MODELLING OF A MICROGRID USING Z NOTATION

by

Binisha Shrestha

Master of Science in Computer Science, University of North Dakota 2020

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota

December
2020

Name: Binisha Shrestha

Degree: Master of Science

This document, submitted in partial fulfillment of the requirements for the degree from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

DocuSigned by:
Emanuel S. Grant
18A2B28E81AE418...
Emanuel Grant

DocuSigned by:
Wen-Chen Hu
EE71D886B1C04D0...
Wen-Chen Hu

DocuSigned by:
Thomas Stokke
C2B5FA247FC44E6...
Thomas Stokke

This document is being submitted by the appointed advisory committee as having met all the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

DocuSigned by:
Chris Nelson
2E0AF088C733403...
Chris Nelson
Dean of the School of Graduate Studies

12/7/2020
Date

PERMISSION

Title Modelling of Microgrid using Z Notation

Department School of Electrical Engineering and Computer Science.

Degree Masters of Computer Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Binisha Shrestha

Dec 9, 2020

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to the members of my advisory Committee for their guidance and support during my time in the master's program at the University of North Dakota.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

ABSTRACT

A Microgrid is a group of electrical sources and connected loads that operate energy grids in grid-connected or islanded mode. Microgrid usage has increased recently due to improved technology and the effectiveness of renewable energy sources. To produce a balanced and stable power supply from microgrids and meet the load demand is a challenging research area in both the electrical engineering and software engineering fields. This work presents a formal model for representing the microgrid system to prevent failure or inconsistencies in the power generation and usage. A methodology for creating a formal model for a microgrid is a critical approach to overcoming the challenges of microgrid management and is examined in this work. The work was studied in two parts. The first part assessed the microgrid's existing class diagram that is then transformed into a precise representation in the Z notation. The Z notation is a mathematical specification language used for describing system properties, and to reason about possible refinements of a design. The second part involved verifying and validation of microgrid system through the creation of a structured specification using Z. The research addressed class diagram faults in model-based testing. Hence, the class diagrams are analyzed, recreated, and then designed using the formal notation in an iterative process, resulting in a precise description of the microgrid structure in a formal, unambiguous, and effective manner. This description can then be analyzed to determine the correctness of the UML description that will be used to design a microgrid power management system.

Index Terms - Microgrid, Renewable energy, Battery energy storage systems, energy storage system, Distributed generator, Z notation

CHAPTER 1

INTRODUCTION

This chapter first introduces the background of the microgrid system and gives the general problem description. Section 1.2 shows the scope of the work. The scope of the work consists of the general methodology description. Section 1.3 presents the motivation of the research towards the use of formal specification. Lastly, section 1.4 shows the approaches to solve the problem and the expected results.

1.1 **Problem Definition**

A microgrid is a distributed power system that has distributed energy resources (DER) effectively across multiple sources of power generations [1]. These sources include traditional power sources such as diesel, natural gas to more renewable sources such as solar, wind, or energy storage systems such as batteries. In addition to these DERs, the microgrid has the capability to operate in a grid-connected or islanded mode. In the island mode, the microgrid can operate if disconnected from the main grid. This allows it to continue operating even if the main grid goes down. In the grid-connected mode, microgrid controls the supplied power from microgrid to the main grid and exchanges the power when any fault or emergency occurs in main grid.

Microgrids are used to achieve diversification of energy sources, improve reliability, and reduce cost of load[2]. By enabling the integration of distributed resources such as wind and solar, these systems can operate while the main grid is down or help mitigate grid disturbances. The microgrid integrates distributed generators (DG), an energy storage system (ESS), and load. Figure 1.1 shows the typical structure of a microgrids comprising DG, battery energy storage system (BESS), solar photovoltaic systems (PV), and wind power systems connected

to distribution networks.

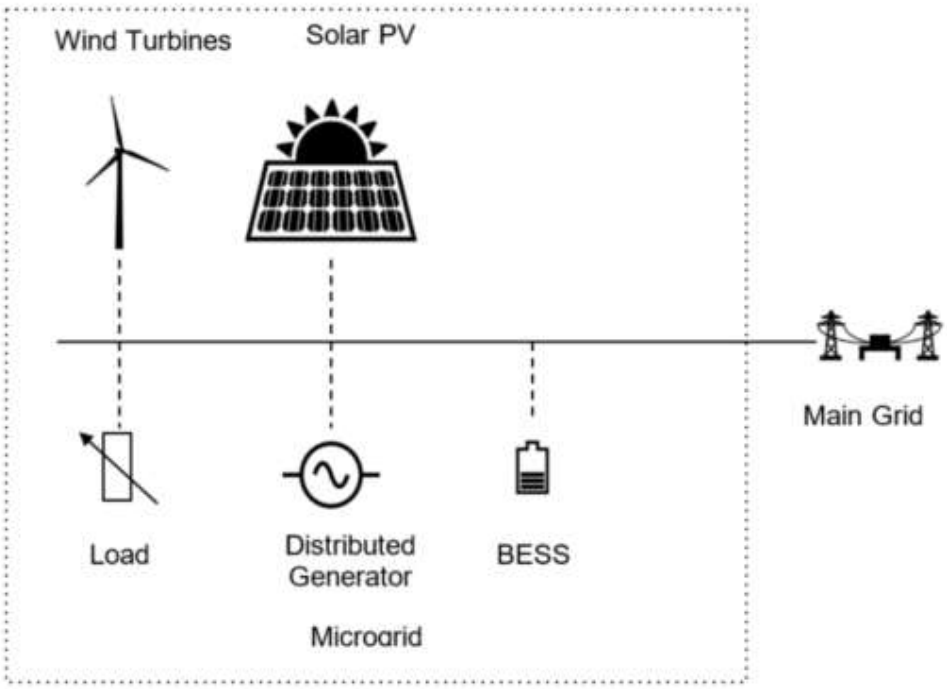

Figure 1. 1: Interactive operation of the microgrid

Traditionally, power grids are highly centralized, with power and energy flowing from large synchronous generators being distributed to end users [1]. However, the technological issues associated with traditional power grids have stimulated new power system technologies. With the emergence of DERs, microgrid technologies are playing an essential role in the electric power and energy system. The development of microgrids has brought many benefits, but does present significant challenges. For instance, microgrids include renewable sources, energy storage systems (ESS), distributed generator (DG), and software controls. To ensure power grids operate as efficiently care must be taken to ensure all these systems work together. It is necessary to ensure the operation of microgrids follow intelligent control strategies. To fulfill the purpose of microgrids being controllable, reliable, and to ensure security, strict

specification is required to prevent severe failure. To make the system more reliable the use of a formal specification has been adopted. A formal specification is a series of steps involving verification and refinements, which leads to a precise and unambiguous description of the system. This lets us help to determine the correctness of the design and implementation.

1.2 **Scope of work**

The scope of this work falls under the formal modeling of a microgrid system. Formal models are used to describe the system being developed, detail designs and it is used to verify that the requirements of the system being developed have been accurately specified. The components, attributes modeled through formal modeling, have been thoroughly described. For the UML modeling, I followed the IEEE 1547 "IEEE Standard for Interconnecting Distributed Resources with Electric Power Systems" guidelines [3] [4]. The document provides the uniform standard for the technical requirements for distributed energy resources and a suitable set of rules.

1.3 **Motivation**

Formal specification techniques and analysis have proven to be an efficient approach in providing understanding of defining software requirements and of software design. The formal specification is mostly being used in the field of safety-critical system development. In the microgrid system, I have introduced formalizing the specification using Z notation as a means of validation and verification of the system being developed. So the question arises, "why does it matter in a safety-critical system, and why is the correctness needed?". Microgrids having intricate design fall under safety-critical systems which require a high degree of safety assurance. Formal models functions as a concise and precise description of the behavior of the system and helps to prevent ambiguity. Therefore, this approach is an important step in trying

to ensure the correction of the system.

1.4 **Approach**

As stated earlier, the work was studied in two parts. The first part assessed the microgrid's existing class diagram [5], which was then transformed into a precise representation in the Z notation. UML class diagrams were used to outline the structure and relationship between the parts of the microgrid. The second part involved transforming the UML class diagram into corresponding formal specification with respect to data interactions and constraints using mathematically defined types. The formal specification was used as an unambiguous form of notation to analyze the requirements and during the implementation of the system. It allowed us to verify the system design, resolve ambiguities, and detect errors on the specification before system development. In this thesis, the formal specification of the microgrid involved various conditions, including internal state, outputs, and execution of the faulty statements.

The work represented in this thesis follows the model-based software development (MBD) technique. MBD employs the development of a series of UML requirements, design, and deployment that are derived through iterative steps [6]. The methodologies for transforming the class diagram into formal methods are specified by a set of rules. The rules are thoroughly described in the methodology section. The UML class diagram doesn't satisfy the techniques, it might satisfy the requirements set by the techniques; otherwise the class diagram further changes in an iterative process, based on the derived formal specification.

1.5 **Expected outcome**

The purpose of my research is to explore the concept of energy management systems (EMS) with a focus on modeling, testing, and validation prior to the integration of a microgrid. In this research, the expected outcome is to demonstrate the feasibility and benefits of applying formal

specification techniques in software development by documenting its use in developing energy management systems for microgrids. Further, I also expect to obtain a formal specification to analyze the correctness of the microgrid system.

1.6 **Thesis Structure**

The organization of the paper is as follows. Chapter II presents the theoretical background of the research. It introduces the microgrid system, energy management system, UML class diagram, formal method Z notation, and model-based software development techniques. Chapter III outlines the methodology applied and the methods used for this research and description of the process for developing Z notation. Chapter IV describes the result of the work. And lastly, chapter V concludes with the conclusion along with suggestions for future research.

