, environmental conditions etc.
    - Capability to properly maintain the power demand-response and also to restore controllable loads (nonsensitive loads) that are not taken in consideration at the time of power shortage.
    - Capability to always maintain an adequate level of storage capacity that would be applicable for an optimal operating point of DER generation, and an optimal operation point is fixed based on some optimization processes. For example, maximize the renewable energy generation and minimize the generation cost.

Centralized Control of EMS

The centralized control of EMS is a root decision point for the entire microgrid system. To take an economic, cost effective and goal oriented decision, it must depend on associated LCs. Each of LCs communicates with MGCC to provide information based on their own setting. For example, energy production status, storage condition and load requirement information are provided by LC of DER, DES and load unit, respectively. A two-way communication medium is used by MGCC and LCs to perform communication between them.

MGCC receives bids from LC of DER and load. After that, by considering the market price, it defines an optimal energy scheduling (set points), and the aim of the scheduling is to fulfill the objective function, such as minimization of power cost, maximization of renewable energy usage, and economic balance of power supply and demand requirements. MGCC delivers the defined energy scheduling decision to the LCs [55]. In addition, MGCC observes the operational condition of the whole system and decides whether to change the operational mode (interconnected or islanded) of the microgird.

The centralized control approach should be computationally more strong to facilitate the whole system to process real-time data coming from multiple LCs (DER/DES/Load) [52]. Network security issues and network capacity are a concerning factor for such kind of frequent communication. Therefore, there is a single point of failure possibility of the centralized control approach. Since a microgrid is formed based on geographical conditions so its communication and associated components structure is changeable. For that reason, the centralized approach would not be an effective approach for future grids [40]. Figure 3.4 illustrates the centralized control of microgrid EMS.

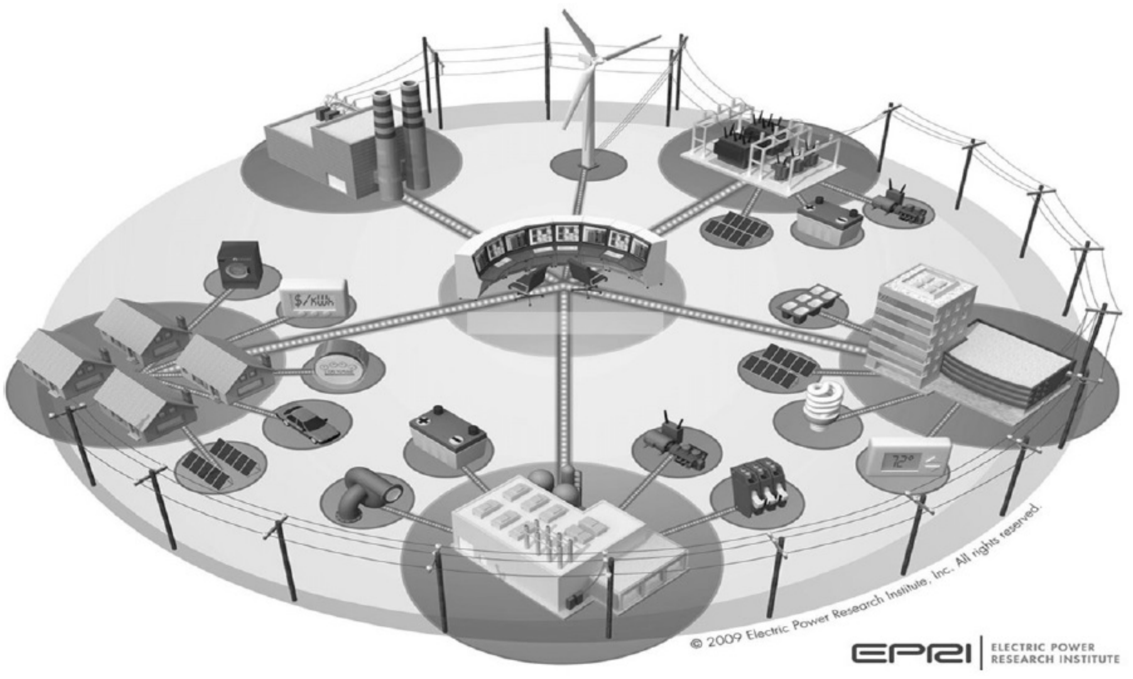

$/kWh
© 2009 Electric Power Research Institute, Inc. All rights reserved.
EPRI ELECTRIC POWER RESEARCH INSTITUTE

Figure 3.4: Centralized control of microgrid EMS [40]

Despite of some disadvantages of the centralized control of microgrid EMS, it is widely used controlling approach for power management because it is mature and well-developed approach [40]. The implementation and maintenance of centralized microgrid is not so difficult. It needs relatively low-operation cost. Only a single control unit is sufficient for entire power flow of the system which brings a remarkable benefit for the microgrid EMS.

Decentralized Control of EMS

Distributed decentralized control strategy is appropriate to maintain a complex microgrid system. Microgrid components like DER, DES and loads are controlled by one or multiple local controllers (LCs) and each of the local controller is responsible to make an operational decision by itself in contrast to the centralized local controllers of EMS. By using a communication network, each local controller communicates with another local controller that helps those local controllers to share their information with the neighbour controller within a minimum effort [52]. Therefore, local controllers do not need to wait for set points which are provided by MGCC in centralized microgrid EMS. In this decentralized distributed control approach, local controllers are more flexible to perform their tasks, and, in addition ensure parallel computation of components of the whole system. Figure 3.5 represents the decentralized control of EMS [52].

Distributed decentralized control of EMS is very effective in preventing the single point of failure issue in comparison to the centralized control. As the local controllers work independently, any disturbance in one of them does not badly effect the entire microgrid system [24, 40]. This approach potentially reduced the computational burden across different local controllers, and, for that reason, it is able to find a cost-effective solution.

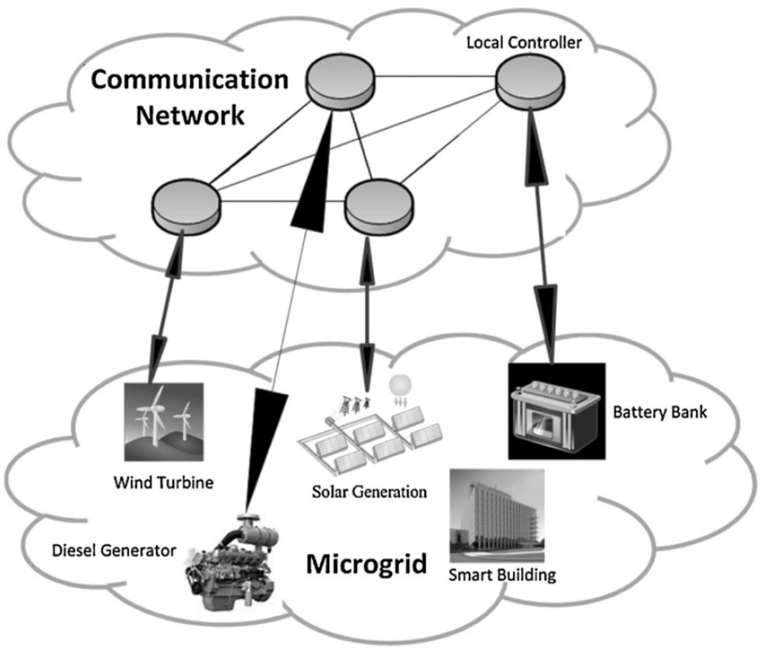


Figure 3.5: Decentralized control of EMS [40]

As it is mentioned earlier, the distributed decentralized approach is appropriate for a complex system but its implementation and maintenance is literally difficult. To upgrade an existing system, one needs to redesign the overall network topology as well as reconstruction of the system architecture, which obviously causes higher cost. Both cyber and physical security issues can create a reverse effect in normal operation of the microgrid system, because the local controllers are operated by themselves, which is an alarming factor to troubleshoot and trace the actual problem within a short time [52].

Table 3.3 shows the comparison between centralized and decentralized EMS.

Table 3.3: Comparison between centralized and decentralized EMS [24, 40, 52]

Centralized EMS	**Decentralized EMS**
Advantages of centralized EMS 1. Implementation is easy. 2. Maintenance is also easy. 3. Extensively used in microgrid EMS.	Advantages of decentralized EMS 1. Parallel computation. 2. Applicable for complex system. 3. Cost of computation is not high. 4. There is no single point of failure. 5. Expansion is easy.

Disadvantages of centralized EMS	Disadvantages of decentralized EMS
1. Single point of failure.	1. Communication network security issues.
2. Computational burden.	2. Synchronization is mostly required.
3. Expansion is hard.	3. Designing a new communication structure.
4. Plug and play is not so strong.	4. Upgrading existing system is costly.

3.2.2 Functionalities of Microgrid EMS

Functionalities always represent the capability of a system. There are a lot of functions used in a microgrid EMS. Type and number of functionalities vary depending on business logic of microgrid EMS. Although there is no fixed number of functionalities for a micrgrid EMS, there are four key functions to build any type of microgrid EMS that optimizes the microgrid operation while satisfying the technical requirements [28, 47, 54, 55]. The key functions of microgrid EMS are: forecasting, data analysis, optimization, and human-machine interface. Figure 3.6 represents the functions of microgrid energy management system [55].

Forecasting

In a micrgrid EMS, forecasting is obviously vital to predict the future condition of distributed energy resources, energy market and loads. Generally, to generate an effective forecast model, historical data and other data, such as weather, geographical location etc. are determined as prerequisite requirement [47]. As some renewable energy resources (wind turbine, photo voltaic) are variable power providers and some controllable loads (electric vehicle) are adjustable based on microgrid conditions, it is really hard to predict with 100% accuracy [54]. In optimization processes, forecast data also plays an important role. It is quite clear that the forecast data is so crucial to support the real-time demand response situation in microgrid EMS.

Data Analysis

A lot of data that an EMS has to handle comes from the energy market, loads (controllable and critical), distributed energy resources and storage systems. Efficient analysis of these collected data is important to understand the behavior of the microgrid components. Data analysis could be performed in two ways:

1. **Historical data analysis:** Historical data analysis helps to generate knowledge from historical data. This data analysis is helpful to predict the upcoming data.
2. **Forecast data analysis:** Forecast data analysis can be used as a verification techniques to examine whether the prediction is right or wrong.

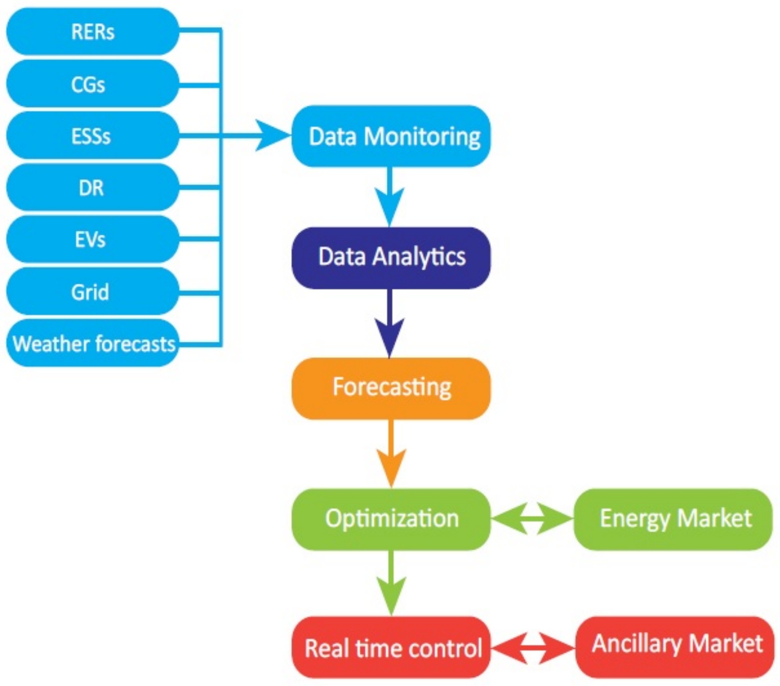


Figure 3.6: Functions of microgrid EMS [55]

Likewise forecasting, data analysis can also be used in optimization processes to enhance the performance of optimization [28].

Optimization

Optimization process is a key functionality of a micrgird EMS. It is used to determine a control decision by considering the energy generation, storage, controllable and critical loads to economically balance the power flow of the entire microgrid system. Optimization ensures the microgrid EMS installation ambition, i.e., minimum cost, maximum energy utilization etc. There are some common familiar methods, for instance, heuristic methods, dynamic programming, mixed-integer programming that are not able to successfully solve the economic dispatch optimization problem [54]. Recently, a couple of new optimization algorithms and methods are used to solve the optimization problem, such as ant colony optimization algorithms, artificial neural networks, multi-agent methods.

Human-Machine Interface (HMI)

Real-time data monitoring and controls are performed through human-machine interface in microgrid EMS [54]. HMI provides accessibility of the EMS to the system users based on their way of interaction. Human understandable information (graph, chart) are provided by HMI besides the usual raw data. This interface can also be used to protect privacy of the users in order to access the data of the system.

Saima Aman et al. define eight essential key requirements for energy management systems. These requirements are also applicable for any type of microgrid energy management system [5]:

1. Monitoring,
2. Dis-aggregation,
3. Availability and Accessibility,
4. Information Integration,
5. Affordability,
6. Control,
7. Cyber-security and Privacy,
8. Intelligence and Analytics.

3.2.3 Microgrid EMS with SCADA

Supervisory control and data acquisition (SCADA) has been used since 1980s in industries to control and monitor a plant or equipment [40]. SCADA systems are used in microgird EMSs for collecting and analyzing the real-time data of microgrid. It is a software system whose architecture is based on multiple hierarchical layers. The general architectural design layers are:

- **Web service** : It is mainly used to communicate between users and the existing system which plays a human-machine interface role. By using this layer, the inter-operability is ensured and heterogeneous users requests are easily maintained within minimum effort. This layer has different components, such as an open platform communication data access (OPC-DA) client, OPC-DA server, XML-DA server, Representational State Transfer (REST) gateway and web interface. The OPC is a standard created by an industrial automation task force, it is a non-propietary technical specification [4]. OPC-DA client communicates with OPC-DA serves and transmits information to the XML-DA server. XML-DA server receives REST requests and sends data through REST gateway. REST gateway works as a medium between the system and the web architecture [4]. The communication between XML-DA server and web is performed through the REST gateway using the hypertext transfer protocol (HTTP). HTTP supports some operations for communication in an inter-operable manner by utilizing GET, POST, PUT and DELETE. Figure 3.7 illustrates the general architecture of a SCADA system.
- **Central controller** : Central controller is the root unit of the microgrid EMS. It maintains two communication links between local controllers and OPA-DA serves. It gathers the overall information from the local controllers and provides on demand information the OPA-DA serves which are actually requested by web users.
- **Local controller** : In this layer, multiple local controllers exist. Each of local controllers is responsible for a particular microgrid component. Renewable energy generation controller (wind turbine, photo-voltaic), non-renewable energy generation controller (diesel, coal), energy storage controller (battery) and loads controllers (home, commercial) etc. are common instances of local controllers.

