sustainability

MDPI

Article

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

Claudia Zanabria [1,*], Filip Pröstl 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 CO_2 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/red 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 highlight. 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.

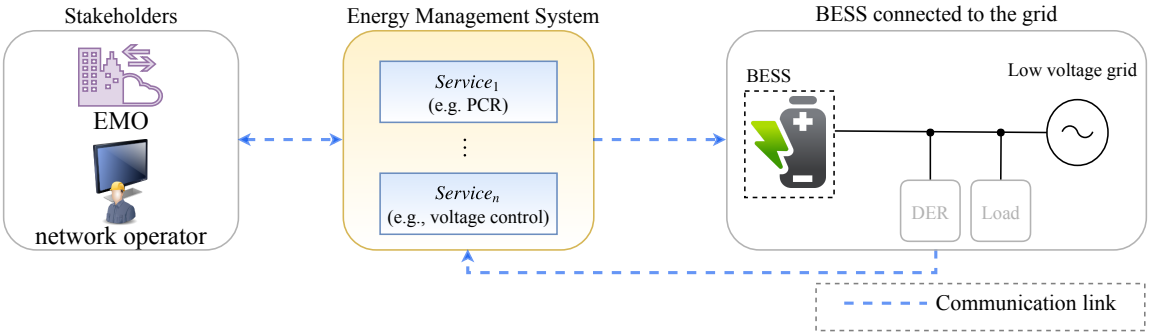


Figure 1. Framework of the BESS control applications.

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.

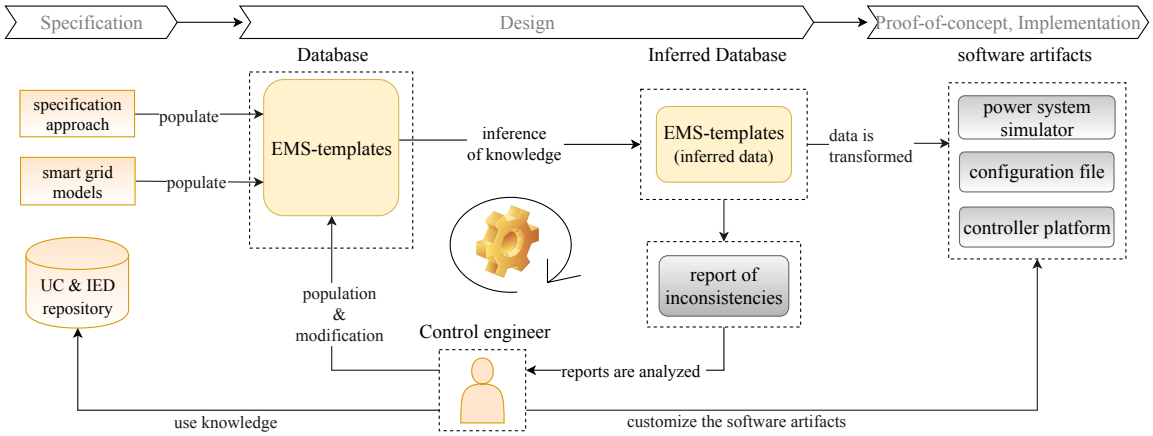


Figure 2. EMSOnto in practice by control engineers.

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.

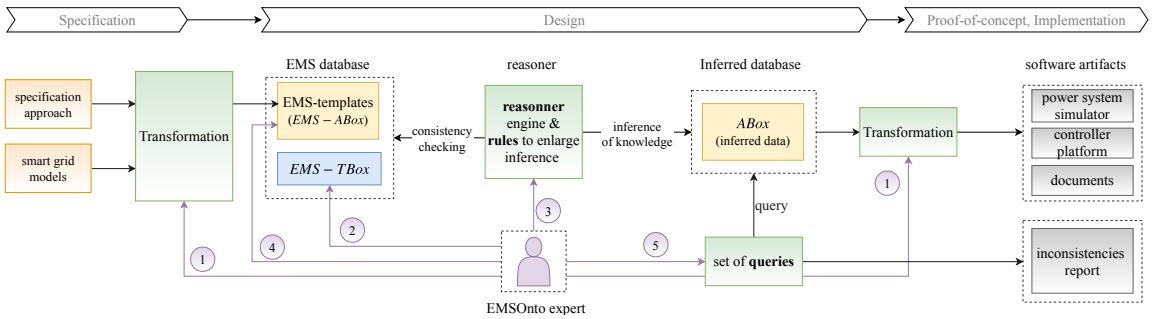


Figure 3. Tasks to be performed by EMSOnto expert.

