



DISSERTATION

Adaptable Engineering Support Framework for Multi-Functional Battery Energy Storage Systems

carried out for the purpose of obtaining the degree of Doctor technicae (Dr. techn.)
submitted at TU Wien, Faculty of Mechanical and Industrial Engineering,

by

Claudia ZANABRIA

Matr.Nr.: 01528041

under the supervision of

Privatdoz. Dr. Thomas I. Strasser
Institute of Mechanics and Mechatronics (E325)
Division of Control and Process Automation

reviewed by

A.o.Univ.Prof. Dr. Wolfgang Kastner
Vienna University of Technology (E191)
Institute of Computer Engineering
Karlsplatz 13, 1040 Wien

Univ.Prof. Dr. Alois Zoitl
Johannes Kepler University
Linz Institute of Technology
Altenberger Straße 69, 4040 Linz

I confirm, that going to press of this thesis needs the confirmation of the examination committee.

Affidavit

I declare in lieu of oath, that I wrote this thesis and performed the associated research myself, using only literature cited in this volume. If text passages from sources are used literally, they are marked as such.

I confirm that this work is original and has not been submitted elsewhere for any examination, nor is it currently under consideration for a thesis elsewhere.

Vienna, am 19. November 2018

Claudia Zanabria

*This thesis is dedicated to
Mili, Ivan, Fabrizzio, Claudio y Susy.*

Acknowledgment

I would like to express my sincere gratitude to my advisor Thomas Strasser as well as to my colleague Filip Pröbstl Andrén for many fruitful and interesting discussions, for their patience, encouragement and continuous support over the last four years during the realization of this thesis.

Kurzfassung

Multifunktionale Batteriespeichersysteme gelangen zunehmend an Bedeutung, da sie nicht nur einen Beitrag zur Eigenverbrauchsabdeckung von Konsumenten, sondern auch zur Stützung des Betriebs von elektrischen Verteilnetzen beitragen können. Darüber hinaus können diese Speichersysteme auch an diversen Energiemarktdiensten teilnehmen. Üblicherweise werden die vorhin genannten Speicherdienste in zugehörigen Energiemanagementsystemen implementiert, wo oftmals zeitnah auch andere Services bedient werden müssen. Die Realisierung im jeweiligen Zielsystem wird ebenfalls anspruchsvoller, da es durch die Anzahl und Komplexität der diversen Dienste und Services zu möglichen konfliktären Zuständen kommen kann. Aktuelle Steuerungsimplementierungen dieser Speichersystems lassen eine flexible Handhabung dieser Situationen oftmals nicht zu. Da auch zukünftig eine großflächige Installation dieser Geräte im Feld erwartet wird, ist ein entsprechender Engineering- und Implementierungsansatz, der eine schnelle Anpassung an geänderte Rahmenbedingungen ermöglicht, notwendig. Aktuelle Ansätze sind nur bedingt dazu geeignet; ein integrativer Entwicklungsansatz mit zugehörigen Tools ist daher notwendig.

Die in dieser Arbeit vorgeschlagene Lösung basiert auf modernen Ansätzen aus der Informatik und des Systems- bzw. Wissensengineerings. Die Basis bildet dabei eine auf einer Ontologie beruhenden Wissensrepräsentation von Diensten bzw. Services eines multifunktionalen Batteriespeichersystems. Des Weiteren werden einschlägige Normen und Standards berücksichtigt, um eine interoperable Lösung zu gewährleisten. Ebenfalls werden Inferenzverfahren als ein wesentliches Merkmal von Ontologien zur Identifikation von potentiellen Steuerungskonflikten verwendet.

Das Hauptergebnis dieser Arbeit ist daher eine Art Entwicklungs- und Implementierungsframework für Batteriespeichersysteme, die eine rasche Umsetzung von zugehörigen Applikationen ermöglicht. Um das Potential der vorgeschlagenen Lösung validieren und mit gebräuchlichen Ansätzen vergleichen zu können, wurde eine prototypische Realisierung an ausgewählten Anwendungsfällen simulativ als auch labortechnisch umgesetzt. Die erzielten Ergebnisse sind dabei vielversprechend verglichen mit herkömmlichen Design- und Entwicklungsansätzen.

Abstract

Multi-functional battery energy storage systems are getting popular since they can offer not only self-consumption but other services that contribute to the stability of supply of power grids. Furthermore, those storages may also participate in market services, demand response, among others. They are usually implemented in corresponding energy management systems. Therefore, those systems should carry out a simultaneous operation of services and be interoperable with different other actors and devices in the domain of power and energy systems. The implementation of the referred systems becomes a challenging task since unattended couplings across services may harm their overall operation. Furthermore, current engineering approaches are not flexible enough to accept an easy and rapid integration of new components. In addition, since a massive deployment of battery energy storage system is expected in a near future, a rapid prototyping of their corresponding services becomes a requirement. The mentioned issues should be addressed during the engineering process of the energy management systems, which goes from specification to realization. Existing engineering approaches support certain stages of this development process. However, an integrated solution that fully covers the identified issues is still missing. In this context, the present work proposes an adaptable engineering framework which covers all the mentioned open issues.

The proposed solution is based on model-driven engineering and ontologies concepts. The core part of this solution is an ontology that represents a common understanding of energy management systems of storages which is aligned with domain standards. Moreover, inference procedures, an important feature of ontologies, are being used to identify inconsistencies at the design level. The resulting ontology is considered as a reference for a platform independent model to achieve a rapid prototyping of the applications under study.

The main result of this work is an engineering approach that support the implementation of an error-free and interoperable energy management system as well as a rapid and flexible prototyping of it. A prototype implementation of the approach is developed in this work to assess its effectiveness. Hence, the approach is validated on selected battery energy storage systems by performing pure software simulations and lab-based experiments. A comparison between the proposed approach and others existing shows promising results.

Contents

1	Overview	1
1.1	Motivation and Problem Statement	1
1.1.1	Problem Definition	2
1.1.2	State-of-the-Art and Related Work	4
1.2	Research Goals	6
1.3	Methodology	7
1.3.1	General Overview of the EMSOnto Approach	8
1.3.2	Modeling of Control Applications	8
1.3.3	Inconsistencies Identification	12
1.3.4	Model and Code Generation	14
1.3.5	Adaptation of the Engineering Support Framework	16
1.3.6	Proof-of-Concept Evaluation	16
1.4	Summary of Scientific Publications	18
1.5	Scientific Contributions of this Work	19
1.6	Concluding Remarks	20
2	Publications	22
2.1	Publication A	23
2.2	Publication B	56
2.3	Publication C	85
	Bibliography	110
	Curriculum Vitae	114
	List of Scientific Publications	115

Chapter 1

Overview

1.1 Motivation and Problem Statement

Currently, the main use of Battery Energy Storage Systems (BESS) is to support end-customers (industry/household) to reach self-consumption. However, in a near future BESS will play an important role in the power grid since they could also contribute to power stability and quality [1]. Besides this, those BESS may also contribute to the improvement of the hosting-capacity of the electricity grid and participate on power trading, balancing market services, etc. As a consequence, a provision of cost-effective solutions for the end-user and other services to answer stakeholders demands would require the implementation of a multi-functional BESS. The aforementioned BESS services are mainly deployed within an Energy Management System (EMS). However, they could also coexist in other systems placed on the stakeholder side [2], as shown in Figure 1.1. By following that configuration, the EMS controls the storage system and should be interoperable with a wide range of electrical components and systems such as Intelligent Electronic Devices (IEDs), Distributed Energy Resources (DERs), network operators, etc. Therefore, the interactions across different EMS related services should be handled in order to avoid unattended conflicts that may harm the overall operation of the EMS. The control architecture of the referred EMS is depicted in Figure 1.1.

Given the above concerns, it can be concluded that the implementation of such EMS is a challenging task that involves conflict, interoperability and flexibility issues. Hence, the development process of those EMS should be addressed by modern engineering approaches. In fact, existing approaches support control engineers during the development of smart grid systems [3]. At the specification stage, IntelliGrid (IEC 62559) is recommended since it introduces a methodology to document smart grid Use Cases (UC) [4]. Smart Grid Architecture Model (SGAM) and System Modeling Language (SysML) are suggested at the design phase [5]. Power System Automation Language

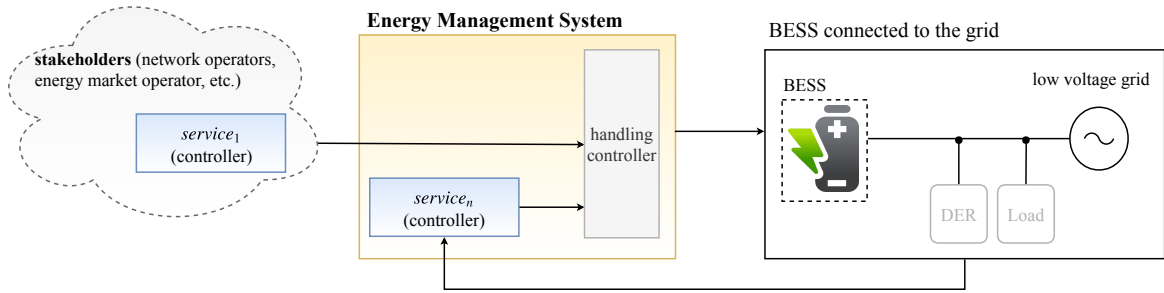


Figure 1.1: An overview of the control architecture of a multi-functional BESS.

(PSAL) is a holistic approach that supports the automation of the design and implementation phases [6]. Nevertheless, an integrated solution that overcomes the already identified issues is still missing. Hence, a suitable methodological framework that supports control engineers during the whole engineering process of multi-functional BESS applications motivates this work.

1.1.1 Problem Definition

The development process of smart grid control applications, which includes also EMS, is usually carried out by a series of steps [7] as outlined in Figure 1.2. At the specification phase, control engineers analyse the requirements and specifications to be met by the EMS. Based on that, services committed by the EMS are evaluated (e.g., frequency and voltage support). Hence, control strategies are proposed and conceptualized. This point requires an understanding of the EMS and its environment. Therefore, models of the following elements may be required: BESS, loads, metering devices, etc. Furthermore, control strategies are supported by calculations and control algorithms that conduct the behaviour of the EMS. Subsequently, an early proof of the resulting design is performed. At that stage, offline simulations of control strategies and the power system model are often executed. This involves a parametrization of electrical components, controllers, etc. The referred simulations are usually carried out in a power system simulation environment (e.g., MATLAB/Simulink, DlgSILENT/Power Factory, Modelica).

After that, the validated concept is implemented to proceed with the deployment of the solution into real-hardware components, as shown in Figure 1.2. The implementation is usually carried out in different automation platforms (e.g., IEC 61499, IEC 61131-3) and programming languages. Different software and hardware architectures are available for the validation of the developed EMS [8]. For instance, validations can be carried out in pure software, Controller-Hardware-In-the-Loop (CHIL) or Software-In-the-Loop (SIL) simulations. The selection of validation methods, tools and software platforms depends

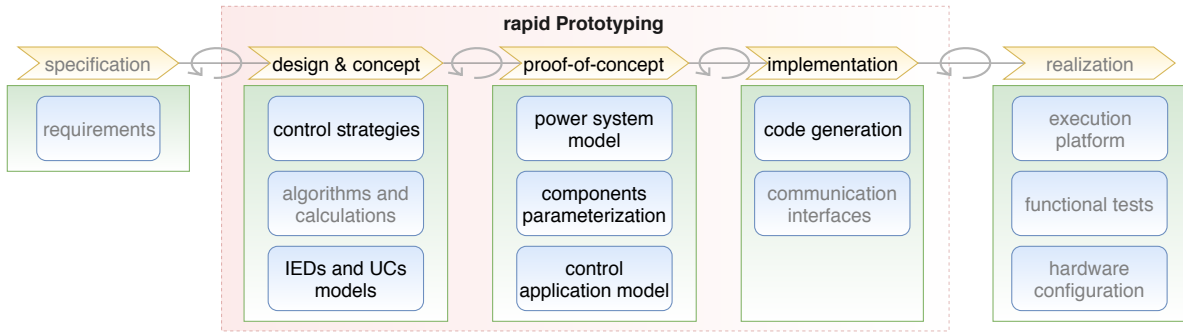


Figure 1.2: Overview of the engineering process of smart grid control applications (incl. BESS and EMS).

on control engineer practices and legacy solutions. On that basis, an investigation of the issues to be addressed in order to guarantee a flexible, error-free and automated engineering process is outlined below.

- *Inconsistencies at Design and Concept Level:* A scenario with a high penetration level of DERs together with BESS would involve the implementation of a wide range of services within an EMS (ancillary services, demand response, etc.). Those services are not only embedded within the EMS, they can also be executed in external systems under a Distributed Control System (DCS) architecture [2]. A non-coordinated interaction of the EMS related services could entail a destabilization of the EMS operation and therefore no provision of the committed services. As a consequence, couplings between services should be identified at an early stage before any further unnecessary implementation. In addition, other kind of inconsistencies such as the configuration of IED registers with wrong parameters and unit values should also be early identified.
- *Error-Prone Engineering Process:* Information regarding power system topology, DERs models, communication architecture, etc. is considered as information source to be employed along the development process. Hence, that information is used to propose a solution at design phase and also for validation purposes at proof-of-concept and implementation stages. Traditional methodologies follow a manual treatment and collection of data [9]. However, these practices may lead to potential errors due to human participation.
- *Data Interoperability:* The integration of new actors such as Electrical Vehicles (EV) and Energy Market Operator (EMO) into the distribution grid may motivate the implementation of new services within an EMS [10]. Therefore, data interoperability between the EMS and upcoming actors should be guaranteed.

Nevertheless, traditional EMS control applications are not well prepared for a rapid and easy integration of new components.

- *Vendor-Independent Solution:* EMS are implemented and validated according to specific control engineers needs. Those needs are strictly aligned to usual practices and legacy solutions. As a result, a wide range of emulators, tools, software platforms, etc. may be used [11]. For instance: power system simulators (e.g., DlgSILENT PowerFactory), communication tools (e.g., NS-3), automation tools (e.g., IEC 61131-3), co-simulation frameworks (e.g., mosaik), etc. However, a vendor-independent solution that supports the EMS implementation and validation under different software platforms and tools is still missing.

1.1.2 State-of-the-Art and Related Work

Development Approaches

Existing engineering approaches support different stages of the engineering process of EMS applications at different levels. A comparison of those approaches is carried out in [12] as part of this thesis. That study leads to conclude that a suitable approach that addresses comprehensively the previously introduced open issues is still missing. An evaluation of benefits and limitations of the compared approaches is discussed below.

The IntelliGrid methodology introduces IEC 62559 use case templates to guide control engineers during the specification of smart grid projects [13]. This involves the definition of business cases, smart grid use cases and requirements. As a consequence, IntelliGrid has been broadly employed at the specification stage. However, it is not recommended for design purposes. Instead, the Unified Modeling Language (UML) is a widely used general-purpose language to support the specification and design of software applications in an object-oriented way [14]. UML offers visual diagrams to model the behaviour and structure of software solutions. Therefore, this modeling language is getting more attention from engineers in charge of the design of smart grid solutions. UML is embedded in different tools that support corresponding code generation out of UML models. However this code is neither fully customized to engineer's needs nor oriented to EMS control applications. Alternatively, SGAM has been introduced by the Smart Grid Coordination Group/Reference Architecture (SG-CG/RA) [3] to model smart grid use cases. SGAM suggests the following layers to map use cases within a smart grid model: communication, information, function, component and information. Thereby, a good overview of the smart grid UCs under study is achieved. An implementation of the SGAM methodology is reached by a UML-based Domain Specific Language (DSL) called SGAM-Toolbox (SGAM-TB) [5]. This toolbox fosters the use of

SGAM in European and international power systems projects (FP7 ELECTRA IRP, FP7 DISCERN) [4]. Nevertheless, SGAM-TB does not propose a mechanism to check the consistency of information contained within SGAM models.

PSAL, a novel DSL for power systems was suggested at the design and implementation phase [6]. This holistic approach provides many functionalities such as rapid prototyping of smart grid applications as well as an automatic generation of IEC 61850 communication interfaces. It covers many aspects not considered by the previously mentioned ones. However, since it is designed for generic smart grid applications, concepts for EMS and BESS domains are not really addressed in a comprehensive manner. Besides of that, detection of inconsistencies within the planned design is not supported. Laboratory experiments demonstrated that PSAL is compatible with specification approaches (SGAM), a communication information model (IEC 61850) and an automation approach for distributed control systems (IEC 61499) [15]. However, the adaptability of PSAL to different environments is not yet addressed.

Rapid Prototyping

Ontologies support the alignment of related domain models as well as the integration of knowledge [16]. In this domain, an ontology matching process is implemented to achieve an alignment of two broadly recognized power utility standards Common Information Model (CIM) and IEC 61850 [17]. Besides this, Dubinin et al. [18] employs an ontology-driven approach to generate IEC 61499 applications from an ontological description of a control system. Similar results where transformation of domain models takes place can also be achieved by applying Model Driven Engineering (MDE) concepts. MDE defines mechanisms to achieve a partial automation of the engineering process [19]. An implementation of MDE practices is introduced by the Object Management Group (OMG) under the name of Model Driven Architecture (MDA).

MDA techniques are employed within PSAL for an automatic generation of executable code and also communication interfaces from SGAM UCs specifications [6]. Besides this, Andr en et al. [20] supports the generation of IEC 61499 applications out of IEC 61850 descriptions. Another research area to consider is the use of ontologies to infer information from explicit knowledge by means of reasoning mechanisms. This feature is employed in a building energy management system context to reach an automatic matching of services and communication technologies [21]. Besides this, a diagnostic of electrical faults in power transformers is carried out by ontologies in combination with fuzzy logic theory [22]. Even if none of the referred studies is multi-functional BESS oriented, they motivate the use of ontologies for inconsistencies detection.

Smart Grid Domain Standards

Standards to facilitate the interoperability across power systems are outlined in the IEC Smart Grid Standardization Roadmap [3]. Those standards cover semantic data models for EMS, metering devices, DERs, smart grid topology networks, etc. The following standards are broadly used in smart grid control applications: CIM ensures interoperability in power networks, IEC 61850 is mainly conceived to improve interoperability between IEDs. Besides that, IEC 62056 DLMS/COSEM enables the exchange of data with metering devices and IEC 62325 is focused on market communication using CIM. The selection of the standards to be considered within smart grid projects depends on stakeholders requirements, communication architectures already deployed and also the application domain (e.g., BESS).

From the mentioned concepts and research studies, it can be assumed that the combination of ontologies with MDE may conform a good candidate to conceive a new holistic, automated and intelligent approach that will answer the open issues faced by this thesis. This hypothesis is considered in the definition of the research objectives presented in the following section.

1.2 Research Goals

An assessment of the efforts carried out to address the previously identified issues leads to formulate the main research goal of this work as follows: *“To propose an adaptable and rapid prototyping framework to be used by control engineers in order to achieve error-free and interoperable multi-functional BESS control applications”*. This goal is further divided into the following three main research goals:

- *Rapid Prototyping: “To achieve a rapid prototyping of control applications to be interoperable with different components and actors of the power system domain”*. This goal motivates the development of an ontology for a common understanding of multi-functional BESS control applications. Such an ontology should be compatible with available smart grid and automation domain models. Moreover, the reinforcement of the resulting ontology with MDE techniques will be evaluated as a mechanism to reach an automated development process, as shown in Figure 1.2. This goal is mainly addressed in Publication A (see Section 2.1).
- *Identification of Conflicts: “To propose a mechanism to automatically detect the occurrence of conflicts across services related to multi-functional BESS control applications. This detection should be done at an early stage of the engineering process.”* This goal requires to investigate conflicting scenarios that would impact

the expected behaviour of EMS and therefore the non provision of committed services. Moreover, the inference of knowledge provided by ontologies will be assessed as a possible solution. This goal is covered by Publication B (see Section 2.2).

- *Adaptable Engineering Support*: “To define a methodology aimed to adapt smart grid engineering approaches to control engineers requirements and practices.” This goal implies to identify and select use cases reflecting different control engineers needs. In addition, those UCs should involve different tools, software platforms, domain standards, etc. As a result, an understanding of the flexibility needed along the engineering process is expected. The fulfilment of this goal is investigated in Publication C (see Section 2.3).

1.3 Methodology

The methodology proposed in this work should be applicable to different services derived from the penetration of BESS. An extensive list of services is presented in the report “control algorithms for storage applications in smart grids“ [1]. Besides this, it should be considered that services are still evolving due to the concurrence of new participants such as EV and EMO. Thereby, a complete and fixed list of services cannot be defined. This work selects the current most recognized BESS services to design a pattern for EMS applications and also to analyse the occurrence of conflicts. The obtained pattern supports the design of an ontology which represents a core part of this work. The selected UCs correspond to voltage control (*UC1*) [23], frequency support (*UC2*), self-consumption (*UC3*) [24], minimization of costs with peak-shaving (*UC4*) [25] and a multi-functional BESS (*UC5*) [26]. Those services are detailed in Publication A (see Section 2.1).

Ontologies and MDE concepts are the foundations of the framework proposed in this thesis, which goes under the name of EMSOnto. As a result, an adaptable, error-free and flexible engineering process for BESS control applications is reached. A prototype implementation of EMSOnto is carried out to assess its effectiveness. Hence, software simulations and lab-based tests were performed on selected use cases. A comparison of EMSOnto with other modern approaches is conducted to highlight its benefits and to identify its limitations. In the following sub-sections and in the corresponding three journal articles the proposed approach relies on an ontology called EMS-Ontology.

1.3.1 General Overview of the EMSOnto Approach

An overview of the main concepts behind EMSOnto and the benefits provided by it are depicted in Figure 1.3. The core part of the engineering support is an ontology (EMS-ontology) that conceptualizes a common representation of an EMS control application and its environments. This ontology sets the structure of a database (EMS-database) conceived to collect relevant information for the EMS design. Such information may come from software artifacts available at the specification stage or from control engineers knowledge. The whole database is checked and validated against terminological axioms contained in the EMS-ontology. The validated database is afterwards processed by a reasoner engine which performs complex axioms and logical rules in order to deduce new information. Subsequently, the inferred EMS-database is queried to extract relevant information regarding the identification of inconsistencies and also the handling of conflicts. This information is included within reports to highlight critical and severe issues to be addressed. At this stage, control engineers analyse those reports and decide improvements and modifications within the EMS design. The referred process is recursively executed in order to achieve an error-free design.

Apart from that, current smart grid standards provide device-specific models of DERs and IEDs [27]. Those kind of models and the ones that are repetitively used along the engineering process are stored within a repository, see Figure 1.3. This repository as well as the EMS-database are filled by a user-friendly mechanism based on spreadsheet templates (EMS-templates). In addition, the rapid prototyping of the whole engineering process is achieved by executing transformation rules. As a result, software artifacts can be automatically generated from the EMS-database or imported into the database. On top of that, a methodology to adapt EMSOnto to different control engineers requirements is covered by a new actor, the so called EMSOnto expert. This actor receives requirements from control engineers in order to achieve a customized EMSOnto.

1.3.2 Modeling of Control Applications

The scope and design of the EMS-ontology is addressed in this section. The EMS-ontology encompasses a representation of conflicts, information, and functional aspects of EMS control applications, as depicted in Figure 1.4. Therefore, a kind of structured approach for information representation is necessary as it is introduced by the SGAM. It recommends to structure smart grid control and EMS applications under information, function, communication, business, and component views. The scope of this work is focused on information and function domains since the other layers (e.g., communication) are already addressed by other approaches like PSAL or IEC 61499. A deeper analysis of the function domain considers that it is composed of structure

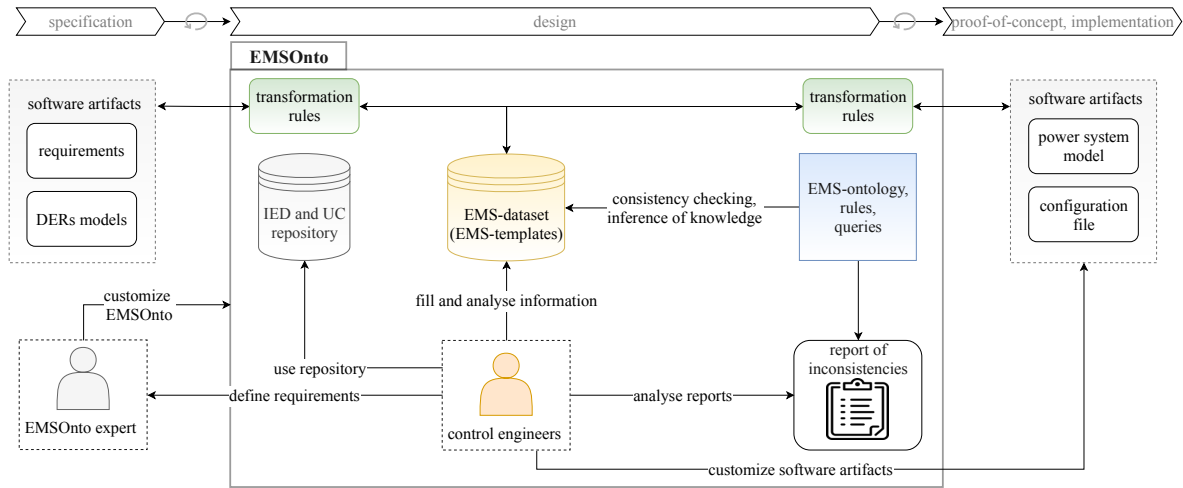


Figure 1.3: Benefits of EMSOnto along the engineering process.

and behaviour views. The behaviour view is dismissed since the design of it is considered, e.g., by UML and SysML approaches. In addition to that, the proof-of-concept, implementation and execution of functions behaviour is well supported, e.g., by MATLAB/Simulink, IEC 61499, and IEC 61131-3. Given the above factors, information and function (structure) constitute the domains addressed by EMS-Ontology. The scope and structure of EMS-ontology is outlined in Figure 1.4.

A main objective to be covered by EMS-ontology is to reach interoperability with smart grid automation approaches as well as to facilitate information-sharing towards an automation of the development process. Thus, selected domain standards are being considered in the specification of the EMS-ontology. Hence, the information view borrows concepts from an interoperability approach for IEDs, i.e., IEC 61850. Moreover, the functional view is based on reference architectures for distributed automation and control, i.e., IEC 61499, and SGAM, as highlighted in Figure 1.4. This results in an integrated domain model for BESS applications that is able to address data interoperability along the engineering process.

Function and Information Domains

An ontology is composed of a terminological component (*TBox*) that defines concepts, roles, and axioms which constraint the relations between concepts and roles [28]. Since the proposed EMS-ontology is developed from smart grid and automation domain standards, an evaluation of the concepts and roles to be borrowed from those standards is done as follows. It should be underlined that this manuscript uses the notation $\text{concept/role}^{\text{ontology}}$ to represent the membership of a concept or role into an ontology.

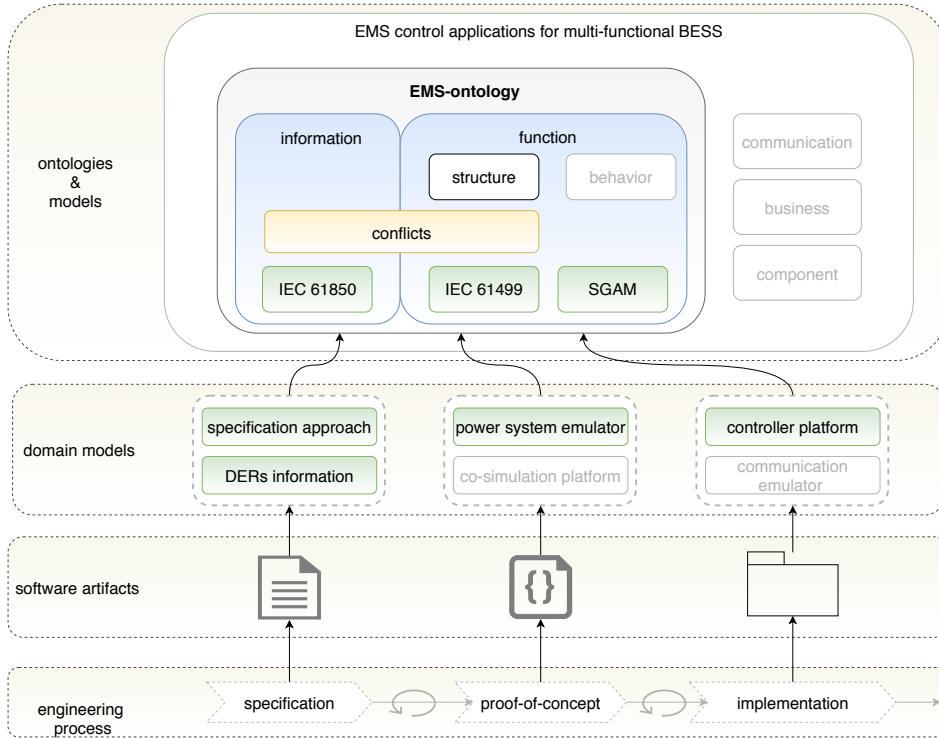


Figure 1.4: Structure and scope of the EMS-ontology which is aligned to smart grid and automation domain models.

The membership to EMS-ontology omits the upper term.

The SGAM and Smart Grid Coordination Group/ Sustainability Process (SG-CG/SP) [29] approaches are accepted by the power system community and are currently being used in different smart grid projects [4]. Therefore, concepts from those standards may be contemplated within a pattern for the structure of EMS related services. Hence, concepts such as Use Case (UC), High Level Use Cases (HLUC), Primary Use Cases (PUC), etc. are being used in the EMS-ontology. However, since the referred standards represent generic use cases, specific concepts for the EMS domain are missing. Thus, new concepts are created and added to the EMS-*TBox* which represents the *TBox* of the EMS-ontology. For instance, a taxonomy hierarchy is designed to classify HLUC. Hence, the concepts Power Quality (PQ), Power Stability (PS), Minimization of Supply Costs (MSC), etc. form the subconcepts of a HLUC.

In the automation domain, the IEC 61499 standard aims to reach interoperability within DCS by proposing a vendor-independent architecture model. This model is evaluated to support the function view representation in the proposed work. The referred standard proposes the concept System^{61499} as the root of the model, a System^{61499} encapsulates

Applications⁶¹⁴⁹⁹ conformed by a network of functions blocs (**FB**⁶¹⁴⁹⁹) that exchange data flows (**dataConnections**⁶¹⁴⁹⁹) across them. Hence, the following concepts are added to the *EMS-TBox*: **System**, **Application**, **InformationFlow**, etc.

The resulting EMS-ontology is complemented with a representation of the information view which may be supported by a wide range of information models available in the smart grid domain. This work considers IEC 61850 as baseline model since it defines IEDs and control function models relevant for multi-functional BESS. Since IEC 61850-7-3 proposes data models for run-time signals, the following concepts are made available to classify information exchanged between IEDs: **AnalogSettings**⁶¹⁸⁵⁰, **StatusSettings**⁶¹⁸⁵⁰, **ControlableStatusInfo**⁶¹⁸⁵⁰, etc. Those concepts motivate the creation of **Control**, **Status**, **Setpoint**, etc. The format of the exchanged data comprises the following concepts: **AnalogValue**⁶¹⁸⁵⁰, **BOOLEAN**⁶¹⁸⁵⁰, **STRING 255**⁶¹⁸⁵⁰, among others. Such representation leads to create the concepts **Numeric**, **Binary** and **Char**. Besides that, a taxonomy for parameters and dynamic variables is also developed to achieve parametrization of components such as controllers, IED, DERs. Therefore, **P**, **Pmax**, **SettlingTime** concepts among others are created. **P** represents active power, **Pmax** models maximum active power and **SettlingTime** is the settling time to be considered by a controller. An exhaustive list of concepts, roles and axioms within the *EMS-TBox* is presented in Publication A (see Section 2.1).

Conflicts within UCs

Until now the developed ontology cannot detect any service interactions that may harm the correct operation of EMS control applications. This point is addressed by extending the EMS-ontology as shown below.

The identification of conflicting scenarios within multi-functional BESS control applications is carried out in this work. As a result, a classification of conflicts in six categories ($C_I - C_{VI}$) is achieved. Such classification is reflected in an extended version of EMS-ontology. For instance, the conflict C_I considers at least two different PUCs offered by an EMS. The aim of each PUC is to implement control goals that manipulate states of the system under study (e.g., BESS connected to a grid). The conflict C_I arises when a control goal is affected simultaneously by at least two different PUC. For instance, a PUC seeks to minimize energy costs to be paid by end-users. Hence, the control goal behind it manipulates the power injected into the grid (P_{grid}) in order to maintain the value to zero. At the same time, another PUC requires to manipulate the value of P_{grid} to support grid frequency fluctuations. As a consequence, the referred control goals are in conflict due to a simultaneous manipulation of P_{grid} . An abstract representation of conflict C_I requires to create new concepts and roles: **ControlGoal**, **Conflict**, **Manipulate**, etc.

Besides the identification of conflicts, the handling of them is also considered within the EMS-ontology. Since handling methods requires to carry out algorithms. EMS-ontology models the dataset used by a software program destined to resolve conflicts. For the handling of conflict C_I , the setting up of priorities for each PUC is suggested. This means that only the PUC with the highest priority will operate, the others PUCs will be disable. To achieve this, definition of new concepts are also required (e.g., **Priority** and **Ena**). **Priority** represents the priority defined at PUC level and **Ena** holds the information regarding PUC activation. Concepts and roles of EMS-ontology are now established, however the expressivity of the ontology should be improved in order to achieve the deduction of new information and therefore the identification of inconsistencies. This topic is addressed in the following paragraph.

1.3.3 Inconsistencies Identification

One main objective of the EMS-ontology is to support the identification of inconsistencies. This is planned to be achieved by the inference of knowledge, an important feature provided by ontologies. Since axioms and rules are required for an efficient inference procedure, the achieved ontology is complemented with axioms within the *EMS-TBox* as well as logical rules. Besides this, another important point to be considered is the assertional component of the EMS-ontology (*EMS-ABox*) which states the participation of individuals in concepts and roles. This means that *EMS-ABox* contains assertions about the EMS control application to be designed. Such assertions are given by control engineers at the beginning of the design process, see Figure 1.5. In order to support that task, the use of spreadsheet templates (i.e., EMS-templates) are suggested. Once this task is completed, a reasoner engine executes the established axioms and rules on the *EMS-ABox* in order to come up with new assertions. The enhanced *EMS-ABox* is queried to detect inconsistencies as well as to generate additional information for the EMS design (e.g., data for the handling of conflicts). The inconsistencies detected are registered on reports, in the meanwhile relevant inferred data is added to the EMS-templates. This data may correspond to solutions for the identified conflicts as well as to other kind of inferences to be customized by control engineers. All this information is handed over to control engineers, who decide modifications and improvements within the EMS design. An overview of the referred process is depicted in Figure 1.5.

Axioms, Rules and Queries

The vocabulary conformed by concepts, roles and individuals established within EMS-ontology is not enough to represent the knowledge around the interactions of control

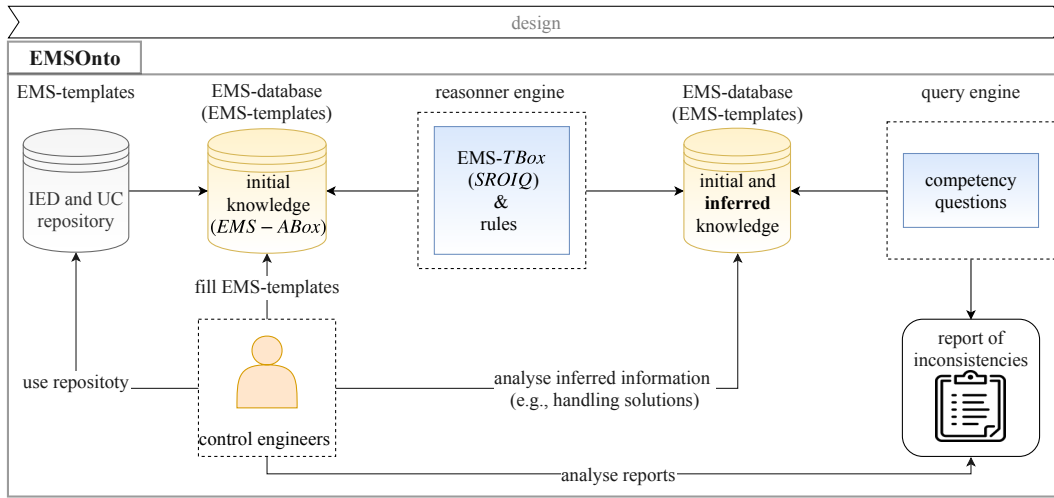


Figure 1.5: Methodology to achieve detection of inconsistencies and the inference of data.

strategies. Hence, a set of terminological axioms to constraint the relations between concepts and roles and therefore to enhance the expressivity of the EMS-ontology is required. To this end, the description language $SROIQ(D)$ is selected for a complete representation of the EMS- $TBox$ [30]. This choice is based on the role and concept constructors (composition, inverse role, qualified cardinality restriction, etc.) required by the EMS-ontology. Although, the expressivity reached with $SROIQ(D)$ is quite good, it is not sufficient for addressing the EMS-ontology requirements. Therefore, logical rules whose main purpose is the inference of dependencies among individuals are elaborated. This improves the intelligence achieved with the EMS- $TBox$. The established rules and terminological axioms are executed by a reasoner engine on the EMS-database. As a consequence, inferred knowledge is achieved resulting in an enhanced and more complete database.

Apart from that, a list of competency questions that targets the main objectives of EMS-ontology such as conflicts identification are elaborated. For instance, the detection of the conflict C_I is supported by the question: *What are the control goals that are manipulated or affected by at least two different services?*. The inferred EMS-database is consulted by a query engine in order to answer the complete list of competency questions, as shown in Figure 1.5. The obtained results are included within a report that points out inconsistencies within the planned design. Besides those reports, the query engine investigates information to be used by algorithms for conflicts handling, such information is included within the EMS-templates for further verifications.

Gathering EMS knowledge

It is worthy of note that the effectiveness of EMSOnto highly depends on the quantity and quality of the collected informations (EMS-*ABox*). Hence, the following requirements have been set as guidelines: (i) constraint the insertion of data, (ii) gathering of unlimited amount of data, (iii) user-friendly and easy to handle. Based on the stated requirements, spreadsheets templates having as headlines the names of concepts and roles defined in the EMS-*TBox* are elaborated and proposed in this work under the name of EMS-templates. Those templates implement the constraints between concepts and roles already defined in the terminological axioms of the EMS-ontology. This is achieved by using filtering, comparison, and formatting functions, among others, which are available in a large set of tools used to process tabular formats (e.g., OpenOffice, Microsoft Excel). Furthermore, since control engineers are usually familiar with spreadsheet formats, an extensive training to get familiar with the proposed methodology is not needed. Apart from that, because of their tabular format, an easy and unlimited collection of data is guarantee. In contrast, this is not facilitated by any of the existing engineering approaches (e.g., IntelliGrid, PSAL). The explained facts motivate the setting up of a repository, based on EMS-templates, which stores IED and UC models to be reused through the development process.

1.3.4 Model and Code Generation

One goal of this work is the reuse of information along the whole engineering process in an automated way to achieve a rapid prototyping of EMS control applications. A solution to that is discussed in the following.

The proposed architecture for the rapid prototyping of EMS control applications is depicted in Figure 1.6. The MDA approach introduces Platform Independent Model (PIM) and Platform Specific Model (PSM) viewpoints to represent a system. Hence, a PIM of EMS control applications is given by the EMS-data model (EMS-DM). PSMs are represented by specific power system and automation domain models (e.g., MATLAB/Simulink and IEC 61499 models). The referred EMS-DM together with PSM definitions and model transformations enable the automation of the whole engineering process. For instance, an automatic filling of EMS-templates and generation of executable code to be used for EMS validation could be achieved. In that case, the role of the control engineers is to verify the data imported within EMS-templates and to customize the generated software artifacts.

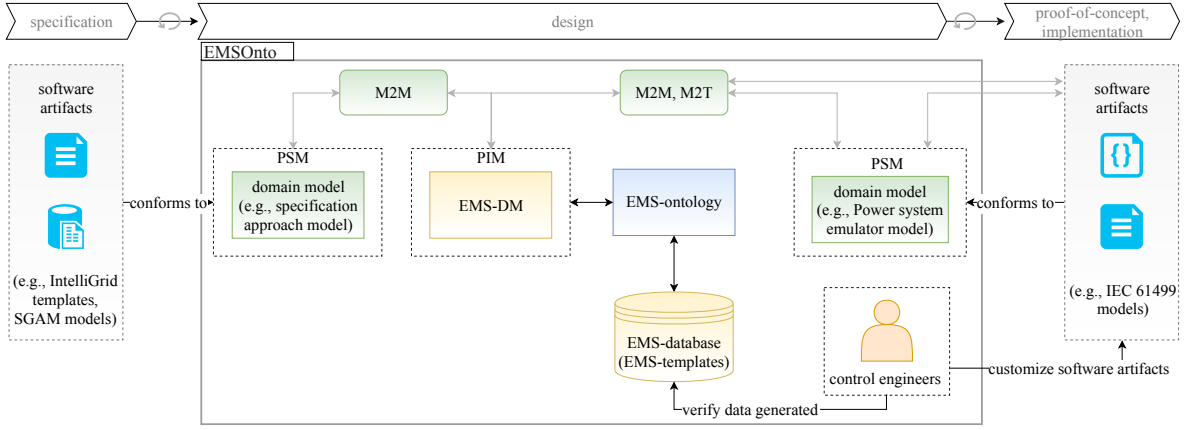


Figure 1.6: Software architecture solution to automate the EMS engineering process based on MDE and ontology concepts.

From EMS-ontology to EMS-DM

The MDA engineering approach is supported by OMG specifications such as UML and SysML. In this sense, different researches suggest the use of UML to integrate MDA with ontological engineering [31]. Therefore, this work designs a UML class representation derived from the concepts and roles established in the *EMS-TBox*. Since the referred ontology is expressed by the *SROIQ(D)* description language, a mapping between *SROIQ* elements and UML components is proposed. Hence, the following mappings are performed: $concept^{SROIQ}$ into $class^{UML}$, $role^{SROIQ}$ into $association^{UML}$, etc. The implementation of the established mappings results in a UML class representation of EMS-ontology called EMS-DM, which corresponds to the PIM model of the MDA approach. It is worth to mention that although the resulting UML model is fully aligned with the EMS-ontology, it does not represents all the knowledge abstracted by the EMS-ontology. The referred mappings together with the resulting UML model (EMS-DM) are detailed in Publication C (see Section 2.3).

Model Transformations

Divers software artifacts involved along the engineering process should be used in an automated manner towards a rapid development of BESS applications. This point is addressed by MDA techniques which are based on the setting up of PIMs, PSMs and transformation rules. Hence, M2M transformations between PIMs and PSMs are performed as well as M2T transformations to generate code from models, see Figure 1.6. In order to set up a MDA solution, an identification of data manipulated along the development process is required. This data belong to a wide range of software arti-

facts, for instance: specification documents (IEC 62559 UC templates), configuration files (IEC 61850 configuration files), controllers code (IEC 61499 applications), power system models (MATLAB/Simulink model), etc.

Subsequently, a PSM of the referred data is developed (e.g., SGAM model). Considering that the PIM is represented by the EMS-DM, mappings are established across the EMS-DM and PSMs. The main challenge here is to achieve an EMS-DM to be interoperable with different PSMs of the power system and automation domain (e.g., IEC 61850, IEC 61499 models). Since the EMS-ontology was conceived under IEC 61850, IEC 61499 and SGAM models, the EMS-DM is also compatible with those standards. Standards not considered may entail an extension of the EMS-ontology and therefore the EMS-DM. As a final step, the achievement of executable code and text is reached from the EMS-DM or the PSMs, as depicted in Figure 1.6.

1.3.5 Adaptation of the Engineering Support Framework

The features offered by EMSOnto should be customizable to different practices and needs of control engineers. Therefore, an analysis and classification of control engineers requirements is carried out. As a result, the following requirement types were identified: to provide flexibility for the generation of models and code, to benefit from a wide range of information sources available at the design stage and finally to infer new information besides conflicts detection. Those requirements are addressed by a list of procedures and rules to be followed by a new actor, the so called EMSOnto expert. Hence, suggestions to upgrade the concepts, roles and axioms of EMS-ontology as well as the process by which information is inferred are made. Moreover, other tasks encompassing MDE practices are also established. The referred list of procedures is detailed and validated on an EMS use case example in Publication C (see Section 2.3).

1.3.6 Proof-of-Concept Evaluation

The research goals introduced in Section 1.2 were addressed in this work by a modern engineering framework (i.e., EMSOnto). A prototypical implementation of it was carried out mainly by using semantic web technologies and the Eclipse Modeling Framework (EMF), a tool that implements the key aspects of MDA. This prototype enables the test of EMSOnto on different use cases scenarios. Hence, the benefits claimed by EMSOnto have been assessed on BESS control applications by pure software simulations and also by selected laboratory experiments. Such control applications are represented by a Smart Low Voltage Grid Controller (SLVGC) and a Customer Energy Management System (CEMS). The SLVGC controller executes self-consumption,

market services and voltage control, while the CEMS embeds frequency support and self-consumption services.

The effectiveness of EMSOnto with regards to detection of conflicts is measured by *F – measure* which is used in search and document classification areas to evaluate algorithms and systems performance [32]. The referred value is calculated from *precision* and *recall* measures, *precision* is the fraction of corrected identified conflicts out of the total number of identified conflicts, while *recall* represents the fraction of corrected identified conflicts out of the total number of existing conflicts. By executing inferences procedures on the SLVGC controller, 4 conflicts were detected, from there only 3 were correct and 1 conflict was not detected. This shows that the inference procedures for conflicts identification have a good performance for that particular test. That result could not be evaluated with respect to results from other engineering approaches since comparable approaches are missing. Moreover, it is clear that further tests need to be done to better estimate the performance of the proposed solution.

The rapid prototyping feature was evaluated by identifying the tasks automated along the whole engineering process. An overall picture of the automated tasks helps to evaluate the reduced amount of manual work in favour of control engineers. For this evaluation, the CEMS was implemented by using EMSOnto. As a result, the automated tasks correspond to an automatic generation of code and models for validation purposes, integration of information sources within a database, etc. Nevertheless, fully automated tasks were not achieved since control engineers intervention was necessary to integrate aspects not considered by EMSOnto such as control application behaviour and also the implementation of communication interfaces, as shown in Figure 1.6. In other words, code generated needed to be refined and tailored to achieve operational software artifacts. Apart from that, this work addresses the goal regarding adaptability of the engineering process by suggesting a procedure list to be performed by the EMSOnto expert. This procedure is tested on a new use case containing a wide range of control engineers requirements set to enhance and make easier the CEMS engineering process. As a result, new smart grid standards, software artifacts and tools were encompassed by EMSOnto. Moreover, not only conflicts issues were identified but also other kind of inconsistencies. The implementation of that use case helps to identify future needs such as the extension of the EMS-ontology with temporal logic, the use of reverse engineering techniques, the consideration of communication and component domains, among others. Future work contemplates to reach a full implementation of EMSOnto in order to test the support framework on different configurations of multi-functional BESS control applications. This would lead to better estimate the whole performance of EMSOnto as well as its limitations and further improvements.

