A Model-Driven and Ontology-based Engineering Approach for Smart Grid Automation Applications

Claudia Zanabria*, Filip Pröstl Andrén†, Thomas I. Strasser†‡, Wolfgang Kastner‡

*Réseau de Transport d'Electricité (RTE), Paris, France, claudia.zanabria@rte-france.com
†AIT Austrian Institute of Technology, Vienna, Austria {filip.proestl-andren, thomas.strasser}@ait.ac.at
‡Vienna University of Technology, Vienna, Austria {thomas.strasser, wolfgang.kastner}@tuwien.ac.at

***Abstract*—Due to the increasing power system complexity caused by a higher penetration of distributed energy resources, advanced engineering methods are required which can handle power system aspects together with information and communication technologies and new automation concepts. In order to handle such a complexity, various engineering concepts have been introduced by providing support such as guidelines, code generation, and reasoning for the development of smart grid use cases and applications in the last few years. Nevertheless, none of these approaches covers all phases in the engineering of smart grid solutions and the corresponding requirements. In this paper, two approaches—focusing on rapid development and on the identification of application inconsistencies—are selected and combined to cover a wider scope of the requirements for smart grid automation application engineering. The integration of the two approaches is shown using an example use case, where multiple functions for a battery energy management system are designed, implemented, and validated.**

I. Introduction

The ongoing large-scale integration of Photovoltaic (PV), wind, and other Distributed Energy Resources (DER) into the power system—especially at distribution level—is one of the main reasons for implementing new and advanced control solutions to guarantee supply of electricity [1]–[3]. Additionally, to accommodate many new DER units also new energy market approaches are being developed and implemented [4]. While new intelligent methods for operating the power system emerge, the complexity of the corresponding engineering process is increasing as well (e.g., due to new business models and services, the handling of interoperability between various devices). The realization of complex solutions is associated with increasing engineering complexity, which usually results in higher development costs. This motivates the investment in research and development of innovative engineering and validation approaches [5], [6].

In order to tackle these issues, several new and advanced smart grid engineering approaches and concepts have been introduced during the last years [7]–[12]. However, when comparing them along the design, development, and validation phases it turns out that there is still a lack in the fulfillment of important engineering requirements [13]. Of course, due to the different characteristics of various tasks within each phase, it is difficult to cover all requirements in only one approach. However, the comparison also shows that gaps could be probably closed or narrowed down when existing methods are combined with each other [13]. Until now, this has never been investigated.

This paper examines the potential of such a solution by combining two of the most recently published engineering approaches: the Power System Automation Language (PSAL) [11] and the Energy Management System Ontology (EMSOnto) [14]. Compared to other smart grid engineering approaches, PSAL and EMSOnto were both explicitly developed to support rapid prototyping of smart grid applications. Rapid prototyping is important for the development of smart grid applications since it provides methods implementing and testing new solutions in a rapid and secure manner, without jeopardizing the existing infrastructure [15]. In the paper, an integration of PSAL and EMSOnto is presented. Whereas PSAL focuses more on providing code generation support, EMSOnto's ontology-based approach can provide the engineer with inferred information, such as reports on controller conflicts. Thus, the two approaches complement each other and a combination can provide a user with even better engineering support. The integrated approach is illustrated by developing an example use case starting with a design, followed by a simulative validation, and completed with code generation for hardware devices and controllers.

The remaining part of the paper is organized as follows: Section II presents related and previous work, which this paper is based upon. In Section III, the example use case is presented which is later used to show the potential of the combined engineering solution. In the following Section IV, the combination of PSAL and EMSOnto is discussed. Section V presents a simulative proof-of-concept validation and shows how code can be generated for real controllers. Finally, the paper is concluded with Section VI.

II. Related Work

A. Engineering Methods for Smart Grids

Through an increasing digitalisation, today's power systems are in a transit towards cyber-physical energy systems (also referred to as "smart grids") [9], [16]–[18]. As a consequence, classical engineering methods are replaced by advanced engineering approaches where automation and Information and Communication Technologies (ICT) concepts are increasingly used. Due to this, common software and systems engineering and modeling approaches—such as the Unified Modeling

Language (UML) or Systems Modeling Language (SysML)—are becoming also more common in the domain of power and energy systems [13].