(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 `Pmax` 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 Pröstl 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.

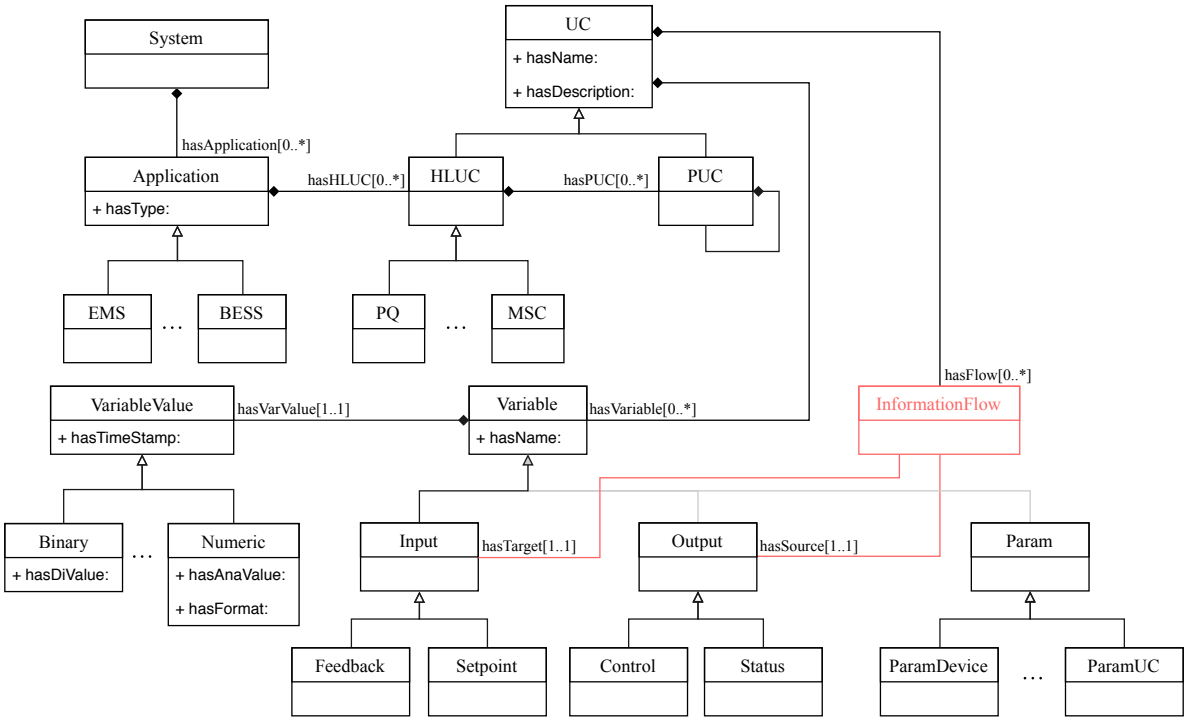


Figure 4. UML class diagram representing the EMS-$TBox$ (reduced model).

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

DL ($SROIQ(D)$)	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.

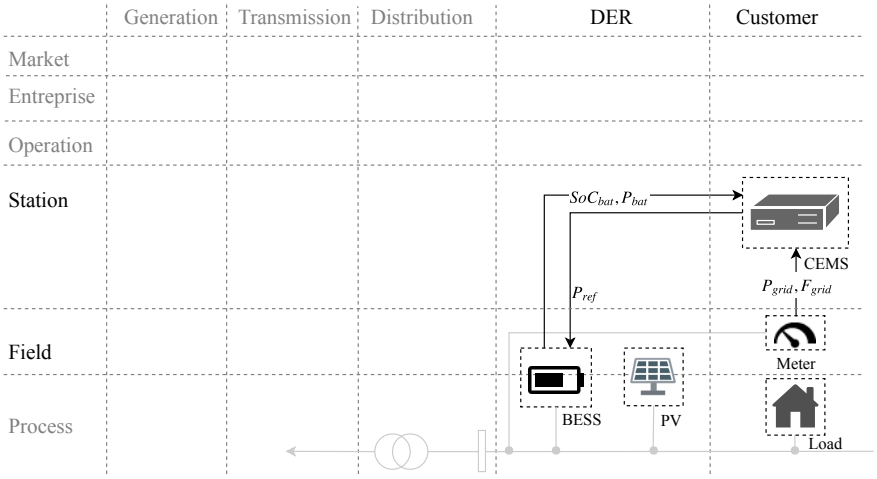


Figure 5. SGAM information layer representation of the CEMS (UC example) [8].

