Towards a model-centric approach for developing dependable Smart Grid applications

1st Michael Fischinger
Center for Secure Energy Informatics
Salzburg University of Applied Science
Puch/Salzburg, Austria
michael.fischinger@fh-salzburg.ac.at

2nd Norbert Egger
Center for Secure Energy Informatics
Salzburg University of Applied Science
Puch/Salzburg, Austria
norbert.egger@fh-salzburg.ac.at

3rd Christoph Binder
Center for Secure Energy Informatics
Salzburg University of Applied Science
Puch/Salzburg, Austria
christoph.binder@fh-salzburg.ac.at

4rd Christian Neureiter
Center for Secure Energy Informatics
Salzburg University of Applied Science
Puch/Salzburg, Austria
christian.neureiter@fh-salzburg.ac.at

***Abstract*—The Smart Grid is the leading example when talking about complex and critical System-of-Systems (SoS). Specifically regarding the Smart Grids criticality, dependability is a central quality attribute to strive for. Combined with the desire of agility in modern development, conventional systems engineering methods reach their limits in coping with these requirements. However, approaches from model-based or model-driven engineering can reduce complexity and encourage development with rapidly changing requirements. Model-Driven Engineering (MDE) is known to be more successful in a domain specific manner. For that reason, an approach for Domain Specific Systems Engineering (DSSE) in the Smart Grid has already been specially investigated. This Model-Driven Architecture (MDA) approach especially aims the comprehensibility of complex systems. In this context, the traceability of requirements is a centrally pursued attribute. However, achieving continuing traceability between the model of a system and the concrete implementation is still an open issue. To close this gap, the present research paper introduces a Model-Centric Software Development (MCSD) solution for Smart Grid applications. Based on two exploratory case studies, the focus finally lies on the automated generation of partial implementation artifacts and the evaluation of traceability, based on dedicated functional aspects.**

***Index Terms*—Smart Grid, Dependability, SGAM, DSSE, Traceability, Model-Centric Software Development, Code Generation**

I. Introduction

In recent years, the sustainable development of future power systems has gained significant relevance. In order to reduce energy resources like fossil fuel or nuclear power, renewable energy is increasingly encouraged. Due to the granularity and the dispersion of renewable energy production facilities, the traditional power grid more and more changes towards a dynamic network, containing multiple producers and consumer, the so-called Smart Grid. Furthermore, by the increasing number of Smart Grid based enterprises and the associated autonomy of the individual applications, the grid progressively emerges to a System-of-Systems (SoS). In order to force the functional safety of the Smart Grid and its applications, factors such as the *dependability* for SoS have to be ensured. Dependability includes the attributes reliability, availability, maintainability, safety and the security-related attributes integrity and confidentiality [1]. However, since conventional systems engineering methods can fulfill these requirements in simple systems, the question is: How can dependability be addressed in the context of SoS?

Apparently, new methods for system development and management need to be defined. Based on the Smart Grid Architecture Model (SGAM) [2], the research of [3] introduced the approach for Domain Specific Systems Engineering (DSSE) in the Smart Grid. As defined by the concepts of Model-Driven Architecture (MDA), DSSE describes an optimized development process for systems based on the Smart Grid. The traceability of system requirements throughout the various layers of SGAM is a central aspect of this modeling approach. Beyond that, however, tracing these requirements down to executable software is not yet covered in this context. The stated goal of the present research is the implementation of an interface for source code generation, meeting the requirements of DSSE and the target platform FREDOSAR[1] [5]. This will be addressed with an adequate life-cycle integration of a Model-Centric Software Development (MCSD) approach. Finally, for evaluation, the achievement of traceability between model components of a system and concrete source code should be surveyed. Therewith a further step towards a sustainable method for approaching dependability in SoS should be made.

The remainder of this paper is structured as follows: Section II gives a more detailed overview of related work and the background of DSSE for Smart Grid applications. In Section III the approach is described in detail and the selected research methodology is depicted shortly. After that, the actual implementation of several case studies is demonstrated in Section IV. Finally, the chosen case studies are evaluated in Section V and the paper is summarized and concluded in Section VI.

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

II. Related Work

This section gives an overview of the state-of-the-art and the related work. Since this paper especially targets dependability in complex SoS, first, some system-related terms have to be introduced.

A. Classification of Systems

Basically, systems can be distinguished based on the attributes dynamic and alterability as well as diversity, variety and scale. A small number of elements with static interconnections is defined as *simple* system. Increasing the number of elements and including modifiable interconnections, results in a *complicated* system. Finally, a system with a large number of changing elements and a dynamical interaction behavior between those elements is classified as a *complex* system [6]. However, if the complex system's individual participants also have an autonomous character, the term "System-of-Systems (SoS)" is suggested to be used. Maier [7], Sage and Cuppan [8] and DeLaurentis [9] furthermore elaborated a number of traits describing SoS. It includes operational independence, geographic distribution, evolutionary behavior, emergent behavior, independent system networks, heterogeneity and trans-domain. Hence, the initial idea of the Smart Grid finally ends in a SoS, including infrastructure, functionality, services and interfaces, which can be applicable for a broad range of enterprises.

However, due to the complexity and criticality of the Smart Grid new challenges for risk management arise. Managing complexity, maintaining consistency and assuring traceability during system development are the key tasks for minimizing potential faults. This is where conventional systems engineering methods get to their limits. Although, system models can provide crucial support.