1.4 Summary of Scientific Publications

In **Publication A** (see Section 2.1), a framework for the rapid prototyping of multi-functional BESS applications is achieved. The referred framework is meant to support control engineers during the whole development process. The cornerstone of this approach is an ontology (EMS-ontology) that reflects a common understanding of information and function domains of smart grid applications. This ontology is built from smart grid standards and is supported by MDE techniques to achieve a semi-automated engineering process. Benefits from the proposed framework include a BESS control application design compatible with broadly recognized smart grid models such as IEC 61850 and SGAM. In addition, software artifacts employed at the proof-of-concept stage can be semi-automatic generated. The population of EMS-ontology requires the collection of data from control engineers in order to achieve the referred benefits. This point is addressed by a user-friendly mechanism based on spreadsheet templates. The proposed framework is implemented in a prototype to tests its efficiency and performance. In that context, software simulations and laboratory experiments in a hardware-in-the-loop basis were performed on a selected use case example.

In **Publication B** (see Section 2.2), a framework for the detection and handling of control conflicts due to the overlapping of use cases within a multi-functional BESS application is introduced. Therefore, control schemes of those use cases were analysed with the aim to identify conflicting scenarios that would harm the BESS control application operation. As a result, a classification of conflicts scenarios and subsequently a pattern were identified. This supported the development of an ontology that by means of inferences procedures identifies the referred conflicts. That ontology is resulted from an extension of the one achieved in Publication A (see Section 2.1), the so called EMS-ontology. Apart from that, conflicts resolution mechanisms were also studied and adopted within the ontology. Pure software simulations within a power system simulator were performed to demonstrate the effectiveness of the proposed solution.

In **Publication C** (see Section 2.3), a methodology to adapt current engineering approaches to control engineers needs by means of ontologies and MDE basis is proposed. Therefore, engineers needs aiming to automate and guide their tasks along the engineering process are investigated. Those requirements are addressed by a list of procedures that intends to embrace different smart grid standards, software artifacts and tools involved within the development process. In addition, different kind of inconsistencies at the design level were also addressed. The usability of the proposed methodology is tested on the framework resulted from Publication A (see Section 2.1). Hence, a UC related to the integration of BESS into the distribution grid was implemented.

1.5 Scientific Contributions of this Work

The main scientific contribution of this work is an engineering support framework where ontologies and MDE techniques are combined to offer an error-free, rapid and flexible implementation of multi-functional BESS control applications. The achieved engineering support can also be applied to other types of smart grid control applications providing a proper tool for the massive roll-out of new solutions and components. In order to test the performance of the referred approach a prototypical implementation of it is developed. The contributions of this work are summarized as follows:

- Engineering support framework, with a firm MDE and ontologies backbone, for the rapid prototyping of BESS applications.
 - An ontology representing functional and information aspects of multi-use BESS control applications, as reflected in Figure 1.4.
 - A vendor-independent solution to increase reusability and consistency of the information used across the EMS engineering process.
 - Spreadsheet templates to easily collect and treat relevant information for the design of BESS control applications.
 - An architecture for a flexible generation of code and models based on MDE and ontologies to be applied in the power system domain, as shown in Figure 1.6.
 - A lab-based validated engineering framework.
- Inference procedures to identify and handle conflicts within BESS applications.
 - A classification of conflicts arisen from the interaction of services embedded within multi-functional BESS control applications.
 - An ontology that supports the identification and handling of conflicts.
 - Logical rules, terminological axioms and queries that support the inference of information.
 - A validated approach through power system simulations.
- Procedures and rules for adapting EMSOnto to new requirements and domains.
 - The proposed methodology can also be applied to other existing engineering approaches of the smart grid domain (e.g., PSAL).
 - EMSOnto is now able to encompass different smart grid standards, tools and software artifacts.
 - A validated procedure through the implementation of a control application for BESS.

1.6 Concluding Remarks

In order to show the benefits of EMSOnto with regard to other modern approaches a comparison study was carried out. This study evaluates how suitable is an approach based on the examination of delivered functionalities through the engineering process and SGAM layers. The referred comparison was previously published in [12] as part of this thesis. Slight modifications were taken to contemplate EMSOnto functionalities such as inference of information and adaptability. Moreover, only design, proof-of-concept and implementation stages were evaluated.

The criteria considered for this comparison study are detailed as follows. At the function level, the provision of semantic models for control strategies, templates for UCs, and the possibility of detailing function behaviour by graphical representation (e.g., dataflow) are examined as well as the detection of inconsistencies in the design. At the information level, mechanisms to document inputs, outputs and parameters and also the provision of semantic data models for information are evaluated. On top of that, procedures for an easy collection and treatment of data were considered. Besides that, the implementation of control strategies for an early proof-of-concept as well as rapid prototyping mechanisms are assessed. The obtained results are depicted in Table 1.1.

Table 1.1: Comparison of approaches and tools.

Phase Approach	Design					Proof	Implementation	
	function	inform.	comm.	comp.	inconst.	function	function	rapid prot.
<i>UML</i>	☹	×	×	×	×	×	☹	×
<i>SysML</i>	☹	☹	×	×	×	×	☹	×
<i>IntelliGrid</i>	☹	×	×	×	×	×	×	×
<i>SGAM-TB</i>	☹	×	☹	☹	×	×	☹	☹
<i>PSAL</i>	☹	☹	✓	✓	×	✓	✓	☹
<i>MATLAB</i>	☹	☹	×	☹	×	✓	✓	☹
<i>IEC61499</i>	☹	☹	✓	✓	×	✓	✓	×
<i>IEC61131-3</i>	☹	☹	✓	☹	×	✓	✓	×
<i>EMSOnto</i>	☹	✓	×	×	✓	✓	✓	✓

Legend: not supported at all (×), not recommended (☹), supported but not totally (☹), and well supported (✓).

This comparison demonstrates that EMSOnto stands out from other modern approaches at almost all the assessed features. In fact, EMSOnto is the only one that identifies inconsistencies within the planned design. Moreover, it is a good candidate for the rapid prototyping of EMS control applications. Furthermore, EMSOnto stands out from the others since it also provides a methodology to customize the engineering process. However, it is not suggested for conceptualizing communication and components aspects of control applications. In contrast, those requirements are covered by PSAL and IEC 61499. Moreover, at the function level, there is not a best approach to be suggested. For instance, EMSOnto does not provide a graphical representation of the function behaviour. On the contrary, that aspect is well supported by UML. In conclusion, it is recommended to integrate different approaches to fully support the engineering process. Furthermore, the selection of the appropriate approaches depends on control engineers needs and practices as well as legacy solutions, among others factors.

Chapter 2

Publications

List of Publications

Publication A

C. Zanabria, F. Prörtl Andrén , J. Kathan, and T. I. Strasser.

Rapid Prototyping of Multi-Functional Battery Energy Storage System Applications.

Applied Sciences, vol. 8, no. 8, 32 pages, 2018.

Publication B

C. Zanabria, F. Prörtl Andrén , and T. I. Strasser.

An Adaptable Engineering Support Framework for Multi-Functional Energy Storage System Applications.

Sustainability, vol. 10, no. 11, 28 pages, 2018.

Publication C

C. Zanabria, A. Tayyebi, F. Prörtl Andrén , J. Kathan, and T. Strasser.

Engineering Support for Handling Controller Conflicts in Energy Storage Systems Applications.

Energies, vol. 10, no. 10, 24 pages, 2017.

2.1 Publication A

C. Zanabria, F. Pröbstl Andrén , J. Kathan, and T. I. Strasser.

Rapid Prototyping of Multi-Functional Battery Energy Storage System Applications.

Applied Sciences, vol. 8, no. 8, 32 pages, 2018.



Own contribution

The problem analysis, definition of research goals, and conceptualization of the methodology were carried out by the applicant. In addition, the applicant implemented a research prototype of the proposed solution and carried out all the corresponding validation tests. A comparison study of the proposed approach with existing ones is also done by the applicant as well as the structuring, writing and editing of the manuscript. The second, third and last author proofread the manuscript and contributed in the problem analysis. In addition, the second and last author participated in the conceptualization of the methodology and conducted the supervision of the overall work.



Article

Rapid Prototyping of Multi-Functional Battery Energy Storage System Applications

Claudia Zanabria ^{1,*} , Filip Pröbstl Andrén ¹, Johannes Kathan ¹ and Thomas I. Strasser ^{1,2} 

¹ Center for Energy–Electric Energy Systems, AIT Austrian Institute of Technology, 1210 Vienna, Austria; Filip.Proestl-Andren@ait.ac.at (F.P.A.); Johannes.Kathan@ait.ac.at (J.K.); Thomas.Strasser@ait.ac.at (T.I.S.)

² Institute of Mechanics and Mechatronics, Vienna University of Technology, 1040 Vienna, Austria

* Correspondence: claudia.zanabria@ait.ac.at

Received: 18 July 2018; Accepted: 4 August 2018; Published: 8 August 2018

Abstract: Battery Energy Storage Systems (BESS) are starting to play an important role in today's power distribution networks. They provide a manifold of services for fulfilling demands and requests from diverse stakeholders, such as distribution system operators, energy market operators, aggregators but also end-users. Such services are usually provided by corresponding Energy Management Systems (EMS). This paper analyzes the complexity of the EMS development process resulting from an evolving power utility automation. As a result, flexibility, complexity, interoperability, and overlapping issues were identified as main concerns to be faced during the design and implementation stages of BESS control applications. Current efforts from smart grid and power utility automation standards partially tackle the issues mentioned above. Nevertheless, an integrated methodology that guides and supports control engineers during the whole development chain is still missing. Hence, the conception of EMSOnto is introduced. The main achievements of this approach include the alignment of BESS design with broadly accepted smart grid standards (i.e., IEC 61850, smart grid architecture model), a common understanding of EMS control applications based on the conception of an ontology, the identification of conflicts within a multi-function BESS and the semi-automatic generation of software artifacts mainly important for EMS validation. To demonstrate the effectiveness of the approach, a selected use case example is designed and validated in a hardware-in-the-loop basis. This proves that EMSOnto eases the work of control engineers resulting in a flexible, adaptable, and error-free EMS design. In addition to this, limitations of EMSOnto as well as future work are grasped.

Keywords: battery energy storage systems; rapid prototyping; conflicts identification; power utility automation; power distribution grid; semantic web technologies; ontology; description logics; model-driven engineering; smart grid architecture model; IEC 61850

1. Introduction

The reduction of greenhouse gas emissions motivates a high penetration of renewable Distributed Energy Resources (DERs). However, this may negatively influence the power quality (i.e., frequency and voltage) and surpass the hosting capacity of the corresponding power distribution grids. Battery Energy Storage Systems (BESS) are becoming an important actor in the power utility grid due to their flexibility and support that they offer. Use Cases (UCs) based on BESS participation are considered in different studies [1,2], attempting to contribute to voltage and frequency regulation. Other services such as improvement of the hosting capacity and peak-shaving are also possible [3]. Additionally, BESS supports Energy Market Operators (EMO) to benefit from the spot market prices [4] and the end-user to minimize their energy costs [5]. The implementation of the mentioned UCs implies an efficient use of the Information and Communication Technology (ICT) infrastructure and the integration between various stakeholders and DERs.

Energy Management Systems (EMS) carry out the integration of BESS UCs. Thus, they require to reach a high degree of flexibility and adaptability. Furthermore, their design becomes quite complex since different power utility equipment and stakeholders are involved. Additionally, due to the offering of many services, a coordination and alignment of control strategies is required. Those issues among others are faced during the development process of EMS. Motivating the conception of a common vocabulary to be approved by different control engineers dedicated to EMS implementation.

Existent approaches support control engineers during the engineering process; Systems Modeling Language (SysML) and Smart Grid Architecture Model (SGAM) are highly recommended at the specification stage [6]. Power System Automation Language (PSAL), a Domain Specific Language (DSL) for power systems, automates the design and implementation of smart grid applications [7]. Besides this, a power utility automation standard called IEC 61850 models DERs functionalities [8]. The implementation of them is driven by automation standards such as IEC 61499 [9] and IEC 61131-3 [10]. On this basis, this paper analyzes current smart grid and automation approaches and points out the lack of an integrated framework and methodology that guides control engineers during the design and implementation phase of EMS applications. As a solution, this paper proposes EMSOnto, a framework focused on interoperability, flexibility, complexity, and overlapping issues raised during BESS UCs implementation. Current efforts to handle those issues are evaluated, hence a set of four requirements are identified which provides the basis for the EMSOnto conception.

The core part of the EMSOnto approach is an ontology (i.e., EMS-ontology) that models different aspects of a multi-functional BESS such as potential conflicts between UCs, variables exchanged across the EMS communication architecture, structure of control strategies, etc. This modeling process is aligned with domain-specific approaches like IEC 61850 and SGAM. PSAL proposes a data model for power systems also based on SGAM and IEC 61850, but concepts for the abstraction of BESS services were not tackled. A main benefit of ontologies is to infer new knowledge from explicit knowledge. This benefit is key in the identification of conflicts, a feature supported by EMSOnto and not contemplated by any of the previous mentioned approaches. In addition to this, a friendly method to gather information from control engineers based on spreadsheet templates (i.e., EMS-templates) is exposed. This mechanism enables the population of the EMS-ontology. IntelliGrid enables the gathering of data by UC templates, however they are not suitable for collection of static and dynamic variables presented in an EMS. On the other hand, Model-Driven Engineering (MDE) is employed to semi-automatically generate software artifacts to be deployed in power system and automation tools. The rapid prototyping of smart grid applications by means of MDE techniques is not a completely new topic since it was already tackled by PSAL and other works [11]; however, the benefits of an automatic generation of software artifacts for the proof-of-concept of BESS applications is not yet addressed.

This paper is structured as follows: Section 2 presents UCs mainly important to BESS support, their corresponding development process are analyzed resulting in the identification of issues to be target by EMSOnto. Section 3 addresses those issues by analyzing current efforts carried out to handle them. From this evaluation, gaps and open issues are identified resulting in the statement of four requirements to design EMSOnto. Section 4 presents the core of EMSOnto, an ontology mainly designed to meet specific requirements. Additionally, a friendly method to populate the ontology is addressed by spreadsheet templates. As a sequel, Section 5 designs and implements a selected UC following the EMSOnto basis. This enables the identification of gaps and guidelines to improve EMSOnto. Finally, Section 6 summarizes and concludes this work.

2. Multi-Functional Energy Storage System Application Development

This section presents a list of selected UCs that address demands from various stakeholders. A correct operation of those UCs requires the follow up of an engineering process. This involves specification, design, implementation, and realization stages. Furthermore, the whole development chain of an EMS realization is also exposed. In that context, important issues to facilitate the control engineer's work and also to reach a flexible and interoperable EMS are raised.

2.1. Use Cases and Applications

The selection of use cases UC1–UC5 is based on common services provided mainly by small scale energy storage systems ($\sim > 3.6 \text{ kW}$ & $< 4.8 \text{ kW}$), see Table 1. An extended list can be found in the study [4] where a categorization by objectives, such as reduction of energy costs, power system stability, and market integration is carried out. A scheme that represents electrical devices connection and communication links, in the frame of those UC, is depicted in Figure 1. A BESS is integrated within the low-voltage grid to mainly support the user to minimize energy costs. This is reached by storing the Photovoltaic (PV) generated energy not consumed by the local load. This mechanism corresponds to UC3 where the customer is the main benefactor. However, those batteries are able to provide support to the grid operator for power system stability purposes. Thus, not only self-consumption (UC3) is pursued but other services such as voltage and frequency regulation (UC1–UC2). Following that premise, the study [5] is also selected (UC4); it defines a multi-functional system embedded with peak-shaving and reduction of price services. An ensemble of storage services is also met by UC5 [1]. That case study gathers voltage control and two other services, it is an appealing example since each service rents one part of the whole capacity of a medium scale BESS. At first glance, it might seem that services do not need to be aligned because of the capacity allocation procedure. However, study [1] analyzes conflict issues regarding a full provision of those services. Since conflict identification is a main requirement to be covered by the EMS-ontology, UC5 is included in the selection of UCs.

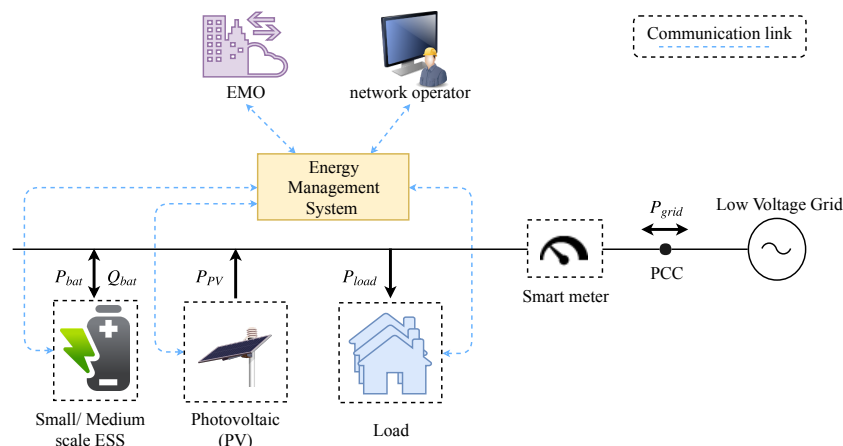


Figure 1. EMS setup with corresponding communication architecture.

Table 1. Use cases derived from the integration of BESS into a low voltage grid

Use Case, Controller Type	Description
Volt-VAr mode (UC1)	Reactive power from the battery (Q_{bat}) is injected to stabilize the voltage at the Point of Common Coupling (PCC), see Volt-Var mode VV1 in [12].
Frequency-watt (FW) (UC2)	Active power from the battery (P_{bat}) supports the balancing of frequency of the grid (F_{grid}), see FW mode FW21 in [12].
Self-consumption (UC3)	P_{bat} helps to minimize the energy consumed from the grid (P_{grid}), see PVBat1 control strategy in [13].
Min. of costs with peak shaving (UC4)	Electricity costs are reduced and peak-shaving is pursued, see [5].
Multi-functional BESS medium scale (UC5)	Self-consumption is provided to a group of households by injecting active power (P_{bat}), Primary Control Reserve (PCR) participation follows the logic in UC2. Voltage is controlled by reactive power provision Q_{bat} ; to accomplish this, a Proportional Integral (PI) controller is designed as exposed in [1].

2.2. Specification and Development Process

The realization of EMS control applications requires following certain stages, which involves specification, design, proof-of-concept, implementation, and deployment [14]. Those steps are addressed in this section.

At the specification stage, the stakeholders, DERs, and Intelligent Electronic Devices (IEDs), involved in the system under study, are identified. In addition to this, the communication and physical architecture are suggested. This results in a list of requirements to be addressed during the design phase. Subsequently, at the design level, solutions to define the specific behavior and structure of control strategies embedded by an EMS are evaluated. Hence, a list of variables encompassing control, setpoint and measurement signals are properly described. This description also gathers information related to protocols communication. Thus, the communication interfaces that allow interconnection across the power distribution application are also specified. Furthermore, the allocation of control strategies through the ICT hardware infrastructure is determined as well.

The validation of the proposed control strategies is carried out at the proof-of-concept phase. At that stage, models of IEDs (e.g., smart meters) and DERs (e.g., BESS) are built and deployed into a power system emulator (e.g., MATLAB/Simulink, DlgSILENT PowerFactory), along with the control logic of the EMS [15]. The transition from design to proof-of-concept may entail communication and stability issues that were not considered at the design stage. Hence, an iterative refinement of the control algorithms are considered. Afterwards, the deployment of the validated control algorithms takes place at the implementation phase. A programming language compatible with a specific hardware controller is chosen to deploy the logic, usually this is driven by automation standards such as IEC 61499 [9] and IEC 61131-3 [10]. The validation of implementation phase is carried out by real-time simulations, controller-hardware-in-the-loop, and/or laboratory-based tests [16].

2.3. Open Issues

A closer analysis of the abovementioned design and engineering process results in the following issues which need to be addressed:

- *Flexibility*: The integration of new and different actors into the power distribution grid such as Electrical Vehicles (EV) and EMO motivates evolving services focused on BESS participation. Thus, EMS should be flexible enough to implement those services and future ICT requirements.
- *Complexity*: The complex-nature of EMS is based on the manifold of services provided by BESS and the deployment of them in the ICT and power system infrastructure. The implementation of such services may be centralized or decentralized. In case of a decentralized configuration, where services are deployed in different hardware controllers, more than one control engineer may be required for the EMS design. Thereby, the lack of a common vocabulary and design language may lead to errors during EMS design and later on operation in the field.
- *Overlapping*: An EMS focused on multi-services for BESS is vulnerable to conflicts between UCs. This overlapping across them could cause a non-expected function behavior. For instance, an EMS that operates UC2 and UC3 would need to setup mechanisms to align both use cases in the event of a simultaneous operation. Otherwise, those UCs would not be attained.
- *Interoperability*: The realization and validation of smart grid and BESS-based applications entails the use of a wide range of power system, automation and communication tools. This goes from power system emulators/design, network simulators to co-simulation frameworks. However, a method to interoperate those tools during the whole chain of the engineering process is missing.

3. Related Work and Background

Issues which need to be tackled during the engineering process were previously highlighted. In this section, the state-of-the-art and related work are analyzed and discussed. Necessary requirements for improving the status-quo are identified, which provide cornerstones for the rapid development of UCs and applications.

3.1. Smart Grid Domain Standards

The main concern of the current study is to employ information models from recommended international standards to derive an interoperable and flexible solution. Data models for the power system domain are outlined in the IEC Smart Grid Standardization Roadmap [17]. On this basis, a collection of standards mainly important to DER involved in BESS UCs has been gathered, resulting in the following list:

- *IEC 61970/IEC 61968 Common Information Model (CIM)*: Its main concern is to ensure interoperability across power networks, [18]. Thereby, it offers a common semantic for EMS, Supervisory and Control Data Acquisition (SCADA) and power system topology. This is formally represented by a Unified Modeling Language (UML) model, including over 1300 classes.
- *IEC 61850*: It is mainly conceived to improve interoperability between IEDs in a power system substation; nowadays, it has been extended to DERs and power utility components among other areas [8]. Functional aspects of an IED are mapped into a Logical Node (LN). They are referenced in the standard IEC 61850-7-4 and extended by IEC 61850-7-420 [19], where functionalities of DERs such as inverters and batteries are modeled.
- *IEC 62325*: It provides a set of standards related to market communication using CIM, covering data models for market participants and market operators.
- *IEC 62056*: It applies mainly to smart home and data exchange for meter reading.
- *Open Platform Communications Unified Architecture (OPC UA)*: Since the EMS should support the exchange with other IEDs distributed along the power utility grid, the consideration of information models from the automation domain such as OPC UA is relevant. It is a standard of OPC foundation that focuses on exchanging real-time data within the process automation. It is platform-independent, thereby not tied to one operating system. It is standardized under IEC 62541, part 5 of that standard defines the information model (IEC 62541-5) [20].

3.2. Smart Grid Control Application Development Approaches

The development of smart grid control applications has been supported by a collection of approaches. Those are characterized by domain standards (SGAM, CIM, UML, etc.), well recognized tools (MATLAB/Simulink, etc.) and frameworks (PSAL, etc.). A comparison that measures the applicability of them through the design, proof-of-concept and implementation phase is carried out in this section. A brief introduction to the approaches is firstly described below:

- *Unified Modeling Language*: UML is a general-purpose language for an object-oriented software development [21]. It has been standardized by the Object Management Group (OMG) and approved as an ISO standard. UML specifies behavior and structural diagrams to model software applications. Thereby, it has been used in the power system domain to model use cases [22].
- *System Modeling Language (SysML)*: SysML is conceived as an extension of a subset of UML. It is mainly designed to support the system engineering process (i.e., specification, design). Thereby, elements in UML not required by systems engineering were excluded to conceive SysML [21].
- *IntelliGrid (IEC 62559)*: Intelligrid introduces a methodology to document and detail smart grid UCs. From this, a UC template [23] that covers best practices to describe requirements engineering is originated. In those templates, a UC may be described by a visual representation (e.g., UML diagram). IEC 62559 is largely accepted by the power system community as an example the adoption of it by the ELECTRA project [22].
- *Smart Grid Architecture Model*: SGAM is introduced by the SG-CG/RA (Smart Grid Coordination Group/Reference Architecture) [24]. SGAM is conceived to model smart grid uses cases architecture in a technology-neutral manner. It consists of five interoperability layers, those layers are matched to a smart grid plane composed of domains and zones. The domains cover the electricity conversion chain (i.e., transmission, distribution, etc.). The zones cover the automation pyramid (i.e., process, field, etc.). An implementation of SGAM is conceived in [25] as a plugin of the Sparx Systems Enterprise Architecture tool, it is called SGAM-Toolbox (SGAM-TB).

- *Power System Automation Language*: PSAL is a DSL to model SGAM compatible use case specifications. It supports control engineers during the whole development process, from use case design to implementation and deployment. In addition to this, PSAL offers executable code and communication configuration file generation [7].
- *MATLAB/Simulink*: This tool provides libraries to model and simulate electrical power systems. C-code generation is also contemplated for a rapid prototyping of MATLAB/Simulink models. Since this tool is largely accepted and employed in the power system domain, it is considered as a worthy approach to be analyzed along the control application development process.
- *IEC 61499*: This approach provides a reference model for Distributed Control Systems (DCS) in the domain of industrial processes. A main objective is to reach portability, configurability and interoperability across DCSs [9].
- *IEC 61131-3*: IEC 61131 is an IEC standard supported by Programmable Logic Controller (PLC). One of the main goals of IEC 61131 is to standardize the programming approaches. Thus, the part IEC 61131-3 offers graphical and textual programming languages such as function block diagram and structured text among others [10].

A comparison of the aforementioned approaches is outlined in [6], covering mainly specification and design stages. A brief discussion of features analyzed at the design stage is addressed in the following. At the function level, data models for UCs and graphical representations to detail the structure and behavior of control applications are considered. Semantic data models of information within control applications and DERs devices are evaluated at the information level. In turn, mechanisms for representing communication architectures and the allocation of logical components across hardware devices are assessed at the communication and component level.

The referred comparison study highlights that, in the specification phase, the outstanding approaches are SysML and SGAM; however, they are not recommended at the design phase. In turn, PSAL is recommended at the design stage. Nevertheless, the referred study does not consider overlapping and complexity issues important to a multi-functional BESS implementation. Those issues are addressed in the present study resulting in Table 2. This analysis demonstrates that there is no preferable candidate that covers function and information domain at the design phase.

Execution of control strategies (i.e., function domain) at the proof-of-concept and implementation stages are well supported by IEC 61499, IEC 61131-3 and MATLAB/Simulink. Rapid prototyping is a feature not well-exploited by all the approaches and only PSAL and MATLAB address it. In this context, MATLAB proposes generation of C-code from Simulink models and PSAL generates control applications compliant with IEC 61499.

Table 2. Comparison of approaches and tools.

Phase	Design				Proof	Implementation	
	function	inform.	comm.	comp.		function	rapid prot.
UML	☹	×	×	×	×	☹	×
SysML	☹	☹	×	×	×	☹	×
IntelliGrid	☹	×	×	×	×	×	×
SGAM-TB	☹	×	☹	☹	×	☹	☹
PSAL	☹	☹	✓	✓	×	✓	✓
MATLAB	☹	☹	×	☹	✓	✓	✓
IEC61499	☹	☹	✓	✓	✓	✓	×
IEC61131-3	☹	☹	✓	☹	✓	✓	×

Legend: not supported at all (×), not recommended (☹), supported but not totally (☹), and well supported (✓).

3.3. Ontologies for Smart Grid Applications

Ontologies search for a formal representation and categorization of information abstracted from real systems. This is achieved by referencing the components: classes, individuals, relations, attributes, rules, etc. [26]. The main advantage of ontologies is the inference of new knowledge from explicit knowledge by the execution of reasoning mechanisms. The most well known ontology language is Ontology Web Language (OWL), a key benefit offered is the expression of complex knowledge in a machine-readable form. Moreover, due to OWL being based on description logics (DL), an expressive formal knowledge representation is obtained.

DL give mechanisms to describe accurately ontologies by introducing the terminological (*TBox*) and assertional (*ABox*) component [27]. The *TBox* contains sentences describing concepts and relation between concepts. Instantiation of those concepts are carried out by the *ABox*. For instance, the statement: *an EMS embeds more than one service*, belongs to the *TBox*, and the concepts identified are *EMS* and *service*. In turn, the sentence: *frequency-watt mode is a service* belongs to the *ABox*, where the individual *frequency-watt* is an instance of the concept *service*.

A mechanism to enhance the reasoning process embedded within ontologies is by stating rules, which are statements in the form of "if X then Y" sentence. The W3C consortium recommends Semantic Web Rules Languages (SWRL) to denote them.

The tasks typically solved with ontologies corresponds mainly to three categories. The first one is the ontology alignment, the study [28] proposes an ontology matching process to align data models of two broadly accepted smart grid standards the CIM and IEC 61850. Additionally, in the study [29] an ontology-driven approach transforms control systems into IEC 61499 control applications by using eSWRL an extension of the SWRL. Another goal of the various ontology applications is the integration of knowledge (second category). An ontology for smart grid interactions in Building Energy Management Systems (BEMS) to optimize end-user consumption is built in [30]. In that light, models for communication technologies, interaction between stakeholders and grid structure were addressed. Another example is an smart grid information model that combines weather, spatial and time ontology applied to dynamic demand response applications [31]. The third category is related to identification of critical errors and faults at the design phase of the engineering process [32,33]. It is worth noting that none of the exposed approaches is multi-functional BESS oriented.

3.4. Model-Driven Engineering in Power Systems

The main focus of MDE is to improve the engineering process by enhancing the compatibility between systems. Thus, MDE defines a common understanding of the system under study by the establishment of models. In that basis, meta-modeling, model transformation and code generation mechanisms are carried out [34]. MDE has been largely applied to manufacture systems [35] but not substantially to the power utility domain. In this light, studies that support the automation of the smart grid control application process by means of MDE are given in [7,11,36].

3.5. Research Needs and Requirements

A previous study evaluates the efforts done to tackle the issues exposed in Section 2.3, resulting in the following four main requirements for this work:

- *Alignment with Domain Standards (R1)*: An appropriate solution to model a multi-functional BESS should benefit from the extensive data models provided by the mentioned standards in Section 3.1.
- *Common Ontology for Multi-Functional BESS (R2)*: A comparison of smart grid control development approaches was carried out in Section 3.2. This demonstrates a lack of one approach that fully covers the function and information domains during the design phase. What is not covered is a method to handle a high amount of data and also a vocabulary for a common understanding of EMS applications. The characterization of EMS behavior is dismissed since it is completely covered by other approaches (e.g., MATLAB, IEC 61499). Attempts to propose a pattern for control

strategies were done in SGAM. However, it needs to be enlarged in order to cover features mainly important to BESS use cases. An ontology is a good candidate to integrate the knowledge from EMS as highlighted in Section 3.3. Thereby, R2 searches for an ontology focused on multi-functional BESS and also for mechanisms to achieve a population of it. The resulting ontology that represents and describes EMS should guide control engineers during the design stage. A derived benefit from this is to enable the access of EMS capabilities to external systems (e.g., EMO).

- *Identification of Conflicts and Inconsistencies (R3)*: An early diagnosis of potential conflicts within EMS corresponds to the overlapping issue in Section 2.3. That topic was tackled in [37], as a result a classification of conflicts (Conflict $C_I - C_{VI}$) and its modeling by ontologies is achieved. In addition to this, the study formally defines DL queries to identify the classified conflicts and a handling solution per conflict type is exposed and recommended. The outcomes from [37] should be implemented in a whole solution that supports the design of EMS.
- *Generation of Software Artifacts (R4)*: One of the issues highlighted in Section 2.3 requires implementing EMS applications into different power system tools (i.e., design, emulator). Thereby, the generation of software artifacts at the proof-of-concept and implementation phase should be assured. MDE invokes two types of transformations: Model-to-model (M2M) and Model-to-text (M2T) transformations by the setting up of transformation rules. Both transformations should be carried out at the design stage, and this would improve the rapidity and interoperability of the EMS engineering process. The generation of code and models from an EMS platform-independent model is part of the requirement R4.

In summary, current efforts to face the issues encountered during the EMS development process were evaluated. Thereby, current standards for the power utility automation domain are studied (e.g., IEC 61850). As a result, R1 motivates the use of existent information models. Moreover, approaches that support the whole chain of the development process (e.g., SGAM, PSAL) are compared to realize the specific points that need to be targeted to address the complexity issue. This results in R2, and it searches for a common vocabulary for BESS UCs. Afterwards, the overlapping within a multi-functional BESS motivates the assessment of ontologies in smart grids. That topic was already addressed in [38]. Hence, R3 envisages the implementation of the outcomes from the mentioned study in EMSOnto. The interoperability issue is tackled by model-driven engineering techniques, and this is conceptualized in R4. As a result, the basis for the conception of EMSOnto are formulated under requirements R1-R4.

4. Application Development with the EMSOnto Approach

In the last section, the need for an engineering support for designing multi-functional BESS control applications has been identified and discussed. This design is driven by the fulfillment of certain requirements (i.e., R1–R4). In this context, model-driven engineering together with ontologies are combined to offer a flexible solution—the so-called EMSOnto approach. This section introduces the main ideas and concepts behind it, especially the EMS-ontology which forms the main part of this approach. Thus, the design of the EMS-ontology is presented and discussed accordingly to requirements R1–R4. Additionally, a user-friendly method to populate the ontology is also addressed.

4.1. General Overview of the Approach

The whole picture of the EMSOnto framework and the support it provides to control engineers during EMS-development are shown in Figure 2. Across the different phases in the design and development process, requirements R1–R4 are properly addressed.

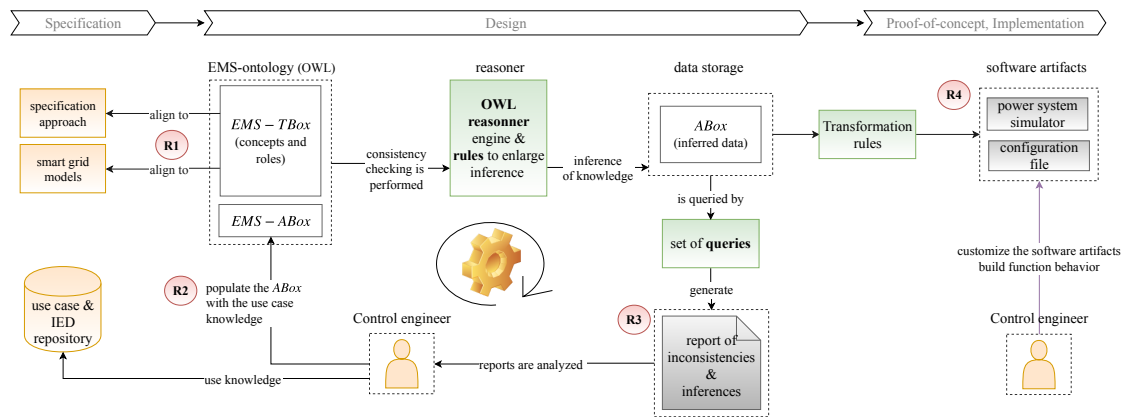


Figure 2. Overview of the EMSOnto framework and corresponding design and engineering process.

As an initial step, the control engineer defines the specification of an EMS by selecting one of the approaches from the list presented in Section 3.2. This knowledge may be used during the design stage to pre-fill the assertional component of the EMS-ontology (*ABox*). Additionally, at the design phase, further information about structure and data exchanged in the scope of the EMS application is included within the EMS-*ABox*. Current smart grid data models ease the realization of the EMS-*ABox* by providing knowledge of IEDs and corresponding control functions (like the Frequency-Watt or Volt-VAR functions in IEC 61850 [12]). Such device-specific models are usually stored within a repository and are being made available to the control engineer during the whole engineering process.

The other component needed to fully define the EMS-ontology is the terminological box (*TBox*). A set of axioms conforms the EMS-*TBox* which serve as the basis for the inference of knowledge. A main advantage of ontologies is consistency checking of the *ABox* regarding the pre-defined *TBox*. This and the inference of knowledge are achieved by the setting up of a reasoner engine [39]. Additionally, a set of premises that lead to conclusions (i.e., rules) extends the quantity of knowledge that can be inferred by just the EMS-*TBox* setup. On top of that, a list of queries leads to the extraction of selected data. As a result, to achieve data inference the EMS-*TBox*, pre-defined rules and also queries are implemented by the EMSOnto. This setup is transparent to the control engineer.

Once the *ABox* is fully defined and compliant to the *TBox*, corresponding reports can be generated showing the critical and severe issues regarding the *ABox*. At this stage, the control engineer gets helpful information about potential controller conflicts (i.e., at *ABox* level) which can be solved before starting the implementation and proof-of-concept of EMS applications. Additionally, a list of potential conflict-solving solutions per triggered inconsistency is also generated. At this stage, control engineers require adapting the EMS-*ABox* to reach an error-free EMS database. The final step in the overall process is to use model-driven engineering techniques in order to generate software artifacts and platform-specific code as outlined in Figure 2.

4.2. Core EMS-Ontology Use Cases

The core part of the EMSOnto approach as depicted in Figure 2 is based on the design of the EMS-*TBox*. This involves a deep understanding of the EMS application itself, the services deployed within it, and the components controlled or monitored by the EMS. Because of that, an evaluation of services provided by the EMS is carried out. Those services are exemplified by use cases UC1–UC5 (already introduced in Table 1). From that, a pattern to guide the design of the EMS-*TBox* is presented.

Typically, an EMS application involves a main controller and its environment as shown in Figure 3. The controller *K* carries out the corresponding control and optimization functions. The EMS itself is focused on the provision of multiple services supported by a BESS (see UC1–UC5). Thus, many

services ($Service_1, \dots, Service_n$) may be embedded in K . Each service follows specific control strategies. Hence, they need to measure states from the electrical process G , composed by IED such as batteries, smart meters, photovoltaic generators, etc. Furthermore, services should receive setpoints (w_1, \dots, w_n) to regulate certain states of the electrical process (y). As a result of the regulation process, command values (u) are sent to G . Those control values are taken into account by G as long as a setpoint (m) is available within IEDs. A similar occurrence is noted in the measurements f . They are available only if the IEDs give access to their states (s). Besides the aforementioned values, a set of parameters within the controllers (K_1, \dots, K_n) defines the control behavior. For instance, the controller within UC1 ($Q_{bat} = \alpha V_{pcc} + \beta$) requires the setting up of α and β .

Use cases UC1–UC5 are mapped into the control scheme presented in Figure 3 resulting in Table 3. This mapping validates the modelling of EMS by the previous mentioned information (i.e., $w, Service, f$, etc.). Other elements to be considered in the modelling of EMS are systems that interact with EMS besides the process G . The aforementioned use cases address services interacting with other stakeholders spread out in the distribution network (Distribution System Operator (DSO)) and the transmission network (Transmission System Operator (TSO)). In certain use cases, those systems, denoted by A , require setting variables of K remotely (e.g., parameters). For instance, UC2 deploys within the controller K the logic $P_{bat} = \gamma F_{grid} + \theta$. The charging of the battery (P_{bat}) may be configured remotely by the TSO by modifying the parameters γ and θ . Furthermore, case A needs to monitor values from K , and the controller K should be capable of sending status values (s). At the same time, A would receive feedback (f) from K . This leads to the conclusion that systems A, K and G share many similarities and can be represented under a common pattern composed of variables shown in Figure 3 and Table 3 (i.e., $y, u, w, K, Service, \dots$).

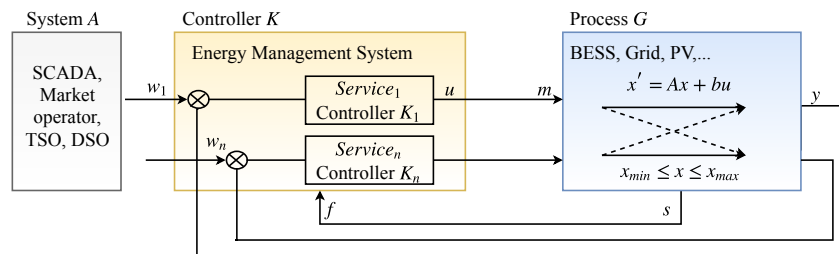


Figure 3. Usage of EMS in the context of multi-functional BESS.

Table 3. Use cases built-in an EMS.

$Service_n$	Regulat.- y	Setpoint- w_n	Controller- K_n	Control- u
Volt-VAr mode (UC1)	δV	$\delta V = V_{pcc} - V_{nom}$ $\delta V \leq 3\% V_{nom}$ V_{pcc} : voltage at PCC V_{nom} : nominal voltage	open loop $Q_{bat} = \alpha V_{pcc} + \beta$ α, β : constants	Q_{bat}
Frequency-watt (UC2)	δF	$\delta F = F_{grid} - F_{nom}$ $\delta F \leq 2\% F_{nom}$ F_{nom} : nominal frequency	open loop $P_{bat} = \gamma F_{grid} + \theta$ γ, θ : constants	P_{bat}
Self-consumption (UC3)	P_{grid}	$P_{grid} = 0$	PI control	P_{bat}
Minimization of costs with peak shaving (UC4)	$CF = P_{grid} \cdot FiT - P_{grid} \cdot EgP$ CF: cash flow FiT: feed-in tariff EgP: electricity grid price	-	dynamic programming, $Min \sum_{t=0}^T (CF)$	P_{bat}
Multi-functional BESS (UC5)	δV P_{bat}	$\delta V \leq 3\% V_{nom}$ $P_{bat} = P_{hh} + P_{pcr}$ P_{hh} : setpoint from house-holds P_{pcr} : setpoint from PCR service	PI control	P_{bat} Q_{bat}

4.3. Alignment with Smart Grid Domain Models

The EMS-ontology is aligned with smart grid domain standards to provide a broadly accepted generic use case representation, covering requirement R1. In this context, appropriate domain approaches have been analyzed and used as basis to formulate the EMS-ontology. In the following, the concepts taken from those approaches are outlined.

The structure of the EMS-ontology is motivated from concepts defined within the SGAM framework [24] and by elements introduced in the generic use case suggested by the Smart Grid Coordination Group (SGCG)—sustainable processes [40]. Both approaches are being used in various smart grid research and demonstration projects and are taken as a basis for modelling use cases [25], indicating a broad acceptance of them by the power system community. The study [40] suggests a classification of devices, actors and systems involved in a use case. Some of them are following outlined: the concept High Level Use Case (HLUC) gives a general idea of a function and is technology-neutral. An HLUC invokes Primary Use Cases (PUC) to detail its functionalities. In turn, the concept System is an arrangement of components and systems to accomplish a use case. The mentioned concepts are taken as a basis for modelling the function layer of the EMS-ontology.

Previous studies carried out in Section 3.1 presented a list of smart grid data models. Among them, this article selects an IEC 61850 approach because of its maturity and the availability of data models for power system components and functions mainly concerned by UC1–UC5. The standard IEC 61850-7-3 [41] proposes data models for run-time signals in a communication structure. Hence, the definition of common data classes for status information (CDCStatusInfo), measured information (AnalogueInfo), control signals (CDCStatusCtl, CDCAnalogueCtl), status settings (CDCStatusSet), etc. The attributes of that data are of type Boolean, TimeStamp, String, etc. The mentioned concepts motivate the design of the information layer within the EMS-ontology.

4.4. EMS-Ontology for Modelling BESS Applications

This section provides an overview of the core elements of the EMS-ontology covering concepts, roles, individuals and axioms.

As shown in Figure 3, the implementation of any UC requires the modeling of a controller K , external systems A and electrical devices defined as components of the process G . The aforementioned elements (A, K, G) are represented by the concept Application. Hence, Application contains sub-classes of type Battery, EMS, Meter, etc. The concept System is created as a root of the EMS-ontology. It gathers Application by the role hasApplication. As a consequence, a System is composed of control applications (e.g., EMS), external systems (e.g., SCADA) and electrical devices (e.g., BESS, PV). On the other hand, the controller K may implement more than one service ($Service_1, \dots, Service_n$). To represent those services, the concept HLUC is introduced. For instance, voltage control, Primary Control Reserve (PCR) and self-consumption are abstracted by an HLUC. Furthermore, a categorization of HLUC is carried out by the concepts: Power Quality (PQ), Power Stability (PS), Minimization of Supply Costs (MSC), etc. [4].

Further details of an HLUC are given by a PUC, where the services provided by a HLUC are listed. Moreover, a PUC may contain other services, which in turn are also represented by a PUC. For instance, a PUC(voltage control) requires embedding further services such as a PUC(state estimator) and PUC(PI controller). To achieve that, the role hasPUC is created. It has the concepts PUC, HLUC as domain and PUC as range. Additionally, in case a PUC requires implementing a control scheme, the concepts ControlGoal and Constraint are suggested. ControlGoal represents a control problem $\lim_{t \rightarrow \infty} (y_{ref} - y) \rightarrow 0$ or an optimization problem $Minimize f(x)$ subject to $g_i(x) \leq 0, h_j(x) = 0$ constraints. The equality and inequality constraints are represented by the Constraint concept.

The introduced concepts and roles are illustrated in Figure 4. To fully understand such representation, it is imperative to comprehend the core elements of an ontology (i.e., role, concepts, etc.) [26]. The formal

representation of EMS-ontology is based on DL [27] extended with the concrete-domain concept [42]. A full description of it is given in Appendix A.

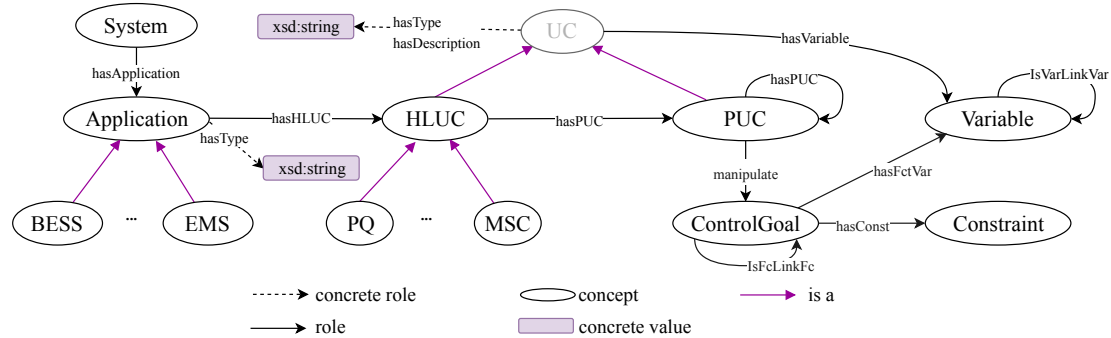


Figure 4. Graph representation of EMS structure.