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 sough 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.

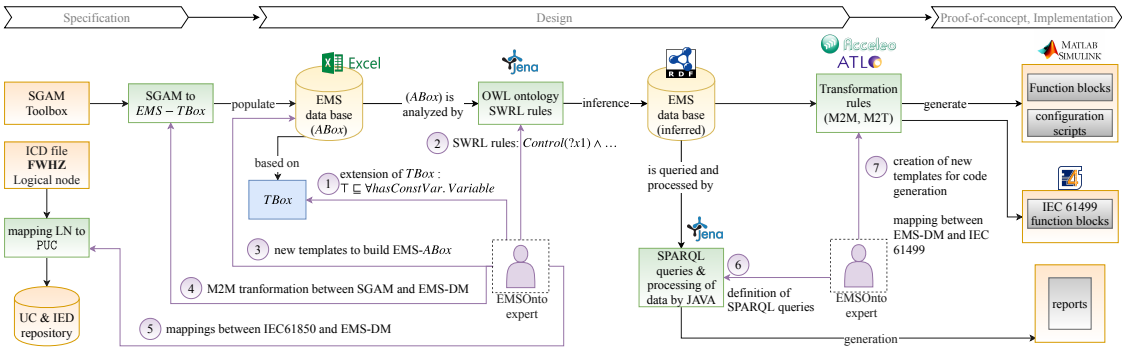


Figure 6. Design and Implementation of CEMS UC— automated by EMSOnto expert.

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 `LA`s and `PUC`s 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
$UC \sqsubseteq \exists hasFlow.InformationFlow \sqcap \ldots$, $InformationFlow \sqsubseteq \leq 1 hasSource.Output \sqcap \leq 1 hasTarget.Input \sqcap \exists hasSource.Output \sqcap \exists hasTarget.Input$	`InformationFlow` represents information exchanged between `UC`s. Hence, the role `hasFlow` relates the concepts `UC` and `InformationFlow`. `InformationFlow` 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 $\boldsymbol{x}$ 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 ($\boldsymbol{x}$), 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` $\{x_1, \ldots, x_n\}$, `Constraint` $\{u_{\max}, \ldots\}$. 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`.

$$\boldsymbol{x}' = A\boldsymbol{x} + Bu, \quad y = C\boldsymbol{x} + Du, \quad \boldsymbol{x} = [x_1 \quad \ldots \quad x_n]^T \tag{1}$$

$$\boldsymbol{x}_{\min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\max}, \quad u_{\min} \leq u \leq u_{\max}, \quad y_{\min} \leq y \leq y_{\max} \tag{2}$$

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 `Constraint` owns `Variables`, this relation is represented by `hasConstVar`. The inverse role of `hasConstVar` is given by `hasVarConst`. The role `IsConsLinkCons` relates `Constraints` and the transitivity property (*trans*) is assigned to it.
$Variable \sqsubseteq \exists hasVarConst.Constraint \sqcup \ldots$, $hasVarConst \circ IsConsLinkCons \sqsubseteq hasVarConst$	A `Variable` belongs to a `Constraint` is represented by `hasVarConst`. A `Variable` can inherit `Constraints` from other `Variables`, 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`$\{Vnom, SoC\}$, `ParamDevice`$\{SoCini, CAh\}$ and `ParamUC`(CAh_UC). Nevertheless, a further representation of those variables is required, resulting in new concepts. For instance, the concept `Vnom` 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_O` facilitates a remote control of internal variables defined within the scope of a `UC` as detailed in Table 4.

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

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

Concepts, Roles, OWL Axioms	Description
$Vnom \sqcup SoC \sqcup P \sqcup I \sqcup \cdots \sqsubseteq State$	`Vnom` 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}).
$SoCini \sqcup CAh \sqcup \cdots \sqsubseteq ParamDevice$	`SoCini` represents an initial SoC (e.g., SoC_{ini}), `CAh` models the full capacity of an energy storage device (e.g., Q_{bat}) .
$CAh\_UC \sqcup \cdots \sqsubseteq ParamUC$	Capacity assigned to a UC is represented by `CAh_UC` (e.g., Q_{hluc}).
$Application \sqsubseteq \exists IsConnectedTo.Application \ldots$, $HLUC \sqsubseteq \exists ControlBESS.BESS \ldots$	The role `IsConnectedTo` relates two applications and the role `ControlBESS` relates a `HLUC` that is connected to a `BESS`.
$hasI\_O \quad keyfor \quad Internal$	`hasI_O` 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`, ...).