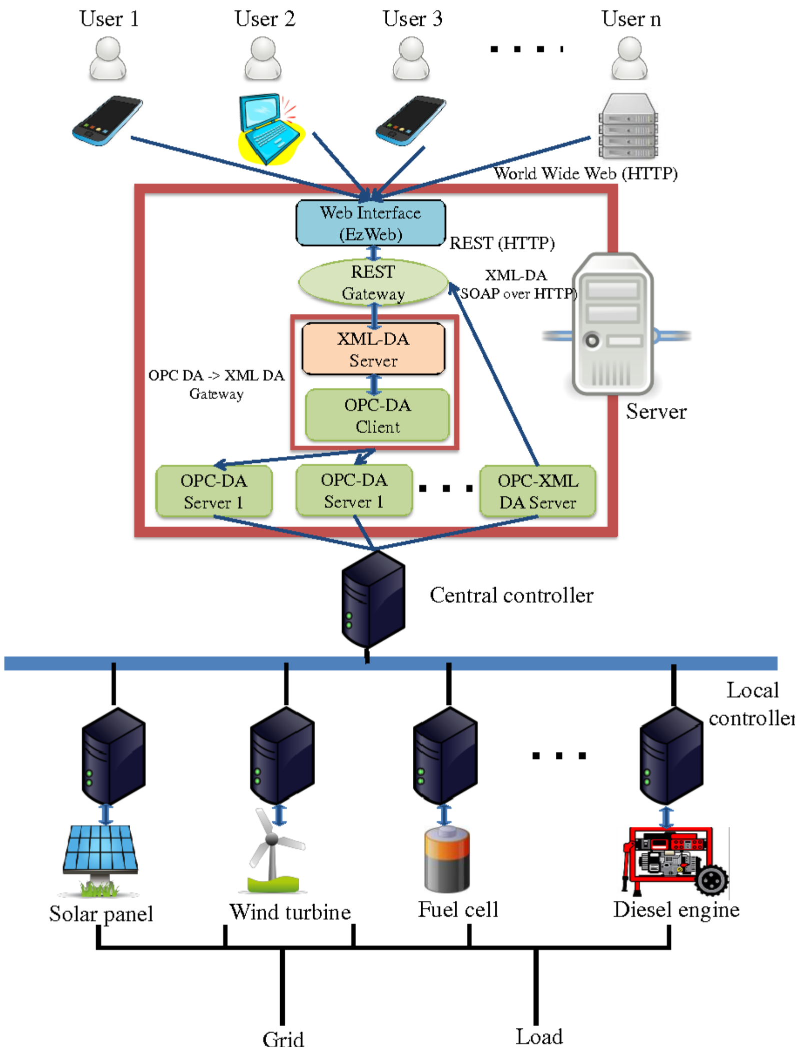


Figure 3.7: The general architecture of SCADA system [4, 40]

SCADA provides a more convenient way to configure and supervise microgrids than tranditional power management systems. It enables scalability by installing new component to the system and removing existing component from the system on demand. Such kind of modification does not affect the overall energy management of the microgrid. By using Web interface, end-users can start and stop any component of the system [4, 40]. SCADA system delivers real-time information of the entire microgrid to the users that helps to take decision to handle unexpected situations within very short time. As it is inter-operable, different types of users communicate spontaneously with the system without facing any operational problem. The web interface part of SCADA system is always operational and delivers real-time information.

3.2.4 Microgrid EMS Challenges

There are few design and implementation challenges in microgrid EMSs because of their complex structure in comparison to traditional EMSs. Minimizing the cost, maximizing the use of renewable energy, using the best optimization strategy, balancing variable energy supply and demand, reliable communication and cyber security are the most common challenges for microgrid EMS.

Operational Cost

Minimize the overall operational cost is one of the essential challenges for microgrid energy management systems [19]. Operational cost of a microgrid EMS extensively depends the on microgrid infrastructure and management strategies. Depending on demand, supply and storage, the choice of power which control strategy is appropriate like centralized or distributed and which mode of operation is needed to reduce the operational cost to appropriately manage the power of a microgird is really a considerable challenging factor.

Power Optimization

To manage economic power flow in a microgrid, a well-suited optimization technique is a crucial factor [54]. Based on market price, forecasting of supply and demand, it provides a best solution to optimally balance the power flow of the entire system. Generally, the prime goal of optimization techniques are to ensure the maximum usability. There are a lot of strategies and solution approaches, such as linear programming, dynamic programming, nonlinear programming etc. that are used for microgrid EMSs [46, 55]. Therefore, to identify an appropriate optimization strategy is important for micogrid EMSs.

Generation and Load Variation

From the production point of view, generally, the majority of renewable power generation components are injected in microgrid system, since those are totally dependent on the environmental situation and do not provide constant power to the microgrid that is a concerning fact to reliably manage energy for EMS. Similarly, from the consumer point of view, critical loads are more or less constant but non-critical i.e controllable loads can fluctuate in order to make sure the equilibrium level of energy. So, in any situation, to maintain the reliable power flow based on vast amount of renewable power resources and controllable loads is obviously a challenging term for microgrid EMS.

Reliable Communication

Reliable communication is also a challenging factor for microgrid EMSs in order to perform operation successfully. In centralized or decentralized control of microgrids, microgrid components communicate with controllers by using communication network. It needs to use a network architecture like local area network (LAN), wide area network (WAN), field area network (FAN), home area network (HAN) for reliable communication [52]. To perform a successful communication, there is a possibility to face some failures, such as network failures, time-out failures, resource failures. So, any kind of disruption of network communication can eventually create an inconsistency in the microgrid operation that finally generates an imbalance power flow of the entire microgrid system.

3.3 Service-Oriented Computing (SOC)

SOC is a computing paradigm that utilizes services as a core element for developing platform independent, inter-operable, easily composable and low-cost distributed applications [36, 37]. According to the concept of Service-Oriented Computing, Web services are the most promising technology [37]. Web services are self-contained, self-described and modular applications that can be published, invoked and located over the internet [27]. Web services use a standard communication medium and protocols over the network to ensure reliable support across heterogeneous environments. For developing a service model, SOC depends on the Service-Oriented Architecture (SOA) which is used to design a software system in a logical way [36, 37].

3.3.1 Service Design Principles

There are some service design principles of the SOA, such as service statelessness, service autonomy, service abstraction etc. The details information of the service design principles can be found in [14]. The service design principles are as follows:

- **Standardized service contract:** The purpose of standardized service contract is to provide an understandable description of the services. Standardized service contract is considered as the most fundamental design principle of the service-orientation architecture [14]. It provides a clear description about the services that helps the end-users to get an idea of inputs, outputs, and data types of the services.
- **Service statelessness:** The purpose of service statelessness design principle is to ensure that the server does not remember the previous state of interaction. In a client-server interaction paradigm, RESTful services use HTTP protocol as a communication medium.
- **Service abstraction:** Abstraction design principle hides the underlying details information of a service as much as possible [14]. It does not disclose the underlying logics of the services, it only expresses the basic information, such as methods, resources, and query strings of the services to the end-users/clients.
- **Service loose coupling:** Loose coupling design principle focuses on decreasing the dependency between service consumers and service implementation. The consumers of services are independent from the implementation of the services. If the consumers change something in their applications, that change does not create an effect on implementation of the services. On the other hand, any logical change of service implementation does not break the consumers applications.
- **Service autonomy:** Autonomy design principle ensures platform independence execution of services. A service can be executed in different environments, and providing generic solution without knowing the details of executed environments.

- **Reusability:** Reusability design principle ensures that a service can be used multiple times without reconstructing it to support various end-users. By using standard communication protocols, interfaces, and representation formats, service reusability can be achieved [17].

3.3.2 REST based Web Service

Web services can be divided into two types: REST based Web service, and SOAP/WSDL based Web service. Our aim is to implement the REST based Web service for our service-oriented microgrid EMS. In this section, we only describe the elements, methods, and design rules of the REST based Web services.

Representational State Transfer (REST) is an architectural style. It provides uniform interface for interaction and it is stateless i.e. there is no knowledge about previous interaction. RESTful Web services could be XML (Extensible Markup Language) or JSON (JavaScript Object Notation) based. Recently, most of the Web services use JSON based representation [49]. The aim of REST is to ensure the simplicity, scalability and data independency. This section describes the REST based Web service based on [3, 31].

Web Architecture Characteristics

The most noticeable Web architecture characteristics are the following:

- Statelessness,
- Generic interactions, such as GET, POST etc.,
- Data format is well known or standard (XML, HTML etc.),
- Message context is understandable,
- Human readable messages,
- Unified Resource Identifier (URI) is used for interaction.

Elements of REST

There are three main elements of REST. The elements are: resources, representations, and interaction.
Resources: URI is used to identify resources. The format of URI is given below:

```
URI = scheme "://" authority "/" path [ "?" query ] [ "#" fragment ]
Example: http://testmicro.com/temperature?date=2019-10-10
```

Representation: Standard or well-known representation of data, such as XML, JSON etc.

```
<energyresources>
  <energyresource>
    <name>photovoltaic</name>
    <amount>30</amount>
    <unit>kWh</unit>
  </energyresource>
</energyresources>
```

The same data can be represented in JSON as follows.

```
{
  "energyresources": {
    "energyresource": {
      "name": "photovoltaic",
      "amount": "30",
      "unit": "kWh"
    }
  }
}
```

Interaction: The protocols, which support the URIs and representations for interaction. Protocols could be Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP) etc.

Generic Methods of REST

GET: GET method is used for retrieval. It is not a common practice to use GET method for update. GET is idempotent and safe. HTTP status code 200 means server successfully process the request. The Table 3.4 describes the categories of response status code.

```
# Request
GET /DES/20191010 HTTP/1.1
Host: www.testmicro.com
```

```
# Response
HTTP/1.1 200 OK
Content-Type: application/json
{
    "DES": {
        "energyresources": {
            "energyresource": [
                {
                    "name": "photovoltaic",
                    "amount": "30",
                    "unit": "kWh"
                },
                {
                    "name": "wind turbine",
                    "amount": "20",
                    "unit": "kWh"
                }
            ]
        },
        "date": "2019-10-10",
        "time": "20"
    }
}
```

Table 3.4: Categories of response status [31]

Status code	Label	Description
1XX	Informational	Continuing process, Request received

2XX	Success	Request was successfully accepted
3XX	Redirection	Additional action must be taken in order to complete the request
4XX	Client Error	Request is associated with bad syntax
5XX	Server Error	Server is not able to process the request

POST: POST method is used to create a resource. It is also used to update a resource. POST is not safe and idempotent. HTTP response status code is important for POST request. If POST is used for creating a resource then the status code will be 201 (created) and for updating a resource, it could be 200 (OK) or 204 (No Content).

```
# Request
POST /users HTTP/1.1
Host: www.testmicro.com
Content-Type: application/json;charset=UTF-8
{
    "user": {
        "firstname": "John",
        "lastname": "Smith",
        "email": "john.smith@testmicro.com",
        "gender": "male"
    }
}
```

```
# Response
HTTP/1.1 201 Created
Location: http://www.testmicro.com/users/1
```

PUT: PUT method is used to update an existing resource. It is also used to create a resource. Generally, updating an existing resource is common practice instead of creating a resource. POST is not safe and but idempotent. HTTP response status code is important for PUT request. If PUT is used for creating a resource then the status code will be 201 (created) and for updating a resource, it could be 200 (OK) or 204 (No Content).

```
# Request
PUT /users/1 HTTP/1.1
Host: www.testmicro.com
Content-Type: application/json;charset=UTF-8
{
    "user": {
        "firstname": "John",
        "lastname": "Smith",
        "email": "john.smith@testmicro.com",
        "gender": "male"
    }
}
```

```
# Response
HTTP/1.1 200 OK
```

DELETE: DELETE method is used to remove an existing resource. It is not safe and but idempotent. HTTP response status code is important for a DELETE request. If a resource is successfully deleted then the status code will be 200 (OK) or 202 (Accepted), which means deletion is being processed, the requested resource(s) is not fully remove the requested resource(s). The response status code could be 204 (No Content) if there is nothing at the URI.

```
# Request
DELETE /users/1 HTTP/1.1
Host: www.testmicro.com
```

```
# Response
HTTP/1.1 200 OK
```

Table 3.5 represents the basic differences of generic methods of REST.

Table 3.5: Generic methods of REST [3, 31]

GET	POST	PUT	DELETE
Retrieve resource	Create resource	Update resource	Remove resource
Safe	Unsafe	Unsafe	Unsafe
Idempotent	Not idempotent	Idempotent	Idempotent
Response status code 200	Response status code 201	Response status code 200 or 204	Response status code 200 or 202

REST API Design Rules

There are many rules to design RESTful Web service. In this sub-section, we describe the most important ones in accordance with [31].
URIs design rules:

- To specify a hierarchical relationship, forward slash (/) must be used.
- Lowercase letters should be preferred in URI paths.
- To enhance readability, Hyphens (-) should be used in URIs.
- Underscore (_) should not be used in URIs.
- Singular nouns should be used to specify atomic resources and plural nouns should be used to indicate collection of resources.

Interaction design rules:

- GET must be used to retrieve a representation.
- PUT must be used to insert and update.
- POST must be used to create a new resource.
- HEAD should be use to retrieve response headers.
- DELETE must be used to remove resource(s).

Representation design rules:

- Representation should be XML, JSON or others based on client demand.
- XML, JSON or others must be properly nested.
- Error responses should include meaningful necessary information such as date and time, response status code (404), short description, possible solution links (if available).

3.3.3 RESTful Service Description Language (RSDL)

The RSDL is an XML vocabulary which is used to describe the RESTful Web services. The RSDL describes the methods, data types, media types, resources etc. for designing and documenting a RESTful Web service. Usually, a RSDL document consists of eight different items for documenting a Web service. The items are as follows [23]:

1. Media types,
2. Resources,
3. Links,
4. Methods,
5. HTTP headers,
6. Authentication,
7. URI parameters and templates,
8. HTTP status codes.

An RSDL description follows a structure to describe the Web services. “Service” is the root element of the RSDL documents. The service element structure in Relax-NG Compact syntax is as follows:

```
start = service
service =
    element service {
        id?,
        name,
        identity-provider-ref?,
        documentation?,
        service-start,
        media-types,
        resources,
        link-relations?,
        headers?,
        authentication?,
        status-codes?,
        uri-parameters?
    }

service-start = element start { idref }
idref = attribute ref { xsd:IDREF }
```

```
    identity-provider-ref = attribute identity-provider-ref { xsd:IDREF }
```

Another element is “media types” which is used to describe the representations of the resources. A resource representations could be in JSON, XML, text, HTML or other formats. The media types element structure is as follows:

```
    media-types = element media-types { documentation?, media-type* }
    media-type = element media-type { id?, name, documentation?, description* }
    media-type-ref = attribute media-type-ref { xsd:IDREF }

    description = element description { type, href, documentation? }
    type = attribute type { "rnc" | "rng" | "xsd" | "JSONSchema" | "sedola" | "text" | "html" }
    href = attribute href { xsd:anyURI }
```

We mention only two fragments of structure as an example from the whole structure of the RSDL description. The full description can be found in [23].

4 Design of Microgrid EMS

This chapter describes our proposal for the design of a microgrid EMS. A service-oriented microgrid EMS has to ensure interoperability of the system that helps heterogeneous users and systems to communicate with the system without knowing of location or platform.[1] Services are considered as a key component for designing a service-oriented system [36]. We divide this chapter into three parts, such as a conceptual model, service design principles, and microgrid Web services. The conceptual model represents a general structure of the microgrid EMS. In the service design principles part, we discuss the application of the design principles on the capabilities of the microgrid EMS. In the microgrid Web services part, we design some Web services for the microgrid EMS including interfaces, request, response and accessible methods of the Web services.

