Binder *et al. Energy Informatics* (2019), 2 (Suppl1): 20
https://doi.org/s42162-019-0084-0

Energy Informatics

RESEARCH Open Access

# Enabling architecture based Co-Simulation of complex Smart Grid applications

Christoph Binder*, Michael Fischinger, Lukas Altenhuber, Dieter Draxler, Goran Lastro and Christian Neureiter

*Correspondence:
christoph.binder@fh-salzburg.ac.at
Salzburg University of Applied Sciences, Center for Secure Energy Informatics, Urstein Süd 1, Puch/Salzburg, Austria

**Abstract**

The integration of decentralized prosumers into current energy systems leads to continuously increasing complexity in today's popular term of the Smart Grid. Since conventional engineering methods reach their limits when dealing with the challenges in developing such systems, model-driven approaches like Domain Specific Systems Engineering (DSSE) gain significant importance. Contributing to the agile development of such a System of Systems (SoS), the application of the DSSE approach is furthermore supported by the introduction of the Smart Grid Architecture Model (SGAM) and Mosaik. However, with both concepts being individual methodologies, their interconnection is missing specifications. Therefore, this paper proposes the development of an interface between architecting and simulating a complex Smart Grid. To achieve this, the concepts of SGAM and Mosaik are analyzed in the first place in order to set up a suitable architectural model of an energy system and the corresponding simulation scenario. Subsequently, the applicability of the present approach is demonstrated by utilizing an excerpt of a real-world case study, the charging behavior of an Electric Vehicle (EV).

**Keywords:** Smart grid, Domain specific systems engineering (DSSE), Co-Simulation, System architecture, Electric vehicles

## Introduction

Although first steps have been taken over a decade ago, the transition of the original power grid towards the so-called Smart Grid is gaining in importance in recent years. This is mainly encouraged by the recent discussion about the achievement of climate goals as well as novel technological advances provided by the Internet of Things (IoT). However, to find a common definition for the term *Smart Grid* has proven to be rather difficult, since different standardization bodies propose varying explanations (Greer et al. 2014; SMB Smart Grid Strategic Group (SG3) 2010; United States Department of Energy 2010). Nevertheless, especially Farhangi et al. (2010) provide a valuable and detailed illustration of how a Smart Grid can be distinguished from a conventional power grid. According to the authors, some characteristics like a digitalized two-way communication, distributed generation, pervasive control and the possibility to monitor itself are indications for a power grid to be intelligent. Those features are mainly enabled by combining aspects from power systems with methods from Communication and Information Technology,

but they also entail further challenges. For example, through the integration of renewables or decentralized prosumers, issues like electric storage or demand response management need to be dealt with. Moreover, by adding a large number of sensors or a wide range of intelligent agents, efficiency and reliability are secured on the one hand, but on the other hand the question of how to expand the infrastructure of the grid with the goal to secure interoperability arises. Taking the interplay of these Smart Grid components into further consideration, the overall aim needs to be maintaining the energy balance of the complete power system, which can be achieved by dynamically managing the power demand or by regulating flexibilities. Thus, due to the huge number of elements within a Smart Grid and their dynamic behavior, such a system needs to be classified as a complex system according to the scale introduced in Haberfellner et al. (2015). To further underpin this statement, in Binder et al. (2019) it is shown that even the further specifying term System of Systems (SoS) has to be used when describing this kind of system, since it fulfills the typical traits summarized by the contributions of Maier (1998); Sage and Cuppan (2001); DeLaurentis (2005).

As apparent from the previous assertions, developing and implementing future energy systems is a demanding task including numerous considerations. Thus, the authors of this paper previously introduced an approach for developing Smart Grid applications based on the Smart Grid Architecture Model (SGAM) (CEN-CENELEC-ETSI Smart Grid Coordination Group 2012), called Domain Specific Systems Engineering (DSSE) (Neureiter 2017). By applying the concept of Model-Based Systems Engineering (MBSE) and making use of well-known methodologies like Model-Driven Architecture (MDA) (Object Management Group 2014), the main result of their work is the provision of an adequate modeling tool going by the name of *SGAM Toolbox*[1]. However, regarding the findings gained through years of utilization in international projects and falling back to the fact of the Smart Grid being a SoS, unpredictable behaviors contradictory to what is described in the model become observable. In order to deal with these mostly unpredictable and usually undesirable effects before implementing the system, a Co-Simulation is considered to be an appropriate measure for realistically analyzing the behavior of the components within the Smart Grid during its runtime.

However, modeling and simulating in the Smart Grid are not completely new topics to discuss about. Several publications deal with Model-Driven Development (MDD) in this domain (Lampropoulos et al. 2010; Lopes et al. 2011; Andrén et al. 2013), whereas others propose the possibilty of simulating such a system (Godfrey et al. 2010; Yang et al. 2013; Palensky et al. 2013). Nevertheless, considering the number of stakeholders and the wide range of tools when developing an energy system, utilizing one particular tool would result in loss of information and limit the results. This means, the goal to follow is to connect specific tools that remain in their intended environment rather than to create an all-inclusive solution. Thus, the intention of this paper is to propose an interface developed for the SGAM Toolbox in order to exchange the model data with the simulation framework *Mosaik*[2] (Schütte et al. 2011), which has been chosen for its applicability and suitability towards the Smart Grid. By doing so, the focus is set to faciliate the process from modeling to simulating between the mentioned tools in order to enable their toolchain integration. More precisely, the main contribution of the conducted research

[1] https://sgam-toolbox.org/
[2] https://mosaik.offis.de/

is a newly created possibility to simulate an SGAM based model with just a few clicks, which enhances the design and concept workflow of such systems and also paves the road for further projects in this area. Thus, the developed interface can be seen as first prototype indicating the similarities between SGAM and Mosaik by outlining their respective differences in defining a Smart Grid at the same time.

To address these aspects, the remainder of this paper is structured as follows: "Related work" gives a more detailed overview of related work and the background of DSSE and Co-Simulation. After that, the intended "Approach" is depicted in detail and the research methodology is outlined in short. The most important implementation details of the interface are then presented in "Development of the interface", before the applicability is tested and evaluated in "Application of the Co-Simulation", using an exploratory case study. Finally, the paper is summarized and the conclusion is given in "Conclusion & future work".

## Related work

This section gives an overview of the related work and the state-of-the-art. As already mentioned, modeling in the Smart Grid and the simulation of its components during runtime is no unknown territory. Similar approaches have been in the focus of several research projects during the last decade. For example, the first steps of utilizing MBSE to model a power system from a SoS perspective in order to manage its complexity and address the concerns of different stakeholders has been set by Lopes et al. (2011). Although being in early stages of research, some proposed concepts find still usage at current times, like using planes for structuring a Smart Grid or applying SysML for modeling the system. Additionally, in Lampropoulos et al. (2010) a methodology specially focused on modeling the behavior of prosumers in the Smart Grid is introduced. By making use of simulating Electric Vehicles (EVs), first effects of their individual behaviors towards the complete power system could be investigated. A few years later, it has been recognized that standards for defining a Smart Grid need to be introduced in order to build a common foundation, which can exemplatory be seen in the *IEC Smart Grid Standards Map*[3]. Thus, a framework for Model-Driven Engineering (MDE) in the Smart Grid implementing the Common Information Model (CIM), the IEC 61850 as well as the IEC 61499 has been developed by Andrén et al. (2013). However, a more complete approach inheriting an underlying architectural framework and introducing an adequate development process has already been proposed by the authors of this paper (Neureiter et al. 2016). Moreover, this methodology has been in constant development resulting in the proposal of DSSE, which is observable by the continously updated SGAM Toolbox. Since this paper deals with further enhancing its functionality by developing an interface to Mosaik, used technologies like the SGAM and the particularities of this approach itself are described in more detail in the following, with a short introduction of Co-Simulation in the Smart Grid.