B. Model Support

For the mastery of the aforementioned systems engineering challenges Wymore [10] introduced the Model-Based Systems Engineering (MBSE) paradigm. Generally, Model-Based Engineering (MBE) refers to a process in which software models play an important role without being the key artifacts for development. As a consequense of the changing requirements of systems engineering the concepts of MBE have been evolving over time. Madni and Sievers [11] give a timely view into MBSE and its motivation, the current status and needed advances.

In contrast to MBE, Model-Driven Engineering (MDE) and Model-Driven Development (MDD) use the descriptive models as primary artifacts for system and software development. Model-Driven Architecture (MDA) is a more specialized concept of MDD, which refers to the standards of the Object Management Group (OMG). Another slightly modified approach is Model-Centric Software Development (MCSD), which especially focuses on the automated generation of partial implementation artifacts dependent on the aspects of interest [12]. Independent from these gradations, model-supported approaches are particularly applicable for expressing domain concepts and for reducing complexity of developing. MDE furthermore addresses development with rapidly changing requirements and prevents from many errors early in the engineering life-cycle [13]. The research of Whittle, Hutchinson and Rouncefield [14] reports on a study that surveys the state of practice in MDE. It especially highlights that domain specific concepts are more successful than general approaches. Domain Specific Systems Engineering (DSSE) is therefore the suggested solution for MDA in the Smart Grid.

C. Domain Specific Systems Engineering in the Smart Grid

The launch of the Smart Grid Architecture Model (SGAM) by the European Standardization Mandate M/490 was one of the biggest achievements in the era of Smart Grid engineering [2]. An SGAM-based model is a virtual representation of a Smart Grid system or a specific use case in this domain. As depicted in Figure 1, it includes domain-specific viewpoints, giving space for the needs of various stakeholders and their requirements. The SGAM is structured three-dimensionally, including *Domains* and *Zones* as well as an *Interoperability Dimension*. Every element of a Smart Grid model can be clearly assigned within this 3D categorization. The Domain-axis of the model decomposes a Smart Grid system on basis of the position of an element in the electricity grid. The Zone-axis on the other hand distinguishes the element's roles with regard to automation possibilities. Finally, the *Interoperability Dimension* defines layers for *Business*, *Function*, *Information*, *Communication* and *Components*.

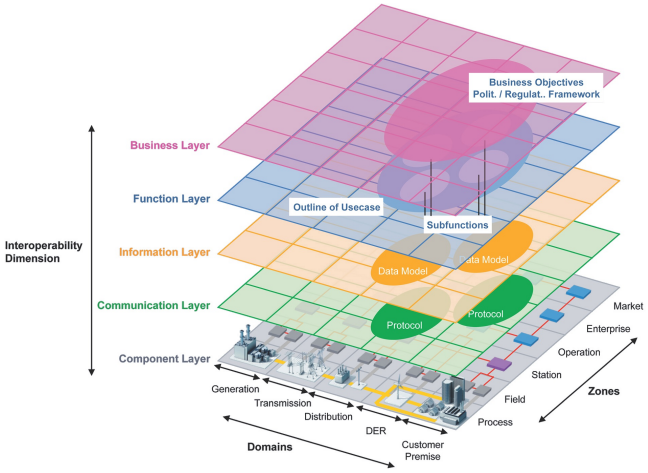

Fig. 1. The Smart Grid Architecture Model (SGAM) [2]

Beyond SGAM as a theoretic specification, the work of Neureiter [3] outlines a synopsis of several years of research for practically applicable system development in the Smart Grid. The background of this research is another issue to be pointed out in more detail. First, a MDA-based process for the Smart Grid has been launched [15]. For real world applicability, the so-called SGAM-Toolbox[2] is an essential

[2]https://sgam-toolbox.org/

side benefit of the mentioned research. The methodology as well as the tool especially focus on the requirements-based development of Smart Grid use cases. Mostly, the development is initiated by business cases, which orientate to business goals and general politically-driven restrictions. The continuing development with DSSE then requires the traceability of the stated objectives throughout all layers of SGAM. By adhering to the principles of MDA, a high level of consistency can be achieved with DSSE. To give a more detailed insight, the DSSE-based process of developing Smart Grid applications is shortly described in the following.

D. Process Model (PM)

The Process Modell (PM) of DSSE provides a guidance through the whole developing processes of Smart Grid applications. The process is divided into the three sequential engineering phases: System Analysis, System Architecture as well as Design and Development.

System Analysis Phase
Based on stakeholder needs, in this phase, a business analysis is done and a Business Use Case (BUC) is defined. Therefrom, corresponding Business Actors (BAs), Business Goals (BGs) and High Level Use Cases (HLUCs) are derived. The output of this phase is the SGAM Business Layer. In a further step, the BAs are transformed to Logical Actors (LAs). Finally, in the requirements specification, requirements are defined for every LA. Hence, another result of the System Analysis is the SGAM Function Layer. These two layers correspond to the Computational Independent Model (CIM) specified in the MDA. This is the base for further development.

System Architecture Phase
In the System Architecture Phase possible architectural solutions are developed. Thereby, a model transformation maps the LAs to physical components. This transformation aims the linkage between domain specific aspects and technology specific realizations of components, including individual attributes, behavior and interactions. This phase therefore mainly concerns the SGAM layers for Communication, Information and Components, which represent the Platform Independent Model (PIM) of MDA.

Design and Development Phase
Based on the achievements of the preceding phases, this phase deals with the concrete realization of the particular components. It is divided in two steps. First, a detailed architectural design is created. This transformation step turns the system components from black boxes to white boxes. Thereafter, as the second step, physical components and concrete software solutions are implemented. This phase reflects the Platform Specific Model (PSM) and the Platform Specific Implementation PSI of MDA.