4.1 Conceptual Model of EMS

In this section, we discuss our conceptual model of a microgrid EMS. Our conceptual model consists of four parts based on the basic functionalities of microgrid EMSs (Chapter 3, Section 3.2.2). The parts are Web Interface, Data Collection, Optimization, and Forecast & Data analysis. To construct this model, we follow the architectural concept of the SCADA system because it enables on demand scalability by adding new component to the system and removing existing component from the system without affecting the overall energy management of the microgrids. The detail description of the SCADA system is in Chapter 3 under Section 3.2.3. Figure 4.1 represents the conceptual model of the microgrid EMS.

4.1.1 Data Collection

Data Collection is responsible for the collection of two types of data for the EMS. Internal data and external data are the main data sources of the EMS to make the system operational. The loads data, DERs data and DESs data are the internal data sources of the EMS. To collect internal data from microgrid devices (wind turbine, photovoltaic, battery etc.) via a central controller, which has been described in Chapter 3 under Section 3.2.3, Message Queuing Telemetry Transport (MQTT), IEC 61850 or other proprietary communication channels can be used [49]. The internal data will be collected through the HTTP communication protocol because, the RESTful Web services (Section 4.3) use HTTP as a communication protocol which is a global standard and highly interoperable [32]. The external data are collected from different external service providers via a HTTP communication channel. Weather data and real-time market price data are an example of external data of the EMS.

[1] https://www.ibm.com/developerworks/library/ws-inter/index.html

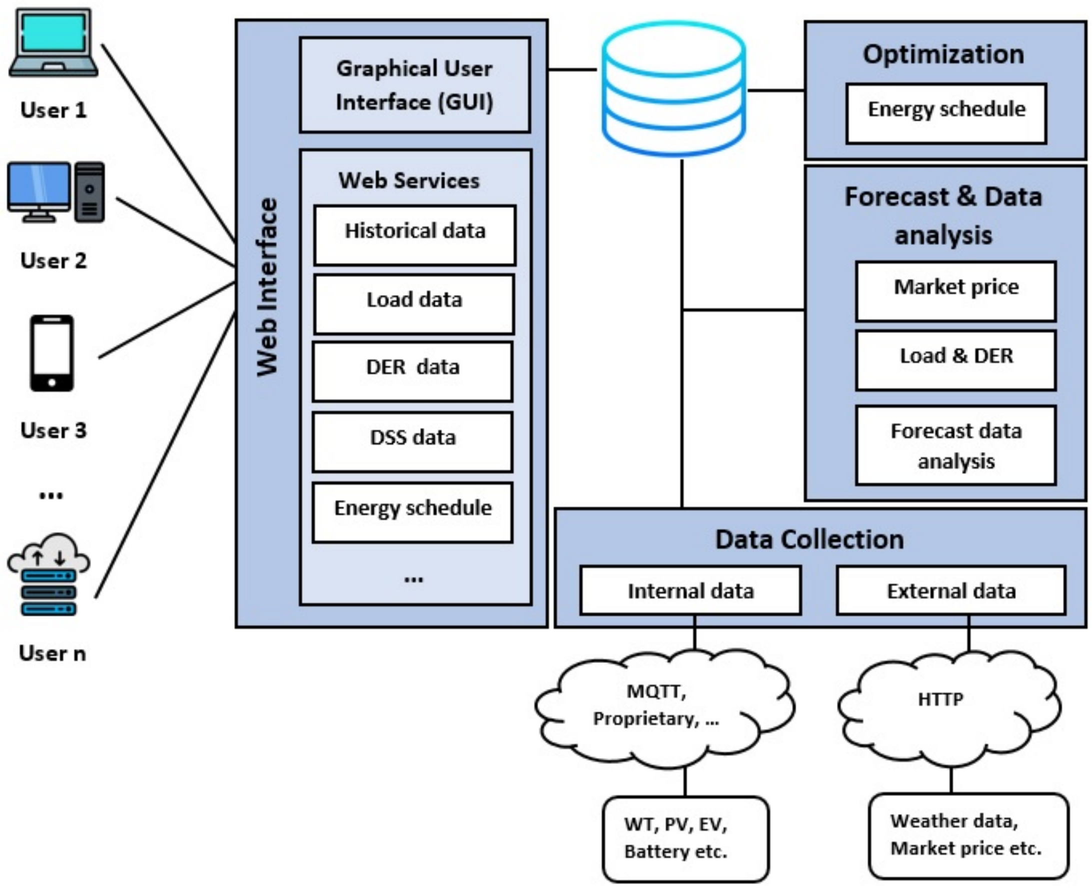


Figure 4.1: Conceptual model of microgrid EMS

4.1.2 Web Interface

Web Interface is responsible for the communication with the external heterogeneous end-users and systems, such as commercial users and household users. The Web interface part is divided into two sub-parts, namely Graphical User Interface (GUI), and Web services. The main goal of the Web interface is to facilitate the users by visualizing and collecting real-time data of the EMS. The GUI sub-part of the Web interface provides data visualization, monitoring and control facility to the end-users of the system, and the Web services sub-part of the Web interface delivers a real-time data, analysed data, historical data, and other types of data of the EMS to the end-users. The Web services sub-part is accessible via different generic methods, such as GET, POST, PUT, and DELETE which have been discussed in Chapter 3 under Section 3.3.2.

4.1.3 Optimization

Optimization is responsible for providing an optimal energy schedule of the EMS. The provided energy schedule delivers a cost effective solution for our EMS that ensures the minimum cost of energy, and the maximum profit of the microgrid EMS. To formulate an energy schedule model as well as to get the optimal solution, we use an Energy Schedule Web service. The Energy Schedule Web service requires some user inputs, such as the loads data, DERs data, DESs data etc. This service is accessible via a GET method, and it returns JSON data to the requesters. The detail description of the optimal Energy Schedule Web service is described in Section 4.3.

4.1.4 Forecast & Data Analysis

Forecast & Data Analysis is responsible for the prediction of the upcoming data of our EMS that could be used as an input for the optimization process as well as analyzing the predicted data that could help to make a control decision. Usually, a day-ahead market prices data are collected from the third party service providers, and the DERs forecasted energy data are collected by simulating or connecting to real energy resources (wind turbine, photovoltaic etc.) based on weather data like wind speed, temperature, solar irradiation etc. After collecting the forecasted data, the data analysis operation is performed to make a control decision for the DERs. This part of the EMS is not only responsible for analysing the forecasted data, but other data, such as the loads data, historical data could also be analysed in order to monitor the overall EMS.

4.2 Design Principles for Service Orientation

The service design principles of SOA help to address the non-functional challenges of a microgrid EMS. Generally, the capabilities of a traditional microgrid system i.e. a legacy software system are not reusable, independent and loosely coupled. Such a system is commonly developed for a specific executing environment as well as its capabilities are not scalable and interoperable. The capabilities of a microgrid EMS are designed as a small piece of software programs that are introduced as services in SOC [17, 38, 49].

With the design principle of loose coupling, adding or modifying capabilities do not create an adverse effect on the entire microgrid EMS . Microgrid EMS capabilities, such as DERs, loads are represented as services and each of the services can be used to add any number of resources or modify existing resources. For example, by using a DER service, we can add multiple renewable energy resources such as wind turbine, photovoltaic, biomass (Section 4.3), and we can also modify an existing resource by changing the parameter value of the interface of the DER service that has no impact on the stability of the entire microgrid EMS.

The SOA autonomy design principle ensures platform-independent execution of services (Chapter 3, Section 3.3.1). Microgrid services can be executed in different environments without knowing the details of the underlying executed environments. For example, the (controllable) load service is used to calculate the energy consumption of household appliances, such as a water heater or cloth washer. If the load service is executed on a Linux operating system environment and Windows operating system environment, the output of the service is same for the Linux environment or the Windows environment.

The design principle of reusability ensures services that can be achieved by using a standardized service communication via standard interfaces, protocols and data formats [17, 37]. The microgrid services, such as DER service, DES service, load service communicate through the HTTP communication protocol and using JSON based messages. Therefore, the capabilities of the microgrid EMS are reusable for the heterogeneous end-users and systems. As the microgrid EMS capabilities are being reusable, they are used multiple times that save valuable time to rebuild them.

The SOA abstraction design principle hides the underlying implementation detail of a service as much as possible [14]. Each of the microgrid EMS services has one or more interfaces, and each of the interfaces represents a specific functionality that requires some essential parameters. The interfaces of the microgrid services describe what they do, but do not disclose how they do it. For example, the read wind turbine interface of the DER service requires some parameters, such as windSpeed, bladeLength etc. (Section 4.3) and provides the generated power of the wind turbine. Therefore, we can understand the requirements of a wind turbine for producing the power, but the underlying details information i.e. the calculation of a wind turbine is not disclosed.

4.3 Microgrid Web Services for the EMS

Web services help to develop a platform independent software system [14]. By using the microgrid services, an EMS could be different from another EMS depending on their own business logic and implementation. Microgrid services are really helpful for rapid development of a microgrid EMS. In this section, we design some Application Programming Interfaces (APIs) of the microgrid services. In the existing studies of microgrid EMSs, such as [28, 40, 49, 52], one can notice that microgrid EMSs have commonalities, such as energy resources, loads, storages etc. Depending on such commonalities, we design five categories of Web services. The services are: DER service, DES service, Load service, Energy Schedule service and Data service. Each category of Web services has some interfaces, and each of the interfaces has an accessible method, resource and a brief description.

4.3.1 DER Service

Different types of renewable energy resources are used in microgrid EMSs. These resources generate variable amount of power depending of the weather condition. Most of the microgrid EMSs consider two types of renewable resource, such as wind turbine and photovoltaic [15, 28, 34, 43, 49, 54]. We design two interfaces for these two renewable energy resources of the DER service. The interfaces are: read wind turbine, and read photovoltaic. We also design an interface for another potential renewable energy resource which is biomass. Read biomass is the name interface of the resource.

Interfaces of the DER service

To get produced energy of a wind turbine through the read wind turbine interface, some essential parameters are required [22, 34, 45, 51]. Each of the parameters is used to calculate the generated energy of a wind turbine in back-end. A wind turbine has three main mechanical parts, such as rotor, gear box, and generator [45]. Each of the mechanical parts has different level of efficiency. The parameters of the read wind turbine interface are:

- windSpeed: The wind speed value (metres per second) of the environment.

- bladeLength: The blade length (meters) of a wind turbine which is used to calculate the swept rotor area of the wind turbine [22, 45].
- airDensity: The air density at sea-level (kilograms per cubic metre). The air density can be calculated by using temperature, humidity and pressure [51].
- rotorEffici: The rotor efficiency of a wind turbine (%).
- gearBoxEffici: The gear box bearing efficiency of a wind turbine (%).
- genEffici: The generator efficiency of a wind turbine (%).

Six necessary parameters are requied by the read photovoltaic interface for calculating the produced energy of a photovoltaic [7, 9]. These parameters provide some numerical values to calculate the produced energy. The parameters of the read photovoltaic interface are:

- panelArea: The total area (width and height) of a solar panel.
- panelYield: The efficiency of a solar panel.
- panelAngle: A solar panel is installed in a specific angle, such as 25 degrees, 30 degrees.
- panelLatitude: A specific latitude (degree) of a solar panel.
- solarHorizon: It is calculated based on the solar radiation on a horizontal surface [9].
- temperature: The current temperature (Celsius) of the environment.

To calculate the produced energy of a biomass plant, four parameters are required [10]. The parameters are as follows:

- bioQty: The biomass quantity (cubic metres per hour).
- bioVolu: The biomass volumetric mass (kilograms per cubic metre).
- bioLHV: The lower heating value of the biomass (megajoules per kilogram).
- plantEffici: The biomass plant efficiency (%).

The required data types of these parameters are discussed in the RSDL document for the DER service in Appendix A. Table 4.1 represents the accessible methods, resources, and descriptions of the interfaces.

Table 4.1: Interfaces of the DER service

Interface	**Method**	**Resource**	**Description**
Read wind turbine	GET	`(Domain-Name)/windturbine? windSpeed&bladeLength &airDensity&rotorEffici &gearBoxEffici&genEffici`	Provide the hourly generated power of the wind turbine based on the given parameters.

Read photovoltaic	GET	`(Domain-Name)/photovoltaic?panelArea&panelYield&panelAngle&panelLatitude&solarHorizon&temperature`	Provide the hourly generated power of the photovoltaic based on the given parameters.
Read biomass	GET	`(Domain-Name)/biomass?bioQty&bioVolu&bioLHV&plantEffici`	Provide the hourly generated power of the biomass based on the given parameters.

4.3.2 DES Service

The DES service is responsible to provide energy storage service for the microgrid EMSs. According to the studies of energy storage system, we find that battery energy storage systems are very common [15, 33, 42, 53]. For storing renewable energy, the battery energy storage system is the most preferable choice due to its continual improvement and reliability [33]. It ensures high accuracy with minimal computational effort [42]. Currently, we design three interfaces of the DES service. The interfaces are: read battery, charge battery, and discharge battery.

Interfaces of the DES service

The read battery interface requires some necessary parameters for constructing a battery energy storage model [15, 42, 53]. The parameters of the read battery interface are as follows:

- maxCapacity: The maximum capacity of a battery storage system (kW).
- minCapacity: The minimum capacity of a battery storage system (kW).
- charEffici: The charging efficiency of a battery storage system (%).
- discharEffici: The discharging efficiency of a battery storage system (%).

The charge battery and discharge battery interfaces require one parameter which is called power. This parameter is used to store/release a specific amount of power (kW) in/from the battery storage system. The above mentioned interfaces have different types of accessible methods, and their resource structures are also different. Table 4.2 represents the accessible methods, resources, and descriptions of the interfaces.

Table 4.2: Interfaces of the DES service

Interface	Method	Resource	Description
Read battery	GET	`(Domain-Name)/battery?maxCapacity&minCapacity&charEffici&discharEffici`	Provide a unique non negative integer number as an instance of a battery storage system.

Charge battery	POST	`(Domain-Name)/battery/{id}/charge?power`	A certain amount of power is stored in the battery storage system.
Discharge battery	POST	`(Domain-Name)/battery/{id}/discharge?power`	A certain amount of power is released from the battery storage system.

In Appendix B, the RSDL document for the DES service describes the datatypes of these parameters of the interfaces.

4.3.3 Load Service

Two types of loads, such as controllable and uncontrollable loads are considered as demand of a microgrid EMS. In this section, we design the controllable load service of the microgrid EMS. The controllable load service can provide various kinds of services, such as water heater service, cloth dryer service, cloth washer service, dish washer service, printer service etc.

Interfaces of the controllable load

As there can be many types of controllable loads, we choose to design interfaces of three such types as a proof of concept. The controllable load service interfaces are: read electric water heater, read electric cloth dryer, and read electric cloth washer. The read water heater interface requires four essential parameters to calculate the energy consumption of a water heater [11, 41]. The parameters of the read electric water heater interface are:

- volume: The capacity of a water heater (liters).
- hotTemp: The desired temperature of water (Celsius).
- coldTemp: The initial temperature of water (Celsius).
- PR: The efficiency or performance ratio of an electric water heater is in the range of 85-94% [41].