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): $Atom \wedge Atom \wedge \cdots \rightarrow Atom$. 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**$(?x1, ?c1)$r	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 `Costraints` 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 *r*3 to identify when a `HLUC` is controlling a `BESS` by means of `ControlBESS`. As a sequel, *r*4 attaches a new `PUC` (*SoC_estimator*) to the identified `HLUC`s. Finally, *r*5 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 `ControlBESS`.
r4: *PREFIX CEMS :< http://.../CEMS# > INSERT{* ?*x*1 **CEMS:hasPUC** **?x3**. ?*x*3 *rdf* : *type CEMS* : *PUC*. ?*x*3 **CEMS:hasType** **"SoC_estimator"**$^{\wedge\wedge}$*xsd* : *string*. ?*x*3 *CEMS* : *hasName* ?*x*3*N* } *WHERE{* ?*x*1 *rdf* : *type CEMS* : *HLUC*. ?*x*1 *CEMS* : *ControlBESS* ?*x*2. ?*x*1 *CEMS* : *hasName* ?*x*1*N*. *BIND*(*URI*(*CONCAT*("*SoC_*", *STR*(?*x*1))) *as* ?*x*3). *BIND*(*CONCAT*(*STR*(?*x*1*N*), "*_SoC*") *as* ?*x*3*N*)}	A `PUC` of type '*SoC_estimator*' 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 '*_SoC*'. For instance, a `PUC` called *FW_SoC* is assigned to a `HLUC`(*FW*) named *FW*.
r5: *PREFIX CEMS :< http://.../CEMS# > INSERT{* ?*x*5 **CEMS:IsAssignedTo** ?*x*4. } *WHERE{* ?*x*1 *CEMS* : *ControlBESS* ?*x*2. ?*x*1 *CEMS* : *hasPUC* ?*x*3. ?*x*3 *CEMS* : *hasType* "*SoC_estimator*"$^{\wedge\wedge}$*xsd* : *string*. ?*x*3 **CEMS:hasFeedback** **?x4**. ?*x*4 *CEMS* : *hasType* "*Vnom*"$^{\wedge\wedge}$*xsd* : *string*. ?*x*2 **CEMS:hasStatus** **?x5**. ?*x*5 *CEMS* : *hasType* "*Vnom*"$^{\wedge\wedge}$*xsd* : *string* }	A BESS's `Status` is assigned to a `Feedback` of the function `PUC`(*SoC_estimator*). For simplicity, only the inference of relations between BESS's V_{nom} and `PUC`(*SoC_estimator*) are shown. Thus, since V_{nom} is needed to calculate SoC_{hluc}, a role `IsAssignedTo` representing the relation of `Status`(V_{nom}) and `PUC`(*SoC_estimator*)'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

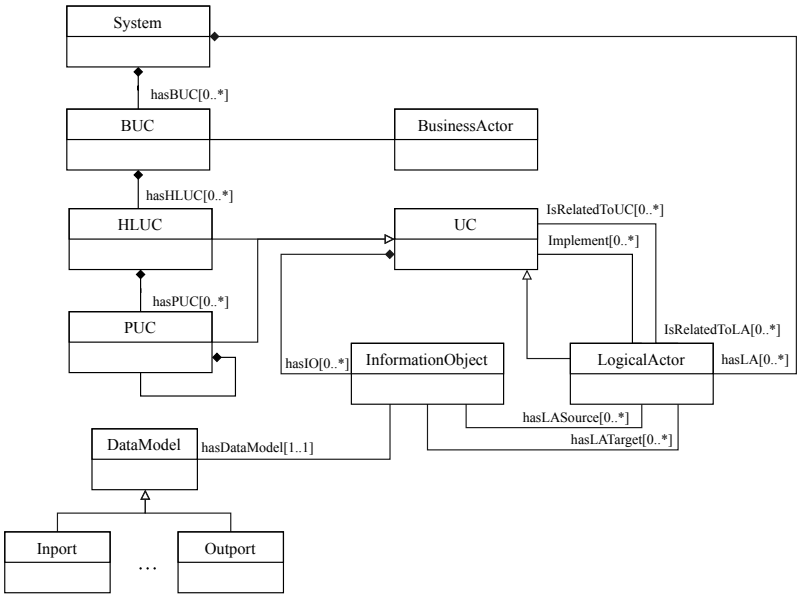
Table 9. Parameters and states relevant to UC(*SoC_estimator*).