The implementation of a multi-functional system within the controller requires the exchange of information with the field process and other systems. A model of this data is formally defined by a set of concepts having as the main class the concept External (see Figure 5). External is of type Input or Output. An Input is classified by Feedback and Setpoint. In turn, the concept Output is specified under Control and Status. A mapping of those concepts to the scheme in Figure 3 is stated as follows: Control(u), Setpoint $\{w_1, \dots, w_n, m\}$, Feedback (f), Status (s).

On the other hand, internal variables of A, K, G are modeled by the Internal concept. Thereby, Internal is structured under State and Param. Param represents variables that do not change dynamically (i.e., parameters). State conceptualizes variables that change over time. For instance, if G is represented under a Linear Time Invariant (LTI) Single Input Single Output (SISO) system $x' = Ax + bu$, then the variable x is represented by a State and b by a Param.

Values attributed to dynamic and static variables may be binary (i.e., 1 and 0), numeric data (i.e., continuous and discrete) and character. Those are represented by Binary, Numeric, Char concepts. Attributes assigned to them are modeled by the roles hasTimeStamp, hasUnit, hasFormat, etc. (see Figure 6). The role hasTimeStamp is used to reach accuracy in acquired values, enabling the synchronization of an internal time (e.g., time in K) with an external time (e.g., time in G). Other attributes of numeric values such as the format, unit and range are also addressed.

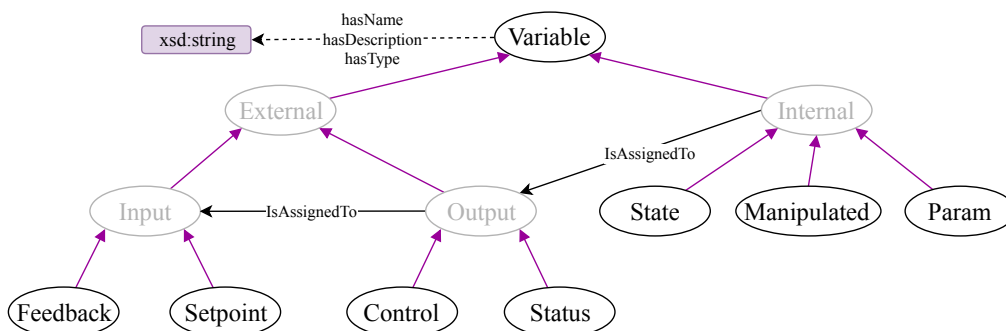


Figure 5. Graph representation of variables' types.

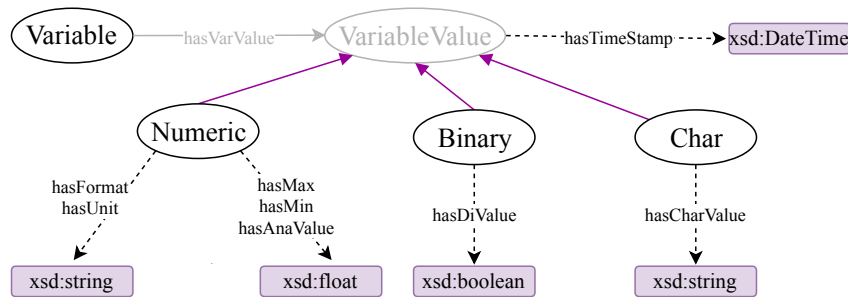


Figure 6. Graph representation of variables' attributes.

UCs embedded within an EMS that involves battery support are surrounded by common particularities. Thus, a pattern that represents a generic UC can be reached (i.e., HLUC or PUC). One of those particularities is based on the fact that a UC is bound to a minimal storage capacity. For instance, a PCR use case requires a minimum of 1 MW to allow a pool of small BESS to participate [2], hence the concept *BessSize*. In a multi-functional BESS context, the battery needs to share its full capacity among services. Thus, the capacity rented per HLUC is modelled by the *UCAh* concept. An example of this is presented in UC5, where three services rent the whole capacity of the BESS. Household, PCR, and voltage control rent 20%, 40% and 40%, respectively. Those values are modelled by *UCAh*. On the other hand, not all the UCs are enabled at the same time, thus a flag that represents the current operation of a UC is modeled by *Ena*. Moreover, a State-of-Charge (SoC) is calculated per UC and represented by the concept *UCSoC*. This value monitors the support given by a battery to a certain UC and guides the monetization of provided power. Furthermore, a way to manage conflicts when more than one UC operates is by setting up priorities, hence the concept *Priority*.

The mentioned concepts are subclasses of the concept *ParamUC* (see Figure 7). Additionally, parameters of a battery are grouped and modelled within the concept *ParamDevice*. For instance, the concept *Pmax* represents maximum active power to be set into the battery and *BattAh* the capacity of a battery. States of an electrical device such as active power, reactive power, etc. are modelled within the concept *State* by *P*, *Q*, respectively. In addition to this, information regarding controller requirements such as error, settling time, available time, etc. are represented under the concept *ParamCont*.

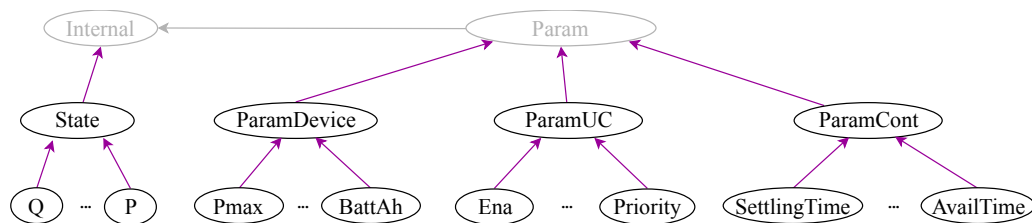


Figure 7. Graph representation of parameters and states.

Concepts and roles of the *TBox* component were previously addressed. They enable the classification of EMS-knowledge by concepts and the modeling of relations between data according to roles specification. Besides this, axioms defined within the *TBox* and a set of rules guarantee the inference of implicit data from initial knowledge stated by domain experts. An extract of those rules and axioms are described by DL language and SWRL in Table 4 as follows.

Table 4. Axioms within the EMS-*TBox* and SWRL rules.

SWRL Rule & DL Axiom	Description
$hasControl \sqsubseteq hasVariable,$ $T \sqsubseteq \forall hasControl.Control$	The role <code>hasControl</code> is included within <code>hasVariable</code> . The range of <code>hasControl</code> is <code>Control</code> .
$hasVariable(x, y) \wedge Control(y) \rightarrow$ $hasControl(x, y)$	An individual x <code>hasVariable</code> y , if the variable y is of type <code>Control</code> . Then, it is deduced that, the individual x <code>hasControl</code> y .
$T \sqsubseteq \forall IsVarLinkVar.Variable,$ $\exists IsVarLinkVar.T \sqsubseteq Variable$	The role <code>IsVarLinkVar</code> has as range and domain the concept <code>Variable</code> . The assertion <code>IsVarLinkVar(x,y)</code> states that a manipulation of <code>Variable(x)</code> would affect the value of <code>Variable(y)</code> .
$hasFctVar \circ IsVarLinkVar \sqsubseteq hasFctVar,$ $hasFctVar \circ hasFctVar \sqsubseteq IsFcLinkFc,$ $manipulate \circ IsFcLinkFc \sqsubseteq manipulate$	The role <code>hasFctVar</code> relates a <code>ControlGoal</code> with a <code>Variable</code> . The role <code>IsFcLinkFc</code> has as range and domain the concept <code>ControlGoal</code> . The constructor \circ is of type composition [27]. It allows for stating a complex role inclusion axiom. The referred axioms investigate a <code>ControlGoal</code> that is manipulated by a PUC.

4.5. Templates for Application Design

This section addresses requirement R2. Thereby, a mechanism to instantiate the previous introduced EMS-*TBox* is proposed, resulting in the realization of the EMS-*ABox*.

In order to provide the necessary information about the EMS, EMSOnto proposes the use of EMS-templates, a set of spreadsheets providing an environment for collecting and storing EMS-application related data. In this context, the usability of those templates is demonstrated by considering use case UC5 as an example.

The structure of UC5 is represented in Table 5. From there, the following knowledge is available: A `System(Sys)` contains four applications: `Application{Battery, HouseHold, Meter, EMS}`. In addition to that, an application type is conferred to them by instantiating the role `hasType`. Hence, `hasType{(Battery, BESS), (HouseHold, Load), ...}`. This allows for importing models already defined within the IED repository such as `BESS`, `Meter`, `Load`. The `Application(EMS)` requires providing three services that are represented under `HLUC{VoltageCtr, FW, SelfC}`, the description of them is given in Table 5. Sub-concepts of `HLUC` (e.g., `PQ`, `PS`) are introduced to classify the services, hence the definition of `hasType{(VoltageCtr, PQ), (FW, PS), (SelfC, MSC)}`. Furthermore, an extended specification of functions within `HLUC` are given by the role `hasPUC` as follows `hasPUC{(VoltageCtr, PControl), (FW, K1), (SelfC, K2)}`. The first pair represents the assertion: an `HLUC` called `VoltageCtr` contains a `PUC` called `PControl` (i.e., PI controller). The same is expected for the other pairs. Further details of an `HLUC` and `PUC` are hereinafter given.

The concept `ParamUC` suggests parameters for a generic UC (`PUC` or `HLUC`). Thereby, the inheritance of a generic `UC` by the previously defined `HLUC` such as `VoltageCtr` would generate a list of attached parameters. As a result, the inference of the following assertions is achieved: `Priority(prt)`, `UCSoC(SoC)` and `hasParameter{(VoltageCtr, prt), (VoltageCtr, SoC)}`. The role `hasParameter` represents a sub-role of `hasVariable` having as a range the concept `Param`. The `priority` and `SoC` assigned to `VoltageCtr` are represented by `Priority(prt)` and `UCSoC(SoC)`, respectively (see Table 6). Other parameters were also generated, but not presented in the table.

A `PUC` usually needs to read measurement values from a process (G), and, according to stipulated references, a control algorithm is carried out and control commands are sent back to G . Bringing this to the UC5 example, a PI controller searches for the control of the voltage deviation at the PCC point (i.e., δV). To this end, the voltage of the PCC point (fd_V_{pcc}) is measured and retrieved from a meter. Additionally, a control signal is sent to charge or discharge the battery (ct_Q_{bat}). Formatting those assertions according to the *ABox* syntax produces the following instantiations: `hasControl(PControl, ct_Qbat)`, `Control(ct_Qbat)`, `hasFeedback(PControl, fd_Vpcc)` and `Feedback(fd_Vpcc)`. The role `hasControl` was introduced in Table 4, and a similar logic applies for `hasFeedback`, but having as range the concept `Feedback`. Table 7 shows the EMS-template to gather the aforementioned knowledge, and a detail of variables' attributes is also exposed.

Table 5. Structure of an EMS (e.g., UC5).

System	Application	Application Description	Type	HLUC	HLUC description	Type	PUC
Sys	Battery	battery connected to the grid	BESS	-	-	-	-
Sys	Meter	model of a meter located at PCC	Meter	-	-	-	-
Sys	HouseHold	pool of households that demands active power	Load	-	-	-	-
Sys	EMS	control application that deploys multi-services	-	VoltageCtr	battery supports the control of voltage at PCC point	PQ	PIControl
Sys	EMS	control application that deploys multi-services	-	FW	battery supports the regulation of grid frequency	PS	K1
Sys	EMS	control application that deploys multi-services	-	SelfC	self-consumption is provided to a pool of houses	MSC	K2

Table 6. Parameters of a HLUC (e.g., VoltageCtr).

HLUC	Variable	Type	Variable Description	Value	Format	Unit	Min	Max
VoltageCtr	prt	Priority	priority of the use case	2	Float	-	-	-
VoltageCtr	SoC	UCSoC	SoC of the use case	40	Float	%	20	80

The table referred to also exposes the knowledge needed to represent the control goal assigned to PUC(*PIControl*). The value to be regulated by PUC(*PIControl*) is represented by *ControlGoal(Cg1)* and *manipulate(PIControl, Cg1)*. Due to *ControlGoal(Cg1)* searching for the regulation of $(V_{pcc} - V_{nom})$, the instantiations *Manipulated(V_{pcc})* and *Param(V_{nom})* are created. The association of them to *ControlGoal(Cg1)* is given by the role *hasFctVar*, previously introduced in Table 4.

Table 7. Tabular data to implement detail of a PUC (e.g., *PIControl*).

PUC	Variable	Type	variable description	Value	Format	Unit	Min	Max
<i>PIControl</i>	$fd_{V_{pcc}}$	Feedback	voltage read from meter	230	double	volt	-	-
<i>PIControl</i>	$ct_{Q_{bat}}$	Control	signal to control react. power of a battery	8	double	kVar	-10	+10
PUC	ControlGoal	control goal description		Var.	Type	IsVarLinkVar		
<i>PIControl</i>	<i>Cg1</i>	$(V_{pcc} - V_{nom})$ is regulated		V_{pcc}	Manipulated	-		
<i>PIControl</i>	<i>Cg1</i>	$(V_{pcc} - V_{nom})$ is regulated		V_{nom}	Param	-		

Elements within the process *G* and external applications *A* are also modelled by the EMS-ontology using the previously presented templates. Benefits arising from EMS-template practices are discussed below:

- *Knowledge is consistent with the TBox*: EMS-templates set restrictions to the data gathered within the spreadsheets. Control engineers manually fill those spreadsheets, where data entered is pre-checked to be aligned to the already specified *TBox*. For instance, the knowledge: *a control signal is assigned only to variables of type setpoint*, is represented by the axiom $Control \sqsubseteq \forall IsAssignedBy.Setpoint$ within the *TBox*. Due to EMS-templates being built under the *TBox* basis, only setpoints can be associated with control variables in the scope of EMS-templates.

- *Gathering unlimited amount of data:* None of the approaches already presented in Table 2 support control engineers with an easy collection and treatment of data. EMS-templates ease the gathering of data because of the tabular format. Moreover, filtering, comparison, ordering, exporting, data formatting, etc. are also possible in a large set of tools to process tabular format (e.g., Microsoft Excel, OpenOffice, LibreOffice Calc, etc.).
- *Setting-up of models:* EMS-templates facilitate the creation of pre-defined models to be stored within the IED and use case repository for further use (e.g., *VoltageCtr*, *PIControl*). Those models are afterward assigned to individuals of type EMS, BESS, HLUC, etc. by the role *hasType*. Moreover, EMS-templates enable the inspection of available models within the repository.

4.6. Controller Inconsistencies Identification

At this stage, all the required knowledge is gathered through EMS-templates. Further analysis enables consistency checking and inference of conflicts. Consistency checking determines if the *ABox* is an instance of the *TBox*. Once this is accomplished, inferences within the *ABox* are discovered due to axioms defined in the *TBox* and also a set of rules as depicted in Table 4. This section presents the inconsistencies addressed by EMSOnto, conforming to the requirement R3.

A deep study of conflicts identification and handling is carried out in [37]. A classification of conflicts types ($C_I - C_{VI}$) is formally defined and evaluated there. The encoding and formulation of conflict detection are defined by DL queries. The ontology defined in [37] is considered in the EMS-ontology design. As a consequence, EMS-ontology is able to respond to conflict identification and their handling. To cover the mentioned features, an extension of EMS-templates targeting knowledge especially important to conflict detection is carried out. An extract of those EMS-templates is presented in Table 8.

The example in Table 8 states assertions regarding a control goal and its environment in order to investigate the conflict C_I . To this end, the following assertions are included: $\text{manipulate}\{(puc1, Cg1), (puc2, Cg2)\}$, $\text{Control}(x)$, $\text{Manipulated}(y)$, $\text{hasFctVar}\{(Cg1, x), (Cg2, y)\}$ and $\text{IsVarLinkVar}(x, y)$. Applying the axioms regarding the role *manipulate* and *hasFctVar* defined in Table 4, the inference $\text{manipulate}(puc1, Cg2)$ is raised. Inferred knowledge is highlighted in Table 8.

The aforementioned knowledge is enough to investigate the conflict C_I by executing the query Q_{CI} given by the axiom: $\text{ControlGoal} \sqcap 2 \text{Manipulated}^- . PUC$, which investigates control goals manipulated or affected by at least two different UCs. The execution of that query concludes that $\text{ControlGoal}\{Cg2\}$ and $\text{ControlGoal}\{Cg1\}$ are in conflict with type C_I .

Table 8. EMS-template to detect conflict C_I .

PUC	ControlGoal	Control Goal Description	manipulated ⁻	Variable	Type	IsVarLinkVar
puc1	Cg1	$\text{Min}(F(x))$	puc1	x	Control	y
puc2	Cg2	$\lim_{t \rightarrow \infty} (y - y_{ref}) \rightarrow 0,$ y : manipulated value, y_{ref} : setpoint	puc2, <u>puc1</u>	y	Manipulated	-

4.7. Generation of Software Artifacts

As a final step, the generation of code from a neutral-technology platform is achieved. This code is not restricted to programming languages. Thus, other kinds of software artifacts such as documentation or configuration files may be produced as well. To accomplish that, specific information from EMS-templates is extracted and formatted according to a target platform. That information is gathered by the means of selected queries. A non-exhaustive list of them is presented in Table 9. Information retrieved from query Q_I helps to gather the attributes regarding the status of an HLUC (i.e., enabled or disabled). Besides this, the query Q_{II} is used to know the priority values set into a HLUC. On the

other hand, reports that show the functionalities provided by an EMS can be obtained from the queries $Q_{III} - Q_V$.

In this work, a MATLAB/Simulink block is generated as well as MATLAB-code (for details, see Section 5). It should be stressed that EMS-ontology do not offer support for function behaviour representation. Indeed, in the related work section (see Section 3.2), it is mentioned that other approaches such as IEC 61499, IEC 61131-3 and MATLAB/Simulink cover adequately that feature. Hence, EMSOnto is meant to work with the support of other approaches to accomplish a full control application implementation.

Table 9. Querying the EMS-ABox.

Query	Description	DL Query
Q_I	What is the Enable parameter of a specific HLUC named $hluc$?	$Ena \sqcap hasVariable^- .HLUC\{hluc\}$
Q_{II}	What is the Priority parameter of a specific HLUC named $hluc$?	$Priority \sqcap hasVariable^- .HLUC\{hluc\}$
Q_{III}	What are the main functionalities to be provided by an application of type EMS?	$HLUC \sqcap hasHLUC^- .(Application \sqcap EMS)$
Q_{IV}	What are the sub-functionalities provided by a HLUC?	$PUC \sqcap hasPUC^- .(HLUC)$
Q_V	What are the parameters of a specific controller named $cont$?	$ParamCon \sqcap hasVariable^- .PUC\{cont\}$

5. Proof-of-Concept Evaluation

In this section, the steps to be followed by a control engineer to achieve the design and validation of a selected use case example by the support of EMSOnto are shown. In this context, the phases specification, design, proof-of-concept and laboratory validation are covered. The realization of the mentioned development process enables demonstrating the benefits of EMSOnto as well as to grasp its limitations. Moreover, potential future work to improve EMSOnto are also apprehended.

5.1. Framework Prototype and Validation Example

EMSOnto is evaluated under the execution of a control application that implements an arrangement of use cases referenced in Table 3. The control application to be analyzed implements Frequency-Watt (FW)-UC2 and Self-Consumption(SelfC)-UC3. This UC example is called Customer Energy Management System (CEMS), a schema of the CEMS architecture is depicted in Figure 8. Mathematical models of the battery and control strategies embedded within the CEMS, besides profiles for PV generation, load consumption and grid behavior are outlined below.

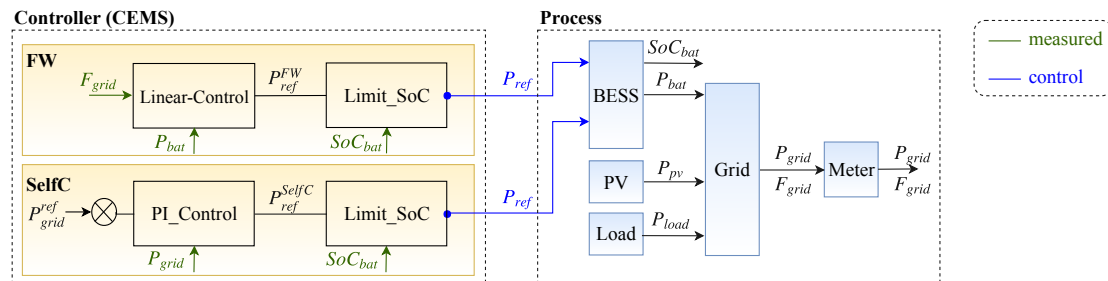


Figure 8. CEMS structure, systems and sub-systems

5.1.1. BESS Model

The used battery model follows the mechanisms as described in [43]. Thereby, the model is represented by a voltage source (E_{bat}) in series with a resistance (R_{bat}) as indicated in Equation (1). The voltage of the battery (V_{bat}) depends nonlinearly on the SoC of the battery (SoC_{bat}), and this is derived from the value of E_{bat} given by Equation (2). The identification of the parameters A and B are

detailed in [43]. The SoC_{bat} is estimated by integrating the battery's current (I_{bat}) on an interval time as shown in Equation (3), where Q represents the battery's total capacity and $SoC(0)$ the initial SoC:

$$V_{bat} = E_{bat} - R_{bat}I_{bat}, \quad (1)$$

$$E_{bat} = E_0 - k\left(\frac{1 - SoC_{bat}}{SoC_{bat}}\right)Q + Ae^{-B(1-SoC_{bat})Q}, \quad (2)$$

$$SoC_{bat} = SoC(0) + \int \frac{I_{bat}}{Q} dt. \quad (3)$$

5.1.2. PV, Load, Grid Profiles

The PV DC generation was obtained from a model dependent in temperature and irradiance level as suggested by [44]. The temperature and irradiance data are measured with a sampling rate of 60 s and were taken over the year 2005 in Vienna, Austria from the satellite-derived HC3 Archives Web service [45]. A 60 s PV DC measurement profile is used as input data for the PV inverter model. This DC measurement is converted into AC power by considering an efficiency conversion factor. Active power generation data from Austrian households was provided by the project Autonomous Decentralised Renewable Energy Systems (ADRES) [46]. The measurements correspond to the year 2009 and are taken with a resolution of 1 s. On the other hand, the frequency profile was obtained from ENTSO-E Netzfrequenz [47] and corresponds to January 2018. The power of the grid is calculated from Equation (4) assuming that the measurements P_{bat} , P_{pv} , P_{load} are known:

$$P_{grid} + P_{bat} + P_{pv} + P_{load} = 0. \quad (4)$$

5.1.3. Control Mechanism

The CEI 0-21 standard specifies a frequency-dependent active power limitation for DER connected into the low-voltage grid [48]. Guidelines defined in that standard are fulfilled in the FW mode. In that sense, FW commands the active power of the battery to balance the frequency of the grid as stated in Equation (5). The resulted active power value P_{ref}^{FW} is restricted to technical limitations of the battery (i.e., SoC and active power limits) as indicated in Equation (6). SelfC controls the BESS to assure that no energy from the grid is taken. To achieve that a PI controller is designed by following Equation (7), the resulted active power value P_{ref}^{SelfC} is also restricted by Equation (6):

$$\Delta P_{ref}^{FW} = \alpha \Delta F_{grid}, \quad (5)$$

$$SoC^{min} < SoC < SoC^{max}, \quad P_{bat}^{min} < P_{bat} < P_{bat}^{max}, \quad (6)$$

$$e = P_{grid}^{ref} - P_{grid}, \quad P_{grid}^{ref} = 0, \quad P_{ref}^{SelfC} = K_p e + K_i \int_0^t e(\tau) d\tau. \quad (7)$$

The specification, design and proof-of-concept of CEMS are deployed under EMSOnto, and these steps are addressed in the following.

5.2. Realized Development Example

This section focuses on the steps carried out by the control engineer in order to realize the aforementioned CEMS application. The whole realization process is structured in four steps as shown in Figure 9. The first step is focused on the description of function and information domain of the CEMS, and this knowledge is gathered within the EMS-ABox. This is realized by the control engineer having

as a guideline the EMS-templates. Once the EMS-*ABox* is filled, an analysis to infer implicit knowledge is undertaken by one engine reasoner. The inferred data is queried to evaluate inconsistencies that need to be solved before any further design of the CEMS, and those inconsistencies are presented in the form of reports. They are analyzed by the control engineer. This in turn performs a clearance and corrections over the EMS-*ABox*. This process is repeated until the control engineer approves the absence of inconsistencies within the EMS-*ABox*. The EMS-*ABox* is transformed into software artifacts that are useful during the proof-of-concept phase. A clearance and adjustment of them is also required. Details of the aforementioned steps are explained as follows.

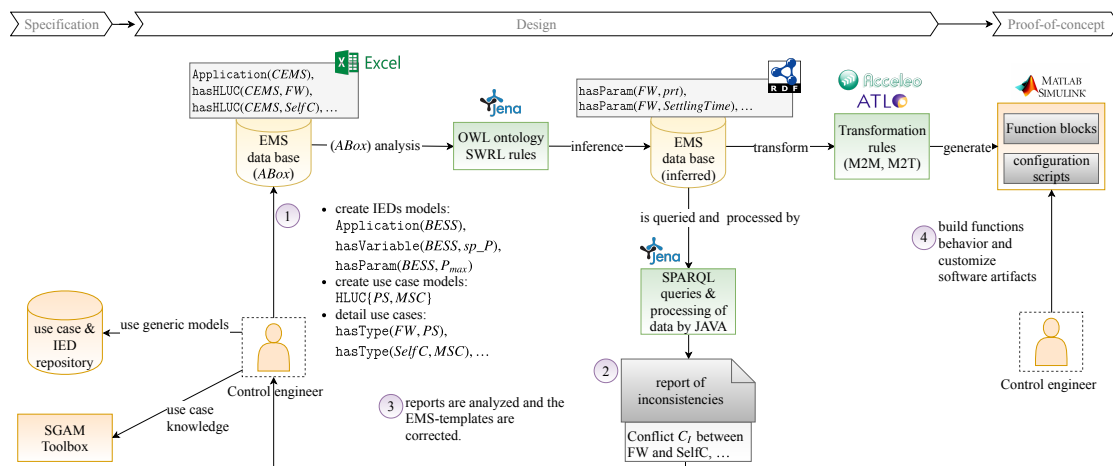


Figure 9. EMSOnto in practice by realizing the CEMS.

5.2.1. Step 1: CEMS Description via EMS-Templates

Control engineers employ the EMS-templates to specify knowledge about the information, function and component layers of the CEMS. This action is referenced in Figure 9 by Step 1 and detailed in the following paragraphs.

In the specification phase, a list of approaches are offered to control engineers to formally describe the CEMS (e.g., UML, SysML) as exposed in Table 2. Among the different options, the SGAM-Toolbox was chosen. Hence, a representation of the CEMS previously outlined in Figure 8 is matched to the SGAM matrix (information layer), where components of the system under study and also the exchanged information across them is illustrated, see Figure 10. The CEMS controller is located at the customer premises. The BESS and PV are situated at the DER-domain and field/process-zone. The meter located at the PCC point belongs to the customer-domain and field-zone. In turn, the low-voltage distribution grid and loads are matched to the process-zone. SGAM-Toolbox offers more schemes to detail the structure and behavior of control applications, covering the other layers of SGAM such as component and communication.

After the specification phase is concluded, control engineers reuse the derived knowledge at the design stage to fill the EMS-templates. One template is focused only on the structure and functionalities covered by CEMS (see Table 10). A **System(Sys)** is composed of **Application(CEMS, BESS, PV, ...)**. Services carried out by them are described under **HLUC**, for instance **Application(CEMS)** owns **HLUC{FW, SelfC}**. A description and a type are assigned to **HLUC** and **Application** by the roles **hasDescription** and **hasType**. Hence, **hasDescription(FW, active power ...)** and **hasType(FW, PS)** belong to **HLUC(FW)**.

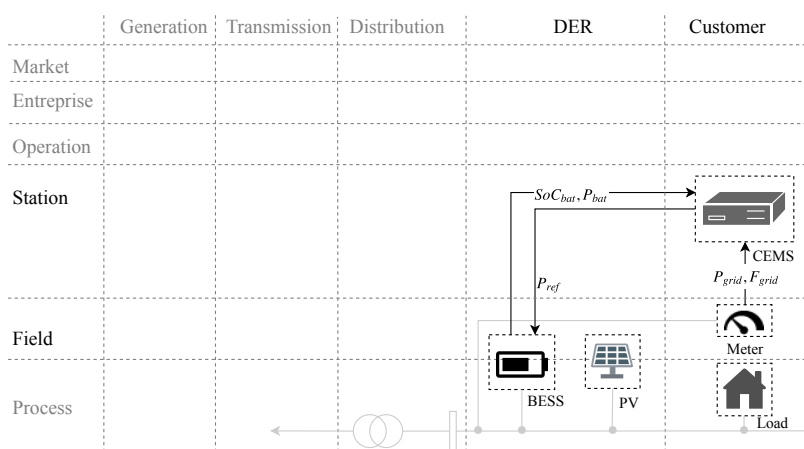


Figure 10. CEMS represented under a SGAM information layer.

Table 10. Knowledge of CEMS-structure.

System	Application	Description	Type	HLUC	Description	Type
Sys	CEMS	customer energy management system	-	FW	active power is injected to support frequency regulation	PS
				SelfC	power from the grid is avoided	MSC
	BESS	battery energy storage system	BESS	-	-	-
	PV	photovoltaic system	PV	-	-	-
	Meter	meter located at the pcc point	Meter	-	-	-
	Load	consumption of energy by the household	Load	-	-	-
	Grid	grid simulation to model the frequency and power behavior	Grid	-	-	-

A further description of HLUC is given by the concept PUC as shown in Table 11. Hence, the HLUC(FW) contains PUC[Linear-Control, Limit_SoC]. A description of PUCs is given as well as the category details (e.g., FWHZ). Type specification leads to inherit Variables from the UC repository. Therefore, variables not inherited should be specified by the control engineer as follows: the PUC(Limit_SoC) controls Application(BESS) through Control(ct_BESS_P), the measurement of the frequency at the PCC point is derived from Feedback(fd_Meter_F). Moreover, PUC(PI_Control) controls the battery by commanding Control(ct_BESS_P).

A generic HLUC was achieved by the definition of ParamUC and ParamCont as exposed in Section 4.4. PS and MSC being generic HLUCs, the matching of HLUC{FW, SelfC} to PS and MSC respectively derives the population of HLUCs as shown in Table 12. In this context, a priority is set to HLUC(FW) by stating ParamUC(Prt). Information related to the controller features is described under ParamCont. Hence, the settling-time of HLUC(FW) is set to 30 s and stated by ParamCont(SettlingTime).

IED data models (e.g., BESS, meter) are created by the control engineer and stored within the IED repository to be used in future EMS design. A battery model is shown in Table 13. It was not the aim of EMS-templates to gather data from control algorithms deployed within the BESS. Thus, only internal and external variables of the battery are considered within the model. In addition to this, those models assist with identifying the variables available to the CEMS such as Setpoint and Status.

Table 11. Description of PUCs within HLUC(FW, SelfC).

HLUC	PUC	Description	Type	Variable	Description	Type
FW	Linear-Control	The active power is decreased or augmented linearly according to frequency variations	FWHZ	fd_Meter_F	feedback of the frequency at the PCC point	Feedback
FW	Limit_SoC	SoC of the battery is monitored before setting P_{bat}	-	ct_BESS_P	signal to control the active power of the battery	Control
SelfC	PI_Control	PI control is executed to keep P_{grid} equal to zero	-	ct_BESS_P	signal to control the active power of the battery	Control

Table 12. Parameters of HLUC(FW).

HLUC	Variable	Description	Type	Value	Format	Unit
FW	Prt	priority of the use case	ParamUC	1	Float	-
FW	SettlingTime	time within the reference value is reached	ParamCont	30	Float	sec

Table 13. Battery model created by a control engineer.

Application	Variable	Description	Type	Value	Format	Unit
BESS	sp_P	reference active power to control charge/discharge	Setpoint	-	double	kW
	Pmax	maximum active power	Param	4	double	kW

The IED repository leverages models from IEC 61850, a large list of function and IED models is detailed in the standard. A model suitable to PUC(Linear-Control) is extracted from the logical node FWHZ exposed in the standard IEC 61850-90-7 [12], resulting in Table 14.

Table 14. PUC(Linear-Control) conception from LN-FWHZ.

PUC	Variable	Description	Type	Value	Format	Unit
Linear-Control	Wgra	active power gradient in percent of frozen active power value per Hz	Param	40%	Float	W/Hz
Linear-Control	HzStr	delta frequency between start frequency and nominal frequency	Param	0.04	Float	Hz
Linear-Control	HzStop	delta frequency between stop frequency and nominal frequency	Param	0.01	Float	Hz

The detection of certain inconsistencies requires the gathering of extra specific information—for instance, the detection of conflict C_I requires filling Table 15. That conflict looks for a coupling across PUCs or HLUCs. The C_I identification is based on a proper definition of control goals to be carried out by PUCs. In this light, the variables involved within a control goal are specified (i.e., Manipulated, Control) as follows: PUC(Linear-Control) performs ControlGoal(Cg1), its objective is to manipulate the power at the electrical connection point of the battery (P_{bat}). In turn, PUC(PI_Control) satisfies ControlGoal(Cg2) and requires the manipulation of P_{grid} . In addition to this, the dependency between the variables P_{bat} and P_{grid} is derived from $P_{bat} + P_{load} + P_{grid} + P_{pv} = 0$. Hence, Linear-Control. P_{bat} affects the value of PI_Control. P_{grid} .

Table 15. Information to detect conflict C_I .

PUC	ControlGoal	Control Goal Description	Variable	Type	IsVarLinkVar
Linear-Control	Cg1	$P_{bat} = \gamma F_{grid} + \theta$	P_{bat}	Manipulated	PI_Control. P_{grid}
PI_Control	Cg2	$P_{grid} = 0$	P_{grid}	Manipulated	-

5.2.2. Step 2: Assertions Derived from EMS-ABox

This section describes the support given by EMSOnto to control engineers in order to build an error-free EMS-ABox. In that sense, an inference process based on the knowledge reached from Step 1 is carried out. The inferred data is queried to generate reports that highlight inconsistencies within the EMS-ABox. Additionally, conflict-solving solutions are included in those reports. Tools and approaches to implement the mentioned procedures are also introduced in this section.

Once the gathering of knowledge about the CEMS is complete (EMS-ABox), mechanisms for consistency-checking and inference need to be carried out. Those mechanisms are implemented by the Apache Jena framework, an open source Java framework used to implement semantic web applications. Jena provides a predefined OWL reasoner and a generic rule reasoner, especially important to SWRL rules' definition. It is worth mentioning that Jena requires the knowledge in Resource Description Framework (RDF) format to perform any type of inference. Thus, Google Refine with RDF extension is employed to transform data from EMS-templates into RDF format.

The inferred knowledge is queried to localize any inconsistency as defined in Section 4.6. Those queries are implemented using SPARQL update, an RDF query language. A set of handling solutions to detect conflicts (C_I, \dots, C_{V_I}) is properly addressed in [38] and implemented by EMSOnto. In case C_I and C_V are raised together, EMSOnto proposes the setting up of priorities per PUC. The PUC with highest priority is considered and the others are dismissed. Knowledge needed to carry out the mentioned handling solution is attached to a PUC named CI .

The data retrieved from EMS-ABox is formatted and handled to control engineers in the form of reports, and an extraction of those reports is shown in Table 16. The final analysis carried out by control engineers is also depicted. Due to the conflicts C_I and C_V being raised together, the handling solution advised by EMSOnto is PUC(CI).

Table 16. Conflicts triggered by reports and control engineer analysis.

Inconsistency	Detected	Conclusion Derived from Queries	Control Engineer Analysis
Multi-objective optimization/ C_I	✓	Linear-Control and PI_Control affect the same variable P_{grid} . PI_Control requires P_{grid} to be zero and Linear-Control requires P_{bat} equal to $\gamma F_{grid} + \theta$, which indirectly affects P_{grid}	This conflict is considered as severe, the recommendation from EMSOnto (PUC(CI)) is implemented
Setpoint set by at least two use cases/ C_V	✓	The setpoint sp_P of the PUC(BESS) is set by two PUC{Limit_SoC, PI_Control}	The commanding of the same setpoint by two different PUC are not considered as a harmful conflict. However due to the presence of C_I , only one control signal should be sent

5.2.3. Step 3: Design of the EMS based on Analysis of Reports

Step 3 involves the manual work realized by control engineers to reach an error-free EMS-ABox that will be deployed during the proof-of-concept and implementation phase.

An analysis of reports generated during Step 2 is carried out, leading in some cases to an adjustment and correction of the EMS-ABox. This correction may depend on the handling solutions provided by EMSOnto. EMSOnto owns a set of pre-defined PUC to handle conflicts, and they are available within the UC repository. The one that manages the conflict C_I and C_V is PUC(CI).

The PUC(Handling) is created by the control engineer to manage C_I and C_V . The type of it is assigned to CI , thus an automatic generation of variables is achieved. An extract of this is shown

in Table 17. A unique active power signal to control the battery is chosen among the variables *SelfC_BESS_P* and *FW_BESS_P*. The higher priority is allocated to HLUC(FW) as assigned in Table 18. Once the EMS-ABox is updated and extended, Step 2 is executed again, and the new report does not detect new inconsistencies.

Table 17. PUC(Handling) to resolve C_I and C_V .

HLUC	PUC	Description	Type	Variable	Description	Type
CEMS	Handling	A function that resolves the conflicts between FW and SelfC	C_I	SelfC_BESS_P	signal sent by the HLUC SelfC to control the active power of the battery	Feedback
				FW_BESS_P	signal sent by the HLUC FW to control the active power of the battery	Feedback

Table 18. Configuration of priorities.

HLUC	Variable	Description	Type	Value	Format	Unit
FW	Prt	priority of the use case	ParamUC	2	Float	-
SelfC	Prt	priority of the use case	ParamUC	1	Float	-

5.2.4. Step 4: Customization of the Software Artifacts

The knowledge derived from Steps 1–3 is transformed into software artifacts important for the validation of EMS. This is achieved in Step 4 by performing model-driven engineering techniques.

The proof-of-concept of the CEMS is implemented in MATLAB/Simulink. A simulink model supports a block diagram representation, code generation, simulation of dynamic systems and customized block libraries. In turn, MATLAB provides the development of algorithms in textual programming form. The EMS-templates contain information about the CEMS structure, which is transformed into block diagrams within a Simulink model. This is reached by a set of transformation rules implemented in Atlas Transformation Language (ATL) and Eclipse Modeling Framework (EMF). EMF implements the key aspects of MDE practices by exploiting the facilities provided by Eclipse tools. Furthermore, the EMS-ABox being in RDF format, a conversion from RDF into a format to be processed and understood by EMF is required, and this is reached by employing the EMFTriple project. As a sequel, a transformation from the EMF model into a MATLAB/Simulink model (M2T) is automatically generated by the Massif framework for Eclipse, resulting in the model depicted in Figure 11, which was tailored and customized by the control engineer. A matching between knowledge regarding EMS structure in Table 10 and the Simulink/model is resulted.

An extra amount of work realized by the control engineer is the design of control strategies. The behavior to be performed within PUC{Linear-Control, Limit_SoC, ...} is not considered within the EMS-templates, and this is carried out in MATLAB/Simulink by customized block libraries and MATLAB-scripts. For instance, PUC(Linear-Control) is performed by a state-flow diagram in a Simulink model. As previously stated, PUC(Linear-Control) follows the control behavior defined in the Italian standard [48]. The mechanisms to be followed due to under-frequency events are illustrated in Figure 12. When the event $[ECPNomHz - Fgrid \geq DeltaHzStr]$ takes place, active power is injected into the grid. Power injected depends linearly on the frequency, and it is calculated in the state *Ramp*. A return to normal condition is reached when the event $[ECPNomHz - HzStop < Fgrid]$ is raised. *ECPNomHz* is the nominal frequency, and *DeltaHzStr* and *HzStop* represent deviations from *ECPNomHz*. See Table 14.

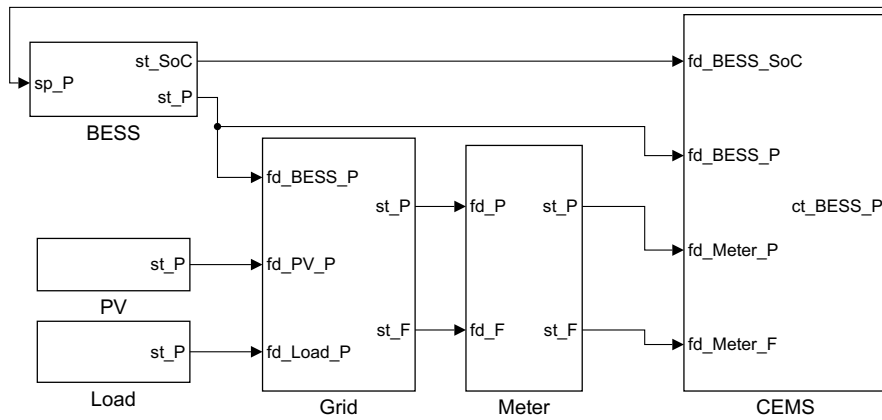


Figure 11. Structure of the CEMS-Simulink model generated.

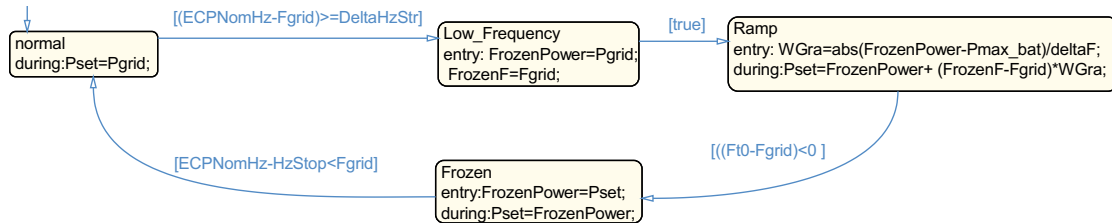


Figure 12. Behavior of the PUC(Linear-Control).

The information regarding conflict resolution exposed in Table 17 is transformed into a block diagram as shown in Figure 13a. However, not all the connections and inputs are generated. Some of them are manually included by the control engineer—for instance, the input fd_{FW_Ena} within the SelfC-block. The need for that input was realized during the running of simulations.

Besides block generation, the function code embedded within the handling-block is also generated as shown in Figure 13b. The function CI retrieves the two control signals sent by PUC{SelfC} (Control1{Self_BESS_P}) and PUC{FW} (Control1{FW_BESS_P}). The priorities defined within the PUCs are also retrieved by the inputs $Self_prt$ and FW_prt . Based on that, the code decides which control signal should be sent to the battery. The data depicted in the code is defined within the EMS-ABox and were retrieved by performing certain SPARQL queries such as Q_I and Q_{II} already defined in Table 9. The performing of M2T transformations is reached by Acceleo, an open-source code generator, where templates are designed according to models and a conversion from models into text is achievable.

Simulations are performed to validate the behavior of control applications within PUC{SelfC, FW}. The power balancing required to meet $P_{grid} = 0$ and $P_{bat} = \gamma F_{grid} + \theta$ is shown in Figure 14. Parameters related to PUC(FW) function are: $ECPNomHz = 50$, $DeltaHzStr = 0.04$ and $HzStop = 0.01$. Thereby, limits for over and under-frequency events are reached at 50.04 Hz and 49.96 Hz respectively, indicated by red and black dashed lines. At time 3640 s, an under-frequency event takes place, producing a discharge of the battery to inject more power into the grid. Once the frequency starts recovering, a snapshot of the instantaneous power is taken and used as a cap to set P_{bat} . This continues until the frequency value is higher than 49.99 Hz ($ECPNomHz - HzStop$). At that moment, the cap on the power output is removed. Additionally, the PUC(FW) is released and PUC(SelfC) is activated. Thereby, the battery is still discharging to support the load consumption, resulting in $P_{grid} = 0$. The control signal ct_{BESS_P} coming from the PUC(FW) to set the active power of the battery is depicted in red color. Thereby, this setpoint is followed by the battery during the under-frequency event.

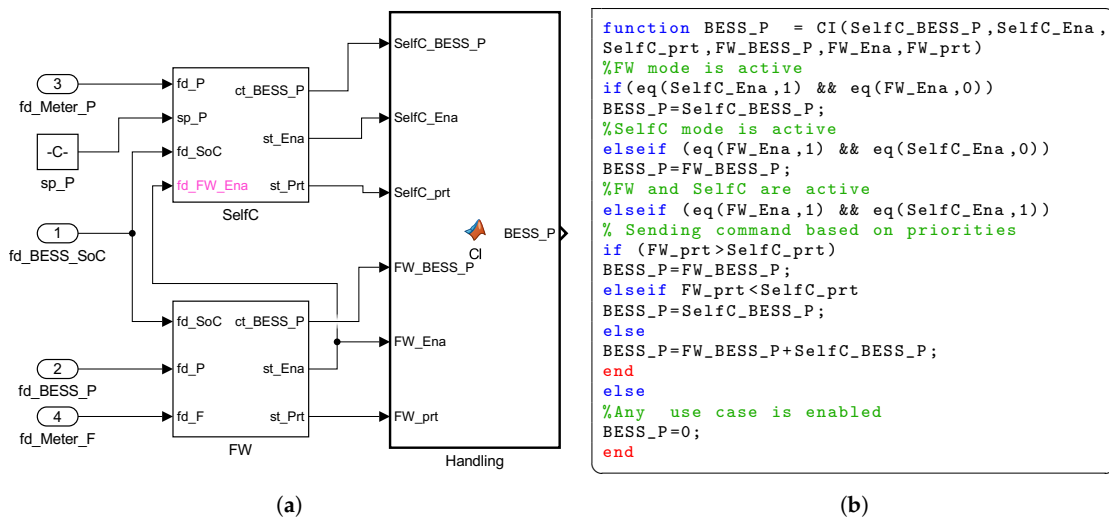


Figure 13. Software artifacts generated to be loaded within the power system emulator. (a) Simulink model generated; (b) CI function code generated.