The read electric cloth washer interface is used to calculate an electric cloth washer energy consumption. The interface needs some vital parameters as follows [11, 12, 26]:

- laundryWeight: The total laundry weight (kg).
- washerCapacity: The capacity of a cloth washer (kg).
- duration: The estimated duration of the main wash (minute).
- temperature: The washing temperature of cloth washer (Celsius).

The energy consumption of an electric cloth dryer is calculated based on some necessary parameters. The read electric cloth dryer interface requires four parameters as an user input [6, 11, 12, 41]. The parameters are as follows:

- initialWeight: Water weight before drying (lbs).
- finalWeight: Water weight after drying (lbs).
- duration: Cloth dryer drying capacity (minute).
- EF: The Energy Factor of a cloth dryer. The minimum EF of an electric cloth dryer is 3.01 [6, 11].

The datatypes of these parameters of the interfaces are described in a RSDL document in Appendix C. Table 4.3 represents the accessible methods, resources, and descriptions of the interfaces.

Table 4.3: Interfaces of the controllable load service

Interface	Method	Resource	Description
Read electric water heater	GET	`(Domain-Name)/load/waterheater?volume&hotTemp&coldTemp&PR`	Provide the energy consumption data of the water heater.
Read electric cloth washer	GET	`(Domain-Name)/load/clothwasher?laundryWeight&washerCapacity&duration&temperature`	Provide the energy consumption data of the electric cloth washer.
Read electric cloth dryer	GET	`(Domain-Name)/load/clothdryer?initialWeight&finalWeight&dryerCap&EF`	Provide the energy consumption data of the electric cloth dryer.

4.3.4 Energy Schedule Service

Energy schedule service provides an optimal solution in the form of a schedule that balances the demand and supply of energy of the microgrid EMS. The main objective of the energy schedule is to minimise the costs of energy as well as to maximise the revenue of the microgrid. This service has only one interface which is read optimal energy schedule.

Interfaces of the energy schedule service

The above mentioned interface needs nine essential parameters for calculating the optimal energy schedule of the microgrid EMS [15, 28, 47, 53]. Some parameters, such as maxGeneration, minGeneration, maxLoad etc. are used as constrains of the energy schedule model. For example, the hourly power generation of the DERs must be in between minGeneration and maxGeneration. Some parameters, such as generatedEnergy, loads, and energyStorage are used to balance the demand and supply of energy of the microgrid EMS. The parameters of the read optimal energy schedule are as follows:

- maxGeneration: The maximum hourly power generation of the DERs (kW).
- minGeneration: The minimum hourly power generation of the DERs (kW).
- generatedEnergy: A list of 24 hours power generation. The list has 24 (0-23) entries that indicate the amount of power (kW) is generated by the DERs in each hour.
- maxLoad: The maximum hourly demand of power of the consumers (kW).
- minLoad: The minimum hourly demand of power of the consumers (kW).
- loads: A list of 24 hours demand of power. The list has 24 (0-23) entries that indicates the amount of power (kW) is required by the consumers in each hour.
- energyStorage: A list that contains one or more battery(ies). Each of the battery has the following properties which are efficiency (%), maximum limit of charge and discharge (kW), maximum limit of stored energy (kW), and charging/discharging rate (%).
- appliances: A list that contains one or more appliance(s), such as dish washer, cloth washer. Each of the appliances has the following properties which are earliest start time (hour), latest end time (hour), power (kW), and length of operation time (minutes).
- marketPrices: A list that contains the 24 hours market prices of energy which is collected from third party service providers.

In Appendix D, the RSDL document for the Energy schedule service describes the datatypes of the parameters. Table 4.4 represents the accessible method, resource, and description of the interface.

Table 4.4: Interfaces of the energy schedule service

Interface	Method	Resource	Description
Read optimal energy schedule	GET	`(Domain-Name)/ energyschedule?maxGenerati-on&minGeneration&generated-Energy&maxLoad&minLoad&lo-ads&energyStorage&applianc-es&marketPrices`	Provide an optimal energy schedule of the microgrid EMS.

4.3.5 Data Service

Microgrid EMS data service delivers the microgrid data to the external users of the EMS as well as to the GUI component of the EMS. Currently, we design to provide four types of data services of the microgrid EMS. The different types of data services are as follows:

1. **Real-time data service:** This service provides the current state of loads, generations and storages of energy of the EMS instead of the database storage data. For transferring huge amount of data, it potentially provides more benefits than publish-subscribe mechanism [32].
2. **Historical data service:** This service provides data from the database of the microgrid EMS. This service is also responsible to provide energy resources data, loads data, and energy storages data based on the given time interval.

3. **External data service:** This service provides environment data (temperature, wind speed etc.), market price data, and other data collected from the external sources.
4. **Energy scheduling data service:** This service provides daily basis an optimal energy schedule data of the EMS. It is also possible to get the previous energy schedule data depending on the given time interval.

Interfaces of the microgrid EMS data service

The above mentioned Web services have multiple interfaces. Some interfaces have more than one parameter. Table 4.5 represents the accessible methods, resources, and descriptions of the interfaces.

Table 4.5: Interfaces of the data service

Interface	Method	Resource	Description
Read current data of wind turbines	GET	`(Domain-Name)/data/WT /current`	Provide the current data of the wind turbines of the EMS.
Read current data of photovoltaics	GET	`(Domain-Name)/data/PV /current`	Provide the current data of the photovoltaics of the EMS.
Read current data of storage system	GET	`(Domain-Name)/data/BT /current`	Provide the current data of the batteries of the EMS.
Read periodic data of wind turbines	GET	`(Domain-Name)/data/WT /periodic?startDate&endDate`	Provide the historical data of the wind turbines of the EMS.
Read periodic data of photovoltaics	GET	`(Domain-Name)/data/PV /periodic?startDate&endDate`	Provide the historical data of the photovoltaics of the EMS.
Read periodic data of storage system	GET	`(Domain-Name)/data/BT /periodic?startDate&endDate`	Provide the historical data of the batteries of the EMS.
Read hourly day-ahead price	GET	`(Domain-Name)/data/price /hourly?hour`	Provide a hourly day-ahead market price of the EMS.
Read daily day-ahead price	GET	`(Domain-Name)/data/price /daily`	Provide a list of day-ahead market prices of the EMS.
Read current temperature	GET	`(Domain-Name)/data /temperature/current`	Provide the current temperature data of the DERs of the EMS.

Read current wind speed	GET	`(Domain-Name)/data /windspeed/current`	Provide the current wind speed data of the DERs of the EMS.
Read periodic temperature	GET	`(Domain-Name)/data /temperature/periodic? startDate&endDate`	Provide the historical temperature data of the DERs of the EMS.
Read periodic wind speed	GET	`(Domain-Name)/data /windspeed/periodic? startDate&endDate`	Provide the historical wind speed data of the DERs of the EMS.
Read current energy schedule	GET	`(Domain-Name)/data /energyschedule/current`	Provide the current optimal energy schedule data of the EMS.
Read periodic energy schedule	GET	`(Domain-Name)/data /energyschedule/periodic? startDate&endDate`	Provide the historical energy schedule data of the EMS.
Read current peak time	GET	`(Domain-Name)/data/peaktime /current`	Provide the pick time of the current year by analyzing the forecasted data.
Read periodic peak time	GET	`(Domain-Name)/data/peaktime /periodic?year`	Provide the historical pick time for the requested year.

The datatypes and a brief description of the parameters of the interfaces are described in the RSDL document for the Data service in Appendix E.

5 Implementation

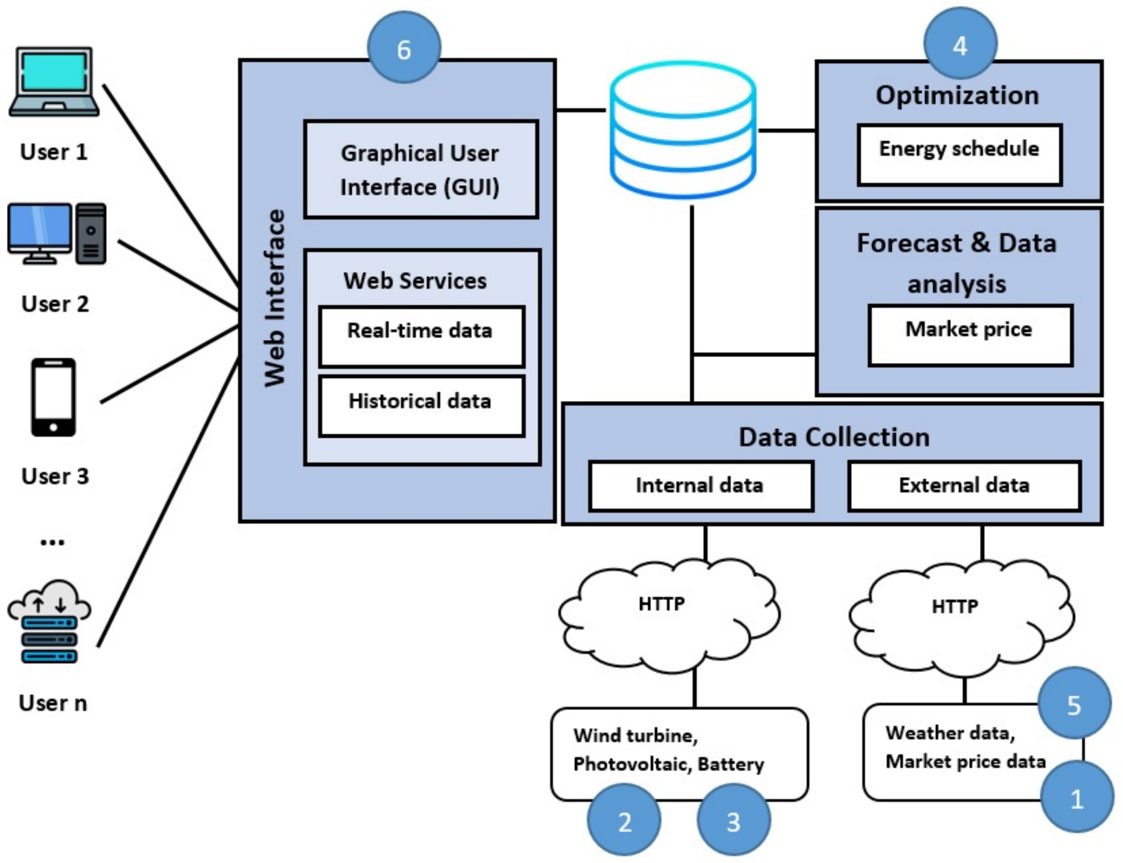
In this chapter, we describe the implementation of our EMS. We organize this chapter into three parts, such as model of EMS, data collection & implementation, and data visualization. The model of EMS part presents a graphical model of our microgrid EMS as well as a service-oriented structure of the EMS. The data collection & implementation part describes the process of data collection and implementation of our Web services. The data visualization part presents some graphical figures, such as wind turbine energy generation, photovoltaic energy generation as a proof of implementation of the EMS.

5.1 Model of EMS

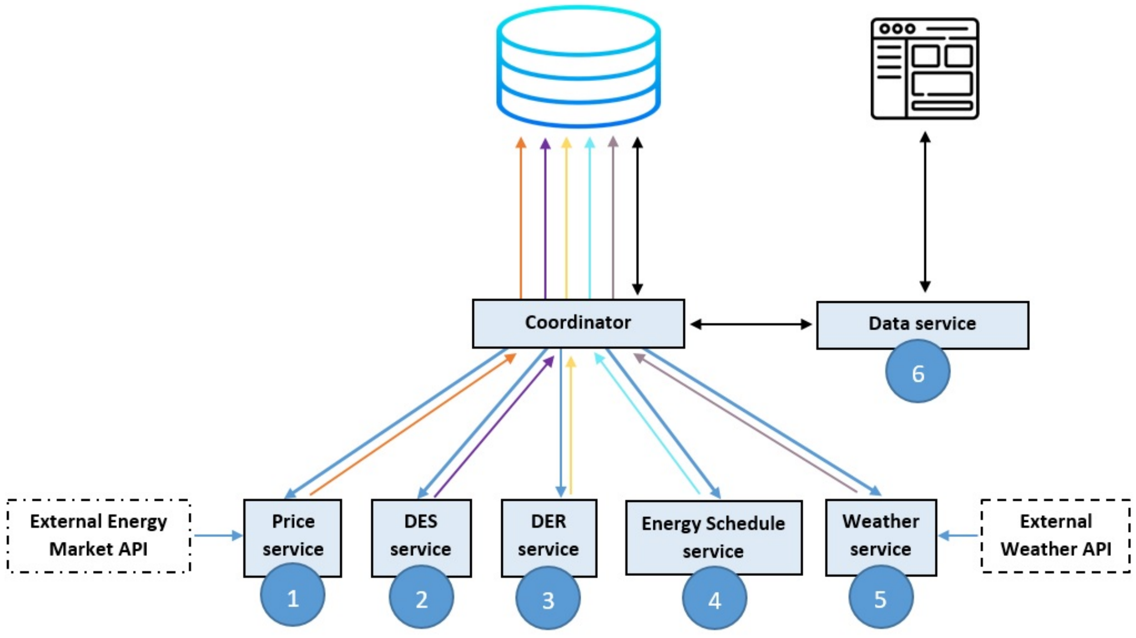
As a proof-of-concept implementation, we consider a microgrid EMS with a wind turbine, a photovoltaic, and a battery. Figure 5.1a represents the concrete microgrid EMS. The EMS consists of four basic functionalities as follows:

- **Data collection:** Two types of data are required for the EMS, such as internal data and external data. We collect internal data (wind turbine, photovoltaic, battery) by using our designed DER service and DES service (Chapter 4, Section 4.3) via HTTP communication channel. We collect external data (weather, market price) from the third party service provider via HTTP communication channel.
- **Web interface:** Graphical User Interface (GUI) and Web service are the sub-parts of the Web interface. The GUI represents the graphical view of energy generation of wind turbine, photovoltaic etc. for the end-users. The Web service provides current and historical energy generation and storage data of the EMS to the end-users of the system.
- **Forecast & Data analysis:** By analyzing the real-time market prices and hourly supply & demand of the EMS, the amount of profit or loss of the EMS is determined in each hour.
- **Optimization:** Provides the optimal energy schedule of the EMS by using the Energy Schedule service (Chapter 4, Section 4.3).

Figure 5.1b represents the service-oriented structure of the EMS. In that figure, one can notice that we have a central logical component this is called Coordinator. The Coordinator calls various services, such as DER service, DES service, Weather service, and it receives various types of data from those respective services, after that it stores those data to the central database of the EMS. To collect hourly energy generation data of the photovoltaic and wind turbine, the Coordinator calls the DER service each hour. The Coordinator provides the EMS data through the data service to the GUI component to visualize the EMS data. The end-users of the EMS can also collect the current and historical raw data (wind turbine, photovoltaic, battery) from the Coordinator via the Data service of the EMS.