A more detailed insight into the PM is given in [3]. In general, the DSSE approach and the SGAM-Toolbox are the base accomplishments of this continuing research. Moreover, the given approach is suggested to be extended and integrated to a comprehensive SoS Integration Toolchain for Smart Grids.

E. System-of-Systems Integration Toolchain

The SoS Integration Toolchain has been introduced by [3]. It is illustrated in Figure 2. The concepts of the toolchain try to give an appropriate solution for the sustainable development and integration of future energy systems. The proposal of the toolchain is composed of the following eight steps:

1. GIS data import:
The first step of the toolchain comprises the import of Geographic Information Systems (GIS), which includes electricity based data of energy grid analysis.

2. Use Case import:
In the second step typical Smart Grid related use cases are imported. This is done on the basis of community-based repositories, called Use Case Management Repository (UCMR).

3. Reference Architecture import:
This is followed by the import of a general Reference Architecture (e.g. the NIST Conceptual Model [16]) as a starting point for development.

4. Architecture Development:
By means of the SGAM-Toolbox the individual Architecture Development takes place. As an example, a standard-based approach for domain specific modeling of Smart Grid system architectures is accurately described in [17].

5. Model Evaluation:
After the architecture development, the introduced toolchain process suggests a Model Evaluation. Thereby, the developed model can be evaluated on basis of defined Key Performance Indicators (KPIs). In this context, an example for the assessment of privacy indicators is given in [18].

6. 3D Visualization Tool:
A further evaluation approach recommends the manual inspection of the developed architecture model. For the manual evaluation in interdisciplinary environments an adequate 3D Visualization Tool has been developed for support [19].

7. Power System Simulation:
Besides the static evaluation of architectures the analysis of the dynamic behavior of an application in the context of a SoS is recommended. Mosaik[3] is a dedicated Co-Simulation framework developed for evaluating applications for the Smart Grid [20]. A first attempt for the investigation of emergent behavior caused by electric vehicles is stated in [21]. Though, there is still a great potential for further research in this field.

8. Source Code Generation:
Finally, step eight addresses the automated Source Code Generation. To achieve traceability, the goal of the depicted approach is the transfer from a detailed SGAM-models to concrete executable software. For that reason, a general middleware framework FREDOSAR has been

[3]https://mosaik.offis.de/

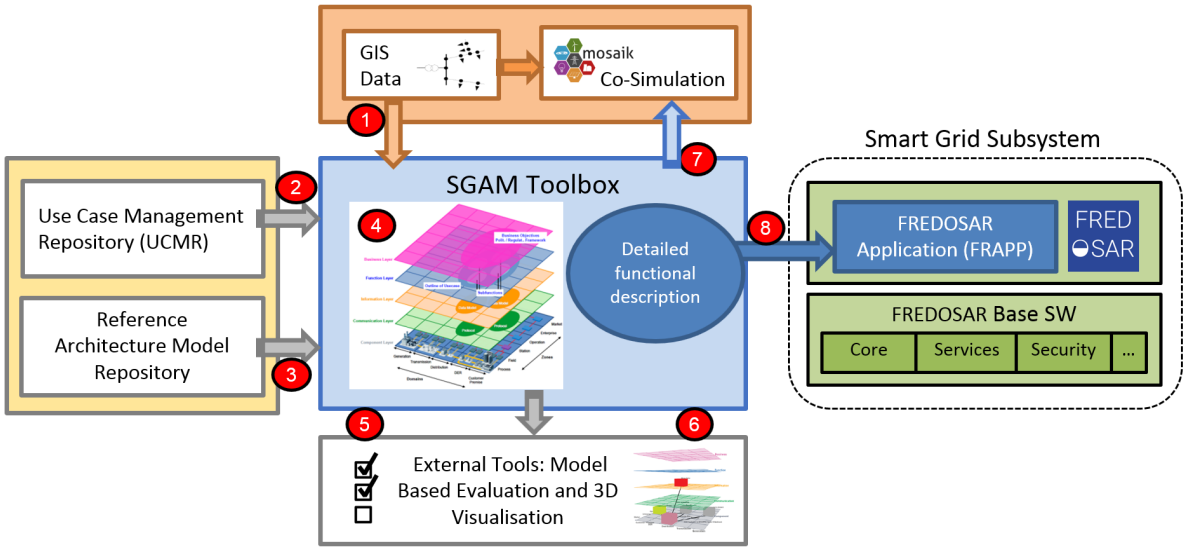

Fig. 2. System-of-Systems "Integration Toolchain" for Smart Grids

developed. To support model-centric approaches, FREDOSAR provides specially designed interfaces. However, the needed applicability has not been evaluated or proofed yet.

This overview of the SoS integration toolchain and the associated research effort indicates the relevance of this research field. Nevertheless, there are still some open issues to be implemented, evaluated and discussed. One of those is the implementation of an interface between the SGAM-Toolbox and the execution platform FREDOSAR. To make the intended interface finally evaluable, the interpretation of "traceability" has to be defined first.

F. Traceability

As a result of the aforementioned research, *traceability* of requirements has been identified as a very reasonable quality factor for dependable SoS engineering. Regarding its terminology, traceability has an origin in various domains. The work of Winkler and von Pilgrim [22] gives a general overview and treats the role of traceability in requirements engineering and MDD. Therefore, the present paper orientates to the IEEE Standard Glossary of Software Engineering Terminology [23], which defines traceability as "the degree to which a relationship can be established between two or more products of the development process". The stated relationship between these products or artifacts is built on so called *traceability links*. A traceability link implies the actual dependencies and influences that exist between artifacts [22]. The distinction between traceability in modeling and source code generation is simply that at MDD the traceability links are generated automatically, whilst the links in a system model are mostly created manually. As DSSE primarily aims the requirements engineering part, the research of this paper focuses on the traceability between the final model and the non-model artifacts generated or derived from it.