CHAPTER 2

BACKGROUND

This chapter introduces the model-based software development technique and the concepts used in this research. It discusses the formal specification language and UML class diagram and control architecture of the microgrid system. Understanding the microgrid background, energy storage, power supply, and the current models used to define microgrids will provide a solid foundation for understanding how microgrid systems emerged and function. On microgrid systems, I explore different concepts required to design control strategies for distributed power systems. Additional information of the current technology provides information on how I can accomplish the microgrid system's improvement using the formal specification.

2.1 **The Unified Modeling Language**

Unified Modeling Language(UML) is the modeling language designed to model systems adopted by Object Management Group (OMG). UML specifies and documents the entire system at a high level of abstraction [7]. UML provides the system's visual model, which helps to give a behavioral overview of the system. UML works best with object-oriented modeling software, and provides a more precise specification than most object-oriented notation [8]. For the microgrid system, UML class diagrams are used to outline the structure of the system. The class diagram is represented in the rectangular box with three compartments: class, the object attributes, and the methods for the class. Initially, classes are identified, then attributes and operations of each class and relationships between each class are shown with different types of the arrow. The relationships includes the association, aggregations, compositions, and generalization. Figure 2.1 represents the UML class diagram as an example.

The advantage of UML diagrams is it can provide a visual model of the system, enabling developers to understand how a system works more efficiently. Furthermore, the use of a UML class diagram in system development helps to describe the significant entities and the relationship between them.

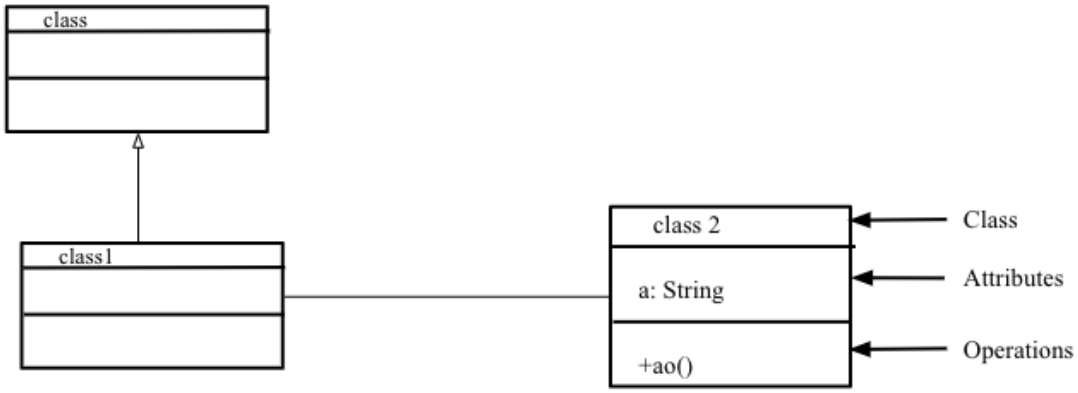

Figure 2. 1 Representation of Class as an example

2.2 **Formal specification**

A Specification language is a formal language that provides an unambiguous specification of the system's static structural and behavior of features. Specification language is based on the mathematical concept. The use of specification language can provide a reliable and precise reference of the system at much higher level that a programming language. It provides a reference for those testing the software, those implementing the software, and those writing instruction manuals for the system [9]. The accuracy and preciseness needed in the development of a formal specification can reduce the number of errors that occur and the number of omissions. Furthermore, the specifications' mathematical nature enables it to be automatically processed for error checking and consistency of the requirement specification of the system.

In my research, I have used Z notation to provide a formal specification of the microgrid system. Z notation is a strongly typed mathematical formal specification language. Although

UML diagrams are useful in modeling system behavior, there are limitations to what they can describe. So, given modern mathematics expression, it is natural to adapt mathematics to a system description. This provides a form of abstraction by enabling developers to focus on precisely what the system does without worrying about the meaning of terms or phrases or worrying about how to gather information from large and complex amounts of code. It also allows for an unambiguous, concise specification of the software [10].

The reason for the use of Z notation is that class diagrams do not offer enough precise to create unambiguous specification; it would be necessary to specify additional constraints about the object in the model. Using Z helps to specify system constraints and concise specifications, useful for software development. Z specifies changes in state, the relationship between the inputs and outputs, the system's operation, description of valid inputs, and the invariant's relationship of the system.

Z schema consists of two compartments, declaration, and predicate. First, a declaration of the attributes or the variables is defined. Figure 3 represents the example of the Z notation. In the example, variable $\mathbb{R}$ (Real) is defined for the type Input. Similarly, an object is defined by declaration, abbreviation, or by axiom. The predicate defines the structure of the declaration, describing the variables and the constraints. The predicate in figure 3 describes an invariant which must be satisfied by input. It asserts that the input must be between 5 and 12.

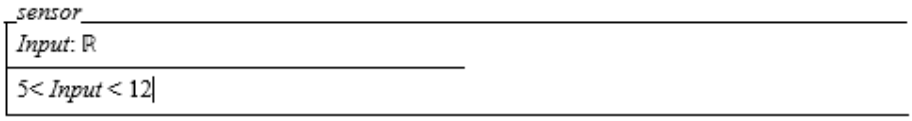

Figure 2. 2 Representation of Z notation as an example

2.3 **Model based software Development**

Model-based software development (MDSD) is a software development methodology that

focuses on an abstract representation of software systems' knowledge and activities [6]. Modeling serves to provide a detailed description of the system, the modeling of reusable classifiers, automatic code generation, and early verification and validation. MDSD has become a predominant paradigm in the development of safety-critical system development [11]. MDSD helps to provide a perspective on correctness, and how safety affects software development. The development process begins with gathering the requirements and developing the architectural design. The architectural design refers to the modeling of the software, which is the high-level decomposition of the system with the interfaces, interaction with the components. Next, a software solution that satisfies the architectural design is developed. This phase includes the definition of data structures and components of the assigned modules. In this research, we have integrated formal techniques with the MDSD approach. The resulting UML design is then analyzed by the formal modeling language.

This research focuses on developing techniques, methods, and processes to generate domain-specific software development environments. Use of MDSD is meant to increase productivity by maximizing compatibility between systems and simplifying design process [10]. In my research, the goal is to provide concise and expressive models throughout the development process such that they can provide developers detailed concepts and notations regarding the characteristics of the system.

2.4 **Microgrid control architecture**

Microgrid integrate distributed energy resources (DER), distributed generators (DG), energy storage systems (ESS), and the load used to supply the power. The prominent feature of the microgrid is controllability [12]. Microgrid control consists of i) determination of operation mode, either grid-connected, or islanded mode. ii) voltage and frequency regulation, power control, and microgrid monitoring, and iii) power control for each local generation and energy

storage system. The control of microgrids rely on communication. Depending on the state of DG in the microgrid, the control architecture can be: i) master-slave control, ii) peer-to-peer control, or iii) hierarchical control [1].

2.4.1 Master-slave control:

When a microgrid is operated in grid-connected mode, the main grid provides the frequency and voltage references for microgrid [1]. When microgrids operate in islanded mode, ESS and DG operate as master controllers to provide voltage and frequency reference for microgrid. When DG is used as a master controller, the microgrid can operate in the islanded mode for a longer period than ESS alone. This is due to ESS having limited storage capabilities compared to DG and a state of charge (SOC) limit during discharging state. However, the combination of ESS and DG helps reduce voltage and frequency fluctuation. This method can allow the microgrid system to stay in an islanded mode for a longer period.

2.4.2 Peer-to-peer control mode:

ESS is a control mode capable of adjusting the voltage and frequency based on the demand. If the frequency of the ESS is decreased, the DG will increase its power output to maintain the frequency. Similarly, if the voltage level drops, ESS will increase the reactive power output. In comparison with master-slave control mode, peer-to-peer control can make decisions with local information. Meanwhile, DG and ESS units still rely on master-slave control in the power-sharing task, which is why it is not widely implemented in practical application of microgrids.[1].

2.4.3 Hierarchical mode:

Hierarchical control is capable of predicting local demand and renewable power generation, so a set of operation plans is developed accordingly. As a result, the operation mode is selected based on the collected information. Through the use of this control architecture, transient power demand-supply can be balanced, and manage the load based on DG outputs and load demands. Even if the communication fails for a short while, microgrid can still maintain normal operation. Compared to the other control modes, the hierarchical mode can operate more efficiently and with more flexibly.

In this section, hierarchical and master-slave control is considered efficient architecture, and the UML class diagram and formal specification are designed based on these two control strategies.

CHAPTER 3

LITERATURE REVIEW

This chapter discusses the related work and identifies what aspects of previous work will be applied. These studies focus on a few selected research that contribute to the use of formal specification and how it can be implemented in many different system development phases. This chapter's primary goal is to provide a review of the literature and present an overview of the current research that contribute to the use of formal specification in system development. Furthermore, this research shows some of the challenges in large scale system development, the benefits of using the formal specification, and the differences between this research, and previous work in this area.

3.1 **Application of formal specification technique to microgrid representation**

Previous research by Maksym Tkach [5] presented formal specification techniques to model microgrids that can serve as a foundation for future work in the microgrid models. The author used UML behavioral diagrams and Object Constraint Language (OCL) specification language to describe the model behaviors. OCL is the formal declarative language used to describe expressions in UML. UML diagram can be used to show the basic structure of the system. OCL can help rigorously define the system design. It uses notation similar to object-oriented programming languages like C++ and Java.

The microgrid modeling begins with a rigorous definition of a microgrid's essential components in a thorough and error-free manner. Next, a class diagram was created to show the microgrid's structural aspects following the requirement elicitation process. Finally, OCL is used to define the system rigorously. OCL offers several essential features to support the verification and validation of the microgrid. OCL also provides a detailed analysis of the model description.

OCL expression supplemented UML in three ways; first, the OCL removes ambiguity and contradictions. Second, OCL expressions are used to create constraints. Constraints are restrictions on attributes that must always hold. Third, OCL expressions are used to define operations and specify the expected behavior of the system. Similarly, the UML-based Specification Environment **(**USE) tool is used for model transformation and validation of UML/OCL models. The study presented the USE as an invaluable error checking tool to ensure the correctness of the OCL syntax and the model behavior.