(a) A simple micorgrid EMS



(b) Service-oriented structure of the EMS

Figure 5.1: A microgrid EMS and its service-oriented structure

5.2 Data Collection & Implementation

The simulations of wind turbine, photovoltaic, battery, and energy schedule are already implemented in an existing EMS (https://gitlab.svccomp.de) which is developed by the students of the Service Computing Department at IAAS at University of Stuttgart. We use the implementation of an existing EMS to collect the energy generation data, storage data, and energy schedule data. For implementing our services, we add few lines of code to the existing implementation, because our designed Uniform Resource Identifier (URI) structure and parameters do not match with the parameters of the existing system. For example, we use some parameters, such as `solarHorizon`, `panelAngle`, `panelArea` in our read photovoltaic interface but the existing system which simulates a photovoltaic that uses `S_horizontal`, `angleOfModule`, `area` as the name of parameters respectively. We add four route elements/blocks to the existing `frontend_service` file, and call the associated functions with our defined parameters that simulates the wind turbine, photovoltaic, battery, and optimal energy schedule. The `frontend_service` file contains all route elements/blocks that are used to receive HTTP GET, POST requests and send the required data to the requesters. The following lines of codes are an example of such modification of the existing implementation of photovoltaic.

```
@app.route('/photovoltaic', methods=['GET'])
power = solar.computePower(req_data['temperature'], req_data['solarHorizon'],
...)
return power.toJson()

instead of

@app.route('/solarpanel/create', methods=['POST'])
session = entityDB.create_session()
solarpanal = entityModels.SolarPanel(req_data['area'], req_data['angleOfModule']
,...)
entityDB.addElementToDatabase(session, solarpanal)
```

From the above lines of codes one can notice that we use GET method instead of POST method. According to the interaction design rules (Chapter 3, Section 3.3.2), POST method is used to create a resource and GET method is used to retrieve a resource. The existing implementation uses POST method to create resource (logically) and stores data to the database. We use GET method because we do not need to store data, we need only simulated data (wind turbine, photovoltaic) of the existing implementation, and sends back these data to our system. The modification of the existing implementation of wind turbine is as follows:

```
@app.route('/windturbine', methods=['GET'])
power = wind.computePower(req_data['airDensity'], req_data['windSpeed'], ...)
return power.toJson()

instead of

@app.route('/windturbine/create', methods=['POST'])
session = entityDB.create_session()
```

```
windturbine = entityModels.WindTurbine(req_data['powerCoefficient'],
req_data['radius'], ...)
entityDB.addElementToDatabase(session, windturbine))
```

The modification of the Energy Schedule service is as follows:

```
@app.route('/energyschedule', methods=['GET'])
opt.addGenerator('PV', req_data['generatedEnergy'][0])
opt.addGenerator('WT', req_data['generatedEnergy'][1])
opt.addPriceBuy(req_data['marketPrices'][0])
opt.addPriceSell(req_data['marketPrices'][0])
...
schedule = opt.getSchedule()
return schedule.toJson()

instead of

@app.route('/optimizer/getSchedule/<string:date_string>', methods=['GET'])
session = priceDB.create_session()
date = date.replace(hour=0, minute=0, second=0)
schedule = session.query(optimizerModel.Schedule).filter_by(date=date).first()
return schedule.toJson()
```

We implement Data service which is used to visualize the EMS data to the GUI component for the end-users as well as the Data service is used to deliver the raw data of the EMS to the end-users. The `GetPVPerodicData()` function handles the historical data of a photovoltaic which belongs to the `DataModel` file of our system. The `DataModel` file contains the Data service related functions, such as `GetPVPerodicData()`, `GetWTPerodicData()`. The `GetPVPerodicData()` function receives two parameters, such as `startDate`, `endDate` from requesters, and fetches data from the database depending on the `startDate` & `endDate` parameters. The following lines of codes represent an example of implementation of historical photovoltaic Data Service.

```
public function GetPVPerodicData($request)
{
    $startDate = $request->input("startDate");
    $endDate = $request->input("endDate");
    ...
    $result =  DB::select($sql);
    ...
    return $this->ResultFormat($result);
}
```

The Coordinator is a logical component of our EMS. The `custom.js` file represents the implementation of the Coordinator. The `custom.js` file contains multiple functions, such as `GetPriceList()`, `GetWeatherData()`, `GetSchedule()`, `GetWTData()`. Each of the functions performs a particular task. For example, the `GetPriceList()` function collects market prices data from external service provider, the `GetWTData()` function collects wind turbine data from the existing implementation. The `StoreData()` function is responsible to store all necessary data to the database. To get the hourly data of a wind turbine and a phovoltaic, the `ReadDataEveryHour()` function is called in each hour. The following lines of codes represent an example of implementation of the Coordinator.

```
function ReadDataEveryHour() {
    ...
    GetWTData();
    GetPVData();
    ...
    StoreData();
    setInterval(ReadDataEveryHour, 1000 * 60 * 60);
}
function GetWTData() {
    $.get("windturbine", { "windSpeed" : 4.72, "bladeLength" : 1.8, ... }
    , function(result){
    ...
    });
}
function GetPVData() {
    $.get("photovoltaic", { "panelArea" : 1.3, "panelAngle": 33, ... }
    , function(result){
    ...
    });
}
```

We collect weather data, such as temperature, wind speed, humidity from the OpenWeatherMap platform and we collect the market prices data from the ENTSO-E platform those are the external service providers. We need some load/demand data to make the EMS operational. As we do not implement a controllable load service because of the time limitation, so, to collect the load/demand data, we use real energy consumption data of a small household and a small commercial building from the Open Power System Data (OPSD) platform.

For implementing the EMS, we use Laravel Framework 6.9.0, PHP 7.3.12, and MySQL 10.4.10 database to store data in back-end. In front-end, to visualize the EMS data, we use basic HTML, CSS, jQuery 3.3, and D3.js 5.4.0 to draw the charts. We use JSON message for each of the Web services.

5.3 Data Visualization

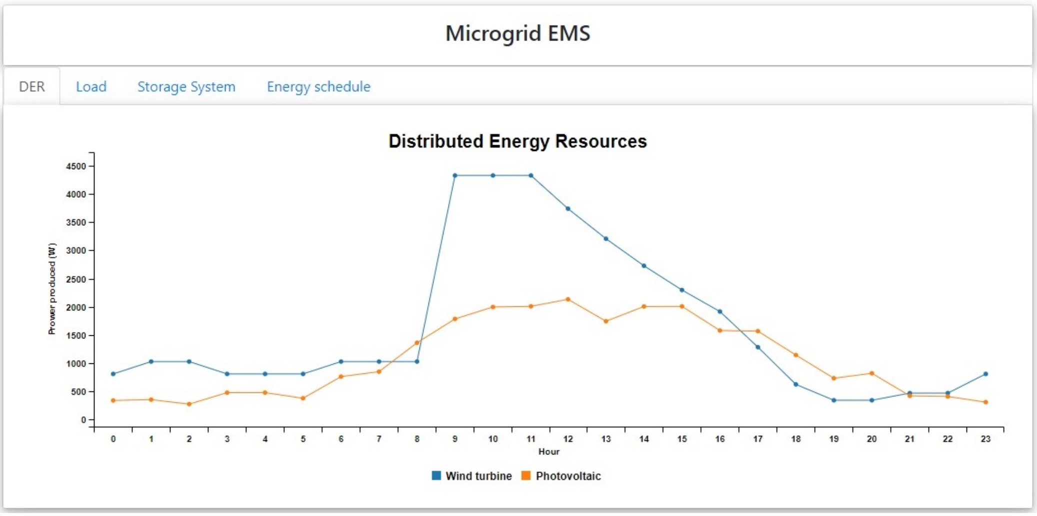
The GUI component of the EMS is used to visualize the various types of data, such as the power generation data of a wind turbine and a photovoltaic, the real load/demand data of a household and a commercial building, and the optimal energy schedule data. The Coordinator (Figure 5.1b) of the EMS provides these data via the Data service to the GUI component. Figure 5.2a, 5.2b, and 5.2c represent the DERs, loads, and optimal energy schedule, respectively. Listing 5.1 represents an example of response message of our implemented Data service. Listing 5.2 represents an example of response message of the Energy Schedule service. The Energy Schedule service response message contains multiple attributes, such as buildings, batteries, and exchange_with_main_grid. The buildings attribute indicates when to start and stop the appliances. The batteries attribute indicates different states of batteries, such as idle, charge, discharge with corresponding rate and energy for each hour. The exchange_with_main_grid attribute indicates the amount of power borrows from the main grid or sells to the main grid in each hour.

```
{
    "Data": {
        "name": "Wind turbine",
        "periodic": {
            "2020-02-14": {
                "0h": 0.8077,
                "1h": 1.0269,
                "2h": 1.0269,
                "3h": 0.8077,
                "4h": 0.8077,
                "5h": 0.8077,
                "6h": 1.0269,
                "7h": 1.0269,
                "8h": 1.0269,
                "9h": 4.3287,
                "10h": 4.3287,
                "11h": 4.3287,
                "12h": 3.7393,
                "13h": 3.206,
                "14h": 2.7259,
                "15h": 2.2964,
                "16h": 1.9145,
                "17h": 1.2826,
                "18h": 0.6221,
                "19h": 0.3407,
                "20h": 0.3407,
                "21h": 0.4674,
                "22h": 0.4674,
                "23h": 0.8077
            }
        },
        "datetime": "2020-02-14 23:31:10",
        "unit": "kWh"
    }
}
```

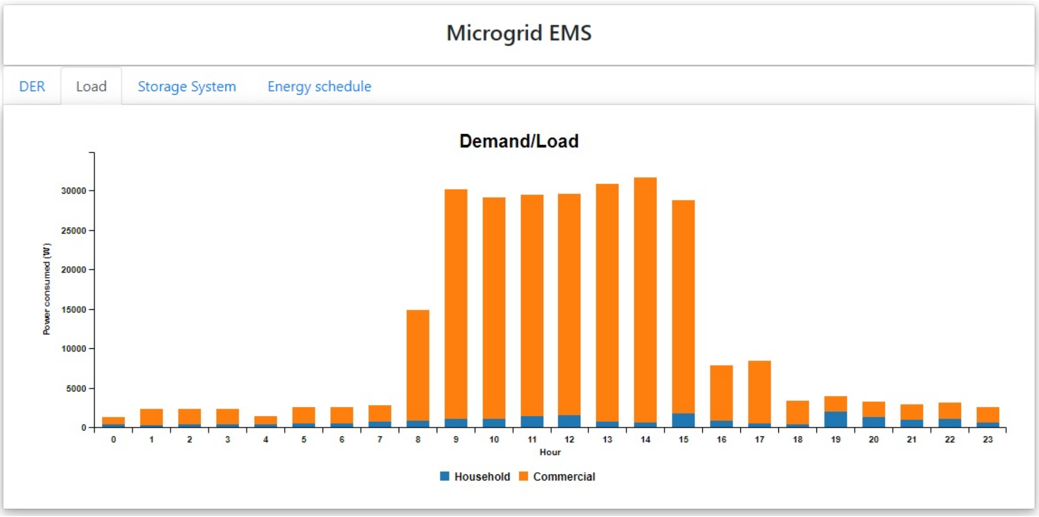
Listing 5.1: Data service (read periodic data of wind turbines)

```
{
   "buildings":{
      "building":{
         "cloth_dryer":{
            "start":13,
            "end":14
         },
         "water_heater":{
            "start":21,
            "end":23
         },
         "cloth_washer":{
            "start":14,
            "end":16
         }
      }
   },
   "batteries":{
      "battery":{
         "state":[
            "idle",
            "idle",
            "idle",
            "idle",
            "idle",
            ...
            "idle"
         ],
         "rate":[
            0,
            0,
            0,
            0,
            0,
            ...
            0
         ],
         "energy":[
            0,
            0,
            0,
            0,
            0,
            ...
            0
         ]
      }
   },
   "exchange_with_main_grid":[
      0.21,
      0.92,
      1.08,
      1.13,
      0.17,
      1.37,
      ...
      4.98
   ]
}
```

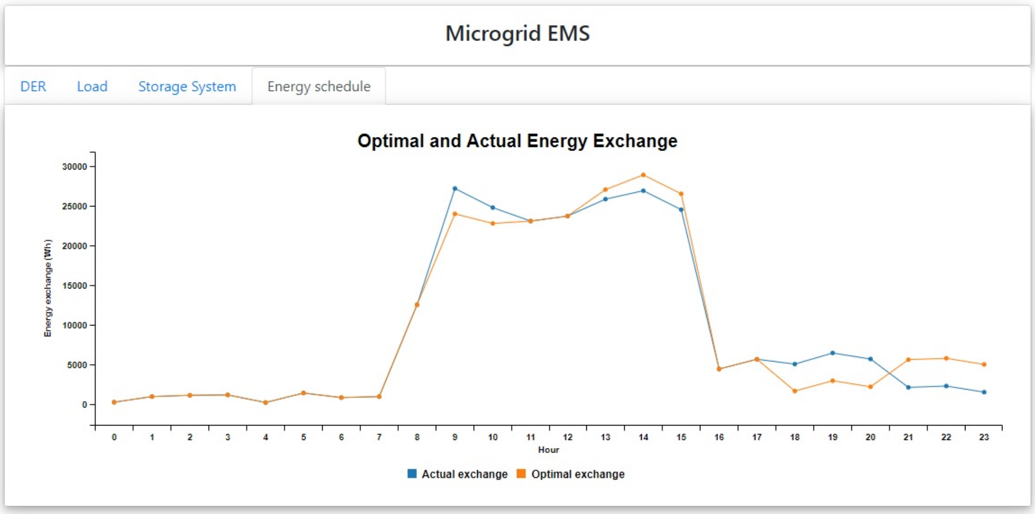
Listing 5.2: Energy Schedule service



(a) Energy generation of DERs



(b) Uncontrollable loads (household & commercial)



(c) Optimal and Actual energy exchange

Figure 5.2: DERs, Loads and Energy Schedule data of the EMS

6 Conclusion and Future Work

The perfect utilization of a microgrid is ensured by an effective EMS. An effective microgrid EMS always has to consider the functional and non-functional properties. In this thesis, we identify some essential functional and non-functional properties of EMSs and discuss the application of the non-functional properties on the capabilities of microgrid EMSs. As services are the basic constructs of SOC, we design five categories of RESTful Web services to build a service-oriented EMS for microgrids. We define interfaces, resources, accessible methods for each of the Web services. By considering the basic functionalities of EMSs, we design a conceptual microgrid EMS model that is applicable for any types of microgrid EMSs.

According to our contribution of this thesis, we find that the application of the design principles of SOC ensure a sustainable development of EMSs for microgrids. A service-oriented EMS is loosely coupled, scalable, reusable, interoperable, and platform independent against the traditional EMSs. We identify that a microgrid EMS needs at least four basic functionalities, such as Human-Machine Interface (HMI), Data collection, Optimization, and Forecast & Data Analysis. The Web services ensure rapid, sustainable, and low-cost development of EMSs. The Web services are easy to use for the end-users without requiring the huge knowledge of a specific domain. By using our designed Web services, we construct a concrete EMS that represents a working example of a service-oriented EMS.

There are some limitations of our thesis. First, our designed conceptual model of microgrid EMSs only supports four functionalities of EMSs rather than other functionalities, such as load profiles, balance plans, CO_2 signals [17]. Second, we are not able to design the Web services for uncontrollable loads, such as household, commercial because, these require real energy consumption data. Third, we implement a concrete EMS depending on the simulated data of DERs and DES.