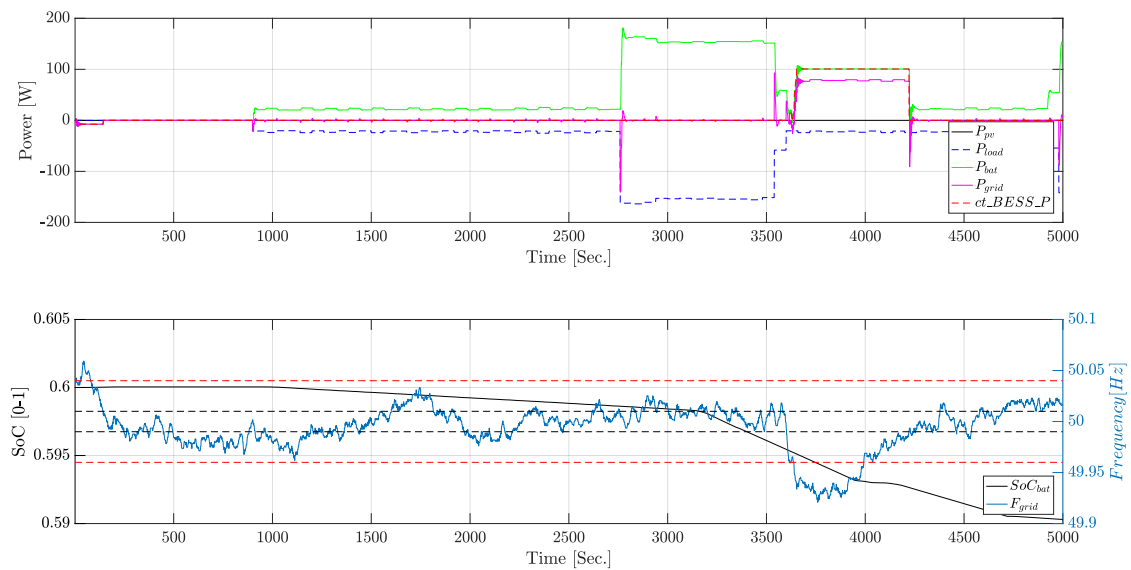


Figure 14. Offline simulation of the CEMS use case to achieve FW and SelfC.

5.3. Laboratory Implementation and Results

The main objective of the laboratory implementation is to validate the executable software artifacts generated by EMSOnto and also to evaluate the benefits and limitations of EMSOnto during the proof-of-concept with real hardware devices. It is important to highlight that performance of control strategies (HLUC {SelfC, FW}) is not the main focus. A representation of the laboratory setup is illustrated in Figure 15. In order to keep a simplicity in the validation experiments, only the battery and battery inverter are represented by real hardware. In this context, a Tesla Powerwall and Sunny boy storage 2.5 are employed. The other components such as smart meter, PV generator, and loads are simulated within the power system emulator (Simulink/MATLAB) as well as the control strategies.

The battery inverter communicates battery status (SoC_{bat} , P_{bat}) by Modbus TCP/IP. The setting up of active power within the battery (P_{ref}) is done also by Modbus TCP/IP through the Sunny Home

Manager, a control center for energy management. An interface between the power system emulator and the real hardware is needed to get and set data. Thereby, a Java program that carries out two Modbus clients and a MATLAB API is implemented. Modbus clients are configured to exchange data with the real devices. In turn, a library that enables Java programs to pass data from MATLAB to Java and vice versa supports the communication between the Java program and the Simulink model (CEMS). The hardware used in the laboratory setup is shown in Figure 16.

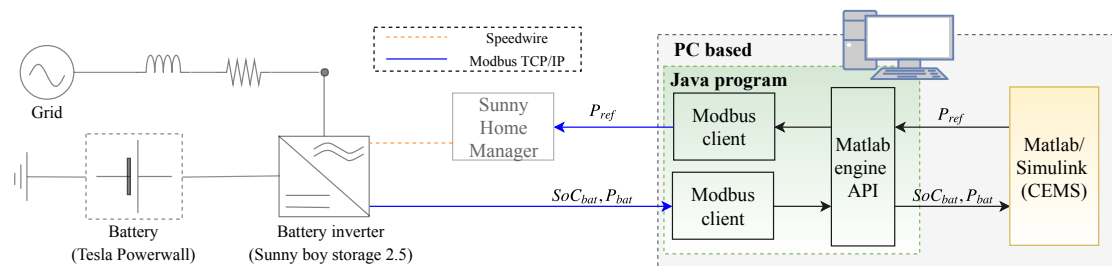


Figure 15. Laboratory setup for the validation of CEMS use case.



Figure 16. Laboratory setup with the Tesla Powerwall and the Sunny boy storage 2.5.

The results obtained from the laboratory tests are shown in Figure 17. In order to show the behavior of CEMS, a specific frequency profile with over-frequency occurrences is built. The FW mode parameters are set to $ECPNomHz = 50$, $DeltaHzStr = 0.3$ and $HzStop = 0.1$. As a consequence, as soon as the frequency exceeds 50.3 Hz ($ECPNomHz + DeltaHzStr$), the Tesla Powerwall starts taking power from the utility grid. The active power charged into the Tesla battery depends linearly on the frequency values. This dynamic continues until the battery reaches its technical limitations at 2.5 kW (Sunny boy storage 2.5 supports a maximum charge of 2.5 kW). The values of 50.3 Hz and 50.1 Hz are represented by magenta and black dashed lines, respectively.

FW mode is not requested more when frequency gets lower than 50.1 Hz ($ECPNomHz + HzStop$). Thereby, at 40,640 s, the FW mode is deactivated and self-consumption conducts the battery behavior. Hence, a PI control drives the value of P_{grid} towards zero. As mentioned before, only the battery is considered as real-device. Thus, P_{bat} and SoC_{bat} are real-values, and other values such as P_{load} , P_{pv} , P_{grid} , F_{grid} were simulated. Communication delays and the time needed by the battery to provide the expected setpoint (P_{ref}) were considered in the synchronization between the Java program, the Simulink model and the real devices.

The lab tests point out the benefits of EMSOnto at the proof-of-concept stage and motivate the extension of EMSOnto features. In that sense, benefits lie in the formal structure behind a UC representation. Thereby, the parameters such as $HzStop$ and $DeltaHzStr$ to be tuned within use cases were rapidly identified. On the other hand, conception of EMSOnto tackles information and function domain but not communication domain, which should be also supported. In that sense, manual work performed by control engineers to implement communication interfaces may be reduced by extending EMSOnto. For instance, an automatic generation and configuration of Modbus clients could be achieved. On top of that, the mapping of variables between MATLAB/Simulink and Modbus clients could also be covered. In addition to this, improvements such as the extension of the battery model presented in Table 13 with the information of Modbus registers are highly recommended.

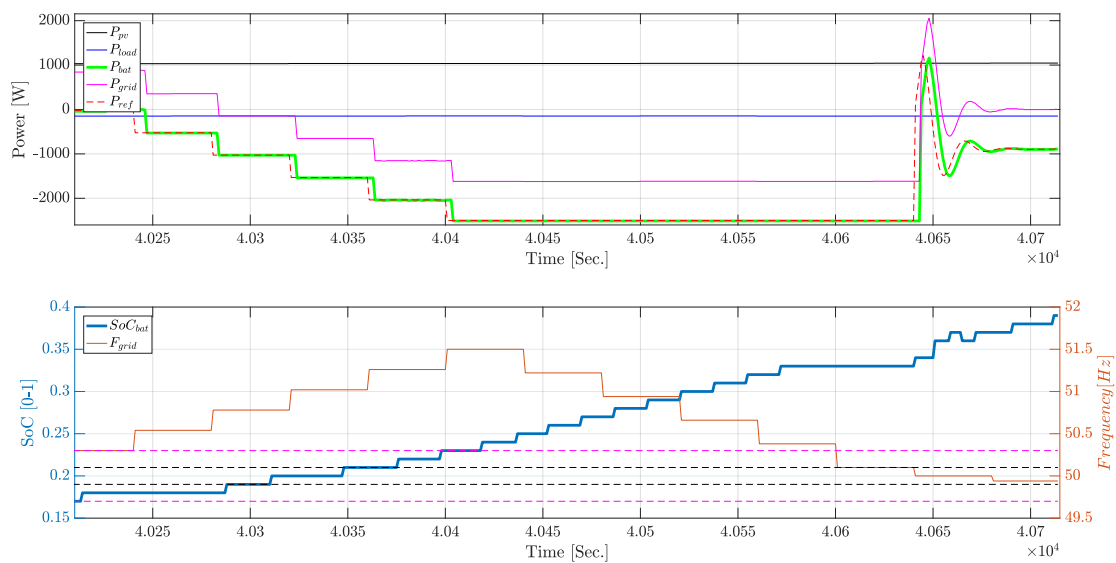


Figure 17. Real measurements (P_{bat} , SoC_{bat}) and simulated values to perform FW and SelfC.

5.4. Evaluation of Requirements and Open Issues

EMSOnto is conceived under the fulfillment of requirements R1–R4. As a result, this approach claims to support control engineers during the design of EMS applications. In order to demonstrate and evaluate it, a use case example (CEMS) is designed and validated. This section analyzes how far the initial stated requirements (R1–R4) were satisfied.

Repository based on SGAM and IEC 61850. R1 requires benefitting from existent smart grid data models. Hence, EMS-ontology is conceived under the standards IEC 61850 and SGAM. This OWL ontology defines the structure of spreadsheet templates, which facilitate the creation of pre-defined models (e.g., battery inverter [19] and frequency-watt [12]) to be stored within the IED and use case repository. The referred repository is built and available to the control engineer to start the design of EMS applications. In the CEMS example, the PUC(Logical-Control) uses knowledge from a FWHZ Logical Node accelerating the collection of CEMS data.

EMS-ontology and EMS-templates. R2 searches for an ontology that reflects a common understanding of multi-functional BESS. Thereby, a pattern that models different BESS use cases (UC1–UC5) is established resulting in EMS-ontology. The proposed OWL ontology was used to detail functions, parameters and information flows within the CEMS use case. This demonstrates that roles and concepts defined in the scope of EMS-ontology are appropriate for EMS description. Population of the ontology is achieved by filling out EMS-templates. Control engineers became familiar with this method quite fast, demonstrating that it is a good practice in gathering and documenting of data at the design phase. A feature not supported by the approaches presented in Table 2.

A benefit derived from ontologies is the extraction of data from *ABox* by means of queries. In that sense, once the filling process of EMS-templates is complete, a set of reports to determine EMS functional capabilities can be queried by an external system (e.g., EMO) (see Table 9). However, it is worth noting that information in the design phase may evolve during the proof-of-concept phase (e.g., a signal *fd_FW_Ena* was included during the validation of the CEMS). Hence, the final state of the EMS knowledge may differ from the information gathered at design time. Thus, a reverse engineering process [49] between the design and proof-of-concept stage is suggested to keep EMS data updated.

As mentioned before, the conception of EMS-ontology is aligned to the broadly recognized SGAM and IEC 61850 data models. Nevertheless, interoperability between EMSOnto and information models such as CIM and OPC UA should also be considered since they are strongly recommended in the semantics for smart grids [50].

Detection and Handling of Conflicts. R3 requires the implementation of the study [38] to support the design of EMS. To achieve the aforementioned, EMS-ontology and the OWL ontology proposed in [38] are aligned. Furthermore, EMS-templates are suggested to gather information mainly important for conflict identification. As a consequence, an error-free EMS-*ABox* is achieved before starting any implementation of the control applications. This was exemplified by the CEMS, where conflicts C_I and C_V were detected and the handling solution proposed by EMSOnto was considered in order to correct the CEMS design. Nevertheless, other kinds of inconsistencies not covered by EMSOnto may be raised during the EMS operation such as an erroneous sensor (i.e., smart meter), a measurement signal with a low sampling rate or the setting up of out-of-range values into the BESS inverter. Those inconsistencies were not considered by EMSOnto, in order to detect them an observation of states over time is necessary. This would require bringing the concept of temporal logic and concrete logic into description logic as performed in [51].

Code and Model Generation. Generation of software artifacts that support the proof-of-concept and implementation phases are required in R4. Hence, code important to the proof-of-concept stage is automatically generated. This involves the creation of M2M and M2T transformations from EMS-templates into a specific platform tool. On that basis, MATLAB code and a Simulink model were generated and upload into a MATLAB/Simulink project for CEMS validation purposes. A benefit from that is the reduction of errors susceptible to be introduced during the implementation of control applications. Moreover, the software artifacts generated were integrated in offline simulations and hardware-in-the-loop tests, contributing to an acceleration of the development process.

Code and models generation were imported in a power system emulator (i.e., MATLAB/Simulink). Nevertheless, other tools such as co-simulators and communication network simulators are also important during the proof-of-concept stage. Thereby, the adaptability of EMSOnto with other tools is contemplated as future work.

A complete EMS implementation requires the establishment of control algorithms' behavior. However, it was not the aim of EMSOnto to model the behavior of BESS use cases. A comparison and analysis of smart grid and automation approaches in Section 3.2 show that supports for implementing system behavior are available (i.e., MATLAB, IEC 61499 and IEC 61131-3). Thus, to reach a full implementation of EMS, the complement of EMSOnto with other existent smart grids and automation approaches is necessary.

6. Conclusions

A BESS provides support to many stakeholders of the grid going from end-user, distribution and transmission system operator to EMO. This support may require the deployment of a large set of use cases into an EMS. Such a development process is surrounded by different issues such as flexibility, complexity, overlapping and interoperability. Mechanisms to overcome the mentioned issues are studied in this paper; as a result, a framework (i.e., EMSOnto) to support control engineers during the design and proof-of-concept of EMS control applications is achieved.

A common understanding about EMS control applications is reached, resulting in the conception of an OWL ontology called EMS-ontology, which focuses on EMS structure and information exchanged across the EMS control architecture. Thereby, EMS-ontology is inspired from broadly accepted smart grid approaches: SGAM and IEC 61850. On the other hand, the population of the ontology is achieved by filling out spreadsheet templates (i.e., EMS-templates). Those templates ease the gathering of information at the design phase, a feature that is not provided by any of the approaches (see Table 2). Moreover, due to EMS-templates being aligned with smart grid standards, a selection of existing data models (e.g., LN from IEC 61850) was carried out to pre-fill a database that assists during the collection of EMS knowledge (use case and IED repository). It is worth mentioning that EMS-templates do not contemplate the behavior of control applications. Thus, EMSOnto needs to be supported by automation approaches such as IEC 61131-3 and IEC 61499 to achieve a full implementation.

Detection of conflicts within BESS control applications was already addressed in [38]. However, control engineers were not encouraged to use the referred study due to the amount of work involved at the design phase. They argued that conflict issues could be handled by a meticulous study of the control application. This encouraged the implementation of [38] within EMSOnto as it contains important knowledge of EMS that can be used for the inference of conflicts.

A UC example (i.e., CEMS) is developed under the EMSOnto basis showing the benefits and restrictions of the framework. Moreover, from that implementation, a list of recommendations for future work came out. Hence, it is suggested to interoperate EMSOnto with other power system tools, in order to perform reverse engineering methodologies for the consistency between the design and implementation phases. Additionally, the use of temporal logic to investigate other kinds of inconsistencies is also encouraged.

Specific planned future work concentrates its efforts in providing flexibility to EMSOnto and to investigate mechanisms for automating the development process. This contemplates enlarging the inference of knowledge and generating software artifacts relevant for the implementation stage. Moreover, information sources available at the specification phase such as IED configuration files, information models, etc. will be exploited to enhance the collection of data within the EMS-ABox.

Author Contributions: C.Z. wrote the paper and carried out the conceptualization, investigation and validation of the work. F.P.A., J.K., and T.I.S. participated in the conceptualization of the proposed approach and reviewed the final manuscript. T.I.S. supervised the overall work.

Funding: This work is partly supported by the Austrian Ministry for Transport, Innovation and Technology (bmvit) and the Austrian Research Promotion Agency (FFG) under the ICT of the Future Programme in the MESSE project (FFG No. 861265).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. TBox of the EMS Ontology

$$\begin{aligned}
 TBox = \{ & System \sqsubseteq System \sqcap \exists hasApplication.Application \\
 & Application \sqsubseteq Application \sqcap \exists hasHLUC.HLUC \\
 HLUC \sqcup PUC \sqsubseteq UC, & HLUC \sqsubseteq HLUC \sqcap \exists hasPUC.PUC \\
 BESS \sqcup EMS \sqcup Load \sqcup Meter \sqcup Generator \sqsubseteq & Application \\
 PQ \sqcup MSC \sqcup PS \sqsubseteq & HLUC \\
 PUC \sqsubseteq PUC \sqcap \exists hasPUC.PUC \sqcap \exists & manipulate.ControlGoal \\
 ControlGoal \sqsubseteq ControlGoal \sqcap \exists IsFcLinkFc.ControlGoal \sqcap & \exists hasFctVar.Variable \\
 \exists hasVariable.T \sqsubseteq (Application \sqcup HLUC \sqcup PUC) &
 \end{aligned}$$

$$\begin{aligned}
& T \sqsubseteq \forall \text{hasVariable.Variable}, \quad T \sqsubseteq \forall \text{hasParam.Param}, \quad T \sqsubseteq \forall \text{hasStatus.Status} \\
& T \sqsubseteq \forall \text{hasControl.Control}, \quad T \sqsubseteq \forall \text{hasSetpoint.Setpoint}, \quad T \sqsubseteq \forall \text{hasFeedback.Feedback} \\
& \text{hasParam} \sqsubseteq \text{hasVariable}, \quad \text{hasStatus} \sqsubseteq \text{hasVariable}, \quad \text{hasControl} \sqsubseteq \text{hasVariable} \\
& \quad \text{hasSetpoint} \sqsubseteq \text{hasVariable}, \quad \text{hasFeedback} \sqsubseteq \text{hasVariable} \\
& \text{Variable} \sqsubseteq \text{Variable} \sqcap \exists \text{IsVarLinkVar.Variable} \sqcap \exists \text{hasVarValue.VariableValue} \\
& T \sqsubseteq \forall \text{IsVarLinkVar.Variable}, \quad \exists \text{IsVarLinkVar.T} \sqsubseteq \text{Variable} \\
& \text{hasFctVar} \circ \text{IsVarLinkVar} \circ \text{hasFctVar}^{-1} \sqsubseteq \text{IsFcLinkFc} \\
& \quad \text{trans}(\text{IsVarLinkVar}), \quad \text{trans}(\text{IsFcLinkFc}) \\
& \text{manipulate} \circ \text{IsFcLinkFc} \sqsubseteq \text{manipulate}, \quad \text{hasFctVar} \circ \text{IsVarLinkVar} \sqsubseteq \text{hasFctVar} \\
& \text{Feedback} \sqcup \text{Setpoint} \sqsubseteq \text{Input}, \quad \text{Control} \sqcup \text{Status} \sqsubseteq \text{Output}, \quad \text{Input} \sqcup \text{Output} \sqsubseteq \text{External} \\
& \quad \text{State} \sqcup \text{Manipulated} \sqcup \text{Param} \sqsubseteq \text{Internal}, \quad \text{Internal} \sqcup \text{External} \sqsubseteq \text{Variable} \\
& \exists \text{IsAssignedTo.T} \sqsubseteq \text{Internal} \sqcup \text{Output}, \quad T \sqsubseteq \forall \text{IsAssignedTo.}(\text{Output} \sqcup \text{Input}) \\
& \quad \text{Numeric} \sqcup \text{Binary} \sqcup \text{Char} \sqsubseteq \text{VariableValue} \\
& (\text{hasName} \sqcup \text{hasDescription}) \text{ keyfor } T, \quad \text{hasType} \text{ keyfor } (\text{UC} \sqcup \text{Application} \sqcup \text{Variable}) \\
& \quad (\text{hasUnit} \sqcup \text{hasFormat} \sqcup \text{hasMax} \sqcup \text{hasMin} \sqcup \text{hasAnaValue}) \text{ keyfor } \text{Numeric} \\
& \text{hasTimeStamp} \text{ keyfor } \text{VariableValue}, \quad \text{hasCharValue} \text{ keyfor } \text{Char}, \quad \text{hasDiValue} \text{ keyfor } \text{Binary} \\
& P \sqcup Q \sqsubseteq \text{State}, \quad \text{ParamDevice} \sqcup \text{ParamUC} \sqcup \text{ParamCont} \sqsubseteq \text{Param}, \quad \text{Pmax} \sqcup \text{Pmin} \sqcup \text{BattAh} \sqsubseteq \text{ParamDevice} \\
& \quad \text{UCAh} \sqcup \text{UCSoC} \sqcup \text{Ena} \sqcup \text{Priority} \sqcup \text{BessSize} \sqsubseteq \text{ParamUC}, \quad \text{SettlingTime} \sqcup \text{AvailTime} \sqsubseteq \text{ParamCont} \}
\end{aligned}$$

References

1. Tayyebi, A.; Bletterie, B.; Kupzog, F. Primary Control Reserve and Self-Sufficiency Provision with Central Battery Energy Storage Systems. In Proceedings of the 2017 Conference on Sustainable Energy Supply and Energy Storage Systems (NEIS), Hamburg, Germany, 21–22 September 2017; pp. 21–22.
2. Braam, F.; Diazgranados, L.M.; Hollinger, R.; Engel, B.; Bopp, G.; Erge, T. Distributed Solar Battery Systems Providing Primary Control Reserve. *IET Renew. Power Gener.* **2016**, *10*, 63–70.
3. Kathan, J. Increasing the Hosting Capacity of Photovoltaics with Electric Storage-Simulation and Hardware-in-the-Loop Concept. Master's Thesis, Vienna University of Technology, Wien, Austria, 2011.
4. EERA Joint Programme on Smart Grids-Sub-Programme 4-Electrical Energy Technologies; Technical Report D4.3 Integration of Storage Resources to Smart Grids: Possible Services, D4.4 Control Algorithms for Storage Applications in Smart Grid; EERA: Brussels, Belgium, 2014.
5. Riffonneau, Y.; Bacha, S.; Barruel, F.; Ploix, S. Optimal Power Flow Management for Grid Connected PV Systems With Batteries. *IEEE Trans. Sustain. Energy* **2011**, *2*, 309–320. [[CrossRef](#)]
6. Zanabria, C.; Pröbstl Andrén, F.; Strasser, T.I. Comparing Specification and Design Approaches for Power Systems Applications. In Proceedings of the 2018 IEEE PES Transmission and Distribution Conference and Exhibition—Latin America, Lima, Peru, 18–21 September 2018; p. 5, in press.
7. Andrén, F.; Strasser, T.; Kastner, W. Engineering Smart Grids: Applying Model-Driven Development from Use Case Design to Deployment. *Energies* **2017**, *10*, 374. [[CrossRef](#)]
8. International Electrotechnical Commission. *IEC 61850: Communication Networks and Systems for Power Utility Automation*; International Electrotechnical Commission: Geneva, Switzerland, 2010.
9. Zoitl, A.; Lewis, R. *Modelling Control Systems Using IEC 61499*; The Institution of Engineering and Technology: Stevenage, UK, 2014.
10. International Electrotechnical Commission. *IEC 61131-3: Programmable Controllers—Part 3: Programming Languages*; International Electrotechnical Commission: Geneva, Switzerland, 2012.
11. Zanabria, C.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. Towards an Integrated Development of Control Applications for Multi-Functional Energy Storages. In Proceedings of the IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 6–9 September 2016; pp. 1–4.

12. International Electrotechnical Commission. *IEC/TR 61850-90-7—Communication Networks and Systems for Power Utility Automation—Part 90-7: Object Models for Power Converters in Distributed Energy Resources (DER) Systems*; International Electrotechnical Commission: Geneva, Switzerland, 2013.
13. Von Appen, J.; Stetz, T.; Braun, M.; Schmiegel, A. Local Voltage Control Strategies for PV Storage Systems in Distribution Grids. *IEEE Trans. Smart Grid* **2014**, *5*, 1002–1009. [[CrossRef](#)]
14. Faschang, M. Rapid Control Prototyping for Networked Smart Grid Systems Based on an Agile Development Process. Ph.D. Thesis, Vienna University of Technology, Wien, Austria, 2015.
15. Faschang, M.; Schwalbe, R.; Einfalt, A.; Mosshammer, R. Controller Hardware in the Loop Approaches Supporting Rapid Prototyping of Smart Low Voltage Grid Control. In Proceedings of the IEEE PES Innovative Smart Grid Technologies, Istanbul, Turkey, 12–15 October 2014; pp. 1–5.
16. Andrén, F.; Lehfuss, F.; Strasser, T. A Development and Validation Environment for Real-Time Controller-Hardware-in-the-Loop Experiments in Smart Grids. *Int. J. Distrib. Energy Resour. Smart Grids* **2013**, *9*, 27–50.
17. International Electrotechnical Commission. *IEC Smart Grid Standardization Roadmap*; International Electrotechnical Commission: Geneva, Switzerland, 2013.
18. Uslar, M.; Specht, M.; Rohjans, S.; Trefke, J.; González, J.M. *The Common Information Model CIM: IEC 61968/61970 and 62325-A Practical Introduction to the CIM*; Springer Science & Business Media: Berlin, Germany, 2012; Volume 66.
19. International Electrotechnical Commission (IEC). *Communication Networks and Systems in Substations—Part 7-420: Communications Systems for Distributed Energy Resources (DER)—Logical Nodes*; International Electrotechnical Commission: Geneva, Switzerland, 2006.
20. Mahnke, W.; Leitner, S.H.; Damm, M. *OPC Unified Architecture*; Springer: Berlin, Germany, 2009.
21. Weilkens, T. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*; Elsevier: New York, NY, USA, 2011.
22. Tornelli, C.; Radaelli, L.; Rikos, E.; Uslar, M. WP 4 Fully Interoperable Systems. *Deliverable R4.1: Description of the Methodology for the Detailed Functional Specification of the ELECTRA Solutions*; Technical Report; European Educational Research Association: Berlin, Germany, 2015.
23. International Electrotechnical Commission (IEC). *IEC 62559-2 Use Case Methodology-Part2: Definition of the Templates for Use Cases, Actor List and Requirement List*; IEC: Geneva, Switzerland, 2015.
24. CEN-CENELEC-ETSI Smart Grid Coordination Group. *Reference Architecture for the Smart Grid*; Technical Report; CEN-CENELEC-ETSI Smart Grid Coordination Group: Brussels, Belgium, 2012.
25. Dänekas, C.; Neureiter, C.; Rohjans, S.; Uslar, M.; Engel, D. Towards a Model-Driven-Architecture Process for Smart Grid Projects. In *Digital Enterprise Design & Management*; Springer: Cham, Switzerland, 2014; Volume 261, pp. 47–58.
26. Hitzler, P.; Krotzsch, M.; Rudolph, S. *Foundations Of Semantic Web Technologies*; CRC Press: Boca Raton, FL, USA, 2009.
27. Baader, F. *The Description Logic Handbook: Theory, Implementation and Applications*; Cambridge University Press: Cambridge, UK, 2003.
28. Santodomingo, R.; Rohjans, S.; Uslar, M.; Rodríguez-Mondéjar, J.; Sanz-Bobi, M. Ontology Matching System for Future Energy Smart Grids. *Eng. Appl. Artif. Intell.* **2014**, *32*, 242–257. [[CrossRef](#)]
29. Dubinin, V.; Vyatkin, V.; Yang, C.W.; Pang, C. Automatic Generation of Automation Applications Based on Ontology Transformations. In Proceedings of the 2014 IEEE International Conference on Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–4.
30. Schachinger, D.; Kastner, W.; Gaida, S. Ontology-Based Abstraction Layer for Smart Grid Interaction in Building Energy Management Systems. In Proceedings of the 2016 IEEE International Energy Conference (ENERGYCON), Leuven, Belgium, 4–8 April 2016; pp. 1–6.
31. Zhou, Q.; Natarajan, S.; Simmhan, Y.; Prasanna, V. Semantic Information Modeling for Emerging Applications in Smart Grid. In Proceedings of the 2012 Ninth International Conference on Information Technology—New Generations, Las Vegas, NV, USA, 16–18 April 2012; pp. 775–782.
32. Samirmi, F.D.; Tang, W.; Wu, Q. Fuzzy Ontology Reasoning for Power Transformer Fault Diagnosis. *Adv. Electr. Comput. Eng.* **2015**, *15*, 107–114. [[CrossRef](#)]
33. Feldmann, S.; Herzig, S.J.; Kernschmidt, K.; Wolfenstetter, T.; Kammerl, D.; Qamar, A.; Lindemann, U.; Krcmar, H.; Paredis, C.J.J.; Vogel-Heuser, B. Towards Effective Management of Inconsistencies in Model-Based Engineering of Automated Production Systems. *IFAC-PapersOnLine* **2015**, *48*, 916–923. [[CrossRef](#)]

34. Brambilla, M.; Cabot, J.; Wimmer, M. Model-Driven Software Engineering in Practice. *Synth. Lect. Softw. Eng.* **2012**, *1*, 1–182. [[CrossRef](#)]
35. Strasser, T.; Rooker, M.; Ebenhofer, G.; Hegny, I.; Wenger, M.; Sündler, C.; Martel, A.; Valentini, A. Multi-Domain Model-Driven Design of Industrial Automation and Control Systems. In Proceedings of the 2008 IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Hamburg, Germany, 15–18 September 2008; pp. 1067–1071.
36. Andrén, F.; Strasser, T.; Kastner, W. Model-Driven Engineering Applied to Smart Grid Automation Using IEC 61850 and IEC 61499. In Proceedings of the Power Systems Computation Conference (PSCC), Wroclaw, Poland, 18–22 August 2014; pp. 1–7.
37. Zanabria, C.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. Approach for Handling Controller Conflicts within Multi-Functional Energy Storage Systems. *CIREC-Open Access Proc. J.* **2017**, *2017*, 1575–1578. [[CrossRef](#)]
38. Zanabria, C.; Tayyebi, A.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. Engineering Support for Handling Controller Conflicts in Energy Storage Systems Applications. *Energies* **2017**, *10*, 1595. [[CrossRef](#)]
39. Sirin, E.; Parsia, B.; Grau, B.C.; Kalyanpur, A.; Katz, Y. Pellet: A Practical OWL-DL Reasoner. *Web Semant. Sci. Serv. Agents World Wide Web* **2007**, *5*, 51–53. [[CrossRef](#)]
40. Working Group Sustainable Processes (SG-CG/SP). *CEN-CENELEC-ETSI Smart Grid Coordination Group—Sustainable Processes*; Technical Report, CEN-CENELEC-ETSI; Working Group Sustainable Processes (SG-CG/SP): Brussels, Belgium, 2012.
41. International Electrotechnical Commission. *Communication Networks and Systems for Power Utility Automation. Part 7-3: Basic Communication Structure-Common Data Classes*; International Electrotechnical Commission: Geneva, Switzerland, 2011.
42. Lutz, C.; Areces, C.; Horrocks, I.; Sattler, U. Keys, Nominals, and Concrete Domains. *J. Artif. Intell. Res.* **2005**, *23*, 667–726.
43. Tremblay, O.; Dessaint, L.A.; Dekkiche, A.I. A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles. In Proceedings of the 2007 IEEE Vehicle Power and Propulsion Conference, Arlington, TX, USA, 9–12 September 2007; pp. 284–289.
44. Skoplaki, E.; Palyvos, J. On the Temperature Dependence of Photovoltaic Module Electrical Performance: A Review of Efficiency/Power Correlations. *Sol. Energy* **2009**, *83*, 614–624. [[CrossRef](#)]
45. HELIOCLIM-3. 2005. Available online: <http://www.soda-pro.com/web-services/radiation/helioclim-3-archives-for-free> (accessed on 1 January 2015).
46. Autonomous Decentralised Renewable Energy Systems-ADRES. 2009. Available online: https://www.ea.tuwien.ac.at/projects/adres_concept/EN/ (accessed on 1 January 2015).
47. ENTSO-E Netzfrequenz. 2018. Available online: <http://www.50hertz.com/de/Maerkte/Regelenergie/Regelenergie-Downloadbereich> (accessed on 15 June 2018).
48. Norma Italiana, CEI. *Reference Technical Rules for the Connection of Active and Passive Users to the LV Electrical Utilities (in Italian)*; Norma Italiana, CEI: Milano, Italy, 2012.
49. Martinez-Gil, J.; Aldana-Montes, J.F. Reverse Ontology Matching. *ACM SIGMOD Rec.* **2011**, *39*, 5. [[CrossRef](#)]
50. Rohjans, S.; Uslar, M.; Juergen Appelpath, H. OPC UA and CIM: Semantics for the Smart Grid. In Proceedings of the Transmission and Distribution Conference and Exposition, 2010 IEEE PES, New Orleans, LA, USA, 19–22 April 2010; pp. 1–8.
51. Baader, F.; Lippmann, M. Runtime Verification Using the Temporal Description Logic ALC-LTL Revisited. *J. Appl. Log.* **2014**, *12*, 584–613. [[CrossRef](#)]



2.2 Publication B

C. Zanabria, F. Prössl Andrén , and T. I. Strasser.

An Adaptable Engineering Support Framework for Multi-Functional Energy Storage System Applications.

Sustainability, vol. 10, no. 11, 28 pages, 2018.

Own contribution

The problem analysis, selection of an adequate methodology, implementation of a research prototype, collection of use cases, implementation of those use cases for validation purposes, and discussions of the methodology were performed by the applicant. The structuring, writing, and editing of the manuscript were also carried out by the applicant. The second and the last author participated in the conceptualization of the methodology. Moreover, they reviewed the manuscript and supervised the overall work.



Article

An Adaptable Engineering Support Framework for Multi-Functional Energy Storage System Applications

Claudia Zanabria ^{1,*}, Filip Pröbstl Andrén ¹ and Thomas I. Strasser ^{1,2}

¹ Center for Energy—Electric Energy Systems, AIT Austrian Institute of Technology, 1210 Vienna, Austria; Filip.Proestl-Andren@ait.ac.at (F.P.A.); Thomas.Strasser@ait.ac.at (T.I.S.)

² Institute of Mechanics and Mechatronics, Vienna University of Technology, 1040 Vienna, Austria

* Correspondence: claudia.zanabria@ait.ac.at

Received: 17 September 2018; Accepted: 31 October 2018; Published: 12 November 2018



Abstract: A significant integration of energy storage systems is taking place to offer flexibility to electrical networks and to mitigate side effects of a high penetration of distributed energy resources. To accommodate this, new processes are needed for the design, implementation, and proof-of-concept of emerging storage systems services, such as voltage and frequency regulation, and reduction of energy costs, among others. Nowadays, modern approaches are getting popular to support engineers during the design and development process of such multi-functional energy storage systems. Nevertheless, these approaches still lack flexibility needed to accommodate changing practices and requirements from control engineers and along the development process. With that in mind, this paper shows how a modern development approach for rapid prototyping of multi-functional battery energy storage system applications can be extended to provide this needed flexibility. For this, an expert user is introduced, which has the sole purpose of adapting the existing engineering approach to fulfill any new requirements from the control engineers. To achieve this, the expert user combines concepts from model-driven engineering and ontologies to reach an adaptable engineering support framework. As a result, new engineering requirements, such as new information sources and target platforms, can be automatically included into the engineering approach by the expert user, providing the control engineer with further support during the development process. The usefulness of the proposed solution is shown with a selected use case related to the implementation of an application for a battery energy storage system. It demonstrates how the expert user can fully adapt an existing engineering approach to the control engineer's needs and thus increase the effectiveness of the whole engineering process.

Keywords: energy management system; energy storage system; semantic web technologies; rules; ontology; engineering support; smart grid architecture model; model driven architecture; IEC 61850; IEC 61499

1. Introduction

The reduction of CO₂ emissions is motivating the integration of renewables into power grids. As a consequence, a higher penetration of distributed generators such as Photovoltaic (PV), wind turbines, and small hydro power stations is taking place [1]. A side effect of this is the perturbation of the power system stability and quality. Those issues were addressed by different studies [2], which encourage the use of Battery Energy Storage Systems (BESS). Moreover, BESS can also support services related to the minimization of supply costs and market integration, among others [3]. Consequently, BESS will play a multi-functional role in the near future. The BESS is often accompanied by an Energy Management Systems (EMS) where the BESS's services and functions are hosted. Hence, the EMS should exchange information with stakeholders and Intelligent Electronic

Devices (IEDs) spread out over the electric grid. Thereby, the EMS development process should consider interoperability across systems as well as evolving requirements of smart grid systems. In this context, the realization of EMS involves challenging tasks, such as alignment with smart grid information models, conflicts resolution within a multi-functional system, as well as handling a diversity of tools for EMS validation. Different approaches handle these issues to different degrees as demonstrated by Zanabria et al. [4]. Nevertheless, a full flexibility within a rapid prototyping context is still not covered by the mentioned approaches. This motivates the open issues addressed in this work. An outlook of them is given below.

At the design phase, control engineers gather documents that encapsulate information regarding IED capabilities, network topologies, control applications structure, etc. Those documents are considered to provide important input for the EMS design. Thereby, a methodology that supports an automated treatment of this information to support the development process is necessary. Santodomingo et al. [5] process models based on the Smart Grid Architecture Model (SGAM) [6] and Use Cases defined with the IEC 62559 [7] standard to support network operators in selecting adequate technical solutions for their business needs. The referred study shows attempts to benefit from available information sources (SGAM models) in an automated way.

Another approach, the so called Energy Management System Ontology (EMSOnto) [8] also proposes a resolution of conflicts within an EMS [9]. Nevertheless, inconsistencies detected within BESS applications imply not only conflicts across Use Cases (UC) but also others such as incompatibility between a BESS and a service, misconfigured units, wrong write/read access permissions, etc. This motivates to look for mechanisms that enable a customized identification of inconsistencies based on evolving engineering requirements.

What is more, due to control engineering practice and legacy solutions a variety of tools, programming languages and documents are likely to be employed during the validation phase. Current approaches [4] do not answer those requirements since models and code generated were tied to a specific platform. However, higher flexibility is expected in terms of software artifacts generation. For instance, EMSOnto automatically generates code to be employed in a power system emulator (e.g., MATLAB/Simulink). Here, the compatibility with only one specific platform is provided. On the other hand, the rapid engineering methodology called Power System Automation Language (PSAL) generates models compatible with IEC 61499 [10]. However, also here, the generation of models in other specific platforms rather than IEC 61499 is not guaranteed.

EMSOnto is one of the more suitable approaches that addresses the above mentioned issues in a holistic manner. EMSOnto guides and supports control engineers during the design, proof-of-concept and implementation stages of BESS services and applications. Consequently, this work considers EMSOnto as reference framework to demonstrate how a flexible and automated engineering process can be attained. By applying the outcomes of the current work to EMSOnto, an improved version of it will be reached. Moreover, the proposed methodology may also be used to reinforce other modern approaches such as PSAL and SGAM.

Since EMSOnto is taken as reference approach, the role of a new actor, the so called EMSOnto expert user, is introduced to answer the exposed open issues. Thus, concepts from Model-Driven Engineering (MDE) and ontologies are combined to offer an adequate solution. The evaluation of the new capabilities of EMSOnto is performed by the realization of a selected use case example. As a result, an acceleration in the realization of EMS applications is gained as well as a further identification of inconsistencies. Moreover, the achieved engineering process is flexible enough to be aligned with different software platforms.

This paper is organized as follows: Section 2 outlines the motivation of this work, the open issues to be addressed, and the role of control engineers in the scope of EMSOnto as well. In Section 3 mechanisms to enhance the engineering process are introduced and discussed. A use case example is used in Section 4 to demonstrate how the control engineer's requirements are precisely addressed.

Thereafter, in Section 5, a UC example is realized with EMSOnto to evaluate and analyze its new features. Finally, Section 6 addresses the conclusions and discusses future needs.

2. ESS Application Development Process Using Modern Engineering Approaches

This section defines the scope of the current work and outlines the engineering process of multi-functional storage systems. Furthermore, an overview of different engineering approaches that support the development process of BESS applications is also presented. Subsequently, issues not yet addressed by those approaches are highlighted. Since EMSOnto is considered to be the most adequate approach, the engineering process under the basis of EMSOnto is presented. Furthermore, the open issues and research goals under the frame of EMSOnto are defined.

2.1. Realization of Multi-Functional ESS Applications

BESS are being used to enhance the power quality and to maintain the power stability, among others. This involves frequency and voltage control, self-consumption, peak-shaving, etc. [3,11,12]. Those functionalities are usually embedded within an external system (e.g., EMS) that controls active and reactive power of the battery to facilitate the referred services. Thereby, a participation of different stakeholders is required, such as network operators, Energy Market Operator (EMO), or Distributed Energy Resources (DER), as outlined in Figure 1. As an example, the UC Primary Control Reserve (PCR) [11] controls the active power of a BESS to support the regulation of the grid frequency. Thereby a communication of the EMS with a smart meter, network operator and a BESS is needed. On the other hand, the UC minimization of costs with peak-shaving [12] would require a communication with a PV generator and households to measure the active power generated and consumed.

The realization of EMS applications often follows certain stages, such as specification, design, implementation, and proof-of-concept evaluation [13]. In the specification stage a definition of requirements to be satisfied by an EMS is carried out. As a sequel, a conceptual design of EMS is elaborated, this implies the definition of control strategies and communication and component architectures. The validation of the proposed control design is mainly carried out in offline power system simulators (e.g., DlgSILENT/PowerFactory) and controller platforms (e.g., IEC 61499) [14]. Often the validation process entails an iterative refinement of the control application design. Subsequently, a prototype is realized and implemented within a real hardware controller.

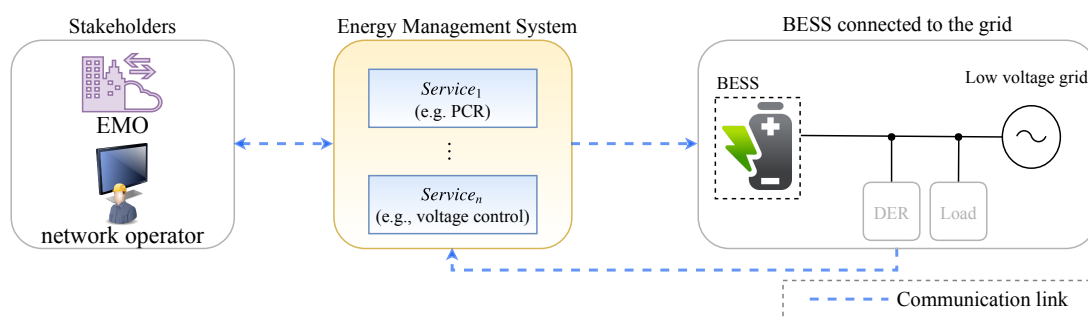


Figure 1. Framework of the BESS control applications.

2.2. Application Engineering Using Modern Approaches

The engineering process of EMS can nowadays be supported by using modern development approaches [4]. Each of them tackles different specific issues. An analysis on how those approaches support the EMS development process is addressed as follows.

An approach called IntelliGrid (IEC 62559) [15] is mainly used to specify use cases and was suggested by EPRI in 2003 to face the complexity involved within power systems. IntelliGrid is the most used standard to describe smart grid use cases. The main outcome of IntelliGrid is a set of use

case templates that guide engineers during the definition of business cases, use cases and requirements. Those templates are supported with a narrative and visual representation. The process involving the realization of IntelliGrid templates is basically a manual work. This means that information sources supporting the EMS specification (e.g., IED's models, component and communication architectures) are not automatically processed to fill out the templates. Besides this, the consistency of the information collected by the referred templates is not verified. This may lead to unattended inconsistent data at the specification stage.

On the other hand, the Unified Modeling Language (UML) is the most prominent general-purpose language to specify and design use cases in an object-oriented approach [16]. UML provides different structure (class, component, etc.) and behavior (sequence, activity, etc.) diagrams to allow a complete modeling of use cases. Nowadays, UML is getting more attention also in the domain of power system to model smart grid solutions [17]. A different approach (SGAM) conceived to model smart grid projects is recommended by [4]. SGAM provides a set of concepts, coordinated viewpoints and a method to map use cases within a smart grid model. An implementation of SGAM is achieved by the SGAM-Toolbox (SGAM-TB), a UML-based domain specific language available as an extension of the Enterprise Architecture (EA) tool [18]. A drawback of both UML and SGAM is that the information gathered at the specification phase is not automatically processed to achieve important knowledge for the design stage. This does not guarantee a consistency between specification and design stages.

A holistic approach called PSAL has as main purpose the modeling of SGAM UCs and at the same time provide automated support for rapid prototyping. To do this, PSAL provides a formal domain specific language to describe smart grid systems. PSAL stands out from the previously mentioned approaches due to the rapid generation of software artifacts (code and communication configurations) offered. However, PSAL does not identify inconsistencies within the planned design. This, in turn, is tackled by EMSOnto, an approach focused on the identification of conflicts and on the rapid prototyping of multi-functional BESS applications. Nevertheless, EMSOnto lacks flexibility due to the restricted set of inconsistencies detection offered. On the other hand, as already mentioned, EMSOnto and PSAL address the rapid generation of software artifacts. However, since the needed type of software artifacts depend on the UCs to be implemented and on needed legacy solutions, it should be possible to support multiple target platforms. This feature is neither covered by PSAL nor EMSOnto. Currently, the generated models and code using PSAL are aligned to the automation standard IEC 61499 and the communication model IEC 61850. EMSOnto currently only provides generation of code for the system simulator MATLAB/Simulink.

The mentioned open issues entail the setting of the following research goals: (i) benefit from information sources to automate the realization of design tasks, (ii) facilitate the deduction of knowledge to be customizable to control engineer's requirements, and (iii) generate software artifacts to support the whole engineering process. To meet those goals, EMSOnto is taken as the reference framework due to its completeness and holistic understanding. Hence, the solution proposed in this work has been aligned to EMSOnto but should still be applicable not only to EMSOnto but also to other existent and upcoming approaches. The next paragraphs outline EMSOnto and formulate the already mentioned open issues and research goals under the EMSOnto basis.

2.3. EMSOnto Development Process

An overview of the work performed by control engineers on the frame of EMSOnto is shown in Figure 2. At the specification phase, engineers study the requirements to be fulfilled by the EMS and starts a rough definition of the behavior and the structure of the EMS control applications. At this stage, different approaches such as SGAM, Unified Modeling Language (UML) and Systems Modeling Language (SysML) for system specification are employed [6,16].

The next step is the design of the EMS, which should be congruent with the specification already defined. To this end, EMSOnto offers spreadsheet templates (EMS-templates) that collect and structure the EMS's information. These spreadsheets have headlines with the attributes of functions and

variables created in the scope of the EMS. Moreover, models of IEDs and UCs derived from smart grids standards (e.g., Common Information Model, IEC 61850) are available within the UC and IED repository. Those pre-built models ease the population of EMS-templates and thus compatibility with existent information models is achieved.

Once the EMS-templates are completely filled a reasoner engine is executed resulting in the derivation of inferred data. The resulted data is queried to identify inconsistencies in the design, a report form gathers those inconsistencies and is handled and analyzed by control engineers. This analysis may lead to adjustments within the planned design. Therefore, at this stage EMS-templates are amended. This process is repeated until control engineers are satisfied with the resulted design. In consequence, a consistent and less error-prone EMS is achieved before any implementation. To support the proof-of-concept a set of software artifacts are made available to control engineers.