The microgrid model described in this paper is focused on residential purposes with single stakeholders. Similarly, this research addressed all the permissible variations of frequency and voltage and defined all the object's actions using OCL. However, the result explained in the context using OCL showed limited specification of the microgrid system. It failed to illustrate all possible energy flow among the system's components but provided insight for using the UML class diagram to see the comparison on the outputs.

3.2 **Formal validation of supervisory energy management systems for microgrids**

Research by G. Sugumar et al. [13] presents the verification and modeling of supervisory energy management systems using timed automata and UPPAAL, a formal verification model checker. UPPAAL is a model checking tool used to verify the correctness of real-time systems. The system from Sugumar et al.'s study consists of a diesel generator, battery energy storage system, photovoltaic array, and load. All of these distributed energy resources are connected with the common DC bus to fulfill the DC load demands. The algorithm for the flow of power in the microgrid system is the algorithm determined by examining the solar power system, battery energy storage, potential power generated by the diesel generator, and power required for the load. The algorithm addresses four major inputs: (i) when the load is satisfied by the solar energy; the controller directs power directly to the load, and if excess capacity is available

from solar that excess can be used to charge the battery storage system. (ii) if the solar energy available is inadequate and the battery storage system is sufficient to meet the load demand, then the storage devices are discharged to drive the load. (iii) The use of diesel generators initiates when the battery storage system and solar power is insufficient to meet the demand. (iv) the overall energy flow from all available sources is insufficient. In this rare situation, non-critical loads like heating devices, and chillers are shut down temporarily to meet the load demand.

The proposed method of modelling and verification of a supervisory energy manage system uses an efficient technique to validate the system, as it checks for all the addressed inputs and ensures the power balance between the load and power source. It also uses formal methods and timed automata (TA) using the UPAAL tool. The usefulness of TA seems very efficient for the real-time system [14]. The use of formal verification to validate the design can clearly be shown to be an efficient technique when demonstrating correctness in a formal specification. From the principles used in this paper, it helped address the questions raised in this paper providing better descriptions and controllability when defining formal specifications.

3.3 **Intelligent Agent-Based Microgrid Control**

The paper presented by Murali K. Kouluri and R. K. Pandey [15] explores implementing a multi-agent system for microgrids using JADE. The multi-agent system contains a control agent, DER agent, and load agent, each with respective responsibilities. The control agent monitors the system voltage and frequency to detect emergency conditions and holds the operation modes of monitoring grid or island. DER is responsible for the monitoring and control of power levels. Load agents contain information on power consumption and load number.

The paper presented – they did it the design and implementation of a multi-agent system for outage control of a microgrid. To verify the multi-agent system, a simulation was developed to

test control actions to/from the microgrid model using MatLab Simulink. The multi-agent system received and sent the messages to the circuits and performed control and management actions. The author presented a case study based on outage control of a microgrid using grid-connected mode, islanded mode, and the transition from grid-connected to islanded mode. This research conducted a comprehensive experiment to study the effectiveness of the multi-agent system through simulation. The paper presented an evaluation of an outage problem near the main grid using grid-connected, islanded mode and transition mode. The case study outlines that the control agent needs to sense the change in upstream outages, and it exchanges the information between agents. The paper is important as it demonstrated how to balance the frequency regulation and control the frequency disturbances.

While the paper presented an impressive capability of the multi-agent system, it focused on the outage problem. However, their system process could be helpful with the microgrid model, as the case study presented in this paper; it solves the timing problem that lacks the model of microgrid used in this study.

3.4 **Formal requirements for microgrid using Kaos and reference architecture**

The paper by Miguel Angel et al. [16] presents a formal specification requirement to model a smart grid system. This paper shows a lack of attention not needed in the design process, resulting in unnecessary costs. Hence the author proposed a systemic approach that combined reference architecture such as IntelliGrid, an architecture proposed by the Electric Power Research [17] or the SGAM architecture [18] and modelled goal-oriented requirement engineering methods for the microgrid design. This architecture recognizes the importance of the requirement phase, which included requirement elicitation, modeling, and using UML. The author followed the IEC 61850 architectures and chose the reference architecture to guide the process. The goal-oriented requirement engineering (GORE) refers to the use of objectives for

defining, requirements, eliminating the tradition dichotomy between functional and non-functional requirements. GORE is used through the KAOS tool (Knowledge Engineering Object System) to model system requirements. KAOS defines specific needs from a generic model based on different situations. It has four primary types of models: goal models, agent models, operation models, and object models. A goal model provides the objective the system should achieve. An agent responsible for meeting the objectives. The operation model specifies the operations that the agent must perform, and an object model identifies the object used in the KAOS model [19]. It provides traceability, completeness, and ambiguity. The formal requirements specification is generated both by KAOS to a formal representation based on based LTL (Linear Tree Logic).

The paper initially extracted the microgrid operation requirements and modeling the needs using the KAOS model. It is then formally analyzed using LTL. The paper demonstrated these models through the diagrams based on the goal, object, operation, and responsibilities model. The KAOS model was shown to be very efficient, although formal presentation through LTL is very light in detail. The paper also mentioned a drawback of LTL of not being appropriated for distributed systems and suggests other formal specification languages. This paper demonstrates the potential benefit of using a formal specification techniques and reliability of the GORE approach.

CHAPTER 4

METHODOLOGY

This chapter describes the Z formalization of the UML class diagram. The evaluation of related work leads us to conclude there is a need for a verification approach for microgrids' stable operation. The methodology employed in this work will transform existing UML class

diagrams to the Z notation by applying the rules specified in the following section. I obtained the microgrid system's properties used in this study from the Electrical department [20] and UML class diagram from [5]. Further, the notation of class, class instance, and association is formally expressed in Z.

The software development process augmented by model-based software development is considered for the modeling of microgrid systems. Model-based software development provides an abstract model of the system, and simplifies the process of design[21]. Model is used to show the system components' particular view and the communication between the components of the system. The model-based software development methodology employs the UML design and deployment development in the series of deployment steps. The UML class diagram provides a nominal baseline of the system. It helps to understand the system and its requirements. The Z formalization of the UML class diagram will facilitate the microgrid system's precise and rigorous analysis. The formal notation is analyzed using Z/EVES, which ensures the model's specification is accurate and complete. I made numerous changes and revisions to the class diagram until Z/EVES did not detect any errors. I have illustrated the formalization with an example in the following section.

4.1 **Illustration of existing class diagram**

As mentioned above, in section 3.1, I used the existing class diagram for formal specifications. Figure 4.1 shows the class diagram designed. The are 7 classes, 41 attributes, and 57 functions. To express the UML class diagram formally in Z notation, each attribute, object identifiers of class, association, aggregation, and generalization are represented in Z schema. The illustration of each step is discussed in the next section.

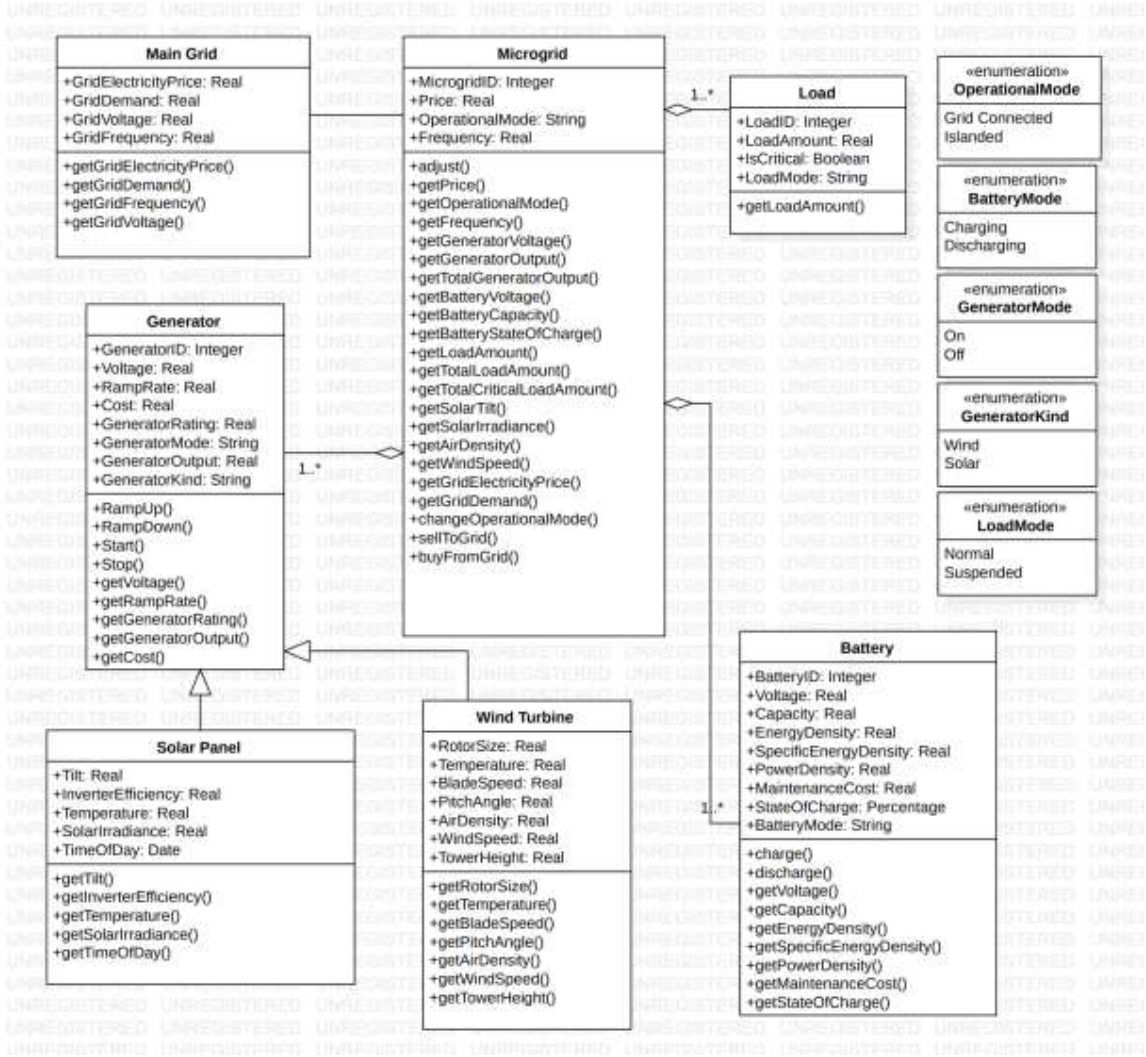

Figure 4. 1 A snapshot of initial class diagram

4.2 **Formalization of UML class diagram**

My approach to model the microgrid system started by naming the system's components and expressing the constraints of the system. I described all the system states using sets, sequences, relations, functions, and operations in pre and post conditions. Microsoft Z/EVES was the tools used for Z notation. This tool is used to analyze Z specifications, which is applied to analyze the systems schema expansions, pre-conditions, syntax, type checking, and to prove theorems. The following steps are followed to create the specification using the class “Battery” as an

example.