### Smart Grid Architecture Model (SGAM)

The *Smart Grid Architecture Model (SGAM)*, introduced by the European Standardization Mandate M/490, is an architecture model specification that gives an holistic

[3] http://smartgridstandardsmap.com/

view on Smart Grid systems (CEN-CENELEC-ETSI Smart Grid Coordination Group 2012). The SGAM provides capabilities for the comprehensible development of renewable energy systems. Figure 1 depicts an overview of the model. Based on the NIST Domain Model (Greer et al. 2014), the automation pyramid and the GWAC Interoperability Stack (Council 2008), the concepts have been extended to a three-dimensional structure, including domain-specific viewpoints.

Basically, the power grid can be distinguished into *Domains* and *Zones*. Whilst the SGAM Domain-axis arranges a system on basis of an element's position in the Smart Grid, the Zone-axis describes roles of elements with regard to automation possibilities. In addition to these basic power grid modeling approaches, the SGAM further comprises the so-called *Interoperabilty Dimension*, introducing layers for *Business*, *Function*, *Information*, *Communication* and *Component* aspects. Dependent on the requirements of a stakeholder, these layers (or viewpoints) can be considered as the entry point for the development of the aspects of interest.

The SGAM primarily delivers a general reference in how to architect Smart Grid systems. Beyond that, however, for the individual development of Smart Grid applications, DSSE combines adequate tools with an applicable methodology (Neureiter 2017).

### Domain Specific Systems Engineering (DSSE)

*Domain Specific Systems Engineering (DSSE)* is an approach that arose from the ideas of MBSE (Wymore 1993). The concepts of MBSE specifically target issues such as dealing with the upcoming complexity during the development of systems. As mistakenly assumend, models are not the key artifacts of development in MBSE itself, since this deals as an umbrella term for more specific methodologies. However, they significantly support

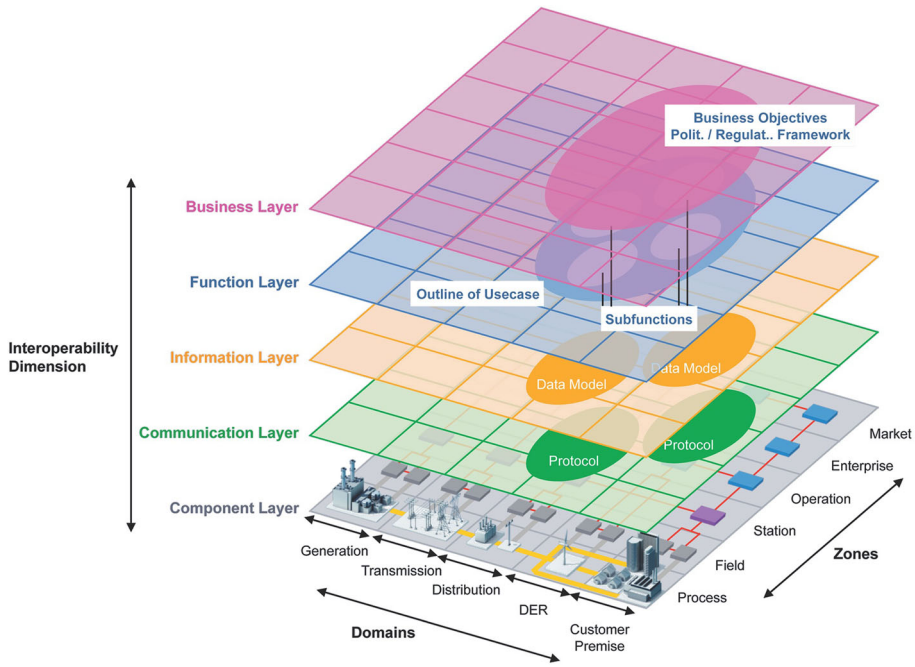


**Fig. 1** The Smart Grid Architecture Model (SGAM), used with permission from CEN-CENELEC-ETSI Smart Grid Coordination Group (2012)

the development of systems. Going one step further, model-driven approaches such as MDE and MDD consider models as key artifacts for system development throughout the whole development life-cycle. The Object Management Group (OMG) further defined an even more specialized concept of MDD, the so-called MDA (Object Management Group 2014). Generally, model support encourages an agile development dealing with rapidly changing requirements (Schmidt 2006).

However, besides to the noted benefits, the authors of Whittle et al. (2014) outline that domain specific engineering concepts are more successful than general model-driven engineering approaches. As a result of several years of research Neureiter (Neureiter 2017) therefore outlines an approach for DSSE in the Smart Grid. DSSE introduces an entire development process for Smart Grid systems. To meet the demand of an agile, sustainable process, DSSE inherits consistent supporting methods for the extensive system development in different phases of the development life-cycle. By establishing a MDA process for the Smart Grid, DSSE was initiated by the research of Dänekas et al. (2014). This research also includes the implementation of the SGAM Toolbox, which especially supports real world applicability. DSSE and the SGAM Toolbox particularly focus on the requirement-specific development of Smart Grid applications. On this basis, Neureiter (2017) recommends an extension of the DSSE approach, to finally reach a SoS "Integration Toolchain" for Smart Grids, which is depicted in Fig. 2. It includes eight steps of integration, which will be described in short in the following:

1. **GIS data import:** The data from the Geographic Information Systems (GIS) includes power system analysis from an electro technical point of view.
2. **Use Case import:** Besides to the electrical models, there are also central community-based Use Case Management Repositorys (UCRMs). The UCRM of the Oldenburg Institute for Informatics (OFFIS)[4] for example makes typical Smart Grid related use cases centrally available.
3. **Reference Architecture import:** As a starting point for development, it is suggested to import a general Reference Architecture, such as the NIST Conceptual Model (Greer et al. 2014).

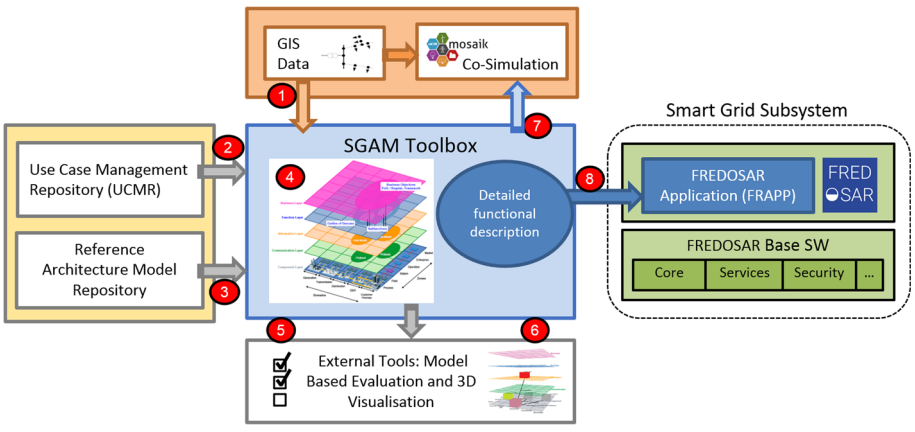


**Fig. 2** System of Systems "Integration Toolchain" for Smart Grids

[4] https://www.offis.de/

4. **Architecture Development:** Based in the previous imports, the architecture development of a specific use case is supported by the SGAM-Toolbox. The authors of Neureiter et al. (2016) describe a standard-based approach for domain specific modeling of Smart Grid system architectures.
5. **Model Evaluation:** The architecture development is followed by a model evaluation, which is based on defined Key Performance Indicators (KPIs). As an example, the assessment of privacy indicators is outlined in Knirsch et al. (2015).
6. **3D Visualization Tool:** The 3D visualization enables the manual evaluation of the architecture model in interdisciplinary environments. Thus, an appropriate 3D Visualization Tool has been developed for enabling this (Neureiter et al. 2014).
7. **Power System Simulation:** In order to evaluate a system's behavior in a greater context, for example as part of a SoS, Co-Simulation approaches can offer potential solutions. Tailored to Smart Grid applications, the Mosaik Co-Simulation framework offers appropriate tool support.
8. **Source Code Generation:** The ongoing research in MDE also includes concepts for the automated generation of source code. The transfer from a detailed functional description to concrete executable software is aimed. The service-oriented middleware framework FREDOSAR[5] is one of the previous achievements for supporting the implementation of model-driven or model-centric software development concepts (Fischinger et al. 2019). However, there is still a great potential for further research in this area.

The seventh step of the SGAM-based toolchain already gave a brief introduction to the simulation of power systems. A more detailed insight into Co-Simulation in the Smart Grid will be given in the following.

#### Co-Simulation

As MBSE has become more popular for developing Smart Grid applications, approaches for simulating them needed to arise. Therefore, a collection of suitable tools and methods has been published by Palensky et al. (2013). However, with most frameworks focusing on simulating the behaviour of single components and the Smart Grid inheriting SoS characteristics, the need for Co-Simulation methodologies has become more obvious in order simulate an energy system on system-level. Thus, first attempts have been proposed by Godfrey et al. (2010), where the authors simulate Photovoltaic Cells (PCs) and try to find countermeasures for dealing with a temporary loss of power. A more generic example has been introduced in Yang et al. (2013), where a framework for event-driven simulation of previously modelled Smart Grid components is provided. A specific feature of this approach is the high degree of freedom in the framework itself, each controller within the Smart Grid can be simulated according to its inteded functionality by describing its function block architecture.

Per definition, Co-Simulation is the coordinated execution of two or more models with different representations and behaviors during execution in the runtime environment (Steinbrink et al. 2017). In more detail, the underlying modeling paradigm purports that each model is represented as a differential equation, which is executed during runtime.

[5] https://www.fredosar.org/

This causes independent development and implementation of individual models in order to provide an optimal overall solution. In a Co-Simulation scenario itself, the specific simulations are considered as black-boxes, which are developed by the teams responsible for the domain specific systems. The most important engineering domains requiring the utilization of Co-Simulation are automotive, heating and ventilation, robotics, or electricity production and distribution domain (Gomes et al. 2017).

In order to work properly, a Co-Simulation is composed of at least two simulators and a master algorithm, which orchestrates the simulators and manages data exchange or time synchronisation. A simulator is a software package or a tool that consits of a model and a solver. However, in the case of a power system simulation, a model contains physical elements and their interconnections. The simulator's job is to take these descriptions of the system and transform them into equations that can be processed by the solver. Thereby, output variables of one simulator become the input of one or more other simulators and vice versa (Palensky et al. 2017). This leads to a dynamic coupling of the different simulators that compose the Co-Simulation. To synchronize the outputs of the different time resolutions, fixed exchange times are defined depending on each simulation scenario, which are defined as steps. To integrate continuous output values, they are treated as fixed stepped output values with a low step range. Combining different fixed step sizes or variable stepped output values, the tool that manages the synchronization must support the possibility to skip simulators at certain steps. This is different for the integration of event driven simulators, where an event can only be handled in the next step. In case of the occurrence of inexact results, a reduction of the step size or the implementation of a roll-back function may improve the outcome of the Co-Simulation.

A practical example of such a Co-Simulation framework is Mosaik. Tailored to the Smart Grid, the open source tool is written in Python and integrates a specific power grid simulator like PyPower. To address all aspects a Co-Simulation has to consider, Mosaik is constituted of four main components. In its core, the Sim Manager is responsible for processing the simulators and their interconnection, while the Scheduler tracks the dependencies between the simulators and performs simulation steps. Additionally, for developing a simulation scenario, the Scenario-API is connected to the core and can be addressed with python code, defined as the Scenario Script. Finally, for enabling the communication between the simulators and Mosaik, a Sim-API is provided by the framework. Thereby, a designated Component Interface is implemented in order to manage the communciation over plain network sockets via JSON encoded messages.

## Approach

With regard to systems engineering in the Smart Grid, the insight into current state-of-the-art research showed that many questions have already been answered. Especially the DSSE methodology and the Co-Simulation approaches with Mosaik offer promising solutions for their problems. Furthermore, the SGAM-Toolchain introduced a general process for the integration of Smart Grid based applications. However, aspects like agility and the development in rapid iterations combined with Co-Simulation have not been covered yet. To completely fulfill this idea of the SGAM-Toolchain, a seamless transition between the model of a system and the simulation is needed. On these grounds, the research of this paper aims the implementation of an appropriate interface between the SGAM Toolbox and the Mosaik framework. To give a short overview, the overall

purpose of this contribution is to clarify the following three points: (1) development of a system architecture and proper Co-Simulation scenarios in the Smart Grid, (2) specifying the corresponding toolchain between the model and the simulation as well as developing its interface and (3) identifying the limitations of the interface. To investigate these research questions, a suitable case study representing a typical energy system example has been created. Therefore, this section gives an overview of the chosen research methodology and outlines the development approach for the intended interface.

As the further development of the Smart Grid and related supporting tools is strongly driven by joint, distributed research activities, an agile research approach is recommended. The Agile Design Science Research Methodology (ADSRM) therefore offers a possibility for application-related, scientific research and development (Conboy et al. 2015). To undermine certain achievements from research and development, the ADSRM orientates to exploratory case studies. Therewith, the applicability and the reasonableness of application-related research can be evaluated. Hence, the methodology of this paper is based on the ADSRM.

According to these considerations, the ADSRM offers various so-called research entry points. For the present research, the question for agility and rapid development combined with Co-Simulation is the favored research entry point. More precisely, the implementation of an applicable interface between the SGAM-Toolbox and Mosaik is aimed. Hence, the interface will be finally evaluated according to its applicability. This will be based on an exploratory case study, making use of a typical contemporary Smart Grid example.

Thus, in this case an electrical power grid including 50 households is used for the simulation scenario. Additionally, 30 EVs and 30 PCs are randomly assigned to the households, resulting in different kinds of consumers and/or producers operating none or at least one EVs as well as PCs. In order to establish the link between the architectural model and the Co-Simulation, in this specific example the charging behavior of each EVs is exported from the model and embedded into the simulation scenario. However, the detailed application of the case study itself in order to evaluate the predefined interface is given in "Application of the Co-Simulation". Beforehand, an overview of the detailed procedure to implement the intended interface is outlined as follows.

The main goal of the aspects mentioned above is the possibity to allow a quick repetition between the problem definition and the generation of code. A simple illustration of the aimed Simulation approach on SGAM models is therefore given in Fig. 3.

Regarding emergent behavior in SoS, for the simulation of Smart Grid related applications, especially functional aspects play an important role. Hence, the Co-Simulation in this study is purposefully centered to detailed functional models out of the SGAM Function Layer. For now, these models are available in the form of Enterprise Architect (EA)[6] *Activity Diagrams* within the SGAM-Toolbox. These detailed functional models are used to generate partial aspects of a software artifact. Then, specially prepared software templates, which are customized for the Mosaik target platform, are used to solve the suitability problem of the generated code. Thereby, the artifacts from the code generation are embedded in the ready-made Mosaik templates. Finally, the resulting code artifacts are well prepared to be part of a Co-Simulation "within" the Mosaik engine.

[6] https://www.sparxsystems.de/

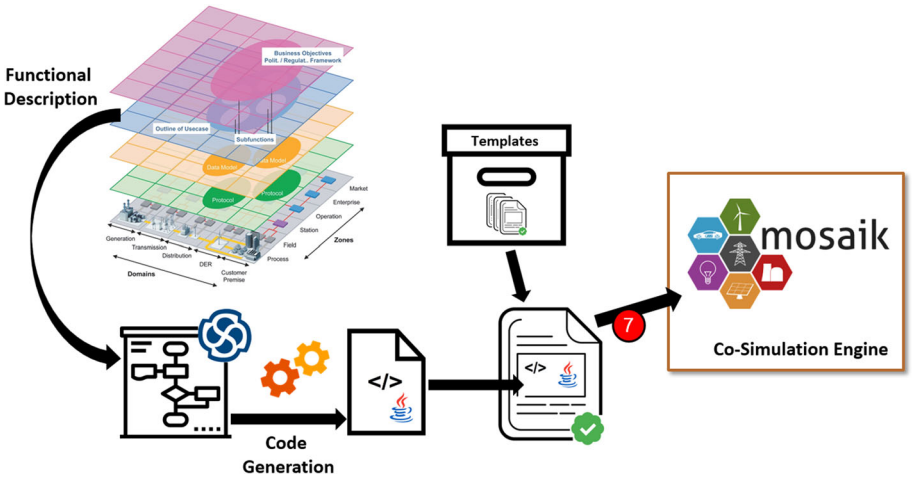


**Fig. 3** Simulation approach based on SGAM models

The illustrated approach serves as a basis for the concrete implementation of the interface. To go into more detail, the most important implementation achievements are outlined in the following section.

## Development of the interface

As previously mentioned, the goal of creating an interface between SGAM and Mosaik is addressed by utilizing the respective programming interfaces in order to transmit the information from the architecture to the simulation. Hence, suitable technologies and frameworks need to be used for developing a software appealing to these specifications. An example of such a tool for modeling current and future energy systems is the SGAM Toolbox. Through years of development and application in international projects it has established itself as a technology driver in this area. Since the SGAM Toolbox is available for free, is offering various functionalities and is easy to adapt, it is the tool of choice for implementing an additional method realizing the interface between SGAM and Mosaik. To specify this in more detail, the toolbox introduces features concerning usability and automation of recurrent modeling processes. One of these features is the provision of a domain-specific modeling language. This so-called Domain Specific Language (DSL) targets the application domain and physical world of the Smart Grid by inheriting domain-specific elements, which are represented by stereotypes and metaclasses derived from the Unified Modeling Language (UML). Therefore, the first step is to extend this DSL in order to consider the information that is needed for simulating the interconnection of elements within an architectural model. For example, a new element called *MosaikSimulatorConfig* is added. As the name assumes, this element deals with providing configuration data for adjusting the exported simulators in Mosaik. By doing so, it is derived from the UML *metaclass* Artifact and extends it with additional attributes for placing the configuration values, which are explained in more detail in the following:

- *fileName:* This attribute is provided by the metaclass Artifact and is used to save the path to the simulator executable file. According to the implementation of the simulator, this file is either a Java or a Python type.

- *confKey:* In this attribute the connection type for the simulator is described. The simulator could be connected in-process, started and executed in an own thread or linked to another running simulator.
- *confValue:* The information needed for starting the simulator is stated here. Depending on the connection type, it could be a Python class, a terminal command or the IP address and the port to a running simulator.
- *isModeled:* This Boolean value is set if the simulator is generated out of the model. If it is true, the functional description in the Primary Use Cases are used to generate code for a simulator. If isModeled is set to false, the simulator itself already exists and only has to be connected to the Co-Simulation.
- *models:* Additionally, if the attribute isModeled is true, all ID's of assigned Primary Use Cases and additional model information are taken for use, which are stored in this attribute.
- *stepSize:* The number of steps that elapse till the simulator is executed again are represented in this attribute. When generating code from the model, currently only a fixed step size is able to be stored, which cannot be changed during runtime.

### Simulator configuration

A specific user interface is designed to ensure the easy usage of the Co-Simulation integration within an SGAM-based architectural model. The main intention of this interface is to enable the possibility to configure the previously mentioned attributes and provide additional model information like contained Primary Use Cases and links to their respective behaviors. Therefore, the user interface is divided into two parts. On top general Co-Simulation settings can be found and at the bottom the particular settings are stated for those simulators, which are generated based on the architectural model. As seen in Fig. 4,

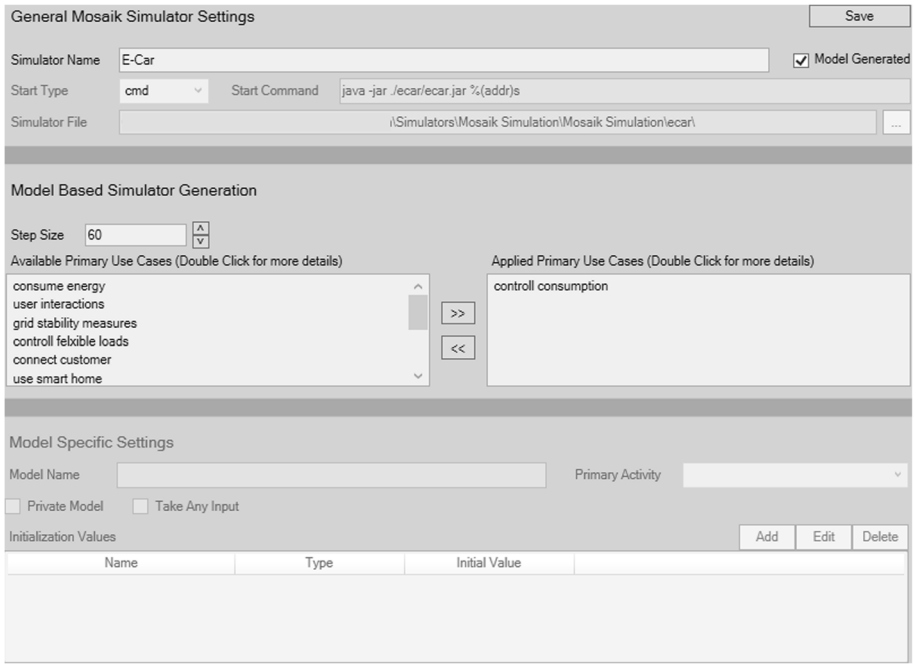


**Fig. 4** Simulator Configuration User Interface

the interface additionally ensures that all inputs are correct, applying them for usage in the subsequent code generation.

In more detail, the settings on top of the user interface are general settings for the simulator like the start type, the start command, the step size, the link to the simulator file, and a check box to generate the simulator based on the model. Those values can be stored for later usage for each simulator separately. However, if the simulators need to be generated from the SGAM model, the bottom area has to be considered. In the left window all Primary Use Cases that are related to the current instance of the simulator configuration are listed. By shifting them to the window on the right-hand side, their behavior is exported to the Co-Simulation. Additionally, by double-clicking on each Use Case, this behavior can be viewed in more detail. Usually, an activity diagram is taken for use in order to precisely model sequence of events. Beyond these windows, the detailed configuration of a single simulator model is given by introducing five settings. The model name titles the model as it is represented in the Co-Simulation, whereas the init values contain the initialization parameters passed to the simulation model at the instantiation. The input and output values on the other side define the variables that can be passed to a simulator model during its runtime. The last setting deals with specifying the root activity by selecting a primary activity in the drop-down menu.

#### Code generation

After setting up the configuration details for all simulators used in the simulation scenario, they have to be exported from the model and imported in Mosaik. This is done by developing an additional method to the SGAM Toolbox. Hence, by clicking on the Co-Simulation function, at first all considered simulators are collected from the model. After choosing a suitable configuration duration in single steps, the behavioral UML diagrams deal as base for the future code generation. Technically speaking, in Mosaik each simulator consists of three Java classes. Consequently, each modeled activity diagram needs to be transformed into those three classes. Since exporting class diagrams as XML-files is a difficult task, templates of Java classes are created in the first place. The export scenario makes use of these templates and replaces or adds each used code snippet during its application. An easy way of dealing with this is provided by the framework *StringTemplate*, which enables to set markers in template files and replace those with generated code. Utilizing this method, the init, input and output values originating of each simulator configuration as well as the behavioral code described in UML are applied to generate the needed Java classes. However, the generation of functional code based on behavioral diagrams in EA has some restrictions, which have to be considered. First, all behavioral diagrams have to be a child element of the respective class. Secondly, additional Activities are not allowed to be a child element of an Activity. This means that all Activities that should be considered for the code generation have to be a direct child element of the respective class. To work around these restrictions, the Add-In deals with moving and copying elements within the model, so that the structure is in the right order for each simulator to be exported correctly. Furthermore, the behavioral diagrams need to be represented as classes themselves since code generation is restricted to this type of UML diagram. Nevertheless, after exporting the Java classes, an executable .jar file is generated and placed in the Mosaik folder. At this place Mosaik itself is able to access the Java code and utilize it during its Co-Simulation run.

## Application of the Co-Simulation

To evaluate the dependencies of the conceptual interface between the architectural model and the Co-Simulation framework, a typical case study is applied. As already mentioned, this work therefore makes use of a representative modern energy-related example, in particular of an EV charging system. By doing so, this specific scenario consists of three steps, realizing an excerpt of the whole toolchain. First, the system itself, including all involved components, is modelled according to the specifications of SGAM and with the help of the associated toolbox. Additionally, the EVs and their charging behavior are also added to the model. In a further step, the functionality of the interface itself is examined by exporting the Co-Simulation settings and the EV behavior from the model. As follow-up, Mosaik is configurated accordingly and the EVs are instantiated as well as embedded in the simulation environment. In the last step the Co-Simulation is applied.

According to these considerations, the Business Layer contains three major actors, Distribution Service Operators (DSOs), Energy Service Providers (ESPs) and Private Customers. Each of these actors is connected to a Business Use Case (BUC) to fulfill the Business Goals. However, those use cases themselves are realized by High Level Use Cases (HLUCs) such as *provide energy*, *consume energy* and *provide grid stability*, to mention some examples. Being the result of the system analysis, the last part of the Computation Independent Model (CIM) is to define requirements, which are derived from the HLUCs. Those requirements deal as a base for the functional description of the system. Therefore, a model transformation between the CIM and the Platform Independent Model (PIM) takes place, where the Business Actors are traced into Logical Actors and Primary Use Cases (PUCs) are developed to describe the sequence of events realizing a requirement's dynamic behavior. To emphasize the overall purpose of the case study of simulating EVs, this example makes use of the PUCs *consume energy*, *grid stability measures* and *user interactions*. The detailed refinement of these use cases via activity or sequence diagrams are the main source of Mosaik to receive the different behaviors of an EV. Considering this in more detail, the charging behavior of the EV is depicted in Fig. 5, where it describes the processes of the PUC *consume energy*. The depicted process represents one step of the charging cycle and is triggered by the "plug-in"-event of the EV. Hence, the goal of the first action is to receive the current price of electricity within the power grid. The next step is to check the charging state of the battery for querying whether it is at 100% or not. If the battery does turn out to be fully charged, the charge cycle is completed. If the battery however is anything below 100%, it is forwarded to the next step. Thereby, the maximum price at which the EV is allowed to charge is determined. If the price is within the range of the EV to be allowed to charge, another activity realizing the actual charging procedure is called. Subsequently, the overall process representing the charging strategy is finished. However, if the price is too high, the charging process will not be triggered resulting in an immediate completion of the step.

As the Function Layer deals with the functional description of the system, the Component Layer indicates the physical representation. This is initiated by transforming the functional elements and logical actors from the PIM to physical components on the Platform Specific Model (PSM). However, this connection is realized by tracing one logical actor to one or many components or vice versa, many to one. By doing so, in this example the logical actor *flexible load* is traced to an EV and an EV controller in the actor mapping model. As explained in "Development of the interface", this physical

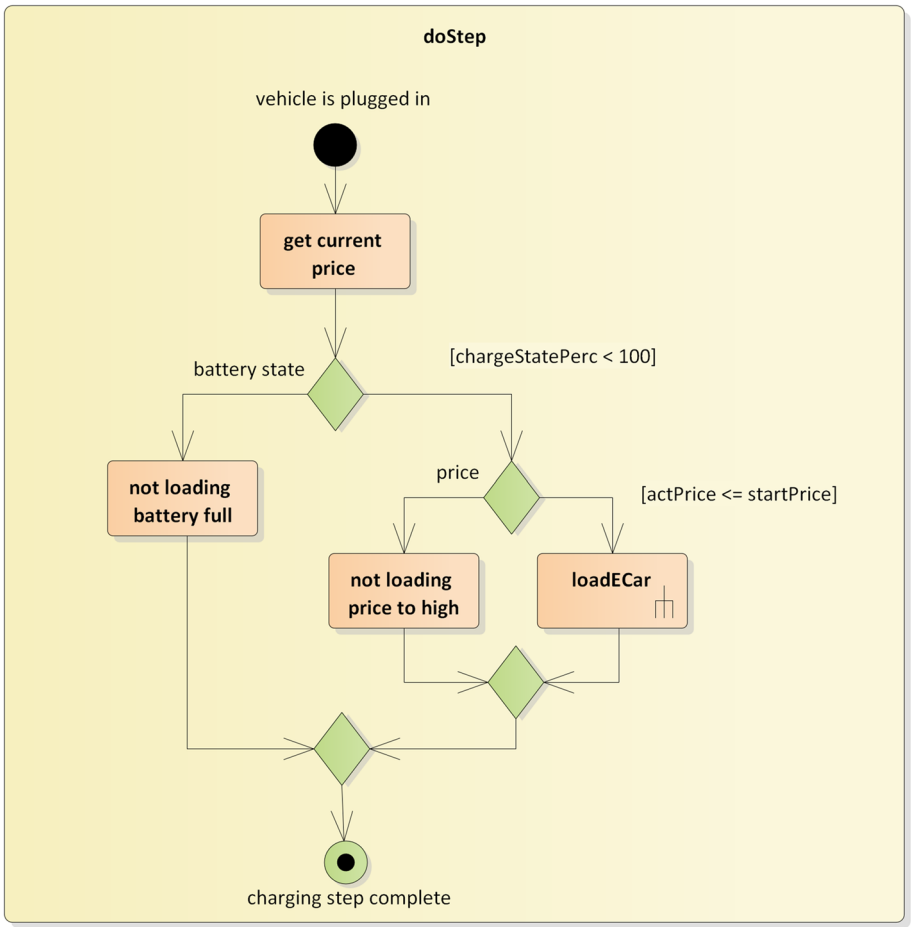


**Fig. 5** Activity diagram of EV charging behavior

representation of the EV contains important information for setting up the Co-Simulation scenario, which is exported and applied with the interface proposed in this work. Additionally, the resulting physical elements can be aligned to the corresponding SGAM pane in order to satisfy the architectural restrictions. This means, since the process zone is representing the energy flow from producing to consuming energy, the EV is placed on the right-hand side in the customer premises field. The purpose of the upper zones is to process the physical element's data. While the Component Layer deals with the phyisical representation, information handling is done in the Communication and Information Layer. Hence, the Information Layer deals with specifying which data is send and the underlying standards while the Communication Layer specifies the protocols or interfaces for transmitting the data. This information is used to interconnect the single simulators with each other in the following Co-Simulation scenario.

#### Co-Simulation execution

Before a simulation run can be executed, Mosaik needs to be configured properly. A designated configuration file defines how each simulator is connected to the scenario. Since this information is exported from the model, this file is generated dynamically from the interface described in this contribution. Furthermore, the component interface, an

additional file inheriting all meta data like input and output variables for each simulator model, is created in the same step. More detailed, a simulator model consists of its own model and an integrated solver. In this example, the exported EV represents the model and the charging process deals as solver for its model. However, during its runtime, multiple of those models are connected to Mosaik via a specifically designed socket and the help of an Application Programming Interface (API). This API deals with initiating and stopping the simulator as well as passing or requesting information. Consequently, all simulators and the surrounding environment is are interconnected in the main scenario script of the Co-Simulation, the so-called "world-file". This control algorithm sets up the simulation by reading all configuration files and instantiating the simulators, as well as defining a scheduler for executing those simulators. The output of the web visualization displaying the stored HDF5 data is visible in Fig. 6.

In this specific example, a total of 30 EVs with the previously defined charging strategy are connected to the scenario. To provide a typical Smart Grid environment, additional 50 households, as each one would operate an EV, and 30 photovoltaic cells are included together with a price generation simulator. After defining the overall execution time of 24 hours and the step sizes for each instantiated simulator, more detailed three minutes for the EV and 15 minutes for other components as well as the price generator being executed every minute, the Co-Simulation scenario is ready for application.

Summarized, the previously defined environment is depicted by 50 nodes, which are represented by grey outlining and connected via power lines. The red ring displays the reference bus and represents the transformation station to the upper grid level. Furthermore, a household simulator, demonstrating the consumption of one household, and an EV simulator are connected to each node, indicated by the blue surroundings. To round

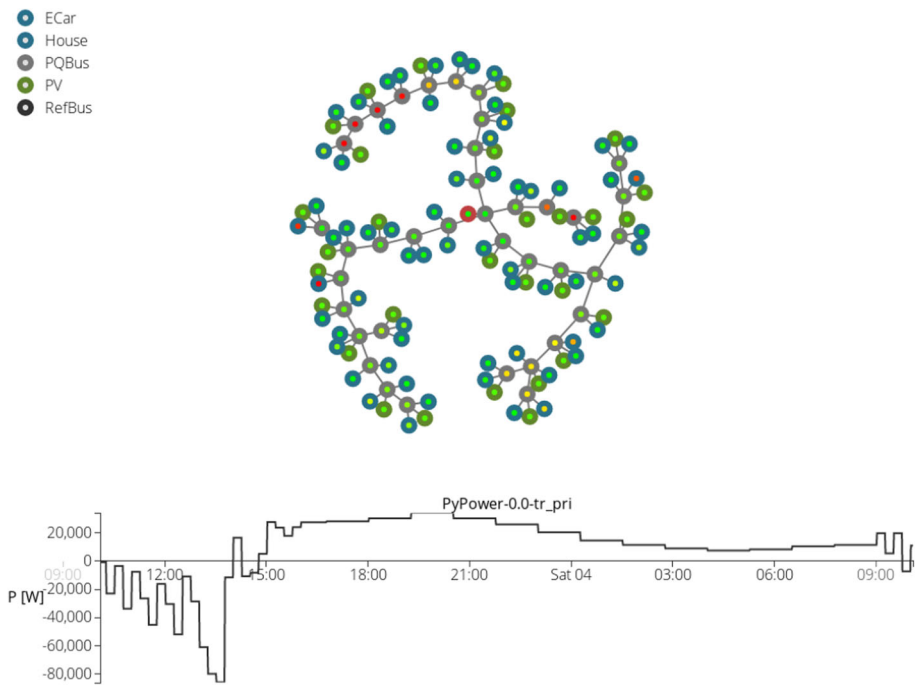


**Fig. 6** Mosaik overall consumption

off the visualized power grid, each of the 30 photovoltaic cells is connected randomly to one of the nodes. At the bottom of the figure the whole energy consumption over to coarse of one day is depicted. As seen, the energy shortage due to the high demand of the households just before noon is superseded by a surplus caused by the high production originating from the photovoltaic cells. However, to consider the energy consumption caused by a single EV simulator, the exported JAVA code is executed within the scenario. The code executed in this example is depicted in Listing 1.

```
if (chargeStatePerc < 100) {
if (actPrice <= startPrice) {
actChargingPower = maxChargingPower;
chargeState += maxChargingPower/4;
chargeStatePerc = chargeState/capacity*100;

if (chargeStatePerc > 100) {
actChargingPower = 0;
chargeState = capacity;
chargeStatePerc = 100;
}
}
else { actChargingPower = 0; }
}
else { actChargingPower = 0; }
}
```

**Listing 1** Generated EV battery charging algorithm

The delineated code is based on and automatically generated from the previously modeled activity diagram and allows the EV only to charge at decent price. Consequently, this results in an exemplatory charging cycle depicted in Fig. 7. There are three periods charging uptimes with the maximum consumption of 2000 W needed for the battery to be fully charged. More detailed, the car is charged in the night during low overall consumption, before noon and in the afternoon, where the PCs are producing energy.

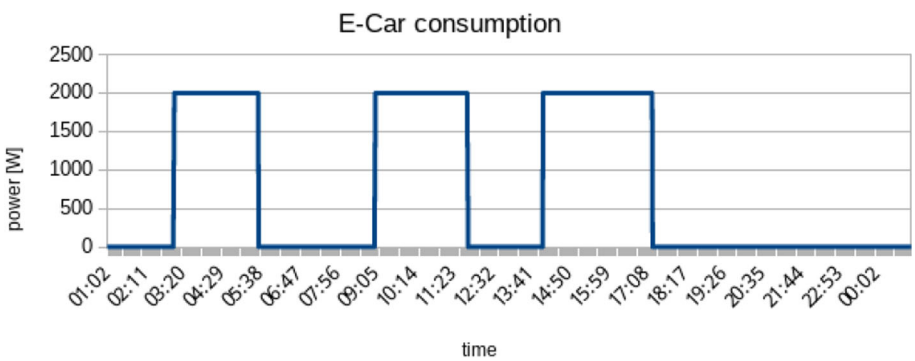


**Fig. 7** Mosaik exemplatory EV consumption

### Findings

The applied case study described in this section has been utilized in order to evaluate the functionality of the previously developed interface. As this link between the SGAM and Mosaik does not represent an out-of-the-box solution but rather can be considered as a first prototype on the way to establishing an integrated engineering toolchain for the SGAM Toolbox, a special focus on answering the research questions defined in "Approach" is set. Since the way to investigate these research questions is the application of typical energy system use case, the chosen evaluation method is observational based, as introduced by Hevner and Chatterjee (2010). In an observational based evaluation, the designed artifact is studied using a case study in a certain business environment. Therefore, the artifact created in this work is the concept and implementation of an interface between the previously mentioned tools in order to provide a solution for simulating architectures of energy systems with a single click. By developing the case study and applying it to this interface, the following findings have been observed.

Developing system architectures according to SGAM has been relieved significantly with the introduction of the SGAM Toolbox. As this tool is broadly accepted and widely used, efforts for describing the architecture of the case study have been kept to a minimum, since similar examples are already existing. As far as Mosaik is concerned, this tool entails several advantages. In detail, it implies a solid documentation, it is freely available, it was developed especially for the application in the Smart Grid, it is relatively easy to use and provides a Java and Python API. As this framework implements a main scenario file per default, the addition of external simulators for usage in the simulation scenario can be managed quite simple. Thus, the next question deals with how the relevant information is exported and provided for the Mosaik API to be further processed.

Due to these insights, the concept for the interface follows two main objectives. The first part deals with the generation of simulators based on the functional description in the model, which is addressed by applying code generation based on activity diagrams. Whereas not all information required for the Co-Simulation is contained in the architectural model, an additional artifact has been added to the DSL of the SGAM Toolbox. By executing the function for code generation, these artifacts can be instantiated. This means, in this case any number of EVs making use of different charging algorithms can be generated and prepared for simulation with Mosaik. The second part of the concept addresses the scenario definition of the Co-Simulation based on the architectural model. As SGAM itself does not provide any possibility to define these values, they need to be added to the previously created artifact. More precisely, during the instantiation of simulators, additional information like their start, instantiation values, connection and data exchange is passed on to Mosaik with the help of this domain-specific element.

With the experience gained in the implementation and utilization of the interface some limitations have anteceded. For example, code generation in EA has some restrictions like missing error outputs. This can lead to problems, if mistakes in activity diagrams occur, the code is not generated and no warning is displayed. Another limitation of the implementation arises from the initialization of simulators. Special initializations like random values have to be added manually to the generated code. This means, the creation of different Co-Simulation scenarios for the same architecture could be a problem at the moment. One way to solve this could be an additional artifact to define multiple scenarios. However, a major concern is the possibility to describe communication

architectures in SGAM, but Mosaik does not provide a solution for treating this. Thus, accounting this in the present example would exceed the scope of this work. As soon as Mosaik offers the possibility to apply a communication simulation, this has to be considered and added to the interface. Hence, the current implementation has been tested with a superficial use case. In a future work more sophisticated use cases are needed to evaluate the interface in more detail and find room for improvements. Nevertheless, the current implementation indicates the direction to follow by providing an overview of the advantages and disadvantages of this approach, described in the following in more detail:

- One benefit of this approach is the special focus on usability. Code can be generated dynamically from the model and provided to the simulation with just a few clicks. Since changes in the model are most likely to occur in such an agile domain, the way to simulate those models has been considerably improved.
- Another advantage is the consistent traceability between the model, the simulation and the actual implementation of single SGAM components. This enables furthermore the possibility to apply round-trip engineering within the DSSE toolchain.
- A disadvantage to mention deals with the integrability. Due to the integration of more tools in order to fulfill the toolchain, several trade-offs needed to be induced in order to keep mutual dependencies, as explained earlier.
- Another drawback entails constrictions in modeling the power system. In order to work properly with Mosaik, a lot of different aspects need to be considered in the model, otherwise the code generation is likely to produce errors. However, this can be solved with implementing correspondence rules or model checkers into the SGAM Toolbox.

## Conclusion & future work

Due to several domains involved in the Smart Grid and the amount of components that are statically or dynamically connected, the Smart Grid can be defined as SoS. In order to develop such a critical infrastructure inheriting considerable complexity, suitable engineering techniques like the DSSE approach need to be applied. Moreover, introducing a DSL, a suitable tool-support and the corresponding toolchain, the introduced method for architecture evaluation is Co-Simulation. However, due to missing specifications of linking the architectural model with a Co-Simulation, there is a gap appearing between the two concepts. Therefore, the conducted research contributes by providing an interface in order to simulate previously designed Smart Grid system architectures. The creation of this interface is described in "Development of the interface" in more detail. Firstly, the behavior of the component to simulate is exported from the model and realized by a solver in the respective simulator. The second part is to export the information for setting up the whole Co-Simulation environment according to the architectural model. Building up on the generated code, the simulation scenario is executed, as described in "Application of the Co-Simulation". Again, falling back on the complexity of such a system, a suitable simulation method needs to be utilized. Hence, the right way to approach this issue is applying a Co-Simulation scenario such as implemented by the Mosaik framework.

## Future work

However, even if the presented concept demonstrates a possible way on how to develop an interface between SGAM and Mosaik, it does not claim to be a "ready to use" methodology, but rather can be seen as a first step on the way to establishment. One of the most important tasks to improve this approach will be the complete scenario definition in the architectural model. This means, all simulators, the surrounding simulation environment and other aspects are directly generated from the model for immediate execution without using pre-defined settings. Afterwards, a more sophisticated case study needs to be utilized for evaluating this complete implementation. Additionally, to increase the interoperability to other Co-Simulation frameworks and to foster the reusability of simulators standards like Functional Mock-up Interfaces (FMIs), the System Structure and Parameterization (SSP) or the Distributed Co-Simulation Protocol (DCP) need to be considered and implemented in future versions. However, the last step in completing the DSSE toolchain is the generation of code to use in real software. Therefore, the approach proposed in this work can contribute to this goal by preparing and evaluating the model-generated code and prepares it for utilization in frameworks like FREDOSAR.

### Abbreviations

ADSRM: Agile Design Science Research Methodology; API: Application Programming Interface; BUC: Business Use Case; CIM: Computation Independent Model; DCP: Distributed Co-Simulation Protocol; DSL: Domain Specific Language; DSO: Distribution Service Operator; DSSE: Domain Specific Systems Engineering; EA: Enterprise Architect; ESP: Energy Service Provider; EV: Electric Vehicle; FMI: Functional Mock-up Interface; HLUC: High Level Use Case; KPI: Key Performance Indicator; MBSE: Model-Based Systems Engineering; MDA: Model-Driven Architecture; MDD: Model-Driven Development; MDE: Model-Driven Engineering; OFFIS: Oldenburg Institute for Informatics; OMG: Object Management Group; PC: Photovoltaic Cell; PIM: Platform Independent Model; PSM: Platform Specific Model; PUC: Primary Use Case; SGAM: Smart Grid Architecture Model; SoS: System of Systems; SSP: System Structure and Parameterization; UCRM: Use Case Management Repository; UML: Unified Modeling Language

### Acknowledgements

The support for valuable contributions of LieberLieber Software GmbH and successfactory consulting group is gratefully acknowledged.

### About this supplement

This article has been published as part of *Energy Informatics* Volume 2 Supplement 1, 2019: Proceedings of the 8th DACH+ Conference on Energy Informatics. The full contents of the supplement are available online at https://energyinformatics.springeropen.com/articles/supplements/volume-2-supplement-1.

### Authors' contributions

CB and MF declared the problem definition, conducted the study, discussed the results and contributed in writing the manuscript. CB furthermore refined the manuscript according to the reviewers' comments and prepared the camera-ready version. LA developed and analyzed the proposed concepts by realizing the implementation with the help of GL. DD evaluated the solution by applying the tool in several projects and lectures and contributed by providing critical feedback. CN provided the initial formal problem definition and contributed academically by offering support and preliminary works. All authors read and approved the final manuscript.

### Funding

The financial support by the Austrian Federal Ministry of Science, Research and Economy, the Austrian National Foundation for Research, Technology and Development and the Federal State of Salzburg is gratefully acknowledged. Publication of this supplement was funded by Austrian Federal Ministry for Transport, Innovation and Technology.

### Availability of data and materials

The architectural model has been created with the help of Enterprise Architect and is publicly avialable at https://sgam-toolbox.org/Co-Simulation. The code developed as result of the present contribution is implemented as part of the SGAM Toolbox, which is free avilable at https://sgam-toolbox.org/download. The source code and documentation of Mosaik can be found on https://mosaik.offis.de/. Additionally, all data generated and analyzed in this contribution are available from the corresponding author on reasonable request.

### Competing interests

The authors declare that they have no competing interests.

Published: 23 September 2019

**References**

Andrén F, Stifter M, Strasser T (2013) Towards a semantic driven framework for smart grid applications: Model-driven development using cim, iec 61850 and iec 61499. Informatik-Spektrum 36(1):58–68
Binder C, Gross J-A, Neureiter C, Lastro G (2019) Investigating emergent behavior caused by electric vehicles in the smart grid using Co-Simulation. In: 2019 14th Annual Conference System of Systems Engineering (SoSE). IEEE, Anchorage. in press
CEN-CENELEC-ETSI Smart Grid Coordination Group (2012) Smart Grid Reference Architecture
Conboy K, Gleasure R, Cullina E (2015) Agile Design Science Research. In: Donnellan B, Helfert M, Kenneally J, VanderMeer D, Rothenberger M, Winter R (eds). New Horizons in Design Science: Broadening the Research Agenda: 10th International Conference (DESRIST). Springer, Dublin, Ireland. pp 168–180
Council T (2008) Gridwise interoperability context-setting framework. GridWise Archit Counc Battelle Memorial Inst 1.1:1–52
Dänekas C, Neureiter C, Rohjans S, Uslar M, Engel D (2014) Towards a model-driven-architecture process for smart grid projects. In: Benghozi P-J, Krob D., Lonjon A., Panetto H (eds). Digital Enterprise Design & Management, Advances in Intelligent Systems and Computing. Springer, Paris, France Vol. 261. pp 47–58
DeLaurentis D (2005) Understanding transportation as a system-of-systems design problem. In: 43rd AIAA Aerospace Sciences Meeting and Exhibit. Aerospace Research Central, Reno, p 123
Farhangi H (2010) The path of the smart grid. IEEE Power Energy Mag 8(1):18–28
Fischinger M, Neureiter C, Binder C, Egger N, Renoth M (2019) Fredosar: Towards a security-aware open system architecture framework supporting model based systems engineering. In: 8th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS). SciTePress, Heraklion, Crete - Greece. in press
Godfrey T, Mullen S, Griffith DW, Golmie N, Dugan RC, Rodine C (2010) Modeling smart grid applications with co-simulation. In: 2010 First IEEE International Conference on Smart Grid Communications. IEEE, Gaithersburg, pp 291–296
Gomes C, Thule C, Broman D, Larsen PG, Vangheluwe H (2017) Co-simulation: State of the art. arXiv preprint arXiv:1702.00686
Greer C, Wollman DA, Prochaska DE, Boynton PA, Mazer JA, Nguyen CT, FitzPatrick GJ, Nelson TL, Koepke GH, Hefner Jr AR, et al. (2014) Nist framework and roadmap for smart grid interoperability standards, release 3.0. Technical report
Haberfellner R, de Weck, O, Fricke E, Vössner S (2015) Systems Engineering - Grundlagen und Anwendung, 13 edn. Orell Füssli, Zürich, Schweiz
Hevner A, Chatterjee S (2010) Design Science Research in Information Systems - Theory and Practice. Springer, Berlin Heidelberg
Knirsch F, Engel D, Neureiter C, Frincu M, Prasanna V (2015) Model-driven privacy assessment in the smart grid. In: Proceedings of the 1st International Conference on Information Systems Security and Privacy (ICISSP). SciTePress, Angers. pp 173–181. Best Paper Award
Lampropoulos I, Vanalme GM, Kling WL (2010) A methodology for modeling the behavior of electricity prosumers within the smart grid. In: 2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe). IEEE, Gothenberg, pp 1–8
Lopes AJ, Lezama R, Pineda R (2011) Model based systems engineering for smart grids as systems of systems. Procedia Comput Sci 6:441–450
Maier MW (1998) Syst Eng J Int Counc Syst Eng 1(4):267–284
Neureiter C (2017) A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid. MBSE4U - Tim Weilkiens, Hamburg, Germany
Neureiter C, Engel D, Trefke J, Santodomingo R, Rohjans S, Uslar M (2014) Towards consistent smart grid architecture tool support: From use cases to visualization. In: Proceedings of IEEE Innovative Smart Grid Technologies (ISGT) 2014. IEEE, Istanbul, Turkey. pp 1–6
Neureiter C, Uslar M, Engel D, Lastro G (2016) A standards-based approach for domain specific modelling of smart grid system architectures. In: Proceedings of International Conference on System of Systems Engineering (SoSE) 2016. IEEE, Kongsberg, pp 1–6. Best Paper Award
Object Management Group (2014) Model Driven Architecture (MDA) MDA Guide rev. 2.0. Technical report, Object Management Group
Palensky P, Van Der Meer AA, Lopez CD, Joseph A, Pan K (2017) Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. IEEE Ind Electron Mag 11(1):34–50
Palensky P, Widl E, Elsheikh A (2013) Simulating cyber-physical energy systems: Challenges, tools and methods. IEEE Trans Syst Man Cybern Syst 44(3):318–326
Sage AP, Cuppan CD (2001) On the systems engineering and management of systems of systems and federations of systems. Inf Knowl Syst Manag 2(4):325–345
Schmidt DC (2006) Model-driven engineering. Comput-IEEE Comput Soc 39(2):25
Schütte S, Scherfke S, Tröschel M (2011) Mosaik: A framework for modular simulation of active components in smart grids. In: Smart Grid Modeling and Simulation (SGMS), 2011 IEEE First International Workshop On. IEEE, Brussels. pp 55–60
Steinbrink C, Lehnhoff S, Rohjans S, Strasser TI, Widl E, Moyo C, Lauss G, Lehfuss F, Faschang M, Palensky P, et al (2017) Simulation-based validation of smart grids–status quo and future research trends. In: International Conference on Industrial Applications of Holonic and Multi-Agent Systems. Springer, Cham, pp 171–185
SMB Smart Grid Strategic Group (SG3) (2010) IEC Smart Grid Standardization Roadmap. U.S. Department of Energy, Washington, pp. 1–136
United States Department of Energy (2010) Communications requirements of smart grid technologies. Technical report
Whittle J, Hutchinson J, Rouncefield M (2014) The state of practice in model-driven engineering. IEEE Softw 31(3):79–85
Wymore AW (1993) Model-based Systems Engineering. CRC press, Boca Raton
Yang C-H, Zhabelova G, Yang C-W, Vyatkin V (2013) Cosimulation environment for event-driven distributed controls of smart grid. IEEE Trans Ind Inf 9(3):1423–1435

## Publisher's Note