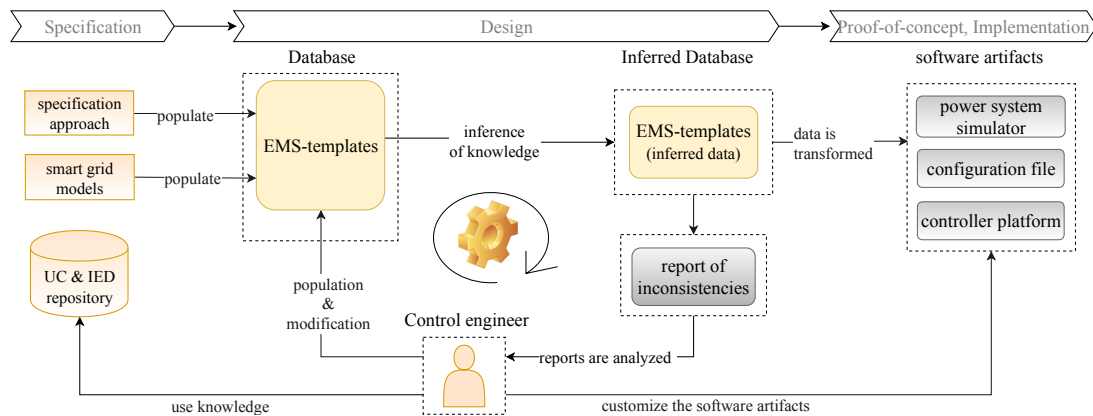


Figure 2. EMSOnto in practice by control engineers.

2.4. Open Issues

EMSOnto was put in practice by control engineers in the scope of a large range of UCs [8]. Those realizations raised a series of requirements understood as open issues within EMSOnto. They are detailed as follows:

- *Information sources are not exploited:* At the design stage control engineers dispose documents that support the design of EMS. This may correspond to files describing IED capabilities, smart grid use cases, communication networks, information models, etc. (e.g., IEC 61850, IntelliGrid, SGAM) [15,18,19]. Since those files contain requirements and important knowledge for the design process, control engineers manually need to import selected data into the EMS-templates. This repetitive manual work is time consuming and exposed to human errors. Hence, an automatic exchange between EMS-templates and other information sources is sought.
- *Restricted inference:* EMSOnto supports the identification of conflicts between use cases. However, this is not the only kind of inconsistency that would harm the suitable operation of EMS. For instance, the setting up of IED registers with a wrong unit value would also impact the correct operation. Besides this, the inference of important data to support the design of EMS's control strategies is also missed. Since knowledge to be inferred depends on engineer's needs a flexible customization of EMSOnto to enlarge inferred knowledge is desired.
- *Limited generation of software artifacts:* EMS-templates can be automatically transformed into models and code compliant with a specific power system simulator (i.e., MATLAB/Simulink). Nevertheless, software platforms to be targeted depend on best practices established for testing and validation. In the power system domain, those platforms involve controller platforms, co-simulation platforms, communication network simulators, etc. (e.g., IEC 61499, Mosaik,

OMNET++) [20–22]. Therefore, generation of software artifacts, compatible with a large set of platforms, should be guaranteed.

Summarizing, taking all the open issues into account an adaptable engineering support framework for ESS applications is required which is introduced and discussed in the following sections.

3. Mechanisms to Automate and Increase Flexibility of EMSOnto

This section introduces the role of a new actor called EMSOnto expert, whose main focus is to offer mechanisms to overcome the aforementioned gaps. A detailed list of actions performed by the expert is exposed as well as the foundations of the proposed solution.

3.1. EMSOnto Expert Participation

The main key of EMSOnto is the conception of an ontology (EMS-ontology) which gives a common understanding of EMS control applications and the process to be controlled by them (e.g., BESS, smart meter). A formal definition of the ontology is reached by Description Logics (DL) [23], a formal language to model real systems, where the classification of common individuals is done by concepts and the relations between concepts are established by roles. DL languages comprises an assertional component (*ABox*) and a terminological component (*TBox*). A *TBox* defines concepts, roles, and constraints between them. In turn, an *ABox* asserts the membership of individuals in concepts and roles. On that basis, the EMS-*ABox* is populated by the EMS-templates and the EMS-*TBox* provides a knowledge representation of EMS applications. The terms EMS-ontology and EMS-*TBox* are used interchangeably as well as EMS-templates and EMS-*ABox*. In this manuscript, concepts are indicated with the font typewriter. A new actor, the EMSOnto expert, is introduced to expand the features of EMSOnto. The actions taken by the expert are depicted in Figure 3.

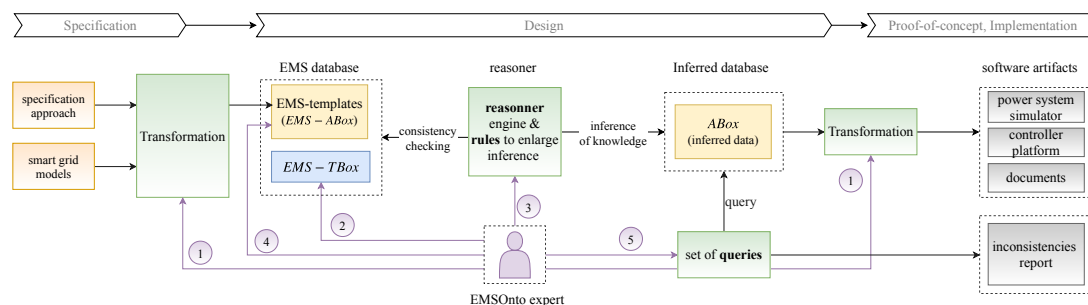


Figure 3. Tasks to be performed by EMSOnto expert.

(1) *Transformation across information*: Control engineers make use of documents which contain specifications regarding the EMS design (e.g., SGAM models), this information is gathered within the EMS-templates. Since EMS-templates are aligned to the EMS-*TBox*, the referred gathering process involves a matching between the information source, collected from specification approaches or smart grid models, and concepts and roles defined within the EMS-*TBox*. For instance, EMS’s structure represented by UML use case diagrams comprise the concepts `UseCase`, `LogicalActor`, etc. [16]. Hence, a transformation of a `UseCase` instance requires a correspondence with the concept `UC` defined within EMS-*TBox*. This is possible since the `UC` and the `UseCase` represent the same information. A similar mechanism is applied for an automatic generation of software artifacts from EMS-templates. In such a case, a transformation from EMS-*ABox*’s individuals into models, code and text compliant with specific platform environments is performed.

(2) *Adjustment of the EMS-*TBox**: Since the EMS-*ABox* should be consistent with the *TBox*, the gathering of new assertions may impact the *TBox*. This applies to knowledge not modeled in the *TBox* but required in the transformation from or to other models. On the other hand, the inference of implicit knowledge may also require an upgrade of the *TBox*. For instance, the detection of “BESS that

do not support the active power required by certain service" would require to create the concept P_{max} which symbolizes the upper limit of an active power's value.

(3) *Creation of new rules*: The reasons that motivated an extension of the EMS-*TBox* could require an improvement on the expressivity afforded by the EMS-*TBox*. This issue is tackled by rules which are used to improve the intelligence own by ontologies. Rules are composed of a premise and a conclusion, satisfaction of the premise results in the derivation of implicit data or conclusions. As a result, any requirement involving complex inferences that are not achievable by the setting up of axioms within the EMS-*TBox* would be resolved by rules.

(4) *Upgrade of EMS-templates*: As explained before, EMS-*TBox* is vulnerable to be extended due to inference of knowledge or software artifacts generation. Since an *ABox* should be consistency with a *TBox* any modification to the EMS-*TBox* would also impact the EMS-*ABox*. This means that, instantiations of new concepts and roles would require an upgrade of EMS-templates.

(5) *Set-up new queries*: The detection of inconsistencies is based on a series of questions to be asked to the inferred EMS-*ABox*. Implementation of this is based on the formulation of query templates that model well-defined questions. Thus, the inferred data resulted from action 3 is queried and processed to collect important knowledge for the inconsistencies report, as shown in Figure 3. Besides that, queries could also target inference of essential knowledge to support the EMS design.

3.2. Transformation Mechanisms and Techniques

The tasks related to transformation of information across domains will be performed by MDE techniques [24], where models are key players. Thereby, a model compliant with a UML class representation is envisaged to represent the EMS-ontology. This model will be the essential part of the model transformations carried out along this work.

3.2.1. Model-Driven Engineering in Power System Domain

The basics principles necessary to achieve the contemplated transformations between source and target data are covered by MDE approach [24]. This section outlines the concepts within MDE, benefits of the approach as well as power system applications where MDE was implemented.

The main focus of MDE is the automation of a system development process. On that basis, MDE analyzes a development process in an object-oriented manner, resulting in the conception of models. Interoperability between models is reached by mapping rules. Which are defined under Model-to-Model (M2M) or Model-to-Text (M2T) transformations, also known as code generation.

Implementation of MDE by the Object Management Group (OMG) goes under the name of Model Driven Architecture (MDA). Within MDA, the phases of a development process are supported by a Platform Independent Model (PIM) and a Platform Specific Model (PSM). PIM defines system functionality and is characterized by an independence with a specific platform solution. PIM is deployed into a concrete platform (PSM) and usually PSM is translated into software artifacts (e.g., C++, Java code). M2M supports transformation from PIM to PSM and M2T from PSM into code.

An automated engineering method to support the complete development process of smart grid control applications is provided by Prössl Andrén et al. [10]. Power System Automation Language (PSAL) is a Domain Specific Language (DSL) and supports the design of UCs compliant with SGAM. MDE techniques are used to generate executable code and communication configuration scripts from UCs specifications. This is a holistic approach that covers the full engineering process.

On the other hand, Andrén et al. [25] proposes an automatic generation of IEC 61499 compliant control applications from IEC 61850 descriptions. This approach implements M2M and M2T transformations. However, the obtained IEC 61499 system is not aligned to SGAM. Another study [26] designs a BESS control application within a power system emulator (Matlab/Simulink) for validation purposes and uses MDE to replicate the validated application into a controller platform (IEC 61499). This study proposes a rapid prototyping that covers proof-of-concept and implementation. However, other stages of the engineering process such as specification and design were not considered.

Furthermore, [18] Dänekas et al. proposes the SGAM-Toolbox (SGAM-TB) by following the MDA approach. The SGAM-TB is implemented as an extension to Enterprise Architect (EA) and therefore code generation is also possible. However, this code is not formatted to specific needs of control engineers.

Summarizing, apart from EMSOnto, PSAL is the only approach that offers a holistic approach. However, some gaps at the function and information layer of PSAL are covered by EMSOnto. Those gaps regard the identification of inconsistencies and the lack of a mechanism to gather unlimited data. Those topics were tackled in the current EMSOnto. The new version of EMSOnto is formulated with the aim of giving flexibility at all the stages of the engineering process. This flexibility is not addressed by any solution already presented and is going to conduct the research of this work.

3.2.2. UML Representation of EMS-Ontology

Since models are the main concept behind MDE, the conception of models across the full development process is a main task to be supported by the EMSOnto expert. In this light, a model of EMS-ontology is achieved and exposed in this section.

OMG recommends the formulation of models by OMG specifications such as UML and SysML, among others [27]. UML, a widely used standard for describing structure and behavior of complex systems is selected to represent the models. On the other hand, the expressiveness needed by the EMS-*TBox* is given by *SROIQ(D)*, a logic defined within DL concept that provides complex role inclusion axioms, inverse roles constructors, qualify cardinality restrictions among others [28]. Thus, mapping rules to relate *SROIQ(D)* elements to UML metamodel elements are established to conceive a UML model of the EMS-*TBox*, see Table 1. A DL concept is transformed into a UML class, a concept subsumption into a UML generalization and so on, a practical implementation of those conversions is given in [29]. It is worth mentioning that the full *TBox* cannot be expressed by a UML class diagram since not all the *TBox* axioms can be transformed into UML elements (e.g., transitivity axioms, composition role constructor). The resulted UML class diagram is presented in Figure 4 and goes under the name of EMS-Data Model (DM). Only a simplify model is depicted. InformationFlow is a new concept derived from the EMSOnto extension, it will be introduced later in the paper.

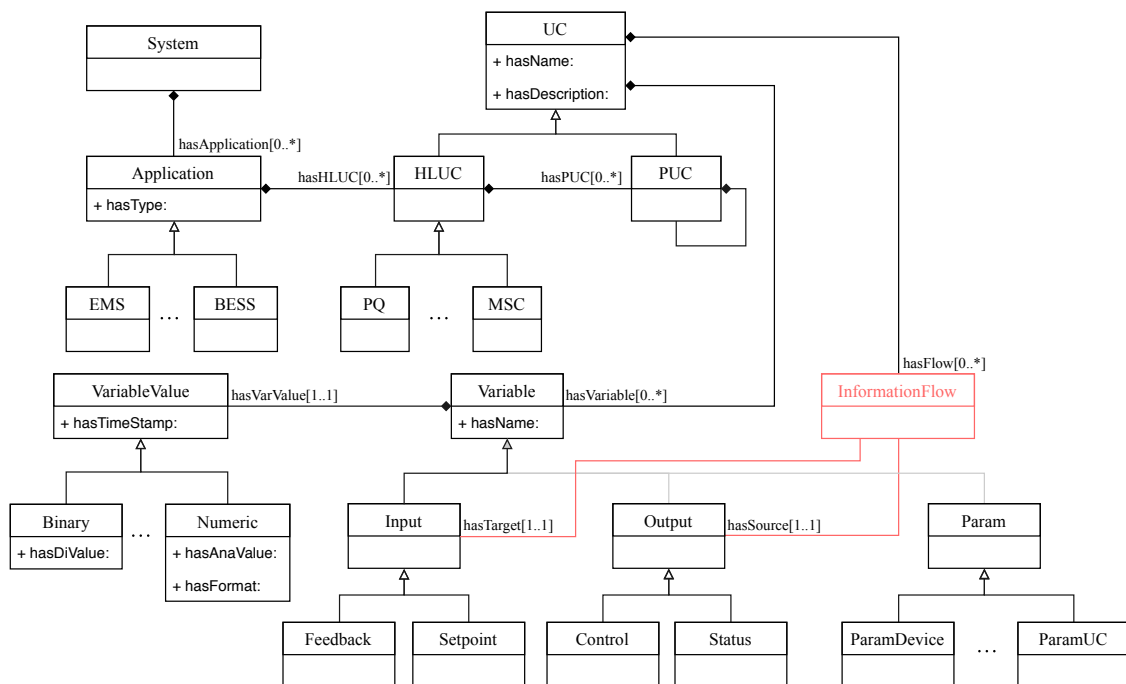


Figure 4. UML class diagram representing the EMS-*TBox* (reduced model).

Table 1. Matching between elements of *SROIQ(D)* and UML metamodel.

DL (<i>SROIQ(D)</i>)	UML
concept	class
concept subsumption (\sqsubseteq)	generalization
data type	datatype
role	association, composition, aggregation
concrete role	attribute

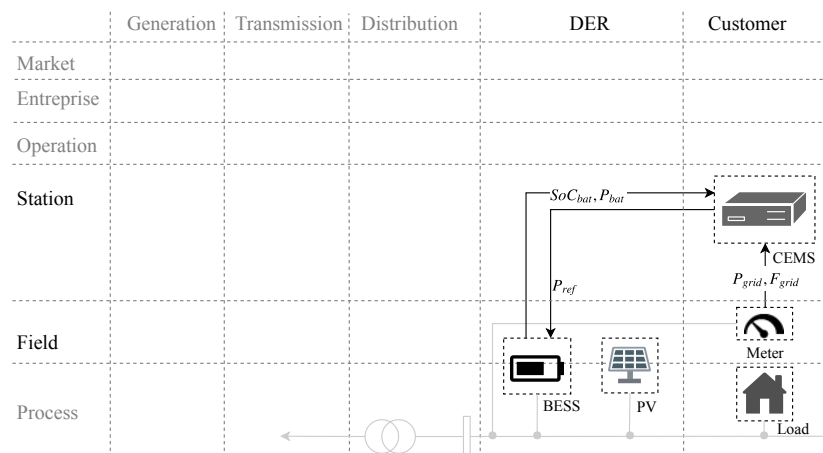
4. Analysis of a Use Case Example by an EMSOnto Expert

Different mechanisms to enlarge the capabilities of EMSOnto were proposed above. This section outlines the role of the EMSOnto expert regarding the implementation of a specific UC.

4.1. Use Case to Be Analyzed by the EMSOnto Expert

The UC Customer Energy Management System (CEMS) was implemented using EMSOnto as a basis [8]. SGAM is the preferred approach among smart grid specification approaches as demonstrated in [4]. The SGAM information layer represent system's components as well as their information flows. Moreover, it combines domain-axis and zone-axis to structure smart grid control applications. For the CEMS in this UC, a detailed model is provided as an SGAM description, as seen in Figure 5. This shows the main CEMS's actors (i.e., BESS, CEMS, Meter, etc.) and the scope of the CEMS, which is located at the customer side and at the station level.

The CEMS is embedded with Frequency-Watt (FW) and Self-Consumption (SelfC) services. Both services control the active power injected/subtracted by the BESS. To this aim, the signals State of Charge (SoC) (SoC_{bat}) and active power (P_{bat}) are retrieved from the BESS. In turn, the frequency (F_{grid}) and active power (P_{grid}) at the Point of Common Coupling (PCC) are taken from a smart meter. Generation of a PhotoVoltaic (PV) DER and the consumption of a household impact the behavior of the CEMS. Thus, they are also considered during design stage. After execution of the SelfC and FW services, a calculated active power setpoint (P_{ref}) is sent to the BESS.

**Figure 5.** SGAM information layer representation of the CEMS (UC example) [8].

4.2. Requirements from Control Engineers

The CEMS is implemented with EMSOnto. Control engineers in charge of the implementation employs specific tools for power systems emulator (MATLAB/Simulink) and controller platforms (IEC 61499 platform) [20,30]. Furthermore, not only specific tools but specific documents (IEC 61850 SCL files, SGAM models) were used for the description of IED's functionalities and control applications architecture. This is due to common practices of control engineers. Nevertheless, other tools and frameworks are also available to support the engineering process of power systems (e.g., Modelica

and IEC 61131-3) [31,32] as well as other information models for smart grid networks (e.g., CIM) [33]. The fact of using specific tools to evaluate the proposed methodology would enable a concrete understanding of the actions taken by the EMSOnto expert.

A series of requirements that motivates an expansion and customization of EMSOnto were figured out by control engineers. These requirements are bound to specific tools, documents and information models as exposed below.

(R1) Benefiting from SGAM models: CEMS is roughly described at the specification stage by means of the SGAM-TB. The information added at that stage is reused at design time to fill the EMS-templates. Therefore, an automatic import of SGAM models into EMS-templates is expected with the aim of reducing the manual work conducted by control engineers.

(R2) Discovering CEMS constraints: The controlled process (i.e., the BESS) is surrounded by constraints that characterizes technical limitations of electrical devices and/or safety restrictions. Those constraints need to be considered during design of control strategies. Therefore, an understanding of them is absolutely required during design time. Hence, an automatic retrieval of constraints from the process is expected.

(R3) SoC estimator function generation: In the scope of multi-functional BESS, the battery provides support not only to an individual service, but more than one. In this context, a solution to track the delivered active power of the BESS to a specific service is performed by control engineers. This consists on the calculation of SoC per service by a function called *SoC_estimator*. As a result, this function is implemented whenever a UC controls a BESS. Hence, an automatic generation of the *SoC_estimator* function is sought in the scope of R3.

(R4) Generation of configuration files: EMS and IEDs need to be parametrized and customized to specific requirements and initial state values. A selection of parameters is gathered in a sort of configuration file that is filled out before any execution or simulation of the mentioned systems. These configuration files support the validation of the EMS through the proof-of-concept and implementation phases. R4 seeks for an automatic generation of such files, which should expose the parameters of functions to be deployed within an EMS as well as the IED's parameters.

(R5) Identification of batteries mismatching services: A service that request BESS support would ask to fill certain requirements such as capacity (kWh), maximal power provision (kW), activation time (s) [34,35]. Thus, not all the BESS would pass the pre-qualification procedures defined by network operators. BESS able to participate in Primary Control Reserve (PCR) should provide certain power. A report that identifies BESS that do not meet this requirement is expected.

(R6) Generation of IEC 61499 models: The design of CEMS is validated within a power system emulator as part of the proof-of-concept. Following this step, the implementation of it is performed within a controller platform fully compliant with IEC 61499. On the other hand, some components of the CEMS (e.g., communication interfaces, control logic) were directly tested in the controller platform. This motivates an automatic generation of the IEC 61499 control application either from EMS-templates or from the MATLAB/Simulink project.

(R7) Benefiting from IEC 61850 ICD files: Power system information models hold data important for the IDE and UC repository. In the scope of CEMS, Information Capability Device (ICD) files are handled to control engineers. ICD files are XML-based, which serialize information regarding functions supported by an IED and are often supplied by the IED manufacturer. Thus, R7 seeks for an automated import of ICD files into the repository.

(R8) Identification of misconfigured units: An EMS controlling and monitoring an IED (e.g., BESS) should be aligned to the units of the IED. In other words, a control signal targeting a setpoint should respect the units of the value to be targeted. Otherwise, a wrong value would be set. This applies also to a status been monitored by feedback. For instance, a BESS is providing its SoC status (SoC_{bat}), which has the value of 80%. An EMS retrieving this value may expect raw data between the range 0–1 (0.8). This mismatch between units would cause a wrong behavior of the EMS. Hence, an identification of such inconsistencies is fundamental.

4.3. Analysis Phase: Analysis of Requirements

An analysis of the mentioned requirements determines the actions to be performed by the EMSOnto expert. This analysis is structured under a succession of steps. The first step corresponds to investigate where the transformation across data models takes place. This corresponds to R1, R4, R6 and R7. Those requirements involve different data models that need to be interrelated with EMS-DM. Therefore, data models of SGAM, IEC 61850, IEC 61499 and Matlab/Simulink are studied and an alignment of EMD-DM to the referred data models is sought. Following this, classes and associations not considered within EMS-DM are created and added to it, affecting the EMS-*TBox*.

The next step is to identify requirements clearly connected to the inference of knowledge. R2, R3, R5 and R8 correspond to this category. A complete understanding of the knowledge to be inferred in the scope of those requirements is needed. Thereby mathematical models representing constraints within a control application (R2) are investigated as well as models that drive the operation of *SoC_estimator* function (R3). Furthermore, an understanding of technical limitations within a BESS are also needed (R5) just as information models that conduct the structure of sources and targets defined within information flows (R8). In the aftermath, the *TBox* is again extended to cope with R2, R3, R5 and R8. Eventually, the creation of queries and rules could also take place.

4.4. Realization Phase: Implementations Performed by EMSOnto Expert

The actions taken by EMSOnto expert are highlighted and enumerated in Figure 6. The first step is dedicated to the creation of new concepts, roles, and constraints within the EMS-*TBox*. The expressivity of EMS-*TBox* depends on the ontology language used to implement it (e.g., OIL, DAML+OIL, OWL 2 [36]). In case that a higher expressivity is needed the implementation of rules is carried out as part of the next step.

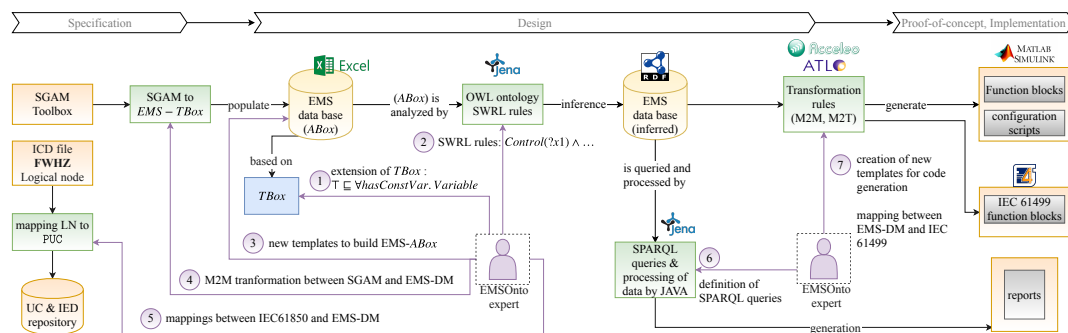


Figure 6. Design and Implementation of CEMS UC— automated by EMSOnto expert.

As a sequel, population of the EMS-*TBox* is done against assertions of the CEMS use case, hence, CEMS knowledge is collected within EMS-templates. EMS-templates are complemented with the information from SGAM models and ICD files. To achieve this, a mapping between the SGAM model and the EMS-DM is performed, as well as a transformation between IEC 61850 models (ICD files) and EMS-DM. When a complete *ABox* is reached the inference of knowledge takes place. Queries are built upon the inferred knowledge to allow the identification of inconsistencies. Finally, the EMSOnto expert establishes M2M and M2T transformation to generate software artifacts as depicted in Figure 6. The implementation of the aforementioned actions are carried out by Semantic Web Technologies, ontology editors and programming frameworks. A detail of the aforementioned actions is addressed below.

4.4.1. Action 1: Extending the EMS-Ontology

Ontology Web Language (OWL) 2, a popular ontology language recommended by the World Wide Web Consortium (W3C) is chosen to implement the expressivity required by EMS-*TBox* (*SROIQ(D)*)

logic) [37]. OWL 2 is based on SROIQ and furthermore it supports data types. Thus, a complete implementation of the *TBox* is reached. On that basis, concepts, roles, and axioms not considered within the EMS-*TBox* are investigated and addressed in this section.

R1 seeks to benefit from UML behavior and structure diagrams specified with the SGAM-TB. Because of that, a concordance between SGAM-TB and EMS-DM should be reached. The EMS-DM focuses on function and information aspects of smart grid control applications. Thereby, the business, communication, and component SGAM layers are out of the scope. The function analysis of SGAM-TB involves different concepts, such as High-Level Use Cases (HLUC), Primary Use Cases (PUC), Logical Actors (LA) [38]. Those concepts are already included in the EMS-DM. However, the concept Information Object (IO) that represents communication between LAs and PUCs was not yet addressed. Hence, it is added to the EMS-DM under the name of *InformationFlow*, which stands for the UC's information that relates one Output to one Input as exposed in Table 2 and Figure 4.

Table 2. Terminological axioms to align EMS-DM with SGAM model.

Concepts, Roles, OWL Axioms	Description
$UC \sqsubseteq \exists hasFlow.InformationFlow \sqcap \dots,$ $InformationFlow \sqsubseteq \leq 1 hasSource.Output \sqcap$ $\leq 1 hasTarget.Input \sqcap \exists hasSource.Output$ $\sqcap \exists hasTarget.Input$	<i>InformationFlow</i> represents information exchanged between UCs. Hence, the role <i>hasFlow</i> relates the concepts UC and <i>InformationFlow</i> . <i>InformationFlow</i> owns one Output as source and one Input as target. A formal representation of this requires the use of qualified number restrictions constructor ($\leq n R.C$, R is a role and C is a concept) [23].

An investigation of mathematical models for IEDs is carried out in the scope of R2. For simplicity, a single-output and single-input process is represented in state space as shown in Equation (1). Where u is the input variable, y is the process output and x the state vector with dimension $n \times 1$ that represent the entire state of the process at any time. In the scope of our study, the process is represented by electrical devices connected along the low voltage distribution grid, such as BESS, PV, load, etc. These processes are frequently surrounded by constraints on control variables (u), state variable (x), or output (y), as described by Equation (2). A control application (i.e., CEMS) that intends to manipulate outputs of the process (y) needs to respect the referred constraints.

From an IED's perspective, an instantiation of previous data with EMS-*TBox* concepts is achievable: Setpoint (u), Manipulated (y), State $\{x_1, \dots, x_n\}$, Constraint $\{u_{\max}, \dots\}$. In this work, only constraints on setpoint variables (u_{\min}, u_{\max}) are inferred, the others (e.g., y_{\max}) follow a similar mechanism. On that basis, the CEMS infers constraints from the IED's setpoints. To achieve this, the concept *Constraint* is further developed as depicted in Table 3, where *Variables* defined within a *Constraint* are modeled as well as the relation between *Constraints*.

$$x' = Ax + Bu, \quad y = Cx + Du, \quad x = [x_1 \quad \dots \quad x_n]^T \quad (1)$$

$$x_{\min} \leq x \leq x_{\max}, \quad u_{\min} \leq u \leq u_{\max}, \quad y_{\min} \leq y \leq y_{\max} \quad (2)$$

Table 3. Concepts, roles and axioms to infer CEMS's constraints.

Concepts, Roles, OWL Axioms	Description
$Constraint \sqsubseteq \exists hasConstVar.Variable \sqcap$ $\exists IsConsLinkCons.Constraint,$ $hasVarConst \sqcap hasConstVar \sqsubseteq T,$ $trans(IsConsLinkCons)$	A <i>Constraint</i> owns <i>Variables</i> , this relation is represented by <i>hasConstVar</i> . The inverse role of <i>hasConstVar</i> is given by <i>hasVarConst</i> . The role <i>IsConsLinkCons</i> relates <i>Constraints</i> and the transitivity property (<i>trans</i>) is assigned to it.
$Variable \sqsubseteq \exists hasVarConst.Constraint \sqcup \dots,$ $hasVarConst \circ IsConsLinkCons \sqsubseteq hasVarConst$	A <i>Variable</i> belongs to a <i>Constraint</i> is represented by <i>hasVarConst</i> . A <i>Variable</i> can inherit <i>Constraints</i> from other <i>Variables</i> , this inference is achieved by complex role inclusion axioms ($R_1 \circ R_2 \sqsubseteq R_3$, being R_1, R_2 and R_3 roles).

The generation of the *SoC_estimator* function is contemplated in R3. Thereby, an analysis of data that conforms such function is carried out. Considering a HLUC controlling the active power supplied by a BESS, the calculation of HLUC's SoC (SoC_{hluc}) follows the logic in Equation [3]. SoC_{hluc} depends on the SoC initially assigned to the HLUC (SoC_{ini}), the current charged into the BESS (I_{bat}), as well as the total capacity of the BESS (Q_{bat}) and a portion capacity assigned to the HLUC (Q_{hluc}). The value I_{bat} is calculated from the active power (P_{bat}) and the nominal voltage (V_{nom}) of the battery. A modeling of the mentioned variables results in the following instantiations: $State\{V_{nom}, SoC\}$, $ParamDevice\{SoC_{ini}, CAh\}$ and $ParamUC\{CAh_UC\}$. Nevertheless, a further representation of those variables is required, resulting in new concepts. For instance, the concept V_{nom} represents nominal voltages and is subsumed by $State$. A detail of all those new concepts is given in Table 4.

On the other hand, the creation of new roles facilitates the generation of *SoC_estimator*. Hence, $ControlBESS$ relates a UC controlling a BESS and the concrete role $hasI_0$ facilitates a remote control of internal variables defined within the scope of a UC as detailed in Table 4.

$$SoC_{hluc} = SoC_{ini} + \int \frac{I_{bat}}{Q_{bat}Q_{hluc}}, \quad I_{bat} = \frac{P_{bat}}{V_{nom}} \quad (3)$$

Table 4. Extension of EMS-TBox to generate *SoC_estimator* function.

Concepts, Roles, OWL Axioms	Description
$V_{nom} \sqcup SoC \sqcup P \sqcup I \sqcup \dots \sqsubseteq State$	V_{nom} and SoC represent a nominal voltage (e.g., V_{nom}) and a state of charge (e.g., SoC_{hluc}) respectively. P models an active power (e.g., P_{bat}) and I a current value (e.g., I_{bat}).
$SoC_{ini} \sqcup CAh \sqcup \dots \sqsubseteq ParamDevice$	SoC_{ini} represents an initial SoC (e.g., SoC_{ini}), CAh models the full capacity of an energy storage device (e.g., Q_{bat}).
$CAh_UC \sqcup \dots \sqsubseteq ParamUC$	Capacity assigned to a UC is represented by CAh_UC (e.g., Q_{hluc}).
$Application \sqsubseteq \exists IsConnectedTo.Application \dots$, $HLUC \sqsubseteq \exists ControlBESS.BESS \dots$	The role $IsConnectedTo$ relates two applications and the role $ControlBESS$ relates a HLUC that is connected to a BESS.
$hasI_0 \text{ keyfor } Internal$	$hasI_0$ gathers information about whether or not a variable of type $Internal$ is assigned to an $Input$ or $Output$. Thereby, variables affected by this role are those subsumed by $Internal$ ($Param, State, \dots$).

4.4.2. Action 2: Enlargement of Inference Process

The inference of data reached by OWL ontologies can be extended by Semantic Web Rule Language (SWRL). This concept is standardized by W3C as language for expressing rules and provides powerful deductible reasoning mechanisms compared to OWL. SWRL is based on OWL DL, Datalog and Horn-like rules [39]. As a result, the syntax of SWRL is represented by atoms, rule body (antecedent) and rule head (consequent): $Atom \wedge Atom \wedge \dots \rightarrow Atom$. If the conditions on the antecedent hold, then also the conditions in the consequent hold. On that basis, SWRL rules $r1$ and $r2$ are designed to support the identification of CEMS's constraints, as shown in Table 5. Where $r1$ deduces $Variables$ assigned to $Constraints$ and $r2$ infers relations between $Constraints$.

Table 5. Rules to infer CEMS's constraints.

SWRL Rules	Description
$r1: Control(?x1) \wedge IsAssignedTo(?x1, ?s) \wedge Setpoint(?s) \wedge hasVarConst(?s, ?c1) \wedge Constraint(?c1) \rightarrow hasVarConst(?x1, ?c1)$	A $Control$ inherits constraints assigned to the $Setpoint$ that it targets. A setpoint variable $?s$ owns the constraint $?c1$, if $?s$ is controlled by $?x1$, then $?x1$ inherits the constraint $?c1$.
$r2: Constraint(?c1) \wedge hasConstVar(?c1, ?x1) \wedge IsVarLinkVar(?x1, ?x2) \wedge Variable(?x1) \wedge Variable(?x2) \wedge hasVarConst(?x2, ?c2) \rightarrow IsConsLinkCons(?c1, ?c2)$	The role $IsConsLinkCons$ is established when a relation between $Constraints$ is detected.

On the other hand, certain rules that requires the creation of new instances, in the consequent that these do not appear in the antecedent, are not possible by SWRL. In those cases, SPARQL Update queries are used instead [40]. SPARQL Update is a standard language that executes updates to triples in a graph store. It uses the data operators INSERT and DELETE for inserting and removing triples.

Following that, mechanisms to generate the *SoC_estimator* function use *r3* to identify when a HLUC is controlling a BESS by means of *ControlBESS*. As a sequel, *r4* attaches a new PUC (*SoC_estimator*) to the identified HLUCs. Finally, *r5* is responsible for the connections across the BESS, the HLUC and the created PUC (*SoC_estimator*). Those rules are detailed in Table 6.

Table 6. Rules to support the inference of the function: *SoC_estimator*.

SWRL Rules and SPARQL Update Query	Description
r3: EMS(?y) ∧ HLUC(?z) ∧ hasHLUC(?y,?z) ∧ BESS(?x) ∧ IsConnectedTo(?x,?y) ∧ hasControl(?z,?x1) ∧ Control(?x1) ∧ IsAssignedTo(?x1,?x2) ∧ P(?x2) ∧ hasVariable(?x,?x2) → ControlBESS (?z,?x)	If an EMS contains a HLUC that controls active power (P) of a BESS. Then, such HLUC and BESS are bound by <i>ControlBESS</i> .
r4: PREFIX CEMS :< http://.../CEMS# > INSERT{ ?x1 CEMS:hasPUC ?x3. ?x3 rdf:type CEMS:PUC. ?x3 CEMS:hasType "SoC_estimator"^^xsd:string. ?x3 CEMS:hasName ?x3N } WHERE{ ?x1 rdf:type CEMS:HLUC. ?x1 CEMS:ControlBESS ?x2. ?x1 CEMS:hasName ?x1N. BIND(URI(CONCAT("SoC_",STR(?x1))) as ?x3). BIND(CONCAT(STR(?x1N),"_SoC") as ?x3N)}	A PUC of type ' <i>SoC_estimator</i> ' is added to a HLUC that controls a BESS. The name assigned to the new PUC is a concatenation of HLUC's name and the string ' <i>SoC</i> '. For instance, a PUC called <i>FW_SoC</i> is assigned to a HLUC(<i>FW</i>) named <i>FW</i> .
r5: PREFIX CEMS :< http://.../CEMS# > INSERT{ ?x5 CEMS:IsAssignedTo ?x4. } WHERE{ ?x1 CEMS:ControlBESS ?x2. ?x1 CEMS:hasPUC ?x3. ?x3 CEMS:hasType "SoC_estimator"^^xsd:string. ?x3 CEMS:hasFeedback ?x4. ?x4 CEMS:hasType "Vnom"^^xsd:string. ?x2 CEMS:hasStatus ?x5. ?x5 CEMS:hasType "Vnom"^^xsd:string }	A BESS's Status is assigned to a Feedback of the function PUC(<i>SoC_estimator</i>). For simplicity, only the inference of relations between BESS's <i>V_{nom}</i> and PUC(<i>SoC_estimator</i>) are shown. Thus, since <i>V_{nom}</i> is needed to calculate <i>SoC_{hluc}</i> , a role <i>IsAssignedTo</i> representing the relation of Status(<i>V_{nom}</i>) and PUC(<i>SoC_estimator</i>)'s Feedback is established.

4.4.3. Action 3: Setting up of New EMS-Templates

This section addresses all the modifications carried out along EMS-templates to keep a consistency with the extended EMS-*TBox*. The extended headlines are highlight with blue color.

The detection of CEMS's constraints would need to gather instances of these general statements: a UC (*s*) contains a Setpoint (*x*), which in turn is constrained by Constraint (*C*₁). Moreover, a Constraint (*C*₁) is related to a Constraint (*C*₂). Thereupon, an extension of the spreadsheet template collecting the UC's attributes is performed and resulted in Table 7. Moreover, the field *Const_Description* is included to gather constraint's descriptions.

Table 7. Extension of EMS-templates to collect constrains of a variable.

UC	Variable	Description	Type	Const.	Const_Description	IsConsLink.
s	x	variables's description	Setpoint	C ₁	$x_{\min} \leq x \leq x_{\max}$	C ₂

The vocabulary defined to model the *SoC_estimator* involves variables of BESS and UCs. The current version of EMS-templates contemplates generic models of BESS and UCs. Nevertheless, those models need to be extended to achieve the implementation of the *SoC_estimator* as shown in Table 8. For instance, the concept CAh_UC that represents the capacity assigned to a UC is now included in the UC(*UC_generic*), just as the CAh concept in the UC(*BESS*). Besides this, a complete model of PUC(*SoC_estimator*) is also required, see Table 9. All the extended models are made available in the IED and UC repository. Hence, control engineers are free to edit all the fields except Type.

Table 8. BESS and UC models concerned by UC (*SoC_estimator*).

UC	Variable	Description	I_O	Type	Value	Format	Unit
BESS	CAh	total capacity of the battery	Status	CAh		double	Ah
	Vnom	nominal voltage of the battery	Status	Vnom		double	V
UC_generic	SoCini	initial SoC of the use case	Status	SoCini		double	%
	CAh	capacity assigned to a UC	Status	CAh_UC		double	Ah

Table 9. Parameters and states relevant to UC(*SoC_estimator*).

UC	Variable	Description	I_O	Type	Value	Format	Unit
SoC_estimator	SoC_UC	SoC of a UC	Status	SoC		double	%
	I	current charged into the battery	Status	I		double	A
	U_bat	voltage of the battery	Feedback	Vnom		double	V
	CAh_bat	total capacity of a battery	Feedback	CAh		double	Ah
	P_UC	active power set by a UC	Feedback	P		double	kW
	SoC_ini	initial SoC of the UC	Feedback	SoCini		double	%
	CAh_UC	capacity assigned to a UC	Feedback	CAh_UC		double	Ah

4.4.4. Action 4: Mapping Between SGAM-TB and EMS-DM

The exchange of data between SGAM models and EMS-templates is addressed using MDA techniques. Hence, a model of SGAM-TB is investigated and proposed in this section.

A UML class representation of SGAM-TB is suggested by Neureiter et al. [38]. Such representation motivates the SGAM-TB model proposed in this work. It is worth highlighting that only function and information layers are modeled. The function layer of SGAM-TB involves description of main functionalities within the concept HLUC, further details of HLUCs are given in PUC. Actors involved within those PUCs are modeled by the LogicalActor. In turn, the information layer includes the InformationObject a concept representing any information flow. Which is structured by a data model (DataModel). The aforementioned concepts are combined to conceive a UML class representation of SGAM-TB, see Figure 7a. Considering this, classes, attributes, and other components of SGAM-TB model are transformed into EMS-DM components, as shown in Figure 7b. For a straightforward interpretation only the mapping between classes is depicted.

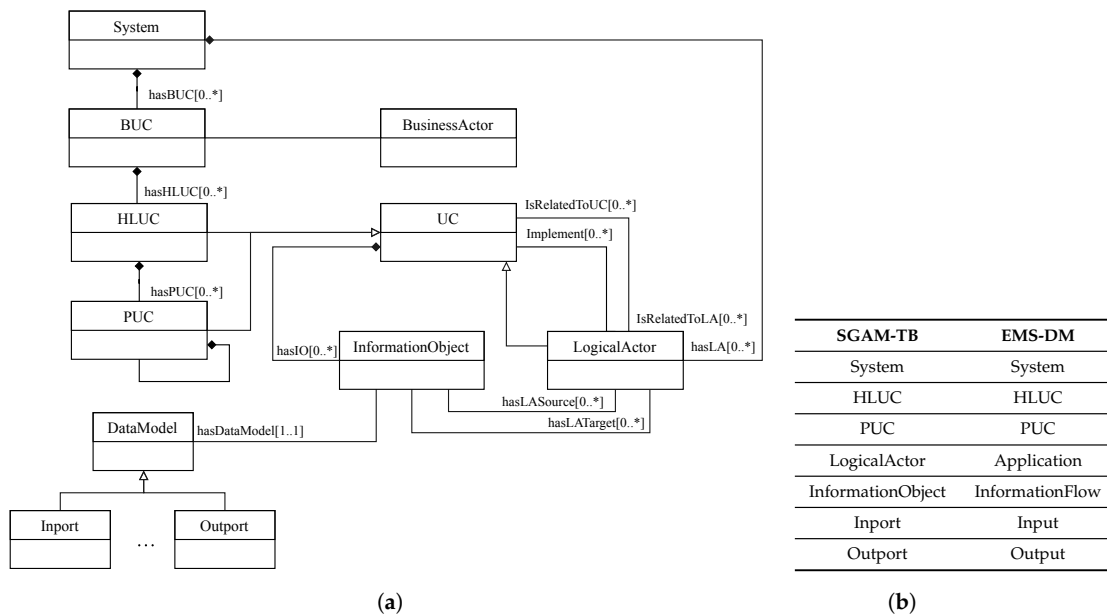


Figure 7. Proposed foundations to achieve a mapping between SGAM-TB and EMS-DM. (a) UML class diagram of SGAM-TB; (b) matching between SGAM-TB and EMS-DM.

4.4.5. Action 5: Mapping Between IEC 61850 and EMS-DM

The IEC 61850 approach is object-oriented and has been lately used in power system domain to improve interoperability between IEDs. The abstraction of services provided by an IED begins with a Logical Node (LN), which models services for protection, measurement, control, etc. Each LN is defined as a class within the IEC 61850-7-4 specification [41]. Data Objects (DO) refine the definition of a LN, they are specified by Common Data Classes (CDC) in IEC 61850-7-3 [42].

On the other hand, IEC 61850 defines a language to configure IEDs under the name of System Configuration Language (SCL) in IEC 61850-7-6. An SCL specifies the capabilities of an IED by LN classes, which are of type `LNNodeType`. Each of them contains DOs of type `DOType` which in turns contain DAs of type `DAType`. A `LNNodeType` should correspond to a LN as well as a `DOType` to a CDC.

The instantiation of the mentioned classes is exemplified by the `LNNodeType:MMXU`, see Figure 8. This notation (`class:individual`) is employed to assign an individual to a class. The mentioned `LNNodeType` represents a calculation of current, voltage, frequency, etc. Thereby, it contains among others the `DO:Hz` for frequency measurement. This DO contains `DA:mag` to abstract the mean value of the frequency.

On the basis of SCL representation, the mapping between EMS-DM and IEC 61850 is carried out as follows: a `LNNodeType` is transformed into a UC and a DO into a `Variable`. The type of `Variable` (i.e., `Feedback`, `Control`, etc.) to be generated is obtained from the attribute `cdc` within `DOType`. Hence, a `MV` (Measured Value) results in a `Status` assigned with a `Numeric` value. The DAs elements of a `DOType` (i.e., `t`, `mag`, `units`, etc.) are mapped into the attributes of `Numeric` (i.e., `hasTimeStamp`, `hasAnaValue`, `hasUnits`, etc.). Those mappings are referenced by color match in Figure 8.

In the previous example a `DO:Hz` representing a MV resulted in a `Status`. However, a DO may belong to the other types such `ASG` (Analogue Setting) or `SPS` (Single Point Status) among others. Depending on that, a generation of either a `Setpoint` or `Status`, etc. is performed. It is worth highlighting that not all the DAs of a `DOType` can be mapped into the EMS-DM. For instance, the `DA:q` that represents the quality of the information is not mapped. To accomplish that, an extension of attributes assigned to `Variable` is required. As a summary, any IED functionality defined within a LN is translated into a UC and upload into the repository.

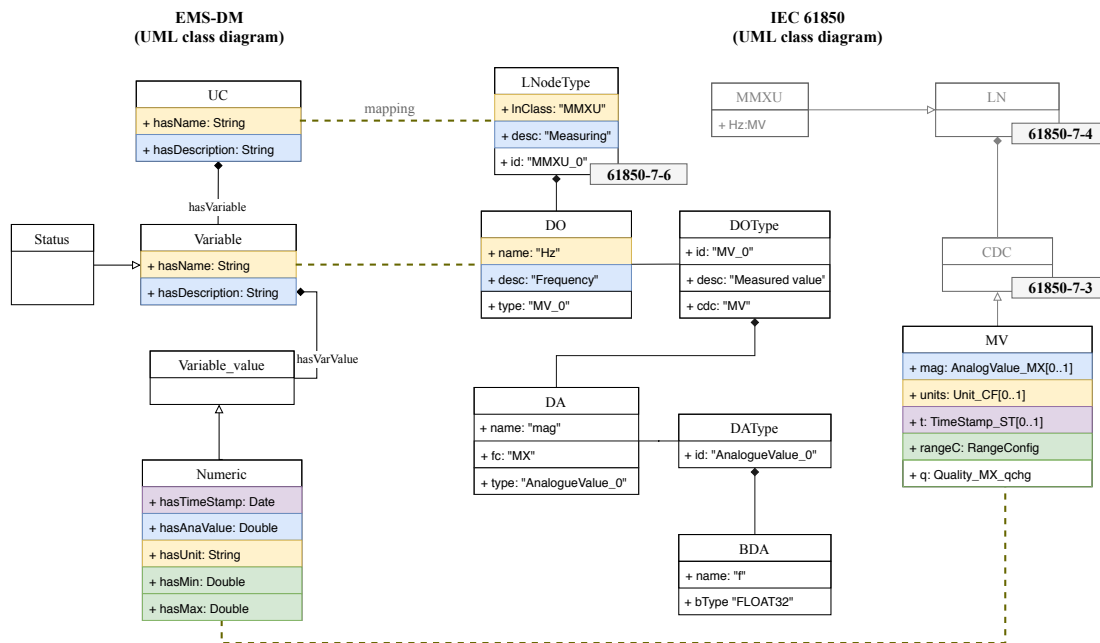


Figure 8. Mapping between IEC 61850 (LN) and EMS-DM (UC).