4.2.1 Type definition: Basic, Composite, and global variables

In Z notation, the formal specification for an equivalent class can be represented by using a schema. Z schema consists of the declaration part, in which variables are declared, and the predicate part, which consists of the constraining predicates [22]. The attributes defined in the class diagram correspond to the predicate part of the Z schema. Similarly, the data type of the attributes correspond to the Z basic types. The type definition is listed in square brackets. As an example, for the class "Battery", shown in Figure 1, is represented in Z.

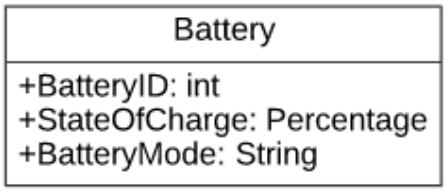

Figure 4. 2 A snapshot of battery class

Basic types: [BATTERY].
The Battery class schema represented in Z notation:

Battery

$batteryID : \mathbb{N}$
$stateofcharge : \mathbb{P}\ \mathbb{R}$
$battery\_mode : STRING$

Figure 4. 3 Representation of Z schema of battery

In the above schema, subsets of types to each type of the class diagram are assigned. As shown in the figure, "*batteryID*" is represented as N, the set of natural numbers. P is defined as a power set. "*stateofcharge*" is represented as the power set of real values. Similarly, "*battery_mode*" is represented as STRING. In the class diagram, "*battery_mode*" is described

as enumeration with literals—charging and discharging. Hence, in Z schema, the combination of colons and equality symbols are used to represent type definition as enumerated types—description of name with the possible values—for example, the statement.

BatteryMode ::= charging | discharging

From this definition, the description of "*battery_mode*" has effects on the operation. The above description of the Battery has a severe flaw since the type definition as *STRING* can accept any type of string. So, to correct the implementation, "*battery_mode"* is defined as the existing "*BatteryMode"* instances. The corrected z schema is given below.

Battery

batteryID: $\mathbb{N}$
stateofcharge: $\mathbb{P}\ \mathbb{R}$
battery_mode: *BatteryMode*

Furthermore, the Figure below shows the detailed illustration of the battery with the statements in the predicate part. As per the axiomatic description given below, "*batteryID*" consists of all the natural numbers greater than zero, where the state of charge must be between 10 and 100.

battery_attributes

batteryID : $\mathbb{N}$
stateofCharge : $\mathbb{P}\ \mathbb{R}$
batteryMode : *BatteryMode*

$batteryID > 0$
$\forall\ st : stateofCharge \bullet 10 \leqslant st \leqslant 100$

Figure 4. 4 Schema of the battery as an example

4.2.2 Representation of association and aggregation in Z

In the UML class diagram, an associations represents the relationship shared among the objects of the classes whereas an aggregation indicates a lifetime dependency among the related parts.

Figure 4.2.2 shows the aggregation structure in which microgrid is an aggregate class with a battery class.

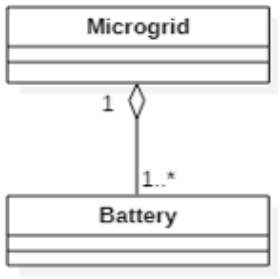

Figure 4. 5: A UML aggregation structure

The approach taken for the representation of aggregation and association followed the rules specified in [23]. The schema illustrated below represents a predicate that defines a relationship (Rel) with a domain (dom Rel), range (ran Rel), and multiplicities (mult1, mult2) of the association between the microgrid and battery classes.

Microgrid_Aggregate_Battery

microgrid: *Microgrid_Collection*
battery: *Battery_Collection*
microgrid_rel_battery: *MICROGRID* ↔ *BATTERY*

---

microgrid_rel_battery ∈ { *Rel*: *MICROGRID* ⤔ *BATTERY* | dom *Rel* ⊆
microgrid. *microgrid_instances* ∧ ran *Rel* ⊆
battery. *battery_instances* ∧ (∀*mult*1: dom *Rel* • #{ *mult*1 } ∈ 1..1)
∧ (∀*mult*2: ran *Rel* • #{ *mult*2 }≥1)}

Figure 4. 6: Schema of the aggregation structure

4.2.3 Representation of schema operation and state representation

The following schema represents the operation for the class battery. To describe the operation for the charging of the battery, I have to perform the operation based on the state of charge illustrated in figure 4.2.3. Here, "*battery_SOC"* operations specified in the problem description

are used to show that if the "state of charge" is less than 100 is to be charged and vice versa. Here is the schema that shows the specification used to determine the state of charging the battery. The declaration Δ specifies that the schema is describing a state change followed by the declaration of the inputs to the battery mode. An inputs always ends with a question mark. The schema below specifies a precondition for "*stateofCharge"*, and if the precondition is satisfied, then "*batteryMode*" is changed accordingly.

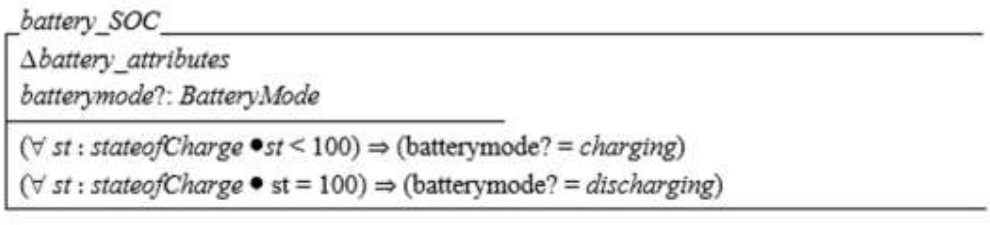

Figure 4. 7 Pre and post-condition of state of charge of the battery as an example

The specification is described in terms of relationships that have both inputs and outputs. The specification's main goal is to clearly define the correct input and handling of the incorrect input.

4.3 **Example error and modification of classes**

While creating the Z schema error checking is carried out in each step. All the tests were self-checking. I discovered numerous errors caused by improper type definitions, syntax errors, attributes conflicts, and some human errors, such as improper logical symbols, incorrect implementation of logic and a typographical errors. As an example, when converting the definition of "*batteryMode"* mentioned in section 4.2.1 resulted in an improper type definition error. A snapshot of the specification analysis is shown in Figures 5.1 and 5.2. The first figure displays the detected error due to syntax and improper data type declaration. After correcting the reported errors, Figure 5.2 shows that the formal specification is syntactically correct, and all the schema is also correct.

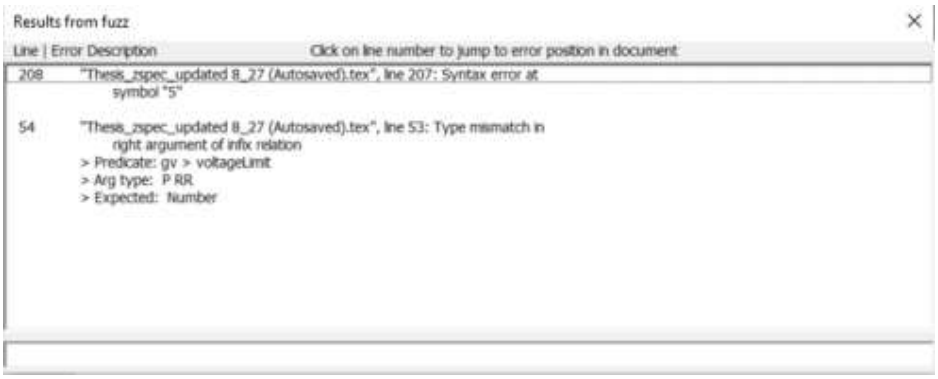

Figure 4. 8 A snapshot of error output

Figure 4. 9. A snapshot of proof correctness of the specification

Furthermore, when formalizing Z schema, it is equally important to mention the power mode (Battery, generator, or DER) of the microgrid operation. This led me to do further research into the standardization of the microgrid operation system and it's control architecture. Hence, I modified the UML class diagram using standard IEEE 1547 "IEEE Standard for Interconnecting Basic Concepts and Distributed Resources with Electric Power Systems" [3][4][24]. The resulting class integrates seven different classes: Main grid, microgrid, load, distributed generator, battery energy storage system, wind, and solar. I used the star UML, an open-source tool for modeling the class diagram. The following description is an overview of each class.

Main Grid: Microgrid is connected to the main grid to maintain voltage at the same level as

the main grid. I present the main grid to show it is connected to the microgrid, but our principal focus is the microgrid.

Microgrid: Controls the operation of the microgrid determining if it should use islanded or grid-connected mode. It receives its energy from the direct load, the energy storage system or the distributed generator.