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.



(**a**)

SGAM-TB	EMS-DM
System	System
HLUC	HLUC
PUC	PUC
LogicalActor	Application
InformationObject	InformationFlow
Inport	Input
Outport	Output

(**b**)

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 an 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 `LNodeType`. Each of them contains `DO`s of type `DOType` which in turns contain `DA`s of type `DAType`. A `LNodeType` should correspond to a `LN` as well as a `DOType` to a `CDC`.

The instantiation of the mentioned classes is exemplified by the `LNodeType`:$MMXU$, see Figure 8. This notation (`class:individual`) is employed to assign an individual to a class. The mentioned `LNodeType` 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 `LNodeType` 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.

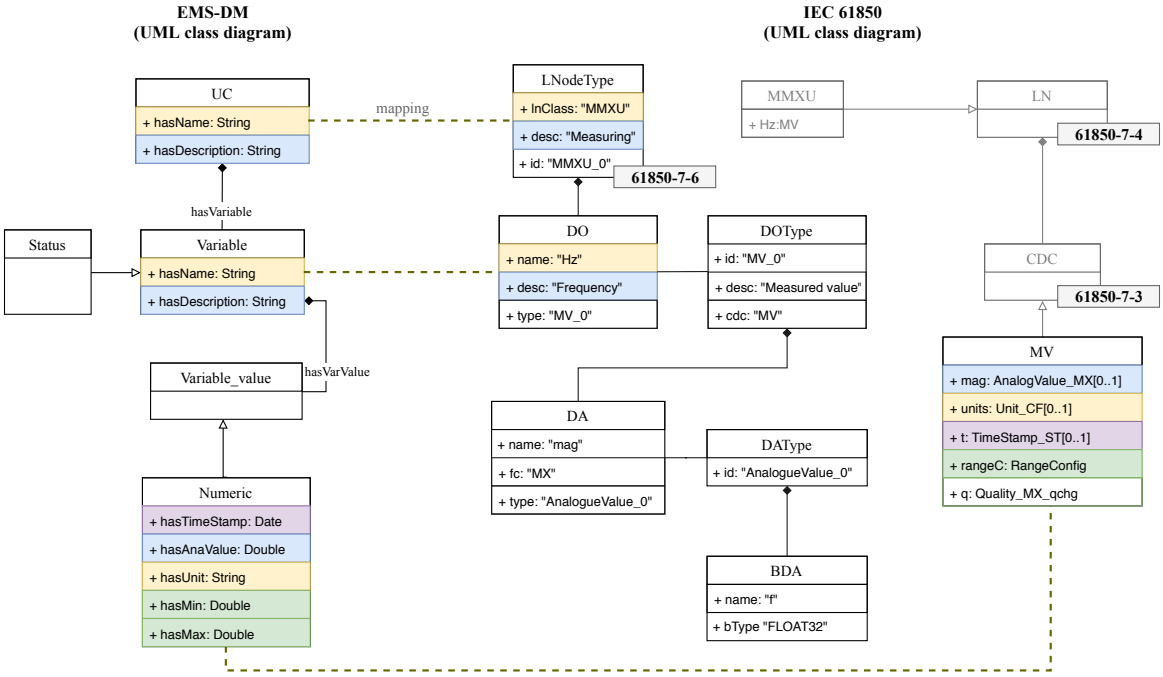


Figure 8. Mapping between IEC 61850 (LN) and EMS-DM (`UC`).