4.4.6. Action 6: Generation of Inconsistency Reports

The identification of certain inconsistencies may require the extension of the EMS-*TBox* as well as definition of new rules and queries besides adjustments to the EMS-templates. However, the fulfillment of R5 and R8 requires just the elaboration of new queries, those are addressed as follows. In the scope of R5, constraints on a BESS are analyzed. The active power value of a battery is limited as shown in Equation [4]. The identification of services that attempt to control P_{bat} with values that exceed the technical limits requires the modeling of P_{max} . Besides this, the power demanded by a service is also needed ($BessSize$). Those concepts were already included in the EMS-*TBox*.

A formalization of competency questions is achieved by DL or SPARQL queries [43]. This depends on the type of question, usually elaborated questions need the expressivity of SPARQL syntax. In turn, simple questions can be formulated by DL. On that basis, R5 implies DL queries to answer questions about BESS’s restrictions and UC’s parameters, see Table 10.

$$P_{min} < P_{bat} < P_{max} \tag{4}$$

Table 10. Querying the EMS-*ABox* to identify batteries mismatching services.

Query	Question	DL Query
Q_I	What is the variable of type P_{max} defined within a BESS(<i>battery</i>)?	$P_{max} \sqcap \exists hasParam^- .BESS\{battery\}$
Q_{II}	What is the active power to be required by a service HLUC (<i>Service_n</i>)?	$BessSize \sqcap hasParam^- .HLUC\{Service_n\}$

On the other hand, since R8 investigates misconfigured units, an analysis of the information flow is required. An information flow is conformed by a source and a target. A source can be of type Control and Output. In turn, a target is of type Setpoint and Feedback. Units of the source should match the units configured at the target. Enough information is already provided within the EMS-templates to conclude mistaken units. The SPARQL query in Table 11 investigates and compares units within a Setpoint and Control.

Table 11. SPARQL query to investigate misconfigured units.

Query	Question	SPARQL Query
Q_{III}	What are the setpoints of a HLUC? What is the unit configured within a setpoint? What is the unit of a control variable targeting certain setpoint? What are the units that mismatch?	<pre> SELECT ?control ?setpoint ?unit_ct ?unit_sp WHERE{ ?HLUC CEMS : hasSetpoint ?setpoint; rdf : type CEMS : HLUC. ?setpoint CEMS : hasUnit ?unit_sp. ?control CEMS : IsAssignedTo ?setpoint; rdf : type CEMS : Control; CEMS : hasUnit ?unit_ct FILTER(!regex(STR(?unit_ct), STR(?unit_sp))) </pre>

4.4.7. Action 7: Software Artifacts Generation

This section tackles the issues exposed in R4 and R6. Therefore, an automatic generation of configuration files and software artifacts compliant with IEC 61499 is pursued.

The achievement of a configuration file from EMS-templates is carried out by M2T transformations. Hence, the model source is given by EMS-DM and the text target is defined by control engineer's practice. As an example, a MATLAB script that supports the proof-of-concept of CEMS is sought. Such script should expose parameters within a HLUC. Hence, a M2T is implemented by Acceleo templates, an open source framework for code generation, see Figure 9. That Acceleo template investigate all HLUCs (i.e., functions) included within an EMS. Afterwards, the parameters (Param) assigned to the identified HLUCs are extracted and customized.

```

[for (app: Application | asystem.hasApplication -> filter(EMS) ) separator('\n')]
  [%[app.eClass().name/] [app.hasName /];
  [for (hluc:HLUC | app.hasHluc ) separator('\n')]
    %HLUC [hluc.hasName/]
    %Parameters of the Hluc:
    %=====
    [for (var:Param | hluc.hasVariable->filter(ParamHluc) )
      [app.hasName.concat('.').concat(hluc.hasName).concat('.').concat('Param').concat('.').
        concat(var.hasName).concat('=')] /] [ var.hasVarValue-> filter(Analog).hasAnaValue/];
  [/for] ...

```

Figure 9. Acceleo template to generate configuration file of HLUCs.

The proof-of-concept of CEMS is also supported by generation of MATLAB/simulink models. A generation of MATLAB/Simulink applications from EMS-templates is based on M2M transformation where the PIM is given by EMS-DM and the PSM by a MATLAB/Simulink model. As first step, a conception of MATLAB/Simulink model should be reached, this model is proposed within the MATLAB Simulink Integration Framework for Eclipse (MASSIF) project [44]. The root of that model is SimulinkModel, which contains Block and SubSystem. A SubSystem is characterized by properties and ports of type Output and InPort. Those ports may be related to a SingleConnection. This configuration is quite similar to EMS-DM structure. Hence, the concepts: Application^{EMS} and UC^{EMS} are transformed into SubSystem^{Sim}, Input^{EMS} into InPort^{Sim}, InformationFlow^{EMS} into SingleConnection^{Sim} and so on, as shown in Table 12. To increase the understanding of classes transformation the syntax class^{Model} is used.

An automatic generation of IEC 61499 applications from EMS-templates requires a model representing IEC 61499 applications. A UML class representation of it is reached from an analysis of the standard IEC 61499 [45], see Figure 10. As it can be noticed the depicted model share many similarities with the EMS-DM, thus a matching between them is possible as shown in Table 12. A UC^{EMS} is mapped to a FB⁶¹⁴⁹⁹, which is defined by a FBType⁶¹⁴⁹⁹. The FBType can take the form of one single block (BasicFB) or an arrangement of blocks (FBNetwork), see Figure 10. On the other hand, Applications^{EMS} is converted into Application⁶¹⁴⁹⁹, Inputs^{EMS} into InputVars⁶¹⁴⁹⁹ and so on.

Model-to-model transformation across MATLAB/Simulink, IEC 61499 and EMS-DM can be implemented by following the mappings referenced in Table 12. This gives a flexibility to transform EMD-DM instances (i.e., EMS-templates) into specific platforms important to the proof-of-concept and implementation phases. Transformation rules are carried out using ATL Transformation Language (ATL), a M2M transformation language that implements unidirectional transformations within the Eclipse platform [46]. To illustrate this, the rules to transform UC^{EMS} into $FBType^{61499}$ are shown in Figure 11. Where a navigation through the components of $FBType$ (i.e., $InputVars$, $OutputVars$, $InternalVars$) enables a match to EMS-DM elements (i.e., Input, Output, Param).

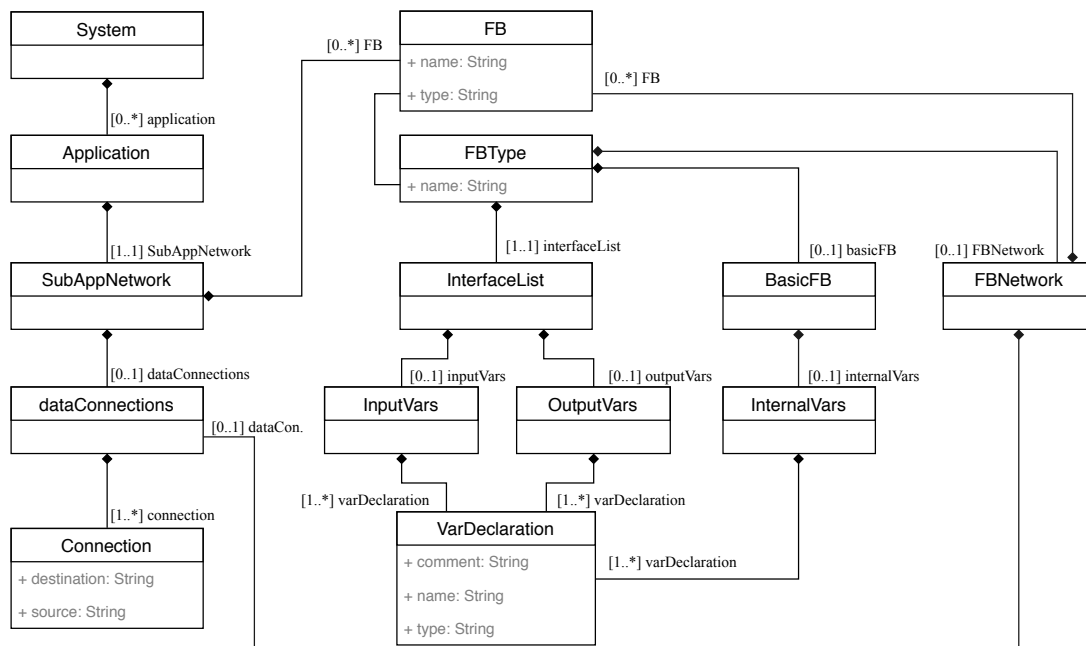


Figure 10. UML class diagram representation of IEC 61499 (PSM).

```

rule UC2FBType {
  from
  s:EMS!UC
  to
  t:iec61499!FBTypeType(
    name <- s.hasName,
    interfaceList <- SinterfaceList),
  SinterfaceList:iec61499!InterfaceListType (
    inputVars <- SInputVarsType,
    outputVars <- SOutputVarsType,
    SInputVarsType: iec61499!InputVarsType(
      varDeclaration <- s.hasVariable -> select(e e.oclIsTypeOf(EMS!Input)) ->
      collect(e thisModule.Input2VarDeclaration(e)), ...

  lazy rule Input2VarDeclaration {
    from
    s:EMS!Input(s.isContainsElement())
    to
    t: iec61499!VarDeclarationType(
      comment <- s.hasDescription,
      name <- s.hasName ,
      type <- thisModule.TypeConversion(s.hasFormat)) }

```

Figure 11. ATL rule transformation from UC^{EMS} into $FBType^{61499}$ (reduced model).

Table 12. Mapping between EMS, MATLAB/Simulink and IEC 61499 data models.

EMS-DM	MATLAB/Simulink	IEC 61499
System	SimulinkModel	System
UC, HLUC, PUC	SubSystem	FB, FBType
Application	SubSystem	Application
Input	Inport	InputVars
Output	Output	OutputVars
InformationFlow	SingleConnection	Connection
Param	Property	InternalVars

5. CEMS Implemented with the Extended EMSOnto

This section shows how the above presented improvements of the EMSOnto are used by control engineers during the realization of a CEMS.

5.1. EMS-Templates

The previously introduced CEMS is specified using SGAM-TB, which is available as an extension of the design tool Sparx Systems EA, thus UML and SysML support the CEMS design. Hence, the structure of the CEMS is specified under UML use case diagrams, the resulting model is depicted in Figure 12. From such a representation it is deduced that the CEMS implements *SelfC* and *FW*, which are associated with a *Meter* and a *BESS*. Moreover, *SelfC* implements the functions *PI_Control* and *Limit_SoC* just as *FW* implements *Linear-Control* and *Limit_SoC*. All those details define the CEMS structure and should be considered at the design phase. To support this, an automatic transfer of information from SGAM models to EMS-templates is executed. This involves UML use case, sequence, and activity diagrams. The resulted EMS-template from a use case diagram is shown in Table 13. This represents a first version of the EMS-ABox and the starting point of the design process.

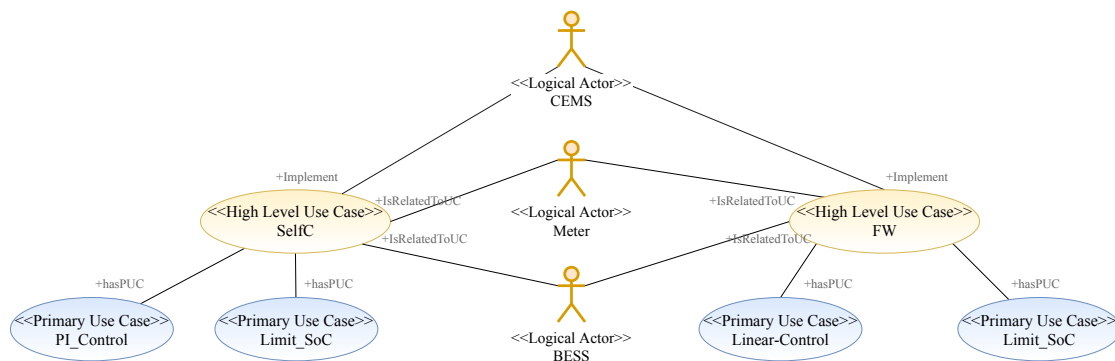


Figure 12. CEMS’s structure represented by a UML use case diagram.

Table 13. Spreadsheet generated automatically from CEMS’s UC diagram.

System	Appl.	Application Description	Type	HLUC	HLUC Description	PUC
Sys	CEMS	customer energy management system	-	SelfC	power from the grid is avoided	PI_Control
			-	FW	active power is injected to support frequency regulation	Limit_SoC
	BESS	model of a BESS	BESS	-	-	-
	Meter	smart meter connected at PCC point	Meter	-	-	-

5.2. UC and IED Repository

At the design phase, ICD files provided by the IED's manufacturer are available to the control engineers. Selected ICD files are imported into the UC repository to benefit from IED and UC models.

A reduced model of an ICD file containing the LN:FWHZ is illustrated in Figure 13. A detail description of that LN is available in IEC 61850-90-7 [47]. The XML file shows variables within the LN:FWHZ, which are represented as DO elements. Attributes of those variables are contained within the DOType identified under ASG_0. Those attributes are of type *AnalogogueValue_0* and own values of type *FLOAT32*. With that in mind, it is understood that LN:FWHZ contains the variables *Wgra*, *HzStr* and *HzStop*. Each of them owns a series of attributes: *setMag*, *units*, *minVal* and *maxVal* which are of type *FLOAT32*. The exposed LN:FWHZ is transformed into PUC(FWHZ), afterward this PUC is load into the UC & IED repository, thus accessible to control engineers.

Since the operation followed by the LN:FWHZ is similar to PUC(*Linear-Control*), it is worth setting the PUC's type to *FWHZ*. As a result, all the variables related to LN:FWHZ are imported within *Linear-Control*, see Table 14. However not all the attributes' values are collected (e.g., min, max, unit), this is because those values are set during real-time operation of IEDs.

```
<LNNodeType lnClass="FWHZ" id="FWHZ_0">
  <DO name="Wgra" type="ASG_0" desc="active power gradient in percent of frozen ...
  <DO name="HzStr" type="ASG_0" desc="delta frequency between start frequency ...
  <DO name="HzStop" type="ASG_0" desc="delta frequency between stop frequency ...
</LNNodeType>
<DOType cdc="ASG" id="ASG_0" desc="Analogue setting">
  <DA dchg="true" fc="SP" name="setMag" bType="Struct" type="AnalogueValue_0"/>
  <DA dchg="true" fc="CF" name="units" bType="Struct" type="Unit_0"/>
  <DA dchg="true" fc="CF" name="minVal" bType="Struct" type="AnalogueValue_0"/>
  <DA dchg="true" fc="CF" name="maxVal" bType="Struct" type="AnalogueValue_0"/>
</DOType>
<DAType id="AnalogueValue_0">
  <BDA name="f" bType="FLOAT32"/>
</DAType>
```

Figure 13. ICD file containing a reduced model of LN:FWHZ.

Table 14. PUC (*Linear-Control*) generated automatically from repository's models (LN:FWHZ).

PUC	Variable	description	Type	Format	Min	Max	Unit
Linear-Control	Wgra	active power gradient in percent of frozen active power value per Hz	Setpoint	FLOAT32			
	HzStr	delta frequency between start frequency and nominal frequency	Setpoint	FLOAT32			
	HzStop	delta frequency between stop frequency and nominal frequency	Setpoint	FLOAT32			

5.3. Constraints of the CEMS

The inference of CEMS's constraints entails a collection of technical limitations regarding the process to be controlled (i.e., DERs, IEDs). This means that IED's constraints are documented within the extended EMS-templates. Once this process is complete, IED models are upload to the repository and CEMS's constraints are automatically resolved.

As an example the BESS's constraints are inferred. The BESS model is built by control engineers, see Table 15. As it can be noticed, the BESS owns the variables active power (*Pbat*) and apparent power (*Sbat*). The active power is set by *Setpoint(sp_Pref)*. This statement is exemplified by the role *IsAssignedBy*. Moreover, constraints on the BESS are detailed. Hence, C1 represents a constraint on *Pbat* and C2 a constraint on *Sbat*. Since *Sbat* and *Pbat* are constrained by $Pbat^2 + Qbat^2 \leq Sbat^2$ the relation *IsConsLinkCons*(C1, C2) is established. The resulted BESS is upload to the repository. Next, a reasoner engine is executed, and new knowledge is provided to the control engineers. One of those

results corresponds to $PUC(Limit_SoC)$ being limited by the constraints C1 and C2. This implicit data is highlight with blue color in Table 16. The role $IsAssignedTo$ is the inverse of $IsAssignedBy$.

Table 15. Extended BESS model apprised by control engineers

UC	Variable	Description	Type	IsAssignedBy	Const.	Const.	Description	IsConsLink.
BESS	Pbat	active power	State	sp_Pref	C1	$P_{min} \leq P \leq P_{max}$		C2
	Sbat	apparent power	State		C2	$P^2+Q^2 \leq S^2$		-

Table 16. Constraints allocated to the PUC ($Limit_SoC$).

PUC	Variable	Description	Type	IsAssignedTo	Const.	Const.	Description
Limit_SoC	ct_Pref	signal to control the charging/discharging of the BESS	Control	sp_Pref	C1	$P_{min} \leq P \leq P_{max}$	
					C2	$P^2+Q^2 \leq S^2$	

5.4. Inconsistencies Report

Inconsistencies reports are generated from querying the CEMS's inferred data. The original EMSOnto already supports reports of conflicts between UCs [8]. On top of that, the extended EMSOnto allows the identification of further inconsistencies such as BESS mismatching a service (I_I) and misconfigured units (I_{II}). During the design process the report shown in Table 17 was provided to control engineers. The report shows I_{II} as an identified mismatch, which led to a correction of the unit configured within $Control(ct_Pref)$, a control signal targeting the setpoint (sp_Pref) of the BESS.

Table 17. Inconsistencies reports handled to control engineers.

Inconsistency	Detected	Conclusion Derived from Queries.	Control Engineer Analysis
Mismatches between a BESS and a service/ I_I	X	The technical limitations of the BESS are not violated.	-
Units are misconfigured/ I_{II}	✓	The unit of the control variable (ct_Pref) is set to W and the unit configured in a setpoint (sp_Pref) is kW.	Correction of the unit at the control level is required.

5.5. SoC Estimator Function

Since FW and $SelfC$ are controlling the BESS they need to estimate how much power was delivered by the BESS. This is reached by instantiating the function $SoC_estimator$ and attaching those instances to $HLUC\{FW, SelfC\}$. EMSOnto provides an automatic creation of $PUC(FW_SoC)$ and $PUC(SelfC_SoC)$ as shown in Table 18. The PUCs are instances of $SoC_estimator$ and therefore share the same attributes (e.g., CAh_UC, I). This new information is highlighted by blue color.

Table 18. SoC of the functions $HLUC\{FW, SelfC\}$ is estimated by adding new PUCs.

HLUC	PUC	Description	Type	Variable	Description
FW	FW_SoC	state of charge of a HLUC	SoC_estimator	CAh_UC	capacity assigned to a UC
SelfC	SelfC_SoC	state of charge of a HLUC	SoC_estimator	I	current charged into the battery

5.6. Software Artifacts Generation

An automatic generation of the MATLAB script shown in Figure 14 is performed. Only the parameters' values are generated. Other attributes such as format and unit are dismissed. Moreover, a refinement of the code was not necessary and it was used as it was generated. This script supports the proof-of-concept of the CEMS simulated within a MATLAB/Simulink platform [8].

On the other hand, the validated CEMS is transformed into IEC 61499 models, the ensemble of function blocks focused on $HLUC(FW)$ is shown in Figure 15. A refinement of the obtained model

consisting on the replacement of function blocks is carried out. Furthermore, two clients simulating the communication with real BESS and meters are generated but not configured (*BESS* and *Meter*).

```

%Application CEMS;
%HLUC FW
%Parameters of the Hluc:
%=====
CEMS.FW.Param.Priority=1.0;
CEMS.FW.Param.CAh=0.3;
CEMS.FW.Param.SoC=10.0;
CEMS.FW.Param.BessSize=100.0;
CEMS.FW.Param.Ena=1.0;
CEMS.FW.Param.SoCini=1.0;

```

Figure 14. Script generated to configure HLUC(FW) presented in a MATLAB/Simulink model.

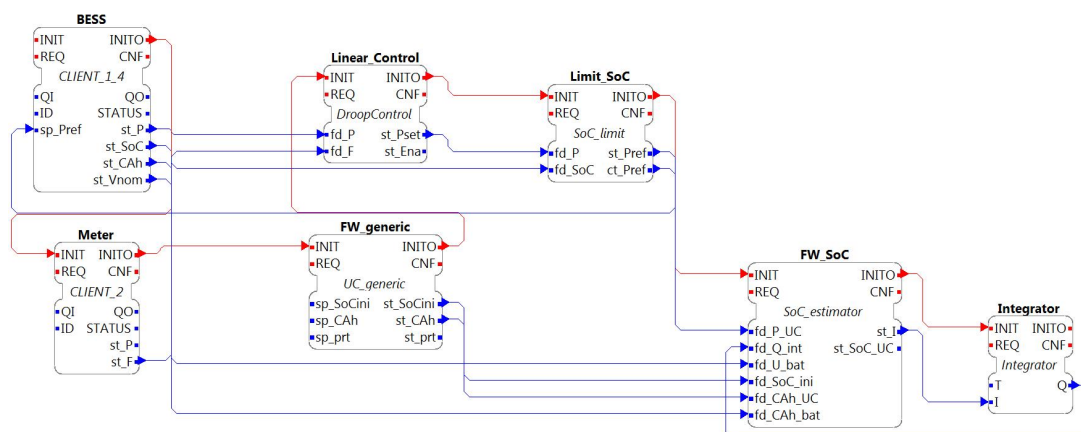


Figure 15. IEC 61499 model issued from EMS-templates (only FW is represented).

5.7. Evaluation of Requirements and Open Issues

This section exposes how the open issues stated in Section 2.4 are addressed by analyzing the implementation of the selected use case example (CEMS).

Documents available at design phase are exploited: The previous experiments regarding the CEMS demonstrate an automatic processing of documents intended to support the CEMS design. In other words, knowledge structured under information models (IEC 61850) and specification approaches (SGAM) are collected to populate EMS-templates.

SGAM models provide a high amount of information regarding the specification of an EMS. Since the design should implement a solution according to EMS specifications, SGAM models were imported within EMS-templates. As a result, at the design phase control engineers starts with a design totally aligned with the requirements defined by stakeholders. Another benefit of this, is that potential manual errors, susceptible to be introduced by manual input of EMS-templates, are avoided. Furthermore, it has been demonstrated that by model-driven engineering techniques it is possible to employ SGAM models in a computerized way to support the whole engineering process. This corresponds to the use of SGAM knowledge for the generation of software artifacts. Nevertheless, the presented experiments did not consider other specification approaches such as Intelligrid and UML. Getting information derived from such approaches would imply to carry out a formal data representation as well as transformation rules and maybe extensions on the EMS-*TBox* as it was done with SGAM. The fact of proving an alignment between SGAM and EMSOnto sets the basis for other possible alignments.

Another executed experiment corresponds to the import of IEC 61850 SCL files into the IED and UC repository. IEC 61850 suggests not only power system components but also IED functionality. In that regard, the PUC(*Linear-Control*) is successfully generated from the LN *FWHZ*. Only the LN's

attributes required by engineers were covered in the experiments. However, since the EMS-*TBox* does not contain all the classes and attributes defined within IEC 61850 a full import of the LN's attributes would involve a further extension of the EMS-*TBox*.

The successful import of SCL files attempts to demonstrate that power utility information models can be used to feed the UC and IED repository. This means that the same mechanisms used to import IEC 61850 models are expected for other models such as energy market communication models formalized in IEC 62325 and also smart meter models defined within IEC 62056 among others [48].

Enlargement of inferred knowledge: Inferences provided by EMSOnto were extended. The inferences performed in this work support control engineers by deducing the constraints to be respected. Besides this, the identification of inconsistencies that may harm the correct operation of control applications was carried out. On top of that, new functions (*SoC_Estimator*) were created to accelerate and support the design of control applications.

Since different sort of inferences were achieved this may lead to think that EMSOnto is flexible to implement any kind of inference. However, this strictly depends on the expressivity provided by the ontology (*SROIQ(D)*) and also the rules. Although *SROIQ(D)* is well defined, the expressivity for rules is determined by rule languages. In this work, SWRL and SPARQL were enough to address the control engineer's requirements. Nevertheless, other requirements may need to use the Rule Interchange Format (RIF), which provides the means to insert primary logic programming languages [37], or even to extend ontologies with probabilistic inference and temporal logic [49].

Flexible generation of software artifacts: CEMS experiments demonstrate that generated code and models support control engineers not only in the proof-of-concept but also during the implementation phase. To make this possible, mappings between a power system emulator (MATLAB/Simulink), a controller platform (IEC 61499 platform) and Emsonto are conceived as well as text templates for code generation (MATLAB script). This evaluation lays the groundwork for implementations to other specific platforms (IEC 61131-3, PowerFactory).

The final realization of control applications requires the participation of engineers for the customization of generated models as well as the definition of the control algorithm's behavior. In the use case example, communication interfaces were generated (*BESS* and *Meter* clients), although not configured. Consequently, the generation of a full control application ready to be tested is not fully automated. Indeed, other approaches cover those gaps. For instance, PSAL supports the design and implementation of communication and component layers [10]. Therefore, the cooperation between different engineering and development approaches is encouraged.

6. Conclusions

In the near future, a high integration of BESS within the distribution grid may very well become a reality since they can provide a broad range of services. Those services are likely to be deployed within an EMS which in turn will control the active and reactive power of a BESS. In that context, this work seeks an automated and adaptable approach that would support the engineering process of such EMS. Therefore, available engineering approaches are studied to realize what are the open issues to be addressed. Among the studied approaches, EMSOnto—a holistic approach for multi-functional BESS—is selected as the reference methodology.

The identified open issues are handled by the EMSOnto expert, whose main role is the customization of the EMSOnto according to control engineer's requirements. As a solution, the expert proposes a set of actions based on model-driven engineering techniques and ontologies. Those are summarized as the extension of the EMS-*TBox*, setting up of new rules and queries, extension of EMS-templates, and finally the conception of data models and transformation rules. All those actions are exemplified by a selected use case example showing the realization of a CEMS, which encapsulates a series of control engineering requirements. The realization of the CEMS allows to evaluate the efficacy and limitations of the proposed solutions. The experiments were tied to specific tools and standards (MATLAB/Simulink, IEC 61850, IEC 61499, SGAM) to show the EMSOnto expert's

actions. These experiments set foundations for further customizations of EMSOnto with the aim of encompassing other standards and tools. An analysis of the tasks performed by the EMSOnto expert, after collecting and understanding the control engineer's requirements, is outlined, and discussed:

1. Define an ontology/data model of the EMS under study
2. Integrate rules and queries to the ontology
3. Propose a methodology to gather knowledge from the EMS
4. Design data models for specific software platforms, IEDs, DERs, etc.
5. Elaborate transformation rules for code/text and model generation

From the abovementioned steps, only steps 4 and 5 should be carried out to exploit information sources available at the specification stage. This is demonstrated by the CEMS example, where EMS-templates are fed with SGAM models. Hence, a consistency between specification and design phases was reached. Similarly, the abovementioned steps can also be applied to other specification and design approaches such as IntelliGrid, PSAL, etc. On the other hand, by following steps 1, 2 and 3 the inference of implicit knowledge is achieved. It was demonstrated by detecting inconsistencies within multi-functional BESS (e.g., misconfigured units) and by deducing functions at the design time (e.g., *SoC_estimator*). The proposed methodology can enable inference within other modern approaches (e.g., PSAL) as well. Besides this, a customized generation of software artifacts is achieved by performing steps 1, 4 and 5. As a result, EMSOnto is now able to support the implementation stage with generated code aligned to a controller platform (IEC 61499). In the same way, other approaches can also benefit from the referred steps to achieve compatibility with different specific platforms.

From the CEMS use case example, it is worth noticing that limitations of inferences depend on the expressivity of the ontology (e.g., OWL ontology) and the intelligence provided by rule languages. Consequently, it is encouraged to choose the right ontology language for the handling of the knowledge to be inferred. Moreover, the validation of EMS involves a dynamic process. Therefore, the identification of certain inconsistencies at that stage would require the time notion, which in a future work can be achieved by extending the expressiveness provided by ontologies with temporal logic [50]. On the other hand, with the current approaches, it is common to document the planned design (e.g., PSAL, SGAM, EMSOnto). Modifications to the stipulated design occurs often during proof-of-concept and implementations stages. However, since the modified information is not tracked then the generated software artifacts would not be aligned with the final validated design. To handle that issue, reverse engineering along the engineering process is encouraged [51]. Moreover, the validation of multi-functional BESS applications often involves tools such as communication networks and co-simulation frameworks. This means that the communication and component domains should be considered by the selected approach. However, this is not always the case since the mentioned domains are not modeled within EMSOnto. Thereby, as future work it is encouraged to combine different approaches, for instance, combining EMSOnto with PSAL where a formal semantic for communication interfaces and components is addressed.

Author Contributions: C.Z. wrote the paper and carried out the conceptualization, investigation and validation of the work. F.P.A. and T.I.S. participated in the conceptualization of the proposed approach and reviewed the final manuscript. T.I.S. supervised the overall work.

Funding: This work is partly supported by the Austrian Ministry for Transport, Innovation and Technology (bmvit) and the Austrian Research Promotion Agency (FFG) under the ICT of the Future Programme in the MESSE project (FFG No. 861265).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alizadeh, M.; Parsa Moghaddam, M.; Amjady, N.; Siano, P.; Sheikh-El-Eslami, M. Flexibility in Future Power Systems with High Renewable Penetration: A Review. *Renew. Sustain. Energy Rev.* **2016**, *57*, 1186–1193. [[CrossRef](#)]
2. Koller, M.; Borsche, T.; Ulbig, A.; Andersson, G. Review of Grid Applications with the Zurich 1MW Battery Energy Storage System. *Electr. Power Syst. Res.* **2015**, *120*, 128–135. [[CrossRef](#)]
3. EERA Joint Programme on Smart Grids-Sub-Programme 4-Electrical Energy Technologies; Technical Report D4.3 Integration of Storage Resources to Smart Grids: Possible Services, D4.4 Control Algorithms for Storage Applications in Smart Grid; EERA: Brussels, Belgium, 2014.
4. Zanabria, C.; Pröbstl Andrén, F.; Strasser, T.I. Comparing Specification and Design Approaches for Power Systems Applications. In Proceedings of the 2018 IEEE PES Transmission and Distribution Conference and Exhibition—Latin America, Lima, Peru, 18–21 September 2018; p. 5.
5. Santodomingo, R.; Uslar, M.; Goring, A.; Gottschalk, M.; Nordstrom, L.; Saleem, A.; Chenine, M. SGAM-Based Methodology to Analyse Smart Grid Solutions in DISCERN European Research Project. In Proceedings of the 2014 IEEE International Energy Conference (ENERGYCON), Dubrovnik, Croatia, 13–16 May 2014; pp. 751–758. [[CrossRef](#)]
6. Working Group Sustainable Processes (SG-CG/SP). *CEN-CENELEC-ETSI Smart Grid Coordination Group Sustainable Processes*; Technical Report; CEN-CENELEC-ETSI: Brussels, Belgium, 2012.
7. International Electrotechnical Commission (IEC). *IEC 62559-2 Use Case Methodology-Part2: Definition of the Templates for Use Cases, Actor List and Requirement List*; IEC: Geneva, Switzerland, 2015.
8. Zanabria, C.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T.I. Rapid Prototyping of Multi-Functional Battery Energy Storage System Applications. *Appl. Sci.* **2018**, *8*, 1326. [[CrossRef](#)]
9. Zanabria, C.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. Approach for Handling Controller Conflicts within Multi-Functional Energy Storage Systems. In Proceedings of the CIRED-Open Access Proceedings Journal, Glasgow, UK, 12–15 June 2017; pp. 1575–1578.
10. Andrén, F.P.; Strasser, T.I.; Kastner, W. Engineering Smart Grids: Applying Model-Driven Development from Use Case Design to Deployment. *Energies* **2017**, *10*, 374. [[CrossRef](#)]
11. Tayyebi, A.; Bletterie, B.; Kupzog, F. Primary Control Reserve and Self-Sufficiency Provision with Central Battery Energy Storage Systems. In Proceedings of the NEIS Conference, Hamburg, Germany, 21–22 September 2017; pp. 21–22.
12. Riffonneau, Y.; Bacha, S.; Barruel, F.; Ploix, S. Optimal Power Flow Management for Grid Connected PV Systems with Batteries. *IEEE Trans. Sustain. Energy* **2011**, *2*, 309–320. [[CrossRef](#)]
13. Boehm, B.; Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents*; Addison-Wesley Professional: Boston, MA, USA, 2003.
14. Andrén, F.; Lehfuss, F.; Strasser, T. A Development and Validation Environment for Real-Time Controller-Hardware-in-the-Loop Experiments in Smart Grids. *Int. J. Distrib. Energy Resour. Smart Grids* **2013**, *9*, 27–50.
15. Gottschalk, M.; Uslar, M.; Delfs, C. *The Use Case and Smart Grid Architecture Model Approach: The IEC 62559-2 Use Case Template and the SGAM Applied in Various Domains*; Springer: New York, NY, USA, 2017.
16. Weillkiens, T. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*; Elsevier: Amsterdam, The Netherlands, 2011.
17. Tornelli, C.; Radaelli, L.; Rikos, E.; Uslar, M. *WP 4 Fully Interoperable Systems Deliverable R4.1: Description of the Methodology for the Detailed Functional Specification of the ELECTRA Solutions*; Technical Report; ELECTRA IRP; 2015. Available online: http://www.electrairp.eu/index.php?option=com_attachments&task=download&id=441 (accessed on 12 March 2017).
18. Dänekas, C.; Neureiter, C.; Rohjans, S.; Uslar, M.; Engel, D. Towards a Model-Driven-Architecture Process for Smart Grid Projects. In *Digital Enterprise Design & Management*; Springer: New York, NY, USA, 2014; pp. 47–58.
19. Higgins, N.; Vyatkin, V.; Nair, N.K.C.; Schwarz, K. Distributed Power System Automation With IEC 61850, IEC 61499, and Intelligent Control. *IEEE Trans. Syst. Man Cybern. Part C* **2011**, *41*, 81–92. [[CrossRef](#)]
20. Zhabelova, G.; Vyatkin, V.; Dubinin, V. Towards Industrially Usable Agent Technology for Smart Grid Automation. *IEEE Trans. Ind. Electron.* **2014**, *62*, 2629–2641. [[CrossRef](#)]

21. Schütte, S.; Scherfke, S.; Sonnenschein, M. MOSAIK—Smart Grid simulation API—Toward a Semantic Based Standard for Interchanging Smart Grid Simulations. In Proceedings of the 1st International Conference on Smart Grids and Green IT Systems, Porto, Portugal, 21 April 2012; SciTePress—Science and Technology Publications: Porto, Portugal, 2012; pp. 14–24. [[CrossRef](#)]
22. Bhor, D.; Angappan, K.; Sivalingam, K.M. Network and Power-Grid Co-Simulation Framework for Smart Grid Wide-Area Monitoring Networks. *J. Netw. Comput. Appl.* **2016**, *59*, 274–284. [[CrossRef](#)]
23. Kroetzsch, M.; Simancik, F.; Horrocks, I. A Description Logic Primer. *arXiv* **2013**, arXiv:1201.4089.
24. Brambilla, M.; Cabot, J.; Wimmer, M. Model-Driven Software Engineering in Practice. *Synth. Lect. Softw. Eng.* **2012**, *1*, 1–182. [[CrossRef](#)]
25. Andrén, F.; Strasser, T.; Kastner, W. Model-Driven Engineering Applied to Smart Grid Automation Using IEC 61850 and IEC 61499. In Proceedings of the Power Systems Computation Conference, Wroclaw, Poland, 18–22 August 2014; pp. 1–7.
26. Zanabria, C.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. Towards an Integrated Development of Control Applications for Multi-Functional Energy Storages. In Proceeding of the IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 6–9 September 2016; pp. 1–4.
27. Morales-Trujillo, M.E.; Oktaba, H.; Piattini, M. The making of an OMG standard. *Comput. Stand. Interfaces* **2015**, *42*, 84–94. [[CrossRef](#)]
28. Horrocks, I.; Kutz, O.; Sattler, U. The Even More Irresistible SROIQ. *Kr* **2006**, *6*, 57–67.
29. Brockmans, S.; Volz, R.; Eberhart, A.; Löffler, P. Visual modeling of OWL DL ontologies using UML. In Proceeding of the International Semantic Web Conference, Hiroshima, Japan, 7–11 November 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 198–213.
30. Andren, F.; Brundlinger, R.; Strasser, T. IEC 61850/61499 Control of Distributed Energy Resources: Concept, Guidelines, and Implementation. *IEEE Trans. Energy Convers.* **2014**, *29*, 1008–1017. [[CrossRef](#)]
31. Franke, R.; Wiesmann, H. Flexible Modeling of Electrical Power Systems—The Modelica PowerSystems Library. In Proceedings of the 10th International Modelica Conference, Lund, Sweden, 10–12 March 2014; pp. 515–522. [[CrossRef](#)]
32. Buscher, M.; Kube, M.; Piech, K.; Lehnhoff, S.; Rohjans, S.; Fischer, L. Towards Smart Grid-Ready Substations: A Standard-Compliant Protection System. In Proceeding of the 2016 Power Systems Computation Conference (PSCC), Genoa, Italy, 20–24 June 2016; pp. 1–6. [[CrossRef](#)]
33. Fiaschetti, L.; Antunez, M.; Trapani, E.; Valenzuela, L.; Rubiales, A.; Risso, M.; Boroni, G. Monitoring and Controlling Energy Distribution: Implementation of a Distribution Management System Based on Common Information Model. *Int. J. Electr. Power Energy Syst.* **2018**, *94*, 67–76. [[CrossRef](#)]
34. Divya, K.; Østergaard, J. Battery Energy Storage Technology for Power Systems—An Overview. *Electr. Power Syst. Res.* **2009**, *79*, 511–520. [[CrossRef](#)]
35. Braam, F.; Diazgranados, L.M.; Hollinger, R.; Engel, B.; Bopp, G.; Erge, T. Distributed Solar Battery Systems Providing Primary Control Reserve. *IET Renew. Power Gen.* **2016**, *10*, 63–70. [[CrossRef](#)]
36. Slimani, T. Ontology Development: A Comparing Study on Tools, Languages and Formalisms. *Indian J. Sci. Technol.* **2015**, *8*. [[CrossRef](#)]
37. Hitzler, P.; Krotzsch, M.; Rudolph, S. *Foundations of Semantic Web Technologies*; CRC Press: Boca Raton, FL, USA, 2009.
38. Neureiter, C.; Uslar, M.; Engel, D.; Lastro, G. A Standards-Based Approach for Domain Specific Modelling of Smart Grid System Architectures. In Proceeding of the 2016 11th System of Systems Engineering Conference (SoSE), Kongsberg, Norway, 12–16 June 2016; pp. 1–6. [[CrossRef](#)]
39. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosz, B. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member Submission*, 21 May 2004; p. 79.
40. Paul, G.; Alexandre, P.; Axel, P. *SPARQL 1.1 Update*; W3C: Cambridge, MA, USA, 2013; Volume 21.
41. IEC. *Communication Networks and Systems for Power Utility Automation. Part 7-4: Basic Communication Structure—Compatible Logical Node Classes and Data Object Classes*; IEC: Geneva, Switzerland, 2010.
42. IEC. *Communication Networks and Systems for Power Utility Automation. Part 7-3: Basic Communication Structure—Common Data Classes*; IEC: Geneva, Switzerland, 2011.
43. Gearon, P.; Passant, A.; Polleres, A. *SPARQL 1.1 Query Language*; W3C Recommendation; W3C: Cambridge, MA, USA, 2013; Volume 21.

44. Massif: MATLAB Simulink Integration Framework for Eclipse. 2016. Available online: <https://github.com/viatra/massif> (accessed on 12 March 2017).
45. International Electrotechnical Commission. *IEC 61499-1/Ed.2: Function Blocks—Part 1: Architecture*; Standard; IEC: Geneva, Switzerland, 2012.
46. van Amstel, M.; Bosems, S.; Kurtev, I.; Ferreira Pires, L. Performance in Model Transformations: Experiments with ATL and QVT. In *Theory and Practice of Model Transformations; Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany; Zurich, Switzerland, 2011; pp. 198–212.
47. International Electrotechnical Commission (IEC). *IEC/TR 61850-90-7—Communication Networks and Systems for Power Utility Automation—Part 90-7: Object Models for Power Converters in Distributed Energy Resources (DER) Systems*; IEC: Geneva, Switzerland, 2013.
48. Uslar, M.; Specht, M.; Dänekas, C.; Trefke, J.; Rohjans, S.; González, J.M.; Rosinger, C.; Bleiker, R. *Standardization in Smart Grids; Power Systems*; Springer: Berlin/Heidelberg, Germany, 2013.
49. Gayathri, K.; Easwarakumar, K.; Elias, S. Probabilistic Ontology Based Activity Recognition in Smart Homes Using Markov Logic Network. *Knowl. Based Syst.* **2017**, *121*, 173–184. [[CrossRef](#)]
50. Baader, F.; Borgwardt, S.; Lippmann, M. Temporal query entailment in the description logic SHQ. *Web Semant.* **2015**, *33*, 71–93. [[CrossRef](#)]
51. Brunelière, H.; Cabot, J.; Dupé, G.; Madiot, F. MoDisco: A Model Driven Reverse Engineering Framework. *Inf. Softw. Technol.* **2014**, *56*, 1012–1032. [[CrossRef](#)]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

2.3 Publication C

C. Zanabria, A. Tayyebi, F. Prörtl Andrén , J. Kathan, and T. Strasser.

Engineering Support for Handling Controller Conflicts in Energy Storage Systems Applications.

Energies, vol. 10, no. 10, 24 pages, 2017.

Own contribution

The problem analysis, investigation of the related work, the selection and conceptualization of the methodology were carried by the applicant. The applicant structured the manuscript, she and the second author wrote the paper as well as performed the validation of the proposed solution. The second author contributed to the conflicts handling mechanisms which were represented in an ontology by the applicant. The third, fourth and last author proofread the manuscript. The last author carried out the supervision and mentoring of the overall work.



Article

Engineering Support for Handling Controller Conflicts in Energy Storage Systems Applications

Claudia Zanabria *,  Ali Tayyebi, Filip Pröbstl Andrén, Johannes Kathan and Thomas Strasser 

Center for Energy–Electric Energy Systems, AIT Austrian Institute of Technology, 1210 Vienna, Austria; Ali.Tayyebi-Khameneh@ait.ac.at (A.T.); Filip.Proestl-Andren@ait.ac.at (F.P.A.); Johannes.Kathan@ait.ac.at (J.K.); Thomas.Strasser@ait.ac.at (T.S.)

* Correspondence: claudia.zanabria@ait.ac.at

Received: 29 August 2017; Accepted: 28 September 2017; Published: 13 October 2017

Abstract: Energy storage systems will play a major role in the decarbonization of future sustainable electric power systems, allowing a high penetration of distributed renewable energy sources and contributing to the distribution network stability and reliability. To accomplish this, a storage system is required to provide multiple services such as self-consumption, grid support, peak-shaving, etc. The simultaneous activation of controllers operation may lead to conflicts, as a consequence the execution of committed services is not guaranteed. This paper presents and discusses a solution to the exposed issue by developing an engineering support approach to semi-automatically detect and handle conflicts for multi-usage storage systems applications. To accomplish that an ontology is developed and exploited by model-driven engineering mechanisms. The proposed approach is evaluated by implementing a use case example, where detection of conflicts is automatically done at an early design stage. Besides this, exploitable source code for conflicts resolution is generated and used during the design and prototype stages of controllers development. Thus, the proposed engineering support enhances the design and development of storage system controllers, especially for multi-usage applications.

Keywords: energy storage systems; energy management system; engineering support; smart grid architecture model; model-driven engineering; rapid prototyping; ontology; semantic web technologies; description logic; conflicts resolution

1. Introduction

A sustainable electric power supply requires the integration of renewable, Distributed Energy Resources (DER) [1]. In addition, Energy Storage Systems (ESS) presents interesting solutions for a higher share of DERs and also to support several stakeholders in electrical systems besides their contribution to maintaining power quality, reducing energy costs, and enhancing grid stability/reliability [2]. In this context, the main goal of small residential storage systems is to consume the self-generated electricity to reduce energy costs. On the other hand, they may also contribute to the enhancement of power quality and system stability (primary control reserve in a pooling scheme [3] and voltage/frequency droop control [4]). Storage systems with higher capacities provide additional economic profits and support with much more services such as participation on balancing market services, power trading, ancillary services, etc. Hence, cost-effective solutions for households/industries and a simultaneous provision of services to stakeholders require the development of a multi-use storage system approach [5]. To this aim, a wide range of controller approaches such as open/close loop, multi-agent systems, optimization methods, etc. are implemented locally/externally to the ESS. A challenge derived from this occurs when the interaction of those controllers is omitted and not handled, resulting in an unexpected controller behavior. For example,

the ELECTRA IRP project [6] points out this issue and develops an analysis of controller conflicts in scenarios with a high penetration of DERs. Concluding that overlapping of controllers could lead to the non-provision of committed services, destabilizing the whole system and in some cases harming the power quality of electrical grid. For instance an over-frequency event requires charging the ESS, at the same time the market operator requires to discharge the battery to optimize energy costs. In this case, a lack of a support to coordinate the mentioned services would prevent the battery of charging/discharging enough power to cover the requirements defined by frequency support and market services. Additionally, a battery discharging behavior, resulted from the overlapping of use cases, could incentive the exacerbation of the over-frequency state of the grid [3]. Hence, both services cannot be supported all at once, then one potential solution is the setting up of priorities per service. The “EERA Joint Programme on Smart Grids” [7], presents examples of multi-functional ESS, one example carries out secondary control power and peak/base arbitrage, those services are simultaneously supported. The alignment of them is based on the allocation of battery capacities. Another example is presented in [8], a voltage control combined with an increased self-consumption strategy is provided by a photovoltaic-battery system, an automatic voltage limitation strategy is run to coordinate the provision of both services. The aforementioned approaches justify the needed for a mechanism to analyze and handle the conflicts within a multi-service ESS context.

The development process of multi-use ESS applications should consider the identification of controllers conflicts in an early stage, before any practical implementation such as the design and elaboration of control schemes, deployment of exploitable source code, execution of offline simulations, etc. An anticipate detection of conflicts would allow to select the right control scheme configuration and to save time on doing unusable and profitable work, then reaching the promised services. Base on this, the current work proposes a methodology for a semi-automatic identification and handling of conflicts within a multi-use ESS, this process is carried out during the specification stage. Additionally, this methodology is used as a support for the rapid prototyping of ESS control applications. Two methods are currently employed to model power system control applications. They are the Smart Grid Architecture Model (SGAM) [9] and the IEC 62599 approach [10], those methodologies attempt to gather and model knowledge of a specific smart grid use case. However, models defined by them are not enough for a full identification and handling of conflicts. A preliminary idea that foster the mentioned statement is briefly outlined in [11]. It proposes a first version of a SGAM-based data model for a partial identification of conflicts within a multi-use ESS application. This work presents a first classification of conflicts and a selected conflict type is analyzed by the SGAM-based model, arguing that SGAM aligned with the IEC 62599 approach need to be extended for a further conflict analysis. In contrast with the mentioned paper, this work extends the aforementioned one by designing and implementing not only a data model but an ontology for a comprehensive analysis and conflicts resolution. Moreover, Model-Driven Engineering (MDE) exploits the proposed ontology to support domain experts during the controllers development process.

The paper is structure as follows: Section 2 addresses the related work, the process development of storage systems applications and the benefits of applying ontology and MDE-based concepts to resolve potential ESS controller conflicts and to support the development of ESS applications. In Section 3 a comprehensive classification of conflicts is provided, enabling the design of an ontology for conflicts identification. As a sequel, in Section 4 conflict resolution approaches are encouraged, motivating the broadening of the ontology to fully cover identification and handling. Finally, in Section 5 the resulted ontology is evaluated by a selected example and exploited by the MDE approach followed by the conclusions and an outlook about the future research work in Section 6.

2. Related Work and Background

2.1. Conflicts Within Storage Systems Use Cases

A brief overview of potential multi-functional Use Cases (UC) related to ESS participation mainly in low voltage power distribution grids is provided in Table 1 including a classification of services per objective and corresponding stakeholder(s). These use cases assume a small/medium scale ESS equipped with an Energy Management System (EMS) to implement and manage the provision of different services [11,12]. Hence, a multi-use ESS is reached. Those services are deployed either remotely (i.e., EMS set-points based on external decision making) or locally (i.e., set-point evaluation based on local available EMS data) as also used in the examples in Section 4. Both deployments require to set up a communication infrastructure between the EMS and external systems such as network operators (e.g., Transmission System Operator (TSO), Distribution System Operator (DSO)) and meters. To illustrate this a communication architecture that carries out the set-up of an EMS connected to an ESS is shown in Figure 1. Besides of this, the EMS sets active and reactive power values to the ESS in order to control the energy injected/consumed from the grid. In this context, a main challenge to deal with the EMS development is the coordination of single-services, thus to identify the occurrence of conflicts.

Table 1. Typical multi-functional storage system use cases.

Use Case/Name	Objective	Stakeholder	Description & Examples
electric energy time shift/UC ₁	market integration	market operator	Economical benefits are maximized. ESS is charged when the spot market prices are low and during off-peak times and usually discharged when prices are high. It is based on a daily optimization strategy [7].
voltage control/UC ₂	power quality	DSO	The rise of voltage levels can be regulated by injecting reactive power or by consuming active power. ESS participates in voltage regulation by implementing different control strategies [8,13].
primary control reserve/UC ₃	power system stability	TSO	A high penetration of DER may result in a change of the grid frequency. The ESS participates in the frequency regulation by injecting/consuming active power [3,14]. Such a service is usually used by the TSO.
peak-shaving/UC ₄	reduction of supply cost	end user	ESS is used to prevent high peaks of consumption. The ESS is discharged when a load higher than a specific set-point is switch on [15].
minimization of prices/UC ₅	reduction of supply cost	end user	Electricity costs are minimized by an objective function. It takes into account the cash received from selling energy and the cash paid for energy consumed. Moreover, forecast on load consumption, DER device generation and electricity prices (€/kWh) are calculated [16].
self-consumption/UC ₆	reduction of supply cost	end user	Local self-consumption is the main target. The difference of the local generation and demand is charged into the battery and discharged from it when the demand exceeds the local generation [7].

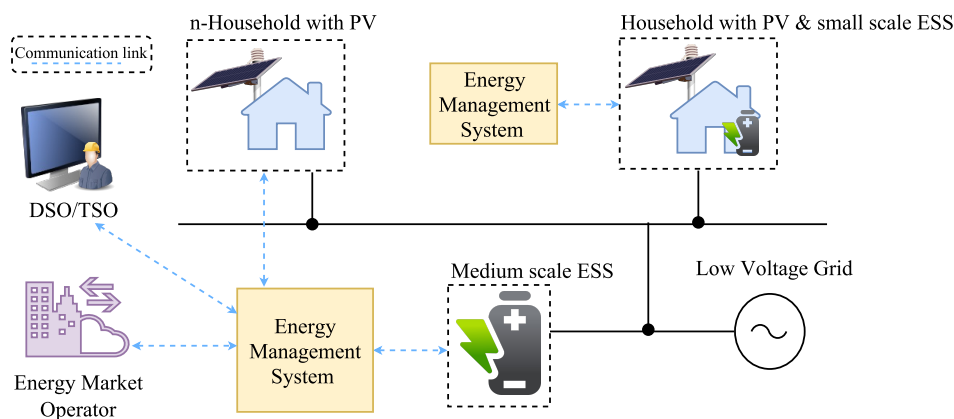


Figure 1. Communication architecture of a multi-use ESS approach.

A conflict may appear from the simultaneous interaction of stakeholders (e.g., DSO and Energy Market Operator (EMO)) orientated services. This may lead to an inefficient behavior of the EMS controller, as well as to the no provision of committed services. A conflict identification approach that targets large flexible power system architectures (e.g., virtual power plants, large-scale deployment of DER) is addressed by the ELECTRA IRP project [6]. In this study, scenarios of controller conflicts and suggestions for conflict's resolution are presented. Furthermore, a methodology for conflict identification based on a set of advises such as detailed use case specification, impact of controllers interactions on system stability, classification of conflicts by acceptable and unacceptable, etc. is presented. Against previous concept, this work proposes an engineering support to semi-automatically identify and resolve controllers conflicts in a specific power system domain: a multi-use storage system. To afford this, a domain expert or user needs to provide information about a multi-functional ESS, this process is manually done and is the starting point to proceed with a fully-automatic identification of conflicts. Additionally, handling approaches are proposed in the form of code and specific models, a validation and correction of them needs to be done manually by the user. Moreover, the proposed approach is developed in a generic way to be used in any ESS control scheme configuration. To this end, the implementation and simulation of services proposed in Table 1 is a meaningful first step to analyze the occurrence of potential conflicts and to derive a classification of causes of conflicts occurrence, this classification is carried out in Section 3.1.

2.2. Storage Systems Application Development

The main steps to be carried out during the development of ESS applications are the design, prototype and realization stages [17,18]. In the first one a conceptual design of the EMS controller is elaborated and verified. It involves a model of the electrical network, the ESS converter controller and corresponding systems or physical devices (e.g., photovoltaic inverter, load profile, Supervisory Control and Data Acquisition (SCADA)). As a sequel, the design of control algorithms to fulfill a set of services requirements is carried out. Those requirements vary depending on time-scale and accuracy. For instance, a voltage control use case requires a response time of at least 60 s and a maximum voltage deviation of $\pm 6.5\%$. Following this, a proof of concept is carried out to validate the initial specifications. This is mainly done in an offline simulator for power system simulations (e.g., Matlab/Simulink, DlgSILENT/PowerFactory) [19]. After the conceptual design is verified, a prototype is realized. Often the transformation from concept design into a prototype entails communication delays or non-linearities issues. Then, an iterative refinement of the algorithms is carried out. The realization stage requires testing the validated prototype in a real environment.

This work is focused on the analysis of requirements at the design phase with the aim of identifying conflicts between control algorithms. Additionally, possible handling solutions for those conflicts are advised and deployed during software and following hardware-based tests (e.g., pure offline and Controller-Hardware-in-the-Loop (CHIL) simulations). An early conflict detection would enable a free-of-conflict storage system application at the design phase, before the elaboration of any hands-on work such as control algorithms models implementation. Additionally, this would also support the selection of the right control scheme for a multi-functional ESS approach.

2.3. Rapid Prototyping of ESS Applications

Ontologies abstract information of a specific domain with the aim to represent objects, type of objects, and their semantic relations in a formal machine-readable way. Additionally, a set of inference rules and restrictions on relations are defined to support semantic interpretation [20]. This kind of modeling approach has also already been successfully applied to the domain of power systems: for example, reference [21] has designed an ontology to model the coordination between building energy management systems and smart grids stakeholders in a demand response context. The main entities to be modeled are communication technologies, service interfaces and grid structure. Thanks

to reasoning techniques, a matching of services and their communication technology is achieved. Besides of this, reference [22] develops an ontology by applying fuzzy theory to diagnostic faults in power transformers, demonstrating that fuzzy ontology identifies faults that are unidentifiable by a basic ontology model. The mentioned studies highlight the use of key features of ontologies such as automatic reasoning and share of knowledge in the smart grid domain. In contrast to the mentioned studies, this paper is focused on a specific power system domain: multi-use storage systems connected to low voltage power distribution grids. This system is analyzed, resulting in the development of an ontology that targets controllers conflicts. This ontology is also used to handle the localized conflicts. To achieve this, data consistency checking and inference of knowledge—assets of ontologies [23]—are performed.

A further interesting modeling approach—Model-Driven Design (MDD) or Model-Driven Engineering (MDE)—focuses on an abstract representation of knowledge by the usage of models specification [24]. An advantage of this is to support the development of software applications by improving the interoperability between systems. In this context, two main types of transformations are considered: (i) Model-to-Text transformation (M2T) that generates executable source code, documentation or other text files from source models; and (ii) Model-to-Model transformation (M2M) that transforms source models into specific target models by executing pre-defined transformation rules [25]. MDE and SGAM are lately employed in smart grid projects to gain a common understanding of smart grid architecture elements [26] and to foster the rapid prototyping of smart grid applications [18,27]. In this context, the proposed ontology is aligned to the SGAM model and exploited by the means of M2T techniques.

In summary, this paper joins concepts from MDE and ontologies to improve the rapid prototyping of multi-use ESS applications and to handle conflicts derived from the overlapping of corresponding applications. Hence, an engineering support consisting in a methodology is derived.

3. Ontologies for Multi-Use ESS Conflicts Identification

The scope of this section is to introduce a methodology supporting the identification of multi-use EES conflicts. This process is carried out during the specification phase of controllers, resulting in an appropriate control scheme to be implemented. To this end, a classification of conflicts based on their causes is proposed. Additionally, this classification is formally modeled by ontologies resulting in an EMS-ontology, a data model that targets conflicts identification. As a sequel, reasoning mechanisms are carried out to entail the deduction of conflicts.

3.1. Categorization of Controller Conflicts

The main objective of a multi-functional ESS control is to provide more than one service by injecting/consuming active or reactive power (either with locally or remotely connected EMS controllers). It is illustrated in Figure 2, where an EMS sets the power to be injected/consumed by the ESS, additionally the EMS communicates with external systems (e.g., DSO, TSO, EMO) depending on the configuration of the EMS controllers. The interaction between those controllers could lead to an undesired behavior of the ESS system and a non-provision of the corresponding service. Because of this, the reason(s) causing the conflicts need to be identified and investigated. Table 2 provides a corresponding overview of typical controller conflicts resulting from the multi-use of ESS (extended from [11]).

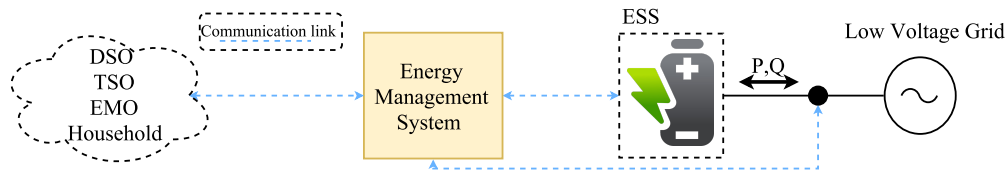


Figure 2. Injection/consumption of power to address a multi-use ESS.

Table 2. Categorization of multi-functional ESS conflicts.

Conflict Name/Type	Description & Examples
Multi-objective optimization/ C_I	Two UCs are optimizing the functions $f(x)$ and $g(x)$ respectively. This entails a conflict when $f(x)$ and $g(x)$ need to be manipulated simultaneously. For instance, an EMS supports to the stabilization of the grid frequency (UC_3) and to the minimization of energy prices (UC_5). An over frequency event would require a reduction of the active power (P). In the meantime, UC_5 requires to increase P , then a coordination of control schemes is needed.
Maximal limit dependency of control variables/ C_{II}	Two UCs are controlling u_1 and u_2 respectively. Additionally, the limits of u_1 depend on the value of u_2 (i.e., $f(u_2) \leq u_1$). For instance, an EMS contributes to voltage regulation (UC_2) by setting the reactive power (Q). Besides of this the EMS receives set-points to deliver active power (P) in a context of UC_6 provision. The limits of P depend on Q according to $P < \sqrt{S_{max}^2 - Q^2}$. Hence, an alignment of services is suitable to avoid the saturation of P and Q .
Set-point out of limits/ C_{III}	This conflict occurs when at least two UCs control the same variable and the total set-point value exceeds the limits. For instance an EMS provides Primary Control Reserve (PCR) and market service (UC_1) by receiving set-points from EMO and TSO to control P . Even when each set-point respects the limits of P an overall violation of the set-points is possible.
Set-point sign conflicts/ C_{IV}	This conflict is based on the tracking of active power provision by monitoring the State of Charge (SOC) of the ESS. On this basis, when more than one service affects the value of P by two set-points with opposite signs, the tracking of single-services is lost. For instance, an EMS provides UC_6 and UC_4 , then the value of P is set. When the total set-point is zero, the SOC remains the same leading to a wrongly non-provision of the services conclusion.
Set-point set by at least two UCs/ C_V	At least two different use cases have the intention of controlling the same variable of a system.
Interrelated manipulated variables/ C_{VI}	A UC is controlling a variable u_1 to manipulate y_1 whereas a second UC is controlling u_2 to manipulate y_2 . Additionally one of the following statements is true: $y_1 = G_{11} \times u_1 + G_{12} \times u_2$ or $y_2 = G_{21} \times u_1 + G_{22} \times u_2$. It means that the manipulated variable y_1 or y_2 is affected by the first UC and the second one. For instance, an EMS provides voltage control (UC_2) by injecting/consuming reactive power (Q). In the meantime, the TSO requires to balance the active power (P) in a UC_3 context. As a consequence, voltage of the grid is regulated by UC_2 however it is also affected by UC_3 . An analysis of the whole multi-functional system (voltage control and PCR) is required.

3.2. Ontologies for Modeling ESS Applications

The previous conflicts categorization enables the identification of data involved in a conflict occurrence. This information is used as a basis to establish a data model for an automatic investigation of conflicts, hence an ontology aligned with the SGAM model and the use case templates suggested by the smart grid coordination group—sustainable processes in [28], is developed (i.e., EMS-ontology). Ontologies introduce the terms of classes and relations [20]. A class intends to classify information by categories and a relation is defined as a relation between classes. Following this, the services involved in a multi-use ESS are classified under the class High Level Use Case (*HLUC*), this concept describes a general requirement and is independent of technical specifications and technologies. A detailed description of the functionalities covered by a *HLUC* is defined under the class Primary Use Case (*PUC*). The EMS application is encapsulated under the class *Application*, the variables controlled by an application are defined by a *ControlVar* class. On the other hand, a relation can be intuitively interpreted by looking at the related classes. For instance, the role *hasHLUC* means that an *Application* has a *HLUC*, the role *optimize* is understood as a *PUC* regulates/optimizes a variable, etc.

An overview of the proposed classes is depicted in Figure 3. This figure also shows the whole EMS-ontology by a graph representation, where a node of the graph is defined as a class and an oriented arc as a relation. Classes are structured under two main concepts: active and passive device. An active device (e.g., EMS) embeds controllers and control a passive device (e.g., ESS). More complex relation between classes (e.g., inclusions, transitivity axioms) cannot be illustrated by a graph. Hence, a formal knowledge representation is employed.

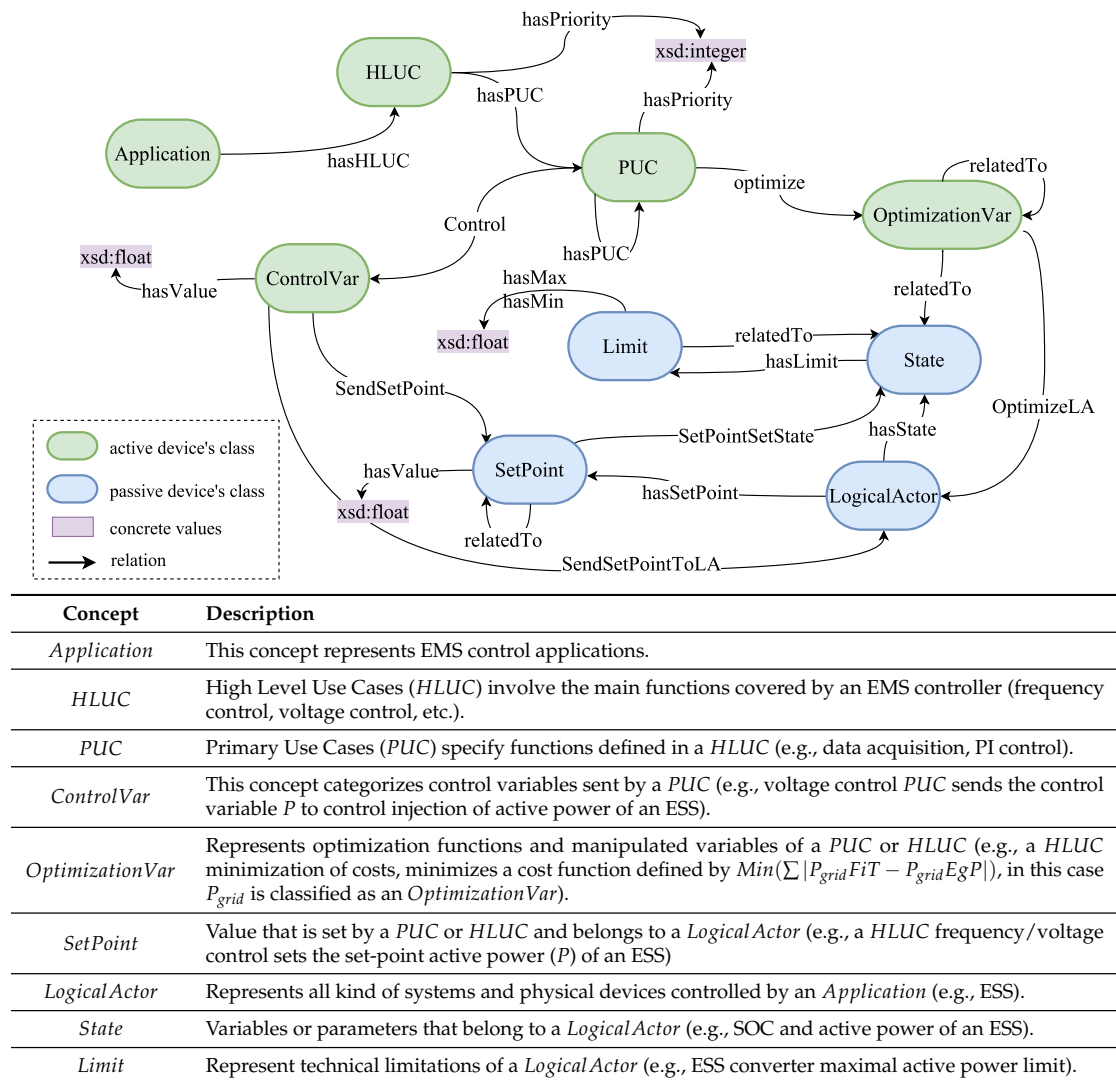


Figure 3. Overview of the proposed EMS-ontology for modeling multi-functional ESS applications.

3.3. Formal Representation of the EMS-Ontology

There exists a variety of languages for a formal knowledge representation of ontologies like Knowledge Interchangeable Format (KIF), Frame Logic (F-Logic), Description Logic (DL), etc. [29]. In this work, DL is more suitable for its expressiveness and rigor. Despite its reduced set of language constructors, DL provides a logic reasoning system, check of consistency and classification of data [23]. Thus, the previously introduced EMS-ontology is formally represented in DL notation.

DL introduces the terms *TBox* for terminological box and *ABox* for assertional box. In general the *TBox* defines concepts (i.e., classes) and roles (i.e., relations) as well as how roles and concepts are related to each other. The *ABox* establishes assertions matching individuals to concepts and roles. For instance the statement “An EMS provides two services frequency-watt and voltage control” belongs to the *ABox*, while the statement “An Application contains at least one HLUC” belongs to the *TBox*. Both assertions correspond to the proposed EMS-ontology, they are shown in Figure 4. Hence a syntax characterized by logical constructors (e.g., \sqsubseteq , \exists , $trans()$) is set [23]. DL notation assists the representation of more complex relations, for instance the constructor $trans(relatedTo)$ characterizes a transitivity of roles. It means, if $var1$ is *relatedTo* $var2$ and $var2$ is *relatedTo* $var3$ then $var1$ is *relatedTo* $var3$. The complete representation of the EMS-ontology is shown in Appendix A.

$$TBox = \{Application \sqsubseteq Application \sqcap \exists hasHLUC.HLUC, trans(relatedTo), \dots$$

$$ABox = \{HLUC(frequencycontrol), HLUC(voltagecontrol), Application(EMS), \dots$$

Figure 4. Formal representation of the EMS-ontology (extract of the *TBox* and *ABox*.)

3.4. Methodology for Identifying Controller Conflicts

One main goal of the above introduced EMS-ontology is to detect the classified conflicts from Section 3.1. To this end, the methodology depicted in Figure 5 is used. This process consists on the extraction of information from an ESS domain expert to create a knowledge base (i.e., the *ABox*). Subsequently, this data together with the *TBox* of the EMS-ontology are analyzed by a reasoner engine. A reasoner performs consistency checking and inference mechanisms to generate additional facts from the existing knowledge base. Additionally, the inferred data is queried with the aim of detecting conflicts. To this effect, a set of queries targeting conflicts is proposed.

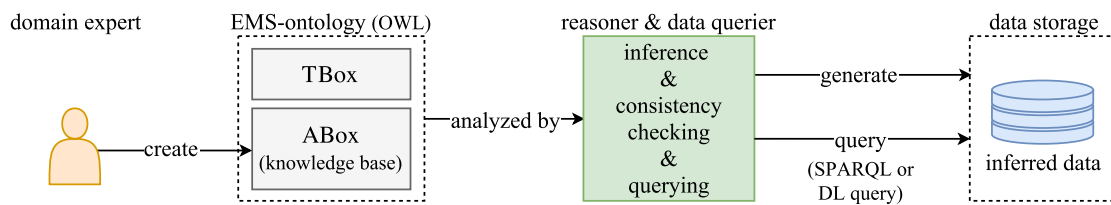


Figure 5. From knowledge base to inferred data.

The evaluation of the EMS-ontology is performed against a set of questions corresponding to the conflicts from Section 3.1. The formalization of those questions is done by stating DL queries using mathematical constructors [23]. This results in a set of appropriate DL queries to detect each conflict type as outlined in Table 3. They are executed in sequence, from this execution a report stating the identification of potential conflicts is provided. One of this DL queries is following described: The axiom $OptimizationVar \sqcap \geq 2Optimize^{-1}.PUC$ addresses variables that are optimized/regulated by at least two services (*PUC*). The other queries are intuitively apprehended by looking at the concepts and constructors meaning. In the case of *Queries 3–4*, a DL notation is not suitable due to the carrying out of arithmetic logic, then SPARQL queries are employed instead (see also Section 3.5).

Table 3. Definition of DL/SPARQL queries per conflict type.

Question/Conflict Type	DL/SPARQL Notation
<i>Question 1:</i> What are the variables that are optimized or regulated by at least two different services?/ <i>C_I</i>	$OptimizationVar \sqcap \geq 2Optimize^{-1}.PUC$
<i>Question 2:</i> What are the states that are controlled by a use case application and its limits are in turn controlled by a second control application?/ <i>C_{II}</i>	$Limit \sqcap \exists relatedTo. (State \sqcap \exists SetPointSetState^{-1}. (SetPoint \sqcap \exists SendSetPoint^{-1}. (ControlVar \sqcap \exists Control^{-1}.PUC)))$
<i>Question 3:</i> Does a set of variables exist which intends to control a set-point of an ESS, causing a violation of technical limits imposed to this set-point?/ <i>C_{III}</i>	supported with SPARQL (see Appendix C)
<i>Question 4:</i> Are there a set of control variables that control a determined set-point of an ESS and the multiplication of their values is negative?/ <i>C_{IV}</i>	supported with SPARQL (see Appendix C)
<i>Question 5:</i> What are the variables that control a same set-point of an ESS?/ <i>C_V</i>	$SetPoint \sqcap \geq 2SendSetPoint^{-1}. (ControlVar \sqcap \exists Control^{-1}.PUC)$
<i>Question 6:</i> What are the variables that are manipulated by a service and are affected by a second use case?/ <i>C_{VI}</i>	$OptimizationVar \sqcap \exists optimize^{-1}.PUC \sqcap \exists relatedTo. (State \sqcap \exists control^{-1}.PUC)$

3.5. OWL and SPARQL to Evaluate the EMS-Ontology

The previous section defined a set of questions to evaluate the efficacy of the EMS-ontology. Some of those questions—*Questions 3–4*—need to carry out automatically a sequence of queries and to deal with concrete values as well. Hence, it is suitable to implement the proposed methodology for conflicts identification by employing Web Ontology Language (OWL) and SPARQL query language approach [20].

OWL is formally founded on description logic principles, it handles concrete properties defined in DL using datatypes (e.g., `xsd:float`). Moreover, OWL is part of the W3C standardized ontology languages. SPARQL a graph-based query language is also an official W3C recommendation. It extracts information from an OWL ontology and introduces a large list of functions for querying data enabling a higher flexibility compared to DL queries (e.g., *GROUP*, *ORDER BY*). Thus, SPARQL notation is used to formulate *Questions 3–4* (for details see Appendix C).

4. Handling of Conflicts within ESS Applications

The previous section classifies conflicts and presents a mechanism to detect them by the usage of ontologies. This section is focused on the handling of those conflicts, thus, potential solutions for conflict's resolution are presented. Afterwards, the implementation of those solutions is addressed by extending the EMS-ontology and by using a MDE-based approach to derive an EMS controller implementation.

4.1. Handling of Solutions per Conflict Type

A multi-functional ESS setup is illustrated in Figure 6. A set of n services (*Service_i* for $i = 1, \dots, n$) is provided by the ESS equipped with an EMS controller. Each service is identified with its corresponding active and/or reactive power set-points P_i and/or Q_i . For some services, such as local grid voltage support, the set-point evaluation is performed locally. On the other hand, set-points can be externally identified—hereinafter called remotely—and transferred to the EMS controller (e.g., for an energy market associated service [12]).

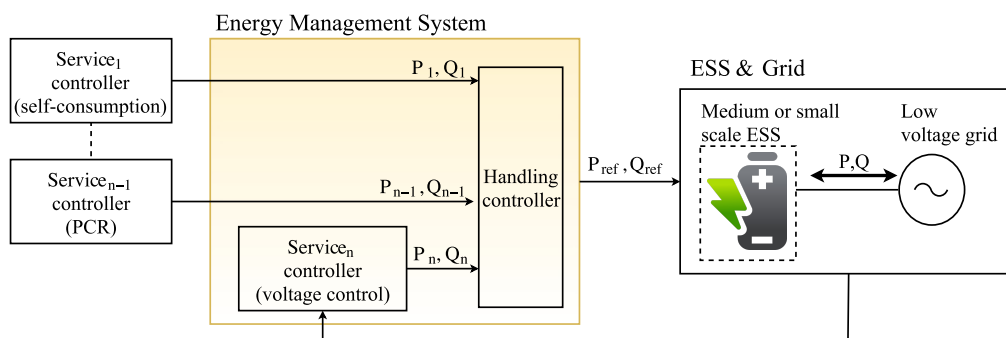


Figure 6. Multi-functional ESS equipped with an embedded EMS controller.

4.1.1. Set-Points Down-Regulation

To begin with, it is supposed that the set-point values P_i are identified and transmitted to the EMS controller. Furthermore, it is assumed that each signal is smaller than the ESS converter maximal active power limit P_{max} (i.e., $|P_i| < P_{max}$ for $i = 1, \dots, n$). As a result of independent set-point identification mechanisms, the total set-point P_{ref} can exceed the P_{max} value (i.e., $P_{ref} = \sum_{i=1}^n P_i > P_{max}$). For instance, a use case set-up is defined by a P_{max} that is limited to 100 kW and two services that intend to control the set-point P_{ref} by the values 40 kW and 80 kW respectively. Even if independently the services do not exceed the technical limitations (100 kW), the accumulation of them (40 kW + 80 kW) could do. Thus, the set-points must be down-regulated. To this end, for each

service a corresponding priority is defined. Assuming that the service n has the lowest priority if the maximal limit is exceeded, the set-point P_n is down-regulated as

$$P_{n-new} = \text{sgn}(P_n) \left| P_{max} - \sum_{i=1}^{n-1} |P_i| \right| \quad (1)$$

Similarly, if still the sum is greater than P_{max} , then P_n is set to zero and the next least-prior term is down-regulated. Hence, the corresponding services are partially provided. This handling solution corresponds to the conflict C_{III} .

4.1.2. Converter PQ Range Limiter

The handling solution proposed in the previous section makes it possible to define the reference ESS active and reactive power such that all the services are considered

$$P_{ref} = \sum_{i=1}^n P_{i-new}, \quad Q_{ref} = \sum_{i=1}^n Q_{i-new} \quad (2)$$

Consequently, the reference apparent power is defined by

$$S_{ref} = \sqrt{P_{ref}^2 + Q_{ref}^2} \quad (3)$$

This value is limited by the ESS apparent power limit (S_{max}). Although all the individual service set-points associated with active and/or reactive power provision are smaller than the maximal limits (i.e., $P_{i-new} < P_{max}$ and $Q_{i-new} < Q_{max}$ for $i = 1, \dots, n$) still the ESS apparent power limit can be violated i.e., $S_{ref} > S_{max}$ exemplifying a conflict C_{II} . Thus, the corresponding P_{ref} and Q_{ref} must be down-regulated such that $S_{ref} = S_{max}$. A potential solution is detailed in the literature [30].

4.1.3. SOC Estimation and Capacity Allocation

In order to illustrate type C_{IV} conflict occurrence in this application, let the service set-points be

$$P_i \neq 0 \quad \text{and} \quad P_j \neq 0 \quad \text{for} \quad i, j = 1, \dots, n \quad \text{such that for any} \quad (i, j), i \neq j \quad \text{and} \quad P_i \cdot P_j < 0 \quad (4)$$

This means that, the set-points P_i and P_j , targeting the active power value of an ESS, have opposite signs. This corresponds to the conflict C_{IV} . In such a conflicted scenario, it is important to virtually track the impact of each set-point. This is realized by virtually allocating a portion of the ESS capacity to each service. Furthermore, a state of charge estimator must be implemented for each capacity portion. State of charge estimator evaluates the SOC changes by considering the allocated capacity, battery voltage measurement and the power set-point defined by Equation (4). A detailed description of this handling solution is described in [30].

4.1.4. Use Case Specific Solutions

A general solution can not be encouraged for conflict C_{VI} . Hence, the handling solution must be employed depending on the use case specification/requirement. For instance, a multi-functional ESS provides services which define the P_{ref} . Additionally, the Point of Common Coupling (PCC) voltage support is also offered, see Figure 1. Voltage support has to be provided so that the PCC voltage V_{pcc} does not violate the limits (V_{upper} and V_{lower}) defined by the DSO. The steady state PCC voltage in terms of active/reactive power injected by the ESS is expressed by

$$V_{pcc} = V - \frac{RP_{ref} + j\omega LQ_{ref}}{V} \quad (5)$$

where the R, L are the approximated grid resistance and inductance and V is the grid voltage. Based on Equation (5) the active power provision affects the PCC voltage. Thus, the simultaneous provision of voltage support (i.e., $Q(V)$ control) and active power associated services are in conflict, this corresponds to a conflict C_{VI} . One solution to handle this conflict is based on a dynamic reactive power control [30]. In this approach, an integral controller with a proportional gain dynamically adjusts Q_{ref} such that the voltage always stays within its limits $V_{lower} \leq V_{pcc} \leq V_{upper}$.

A similar methodology suggesting dynamic regulation of PCC voltage by the means of both active and reactive power is proposed by [13]. Alternatively, the solution presented in [8] evaluates the compensating Q_{ref} according to a pre-set $Q - V$ characteristic curve defined by [4]. Comparing the aforementioned approaches shed lights on the fact that the choice of voltage regulation strategy mainly depends on the application requirements, network voltage level and ESS sizing.

4.2. Extended EMS-Ontology for Handling Conflicts

A main goal of the EMS-ontology, besides conflicts identification, is the handling of them. To this end, an extension of this ontology is required. This extension is based on the definition of patterns targeting the previously addressed handling solutions. A pattern is easily identifiable for the cases where a general solution is provided (e.g., “Converter PQ Range Limiter” as outlined above). Moreover, the information gathered from the current EMS-ontology is enough to identify and propose a corresponding general solution. However, in case of specific solutions the information modeled with the current ontology is not enough to precise a conflict resolution. It means extra information from the multi-functional ESS context is required (ESS capacity, network operator requirements, etc.). This paper proposes an extension of the actual EMS-ontology to implement the mentioned specific/general handling solutions. For simplicity, only the solution “Down-Regulation of Set-points” is discussed in detail in this article.

A graph illustrating new concepts and roles for implementing “Down-Regulation of Set-points” is shown in Figure 7. A validation of this ontology is based on the collection of enough information to carried out according to Equation (1). To this end, information provided by the EMS-ontology such as priority of services, physical device to be controlled (e.g., ESS), set-point values out of limit (e.g., $\sum_{i=1}^n P_i > P_{max}$, for $i = 1, \dots, n$) and set-point to be affected (P_{ref}) are evaluated. As a result, set-points are scaled-down in compliance with the maximal converter active power (P_{max}). An evaluation of the extended ontology is done by an use case implementation in Section 5.3.

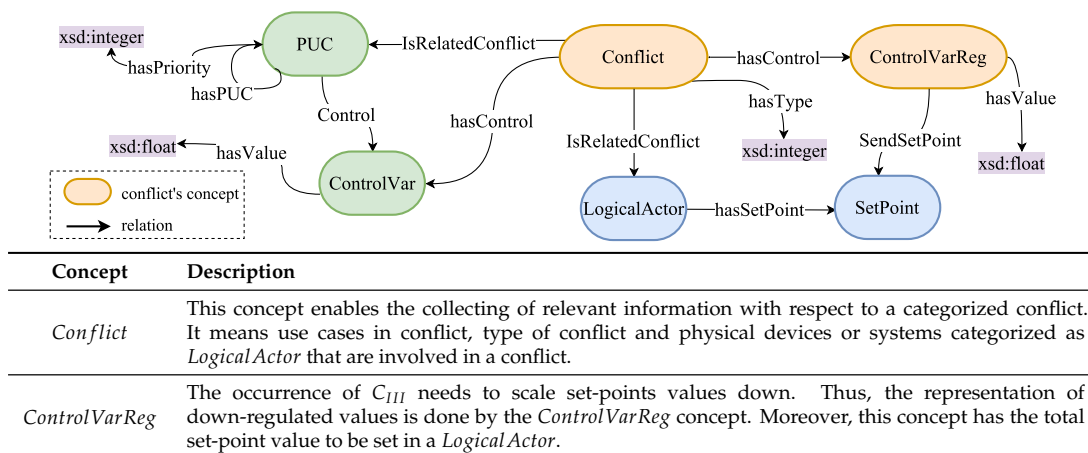


Figure 7. Extension of the EMS-ontology for the handling of controller conflicts.

4.3. EMS-Ontology Exploited by the MDE Approach

The MDE approach is focused on the modeling of applications. Once the model is accomplished the conduct of M2M and M2T takes place. To this end, a meta-model and a source model are required. A way to apply ontology engineering to the MDE approach is discussed in [31]. In this work, a meta-model is considered as an ontology (*TBox*) and a source model as an instance of the ontology (*ABox*). Previous sections described an EMS-ontology to detect and manage conflicts. The fact of seeing EMS-ontology as a meta-model and a multi-functional ESS use case as a source model brings the benefit of generating exploitable source code (e.g., C++, Matlab) and target models (e.g., Simulink models). The resulted model or text is meant to be deployed during the elaboration and validation of the ESS application. This attempts to demonstrate that the EMS-ontology supports during the rapid prototyping of ESS control applications. A proof-of-concept is addressed in the following section.

5. Proof of Concept

In this section a practical multi-functional ESS application is introduced. This example aims to highlight the conflicts occurrence and propose corresponding handling solutions by means of an ontology-based methodology support. Thus, a Smart Low Voltage Grid Controller (SLVGC) is implemented which integrates the handling solutions presented in Section 4. In the following, the application setup and ESS functionalities/services are described. In this control solution the ESS is required to provide self-consumption and market services in smart low voltage grid. Furthermore, from the power quality perspective the ESS has to provide local voltage support. In the following each service and the setup configuration are described.

5.1. ESS Services and Setup

In this setup, the ESS helps the household Photovoltaic (PV) systems in balancing the generation and consumption profiles. In other words, every household rents a portion of the storage capacity in order to store the excess PV generated power and consume it in high-load time intervals. The amount of power to be charged/discharged is equal to the generation-consumption difference and it is calculated by the smart metering devices installed at the household PCCs. Accordingly, these measured values are steadily sent to the SLVGC and all together form the total self-consumption set-point P_{sc} . The individual self-consumption signals and the sum P_{sc} are updated every minute. From the SLVGC point of view, P_{sc} is considered as an externally defined set-point. In this application, self-consumption is provided to 10 households renting 1/2 of the total ESS capacity. The set-point for this service P_{ms} is identified by the EMO. Hence, the storage owner can sell the stored energy in high-price time periods, and recharge the market allocated capacity in a low-price period. This service also rents 1/2 of the total ESS capacity. As a result of the resistive nature of the low voltage grid and multiple service provision, the PCC voltage may exceed the permitted limits. Hence, PCC voltage is regulated by means reactive power injection-absorption (Q_{gs}). PCC voltage controller is one of the SLVGC integrated schemes. Thus, the grid support control variables Q_{ref} is locally evaluated.

The setup configuration is demonstrated in Figure 8. The input signals P_{sc} and P_{ms} are sent to the SLVGC hardware via the communication layer (e.g., utilizing a Modbus TCP protocol). The SLVGC also receives measurements from battery (SOC_{bat} and V_{bat}), injected power by ESS (P and Q), and the PCC voltage (V_{pcc}).

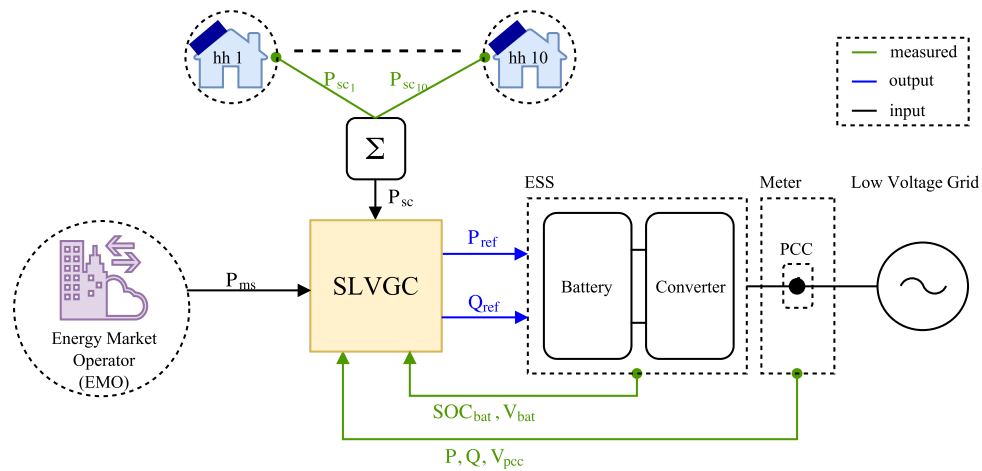


Figure 8. SLVGC control approach for a multi-functional usage of ESS.

5.2. Applying the Ontology-Based Methodology

An engineering support consisting of the above introduced ontology-based methodology for detecting controller conflicts and for handling corresponding solutions is in the following example applied and demonstrated. The detection of the overlapping of ESS services is based on the analysis of data provided by domain experts (e.g., control or power system engineer). This data defined as knowledge base of the SLVGC application (*ABox*) is evaluated by reasoning mechanism resulting in the derivation of additional truths (inferred data). As a sequel, inferred data is queried to identify conflicts. This process continues with the creation of a new instance of the EMS-ontology to manage conflicts (updated data). The whole cycle is shown in Figure 9.

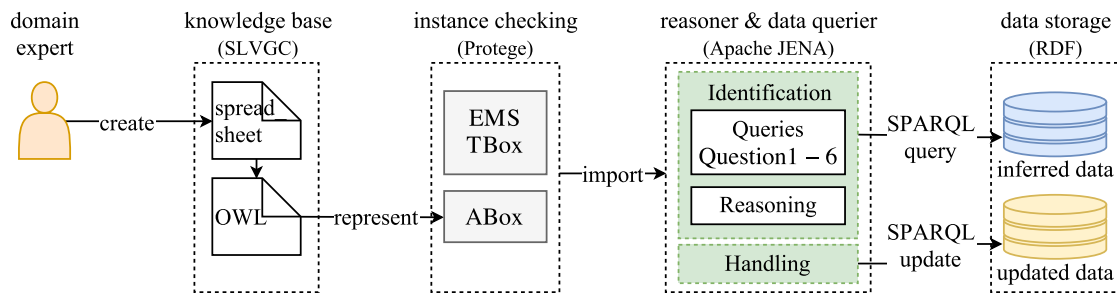


Figure 9. Ontology-based methodology applied to a selected use case example.

Domain specialists own the expertise about the services to be implemented within the SLVGC controller. That is why their knowledge related to the multi-functional usage of ESS need to be collected and analyzed. To achieve a comprehensive data collection, templates for spreadsheets are designed where relevant domain knowledge is stored in tabular and therefore understandable form. However data in that format is not ready to be analyzed, then a transformation from spreadsheets into an instance of OWL EMS-ontology is required. The knowledge resulted from this transformation is shown in Figure 10.

$$\begin{aligned}
 ABox = \{ & Application(SLVGC), HLUC(MarketService), HLUC(SelfConsumption), HLUC(VoltageControl), \\
 & OptimizationVar(V_{pcc}), ControlVar(P_{ms}), ControlVar(P_{sc}), ControlVar(Q_{gs}), \\
 & SetPoint(P_{ref}), SetPoint(Q_{ref}), LogicalActor(ESS), LogicalActor(Meter), \dots
 \end{aligned}$$

Figure 10. Extract of SLVGC knowledge base.

It shows that *SLVGC* is defined as an instance of the concept *Application*, besides of this the concept *HLUC* models the services provided to the EMO, the customers, and the DSO. The *ControlVar* concept gathers control variables. The *SLVGC* implements local voltage control, then the regulation of the voltage at the PCC point V_{pcc} is modeled by the concept *OptimizationVar*. The ESS and the meter that carries information of the low voltage grid are defined by the *LogicalActor* concept. The ESS device incorporates two set-points to set active and reactive power by external systems (i.e., P_{ref} , Q_{ref}). Those values are defined as *SetPoint* concepts. The exhaustive knowledge of the *SLVGC* application is presented in Appendix B.

As a first attempt to validate the EMS-ontology and the constancy of the *SLVGC* knowledge (i.e., *ABox*), the Protege toolkit—an open-source ontology editor—is used. This tool enables the integration of a variety of reasoners (Pellet, FaCT++, HermiT, etc.) and the evaluation of DL and SPARQL queries through the setup of plug-ins. Thus, Protege is convenient for validation of the EMS-ontology and consistency checking of the *SLVGC* knowledge base. However, it is limited in terms of implementation of extensive queries, it is the case for *Question 3–4* (see Table 3). To cover those gaps, JENA—a Java framework [32]—is employed for the implementation of queries as presented in Table 3. It supports the deployment of OWL ontology language, the connection to inference engines and the usage of a SPARQL processor. This query engine performs operations to create, update and remove data from a Resource Description Framework RDF store through SPARQL update [33], as depicted in Figure 9.

5.3. Exploiting Inferred Data

A benefit of an ontology-based application is the derivation of additional truths from the knowledge base. In this context, one assertion derived from *SLVGC* knowledge base (see *ABox* in Figure 10) is that *VoltageControl* service affects the active power value of the grid (P). This deduction is entailed from the axiom $Optimize \circ relatedTo \sqsubseteq Optimize$ and the statements $Optimize(VoltageControl, V_{pcc}), relatedTo(V_{pcc}, P)$. Many others deductions are automated inferred by the engine reasoner. The inferred data is queried by SPARQL and DL queries to achieve the detection of conflicts type. The SPARQL and DL notation used to establish the aforementioned queries is fully presented in Appendix C. Conflicts resolution mechanisms are employed according to the study presented in Section 4. This is shown in Table 4, moreover answers derived from queries are also exposed.