Apart from general-purpose software engineering methods, several domain specific engineering approaches for smart grids have also been developed during the last years. For example, the IntelliGrid method [19], originally developed in 2003, was intended as a response to the increasing complexity of power system automation. Since then, IntelliGrid has also been published as IEC standard IEC 62559. Today, it is one of the most commonly used methods for describing smart grid use cases [8]. Related to IntelliGrid is also the Smart Grid Architecture Model (SGAM), which provides a visually structured approach for modeling advanced power system/smart grid use cases [10]. SGAM proposes different interoperability layers (business, function, information, communication, and component) that can be used to model different parts of a corresponding use case. With the "SGAM Toolbox" [10], a tool-support for the SGAM approach was also introduced.

Several current smart grid engineering approaches were compared by Zanabria et al. [13]. A brief overview of this comparison is shown in Table I. This study mainly focused on the engineering phases covering specification, design, and implementation. One conclusion that can be drawn from the comparison is that approaches that support the specification phase do not fully support the design phase and vice versa. For example, both the SGAM Toolbox and SysML show good support in the specification phase but only limited support for design and implementation. Furthermore, not all approaches provide automated support for the implementation phase [13].

TABLE I: Comparison of engineering approaches for the specification, design, and implementation (adapted from [13]).

Phase	**Specification**			**Design**			**Impl.**
Approach	**bus./ fun.**	**inf.**	**comm./ comp.**	**fun.**	**inf.**	**comm./ comp.**	**rapid prot.**
UML	😐	✓	👎	😐	×	×	×
SysML	✓	✓	✓	✓	👎	×	×
IntelliGrid	😐	✓	×	👎	×	×	×
SGAM TB	✓	✓	✓	😐	×	👎	👎
EMSOnto	😐	✓	×	✓	✓	×	😐
PSAL	👎	✓	✓	✓	✓	✓	✓
MATLAB	×	×	😐	✓	👎	👎	✓

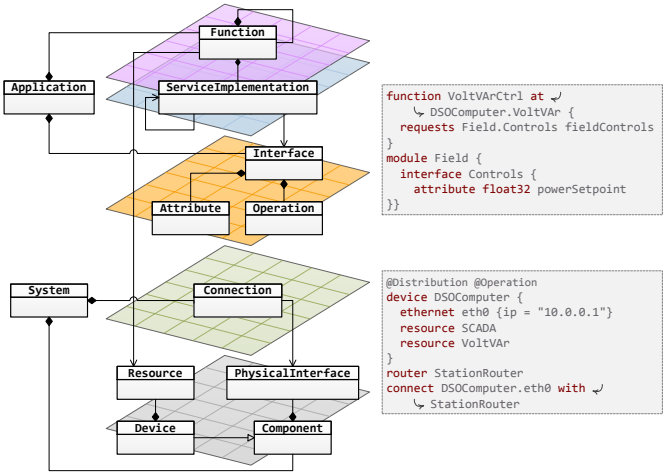
Legend: not supported (×), minimal support (👎), basic support (😐), and fully supported (✓)

As seen in Table I, a good overall support is provided by PSAL, which can be complemented by EMSOnto especially for the specification of business and functional logic. Additionally to the features compared in the study, EMSOnto also offers an automatic detection of application inconsistencies—such as controller conflicts. This is unique compared to the other approaches and allows engineers to receive additional information about their developed use cases. If this can be combined with the good support from PSAL for the design and the implementation phases, the resulting approach will provide users with effective tools for rapid prototyping of smart grid applications. In order to achieve this, the two approaches need to be aligned and integrated with each other without loosing any of their own capabilities. In the following two sections, both PSAL and EMSOnto and their features are briefly summarized.

B. Overview of PSAL and EMSOnto Approaches

1) Power System Automation Language (PSAL) [11]: It has been designed to be a formalized Domain Specific Language (DSL) for SGAM compatible use case design and specification [6]. This was combined with a main focus towards rapid development of automation, control, and ICT functions for smart grid automation applications [20]. Due to this, PSAL both supports modeling of high-level use case descriptions as well as more detailed design intended for implementation.

PSAL assists design of SGAM's layers using a textual language. In Fig. 1, a UML representation of PSAL's metamodel is shown together with two example implementations. PSAL was explicitly designed to promote rapid generation of code and configurations [11].



(a) (b)

Fig. 1: PSAL overview: (a) UML representation, (b) code [13].

The original version of PSAL mainly focused on how to describe functions and the communication between them. Also, the generation of communication configurations such as IEC 61850 or Modbus is supported.

2) Energy Management System Ontology (EMSOnto) [14]: This approach (see Fig. 2) is intended to support control engineers during the conception, prototype, and implementation of Energy Management System (EMS) control applications with a focus on multi-functional storage systems [21]. It focuses on the function and information layers of the SGAM model. A main outcome of EMSOnto is an automatic detection of inconsistencies at the conception level [14]. To achieve this, an ontology that models an EMS and its environment is proposed. This includes the structure of the EMS and the information exchanged within the EMS. Information used to

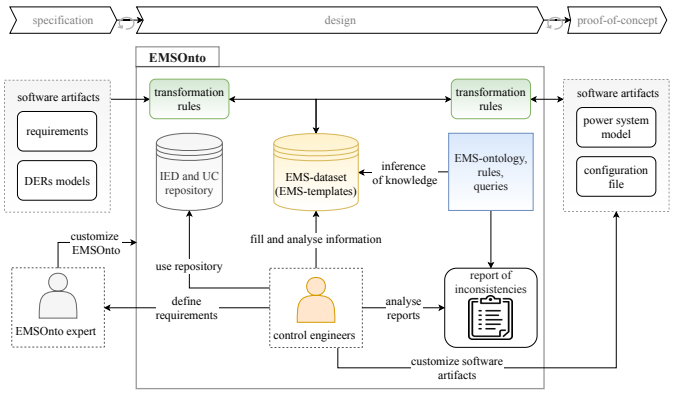


Fig. 2: Overview of the EMSOnto approach.

populate the ontology is gathered by spreadsheet templates [22]. Using logical rules, the deduction of new information and inconsistencies is possible. This ensures an error-free design before an implementation of the proposed solution can be carried out. The proof-of-concept and implementation stages are assisted by an automatic code and text generation which will be deployed within Integrated Development Environment (IDE) for controllers (e.g., IEC 61131-3 and IEC 61499), power system simulators, and other potential platforms.

III. Example Use Case

A Use Case (UC) example is introduced to show the benefits of using EMSOnto and PSAL together. In the UC, a Battery Energy Storage System (BESS) needs to be designed and implemented. It supports two services, a Market Service (MS) where the main actor—an Energy Market Operator (EMO)—requires to charge or discharge the battery according to spot market prices. This is done to maximize monetary benefits. The second service is a Self-Consumption (SC), which assists a group of households connected to DER units, enabling them to minimize the consumption from the electric power utility. Any excess of generated power needs to be stored within the BESS. In Fig. 3, an overview of the UC is seen.

Both services (i.e., MS and SC) are deployed within an EMS that commands the active power fed and retrieved by the BESS by using the command P_{ref}. The setpoints P_{sc} and P_{ms} are power values required by the households and the EMO, respectively. Besides this, the EMS measures the State-of-Charge (SoC) of the battery.

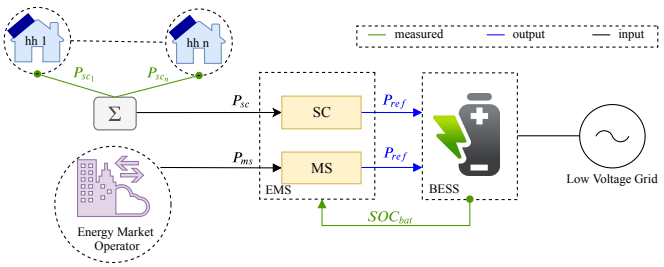


Fig. 3: Use case example representing an EMS with SC and MS services.

IV. Integrated Approach with PSAL and EMSOnto

Although PSAL and EMSOnto target different use case types, they still have some overlaps in the engineering process [13]. Both methods offer code generation support, but for different target systems. In order to benefit from the advantages of both, this paper proposes an integrated approach. Here, a design approach is presented where the two methods, PSAL and EMSOnto, exist in parallel. This will allow users to design their use cases in either PSAL, EMSOnto, or a combination of both. To make that possible, a model mapping between PSAL and EMSOnto is used by a transformation engine, which allows automatic transformation between descriptions in the two methods. An illustration of the proposed approach—applied on the example use case—is depicted in Fig. 4.

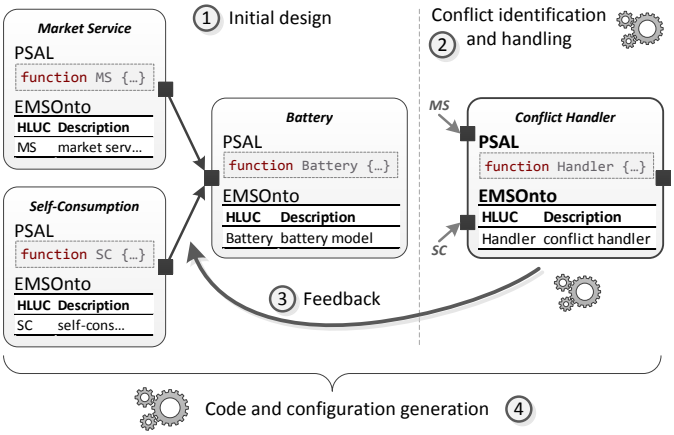


Fig. 4: Integrated engineering approach with PSAL and EMSOnto.

The engineer starts with an initial design of the use case (i.e., Step 1 in Fig. 4). Two services—a *Market Service* and a *Self-Consumption*—are connected to a *Battery*. It is up to the developer if PSAL, EMSOnto, or a mix of the two approaches is used. Once the initial design is finished, Step 2 uses the conflict identification capabilities of EMSOnto to identify any possible controller conflicts in the use case. However, before this step can be executed, the initial model is harmonized into a common model. Step 2 also generates handling solutions based on the identified type of conflict. In the example use case, both the *Market Service* and the *Self-Consumption* service want to change the active power reference of the *Battery*. This could lead to possible conflicts. In order to solve them, a *Conflict Handler* service is generated and inserted into the initial model (i.e., Step 3 in Fig. 4). For the example use case, this results in the new *Conflict Handler* which is inserted between the components involved in the conflict.

Step 1 until Step 3 can be iterated until the user is satisfied with the resulting design. Once that is the case, any of the capabilities of both PSAL and EMSOnto can be used to generate code (i.e., Step 4 in Fig. 4), such as models for Simulink or the Common Information Model (CIM), IEC 61499 code, and IEC 61850 configurations.

Details about the different steps in Fig. 4 are now discussed in more detail in the following sections.

A. Mapping between EMSOnto and PSAL

In order to be able to design a system in both PSAL and EMSOnto, the two approaches must be unified (i.e., mapped) with each other. This is done by a transformation engine which implements the mapping between PSAL and EMSOnto, illustrated by modelling the example use case using both approaches. Matching concepts from the two modeling approaches are discussed and highlighted. The following design steps are indicated as Step 1 in Fig. 4.

First, the UC example is designed and described by using EMSOnto templates, see Table II. Hence, `System`(Sys) contains `Application`{EMS, BESS}, and the EMS is composed of the High Level Use Cases (HLUC) MS and SC. Besides this, the `Application`{BESS} contains a model that simulates the behaviour of the battery (`HLUC`{Battery}). The services MS and SC control the active power of the battery through the command `Control`(Pref) which targets the setpoint of the `HLUC`(Battery), represented by `Setpoint`(Pref). Furthermore, the SoC of the battery is measured by the services MS and SC. Thus, the `Status`(SoC) is provided by the battery as shown in Table III.

TABLE II: High level structure of the UC example.

System	Appl.	HLUC	Description	Variable
Sys	EMS	MS	market service	Pref
Sys	EMS	SC	self consumption	Pref
Sys	BESS	Battery	battery model	Pref

TABLE III: A non-exhaustive list of variables of the HLUCs: MS, SC and Battery.

HLUC	Var.	Description	Type	hasMax	IsAssignedBy
MS	Pref	active power required by MS	Control	80	
SC	Pref	active power required by SC	Control	80	
Batt.	Pref	active power	Setpoint	100	MS.Pref, SC.Pref
Batt.	SoC	state-of-charge	Status	50	

The same use case is modeled using PSAL in Listing 1. Here, the `function`s represent the `HLUC`s in Table II, one for the market service, one for the self-consumption, and one `function` for modeling the battery model.

The variables defined in Table III are modeled with PSAL using the `module Variables`. Here, two information `interface`s with `attribute`s are used to model the variables `Pref` and `SoC`. In order to model the limits of the variables, a constant `attribute` is used with a post-fix (see `Pref_hasMax`).

In Listing 1, the information exchange between the `function`s is also modeled. The `Battery provides` a service based on the `interface BatteryPCtrl`. The same `interface` is also requested by the two other `function`s using the

```
/* market service */
function MS {
  requests Variables.IPref Pref
}
/* self consumption */
function SC {
  requests Variables.IPref Pref
}
/* battery model */
function Battery {
  provides Variables.IPref Pref
}
connect MS.Pref with Battery.Pref
connect SC.Pref with Battery.Pref

module Variables {
  interface IPref {
      /* active power */
    attribute float32 Pref
    const float32 Pref_hasMax = 100
  }
  interface ISOC {
      /* state of charge */
      readonly attribute uint16 SoC
      const uint16 SoC_hasMax = 100
  }
}
```

Listing 1: PSAL code for the example use case.

`requests` expressions. Finally, with the two `connect` expressions the requested services are connected with the provided service from the `Battery`.

Unlike EMSOnto, PSAL does not differentiate between different types of variables (i.e., `Control`, `Setpoint`, `Status`). However, it is possible to define an `attribute` as `readonly`, which implies that its value cannot be overwritten. This can be used to infer the type in EMSOnto. For example, a `readonly attribute` that is provided is mapped to a `Status` variable type in EMSOnto. An overview of the resulting mapping between PSAL and EMSOnto is shown in Table IV. In the integrated approach, this mapping is used by a transformation algorithm to allow automatic transformation between descriptions in PSAL and EMSOnto and vice versa.

TABLE IV: Mapping between EMSOnto and PSAL.

EMSOnto	PSAL
`HLUC`(HLUCName)	`function HLUCName`
`Variable`(VName)	`interface IVName { attribute ... VName }`
`Setpoint`(VName), `Status`(VName)	`provides IVName VName`
`Control`VName	`requests IVName VName`

B. Reasoning in EMSOnto

After the initial design, the next step is to use the reasoning capabilities of EMSOnto to find any possible inconsistencies and conflicts in the model (i.e., Step 2 in Fig. 4). Also, this step utilizes the mapping described in Section IV-A.

EMSOnto proposes an ontology represented by the description language $SROIQ(D)$ [23] to describe EMS control applications. This ontology is automatically populated by

collecting the information provided by the control engineers in the EMSOnto templates during the design phase. By using inference techniques on the ontology, it is possible to find inconsistencies in the planned design.

The inconsistencies that can be detected are classified in six groups ($C_I \ldots C_{VI}$). In the example UC, two types of conflicts are identified: *(i) set-point set by at least two UCs (C_V)*, and *(ii) set-point out of limits (C_{III})*. This is due to different control variables, originating from MS and SC services (`Control`(Pref)), which intend to command the active power injected by the battery. Furthermore, the accumulation of the maximum value of those commands (160 kW) exceeds the maximum allowed power to be discharged or charged by the battery (100 kW).

C. Feedback to PSAL/EMSOnto from Reasoner

EMSOnto proposes handling mechanisms to face the abnormal operation of the EMS, where conflicts were identified. Hence, a `HLUC`(Handling) is generated, This function gathers conflicted information from the `HLUC`(MS, SC) in order to calculate the value to be set within the `Setpoint`(Pref) of the `HLUC`(Battery). This calculation is based on two parameters that hold the priority of each `HLUC`s in conflict. The `HLUC` with the highest priority is preferred among the others. The rule for setting the priorities could come from regulatory frameworks (e.g., energy market regulator, grid codes) in such a case those rules will be used during the design phase. The demand from others `HLUC` can be also treated depending on the capacity of the battery. The resulted handling function is handed over to the control engineers in the form of spreadsheet templates as seen in Table V. This is shown as Step 3 in Fig. 4.

TABLE V: Structure of `HLUC`(Handling), a function generated by EMSOnto to face conflicts within the EMS.

HLUC	Var.	Description	Type	IsAsignedBy
`Handl.`	Pref_ms	active power required by MS	Setpoint	MS.Pref
`Handl.`	Pref_sc	active power required by SC	Setpoint	SC.Pref
`Handl.`	Prt_ms	priority of MS	Setpoint	
`Handl.`	Prt_sc	priority of SC	Setpoint	
`Handl.`	Pref	active power set into the battery	Control	

Based on the `HLUC`(Handling) shown in Table V, the same handling solution is generated for PSAL by the transformation engine (see Listing 2). The transformation follows the same mapping pattern as presented in Section IV-A. Furthermore, the priority variables as depicted in Table V are mapped to `const`s with assigned priority. The PSAL code presented in Listing 2 is added to the code shown in Listing 1.

The steps up until now (depicted in Fig. 4) can be repeated any times until the engineer is satisfied with the design and all conflicts are handled. There can, for example, be cases where the user wants to overrule a detected conflict. This might happen, when the engineer possesses more domain knowledge than is modeled within the use case. Thus, it is possible for the engineer to draw conclusions about the conflicts that cannot be known by the reasoner.

```
1  /* conflict handler */
2  function Handling {
3    /* active power required by MS */
4    provides HandlingInterfaces.HPref Pref_ms
5    /* active power required by SC */
6    provides HandlingInterfaces.HPref Pref_sc
7    /* active power set into the battery */
8    requests Variables.IPref Pref
9  }
10 connect MS.Pref with Handling.Pref_ms
11 connect SC.Pref with Handling.Pref_sc
12 connect Handling.Pref with Battery.Pref
13
14 /* module for conflic handling */
15 module HandlingInterfaces {
16   interface HPref {
17     attribute float32 Pref
18     const float32 Pref_hasMax = 100
19     /* priority of MS */
20     const uint8 Prt_ms = 1
21     /* priority of SC */
22     const uint8 Prt_sc = 2
23   }
24 }
```

Listing 2: PSAL code for the example use case.

V. Proof-of-Concept and Code Generation

Once a model is ready—including the initial design together with added conflict handlers—code and configurations can be generated based on it. For this step, the code generation capabilities of both PSAL and EMSOnto can be used. Based on the example use case from Fig. 3, the following sections show how the code is generated by both approaches.

A. Proof-of-Concept with Simulation Models

Overall, smart grid applications can be validated using a diversity of tools such as co-simulation frameworks, communication network emulators, power system simulators, or IDE controller emulators. [17]. From one perspective, the selection of those tools depends on the SGAM layer that is addressed, for instance using communication emulators (e.g., ns-3) for tests of the communication layer.

EMSOnto mainly addresses the proof-of-concept of the function and information layers. Hence, support is offered to generate code and models from the information collected by the spreadsheet templates. For simulation, artifacts are generated for MATLAB/Simulink [14]. Fig. 5 shows the generated Simulink simulation model of the used example use case. A solution to address the conflicts identified in the example (C_V and C_{III}) is proposed as a MATLAB script, see Listing 3.

Manual work is still needed from control engineers to achieve a full model that is ready to be simulated. This work is related to the behavior of the services: `HLUC`(MS, SC, Battery). Hence, control engineers may use the Simulink libraries to complete the EMS model. The power required by the services `HLUC`(SC, MS) is cut down to respect the technical limitations of the battery. The reduction of power

```
% Handling solution
function Pref = Handling(Pref_ms, Prt_ms, Pref_sc, Prt_sc)
    P_max = 1000; % Maximum allowed power for the battery
    if abs(Pref_ms + Pref_sc) > P_max
        if Prt_ms > Prt_sc
            Pref_sc = sign(Pref_sc) * (P_max -
                abs(Pref_ms));
        else
            Pref_ms = sign(Pref_ms) * (P_max -
                abs(Pref_sc));
        end
    end
    Pref = Pref_ms + Pref_sc;
```

Listing 3: MATLAB script of HLUC(Handling)

is done according to the level of priority set by each service. Since the HLUC(SC) has a higher priority compared to HLUC(MS), the power required by the service MS is cut down from Pms to Pms_{cut}. The setting of priorities is defined by control engineers considering regulatory frameworks for energy storage systems when they are operated in multi-use scenarios. The simulation results in Fig. 6 demonstrate that the code generated regarding the HLUC(Handling) resolves the conflicts between the services MS and SC.

B. Implementation with IEC 61499 and IEC 61850

Once a proof-of-concept validation has been made, both PSAL and EMSOnto provide code generation capabilities that can be used to generate executable IEC 61499 applications [11], [13]. Based on the example use case modeled in the sections above, Fig. 7 shows the resulting IEC 61499 application. For each `function` listed in Listing 1 and Listing 2, an IEC 61499 Function Block (FB) is generated. Based on the

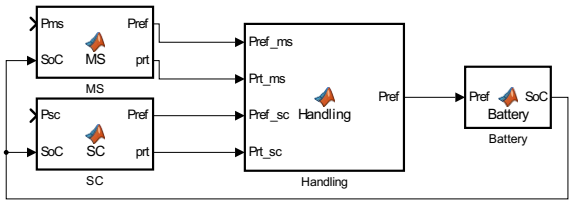


Fig. 5: Simulink model of the EMS (HLUC(MS, SC, Handling)) connected to a BESS, resulted with EMSOnto.

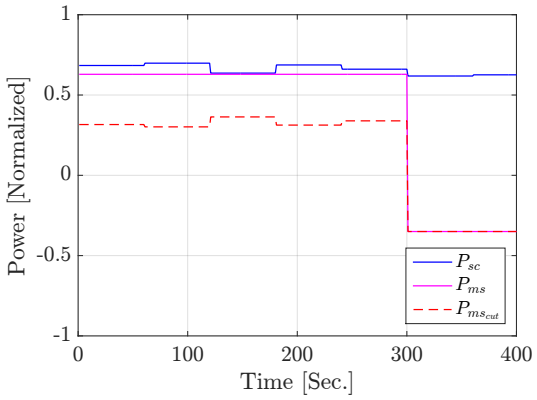


Fig. 6: Power required and finally set by the HLUC(SC)(P_{sc}) and HLUC(MS) (P_{ms}).

PSAL code, interfaces for the FBs and connection in between are also generated [11].

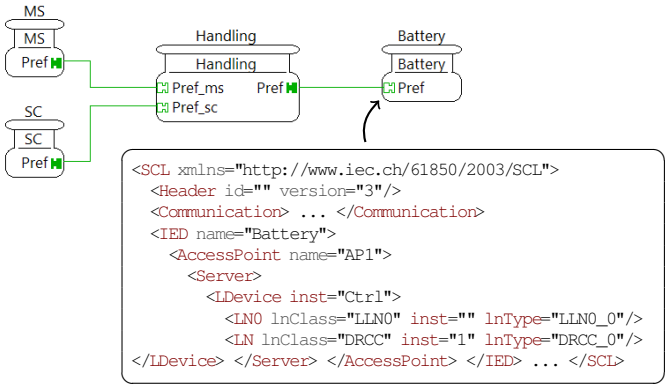


Fig. 7: Generated IEC 61499 application with IEC 61850 SCL configuration.

One of the main features of PSAL is that it can generate communication configurations based on the `interface`s defined in the PSAL code [11], [20]. If for example IEC 61850 communication should be used for the communication with the *Battery*, it can be done by changing the definition of the *IPref* to the following: `interface IPref : IEC61850.DRCC`. PSAL comes with predefined definitions of the most common IEC 61850 Logical Nodes (LN), including the LN *DRCC*[1]. These definitions can be used to specify for which connections IEC 61850 should be used as communication protocol. If that is the case, PSAL generates IEC 61850 configuration files based on the Substation Configuration Language (SCL). An excerpt of an SCL-file used to configure the communication to the battery is also shown in Fig. 7.

VI. Conclusions

Smart grid automation and control solutions tend to increase in complexity—also from and engineering point-of-view. This has triggered the introduction of partly new and advanced design and engineering approaches over the last years. However, none of them is able to support an automation and control engineer during all phases of the engineering process. Therefore, this work analyzed the potential of combining several approaches to reduce human design errors and at the same time provide rapid prototyping of smart grid applications. In a corresponding study the combination of PSAL and EMSOnto has been introduced and discussed.

An advantage of combining PSAL and EMSOnto is the availability of profound and formal methods to describe the different phases accordingly. This can be done by using spreadsheet templates or a textual language. Furthermore, the two approaches complement each other, which allows engineers to cover more aspects when designing smart grid applications, compared to the alone usage of EMSOnto or PSAL. An important advantage that EMSOnto brings to PSAL is the detection of inconsistencies at the conception level. This

[1]DRCC: DER Supervisory Control [24]

is also supported by the suggestion of solutions to address the conflicts among services deployed within the control application under design. Those advantages are demonstrated by a practical use case example where not only the conception was supported but also the proof-of-concept and the implementation phases.

This paper presents one example where the potential of combining PSAL and EMSOnto is clearly shown. However, in order to use this combined approach on a broader scale in different smart grid projects, future work will need to focus on the refinement of an integrated solution, the development of improved engineering tools as well as the validation of the combined approach in a wider range of selected examples to offer better quantitative comparisons. One domain that could benefit from the proposed solution is power system protection for electrical substations since simultaneous operation of different protection functions must be assured [25]. Furthermore, better integration with existing and already widely used approaches, such as IntelliGrid or SGAM is also targeted.

ACKNOWLEDGMENT

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

REFERENCES

[1] EERA Joint Programme on Smart Grids, "Sub-Programme 4: Electrical Energy Technologies," 2014.

[2] R. Hollinger, L. M. Diazgranados, F. Braam, T. Erge *et al.*, "Distributed solar battery systems providing primary control reserve," *IET Renewable Power Generation*, vol. 10, no. 1, pp. 63–70, 2016.

[3] M. Liserre, T. Sauter, and J. Y. Hung, "Future energy systems: Integrating renewable energy sources into the smart power grid through industrial electronics," *IEEE Industrial Electronics Magazine*, vol. 4, pp. 18–37, 2010.

[4] M. Kazemi, H. Zareipour, N. Amjady, W. D. Rosehart, and M. Ehsan, "Operation scheduling of battery storage systems in joint energy and ancillary services markets," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 4, pp. 1726–1735, 2017.

[5] T. Strasser, F. P. Andrén, G. Lauss *et al.*, "Towards holistic power distribution system validation and testingan overview and discussion of different possibilities," *e & i Elektrotechnik und Informationstechnik*, vol. 134, no. 1, pp. 71–77, 2017.

[6] M. Uslar, S. Rohjans, C. Neureiter *et al.*, "Applying the smart grid architecture model for designing and validating system-of-systems in the power and energy domain: A European perspective," *Energies*, vol. 12, no. 2, 2019.

[7] M. A. Al Faruque and F. Ahourai, "A Model-Based Design of Cyber-Physical Energy Systems," in *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014, pp. 97–104.

[8] M. Gottschalk, M. Uslar, and C. Delfs, *The Use Case and Smart Grid Architecture Model Approach: The IEC 62559-2 Use Case Template and the SGAM applied in various domains*. Springer Verlag Heidelberg, 2017.

[9] T. H. Morris, A. K. Srivastava, B. Reaves *et al.*, "Engineering Future Cyber-Physical Energy Systems: Challenges, Research Needs, and Roadmap," in *North American Power Symposium (NAPS)*, 2009.

[10] C. Neureiter, "A domain-specific, model driven engineering approach for systems engineering in the smart grid," Ph.D. dissertation, Carl von Ossietzky Universität Oldenburg, Computer Science Department, 2017.

[11] F. Pröstl Andrén, T. I. Strasser, and W. Kastner, "Engineering smart grids: Applying model-driven development from use case design to deployment," *Energies*, vol. 10, no. 3, 2017.

[12] C. Zanabria, "Adaptable engineering support framework for multi-functional battery energy storage systems," Ph.D. dissertation, Vienna University of Technology, Institute of Mechanics and Mechatronics, 2018.

[13] C. Zanabria, F. Pröstl Andrén, and T. I. Strasser, "Comparing specification and design approaches for power systems applications," in *2018 IEEE PES Transmission Distribution Conference and Exhibition - Latin America (T D-LA)*, 2018, pp. 1–5.

[14] C. Zanabria, A. Tayyebi, F. Pröstl Andrén, J. Kathan, and T. Strasser, "Engineering support for handling controller conflicts in energy storage systems applications," *Energies*, vol. 10, no. 10, p. 1595, 2017.

[15] A. Monti and F. Ponci, "Power grids of the future: Why smart means complex," in *2010 Complexity in Engineering*. IEEE, 2010, pp. 7–11.

[16] V. Gungor, D. Sahin, T. Kocak *et al.*, "Smart Grid Technologies: Communication Technologies and Standards," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[17] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 3, pp. 318–326, 2013.

[18] X. Yu and Y. Xue, "Smart grids: A cyberphysical systems perspective," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1058–1070, May 2016.

[19] J. Hughes, "Intelligrid architecture concepts and iec61850," in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, May 2006, pp. 401–404.

[20] F. Pröstl Andrén, T. Strasser, and W. Kastner, "Applying the SGAM Methodology for Rapid Prototyping of Smart Grid Applications," in *42nd Annual Conf. of the IEEE Ind. Electronics Society (IECON)*, 2016.

[21] B. Bletterie, A. Tayyebi, S. Kadam *et al.*, "A novel concept for combining distribution network and system support services for storage systems," in *2017 IEEE Manchester PowerTech*. IEEE, 2017, pp. 1–6.

[22] C. Zanabria, F. Andrén, J. Kathan, and T. Strasser, "Rapid prototyping of multi-functional battery energy storage system applications," *Applied Sciences*, vol. 8, no. 8, p. 1326, 2018.

[23] S. Gaggl, S. Rudolph, and L. Schweizer, "Fixed-domain reasoning for description logics," in *Proceedings of the Twenty-second European Conference on Artificial Intelligence*. IOS Press, 2016, pp. 819–827.

[24] *IEC 61850-7-420: Communication networks and systems for power utility automation - Part 7-420: Basic communication structure - Distributed energy resources logical nodes*. Geneva, Switzerland: International Electrotechnical Commission (IEC), 2009.

[25] A. Alvarez de Sotomayor, D. Della Giustina, G. Massa *et al.*, "IEC 61850-based adaptive protection system for the MV distribution smart grid," *Sustainable Energy, Grids and Networks*, vol. 15, pp. 26 – 33, 2018, Technologies and Methodologies in Modern Distribution Grid Automation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352467716302077