Generally, the standard RESTful Web services, such as Google APIs, Facebook APIs etc. have an authentication mechanism for the end-users of the Web services. To design a standard authentication mechanism for accessing the Web services is one of the future work. As there can be many types of controllable loads and DERs, we design some interfaces of such types, so, to design more interfaces of the DERs (diesel, fuel cells) and controllable loads (air conditioner, electric vehicle) can be a potential future work. Defining a standard for microgrid Web services will be a remarkable work in future. The RSDL is very essential to describe the RESTful Web services, but it is not stable yet. Therefore, doing more research on RSDL that can support any type of data types will be a great future work.

Bibliography

[1] M. A. Ahmed, Y. C. Kang, Y.-C. K. 3. “Communication Network Architectures for Smart-House with Renewable Energy Resources”. In: *Energies — Open Access Journal* 8 (Aug. 2015), pp. 8716–8735. DOI: 10.3390/en8088716 (cit. on pp. 13, 22).

[2] A. Ali, A. Farooq, Z. Muhammad, F. Habib, S. A. Malik. “A Review: DC Microgrid Control and Energy Management System”. In: *International Journal of Electrical and Electronic Science* 2 (Aug. 2015), pp. 24–30 (cit. on pp. 13, 17).

[3] S. Allamaraju. *RESTful Web Services Cookbook*. First Edition. O’Reilly, 2010. ISBN: 978-0-596-80168-7 (cit. on pp. 35, 38).

[4] E. Álvarez, A. M. Campos, R. García, S. González, C. Díez. “Scalable and Usable Web Based Supervisory and Control System for Microgrid Management”. In: *International Conference on Renewable Energies and Power Quality* 1 (Apr. 2010), pp. 763–768. DOI: 10.24084/repqj08.467 (cit. on pp. 31, 32).

[5] S. Aman, Y. Simmhan, V. K. Prasanna. “Energy management systems: state of the art and emerging trends”. In: *IEEE Communications Magazine* 51 (Jan. 2013), pp. 114–119. DOI: 10.1109/MCOM.2013.6400447 (cit. on pp. 13, 31).

[6] P. Bendt. “Are We Missing Energy Savings in Clothes Dryers?” In: *ACEEE Summer Study on Energy Efficiency in Buildings* (2010) (cit. on p. 48).

[7] M. Braun, K. Büdenbender, D. Magnor, A. Jossen. “Photovoltaic Self-Consumption in Germany - Using Lithium-Ion Storage to Increase Self-Consumed Photovoltaic Energy”. In: *European Photovoltaic Solar Energy Conference* (Sept. 2009), pp. 3121–3127. DOI: 10.4229/24thEUPVSEC2009-4BO.11.2 (cit. on p. 45).

[8] Q. Chen, H. Ghenniwa, W. Shen. “Web-services infrastructure for information integration in power systems”. In: *Power Engineering Society General Meeting* (Jan. 2006). DOI: 10.1109/PES.2006.1709387 (cit. on p. 16).

[9] Christiana Honsberg and Stuart Bowden. *Solar Radiation on a Tilted Surface*. 2019. URL: https://www.pveducation.org/ (cit. on p. 45).

[10] H. Dagdougui, R. Minciardi, A. Ouammi, M. Robba, R. Sacile. “Modelling and control of a hybrid renewable energy system to supply demand of a green-building”. In: *International Environmental Modelling and Software Society (iEMSs)* (July 2010) (cit. on p. 45).

[11] e-CFR. *Electronic Code of Federal Regulations e-CFR*. 2020. URL: https://ecfr.io/Title-10/chapterII (cit. on pp. 47, 48).

[12] M. Eastment, R. Hendron. “Method for Evaluating Energy Use of Dishwashers, Clothes Washers, and Clothes Dryers”. In: *UNT digital library* (Aug. 2006) (cit. on pp. 47, 48).

[13] K. Eger, J. Goetz, R. Sauerwein, R. Frank, D. Boëda, I. M. D. de Cerio, R. Artych, E. Leukokilos, N. Nikolaou, L. Besson. “Microgrid Functional Architecture Description”. In: (Mar. 2013) (cit. on p. 16).

[14] T. Erl. *SOA: Principles of Service Design*. Prentice Hall, 2008. ISBN: 978-0132344821 (cit. on pp. 34, 44).

[15] H. Fan, Q. Yuan, H. Cheng. "Multi-Objective Stochastic Optimal Operation of a Grid-Connected Microgrid Considering an Energy Storage System". In: *Applied Sciences — Open Access Journal* 8 (Dec. 2018) (cit. on pp. 17, 24, 44, 46, 48).

[16] I. Georgievski, V. Degeler, G. A. Pagani, T. A. Nguyen, A. Lazovik, M. Aiello. "Optimizing Energy Costs for Offices Connected to the Smart Grid". In: *IEEE Transactions on Smart Grid* 3.4 (Dec. 2012), pp. 2273–2285. DOI: 10.1109/TSG.2012.2218666 (cit. on p. 13).

[17] I. Georgievski, L. Fiorini, M. Aiello. "Towards Service- Oriented and Intelligent Microgrids". In: *Applications of Intelligent Systems* (Jan. 2020), pp. 1–6. DOI: 10.1145/3378184.3378214 (cit. on pp. 35, 43, 61).

[18] I. Georgievski, T. A. Nguyen, F. Nizamic, B. Setz, A. Lazovik, M. Aiello. "Planning meets activity recognition: Service coordination for intelligent buildings". In: *Pervasive and Mobile Computing* 38.1 (July 2017), pp. 110–139. DOI: 10.1016/j.pmcj.2017.02.008 (cit. on p. 21).

[19] M. El-Hendawi, H. A. Gabbar, G. El-Saady, E.-N. A. Ibrahim. "Control and EMS of a Grid-Connected Microgrid with Economical Analysis". In: *Energies — Open Access Journal* 11 (Jan. 2018). DOI: 10.3390/en11010129 (cit. on p. 33).

[20] M. A. Hossain, H. R. Pota, W. Issa, M. J. Hossain. "Overview of AC Microgrid Controls with Inverter-Interfaced Generations". In: *Energies — Open Access Journal* 10 (Aug. 2017), p. 1300. DOI: 10.3390/en10091300 (cit. on pp. 20, 22).

[21] M. Huhns, M. Singh. "Service-Oriented Computing: Key Concepts and Principles". In: *IEEE Internet Computing* 9 (Feb. 2005), pp. 75–81. DOI: 10.1109/MIC.2005.21 (cit. on p. 13).

[22] Jess. "Wind Turbine Power Calculations". In: *RWE Npower Renewablesement* (2010) (cit. on pp. 44, 45).

[23] R. Jonathan, R. Cavicchio, R. Sinnema, E. Wilde. "RESTful Service Description Language (RSDL): Describing RESTful Services Without Tight Coupling". In: *Balisage Series on Markup Technologies* 10 (Aug. 2013). DOI: 10.4242/BalisageVol10.Robie01 (cit. on pp. 39, 40).

[24] F. Katiraei, R. Iravani, N. Hatziargyriou, A. Dimeas. "Microgrids management: Controls and operation aspects of microgrids". In: *IEEE Power and Energy Magazine* 6 (June 2008), pp. 54–65. DOI: 10.1109/MPE.2008.918702 (cit. on pp. 25–28).

[25] H.-M. Kim, T. Kinoshitar. "A New Challenge of Microgrid Operation". In: *Security-Enriched Urban Computing and Smart Grid* (Sept. 2010), pp. 250–260. DOI: 0.1007/978-3-642-16444-6_32 (cit. on pp. 22, 23).

[26] E. Lasic. "Sustainable use of washing machine: modeling the consumer behavior related resources consumption in use of washing machines". In: (Mar. 2014) (cit. on p. 47).

[27] D. Lau, J. Mylopoulos. "Designing Web Services with Tropos". In: *International Conference on Web Services* (July 2004). DOI: 10.1109/ICWS.2004.1314752 (cit. on p. 34).

[28] E.-K. Lee, W. Shi, R. Gadh, W. Kim. "Design and Implementation of a Microgrid Energy Management System". In: *Sustainability — Open Access Journal* 8 (Nov. 2016). DOI: 10.3390/su8111143 (cit. on pp. 13, 15, 17, 29, 30, 44, 48).

[29] J. Lihu, Z. Yongqiang, W. Yinshun. “Architecture Design for New AC-DC Hybrid Microgrid”. In: *IEEE First International Conference on DC Microgrids* (June 2015). DOI: 10.1016/j.rser.2014.11.054 (cit. on pp. 13, 25).

[30] L. Mariam, M. Basu, M. F. Conlon. “A Review of Existing Microgrid Architectures”. In: *Hindawi Publishing Corporation, Journal of Engineering* 2013 (Mar. 2013), p. 8. DOI: 10.1155/2013/937614 (cit. on pp. 18–20, 25).

[31] M. Massé. *REST API Design Rulebook*. First Edition. O’Reilly, 2011. ISBN: 978-1-449-31050-9 (cit. on pp. 35, 36, 38).

[32] N. Naik. “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP”. In: *International Systems Engineering Symposium (ISSE)* (Oct. 2017). DOI: 10.1109/SysEng.2017.8088251 (cit. on pp. 41, 49).

[33] E. O. Ogunniyi, H. Pienaar. “Overview of Battery Energy Storage System Advancement for Renewable (Photovoltaic) Energy Applications”. In: *Domestic Use of Energy (DUE)* (Apr. 2017). DOI: 10.23919/DUE.2017.7931849 (cit. on p. 46).

[34] G. A. Pagani, M. Aiello. “Generating Realistic Dynamic Prices and Services for the Smart Grid”. In: *IEEE SYSTEMS JOURNAL* 9.1 (Mar. 2015), pp. 191–198. DOI: 10.1109/JSYST.2014.2320800 (cit. on p. 44).

[35] G. A. Pagani, M. Aiello. “Service Orientation and the Smart Grid state and trends”. In: *Service Oriented Computing and Applications* 6.3 (July 2012), pp. 267–282. DOI: 10.1007/s11761-012-0117-z (cit. on p. 13).

[36] M. P. Papazoglou, W.-J. van den Heuvel. “Service-Oriented Design and Development Methodology”. In: *International Journal of Web Engineering and Technology* (Jan. 2006), pp. 412–442. DOI: 10.1504/IJWET.2006.010423 (cit. on pp. 13, 34, 41).

[37] M. P. PAPAZOGLOU, P. TRAVERSO, S. DUSTDAR, F. LEYMANN. “SERVICE-ORIENTED COMPUTING: A RESEARCH ROADMAP”. In: *International Journal of Cooperative Information Systems* 17 (June 2008), pp. 223–255. DOI: 10.1142/S0218843008001816 (cit. on pp. 13, 34, 43).

[38] M. Papazoglou, D. Georgakopoulos. “Introduction: Service-oriented computing”. In: *Communications of the ACM* (Oct. 2003). DOI: 10.1145/944217.944233 (cit. on p. 43).

[39] I. Patrao, E. Figueres, G. Garcerá, R. González-Medina. “Microgrid architectures for low voltage distributed generation”. In: *Renewable and Sustainable Energy Reviews* 43 (Mar. 2015), pp. 415–424. DOI: 10.1016/j.rser.2014.11.054 (cit. on pp. 18, 20–22).

[40] H. Pourbabak, T. Chen, B. Zhang, W. Su. “Control and Energy Management System in Microgrids”. In: *Clean Energy Microgrids* (May 2017). DOI: 10.1049/PBPO090E_ch3 (cit. on pp. 13, 25–28, 31, 32, 44).

[41] E. Pouresmaeil, J. M. Gonzalez, C. A. Canizares, K. Bhattacharya. “Smart Residential Load Simulator for Energy Management in Smart Grids”. In: *IEEE Transactions on Industrial Electronics* 66.2 (Feb. 2019), pp. 1443–1452. DOI: 10.1109/TIE.2018.2818666 (cit. on pp. 47, 48).

[42] G. Rancilio, A. Lucas, E. Kotsakis, G. Fulli, M. Merlo, M. Delfanti, M. Masera. “Modeling a Large-Scale Battery Energy Storage System for Power Grid Application Analysis”. In: *energies* 12 (Aug. 2019). DOI: 10.3390/en12173312 (cit. on p. 46).

[43] E. Rodriguez-Diaz, E. J. Palacios-Garcia, A. Anvari-Moghaddam, J. C. Vasquez, J. M. Guerrero. “Real-Time Energy Management System for a Hybrid AC/DC Residential Microgrid”. In: *IEEE International Conference On DC Microgrids* (June 2017). DOI: 10.1109/ICDCM.2017.8001053 (cit. on p. 44).

[44] A. A. Salam, A. Mohamed, M. A. Hannan. “TECHNICAL CHALLENGES ON MICROGRIDS”. In: *Hindawi Publishing Corporation, Journal of Engineering* 3 (Dec. 2008) (cit. on p. 25).

[45] A. Sarkak, D. K. Behera. “Wind Turbine Blade Efficiency and Power Calculation with Electrical Analogy”. In: *International Journal of Scientific and Research Publications* 2.2 (Feb. 2012) (cit. on pp. 44, 45).

[46] H. Shayeghi, E. Shahryari, M. Moradzadeh, P. Siano. “A Survey on Microgrid Energy Management Considering Flexible Energy Sources”. In: *Energies — Open Access Journal* (June 2019). DOI: 10.3390/en12112156 (cit. on pp. 13, 33).

[47] W. Shi, E.-K. Lee, D. Yao, R. Huang, C.-C. Chu, R. Gadh. “Evaluating Microgrid Management and Control with an Implementable Energy Management System”. In: *Smart Grid Communications* (Nov. 2014). DOI: 10.1109/SmartGridComm.2014.7007658 (cit. on pp. 29, 48).

[48] W. Shi, N. Li, C.-C. Chu, R. Gadh. “Real-Time Energy Management in Microgrids”. In: *Transactions on Smart Grid* 8.1 (Jan. 2017), pp. 228–238. DOI: 10.1109/TSG.2015.2462294 (cit. on p. 23).

[49] Y. Shin, W. Park, I. Lee. “Design of Microgrid Web Services for Microgrid Applications”. In: *International Conference on Ubiquitous and Future Networks* (July 2017). DOI: 10.1109/ICUFN.2017.7993913 (cit. on pp. 15, 35, 41, 43, 44).

[50] P. Sivachandran, R. Muthukumar. “An Overview of Microgrid System”. In: *International Journal of Applied Engineering Research* 9 (Jan. 2014), pp. 12353–12376 (cit. on pp. 24, 25).

[51] J. G. Slootweg, S. W. H. de Haan, H. Polinder, W. L. Kling. “General Model for Representing Variable-Speed Wind Turbines in Power System Dynamics Simulations”. In: *IEEE Power Engineering Review* (Dec. 2002). DOI: 10.1109/TPWRS.2002.807113 (cit. on pp. 44, 45).

[52] W. Su, J. Wang. “Energy Management Systems in Microgrid Operations”. In: *The Electricity Journal* 25.8 (Oct. 2012), pp. 45–60. DOI: 10.1016/j.tej.2012.09.010 (cit. on pp. 13, 15, 17, 18, 20–23, 25–28, 33, 44).