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 $Pmax$. 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 $Pmax$ defined within a `BESS`$(battery)$?	$Pmax \sqcap \exists hasParam^{-}.BESS\{battery\}$
Q_{II}	What is the active power to be required by a service `HLUC` $(Service_n)$?	$BessSize \sqcap hasParam^{-}.HLUC\{Service_n\}$

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?	$SELECT \quad ?control \quad ?setpoint \quad ?unit\_ct \quad ?unit\_sp$ $WHERE\{$ $?HLUC \quad CEMS:hasSetpoint \quad ?setpoint;$ $rdf:type \quad CEMS:HLUC.$ $?setpoint \quad CEMS:hasUnit \quad ?unit\_sp.$ $?control \quad CEMS:IsAssignedTo \quad ?setpoint;$ $rdf:type \quad CEMS:Control;$ $CEMS:hasUnit \quad ?unit\_ct$ $FILTER(!regex(STR(?unit\_ct), \quad STR(?unit\_sp))\}$

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 `HLUC`s (i.e., functions) included within an `EMS`. Afterwards, the parameters (`Param`) assigned to the identified `HLUC`s 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 `HLUC`s.

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 `Outport` and `InPort`. Those `ports` may be related to a `SingleConnection`. This configuration is quite similar to EMS-DM structure. Hence, the concepts: $\texttt{Application}^{EMS}$ and $\texttt{UC}^{EMS}$ are transformed into $\texttt{SubSystem}^{Sim}$, $\texttt{Input}^{EMS}$ into $\texttt{InPort}^{Sim}$, $\texttt{InformationFlow}^{EMS}$ into $\texttt{SingleConnection}^{Sim}$ and so on, as shown in Table 12. To increase the understanding of classes transformation the syntax $\texttt{class}^{Model}$ 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 $\texttt{UC}^{EMS}$ is mapped to a $\texttt{FB}^{61499}$, which is defined by a $\texttt{FBType}^{61499}$. The `FBType` can take the form of one single block (`BasicFB`) or an arrangement of blocks (`FBNetwork`), see Figure 10. On the other hand, $\texttt{Applications}^{EMS}$ is converted into $\texttt{Application}^{61499}$, $\texttt{Inputs}^{EMS}$ into $\texttt{InputVars}^{61499}$ 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 $\texttt{UC}^{EMS}$ into $\texttt{FBType}^{61499}$ 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`).