Battery Energy storage system (BESS): Class provides the charging and discharging mode's charging capacity and provides voltage and frequency references. As mentioned in the background section, BESS plays an essential role in the operating capabilities of microgrids. BESS is used to supply power during the islanded mode to mitigate frequency disturbances. The SOC of the battery is monitored and kept in the range of 10-100%.

Distributed Generator (DG): The distributed generator is connected to the renewable energy resources in microgrids. It is responsible for generating power, and maintaining voltage and frequency within the acceptable range in islanded mode.

Solar: Solar energy is the unsteady power source generated from photovoltaic arrays from sunlight. To generate electricity from the sunlight, a solar panel has to absorb the solar radiation to generate electric current. The solar panel class helps keep the record of solar radiance. The class is also responsible for providing the current temperature and reported forecasted times of sunlight.

Wind: Wind turbines are integrated into the microgrid and are expected to provide the frequency regulation of the wind turbines. This class report the power generated from wind turbines through wind speed, pitch angle, rotor speed, and air density. The power output will be regulated according to the reported wind speed, among other factors.

4.4 Representation of class in formal specification based on the control architecture of the microgrid

As mentioned in section II, the operation of microgrid, ESS, and DG's power supply during islanded mode is closely monitored in each case. When a microgrid operates in grid-connected mode, ESS and DG's power supply is set to zero. ESS can operate as a generator in charging or discharging mode based on the standard limit of the battery's state of charge i.e., greater than 20%. The switching from grid-connected to island connected is achieved during the following cases.

- When a minor voltage fault occurs on the main grid, ESS will supply the required power to fulfill the desired capacity.
- When the total load demand is equivalent to the load generation or the capacity is larger than demand.
- When load generation and battery energy storage is equivalent to load demand.
- When the renewable energy output from solar and wind turbine generators is larger than load demand.

The following control model is closely monitored to ensure the stability of the system.

- Received input voltages for all the sources and distributed power are monitored
- The system should be able to change the configuration of the running generator to supply power load.
- Battery capacity, wind turbine status, and solar power input are monitored.
- Ensure the state of charge of the battery is within 10-100%.

To specify each operation, I considered all the operations very carefully. When the microgrid operates in grid-connected mode, the main grid will provide the microgrid's voltage and frequency references. If a fault occurs on the main grid microgrid needs to transfer to islanded mode. The sudden transition from grid-connected to islanded mode causes a power unbalance

in the microgrid system. Failure to maintain power balance will cause frequency deviation. Hence, ESS is used to improve frequency regulation. To ensure seamless transfer master-slave control method has been introduce, with the intent of ensuring that batteries with a load generator can improve the frequency regulation.

CHAPTER 5

RESULTS AND DISCUSSION

This chapter describes the results of the experiments on formal methods. It presents the experimental design, results, and verification of the formal methods. This chapter initially talks about the identified error on Z/EVES while performing formal specifications and methodology applied to correct the encountered errors. Then, for the formal specification's correctness, five research questions have been addressed in the following.

The experiment results are analyzed in turn, i.e., the class diagrams used are analyzed, recreated, designed using the formal notation in an iterative process, and then compared to each other. As mentioned in the methodology section 4.3, the class diagram was changed due to the identified error during the formalization. The class diagram contained 7 classes with 41 attributes, 7 associations, with 57 operations evaluated.

The change in the class diagram is due to not adhering to the IEEE 1547 standard for the system control architecture for integrating DERs and the errors reported from Z/EVES while performing formal specification and verification. Furthermore, Several specification errors were found during formalization, with attribute conflict in class diagrams being the most common error. Similarly, I discovered other errors including improper variable assignments, syntax errors, and inconsistent data type declarations throughout the formal specification. Hence, it led me to change the class diagram on the association between DG, BESS and microgrid, data types, and representation of the class name. Figure 5.1 shows the final UML class diagram of the microgrid system.

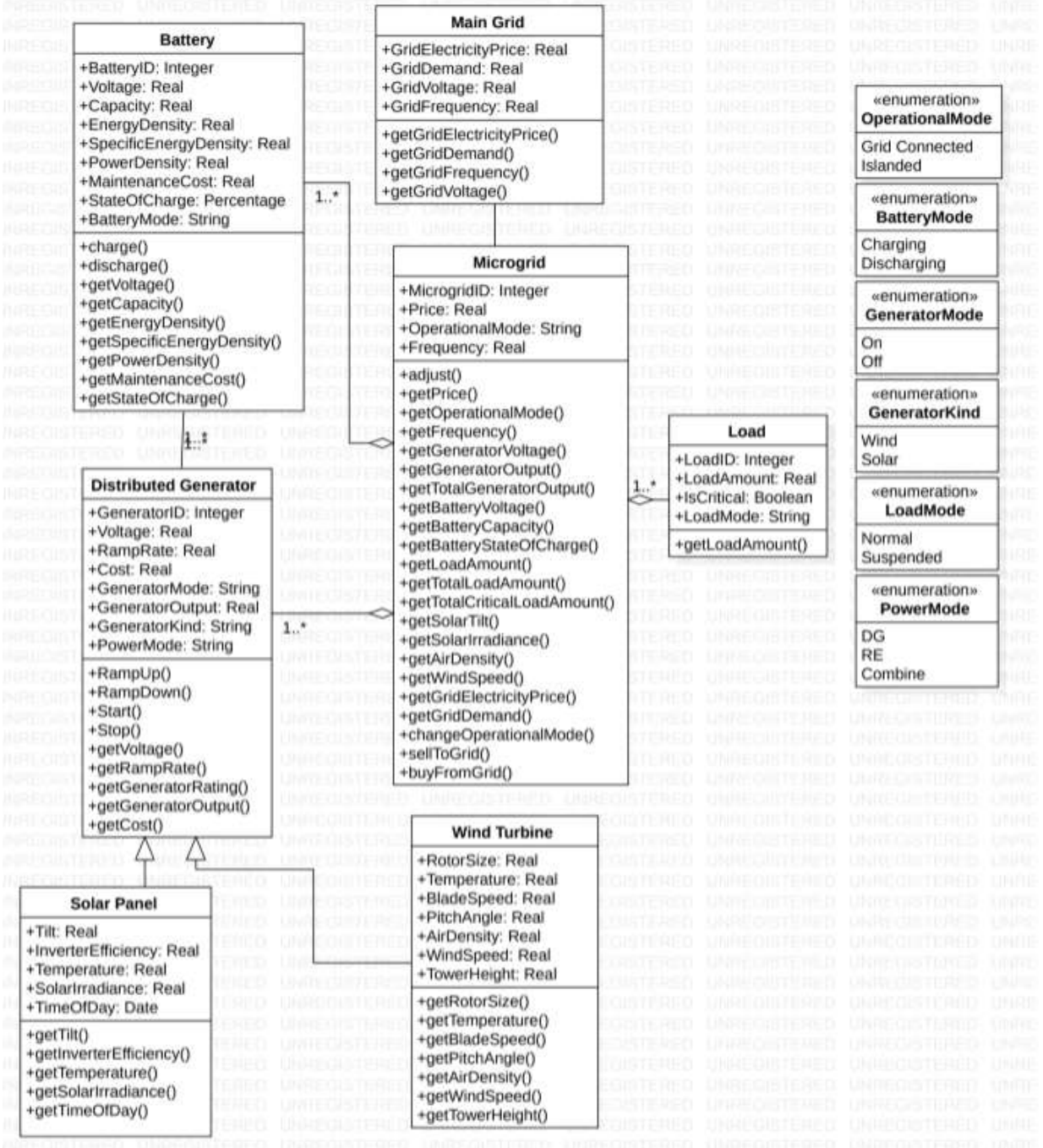

Figure 5. 1 Modified class diagram

In the Z notation, the statement of all the specifications covered 354 lines. I had 28 verification conditions, of which 7 were based on safety and functional properties. We kept close track of all fault errors that occurred while testing using Z/EVES. Some of the faults that occurred during the formalization of UML class diagrams were mostly based on improper paragraph description, improper expression in the predicate part, wrong schema specification, and human errors. The detailed specification is presented in APPENDIX I. The modeling of Z notation is based on the following table 5.1 that shows the description and significant constraints of each class.

Table 5. 1 Description and important constraints of each class.

Class	Description	Constraints
Microgrid	Facilitates the integration of DG, BESS and load and ensure the power is equivalent to load demand	Microgrid id should be a unique natural number and greater than 0.
		59<=Frequency <=60 220 <= voltage <=240
		Power source: BESS < load demand, the operation mode is changed to grid-connected else islanded
		Power source: BESS +DG < load demand, operation mode is changed to grid-connected else islanded
	Facilitates the power	Load id should be a unique natural number and

Load	and utilized to achieve the power balance and avoid peak demand during load demand is higher.	greater than 0.
		Load demand: Check for the load demand
BESS	Facilitates the integration of renewable energy sources in a microgrid to store the excess energy.	State of charge < 80%, battery will be charged
		State of charge = 100%, battery will be discharged
		State of charge< 10%, power output is discharged
		Energy Density, capacity, power Density should always be greater than zero.
Generator	Facilitates the steady and sustained power generated from wind power systems and photovoltaic system in the microgrid	Generator output = load demand
		Generator voltage and output should be greater than 0.
		DG+ BESS = load demand, act as a master controller
		DG = load demand, act as a master controller
Solar	Renewable energy sources from the sun	$18 \leq \text{Tilt} \leq 34$ $15 \leq \text{Temperature} \leq 65$

Wind	Renewable energy sources from the wind	212 ≤ Tower height ≤ 328 13 ≤ Wind speed ≤ 15 -20 ≤ Temperature ≤ 50 -40 ≤ Rotor size ≤ 90

The modeling of the microgrid system using Z notation addressed five questions:

RQ1: If the fault occurs in the microgrid on grid-connected mode, will the microgrid seamlessly transfer to islanded mode?