[53] F. Y.Melhem. “Optimization methods and energy management in ßmart grids"”. In: *Electric power* (Sept. 2018) (cit. on pp. 46, 48).

[54] Y. Zhang, M. Mao, M. Ding, L. Chang. “Study of Energy Management System for Distributed Generation Systems”. In: *Electric Utility Deregulation and Restructuring and Power Technologies* (May 2008). DOI: 10.1109/DRPT.2008.4523825 (cit. on pp. 29, 30, 33, 44).

[55] M. F. Zia, E. Elbouchikhib, M. Benbouzida. “Microgrids energy management systems: A critical review on methods, solutions, and prospects”. In: *Applied Energy* (June 2018), pp. 1033–1055. DOI: 10.1016/j.apenergy.2018.04.103 (cit. on pp. 13, 26, 29, 30, 33).

A DER Service Description

```
<?xml version="1.0" ?>
<service name="DER"
  xmlns:html="http://www.w3.org/1999/xhtml/" xmlns="http://dev.mgems.com/rsdl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dev.mgems.com/rsdl
rsdl.xsd">

  <documentation>
    The DER Web service description.
  </documentation>

  <start ref="res-home"/>

  <media-types>
    <media-type id="med-html" name="text/html">
      <documentation> HTML documents. </documentation>
    </media-type>
    <media-type id="med-json" name="JSONSchema">
      <documentation> JSON document. </documentation>
    </media-type>
  </media-types>

  <resources>
    <resource id="res-home" name="home">
      <location uri="/"/>
      <links>
        <link link-relation-ref="rel-windturbine" resource-ref="res-windturbine"/>
        <link link-relation-ref="rel-photovoltaic" resource-ref="res-photovoltaic"/>
        <link link-relation-ref="rel-biomass" resource-ref="res-biomass"/>
      </links>
      <methods>
        <method name="GET">
          <response>
            <representation media-type-ref="med-html" entity="resources"/>
          </response>
        </method>
      </methods>
    </resource>
    <resource id="res-windturbine" name="windturbine">
      <location template="/windturbine?windSpeed\&bladeLength\&airDensity\&rotorEffici\&gearBoxEffici
\&genEffici">
        <var name="windSpeed" uri-parameter-ref="par-windSpeed"/>
        <var name="bladeLength" uri-parameter-ref="par-bladeLength"/>
        <var name="airDensity" uri-parameter-ref="par-airDensity"/>
        <var name="rotorEffici" uri-parameter-ref="par-rotorEffici"/>
        <var name="gearBoxEffici" uri-parameter-ref="par-gearBoxEffici"/>
        <var name="genEffici" uri-parameter-ref="par-genEffici"/>
      </location>
      <methods>
```

```
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                    <documentation> Provide a JSON document that contains the produced power of the wind
turbine.</documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-photovoltaic" name="photovoltaic">
          <location template="/photovoltaic?panelArea&panelYield&panelAngle&panelLatitude&solarHorizon&
temperature">
            <var name="panelArea" uri-parameter-ref="par-panelArea"/>
            <var name="panelYield" uri-parameter-ref="par-panelYield"/>
            <var name="panelAngle" uri-parameter-ref="par-panelAngle"/>
            <var name="panelLatitude" uri-parameter-ref="par-panelLatitude"/>
            <var name="solarHorizon" uri-parameter-ref="par-solarHorizon"/>
            <var name="temperature" uri-parameter-ref="par-temperature"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                    <documentation> Provide a JSON document that contains the produced power of the
photovoltaic.</documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
      </resources>
        <resource id="res-biomass" name="biomass">
          <location template="biomass?bioQty&bioVolu&bioLHV&plantEffici">
            <var name="bioQty" uri-parameter-ref="par-bioQty"/>
            <var name="bioVolu" uri-parameter-ref="par-bioVolu"/>
            <var name="bioLHV" uri-parameter-ref="par-bioLHV"/>
            <var name="plantEffici" uri-parameter-ref="par-plantEffici"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                    <documentation> Provide a JSON document that contains the produced power of the
biomass.</documentation>
                </representation>
              </response>
            </method>
          </methods>
         </resource>
        </resources>

      <link-relations>
        <link-relation id="rel-windturbine" name="windturbine">
          <documentation>
            This resource simulates the energy production of a wind turbine.
          </documentation>
        </link-relation>
        <link-relation id="rel-photovoltaic" name="photovoltaic">
          <documentation>
            This resource simulates the energy production of a photovoltaic.
          </documentation>
        </link-relation>
        <link-relation id="rel-biomass" name="biomass">
          <documentation>
            This resource simulates the energy production of a biomass.
          </documentation>
        </link-relation>
      <link-relations>

      <uri-parameters>
        <uri-parameter id="par-windSpeed" name="windSpeed" datatype="float">
          <documentation>
            The wind speed value should be in metres per second. For example, windSpeed = 4.72
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-bladeLength" name="bladeLength" datatype="float">
          <documentation>
            The blade length of a wind turbine should be in meter (m). For example, bladeLength = 2.5
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-airDensity" name="airDensity" datatype="float">
          <documentation>
            The air density at sea-level in kilograms per cubic metre. For example, airDensity = 1.23
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-rotorEffici" name="rotorEffici" datatype="float">
          <documentation>
            The rotor efficiency of a wind turbine should be in percentage (%). For example, rotorEffici
= 0.4
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-genEffici" name="genEffici" datatype="float">
          <documentation>
            The generator efficiency of a wind turbine should be in percentage (%). For example,
genEffici = 0.6
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-gearBoxEffici" name="gearBoxEffici" datatype="float">
          <documentation>
            The gear box efficiency of a wind turbine should be in percentage (%). For example,
gearBoxEffici = 0.7
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-panelArea" name="panelArea" datatype="float">
          <documentation>
            The total area of a solar panel should in m2. For example, panelArea = 1.2 m2.
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-panelYield" name="panelYield" datatype="float">
          <documentation>
            The panel yield of a photovoltaic, like panelYield = 0.20.
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-panelAngle" name="panelAngle" datatype="integer">
          <documentation>
            The solar panel angle should be in degree, like panelAngle = 33 or 35 degree.
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-panelLatitude" name="panelLatitude" datatype="integer">
          <documentation>
            The solar panel latitude should be in degree, like panelLatitude = 30 degree.
          </documentation>
        </uri-parameter>
        <uri-parameter id="solarHorizon" name="solarHorizon" datatype="float">
          <documentation>
            The solar radiation on a tilted Surface. More information can be found in https://www.
pveducation.org.
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-bioQty" name="bioQty" datatype="float">
          <documentation>
            The biomass quantity (cubic metres per hour).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-bioVolu" name="bioVolu" datatype="float">
          <documentation>
            The biomass volumetric mass (kilograms per cubic metre).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-bioLHV" name="bioLHV" datatype="int">
          <documentation>
            The lower heating value of the biomass (megajoules per kilogram).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-plantEffici" name="plantEffici" datatype="float">
          <documentation>
            The biomass plant efficiency (%).
          </documentation>
        </uri-parameter>
      </uri-parameters>
    </service>
```

B DES Service Description

```
    <?xml version="1.0" ?>
    <service name="DES"
      xmlns:html="http://www.w3.org/1999/xhtml/" xmlns="http://dev.emc.com/rsdl"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dev.emc.com/rsdl
rsdl.xsd">

      <documentation>
        The DES Web service description.
      </documentation>

      <start ref="res-home"/>

      <media-types>
        <media-type id="med-html" name="text/html">
          <documentation> HTML documents. </documentation>
        </media-type>
        <media-type id="med-json" name="JSONSchema">
          <documentation> JSON document. </documentation>
        </media-type>
      </media-types>

      <resources>
        <resource id="res-home" name="home">
          <location uri="/"/>
          <links>
            <link link-relation-ref="rel-battery" resource-ref="res-battery"/>
            <link link-relation-ref="rel-charge-battery" resource-ref="res-charge-battery"/>
            <link link-relation-ref="rel-discharge-battery" resource-ref="res-discharge-battery"/>
          </links>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-html" entity="resources"/>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-battery" name="battery">
          <location template="/battery?maxCapacity&minCapacity&charEffici&discharEffici">
            <var name="maxCapacity" uri-parameter-ref="par-maxCapacity"/>
            <var name="minCapacity" uri-parameter-ref="par-minCapacity"/>
            <var name="charEffici" uri-parameter-ref="par-charEffici"/>
            <var name="discharEffici" uri-parameter-ref="par-discharEffici"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
```

```
                    <documentation> Provide a JSON document that contains an overall specification of the
battery, and a unique id of the battery. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-charge-battery" name="charge-battery">
          <location template="/battery/{id}/charge?power">
            <var name="id" uri-parameter-ref="par-id"/>
            <var name="power" uri-parameter-ref="par-power"/>
          </location>
          <methods>
            <method name="POST">
              <response>
                <representation media-type-ref="med-json">
                    <documentation> Provide a JSON document that contains the total amount of power after
charging the battery. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-discharge-battery" name="discharge-battery">
          <location template="/battery/{id}/discharge?power">
            <var name="id" uri-parameter-ref="par-id"/>
            <var name="power" uri-parameter-ref="par-power"/>
          </location>
          <methods>
            <method name="POST">
              <response>
                <representation media-type-ref="med-json">
                    <documentation> Provide a JSON document that contains the total amount of power after
discharging the battery. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
      </resources>

      <link-relations>
        <link-relation id="rel-battery" name="battery">
          <documentation>
            This resource simulates a battery as a storage device for the microgrid EMSs.
          </documentation>
        </link-relation>
        <link-relation id="rel-charge-battery" name="charge-battery">
          <documentation>
            This resource simulates the charging functionality for a battery device.
          </documentation>
        </link-relation>
        <link-relation id="rel-discharge-battery" name="discharge-battery">
          <documentation>
            This resource simulates the discharging functionality for a battery device.
          </documentation>
        </link-relation>
```

```
  <link-relations>

  <uri-parameters>
    <uri-parameter id="par-maxCapacity" name="maxCapacity" datatype="integer">
      <documentation>
        The maximum capacity of a battery in kilowatt (KW), like maxCapacity = 100 kW.
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-minCapacity" name="minCapacity" datatype="integer">
      <documentation>
        The minimum capacity of a battery in kilowatt (KW), like minCapacity = 2 kW.
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-charEffici" name="charEffici" datatype="float">
      <documentation>
        The charging efficiency of a battery in  kilowatt (KW), like charEffici = 0.95.
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-discharEffici" name="discharEffici" datatype="float">
      <documentation>
        The discharging efficiency of a battery in  kilowatt (KW), like discharEffici = 0.95.
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-id" name="id" datatype="integer">
      <documentation>
        A unique id of a battery.
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-power" name="power" datatype="float">
      <documentation>
        The amount of power in kilowatt (KW), like power = 8.5 kW
      </documentation>
    </uri-parameter>
  </uri-parameters>
</service>
```

C Load Service Description

```
    <?xml version="1.0" ?>
    <service name="load"
      xmlns:html="http://www.w3.org/1999/xhtml/" xmlns="http://dev.emc.com/rsdl"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dev.emc.com/rsdl
rsdl.xsd">

      <documentation>
        The load Web service description.
      </documentation>

      <start ref="res-home"/>

      <media-types>
        <media-type id="med-html" name="text/html">
          <documentation> HTML documents. </documentation>
        </media-type>
        <media-type id="med-json" name="JSONSchema">
          <documentation> JSON document. </documentation>
        </media-type>
      </media-types>

      <resources>
        <resource id="res-home" name="home">
          <location uri="/"/>
          <links>
            <link link-relation-ref="rel-clothdryer" resource-ref="res-clothdryer"/>
            <link link-relation-ref="rel-waterheater" resource-ref="res-waterheater"/>
            <link link-relation-ref="rel-clothwasher" resource-ref="res-clothwasher"/>
          </links>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-html" entity="resources"/>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-waterheater" name="waterheater">
          <location template="/load/waterheater?volume&hotTemp&coldTemp&PR">
            <var name="volume" uri-parameter-ref="par-volume"/>
            <var name="hotTemp" uri-parameter-ref="par-hotTemp"/>
            <var name="coldTemp" uri-parameter-ref="par-coldTemp"/>
            <var name="PR" uri-parameter-ref="par-PR"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the energy consumption (kWh) data
 of the water heater </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-clothwasher" name="clothwasher">
          <location template="/load/clothwasher?laundryWeight&washerCapacity&duration&temperature">
            <var name="laundryWeight" uri-parameter-ref="par-laundryWeight"/>
            <var name="washerCapacity" uri-parameter-ref="par-washerCapacity"/>
            <var name="duration" uri-parameter-ref="par-duration"/>
            <var name="temperature" uri-parameter-ref="par-temperature"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the energy consumption (kWh) data
 of the cloth washer </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-clothdryer" name="clothdryer">
          <location template="/load/clothdryer?initialWeight&finalWeight&dryerCap&EF">
            <var name="initialWeight" uri-parameter-ref="par-initialWeight"/>
            <var name="finalWeight" uri-parameter-ref="par-finalWeight"/>
            <var name="dryerCap" uri-parameter-ref="par-coldTemp"/>
            <var name="EF" uri-parameter-ref="par-EF"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the energy consumption (kWh) data
 of the cloth dryer </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
      </resources>

      <link-relations>
        <link-relation id="rel-waterheater" name="waterheater">
          <documentation>
            This resource simulates the energy consumption of a water heater.
          </documentation>
        </link-relation>
        <link-relation id="rel-clothdryer" name="clothdryer">
          <documentation>
            This resource simulates the energy consumption of a cloth dryer.
          </documentation>
        </link-relation>
        <link-relation id="rel-clothwasher" name="clothwasher">
          <documentation>
            This resource simulates the energy consumption of a cloth washer.
          </documentation>
        </link-relation>
      <link-relations>

      <uri-parameters>
        <uri-parameter id="par-volume" name="volume" datatype="integer">
          <documentation>
            The amount of water need to heat for a day (liter/day).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-hotTemp" name="hotTemp" datatype="integer">
          <documentation>
            The hot water temperature (Celsius).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-coldTemp" name="coldTemp" datatype="integer">
          <documentation>
            The cold water temperature (Celsius).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-PR" name="PR" datatype="float">
          <documentation>
            The performance ratio of a water heater. The PR is between 0.85 to 0.94.
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-laundryWeight" name="laundryWeight" datatype="float">
          <documentation>
            The total laundry weight (kg).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-washerCapacity" name="washerCapacity" datatype="float">
          <documentation>
            The capacity of a cloth washer (kg).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-duration" name="duration" datatype="int">
          <documentation>
            The estimated duration of the main wash (minute).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-temperature" name="temperature" datatype="int">
          <documentation>
            The washing temperature of cloth washer (Celsius).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-initialWeight" name="initialWeight" datatype="float">
          <documentation>
            Water weight before drying (lbs).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-finalWeight" name="finalWeight" datatype="float">
          <documentation>
            Water weight after drying (lbs).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-dryerCap" name="dryerCap" datatype="float">
          <documentation>
            Drying capacity of a cloth dryer (lbs).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-EF" name="EF" datatype="float">
          <documentation>
            The Energy Factor of a cloth dryer. The minimum EF is 3.01.
          </documentation>
        </uri-parameter>
      </uri-parameters>
    </service>
```