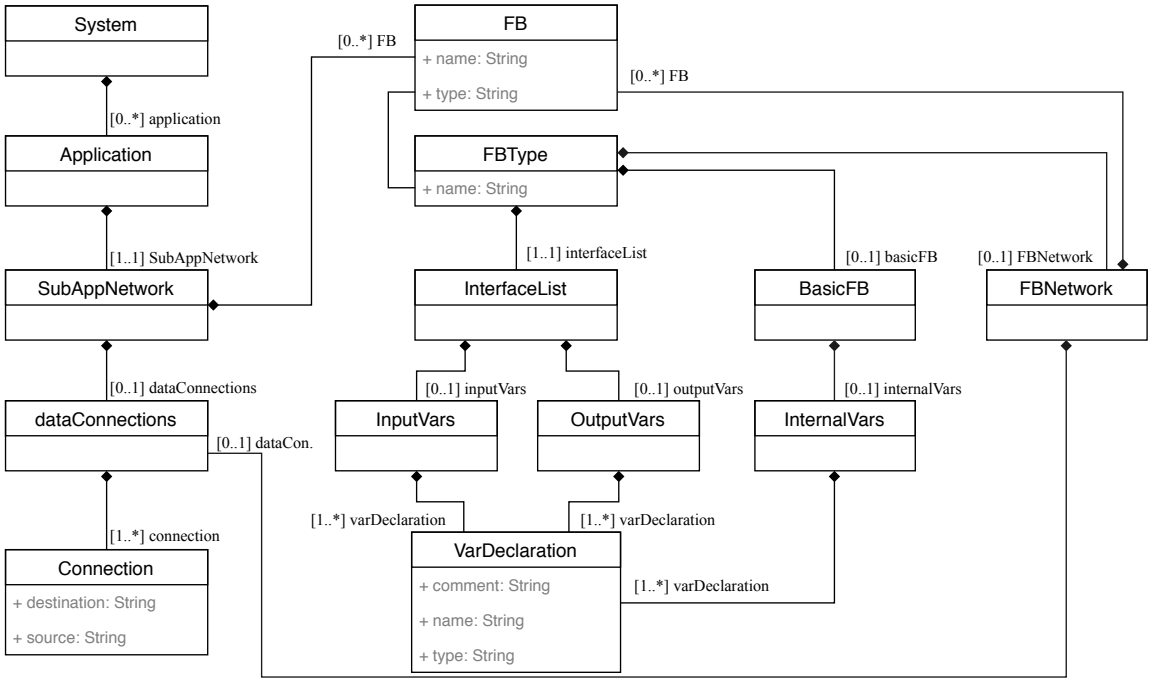


Figure 10. UML class diagram representation of IEC 61499 (PSM).

```
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 $\texttt{UC}^{EMS}$ into $\texttt{FBType}^{61499}$ (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	Outport	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 *Linar-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.

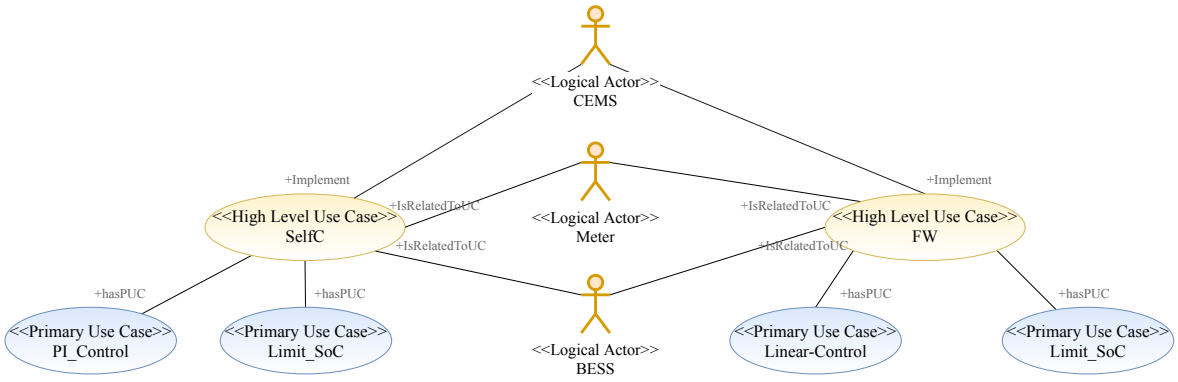


Figure 12. CEMS's structure represented by a UML use case diagram.

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`($Linear\text{-}Control$), it is worth setting the `PUC`'s type to $FWHZ$. As a result, all the variables related to LN:$FWHZ$ are imported within $Linear\text{-}Control$, 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.

```
<LNodeType 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 ...
</LNodeType>
   <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` ($Linear\text{-}Control$) 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 *C*1 and *C*2. 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.

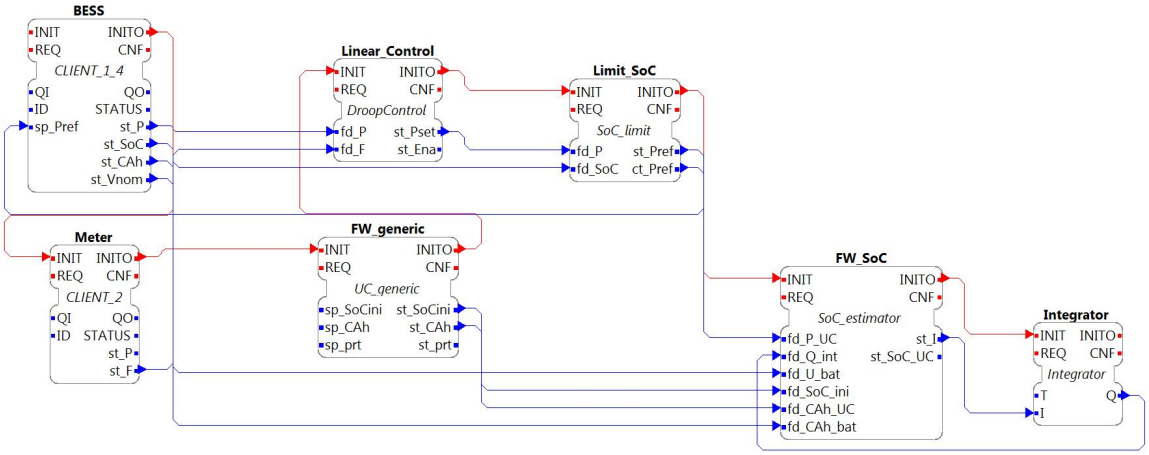


Figure 15. IEC 61499 model issued from EMS-templates (only *FW* is represented).

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.; Pröstl 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 GroupSustainableProcesses*; 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.; Pröstl 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.; Pröstl 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 Systens. 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. Weilkiens, 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.; Pröstl 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.; Grosof, 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]