RQ2: If the energy capacity of ESS is limited, will the renewable energy sources provide the operation of the islanded microgrid?

RQ3: If the ESS and DG fail to operate on islanded mode, will the microgrid seamlessly transfer to grid-connected mode?

RQ4: If the battery's SOC is lower than the upper limit, will it be in the charging mode?

RQ5: If the battery's SOC is higher than the upper limit, will it be in the discharging mode?

The following notations were applied in the formal specification to identify the success and failure of the research question.

To examine RQ1, load demand, voltage, and frequency limit were closely monitored. Main Grid provides voltage and frequency references for microgrids. If the load demand is not equivalent to grid output, microgrid switches from grid-connected to islanded mode. To illustrate the notation, the figure below represents the schema and operation.

Microgrid_Adjustment

Δ*Microgrid_Attributes*
Δ*Battery_Attributes*
Δ*Generator_Attributes*
Δ*Load_Attributes*
operation_mode?: *OperationMode*
batterymode?: *BatteryMode*

(($\forall$*go*: *generator_output*; *la*: *load_amount* • *go* < *la*) $\Rightarrow$ (*batterymode*? = *discharging*))
(($\forall$*go*: *generator_output*; *la*: *load_amount* • *go* > *la*) $\Rightarrow$ (*batterymode*? = *charging*))
(($\forall$*go*: *generator_output*; *la*: *load_amount* • *go* < *la* $\wedge$ ($\forall$ *st*: *stateof_charge* • *st* < 10))
$\Rightarrow$ (*operation_mode*? = *grid_connected*))
(($\forall$*go*: *generator_output*; *la*: *load_amount* • *go* > *la* $\wedge$ ($\forall$ *st*: *stateof_charge* • *st* > 20))
$\Rightarrow$ (*operation_mode*? = *islanded*))

Figure 5. 2 Z Schema for microgrid adjustment method for class microgrid_attributes

It is essential to mention the ESS depends on the energy storage capacity and cannot operate for a longer period as it keeps discharging and eventually run out of power. BESS can only operate as the master unit for only a short period. So, the answer to RQ2 and RQ3 is the following Z notation.

Microgrid_Master_Operation

Δ*Battery_Attributes*
Δ*Microgrid_Attributes*
Δ*Generator_Attributes*
Δ*Load_Attributes*
batterymode?: *BatteryMode*
operation_mode?: *OperationMode*
PowerMode?: *PowerMode*

(*operation_mode*? = *islanded*) $\wedge$ ($\forall$ *g*: *generator_output*; *la*: *load_amount* • *go* > *la*) $\wedge$
($\forall$*st*: *stateof_charge* • *st* < 10) $\Rightarrow$ *PowerMode*? = *DG*
(*operation_mode*? = *islanded*) $\wedge$ ($\forall$ *go*: *generator_output*; *la*: *load_amount* • *go* > *la*) $\wedge$
($\forall$*st*: *stateof_charge* • *st* > 10) $\Rightarrow$ *PowerMode*? = *DGBESS*
: = DG

Figure 5. 3 Z Schema master unit operation at the same time, it is equally important to monitor the SOC of the battery, hence the following Z notation answers the RQ4 & RQ5.

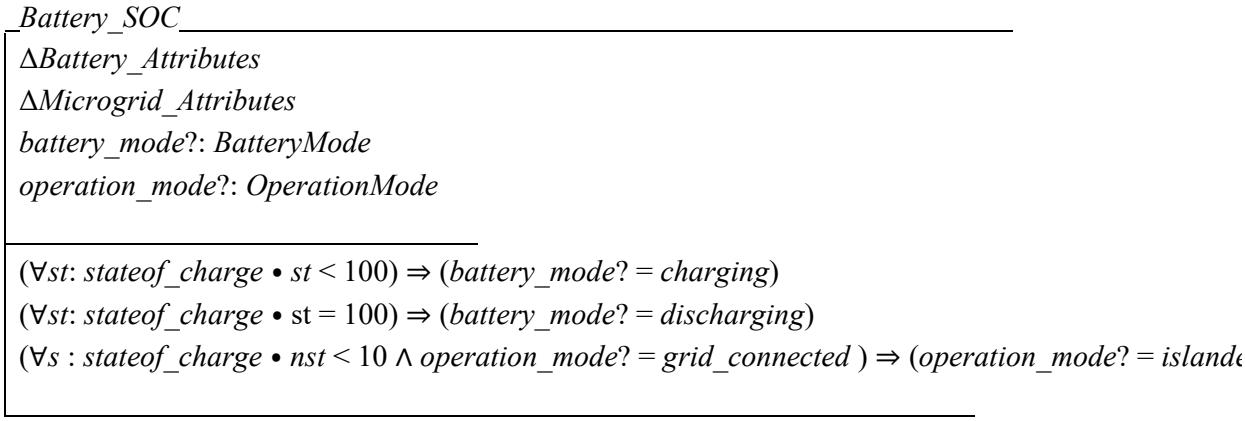

Figure 5. 4 Z Schema for state of charge of the battery

The energy capacity of ESS is limited, which is why it is integrated with distributed generators. DG and BESS, as the combined master unit, can regulate the energy and fulfill the load demand. This situation occurs when the power output from renewable energy sources, PV or wind turbines, is lower than the load demand and the SOC is higher than the lower limit.

I also noticed that the use of Z notation was the most effective process for conducting a serious analysis of the system and especially for detecting faults. However, although it addresses all of the subsystems' reasoning, it is still essential to follow the specification very closely, and future changes have to be addressed if addressed later for the specification of the system.

In the comparison, OCL and Z notation both support classes, attributes, and operations of the class, pre/postcondition, associations, and invariants. However, according to the specification language, Z supports user-defined base types, ability to combine operations (conjunction, sequential, parallel), union data type, behaviors of its individual objects and notion of system class [8]. Z notation is sequential and property oriented. It describes the input and output of the system and focuses on the properties of the system. On the other hand, OCL supports the notion of class, attribute, operation, data type, operations pre/post conditions, associations and

invariants. OCL is model oriented and provides an abstract model of the system [1]. Indeed, both specification languages address the same semantics. The specification is aimed to demonstrate the correctness of the microgrid system. Furthermore, the work initiated by Maksym Tkach [5] needs more elaboration, and detailed specifications are needed on the operation during the microgrid's islanded mode.

From the work effort, I also identified that for novices and programmers without a mathematical background that many expressions from the Z notation are complicated to understand. Simultaneously, I based the control architecture designed using Z notation on a master-slave control unit. It may require extensive communication, and will be challenging to implement in large systems as it might increase the microgrid cost. However, the formal specification helped to achieve the formal Z notation representation from the informal UML models.

CHAPTER 6

CONCLUSION

6.1 **Conclusions**

This research studied microgrid components for generating UML class diagrams and formal models. This research also presented a rigorous form of specification of the microgrid system using Z notation and ensured its correctness. This research addressed five research questions through the design, and the conclusion of the study is summarized below.

First, this research defined the microgrid system's architecture based on unclear and ambiguous requirements and the effort to use the formal specification to solve the problem. For this research, I have used the existing class diagram from [5]. The formal model was designed based on the existing class diagram which identified errors like inconsistent data types declaration, attributes conflict, and improper data types during the Z specification. Then I followed IEEE 1547 and FERC standard guidelines for the microgrid system distributed interconnections and recreated the class diagram. The microgrid control system followed master-slave control architecture, and five research questions based on energy flows among the components were answered through the formal specification.

This research also compared the formal specification of microgrids using OCL and Z notation. I demonstrated semantic consistency. OCL and Z notation is both being an independent language. It is similar to the way we study, but some of the features of OCL are not supported by Z notation and vice versa.

The specification in Z notation closely resembles the functionality of the program. The initial stage of writing the Z notation took a lot of practice, as it involves preconditions, operators, postconditions, and the algebraic notation with respective meaning to it. However, writing the specification of microgrid helped us to rigorously address the problem. Most of the errors that occurred while testing was type errors or the class definition errors.

From the results, we can conclude that formal specification is a critical factor in increasing the microgrid's reliability. Based on the model design, we can state that there is an optimal configuration to approach the microgrid system's development.

Since the microgrid is dependent on renewable energy, it can be challenging to operate microgrids smoothly, especially for island mode. However, the formal specification helped achieve rigorous specification of the microgrid system into formal representation and carried out the system design's corrective action.

6.2 **Future work**

In addition, it would be helpful to adapt to other control architecture that have been mentioned above, more specifically, hierarchy control. Microgrid control systems are highly reliant on good communication [1]. In hierarchical control, central controllers can predict the load demand, and DERs power generation and electricity prices. Hence, I would like to implement central controllers in my UML class diagram and extend the formal specification.

With regard to the class diagram, in the future, I would like to add extra DG. Meanwhile, renewable power generation from (wind and solar PV) undertake the power-sharing task, but the additional class of DG can minimize the system complexity, and I plan to study their effectiveness based on the operating condition and DG control mode. For the research question raised in this research, I want to try more coverage during formal specifications.

In the future, I would like to implement the formally specified requirements simulating the specification to analyze the efficiency of microgrid systems quantitatively. Moreover, I will be looking for the possibilities to use this specification in other real world processes.

APPENDIX I

[*STRING*]
[*PERCENTAGE*]
[*MAINGRID*]
[*MICROGRID*]
[*LOAD*]
[*GENERATOR*]
[*BATTERY*]
OperationMode ::= *grid_connected* | *islanded*
BatteryMode ::= *charging* | *discharging*
PowerMode ::= *DG* | *BESS* | *DGBESS*
GeneratorKind ::= *wind* | *solar*
LoadMode ::= *normal* | *suspended*
BOOLEAN ::= *yes* | *no*
Day == 1..31
Month == 1..12
Year == 1991..2999
Date == *Day* × *Month* × *Year*

voltage_limit: $\mathbb{P}\ \mathbb{R}$
frequency_limit: $\mathbb{P}\ \mathbb{R}$

---

voltage_limit = 220..240
frequency_limit = 59..60

day: $Date \rightarrow Day$
$month$: $Date \rightarrow Month$
$year$: $Date \rightarrow Year$

$\forall dt: Date \bullet \exists d:Day; m:Month; y:Year \mid (d, m, y) = dt \bullet day(dt) = d \land month(dt) = m \land year(dt)=y$
$\forall dt:Date \bullet month(dt) \in \{4,6,9,11\} \Rightarrow day(dt) \leq 30 \land (month(dt) = 2 \Rightarrow$
$((year(dt)\ mod\ 4 = 0 \land year\ (dt)\ mod\ 100 \neq 0) \Rightarrow day(dt) \leq 29) \land$
$((year(dt)\ mod\ 4 \neq 0 \lor year\ (dt)\ mod\ 100 = 0) \Rightarrow day(dt) \leq 28))$

$SOLAR$: $\mathbb{P}\ GENERATOR$

$SOLAR \subseteq GENERATOR$

$WIND$: $\mathbb{P}\ GENERATOR$

$WIND \subseteq GENERATOR$

Maingrid_Attributes

$grid\_electricity\_price$: $\mathbb{P}\ \mathbb{R}$
$grid\_demand$: $\mathbb{P}\ \mathbb{R}$
$grid\_voltage$: $\mathbb{P}\ \mathbb{R}$
$grid\_frequency$: $\mathbb{P}\ \mathbb{R}$

$\forall gv: grid\_voltage \bullet gv \in voltage\_limit$
$\forall f: grid\_frequency \bullet f \in frequency\_limit$

Maingrid_Classifier

Maingrid_Attributes
$maingrid\_attributes$: $MAINGRID \nrightarrow Maingrid\_Attributes$
$maingrid\_oid$: $MAINGRID$

Maingrid_OID

Δ*Maingrid_Classifier*

---

maingrid_oid' = *maingrid_oid*

Maingrid_Collection

maingrids: $\mathbb{P}$ *Maingrid_Classifier*
maingrid_instance: *MAINGRID* $\nrightarrow$ *Maingrid_Classifier*
maingrid_instances: $\mathbb{P}$ *MAINGRID*

---

maingrid_instance = {*main*: *maingrids* • *main.maingrid_oid* $\mapsto$ *main* }
maingrid_instances = *dom maingrid_instance*

Microgrid_Attributes

microgrid_id: $\mathbb{P}$ $\mathbb{N}$
price: $\mathbb{P}$ $\mathbb{R}$
operational_mode: *OperationMode*
frequency: $\mathbb{P}$ $\mathbb{R}$

$\forall m1$: *microgrid_id* • $m1 > 0$
(*operational_mode* = *grid_connected*) $\vee$ (*operational_mode* = *islanded*)

Microgrid_Classifier

Microgrid_Attributes
microgrid_attributes: *MICROGRID* $\nrightarrow$ *Microgrid_Attributes*
microgrid_oid: *MICROGRID*

Microgrid_OID

Δ*Microgrid_Classifier*

---

microgrid_oid' = *microgrid_oid*

Microgrid_Collection

$microgrids : \mathbb{P}\ Microgrid\_Classifier$
$microgrid\_instance : MICROGRID \nrightarrow Microgrid\_Classifier$
$microgrid\_instances : \mathbb{P}\ MICROGRID$

$microgrid\_instance = \{main : microgrids \bullet main.microgrid\_oid \mapsto main\}$
$microgrid\_instances = dom\ microgrid\_instance$

Load_Attributes

$LoadID : \mathbb{N}$
$load\_amount : \mathbb{P}\ \mathbb{R}$
$is\_critical : BOOLEAN$
$load\_mode : LoadMode$

$LoadID > 0$
$is\_critical = no \Rightarrow load\_mode = normal$
$is\_critical = yes \Rightarrow load\_mode = suspended$

Load_Classifier

$Load\_Attributes$
$load\_attributes : LOAD \nrightarrow Load\_Attributes$
$load\_oid : LOAD$

Load_OID

$\Delta\ Load\_Classifier$

$load\_oid' = load\_oid$

Load_Collection

$loads : \mathbb{P}\ Load\_Classifier$
$load\_instance : LOAD \nrightarrow Load\_Classifier$
$load\_instances : \mathbb{P}\ LOAD$

$load\_instance = \{load : loads \bullet load.\ load\_oid \mapsto load\}$
$load\_instances = dom\ load\_instance$

Generator_Attributes

$generator\_id : \mathbb{N}$
$gen\_voltage : \mathbb{P}\ \mathbb{R}$
$rampRate : \mathbb{P}\ \mathbb{R}$
$cost : \mathbb{P}\ \mathbb{R}$
$generator\_rating : \mathbb{P}\ STRING$
$generator\_kind : GeneratorKind$
$power\_mode : PowerMode$
$generator\_output : \mathbb{P}\ \mathbb{R}$

$generator\_id > 0$
$\forall gv : gen\_voltage \bullet gv \geq 0$
$\forall go : generator\_output \bullet go \geq 0$
$\forall c : cost \bullet c \geq 0$

Generator_Classifier

Generator_Attributes
$generator\_attributes : GENERATOR \pfun Generator\_Attributes$
$generator\_oid : GENERATOR$

Generator_OID

Δ *Generator_Classifier*

$generator\_oid' = generator\_oid$

Generator_Collection

$generators : \mathbb{P}\ Generator\_Classifier$
$generator\_instance : GENERATOR \pfun Generator\_Classifier$
$generator\_instances : \mathbb{P}\ GENERATOR$

$generator\_instance = \{gen : generators \bullet gen.generator\_oid \mapsto gen\}$
$generator\_instances = dom\ generator\_instance$

Battery_Attributes

$battery\_id : \mathbb{N}$
$voltage : \mathbb{P}\ \mathbb{R}$
$capacity : \mathbb{P}\ \mathbb{R}$
$energy\_density : \mathbb{P}\ \mathbb{R}$
$specificenergy\_density : \mathbb{P}\ \mathbb{R}$
$power\_density : \mathbb{P}\ \mathbb{R}$
$maintainence\_cost : \mathbb{P}\ \mathbb{R}$
$stateof\_charge : \mathbb{P}\ \mathbb{R}$
$battery\_mode : BatteryMode$

$battery\_id > 0$
$\forall ed : energy\_density \bullet ed > 0$
$\forall sd : specificenergy\_density \bullet sd > 0$
$\forall c : capacity \bullet c > 0$
$\forall pd : power\_density \bullet pd > 0$
$\forall mc : maintainence\_cost \bullet mc > 0$
$\forall st : stateof\_charge \bullet 10 \leq st \leq 100$
$\forall v : voltage \bullet v \in voltage\_limit$

Battery_Classifier

Battery_Attributes
$battery\_attributes : BATTERY \nrightarrow Battery\_Attributes$
$battery\_oid : BATTERY$

Battery_OID

Δ *Battery_Classifier*

$battery\_oid' = battery\_oid$

Battery_Collection

$batterys : \mathbb{P}\ Battery\_Classifier$
$battery\_instance : BATTERY \nrightarrow Battery\_Classifier$
$battery\_instances : \mathbb{P}\ BATTERY$

$battery\_instance = \{bat : batterys \bullet bat.battery\_oid \mapsto bat\}$
$battery\_instances = dom\ battery\_instance$

Solarpanel_Attributes

$tilt : \mathbb{P}\ \mathbb{R}$
$inverter\_efficiency : \mathbb{P}\ \mathbb{R}$
$temperature : \mathbb{P}\ \mathbb{R}$
$solar\_irradiance : \mathbb{P}\ \mathbb{R}$
$timeof\_day : Date$

$\forall tilt : tilt \bullet 18 \leq tilt \leq 34$
$\forall temperature : temperature \bullet 15 \leq temperature \leq 65$

Solarpanel_Classifier

$Solarpanel\_Attributes$
$solarpanel\_attributes : SOLAR \nrightarrow Solarpanel\_Attributes$
$solarpanel\_oid : SOLAR$

Solarpanel_OID

$\Delta\ Solarpanel\_Classifier$

$solarpanel\_oid' = solarpanel\_oid$

Solarpanel_Collection

$solarpanels : \mathbb{P}\ Solarpanel\_Classifier$
$solarpanel\_instance : SOLAR \nrightarrow Solarpanel\_Classifier$
$solarpanel\_instances : \mathbb{P}\ SOLAR$

$solarpanel\_instance = \{solar : solarpanels \bullet solar.\ solarpanel\_oid \mapsto solar\}$
$solarpanel\_instances = dom\ solarpanel\_instance$

WindTurbine_Attributes

$rotor\_size: \mathbb{P}\ \mathbb{R}$
$temperature: \mathbb{P}\ \mathbb{R}$
$blade\_speed: \mathbb{P}\ \mathbb{R}$
$pitch\_angle: \mathbb{P}\ \mathbb{R}$
$air\_density: \mathbb{P}\ \mathbb{R}$
$wind\_speed: \mathbb{P}\ \mathbb{R}$
$tower\_height: \mathbb{P}\ \mathbb{R}$

$\forall tower\_height: tower\_height \bullet 212 \leq tower\_height \leq 328$
$\forall\ wind\_speed: wind\_speed \bullet 13 \leq wind\_speed \leq 15$
$\forall bs: blade\_speed \bullet bs \in wind\_speed$
$\forall\ t: temperature \bullet -20 \leq t \leq 50$
$\forall\ sizemet: rotor\_size \bullet -40 \leq sizemet \leq 90$

WindTurbine_Classifier

WindTurbine_Attributes
$windturbine\_attributes: WIND \nrightarrow WindTurbine\_Attributes$
$wind\_turbine\_oid: WIND$

WindTurbine_OID

Δ *WindTurbine_Classifier*

$wind\_turbine\_oid' = wind\_turbine\_oid$

WindTurbine_Collection

$wind\_turbines: \mathbb{P}\ WindTurbine\_Classifier$
$wind\_turbine\_instance: WIND \nrightarrow WindTurbine\_Classifier$
$wind\_turbine\_instances: \mathbb{P}\ WIND$

$wind\_turbine\_instance = \{wind : wind\_turbines \bullet wind.\ wind\_turbine\_oid \mapsto wind\}$
$wind\_turbine\_instances = dom\ wind\_turbine\_instance$

Generator_Assoc_Microgrid

microgrid: *Microgrid_Collection*
generator: *Generator_Collection*
generator_rel_microgrid: *GENERATOR* ↔ *MICROGRID*

---

generator_rel_microgrid ∈{ *Rel*: *GENERATOR* ⤔ *MICROGRID* | dom *Rel* ⊆
generator. *generator_instances* ∧ ran *Rel* ⊆
microgrid. *microgrid_instances* ∧ (∀*mult*1: dom *Rel* •# { *mult*1 } ≥ 1)
∧ (∀*mult*2: ran *Rel* • # { *mult*2 } ∈ 1..1) }

Microgrid_Aggregate_Battery

microgrid: *Microgrid_Collection*
battery: *Battery_Collection*
microgrid_rel_battery: *MICROGRID* ↔ *BATTERY*

---

microgrid_rel_battery ∈ { *Rel* : *MICROGRID* ⤔ *BATTERY* | dom *Rel* ⊆
microgrid. *microgrid_instances* ∧ ran *Rel* ⊆
battery. *battery_instances* ∧ (∀*mult*1: dom *Rel* • #{ *mult*1 } ∈ 1..1)
∧ (∀*mult*2: ran *Rel* • # { *mult*2 } ≥ 1)}

Microgrid_Assoc_Load

microgrid: *Microgrid_Collection*
load: *Load_Collection*
load_rel_microgrid: *LOAD* ↔ *MICROGRID*

---

load_rel_microgrid ∈ { *Rel* : *LOAD* ⤔ *MICROGRID* | dom *Rel* ⊆
load. *load_instances* ∧ ran *Rel* ⊆
microgrid. *microgrid_instances* ∧ (∀*mult*1: dom *Rel* • # { *mult*1 } ≥ 0)
∧ (∀*mult*2: ran *Rel* • # { *mult*2 } ∈ 1..1) }

Microgrid_Assoc_Maingrid

microgrid: *Microgrid_Collection*
maingrid: *Maingrid_Collection*
maingrid_rel_microgrid: *MAINGRID* ↔ *MICROGRID*

---

maingrid_rel_microgrid ∈ { *Rel* : *MAINGRID* ⤔ *MICROGRID* | dom *Rel* ⊆
maingrid. *maingrid_instances* ∧ ran *Rel* ⊆
microgrid. *microgrid_instances* ∧ (∀*mult*1: dom *Rel* • # { *mult*1} ≥ 0)
∧ (∀*mult*2: ran *Rel* • # {*mult*2} ∈ 1..1) }

Microgrid_ChangeOperation

Δ*Microgrid_Attributes*
operation_mode?: *OperationMode*

(*operation_mode*? = *islanded*) ⇒ (*operation_mode*? = *grid_connected*) ∨
((*operation_mode*? = *grid_connected*) ⇒ (*operation_mode*? = *islanded*))

Microgrid_Adjustment

Δ*Microgrid_Attributes*
Δ*Battery_Attributes*
Δ*Generator_Attributes*
Δ*Load_Attributes*
operation_mode?: *OperationMode*
batterymode?: *BatteryMode*

((∀ *go*: *generator_output*; *la*: *load_amount* • *go* < *la*) ⇒ (*batterymode*? = *discharging*))
((∀ *go*: *generator_output*; *la*: *load_amount* • *go* > *la*) ⇒ (*batterymode*? = *charging*))
((∀ *go*: *generator_output*; *la*: *load_amount* • *go* < *la* ∧ (∀ *st*: *stateof_charge* • *st* < 10))
⇒ (*operation_mode*? = *grid_connected*))
((∀ *go*: *generator_output* ; *la*: *load_amount* • *go* > *la* ∧ (∀ *st*: *stateof_charge* • *st* > 20))
⇒ (*operation_mode*? = *islanded*))

Battery_SOC

Δ*Battery_Attributes*
Δ*Microgrid_Attributes*
battery_mode?: *BatteryMode*
operation_mode?: *OperationMode*

(∀*st*: *stateof_charge* • *st* < 100) ⇒ (*battery_mode*? = *charging*)
(∀*st*: *stateof_charge* • st = 100) ⇒ (*battery_mode*? = *discharging*)
(∀*s* : *stateof_charge* • *nst* < 10 ∧ *operation_mode*? = *grid_connected*) ⇒ (*operation_mode*? = *islande*

Microgrid_Master_Operation

Δ*Battery_Attributes*
Δ*Microgrid_Attributes*
Δ*Generator_Attributes*
Δ*Load_Attributes*
batterymode?: *BatteryMode*
operation_mode?: *OperationMode*
PowerMode?: *PowerMode*

(*operation_mode*? = *islanded*) $\wedge$ ($\forall$ *go*: *generator_output*; *la*: *load_amount* $\bullet$ *go* > *la*) $\wedge$
($\forall$*st*: *stateof_charge* $\bullet$ *st* < 10) $\Rightarrow$ *PowerMode*? = *DG*
(*operation_mode*? = *islanded*) $\wedge$ ($\forall$ *go* : *generator_output*; *la*: *load_amount* $\bullet$ *go* > *la*) $\wedge$
($\forall$*st* : *stateof_charge* $\bullet$ *st* > 10) $\Rightarrow$ *PowerMode*? = *DGBESS*
: = DG

REFERENCES

[1] Gao, David Wenzhong. Energy storage for sustainable microgrids. Academic Press, 2015.

[2] Wu, Lei, Tom Ortmeyer, and Jie Li. "The community microgrid distribution system of the future." The Electricity Journal 29.10 (2016): 16-21.

[3] IEEE Draft Standard for the Specification of Microgrid Controllers," in IEEE P2030.7/D11, August 2017 , vol., no., pp.1-42, 1 Jan. 2017.

[4] IEEE Draft Standard for the Testing of Microgrid Controllers," in IEEE P2030.8/D12, March 2018 , vol., no., pp.1-43, 1 Jan. 2018.

[5] Maksym Tkach, Application of formal specification technique to microgrid representation, Technical Report, School of Electrical Engineering and Computer Science, University of North Dakota, USA, 2020

[6] R. France and B. Rumpe, “Model-driven development of complex software: A Research roadmap,” in Proc. FOSE '07 Future of Software Engineering, Washington, DC, 2007, pp. 37-54.

[7] Harmon, Paul, and Mark Watson. Understanding UML: the developer's guide. Morgan Kaufmann, 1998.

[8] Bettaz, Mohamed, and Mourad Maouche. "UML/OCL or Object-Z?." 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS). IEEE, 2017.

[9] Spivey, J. Michael, and J. R. Abrial. The Z notation. Hemel Hempstead: Prentice Hall, 1992.

[10] Kneuper, R. (1997). Limits of formal methods. Formal Aspects of Computing, 9(4), 379–394.

[11] Bialy, M., Pantelic, V., Jaskolka, J., Schaap, A., Patcas, L., Lawford, M., & Wassyng, A. (2017). Software Engineering for Model-Based Development by Domain Experts. Handbook of System Safety and Security, 39–64.

[12] Bani-Ahmed, Abedalsalam, et al. "Reliability analysis of a decentralized microgrid control architecture." IEEE Transactions on Smart Grid 10.4 (2018): 3910-3918.

[13] G. Sugumar, R. Selvamuthukumaran, M. Novak and T. Dragicevic, "Supervisory Energy-Management Systems for Microgrids: Modeling and Formal Verification," in IEEE Industrial Electronics Magazine, vol. 13, no. 1, pp. 26-37, March 2019

[14] Wang, Farn. (2004). Formal verification of timed systems: A survey and perspective. Proceedings of the IEEE. 92. 1283 - 1305.

[15] Murali Krishna Kouluri and R. K. Pandey, "Intelligent agent-based micro grid control," 2011 2nd International Conference on Intelligent Agent & Multi-Agent Systems, Chennai, 2011, pp. 62-66

[16] Postigo, Miguel Angel Orellana, Javier Martinez, and Jose Reinaldo Silva. "Formal Requirements For Microgrid Using Kaos And Reference Architecture."

[17] Commission, I. E. et al. (2008). Intelligrid Methodology for Developing Requirements for Energy Systems, IEC.

[18] Uslar, M., Specht, M., D¨anekas, C., Trefke, J., Rohjans, S., Gonz´alez, J. M., Rosinger, C. and Bleiker, R. (2012). Standardization in smart grids: introduction to IT-related methodologies, architectures and standards, Springer Science & Business Media

[19] Brown, Greg, et al. "Goal-oriented specification of adaptation requirements engineering in adaptive systems." Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems. 2006.

[20] Akula, Shravan Kumar, and Hossein Salehfar. "Energy Management System for Interconnected Microgrids using Alternating Direction Method of Multipliers (ADMM)."

2018 North American Power Symposium (NAPS). IEEE, 2018.

[21] Basha, N. Md Jubair, Salman Abdul Moiz, and Mohammed Rizwanullah. "Model based software development: issues & challenges." Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT) 2.1 (2012): 2.

[22] Spivey, J. Michael, and J. R. Abrial. The Z notation. Hemel Hempstead: Prentice Hall, 1992.

[23] Grant, Emanuel S. "Towards an Approach to Formally Define Requirements for a Health & Status Monitoring for Safety-Critical Software Systems." Lecture Notes on Software Engineering 4.3 (2016): 169.

[24] Whittingham, S. History, Evolution, and Future Status of Energy Storage. Proceedings of the IEEE, Vol. 100, May 13, 2012.