III. Approach

Reviving the preliminary findings of the state-of-the-art, it becomes apparent that there are still some open issues concerning a consistent MDE approach for Smart Grid applications. Generally, however, the scientific work presented in Section II indicates that MDE is a sustainable answer for engineering in complex systems or SoS such as the Smart Grid. The previous achievements of DSSE therefore especially highlight the usefulness of systems engineering based on MDA. A proper approach for source code generation could be a further step refining the earlier introduced SoS engineering toolchain. Hence, an adequate approach for the depicted open issues has to be devised.

Due to the dimension of projects in this research field and the unpredictability of future applications, domain or system specific requirements, the development process follows the *Agile Design Science Research Methodology (ADSRM)* [24], which is an agile methodology for application-related, scientific research and development. The ADSRM is typically built on exploratory case studies, which are driven by so-called research entry points. In this case, the question for dependability in the Smart Grid and related computing systems is the favored research entry point. Based on the objective-centered initiation of the ADSRM, the present research focuses on the implementation and evaluation of *traceability* to be the centrally aimed objective. The chosen ADSRM approach is built on two exploratory case studies, whose implementation and application will be described in more detail in Section IV.

In order to make the intended methods evaluable, the interpretation of *traceability* is particularly clarified to the following: The focus for traceability especially lies on the linking between models from the SGAM Functional Layer and the source code generated therefrom. To get a better overview of how this should be achieved, the intended development approach is described in more detail in the following.

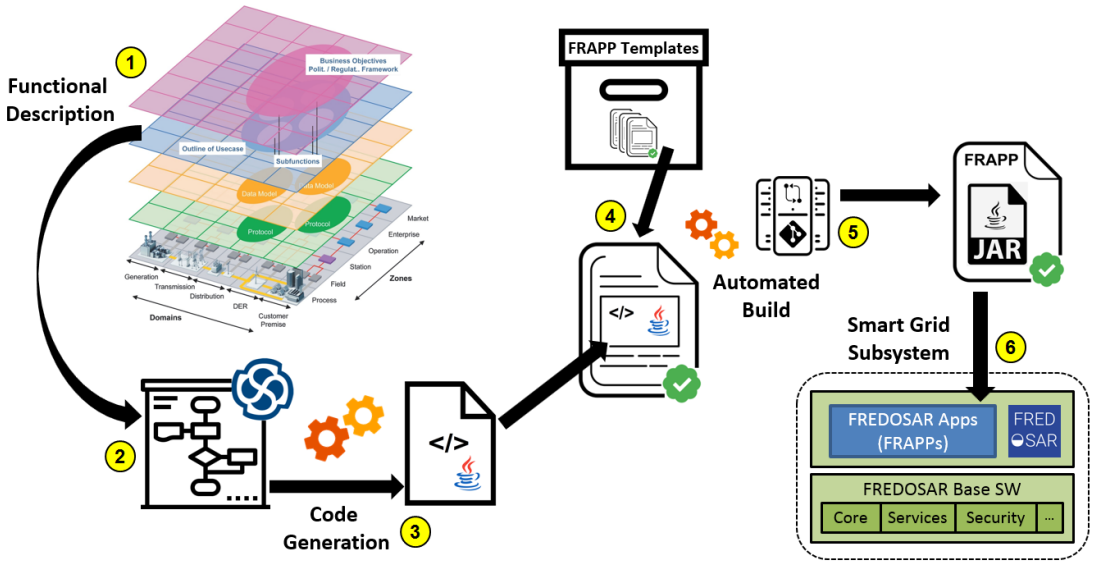

Fig. 3. SGAM-model-centric development approach

A. SGAM-Model-Centric Development

Referring to the definitions presented earlier, the chosen procedure is classified as a model-centric approach based on SGAM-models. Thereby, the automated development especially focuses on deliberately chosen model artifacts. The model-centric approach allows a quick repetition between the problem definition and the implementation of the aspects of interest [12]. Figure 3 depicts the SGAM-model-centric development approach. It includes the following steps:

1. SGAM-Model:
The desired interface should be designed to support the previous achievements of DSSE, including the SGAM-Toolbox as well as the FREDOSAR framework. The SGAM-Toolbox is currently available for the Enterprise Architect[4] (EA) software modeling tool. Hence, the starting point for the MCSD procedure is a consistent SGAM-based model.

2. Interesting Artifacts:
Commonly, the development of a Smart Grid application is initiated by an upcoming business case. Besides to politically forced general requirements, the business case objectives are the main drivers of the system development. In DSSE the system architecture is then developed throughout all SGAM layers, including the Function Layer. Therefrom, some detailed *Functional Descriptions* can be retrieved. This functional descriptions are represented by an Activity Diagram. This diagram includes those functional aspects, that are estimated as potential risks for a safe applicational behavior, primarily in the applications's role in a SoS.

3. Code Generation:
Based on these interesting model artifacts, the source code generator of the EA is used to generate partial implementation artifacts (*Code Generation*).

4. FREDOSAR Template:
The resulting code from the diagram-based code generation is actually not applicable at this point of time, as the derived artifacts are not suitable for any target platforms. To solve this problem, the development approach further applies accordingly prepared and verified code templates. By implicating all relevant (service) dependencies, the applied template perfectly meets the requirements of the FREDOSAR target platform. Basically, the same approach is suggested to be realized for other platforms.