D Energy Schedule Service Description

```
<?xml version="1.0" ?>
<service name="energyschedule"
  xmlns:html="http://www.w3.org/1999/xhtml/" xmlns="http://dev.emc.com/rsdl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dev.emc.com/rsdl
rsdl.xsd">

  <documentation>
    The energy schedule web service description.
  </documentation>

  <start ref="res-home"/>

  <media-types>
    <media-type id="med-html" name="text/html">
      <documentation> HTML documents. </documentation>
    </media-type>
    <media-type id="med-json" name="JSONSchema">
      <documentation> JSON document. </documentation>
    </media-type>
  </media-types>

  <resources>
    <resource id="res-home" name="home">
      <location uri="/"/>
      <links>
        <link link-relation-ref="rel-schedule" resource-ref="res-schedule"/>
      </links>
      <methods>
        <method name="GET">
          <response>
            <representation media-type-ref="med-html" entity="resources"/>
          </response>
        </method>
      </methods>
    </resource>
    <resource id="res-schedule" name="schedule">
      <location temperature="/energyschedule?maxGeneration&minGeneration&generatedEnergy&maxLoad&
minLoad&loads&maxStorage&minStorage&energyStorage&appliances&marketPrices">
        <var name="maxGeneration" uri-parameter-ref="par-maxGeneration"/>
        <var name="minGeneration" uri-parameter-ref="par-minGeneration"/>
        <var name="generatedEnergy" uri-parameter-ref="par-generatedEnergy"/>
        <var name="maxLoad" uri-parameter-ref="par-maxLoad"/>
        <var name="minLoad" uri-parameter-ref="par-minLoad"/>
        <var name="loads" uri-parameter-ref="par-loads"/>
        <var name="energyStorage" uri-parameter-ref="par-energyStorage"/>
        <var name="appliances" uri-parameter-ref="par-appliances"/>
        <var name="marketPrices" uri-parameter-ref="par-marketPrices"/>
      </location>
```

```
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the optimal energy schedule (
minimum cost) data of a microgrid. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
      </resources>

      <link-relations>
        <link-relation id="rel-schedule" name="schedule">
          <documentation>
            This resource processes the real-time energy schedule data of a microgrid.
          </documentation>
        </link-relation>
      <link-relations>

      <uri-parameters>
        <uri-parameter id="par-maxGeneration" name="maxGeneration" datatype="float">
          <documentation>
            The hourly maximum energy generation limit of an energy resource in kilowatt (kW).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-minGeneration" name="minGeneration" datatype="float">
          <documentation>
            The hourly minimum energy generation limit of an energy resource in kilowatt (kW).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-generatedEnergy" name="generatedEnergy" datatype="float">
          <documentation>
            The generatedEnergy parameter is a two dimensional array of float.
            For example,
                float generatedEnergy[m][n]
                m = number of energy resources (wind turbine, photovoltaic).
                n = 24 hours (0...23) energy data in kilowatt (kW).
                generatedEnergy = [[2.03, 3.0, 3.88, ..., 1.14], [1.33, 0.89, 3.33, ..., 0.99]]
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-maxLoad" name="maxLoad" datatype="float">
          <documentation>
            The hourly maximum load of a household or a commercial in kilowatt (kW).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-minLoad" name="minLoad" datatype="float">
          <documentation>
            The hourly minimum load of a household or a commercial in kilowatt (kW).
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-loads" name="loads" datatype="float">
          <documentation>
            The loads parameter is a two dimensional array of float.
            For example,
                float loads[m][n]
                m = number of households or commercials.
                n = 24 hours (0...23) load data in kilowatt (kW).
                loads = [[2.23, 3.50, 1.88, ..., 6.14], [0.33, 8.89, 7.33, ..., 9.00]]
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-energyStorage" name="energyStorage" datatype="float">
          <documentation>
            The energyStorage parameter is an array of batteries.
            For example,
                float energyStorage[battery1, battery2, ...]
                where, battery1/battery2 = [efficiency, max-charge, max-discharge, max-stored-energy,
charging/discharging-rate]
                battery1 = [0.93, 5, 3, 7, 0.2]
                energyStorage = [[0.93, 5, 3, 7, 0.2], [0.95, 5, 2, 7, 0.2]]
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-appliances" name="appliances" datatype="float">
          <documentation>
            The appliances parameter is an array of controllable appliances.
            For example,
                float appliances[appliance1, appliance2, ...]
                where, appliance1/appliance2 = [start time, end time, power, duration]
                appliance1 = [12, 17, 3, 3]
                appliances = [[12, 17, 3, 3], [10, 15, 7, 3]]
          </documentation>
        </uri-parameter>
        <uri-parameter id="par-marketPrices" name="marketPrices" datatype="float">
          <documentation>
            The marketPrices parameter is an array of float.
            For example,
                float marketPrices[n]
                n = 24 hours (0...23) market price data in euro/dollar.
                marketPrices = [0.59, 0.66, 0.45, ..., 0.32]
          </documentation>
        </uri-parameter>
      </uri-parameters>
    </service>
```

E Data Service Description

```
    <?xml version="1.0" ?>
    <service name="microgrid-data"
      xmlns:html="http://www.w3.org/1999/xhtml/" xmlns="http://dev.emc.com/rsdl"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dev.emc.com/rsdl
rsdl.xsd">

      <documentation>
        The microgrid data service description.
      </documentation>

      <start ref="res-home"/>

      <media-types>
        <media-type id="med-html" name="text/html">
          <documentation> HTML documents. </documentation>
        </media-type>
        <media-type id="med-json" name="JSONSchema">
          <documentation> JSON document. </documentation>
        </media-type>
      </media-types>

      <resources>
        <resource id="res-home" name="home">
          <location uri="/"/>
          <links>
            <link link-relation-ref="rel-RT-photovoltaic" resource-ref="res-RT-windturbine"/>
            <link link-relation-ref="rel-RT-photovoltaic" resource-ref="res-RT-photovoltaic"/>
            <link link-relation-ref="rel-RT-battery" resource-ref="res-RT-battery"/>
            <link link-relation-ref="rel-history-windturbine" resource-ref="res-history-windturbine"/>
            <link link-relation-ref="rel-history-photovoltaic" resource-ref="res-history-photovoltaic"/>
            <link link-relation-ref="rel-history-battery" resource-ref="res-history-battery"/>
            <link link-relation-ref="rel-hourly-price" resource-ref="res-hourly-price"/>
            <link link-relation-ref="rel-daily-price" resource-ref="res-daily-price"/>
            <link link-relation-ref="rel-RT-temperature" resource-ref="res-RT-temperature"/>
            <link link-relation-ref="rel-RT-windspeed" resource-ref="res-RT-windspeed"/>
            <link link-relation-ref="rel-history-temperature" resource-ref="res-history-temperature"/>
            <link link-relation-ref="rel-history-windspeed" resource-ref="res-history-windspeed"/>
            <link link-relation-ref="rel-RT-energyschedule" resource-ref="res-RT-energyschedule"/>
            <link link-relation-ref="rel-history-energyschedule" resource-ref="res-history-energyschedule
"/>
            <link link-relation-ref="rel-RT-peaktime" resource-ref="res-RT-peaktime"/>
            <link link-relation-ref="rel-history-peaktime" resource-ref="res-history-peaktime"/>
          </links>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-html" entity="resources"/>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-RT-windturbine" name="RT-windturbine">
          <location uri="/data/WT/current"/>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the real time data of the wind
turbines. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-RT-photovoltaic" name="RT-photovoltaic">
          <location uri="/data/PV/current"/>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the real time data of the
photovoltaics. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-RT-battery" name="RT-battery">
          <location uri="/data/BT/current"/>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the real time data of the
batteries. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-history-windturbine" name="history-windturbine">
          <location template="/data/WT/periodic?startDate&endDate">
            <var name="startDate" uri-parameter-ref="par-startDate"/>
            <var name="endDate" uri-parameter-ref="par-endDate"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the historical data of the wind
turbines. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-history-photovoltaic" name="history-photovoltaic">
          <location template="/data/PV/periodic?startDate&endDate">
            <var name="startDate" uri-parameter-ref="par-startDate"/>
            <var name="endDate" uri-parameter-ref="par-endDate"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the historical data of the
photovoltaics. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-history-battery" name="history-battery">
          <location template="/data/BT/periodic?startDate&endDate">
            <var name="startDate" uri-parameter-ref="par-startDate"/>
            <var name="endDate" uri-parameter-ref="par-endDate"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the historical data of the
batteries. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-hourly-price" name="hourly-price">
          <location template="/data/price/hourly?hour">
            <var name="hour" uri-parameter-ref="par-hour"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains an hour day-ahead market price. <
/documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-daily-price" name="daily-price">
          <location uri="/data/price/daily"/>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains a list of (24 hours) day-ahead
market prices. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-RT-temperature" name="RT-temperature">
          <location uri="/data/temperature/current"/>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the current temperature data of
the DERs (photovoltaic). </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-RT-windspeed" name="RT-windspeed">
          <location uri="/data/windspeed/current"/>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the current wind speed data of
the DERs (wind turbine). </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-history-temperature" name="history-temperature">
          <location template="/data/temperature/periodic?startDate&endDate">
            <var name="startDate" uri-parameter-ref="par-startDate"/>
            <var name="endDate" uri-parameter-ref="par-endDate"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the historical energy schedule
data of the EMS. </documentation>
                </representation>
              </response>
            </method>
          </methods>
        </resource>
        <resource id="res-history-windspeed" name="history-windspeed">
          <location template="/data/windspeed/periodic?startDate&endDate">
            <var name="startDate" uri-parameter-ref="par-startDate"/>
            <var name="endDate" uri-parameter-ref="par-endDate"/>
          </location>
          <methods>
            <method name="GET">
              <response>
                <representation media-type-ref="med-json">
                  <documentation> Provide a JSON document that contains the historical wind speed data of
 the DERs (wind turbine). </documentation>
```

```
              </representation>
            </response>
          </method>
        </methods>
      </resource>
      <resource id="res-RT-energyschedule" name="RT-energyschedule">
        <location uri="/data/energyschedule/current"/>
        <methods>
          <method name="GET">
            <response>
              <representation media-type-ref="med-json">
                <documentation> Provide a JSON document that contains the current energyschedule data
of the EMS. </documentation>
              </representation>
            </response>
          </method>
        </methods>
      </resource>
      <resource id="res-history-energyschedule" name="history-energyschedule">
        <location template="/data/energyschedule/periodic?startDate&endDate">
          <var name="startDate" uri-parameter-ref="par-startDate"/>
          <var name="endDate" uri-parameter-ref="par-endDate"/>
        </location>
        <methods>
          <method name="GET">
            <response>
              <representation media-type-ref="med-json">
                <documentation> Provide a JSON document that contains the historical wind speed data of
 the DERs (wind turbine). </documentation>
              </representation>
            </response>
          </method>
        </methods>
      </resource>
      <resource id="res-RT-peaktime" name="RT-peaktime">
        <location uri="/data/peaktime/current"/>
        <methods>
          <method name="GET">
            <response>
              <representation media-type-ref="med-json">
                <documentation> Provide a JSON document that contains the current peak time data of the
 EMS. </documentation>
              </representation>
            </response>
          </method>
        </methods>
      </resource>
      <resource id="res-history-peaktime" name="history-peaktime">
        <location template="/data/peaktime/periodic?year">
          <var name="year" uri-parameter-ref="par-year"/>
        </location>
        <methods>
          <method name="GET">
            <response>
              <representation media-type-ref="med-json">
                <documentation> Provide a JSON document that contains the historical peak time data of
the EMS. </documentation>
```

```
                </representation>
              </response>
            </method>
          </methods>
        </resource>
      </resources>

      <link-relations>
        <link-relation id="rel-RT-windturbine" name="RT-windturbine">
          <documentation>
            This resource is responsible to provide the real-time energy generation data of a wind
turbine(s).
          </documentation>
        </link-relation>
        <link-relation id="rel-RT-photovoltaic" name="RT-photovoltaic">
          <documentation>
            This resource is responsible to provide the real-time energy generation data of a
photovoltaic(s).
          </documentation>
        </link-relation>
        <link-relation id="rel-RT-battery" name="RT-battery">
          <documentation>
            This resource is responsible to provide the real-time energy storage data of a battery(s).
          </documentation>
        </link-relation>
        <link-relation id="rel-history-windturbine" name="history-windturbine">
          <documentation>
            This resource is responsible to provide the historical energy generation data of a wind
turbine(s).
          </documentation>
        </link-relation>
        <link-relation id="rel-history-photovoltaic" name="history-photovoltaic">
          <documentation>
            This resource is responsible to provide the historical energy generation data of a
photovoltaic(s).
          </documentation>
        </link-relation>
        <link-relation id="rel-history-battery" name="history-battery">
          <documentation>
            This resource is responsible to provide the historical energy storage data of a battery(s).
          </documentation>
        </link-relation>
        <link-relation id="rel-hourly-price" name="hourly-price">
          <documentation>
            This resource manages the market price data of a microgrid for a specific hour.
          </documentation>
        </link-relation>
        <link-relation id="rel-daily-price" name="daily-price">
          <documentation>
            This resource manages the daily (24 hours) market price data of a microgrid.
          </documentation>
        </link-relation>
        <link-relation id="rel-RT-temperature" name="RT-temperature">
          <documentation>
            This resource delivers the real-time temperature data of the DERs.
          </documentation>
        </link-relation>
```

```
    <link-relation id="rel-RT-windspeed" name="RT-windspeed">
      <documentation>
        This resource delivers the real-time wind speed data of the DERs.
      </documentation>
    </link-relation>
    <link-relation id="rel-history-temperature" name="history-temperature">
      <documentation>
        This resource delivers the historical temperature data of the DERs.
      </documentation>
    </link-relation>
    <link-relation id="rel-history-windspeed" name="history-windspeed">
      <documentation>
        This resource delivers the historical wind speed data of the DERs.
      </documentation>
    </link-relation>
    <link-relation id="rel-RT-energyschedule" name="RT-energyschedule">
      <documentation>
        This resource delivers the real-time energy schedule (minimum cost) data of a microgrid.
      </documentation>
    </link-relation>
    <link-relation id="rel-history-energyschedule" name="history-energyschedule">
      <documentation>
        This resource delivers the historical energy schedule (minimum cost) data of a microgrid.
      </documentation>
    </link-relation>
    <link-relation id="rel-RT-peaktime" name="RT-peaktime">
      <documentation>
        This resource delivers the recent year peak time data of the EMS.
      </documentation>
    </link-relation>
    <link-relation id="rel-history-peaktime" name="history-peaktime">
      <documentation>
        This resource delivers the previous years peak time data of the EMS.
      </documentation>
    </link-relation>
  <link-relations>

  <uri-parameters>
    <uri-parameter id="par-startDate" name="startDate" datatype="date">
      <documentation>
        The start date (yyyy-mm-dd) of the query, like startDate = "2019-12-20".
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-endDate" name="endDate" datatype="date">
      <documentation>
        The end date (yyyy-mm-dd) of the query, like endDate = "2019-12-29".
      </documentation>
    </uri-parameter>
    <uri-parameter id="par-year" name="year" datatype="integer">
      <documentation>
        The year (yyyy) of the query, like year = 2018.
      </documentation>
    </uri-parameter>
  </uri-parameters>
</service>
```

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature