FREDOSAR: Towards a Security-aware Open System Architecture Framework Supporting Model based Systems Engineering

Michael Fischinger, Christian Neureiter, Christoph Binder, Norbert Egger and Michael Renoth
Center for Secure Energy Informatics Salzburg, University of Applied Sciences, Urstein Süd 1, 5412 Puch/Salzburg, Austria

Keywords: Open Systems Architecture, Complex Systems, Security by Design, MBSE, Domain Specific SE, Smart Grid.

Abstract: The Smart Grid is the leading example when talking about complex and critical systems. *Model Based Systems Engineering (MBSE)* represents an appropriate approach for the mastery of this complexity. However, achieving full traceability between the model of a system and the concrete implementation is still an open issue. For this, *Model Driven Development (MDD)* is proposed. Hence, the development of a general middleware framework, which comprises a well-defined, customized architecture and provides accurately elaborated interfaces, would probably facilitate the *Model Based Approach* in a decisive manner. Thus, the present work addresses the development of a versatile architecture framework, the *FRee EDucational Open System ARchitecture (FREDOSAR)*, which also includes an open source implementation. Due to the criticality of systems like the Smart Grid, *Security by Design* is another central aspect of this architecture development process. For evaluation, the applicability of the framework is finally verified by the implementation of several case studies based on FREDOSAR, focusing on the quality attributes mentioned.

1 INTRODUCTION

In recent years, a transition of the original power grid towards the so-called Smart Grid has become apparent. This is encouraged by the expected reduction of primary energy resources like fossil fuel or nuclear power. Furthermore, technological advances offer new possibilities for the efficient use of more sustainable and environmentally-friendly sources. This leads to major changes concerning the unidirectional energy flow from centralized facilities towards its customers, resulting in a dynamic network containing multiple producers and consumers. However, since dealing with this complexity in the Smart Grid is a difficult task, new methods for describing and developing current and future energy systems need to be defined.

The first set of problems can be identified by using the classification proposed by Haberfellner et al. (2015). In more detail, the authors distinguish types of systems based on the attributes dynamic and alterability as well as diversity, variety and scale. This results in the description of four different system types: If it is only comprised of a few elements which are statically interconnected, it is defined as a simple system. However, a complicated system emerges by adding a large number of elements and connections or adding dynamical interaction behavior between those elements. By including both of these characteristics, the result is a complex system. By applying this classification scheme, the traditional power system can be considered as a complicated system whereas the Smart Grid is classified as a complex system. Going even further, the term *System-of-Systems (SoS)* is suggested to be used in order to emphasize the autonomous character of the system's individual participants (Maier, 1998).

According to these considerations, the Smart Grid is specified to be a critical infrastructure, which needs to address several requirements in order to integrate functional safety. More specifically, the requirements reliability, availability, maintainability, safety and security are summarized by the term RAMSS and are used to describe the dependability of a technical system (Avizienis et al., 2004). Since one effect resulting from the aforementioned technological advances is the possibility to build powerful network devices on the customer's premises, a special need for considering privacy and security arises. However, the concepts of *Model Based Systems Engineering (MBSE)* specifically target issues such as dealing with the upcoming complexity during the development of a SoS or delivering *Privacy & Security by Design*. Neureiter et al. (2016) and Knirsch et al. (2015) provide several

Fischinger, M., Neureiter, C., Binder, C., Egger, N. and Renoth, M.
FREDOSAR: Towards a Security-aware Open System Architecture Framework Supporting Model based Systems Engineering.
DOI: 10.5220/0007710201080115
In *Proceedings of the 8th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2019)*, pages 108-115
ISBN: 978-989-758-373-5

examples on how those issues are addressed by applying the methods introduced by MBSE for developing domain-specific system architectures. Furthermore, the approach for *Domain Specific System Engineering (DSSE)* (Neureiter, 2017) is a result of several years of research and use in international projects. The research mentioned introduced an *Integration Toolchain* to illustrate the holistic workflow, which describes an optimized development process for systems based on the Smart Grid and inherits consistent supporting methods for developing in different phases throughout the whole life-cycle of a system.

Therefore, this paper introduces the conception and development of an open systems architecture framework that enables the application of complex systems. By doing so, the special focus of the framework is the seamless integration into the DSSE approach and to deliver security by design. Therefore, the remainder of this contribution is structured as follows: In Section 2, an overview of already existing frameworks with similar objectives and open aspects in this area is given, before relevant features are extracted. Based on these features, in Section 3, the requirements for the development are specified. After that, the implementation details of the framework itself are stated in Section 4. However, to demonstrate the applicability of the implemented architecture, a PoC implementation is taken for evaluation in Section 5. Finally, in Section 6 the paper is summarized and then the conclusion is given.

2 RELATED WORK

This section gives an overview of the state-of-the-art and the related work. Basically, the idea of establishing generic software architecture frameworks is not new. Hence, existing solutions and attempts for comparable problem formulations have been analyzed and evaluated.

The *AUTomotive Open System ARchitecture (AUTOSAR)* (Fürst et al., 2009) is therefore the first architecture (framework) to be mentioned. AUTOSAR is a middleware platform for the automotive industry. The standardization of base functionality, the scalability of systems, the transferability of software to different control devices, the cooperation between various manufacturers and the development of highly dependable systems have been the major incentives for founding an open systems architecture framework in the automotive domain. There are obviously some parallels within the given problem formulations. A central aspect is therefore that the software components of AUTOSAR are generated from software models. Besides some of the non-functional requirements, this model based approach is a potential aspect, where the intended framework probably could refer to the architecture of AUTOSAR.

Drawing the attention from the approaches of AUTOSAR in automotive engineering to the Smart Grid and other domains leads to a number of frameworks and architectures that could perhaps be taken as base for further implementations. The most famous solution are probably *OpenHAB* and *Eclipse SmartHome*, shortly described by Kreuzer (2014). Basically, they address the IoT sector and Smart Home. The primary goal is the consolidation of various technologies by adding a framework that acts as an abstraction layer. The implementation is based on OSGi[1]. The research of Pichler et al. (2015) therefore additionally compares potential OSGi based solutions for *Customer Energy Management Systems (CEMS)*. Especially for security- and privacy-related requirements Pichler et al. show, that there is still a great need to catch up.

Summing up the lessons learned from the state-of-the-art analysis, it turns out that there are already a couple of approaches and frameworks addressing modular architectures for IoT. Nevertheless, there are a number of missing features regarding to a model based approach and the abovementioned toolchain integration. Under these circumstances, it makes sense to pursue the implementation of a new, customized framework, the *FRee EDucational Open System ARchitecture (FREDOSAR)*. FREDOSAR for the main part addresses the following features:

Feature **Security by Design:** The work of Pichler et al. (2015) points out, that the most popular comparable frameworks primarily focus on operational functionality. Topics like security and privacy are often disregarded or regarded marginally. For this very reason, the present work also tries to draw its attention to applicable security-related research issues, to make the framework also prepared for trustable environments and applications.

Feature **Model Based Approach:** As FREDOSAR is generally addressing applications in complex, heterogeneous and distributed systems, the framework should also be designed as a feasible tool to obtain full traceability according to the stated approaches in MBSE. The architecture design is guided by the systems engineering approach proposed by Neureiter (2017). *Model Driven Development (MDD)* in a well-defined environment could close the traceability gap between the model of a system and the concrete implementation.

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

How to Address Security by Design?

A central architecture decision regarding the security concept is to follow a layered architecture including appropriate security levels. Thus, there are differing permissions and restrictions which are bound to the FREDOSAR layers *Core*, *Management*, *Service* and *Application*. Another aspect of the security concept is a proper integration of common security practices in regard to integrity and confidentiality as well as making use of encryption mechanisms and protocols.

How to Address the Model Based Approach?

Due to the absence of an adequate implementation platform Neureiter (2017) describes the model based source code generation as ”rather visionary”. Based on the *SGAM Toolbox* and the related toolchain integration, it is described as one of the research issues to be further attended to. For that very reason a part of the toolchain is particularly emphasized in Figure 1. The goal of the depicted **Model Based Approach** therefore is the transfer from a detailed functional description in an architecture model to concrete executable software. Regarding this key challenge, the framework's architecture has to be designed in a modular and precisely defined manner, offering clearly structured interfaces, that can later be traced to detailed architecture model descriptions across the layers of SGAM.

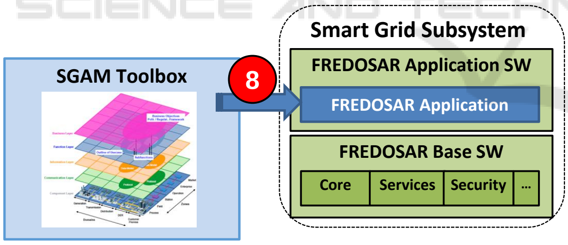

Figure 1: Toolchain Integration.

Basically, the **Model Based Approach** and the link to the toolchain is an attempt to enhance the general awareness concerning the relevance of MBSE, especially for SoS. Hence, the approach is applicable for various domains. Applications in other domains follow the same procedure using other domain specific languages and toolboxes like the *RAMI 4.0 Toolbox* (described by Binder et al. (2019)) for industrial purposes.

After formulating the most relevant features Section 3 pays attention to the requirements engineering, followed by the concrete implementation of FREDOSAR in Section 4.

3 REQUIREMENTS

The requirements engineering process as well as the concrete elaborated requirements are based on the intended features. The requirements engineering process included software architects, software developers, domain experts and stakeholders as well as users.

3.1 Non-functional Requirements

The non-functional requirements have been selected from a number of common quality attributes. The extracted requirements are listed in Table 1. Finally, these requirements have been considered as highly relevant to address the chosen features.

Table 1: Non-Functional System Requirements.

No.	Requirement
1.1	**Modularity**: Separation of Concerns and Loose Coupling should be fulfilled by implementing well-defined interfaces.
1.2	**Extensibility**: The software should provide the ability to be simply extended without influencing the running system.
1.3	**Autonomy**: The software components on the application layer should be kept completely stand-alone according to dependencies between each other.
1.4	**Updatability**: The framework should provide the ability to update any of the systems software components on the fly.
1.5	**Dependability**: The software should be kept reliable and stable and should automatically react to failures (e.g. by restarting).
1.6	**Configurability**: The system should provide an interface to centrally configure applications and/or services.

3.2 Security-related Requirements

This section addresses security-related requirements that are relevant for service-oriented architectures (SOA) in critical domains. Based on Hafner et al. (2008), *Confidentiality* and *Integrity* have been taken as hypernym for a chosen set of security requirements. The specified security-related requirements that have for now been estimated for FREDOSAR are listed in table 2. As a basic principle, the systems architecture should strive towards a secure solution. Thus, the chosen process for (architecture) development includes an ongoing security requirements engineering, approaching upcoming changes regarding to technology, security or privacy aspects.

Table 2: Non-Functional Security-related Requirements.

No.	Requirement
2.1	**Integrity**: The framework should provide the ability to verify the integrity of any application to be executed.
2.2	**Confidentiality**: The framework should restrict the access to internal communication channels to a minimum. Each software component should only be able to receive content that directly concerns it.
2.3	**Access Control**: The framework should confine the service access of applications to a minimum, considering their defined rights.
2.4	**Standard Encryption Protocols Support**: The framework should support the standard encryption protocols HTTPS and TLS.

3.3 Functional Requirements

In the course of an extensive functional requirements analysis, it turned out that keeping the functional requirements in the framework's base software as general and clear as possible is necessary to achieve a comprehensive, well-defined architecture. The more use case specific requirements will therefore be dedicated to the corresponding application software. The finally elaborated functional requirements are stated in Table 3. The clarity and abstraction of the chosen requirements is a fundamental concept to support the **Model Based Approach**.

Table 3: General Functional Requirements.

No.	Requirement
3.1	**Internal Communication Interface**: The software should provide a simple interface for internal communication between applications and/or other system components.
3.2	**Defined Communication Service Interfaces**: The software should provide a general and simple interface to implement any external technology or service (I/O) considering the concomitant communication protocol.
3.3	**Defined Communication Data Structures**: The software should define well-formed data structures for communication between internal components or applications as well as communication to external services.
3.4	**Technology- and Address Management**: The system should offer the functionality for centrally managing multiple communication technologies and the associated address data.