5. Automated Build:
The resulting code artifacts are then centrally managed, versioned and automatically built on a dedicated Gitlab[5] server. As the application's integrity is guaranteed by a signature, the traceability can also be guaranteed for the build and deployment process.

6. FREDOSAR Deployment:
The result of the automated build is an executable FREDOSAR application (FRAPP). From a central repository[6], this FRAPP can finally be deployed to any hardware, which runs the FREDOSAR framework properly.

Summarizing, the intended MCSD approach tries to give an adequate answer to the identified requirements of the missing piece to make up the SoS integration toolchain for Smart Grids. The associated development steps of this model-centric solution, as well as the applicability of the intended approach will be treated in the following section.

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

[5]https://about.gitlab.com/

[6]http://repo.fredosar.org/applications/

IV. Implementation and Application

This section gives an overview of the implemented interface for MCSD. Oriented to the ADSRM, the earlier described approach is applied on the Smart Grid demonstration example *Flexible Loads*, including the case studies *Electric Vehicle Charging (EVC)* and *Variable Renewable Energy Tariffs (VRETs)*. Briefly outlined, in this flexibilization approach, customers allow the energy provider to control their loads. E.g. when the grid reaches a peak, the energy provider can add loads to compensate overproduction. Thereby, individual *Electric Vehicles (EVs)* of a fleet, which are connected to the grid, could act as a flexible energy storage. Compared to lazy power plants, which take time to start up, this energy is available quickly. For the given example, the SGAM Business Layer contains two BAs, the *Distribution System Operator (DSO)* and the *Private Customer (PC)*. VRETs are then calculated and offered by the DSO, which is responsible for the stability of the energy grid and also for pricing in these case studies. EVC, which indirectly provides flexibility to the DSO, is represented by the PC.

Oriented to the process model introduced in Section II, the use cases of these two BAs are traced to LAs and then realized in HLUCs. Using the example of the given case studies, the HLUC ”consume cheap energy” is implemented on the PC side and the HLUC ”grid stability” is implemented on the DSO side. To keep the example case studies of this paper as simple as possible, a very low level of detail has been chosen for the DSSE process model. The mentioned illustrations should primarily clarify, that in DSSE there is a trace between the BAs, the HLUCs and the resulting functions.

For modeling the aforementioned case studies, the SGAM-Toolbox has been used. To generate code from the SGAM-based model, it has to include UML classes and UML behavior diagrams. The UML classes consist of attributes and operations. Attributes are class variables and operations are methods, which execute functionality or implement algorithms. Each operation has its defined behavior. The behavior represents either written code blocks or a linked element. Linked elements are diagrams, which are able to generate source code from. EA generates source code from Activity Diagrams, Sequence Diagrams and State Diagrams. For the given case studies Activity Diagrams have been chosen as these basis linked elements.

The target platform FREDOSAR is linked as depending artifact in the model. The code generator requires this dependency to take care of the middleware specific design. This can be done by either adopting the code generator or by using templates. Compared to a template solution, adopting the code generator is more expensive. Therefore, a skeleton FRAPP template has been developed. This template has an adequate structure and resolves the dependencies. To hook in the generated code, it also includes empty operations with markers. Before modeling the classes, the template has been imported and reverse engineered as UML class. The in code markers are connected to the behavior diagrams through operations. At this point, the preparation work for the code generation is done. The following shows, how the individual case studies EVC and VRETs have been implemented in detail.

A. Case Study 1: Electric Vehicle Charging

Customers have different preferences, how to charge their EVs. It can depend on the price and also on the usage of the EV. Should it be charged over night or during the day? What's the energy price for charging? How is the algorithm designed to charge the car? Does it take care of personal preferences?

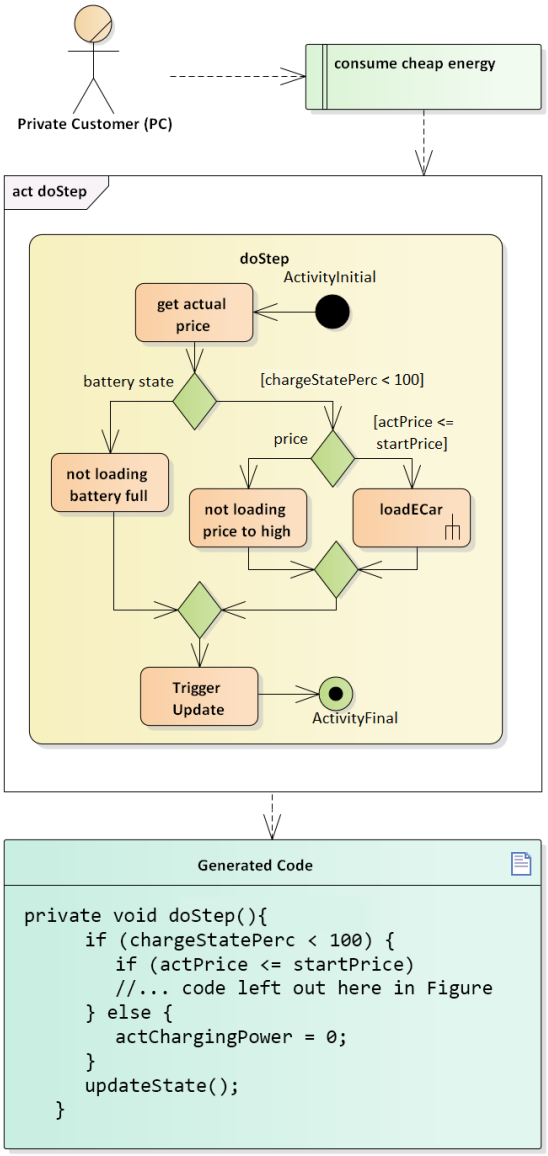

Fig. 4. Electric Vehicle Charging modeled as Activity Diagram

For this case study, six virtual neighbour households are equipped with an EV and an EV controller. The combination EV, EV controller and household is represented by the PC in the model. A general behavior of the PC is to consume energy. Furthermore, when the EV is connected to the grid and at this

time not needed, PCs can provide flexibility. Therefore, the stored energy in the battery can be supplied to the grid.

Following the DSSE approach, the BUC of the PC is traced to LAs and substantiated with HLUCs. A HLUC for the PC is to ”consume cheap energy”. As shown in Figure 4, this HLUC is associated to the PC, which represents the customer. In general, this figure outlines the traceability of the requirements from the SGAM Business Layer to the Function Layer, through an Activity Diagram. A detailed Activity Diagram is derived from the HLUC of the PC. Therein, the charging algorithm *doStep* is modeled. It charges the EV when the electricity price falls below a defined threshold and takes care of the current battery state and overcharging. The *doStep* algorithm is connected as an empty operation via the already mentioned code marker. The result is the UML class *ModelBasedECarImpl*. Finally, the source code for the EV charging application can be generated out of this Class Diagram. The link between the model and the code is established through defining the *doStep* as linked behavior operation in the model. This link is crucial. It closes the gap between the model and the source code. During the code generation, the linked behavior is evaluated and the Activity Diagram is translated to source code. A partial artifact of the resulting source code is also shown in Figure 4 at the bottom.

The connection of all elements through different layers guarantees traceability. The HLUC ”consume cheap energy” can be traced down to the algorithm description and through the linked behavior down to the generated source code line. If the HLUC changes, all affected model elements and also the source code lines are known.

The second Case Study about VRETs is similar, but from a different actor perspective.

B. Case Study 2: Variable Renewable Energy Tariff

Energy is produced in different power plants and also bought at the spot market. Taking renewable energy sources into account, the energy in the grid is volatile. To smoothen peaks, the DSO varies the electricity price depending on the available energy in the grid. This price is calculated on a regular basis. The model for this case study consists of similar elements as the previous case study. It has the DSO as BA, the HLUC ”grid stability”, a detailed functional Activity Diagram and the resulting generated source code. All these elements are shown in Figure 5. To keep focus, the algorithm and the generated code part is cherry picked. The price calculation algorithm is modeled as an Activity Diagram. For an easier understanding, the complexity of this algorithm is reduced to a minimum. First, a function to retrieve the available energy in the grid is called. If there is an overproduction, the price is decreased. If there is an energy demand, the price is increased. In the application, the new price is respectively published to the PCs via a FREDOSAR communication service. As modeled in the PCs Activity Diagram depicted in Figure 4, the PC decides either to charge or not to charge, when a defined price threshold is met. The resulting code after generating the DSOs functionality is completely listed in Figure 5 at the bottom. No manual code writing is done, everything is automatically generated based on the model.

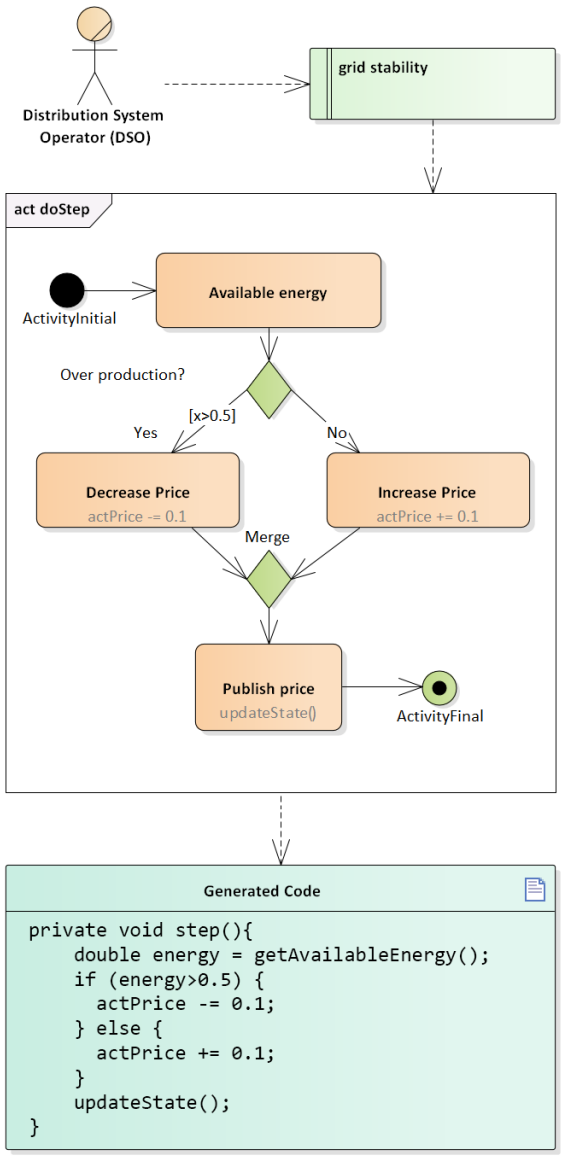

Fig. 5. Simple Price Calculation Algorithm modeled as Activity Diagram

Further insights into the code generation, the automated build and the deployment are given the following.

C. Code Generation, Build, Deployment and Execution

In summary, using EA code can be generated out of the box, if the model meets the requirements of UML classes and linked behaviors. The case study project is configured to produce Java code meeting all FREDOSAR and OSGi dependencies. A more detailed description of the individual steps of the code generation in EA can be found within the project documentation[7].

[7]https://ressel.fh-salzburg.ac.at/FREDOSAR-Frapps/model-based-applications/documentation

The next step after generating the code, is the software build process. It is triggered after the generation and uses Apache Maven[8] to build Java OSGi Bundles, which are conform with the FREDOSAR conventions. FREDOSAR then is able to install and run these bundles straightforward.

For the production environment, FREDOSAR has been installed on a physical real world miniature model, using Raspberry Pis[9], which represent the distributed hardware the SGAM Component Layer. The intend of the miniature model is to simulate a complex SoS, which includes elements like wind turbines, photovoltaics, a substation, a DSO, businesses, households with EVs and further. Each system is interfering with each other. By means of the earlier mentioned FRAPP repository, for execution, the resulting software artifacts of both case studies have been deployed to this SoS environment. The VRET application has been deployed to the DSO premise and the EV charging application to six virtual households. Finally, a web based visualization dashboard displays the real operational behavior of all participants. On this dashboard, the electricity price information and the charging states of the EVs can be traced. As this visualization actually is not part of the present paper, it is not described in more detail. Nevertheless, it offers a good opportunity for live demonstrations of the achieved research results. For the present paper, a more detailed evaluation of these results is given in the following.

V. Evaluation

For the final assessment of the research results, an observational based evaluation method is applied [25]. To recap, the following questions have already been clarified throughout this paper:

How can SoS applications be developed dependably?
SGAM, DSSE and subsequently the SoS integration toolchain deliver adequate answers for sustainable and dependable development approaches.

Which quality attributes have to be pursued?
Model consistency, completeness and the traceability of requirements have been estimated as central quality attributes for system development. Traceability has been specifically investigated in the present paper.

How must existing approaches be extended?
The traceability of requirements within a MDA conform model is fulfilled by the DSSE approach. Starting from such a consistent SGAM-based model in regard to the SoS integration toolchain, the traceability between functional system requirements and concrete source code has been identified as an open gap. Built on the two exploratory case studies, the implementation of an adequate code generation interface and the application of these case studies have been carried out in the previous section. Based on the resulting observations, this section tries to answer to what extent the expectations have been fulfilled. So the question to be answered is:

How far does the MCSD approach close the traceability gap between the SGAM-based model and the source code?
In the application-based evaluation, it finally turned out that the traceability of the relevant model-based aspects between the code and the model is given. Of course, it must be said that for now only selected areas of the model are translated directly into source code. However, assuming that due to the MDA-based DSSE approach, a very high model consistency is given and the detailed functional descriptions in the model are complete, the result of the MCSD approach is very satisfying. Considering the model as given anyway due to the DSSE approach, an automatic source code generation significantly facilitates the development process. Nevertheless, logical conclusions and decision paths in the model must be implemented in great detail. Furthermore, it has to be noted, that with the existing approach, an automated traceability between DSSE-based business requirements and the resulting source code cannot be guaranteed so far. Basically, a changing business requirement also requires a manual change of the underlying Activity Diagram, which is the base for the code generation. Therefore, a step by step implementation and an ADSRM-based evaluation of new ideas and open issues seems to be the right approach to finally find the right mixture between automation and manual engineering. The introduced MCSD approach of this research pointedly focuses to the automated generation of certain aspects of interests.

The obtained findings indicate that this approach is surely a suitable starting point for the investigation of further concepts in this research field. For the Smart Grid, the MCSD approach is furthermore recommended to be used in combination with DSSE and the SGAM-Toolbox. Due to the SoS context, however, the MCSD approach is kept as simple and general as possible. Thus, the integration of further tools and/or an extension to other domains are not excluded. After a short conclusion of this paper, therefore, the following section also illustrates possible issues of future research.

VI. Conclusion

The installation of renewable energy resources in the power grid has been leading to the formation of a complex system, the so-called Smart Grid. The smartness of the grid results of creating a basis for a wide variety of enterprises. Since these enterprises often have an autonomous character as well, the Smart Grid is also referred to as SoS. However, as the Smart Grid on the other hand also comprises survival-relevant infrastructure, new methods for dependable engineering and development in critical SoS need to be defined. The previous research of DSSE laid the foundation for a sustainable approach for MDA in the Smart Grid. In order to take advantage of the achievements of DSSE and the resulting SGAM models, this paper made it a goal to introduce an interface for automated code generation. Based on the state-of-the-art analysis, this has been estimated as an open gap in the SoS integration toolchain for Smart Grids. At the definition of the mentioned interface, the traceability of relevant functional requirements is considered as particularly important. To finally

[8]https://maven.apache.org/
[9]https://www.raspberrypi.org/

generate these so-called aspects of interest, a model-centric approach has been followed for development. To keep the development method for complex systems comprehensible, the functionality of the implemented interface has been reduced to a minimum in this first step. In terms of the dedicated traceability, this simplicity ensures that the overview is not lost. The evaluation in Section V finally shows, that these basic goals have been achieved satisfyingly. However, it also depicts great potential regarding the extensibility of the chosen approach, which could possibly be part of future research.

A. Future Work

Considering SGAM and DSSE, the effort already spent, points to the relevance of the selected research field. The achievements of the present research are further steps towards a sustainable approach developing dependable Smart Grid systems. Especially for practically integrating new systems to the Smart Grid, the output of the present paper could also be applicable combined with Co-Simulation. As part of the SoS integration toolchain, the outcome of the implemented code generator then could first be analyzed based on a power system simulation. Therewith, on the one hand, the traceability of relevant requirements could be extended throughout the Co-Simulation. On the other hand, this concept would increasingly enhance the process by automatically evaluating emergent behavior in the Smart Grid, which is recommended to be done before the system integration.

Besides to this linkage with the Co-Simulation, the interface itself could be further extended. First, the automation of the code generation could be further enhanced by implementing a SGAM-toolbox Add-in. Then the approach could be broadened to meet the demands of different target platforms. Finally, going away from functional system aspects, communication and information services of Smart Grid devices could be preconfigured based on those layers of the SGAM-model.

In addition to the mentioned automation approaches, security-related aspects should never be neglected in any case. FREDOSAR generally provides a security-aware platform to ensure trustability issues. Finally, the integrity of a software should therefore also be comprehensible right up to its connection to the SGAM-model. Hence, the integration of adequate concepts is recommended.

ACKNOWLEDGMENT

The financial support by the Federal State of Salzburg is gratefully acknowledged.

REFERENCES

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[2] CEN-CENELEC-ETSI Smart Grid Coordination Group, "Smart Grid Reference Architecture," 2012.

[3] C. Neureiter, *A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid*. Hamburg, Germany: MBSE4U - Tim Weilkiens, 2017.

[4] Object Management Group, "Model Driven Architecture (MDA) MDA Guide rev. 2.0. Technical report," Object Management Group, Tech. Rep., 2014.

[5] M. Fischinger, C. Neureiter, C. Binder, N. Egger, and M. Renoth, "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)*. Heraklion, Crete - Greece: SciTePress, 2019, in press.

[6] R. Haberfellner, O. de Weck,, E. Fricke, and S. Vössner, *Systems Engineering - Grundlagen und Anwendung*, 13th ed. Orell Füssli, 2015.

[7] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering: The Journal of the International Council on Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[8] A. P. Sage and C. D. Cuppan, "On the systems engineering and management of systems of systems and federations of systems," *Information knowledge systems management*, vol. 2, no. 4, pp. 325–345, 2001.

[9] D. DeLaurentis, "Understanding transportation as a system-of-systems design problem," in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2005, p. 123.

[10] A. W. Wymore, *Model-based systems engineering*. CRC press, 1993.

[11] A. M. Madni and M. Sievers, "Model-based systems engineering: motivation, current status, and needed advances," in *Disciplinary Convergence in Systems Engineering Research*. Springer, 2018, pp. 311–325.

[12] D. Waddington and P. Lardieri, "Model-centric software development," *COMPUTER-IEEE COMPUTER SOCIETY-*, vol. 39, no. 2, pp. 28–29, 2006.

[13] D. C. Schmidt, "Model-driven engineering," *COMPUTER-IEEE COMPUTER SOCIETY-*, vol. 39, no. 2, p. 25, 2006.

[14] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE software*, vol. 31, no. 3, pp. 79–85, 2014.

[15] C. Dänekas, C. Neureiter, S. Rohjans, M. Uslar, and D. Engel, "Towards a model-driven-architecture process for smart grid projects," in *Digital Enterprise Design & Management*, ser. Advances in Intelligent Systems and Computing, P.-J. Benghozi, D. Krob, A. Lonjon, and H. Panetto, Eds. Springer International Publishing, 2014, vol. 261, pp. 47–58.

[16] C. Greer, D. A. Wollman, D. E. Prochaska, P. A. Boynton, J. A. Mazer, C. T. Nguyen, G. J. FitzPatrick, T. L. Nelson, G. H. Koepke, A. R. Hefner Jr *et al.*, "Nist framework and roadmap for smart grid interoperability standards, release 3.0," Tech. Rep., 2014.

[17] C. Neureiter, M. Uslar, D. Engel, and G. Lastro, "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*, Kongsberg, Norway, Jun. 2016, pp. 1–6, best Paper Award.

[18] F. Knirsch, D. Engel, C. Neureiter, M. Frincu, and V. Prasanna, "Model-driven privacy assessment in the smart grid," in *Proceedings of the 1st International Conference on Information Systems Security and Privacy (ICISSP)*. Angers, France: SciTePress, Feb 2015, pp. 173–181, best Paper Award.

[19] C. Neureiter, D. Engel, J. Trefke, R. Santodomingo, S. Rohjans, and M. Uslar, "Towards consistent smart grid architecture tool support: From use cases to visualization," in *Proceedings of IEEE Innovative Smart Grid Technologies (ISGT) 2014*. Istanbul, Turkey: IEEE, Oct. 2014, pp. 1–6.

[20] S. Schütte, S. Scherfke, and M. Tröschel, "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*. Brussels, Belgium: IEEE, 2011, pp. 55–60.

[21] C. Binder, J.-A. Gross, C. Neureiter, and G. Lastro, "Investigating emergent behavior caused by electric vehicles in the smart grid using Co-Simulation," in *2019 14th Annual Conference System of Systems Engineering (SoSE)*. Anchorage, Alaska, USA: IEEE, 2019, in press.

[22] S. Winkler and J. von Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Software & Systems Modeling*, vol. 9, no. 4, pp. 529–565, 2010.

[23] IEEE, "IEEE standard glossary of software engineering terminology," *IEEE Std 610.12-1990*, 1990.

[24] K. Conboy, R. Gleasure, and E. Cullina, *New Horizons in Design Science: Broadening the Research Agenda: 10th International Conference (DESRIST)*. Dublin, Ireland: Springer International Publishing, 2015, ch. Agile Design Science Research, pp. 168–180.

[25] A. Hevner and S. Chatterjee, *Design Science Research in Information Systems - Theory and Practice*. Berlin Heidelberg: Springer, 2010.