Handling solutions addressed in Table 4 are satisfactory implemented. The solution for conflict C_{III} is illustrated in Figure 11. Implementations of calculations introduced in Equation (1) are carried out in the JENA framework. The resulting values are used to create an instance of the EMS-ontology (i.e., *SLVGC_conflict*). This instance gathers active power values coming from market and self-consumption services (P_{ms} , P_{sc}). When the sum of those values exceeds the technical limitations of the ESS ($P_{max} = 100kW$) they need to be down-regulated. A way to exploit the instance *SLVGC_conflict* is by carrying out M2T transformations, resulting in the automatic generation of source code (e.g., Matlab code). This code is deployed into a power system simulator (e.g., Simulink) to execute off-line simulations of the *SLVGC* controller. A similar approach is employed for the other handling solutions mentioned in Table 4. Hence, the reduction of manual work during the controllers development process is achieved.

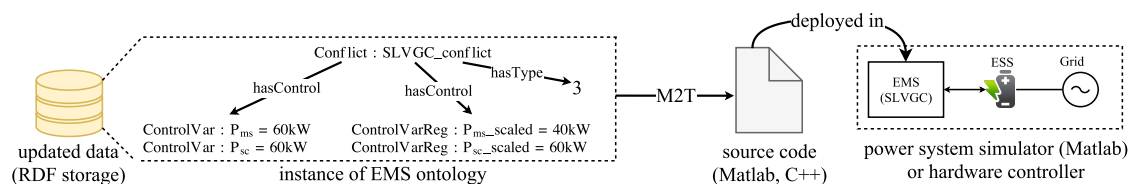


Figure 11. Data for handling conflict C_{III} and the corresponding M2T process.

Table 4. Querying the SLVGC ontology and handling solutions per conflict type.

Conflict/Type	Detected	Conclusion Derived from Queries	Handling Solution
Multi-objective optimization/ C_I	X	There is not any variable that is intended to be optimized or regulated by two different use cases then conflict C_I is not identified. Control strategies regarding self-consumption and market services are run externally, thus the SLVGC application receives set-points from household and EMO. It could be the case that those services are in conflict but as the SLVGC application has no further information regarding those services then C_I is dismissed.	Not required
Maximal limit dependency of control variables/ C_{II}	✓	The limit of Q is defined by the state $P(\sqrt{S_{max}^2 - P^2})$. This state P is controlled by the use cases self-consumption and market service through the control variables P_{sc} and P_{ms} . Additionally, the state Q is controlled by voltage control use case through Q_{gs} . A conflict C_{II} involving the control variables $\{P_{sc}, Q_{gs}\}$ and $\{P_{ms}, Q_{gs}\}$ is detected. Then a coordination between the services in conflict is required to avoid a saturation of P and Q .	Converter PQ range limiter
Set-point out of limits/ C_{III}	✓	The set-point P_{ref} is set by two control variables: P_{ms} and P_{sc} . The total value to be set exceeds the active power limit P_{max} , then a conflict C_{III} is identified	Down-regulation of set-points
Set-point sign conflicts/ C_{IV}	X	The values of control variables P_{ms} and P_{sc} have the same sign then a conflict C_{IV} is dismissed. However those values evolve over time, thus the EMS-ontology cannot predict the conflict C_{IV} . Identification of conflict C_{IV} should take place during real-time operation or simulation of the SLVGC application.	SOC estimation and capacity allocation
Set-point set by at least two use cases/ C_V	✓	The controllers market service and self-consumption have the intention of setting the set-point P_{ref} of the ESS, then conflict C_V is detected. This conflict is not considered harmful by the domain expert. Thus, no handling solutions are executed.	Not required
Interrelated manipulated variables/ C_{VI}	✓	The use cases market service and self-consumption control the state P . Additionally the value of this state affects the PCC voltage ($V_{pcc} = V - \frac{RP + j\omega LQ}{V}$). Thus, V_{pcc} is affected by market service and self-consumption use cases. On the other hand, V_{pcc} is manipulated by voltage control to regulate the PCC voltage, then C_{VI} is identified.	Reactive power voltage controller

5.4. Simulation Result

In this section the simulation results of the multi-functional ESS application introduced above is presented. The ESS and distribution feeder are modeled based on the approach suggested by [30,34,35]. The simulation is performed in Matlab/Simulink. The total simulation time is set to 2 h and the model parameters are described in Appendix D. As previously explained above, the SLVGC receives self-consumption and market service set-points, P_{sc} and P_{ms} respectively. The implemented SLVGC integrates the handling solutions introduced in Section 4.

The achieved results are depicted in Figure 12. In this illustration, the power, voltage and SOC values are respectively normalized with S_{max} , V and 100% (see Appendix D). Furthermore, the measurement reference frame is chosen such that positive/negative power values correspond to charge/discharge operation. According to Equation (5), charge/discharge state results in PCC voltage drop/increase. In Figure 12, the two plots (I) and (II) are associated self-consumption and market service provision. In these plots the original set-points are shown in blue (i.e., P_{sc} and P_{ms}). The time resolution for P_{sc} and P_{ms} are 1 and 15 min respectively. P_{sc} exhibits a period of high PV generation followed by an oscillating profile caused by a sunny-cloudy weather condition. For the sake of compactness, the pure consumption state ($P_{sc} < 0$) is neglected. However, the original P_{ms} shown in plot (II) requires both charge and discharge operations. In these plots, the green signals correspond to the down-regulated signals if the ESS P_{max} limit is exceeded (see Section 4.1.1). In this application, P_{sc} has higher priority, thus the down-regulation is applied to P_{ms} , resulting in

$$P_{sc-new} = P_{sc}, \quad P_{ms-new} = \text{sgn}(P_{ms}) \left| P_{max} - |P_{sc}| \right| \quad (6)$$

This explains the intervals in plot (II) where the green signal does not follow the blue signal. The regulation of the voltage, active and reactive power are shown in Figure 13 by means of plots (III) and (IV). In plot (III), the active, reactive and apparent power profiles are shown. Based on Equations (2) and (3) the ESS set-points are defined. Following the argument presented in Section 4.1.2, if the S_{max}

limit is violated the ESS apparent S_{ref} is limited. This results in down-regulating P_{ref} and Q_{ref} accordingly, hence

$$S_{ref-lim} = S_{max}, \quad P_{ref-lim} = \left(\frac{S_{max}}{S_{ref}}\right)P_{ref}, \quad Q_{ref-lim} = \left(\frac{S_{max}}{S_{ref}}\right)Q_{ref} \quad (7)$$

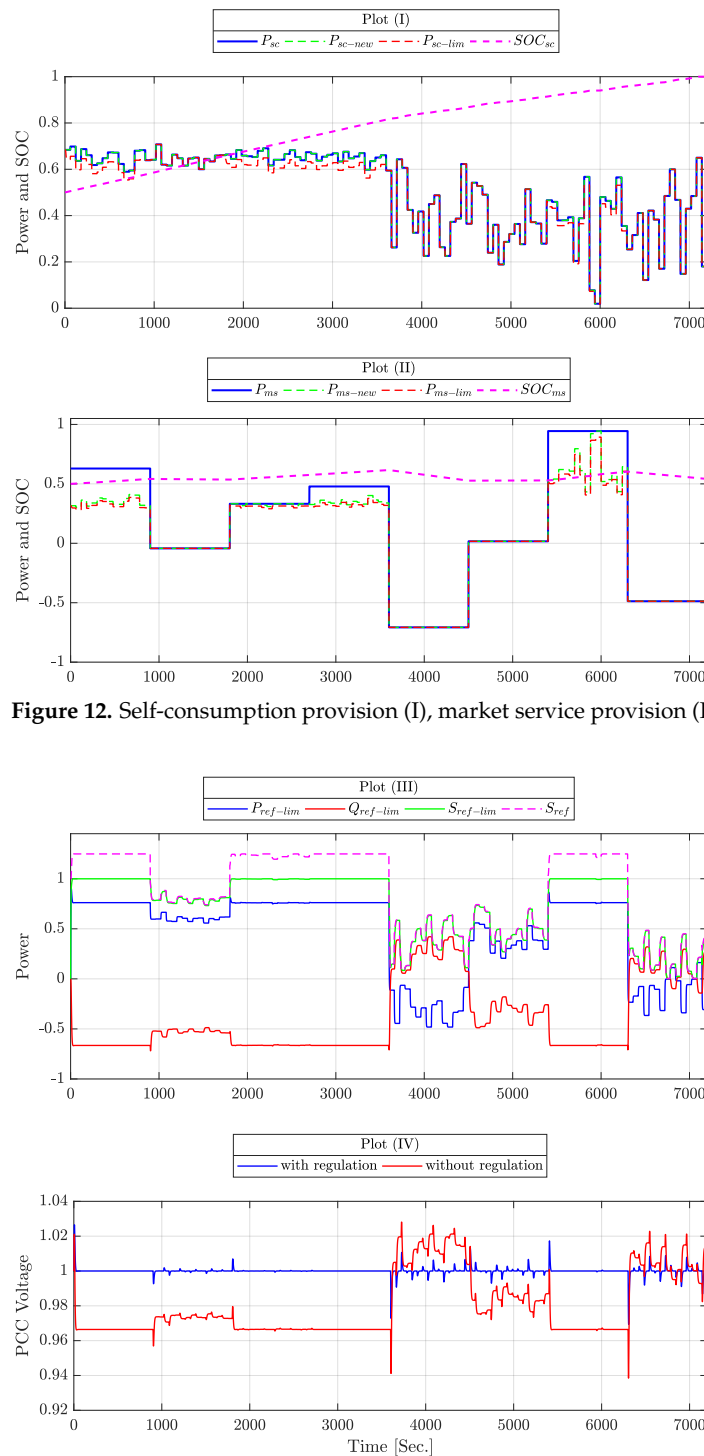


Figure 12. Self-consumption provision (I), market service provision (II).

Figure 13. ESS power profiles (III), PCC voltage control (IV).

This corresponds to the intervals where the green signal is limited to 1 in plot (III). As a consequence of down-regulating P_{ref} to $P_{ref-lim}$, the associated market and self-consumption signals are also scaled down with the same ratio as in Equation (7), resulting in P_{sc-lim} and P_{ms-lim} . These signals are shown in red in plots (I) and (II). Sequentially, these final services set-points are feedbacked to state of charge estimation schemes mentioned in Section 4.1.3. The estimated state of charge evolution is depicted in plots (I) and (II) reflecting the availability of allocated capacity for each service. Last but not least, the reactive power set-point Q_{ref} is evaluated by the PCC voltage controller which is implemented based on the strategy in [30]. By construction, Q_{ref} always has an opposite sign compared to P_{ref} . The PCC voltage controller impact can be observed in plot (IV). For instance, when $P_{ref} > 0$ the ESS is charging. Thus, the PCC voltage drops from its nominal value. Consequently, the controller reacts by adjusting $Q_{ref} < 0$ such that PCC voltage regains the nominal value as it's steady state.

6. Conclusions and Future Work

With the large-scale integration DER into power distribution grids new services provided by ESS are required, thus the development of a corresponding multi-functional usage of them is now becoming a reality. This trend faces a difficult issue derived from the overlapping between control targets. Thus, an engineering support for an early detection and handling of conflicts during the requirements analysis of control structure is necessary. This methodology will enable a correct planning and management of ESS control applications.

The main basis of this solution lays on the definition of an ontology (i.e., EMS-ontology), overly simplified to identify and resolve conflicts. The presented EMS-ontology based approach demonstrates that an early requirements analysis of the control structure enables the identification and handling of conflicts. Moreover, a convenient and user-friendly way to gather knowledge from domain experts is also tackled by proposing a spreadsheet template. This knowledge is modeled by means of ontologies, hence reasoning mechanisms enable the inference of new facts. This resulted information is queried by predefined queries targeting conflicts classified by types (see Tables 2 and 3). Additionally, a set of handling solutions per conflicts type is encouraged, this motivates a semi-automatic generation of code to support conflicts resolution and to be implemented during the development of the system. This enables the rapid prototyping of ESS applications and a preventive correction of handling of conflicts before any practical implementation.

Model-based strategies to design the requirements of power systems are addressed by different standards and approaches. However, information gathered by them are not enough for a full conflict identification. A well-known approach to specify and structure the requirements of smart grid applications is defined by SGAM. This approach is being used in divers smart grids projects [26]. Despite of this, a full conflict detection is not totally ensured [11]. Additionally, the use case methodology introduced by IEC 62559 [10] gives the basis to specify requirements for energy systems control applications, but the identification of conflicts is not really addressed. In contrast with those methodologies, the engineering support approach defined by this paper does not provide a model for designing all the aspects of power systems applications. This motivates to concentrate future work on the alignment of the proposed approach with power system modeling methodologies such as SGAM and IEC 62559. This would improve the development chain of control applications, enabling a full description of power system application as well as an automatic conflict identification from the very beginning of the development process. Furthermore, the support of interoperability between the EMS with external systems such as TSO and EMO to import knowledge about controllers implemented remotely to the EMS should be assured. This entails the alignment of the EMS-ontology with other smart grid information models such as IEC 61850 and Common Information Model (IEC 61970), both are standards to improve the interoperability in electric power systems [36].

The literature shows only first preliminary ideas that addresses conflicts within a multi-use ESS context. An approach that targets the identification of conflicts within electric energy systems is

studied in [37], this study recommends the use of the language Multi-level Flow Modeling (MFM) to model the requirements of control structures. Additionally, this approach is aligned to the IEC 62559 in the scope of the ELECTRA IRP project [38]. However, because of the large range of power system applications covered by this solution, a correct effectiveness and also its limitations are unknown. Compared with this approach, not only the identification but the handling of conflicts are addressed by the EMS-ontology. Moreover, this ontology targets a restricted domain: multi-use ESS, then the objective design criteria is very well defined by a set of selected questions, as outlined in Table 3. On the other hand, a correct design of ontologies depends on the initial knowledge base, the EMS-ontology is built from use cases presented in Table 1, hence is not exhaustive for all the conflicts derived from multi-functional storage systems. Thus an efficient EMS-ontology would require the knowledge of a larger list of use cases. This may entail the extension of handling solutions resulting in an evolution of the ontology as well. Hence, the designing of the EMS-ontology is an ongoing and non-exhaustive process.

Acknowledgments: This work is partly supported by the Austrian Ministry for Transport, Innovation and Technology (bmvit) and the Austrian Research Promotion Agency (FFG) under the ICT of the Future Programme in the OpenNES project (FFG No. 845632).

Author Contributions: All the authors contributed to the main idea of the paper. Claudia Zanabria and Tayyebi Ali wrote the paper. Thomas I. Strasser supervised the overall work. Filip Pröbstl Andrén, Johannes Kathan, and Thomas I. Strasser proofread the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

Acronyms

CHIL	Control-Hardware-in-the-Loop
DER	Distributed Energy Resources
DL	Description Logic
DSO	Distribution System Operator
EERA	European Energy Research Alliance
EMO	Energy Market Operator
EMS	Energy Management System
ESS	Energy Storage System
HLUC	High Level Use Case
IEC	International Electrotechnical Commission
IRP	International Reporting Project
M2M	Model-to-Model
M2T	Model-to-Text
MDD	Model-Driven Design
MDE	Model Driven Engineering
OWL	Web Ontology Language
PCC	Point of Common Coupling
PUC	Primary Use Case
PV	Photovoltaic
RDF	Resource Description Framework
SCADA	Supervisory Control and Data Acquisition
SGAM	Smart Grid Architecture Model
SLVGC	Smart Low Voltage Controller
SOC	State-of-charge
TSO	Transmission System Operator
UC	Use Case

Nomenclature

P_i	Active power set by the service i
Q_i	Reactive power set by the service i
S_{ref}	Apparent power set-point of a battery inverter
P_{ref}	Active power set-point of a battery inverter

Q_{ref}	Reactive power set-point of a battery inverter
Q_{max}	Maximum reactive power limitation of a battery inverter
P_{max}	Maximum active power limitation of a battery inverter
S_{max}	Maximum apparent power limitation of a battery inverter
P_{i_new}	Down-regulated value of P adopted by the service i
Q_{i_new}	Down-regulated value of Q adopted by the service i
V	Voltage of the grid
P	Active power injected/consumed at the PCC node
Q	Reactive power injected/consumed at the PCC node
V_{pcc}	Voltage at the point of common coupling
R	Grid resistance
L	Grid inductance
P_{sc}	Active power set by the self-consumption service
Q_{gs}	Reactive power set by the grid support controller
P_{ms}	Active power set by the market service
P_{sc_new}	Down-regulated value of P_{sc} according to Set-point Down-Regulation strategy
P_{ms_new}	Down-regulated value of P_{ms} according to Set-point Down-Regulation strategy
P_{ms_limit}	Down-regulated value of P_{ms} according to Converter PQ Range Limiter strategy
P_{sc_limit}	Down-regulated value of P_{sc} according to Converter PQ Range Limiter strategy
S_{ref_limit}	Down-regulated value of S_{ref} according to Converter PQ Range Limiter strategy
P_{ref_limit}	Down-regulated value of P_{ref} according to Converter PQ Range Limiter strategy
Q_{ref_limit}	Down-regulated value of Q_{ref} according to Converter PQ Range Limiter strategy
SOC_{sc}	State-of-charge of the self-consumption service
SOC_{ms}	State-of-charge of the market service

Appendix A. TBox of the EMS-Ontology

$$\begin{aligned}
TBox = & \{Optimize \circ relatedTo \sqsubseteq Optimize \\
& relatedTo \circ hasState^- \sqsubseteq OptimizeLA \\
& SendSetPoint \circ relatedTo \sqsubseteq SendSetPoint \\
& SendSetPoint \circ hasSetPoint^- \sqsubseteq SendSetPointToLA \\
& Control \circ SendSetPoint \circ SetPointSetState \sqsubseteq Control \\
& hasSetPoint^{-1} \circ hasState \circ hasLimit \sqsubseteq hasLimit \\
ControlVar \sqsubseteq & ControlVar \sqcap \forall SendSetPoint.SetPoint \sqcap \forall SendSetPointToLA.LogicalActor \\
& hasValue \text{ keyfor } (ControlVar \sqcup SetPoint) \\
& hasPriority \text{ keyfor } (HLUC \sqcup PUC) \\
& (hasMax \sqcup hasMin) \text{ keyfor } Limit \\
PUC \sqsubseteq & PUC \sqcap \forall Optimize.OptimizationVar \sqcap \forall hasSetPoint.SetPoint \\
HLUC \sqsubseteq & HLUC \sqcap \exists hasPUC.PUC \\
Application \sqsubseteq & Application \sqcap \exists hasHLUC.HLUC \\
SetPoint \sqsubseteq & SetPoint \sqcap \exists hasLimit.Limit \sqcap \exists SetPointSetState.State \\
State \sqsubseteq & State \sqcap \exists hasLimit.Limit \\
Limit \sqsubseteq & Limit \sqcap \forall relatedTo.State \\
OptimizationVar \sqsubseteq & OptimizationVar \sqcap \exists OptimizeLA.LogicalActor \\
LogicalActor \sqsubseteq & LogicalActor \sqcap \forall hasSetPoint.SetPoint \sqcap \exists hasState.State \\
& trans(relatedTo), Sym(relatedTo)\}
\end{aligned}$$

Appendix B. ABox of the SLVGC Application

$ABox = \{ Application(SLVGC), HLUC(MarketService), HLUC(SelfConsumption), HLUC(VoltageControl),$
 $PUC(MarketService), PUC(SelfConsumption), PUC(VoltageControl),$
 $OptimizationVar(V_{pcc}), ControlVar(P_{ms}), ControlVar(P_{sc}), ControlVar(Q_{gs}),$
 $SetPoint(P_{ref}), SetPoint(Q_{ref}), LogicalActor(ESS), LogicalActor(Meter),$
 $Limit(P_{max}), Limit(Q_{max}), hasLimit(P, P_{max}), hasLimit(Q, Q_{max}), State(P), State(Q), State(V_{pcc}),$
 $hasHLUC(CEMS, MarketService), hasHLUC(CEMS, SelfConsumption), hasHLUC(CEMS, VoltageControl)$
 $hasPUC(MarketService, MarketService), hasPUC(VoltageControl, VoltageControl),$
 $hasPUC(SelfConsumption, SelfConsumption),$
 $Control(MarketService, P_{ms}), Control(SelfConsumption, P_{sc}), Control(VoltageControl, Q_{gs}),$
 $SendSetPoint(P_{ms}, P_{ref}), SendSetPoint(P_{sc}, P_{ref}), SendSetPoint(Q_{gs}, Q_{ref}),$
 $relatedTo(V_{pcc}, P), relatedTo(V_{pcc}, Q), relatedTo(P_{max}, Q), relatedTo(Q_{max}, P),$
 $hasState(ESS, P), hasState(ESS, Q), hasState(Meter, P), hasState(Meter, Q),$
 $hasState(Meter, V_{pcc}), hasSetPoint(ESS, P_{ref}), hasSetPoint(ESS, Q_{ref}),$
 $SetPointSetState(P_{ref}, P), SetPointSetState(Q_{ref}, Q), Optimize(VoltageControl, V_{pcc})$
 $hasMax(P_{max}, 100), hasMax(Q_{max}, 100), hasMin(P_{min}, -100), hasMin(Q_{min}, -100),$
 $haspriority(MarketService, 2), haspriority(SelfConsumption, 3), haspriority(VoltageControl, 1),$
 $hasValue(P_{ms}, 60), hasValue(P_{sc}, 60), hasValue(Q_{gs}, 60), \}$

Appendix C. Querying the Ontology of the SLVGC Controller

Table A1. Implementation of Question 1.

Multi-Objective Optimization (Conflict C_I)	
DL Query	Query Result
(1) Query to know if there is an optimization variable that is optimized by at least two different use case: <i>OptimizationVar</i> and (<i>InvOptimize min 2 (PUC)</i>)	{}

Table A2. Implementation of Question 2.

Maximal Limit Dependency of Control Variables (Conflict C_{II})	
DL Query	Query Result
(1) Search limits of states that depends on other states: <i>Limit and relatedTo min 1 State</i>	{ <i>P_{max}, Q_{max}</i> }
(2) Search state that owns the limit <i>Q_{max}</i> : <i>State and hasLimit value Qlimit</i>	{ <i>Q</i> }
(3) Search control variables that control the state <i>Q</i> : <i>ControlVar and Control value Q</i>	{ <i>Q_{gs}</i> }
(4) Investigate the PUC that controls the state <i>Q</i> : <i>PUC and Control value Q</i>	{ <i>VoltageControl</i> }
(5) Search state that is related to the limit <i>Q_{max}</i> : <i>State and relatedTo value Q_{max}</i>	{ <i>P</i> }
(6) Search control variables that control the state <i>P</i> : <i>ControlVar and Control value P</i>	{ <i>P_{sc}, P_{ms}</i> }
(7) Search PUC that controls the state <i>P</i> : <i>PUC and Control value P</i>	{ <i>MarketService, SelfConsumption</i> }

Table A3. Implementation of Question 3 and 4.

Set-Point out of Limits (Conflict C_{III}) and Set-Point Sign Conflicts (Conflict C_{IV})	
SPARQL Query	Query Result
(1) Search a set-point that is set by at least two control variables: <i>SELECT DISTINCT ?setpoint ?ControlVar WHERE {?setpoint onto:InvSendSetPoint ?ControlVar.}</i>	$\{P_{ms}, P_{ref}$ P_{sc}, P_{ref} $Q_{gs}, Q_{ref}\}$
(2) Investigate the control variables that sets P_{ref} : <i>SELECT DISTINCT ?controlvar WHERE {?controlvar onto:SendSetPoint ?value.} FILTER (?value=Pset)}</i>	$\{P_{ms}, P_{sc}\}$
(3) Get the value of control variables that set the set-point P_{ref} : <i>SELECT DISTINCT ?controlvar ?controlvalue WHERE { ?controlvar onto:SendSetPoint ?setpoint ?controlvar onto:hasValue ?controlvalue. FILTER (?setpoint=st:Pset)}</i>	$\{P_{ms}, 60$ $P_{sc}, 60\}$
(4) Get the sum of values of control variables P_{sc} and P_{ms} : <i>SELECT (SUM(?controlvalue) as ?TotalSetpoint) WHERE {?controlvar onto:SendSetPoint ?setpoint.?controlvar onto:hasValue ?controlvalue. FILTER (?setpoint=st:Pset)} GROUP BY ?setpoint</i>	$\{120\}$
(5) Find the limitations of the set-point P_{ref} : <i>SELECT ?Max WHERE {?setpoint onto:hasLimit ?limit.?limit onto:hasMax ?Max. FILTER (?setpoint=st:Pset)}</i>	$\{100\}$

Table A4. Implementation of Question 5.

Set-Point Set by at Least Two Use Cases (Conflict C_V)	
DL Query	Query Result
(1) Search set-point that is set by at least two control variables: <i>SetPoint and InvSendSetPoint min 2 ControlVar</i>	$\{P_{ref}\}$
(2) Identify the control variables that set the SetPoint P_{ref} : <i>ControlVar and SendSetPoint value Pset</i>	$\{P_{ms}, P_{sc}\}$
(3) Search the PUC that controls P_{ms} and P_{sc} : <i>PUC and Control value Pms, PUC and Control value Psc</i>	$\{SelfConsumption, MarketService\}$

Table A5. Implementation of Question 6.

Interrelated Manipulated Variables (Conflict C_{VI})	
DL Query	Query Result
(1) Search if there are some variables to be regulated or optimized: <i>OptimizationVar and InvOptimize min 1 PUC</i>	$\{V_{pcc}\}$
(2) Search the PUC that manipulates V_{pcc} : <i>PUC and Optimize value Vpcc</i>	$\{VoltageControl\}$
(3) Search the state that is related to V_{pcc} : <i>State and relatedTo value Vpcc</i>	$\{P, Q\}$
(4) Search the use cases that controls P : <i>PUC and Control value P</i>	$\{SelfConsumption, MarketService\}$

Appendix D. Parameters of the SLVGC Use Case

Table A6. Use case parameters.

Parameter	Value/Setting	Description
R	0.07 Ohm	resistance of the low voltage line
L	0.255 mH	inductance of the low voltage line
V	230 V rms	low voltage grid
f	50 Hz	nominal frequency of the grid
P_{max}	100 kW	maximum active power of the ESS converter
Q_{max}	100 kVAr	maximum reactive power of the ESS converter
S_{max}	100 VA	maximum apparent power of the ESS converter
V_{bat}	660 V	nominal voltage of the battery
C_{bat}	600 Ah	capacity of the battery
$SoC_{bat-init}$	50%	initial SoC of the battery
C_{sc}	50%	capacity allocated to the self-consumption use case
C_{ms}	50%	capacity allocated to the market service use case

References

- Liserre, M.; Sauter, T.; Hung, J. Future Energy Systems: Integrating Renewable Energy Sources into the Smart Power Grid Through Industrial Electronics. *IEEE Ind. Electron. Mag.* **2010**, *4*, 18–37.
- Li, X.; Hui, D.; Lai, X. Battery Energy Storage Station (BESS)-Based Smoothing Control of Photovoltaic (PV) and Wind Power Generation Fluctuations. *IEEE Trans. Sustain. Energy* **2013**, *4*, 464–473.
- Hollinger, R.; Diazgranados, L.M.; Braam, F.; Erge, T.; Bopp, G.; Engel, B. Distributed solar battery systems providing primary control reserve. *IET Renew. Power Gener.* **2016**, *10*, 63–70.
- IEC. *Draft IEC 61850-90-7 TR Communication Networks and Systems for Power Utility Automation*; Technical Report; International Electrotechnical Commission (IEC): Geneva, Switzerland, 2011.
- Büdenbender, K.; Braun, M.; Stetz, T.; Strauss, P. Multifunctional PV Systems offering additional functionalities and improving grid integration. *Int. J. Distrib. Energy Resour. Smart Grids* **2011**, *7*, 109–128.
- Hänninen, S.; Kyritsis, A.; Abdulhadi, I.; Schwalbe, R.; Strasser, T.; Kosmecki, M.; Sobczak, B.; Rink, R.; Kedra, B.; Wilk, M.; et al. WP 6 Controllable Flexibility Detailed Requirements and Constraints for the Control of Flexibility; Technical Report; ELECTRA Internal Report R6.1; ELECTRA IRP, 2015. Available online: http://orbit.dtu.dk/files/125919193/20150116_R6.1_DetailedRequirementsandConstraintsfortheControlofFlexibility_V1.02.pdf (accessed on 28 August 2017).
- EERA. *D4.3 Integration of Storage Resources to Smart Grids: Possible Services, D4.4 Control Algorithms for Storage Applications in Smart Grid*; EERA Joint Programme on Smart Grids Sub-Programme 4 Electrical Energy Technologies; Technical Report; EERA: Brussels, Belgium, 2014.
- Von Appen, J.; Stetz, T.; Braun, M.; Schmiegel, A. Local Voltage Control Strategies for PV Storage Systems in Distribution Grids. *IEEE Trans. Smart Grid* **2014**, *5*, 1002–1009.
- CEN-CENELEC-ETSI Smart Grid Coordination Group. *Reference Architecture for the Smart Grid*; Technical Report; CEN-CENELEC-ETSI: Brussels, Belgium, 2012.
- IEC. *IEC 62559: Use Case Methodology*; Technical Report; International Electrotechnical Commission (IEC): Geneva, Switzerland, 2015.
- Zanabria, C.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. An approach for the handling of controller conflicts within multi-functional energy storage systems. In Proceedings of the 24th International Conference on Electricity Distribution (CIRED), Glasgow, UK, 12–15 June 2017.
- Bletterie, B.; Tayyebi, A.; Kadam, S.; Le Baut, J.; Stöckl, J.; Kathan, J.; Einfalt, A. A novel concept for combining distribution network and system support services for storage systems. In Proceedings of the 2017 IEEE Manchester PowerTech, Manchester, UK, 18–22 June 2017; pp. 1–6.
- Perera, B.; Ciufu, P.; Perera, S. Advanced point of common coupling voltage controllers for grid-connected solar photovoltaic (PV) systems. *Renew. Energy* **2016**, *86*, 1037–1044.
- Consentec GmbH. *Description of Load-Frequency Control Concept and Market for Control Reserves*; Study Commissioned by the German TSOs; Consentec GmbH: Aachen, Germany, 2014.

15. Braam, F.; Hollinger, R.; Engesser, M.L.; Müller, S.; Kohrs, R.; Wittwer, C. Peak shaving with photovoltaic-battery systems. In Proceedings of the Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Istanbul, Turkey, 12–15 October 2014; pp. 1–5.
16. Riffonneau, Y.; Bacha, S.; Barruel, F.; Ploix, S. Optimal Power Flow Management for Grid Connected PV Systems With Batteries. *IEEE Trans. Sustain. Energy* **2011**, *2*, 309–320.
17. Andrén, F.; Lehfuss, F.; Strasser, T. A development and validation environment for real-time controller-hardware-in-the-loop experiments in Smart Grids. *Int. J. Distrib. Energy Resour. Smart Grids* **2013**, *9*, 27–50.
18. Zanabria, C.; Andrén, F.P.; Kathan, J.; Strasser, T. Towards an integrated development of control applications for multi-functional energy storages. In Proceedings of the IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 6–9 September 2016; pp. 1–4.
19. Faschang, M.; Schwalbe, R.; Einfalt, A.; Mosshammer, R. Controller hardware in the loop approaches supporting rapid prototyping of smart low voltage grid control. In Proceedings of the 2014 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Istanbul, Turkey, 12–15 October 2014; pp. 1–5.
20. Hitzler, P.; Krötzsch, M.; Rudolph, S. *Foundations of Semantic Web Technologies*; Chapman & Hall/CRC: London, UK, 2009.
21. Schachinger, D.; Kastner, W.; Gaida, S. Ontology-based abstraction layer for smart grid interaction in building energy management systems. In Proceedings of the IEEE International Energy Conference (ENERGYCON), Leuven, Belgium, 4–8 April 2016; pp. 1–6.
22. Samirmi, F.D.; Tang, W.; Wu, Q. Fuzzy Ontology Reasoning for Power Transformer Fault Diagnosis. *Adv. Electr. Comput. Eng.* **2015**, *15*, 107–114.
23. Baader, F. *The Description Logic Handbook: Theory, Implementation and Applications*; Cambridge University Press: Cambridge, UK, 2003.
24. Brambilla, M.; Cabot, J.; Wimmer, M. Model-Driven Software Engineering in Practice. *Synth. Lect. Softw. Eng.* **2012**, *1*, 1–182.
25. Siegel, J. *Developing in OMG's New Model-Driven Architecture*; Object Management Group (OMG): Needham, MA, USA, 2001.
26. Dänekas, C.; Neureiter, C.; Rohjans, S.; Uslar, M.; Engel, D. Towards a model-driven-architecture process for smart grid projects. In *Digital Enterprise Design & Management*; Springer: Warsaw, Poland, 2014; pp. 47–58.
27. Andrén, F.; Strasser, T.; Kastner, W. Engineering Smart Grids: Applying Model-Driven Development from Use Case Design to Deployment. *Energies* **2017**, *10*, 374.
28. Working Group Sustainable Processes (SG-CG/SP). *CEN-CENELEC-ETSI Smart Grid Coordination Group—Sustainable Processes*; Technical Report; CEN-CENELEC-ETSI: Brussels, Belgium, 2012.
29. Kalibatiene, D.; Vasilecas, O. Survey on Ontology Languages. In *Perspectives in Business Informatics Research*; Grabis, J., Kirikova, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 90, pp. 124–141.
30. Tayyebi, A.; Bletterie, B.; Kupzog, F. Primary Control Reserve and Self-Sufficiency Provision with Central Battery Energy Storage System. In Proceedings of the NEIS Conference, Hamburg, Germany, 21–22 September 2017; in press.
31. Hua, Y.; Zander, S.; Bordignon, M.; Hein, B. From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning. In Proceedings of the 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 6–9 September 2016; pp. 1–8.
32. Apache Jena. Ontology API, 2012. Available online: <https://jena.apache.org/documentation/ontology/> (accessed on 12 March 2017).
33. World Wide Web Consortium. SPARQL 1.1 Query Language, 2008. Available online: <https://www.w3.org/TR/sparql11-query/> (accessed on 11 February 2017).
34. Tayyebi, A. Modelling and Simulation of a Multifunctional PV Electrochemical Storage System. Master's Thesis, University of Oviedo, Oviedo, Spain, 2016.
35. Yazdani, A.; Iravani, R. *Voltage-Sourced Converters in Power Systems: Modeling, Control, and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2010.

36. Bredillet, P.; Lambert, E.; Schultz, E. CIM, 61850, COSEM standards used in a model driven integration approach to build the smart grid service oriented architecture. In Proceedings of the First IEEE International Conference on Smart Grid Communications (SmartGridComm), Gaithersburg, MD, USA, 4–6 October 2010; pp. 467–471.
37. Heussen, K.; Gehrke, O.; Niemann, H. On early conflict identification by requirements modeling of energy system control structures. In Proceedings of the IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg City, Luxembourg, 8–11 September 2015; pp. 1–8.
38. Uslar, M.; Heussen, K. Towards modeling future energy infrastructures—The ELECTRA system engineering approach. In Proceedings of the PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Ljubljana, Slovenia, 9–12 October 2016; pp. 1–6.



© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

Bibliography

- [1] EERA Joint Programme on Smart Grids-Sub-Programme 4-Electrical Energy Technologies. Technical Report D4.3 Integration of Storage Resources to Smart Grids: Possible Services, D4.4 Control Algorithms for Storage Applications in Smart Grid., Brussels, Belgium, April 2014.
- [2] X. Liu, A.s Aichhorn, L. Liu, and H. Li. Coordinated control of distributed energy storage system with tap changer transformers for voltage rise mitigation under high photovoltaic penetration. *IEEE Transactions on Smart Grid*, 3(2):897–906, June 2012.
- [3] SMB Smart Grid Strategic Group (SG3). IEC Smart Grid Standardization Roadmap. Geneva, Switzerland, 2013.
- [4] C. Tornelli, L. Radaelli, E. Rikos, and M. Usler. WP 4 Fully Interoperable Systems. Deliverable R4.1: Description of the methodology for the detailed functional specification of the ELECTRA solutions. *IST project*, 2015.
- [5] C. Dänekas, C. Neureiter, S. Rohjans, M. Usler, and D. Engel. Towards a model-driven-architecture process for smart grid projects. In *Digital Enterprise Design & Management*, pages 47–58. Springer, 2014.
- [6] F. Andrén, T. Strasser, and W. Kastner. Engineering Smart Grids: Applying Model-Driven Development from Use Case Design to Deployment. *Energies*, 10(3):374, March 2017.
- [7] F. Andrén, S. Henein, and M. Stifter. Development and validation of a coordinated voltage controller using real-time simulation. In *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 3713–3718, Melbourne, Australia, November 2011.
- [8] T. Strasser, F. Prörtl Andrén, G. Lauss, R. Bründlinger, H. Brunner, C. Moyo, and others. Towards holistic power distribution system validation and testing—an overview and discussion of different possibilities. *e & i Elektrotechnik und Informationstechnik*, 134(1):71–77, February 2017.

-
- [9] A. Kossiakoff, W. N. Sweet, S. J. Seymour, and S. M. Biemer. *Systems engineering principles and practice*, volume 83. John Wiley & Sons, 2011.
- [10] H. H. Abdeltawab and Y. A. I Mohamed. Market-oriented energy management of a hybrid wind-battery energy storage system via model predictive control with constraint optimizer. *IEEE Transactions on Industrial Electronics*, 62(11):6658–6670, November 2015.
- [11] S. Rohjans, S. Lehnhoff, S. Schütte, F. Andrén, and T. Strasser. Requirements for smart grid simulation tools. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 1730–1736, Istanbul, Turkey, June 2014.
- [12] C. Zanabria, F. Prössl Andrén, and T. I. Strasser. Comparing specification and design approaches for power systems applications. In *2018 IEEE PES Transmission Distribution Conference and Exhibition - Latin America (T&D-LA)*, pages 1–5, Lima, Peru, September 2018.
- [13] International Electrotechnical Commission (IEC). IEC 62559-2 Use Case Methodology-Part2: Definition of the templates for use cases, actor list and requirement list. Geneva, Switzerland, 2015.
- [14] T. Weillkiens. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. Elsevier, August 2011.
- [15] R. Lewis. *Modelling Control Systems Using IEC 61499: Applying Function Blocks to Distributed Systems*. IET, 2001.
- [16] P. Hitzler, M. Krotzsch, and S. Rudolph. *Foundations Of Semantic Web Technologies*. CRC press, 2009.
- [17] R. Santodomingo, S. Rohjans, M. Uslar, J.A. Rodríguez-Mondéjar, and M.A. Sanz-Bobi. Ontology matching system for future energy smart grids. *Engineering Applications of Artificial Intelligence*, 32:242–257, June 2014.
- [18] V. Dubinin, V. Vyatkin, C. Yang, and C. Pang. Automatic generation of automation applications based on ontology transformations. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4, Barcelona, Spain, 2014. IEEE.
- [19] M. Brambilla, J. Cabot, and M. Wimmer. Model-Driven Software Engineering in Practice. *Synthesis Lectures on Software Engineering*, 1(1):1–182, September 2012.

-
- [20] F. Andr en, T. Strasser, and W. Kastner. Model-Driven Engineering applied to Smart Grid Automation using IEC 61850 and IEC 61499. In *Power Systems Computation Conference*, pages 1–7, Wroclaw, Poland, 2014.
- [21] D. Schachinger, W. Kastner, and S. Gaida. Ontology-based abstraction layer for smart grid interaction in building energy management systems. In *2016 IEEE International Energy Conference (ENERGYCON)*, pages 1–6, April 2016.
- [22] F. D. Samirmi, W. Tang, and Q. Wu. Fuzzy Ontology Reasoning for Power Transformer Fault Diagnosis. *Advances in Electrical and Computer Engineering*, 15(4):107–114, 2015.
- [23] B. K. Perera, P. Ciufu, and S. Perera. Point of common coupling (PCC) voltage control of a grid-connected solar photovoltaic (PV) system. In *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*, pages 7475–7480, Vienna, Austria, 2013. IEEE.
- [24] J. von Appen, T. Stetz, M. Braun, and A. Schmiegel. Local Voltage Control Strategies for PV Storage Systems in Distribution Grids. *IEEE Transactions on Smart Grid*, 5(2):1002–1009, March 2014.
- [25] Y. Riffonneau, S. Bacha, F. Barruel, and S. Ploix. Optimal Power Flow Management for Grid Connected PV Systems With Batteries. *IEEE Transactions on Sustainable Energy*, 2(3):309–320, July 2011.
- [26] A. Tayyebi, B. Bletterie, and F. Kupzog. Primary Control Reserve and Self-Sufficiency Provision with Central Battery Energy Storage Systems. In *Proceedings of the NEIS Conference*, pages 21–22, Hamburg, Germany, 2017.
- [27] International Electrotechnical Commission. IEC/TR 61850-90-7—Communication Networks and Systems for Power Utility Automation—Part 90-7: Object Models for Power Converters in Distributed Energy Resources (DER) Systems. 2013.
- [28] F. Baader. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge university press, 2003.
- [29] Working Group Sustainable Processes (SG-CG/SP). CEN-CENELEC-ETSI Smart Grid Coordination Group – Sustainable Processes. Technical report, CEN-CENELEC-ETSI, Brussels, Belgium, November 2012.
- [30] I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. *Kr*, 6:57–67, 2006.

-
- [31] H. Happel and S. Seedorf. Applications of Ontologies in Software Engineering. *Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC*, pages 5–9, 2006.
- [32] D. M. Powers. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Bioinfo Publications*, 2(1):pp-37–63, 2011.

Curriculum Vitae

Claudia Zanabria

Single, French, 32 years old – Währinger gürtel 130/18 1090 Vienna, Austria

+43 66066 88993 • claudiazanabria@gmail.com

Research engineer with experience in the conception of innovative ICT solutions that guarantee the flexibility and interoperability of electricity distribution networks. Strong technical competences in modeling, designing, implementing and validating power system networks. Solid knowledge of smart grid standards and control strategies for the power system domain. Significant experience in field work.

Professional experience

- PhD Candidate** **Vienna**
 - Austrian Institute of Technology-Center for Energy* *October 2014–December 2018*

Realization of methodologies for a rapid and massive implementation of energy management systems focused on energy storage systems integration. Implementation of control applications under SGAM, IEC 61850 and CIM standards. Use of model-driven-engineering techniques to automate the control application development process.
- Automation and Control Engineer for Intelligent Building Systems** **Île de France**
 - Assystem E&OS-BU Automation* *September 2011– September 2014*

Realization of a full ICT solution for the control and supervision of Building Management Systems. Conception of communication and component architectures. Design and implementation of control strategies. Programming of controllers under the standard IEC 61131-3. Conception and realization of SCADA and databases. Planning of the communication network to host intelligent electronic devices. Modeling and simulation of the process under study. Controller-Hardware-in-the-loop validation. Supervision of fieldwork to achieve a full deployment of ICT solutions.
- Control Engineer for the Automation of the A14x86 Motorway in France** **Île de France**
 - Actemium-Paris (Groupe Vinci Energie)* *February 2011–July 2011*

Design and implementation of SCADA systems and PLC programs (Siemens S7-400) for a full control and supervision of the A14x86 motorway. Conduct of FAT and SAT tests. Field work for a massive integration of controllers.
- Junior Research for the Modeling of Electric Distribution Networks** **Grenoble**
 - GIE-IDEA: Group conformed by EDF, Grenoble INP and Schneider Electric* *January 2010–August 2010*

Modeling of failures in electricity distribution networks under the IEC 61850 power automation approach.

Education

- Vienna University of Technology** **Vienna**
 - PhD at Institute of Mechanics and Mechatronics , in process* *2014–2018*
- ENSE3-Grenoble INP** **Grenoble**
 - Diplôme d'Ingénieur in Automatic Control, Systems and Information Technology* *2009–2011*
- National University of Engineering** **Lima**
 - Bachelor of Science in Electronics Engineering* *2003–2008*

Technical and Personal skills

- Programming Languages:** Proficient in: Matlab/Simulink, Java, C#, C++, Assembly, VBA, SQL.
- Industry Software Skills:** PcVue (SCADA), LabVIEW, Step 7 (PLC), CodeSys, MPLAB (microcontrollers).
- General Business Skills:** Good presentation skills. Can write structured reports and journal papers.
- Languages:** Advanced level in Spanish, French, English and Intermediate level in German.

List of Scientific Publications

Peer-reviewed International Journal Papers

C. Zanabria, F. Prörtl Andrén, and T. I. Strasser.

An Adaptable Engineering Support Framework for Multi-Functional Energy Storage System Applications.

Sustainability, vol. 10, no. 11, 28 pages, 2018.

C. Zanabria, F. Prörtl Andrén, J. Kathan, and T. I. Strasser.

Rapid Prototyping of Multi-Functional Battery Energy Storage System Applications.

Applied Sciences, vol. 8, no. 8, 32 pages, 2018.

C. Zanabria, A. Tayyebi, F. Prörtl Andrén, J. Kathan, and T. Strasser.

Engineering Support for Handling Controller Conflicts in Energy Storage Systems Applications.

Energies, vol. 10, no. 10, 24 pages, 2017.

C. Zanabria, F. Prörtl Andrén, J. Kathan, and T. Strasser.

Approach for Handling Controller Conflicts within Multi-Functional Energy Storage Systems.

CIREN - Open Access Proceedings Journal, vol. 2017, no. 1, pp. 1575-1578, 2017.

Peer-reviewed International Conference Papers

C. Zanabria, F. Prörtl Andrén, and T. I. Strasser.

Comparing Specification and Design Approaches for Power Systems Applications.

2018 IEEE PES Transmission & Distribution Conference and Exhibition-Latin America (T&D-LA), September 18-21, Lima, Peru, 2018.

C. Zanabria, F. Prörtl Andrén, J. Kathan, and T. Strasser.

An Approach for the Handling of Controller Conflicts within Multi-Functional Energy Storage Systems.

24th International Conference on Electricity Distribution (CIREN), June 12-15, Glasgow, UK, 2017.

C. Zanabria, F. Pröbstl Andrén, J. Kathan, and T. Strasser.

Towards an Integrated Development of Control Applications for Multi-Functional Energy Storages.

21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '2016), September 6-9, Berlin, Germany, 2016.

C. Zanabria, F. Andrén, F. Leimgruber, R. Bründlinger, and T. Strasser.

A Low Cost Open Source-based IEC 61850/61499 Automation Platform for Distributed Energy Resources.

IEEE PowerTech Eindhoven 2015, June 29 - July 2, Eindhoven, The Netherlands, 2015.

Poster Presentations at International Conferences

C. Zanabria, T. Strasser, F. Andrén, J. Kathan, and S. Jakubek.

Towards a Framework for the Development of Multi-Functional ESS Control Applications.

7th Advanced Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS 2016, April 11-13, Caparica - Lisbon, Portugal, 2016.

C. Zanabria, T. Strasser, F. Andrén, and S. Lehnhoff.

Rapid Prototyping of Control Applications for Multi-Functional Storage Systems.

4th D-A-CH Conference Energieinformatik 2015, November 12-13, Karlsruhe, Germany, 2015.