4 FREDOSAR

This section gives an overview of the concrete architecture of FREDOSAR as well as an insight into some implementation details. The results shown are selective aspects of the associated development. As the progressing work for FREDOSAR does not only include the architecture development but also an Open Source implementation[2], this paper also comprises the most relevant implementation details and technology decisions. Based on incentives from the state-of-the-art for middleware platforms, a SOA has been chosen, using Java and OSGi for implementation. Apache Felix[3] in version 5.4.0 and higher is suggested for execution.

4.1 The *Core* System

Figure 2 illustrates the component model of the architecture. FREDOSAR comprises a layer for *Core*, *Management* and *Service* components in the basic software and a layer for *Application* components in the application layer.

The major and most important part of FREDOSAR is called the *Core* system. In short, the *Core* defines all relevant interfaces, the management of services, communication, typical data structures and further central functionality. A visualization of the *Core* can be seen in the green area of Figure 2. It will be described below.

Interfaces and Types Definition

In the FREDOSAR core the standard interfaces for all further components or services are defined and provided by the *Interfaces* bundle. For standard data structures the *Types* bundle has been established. It centrally handles and provides previously defined *Data Transfer Objects (DTO)*. This architectural decision is a crucial step to reach the maximum level of autonomy. According to the external package dependencies, all other components are therefore only dependent on these two bundles.

The FREDOSAR Whiteboard Pattern

One of the used patterns for FREDOSAR is a slightly modified implementation of the *Whiteboard Pattern* (OSGi Alliance, 2004). In the FREDOSAR implementation, two *OSGi Service Trackers* have been coupled. This concept guarantees control over service

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

[3] http://felix.apache.org/

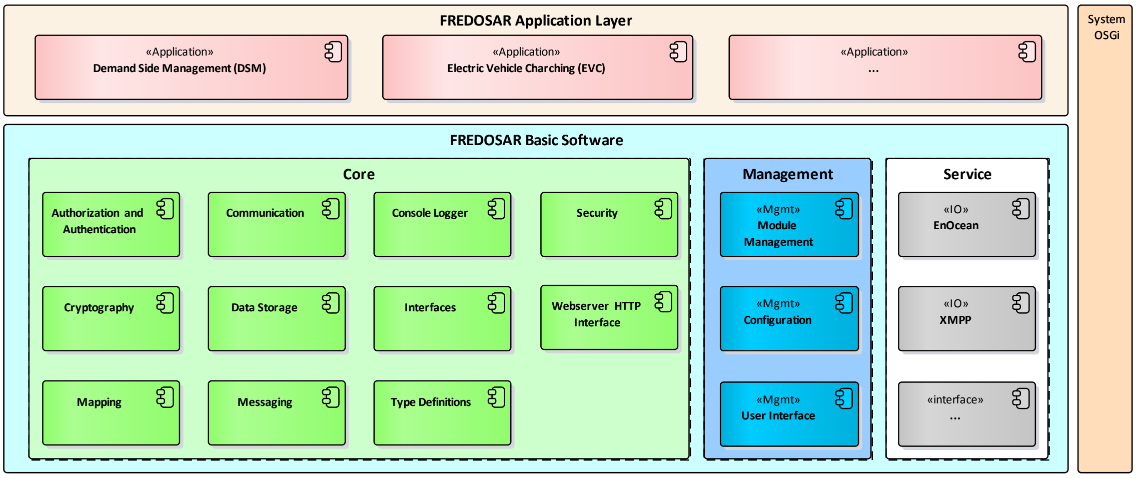

Figure 2: FREDOSAR Layered Architecture Model.

restrictions during the service injection. The whiteboard pattern of FREDOSAR further comprises a special convention. If a service or application wants to access any of the core services, it has to implement the corresponding *aware* interface. All in all the whiteboard pattern has been customized to meet the needs and requirements of FREDOSAR. This means, that regardless of whether using service injection or extending the framework's functionality with further services, this pattern must be applied to meet the FREDOSAR convention.

Internal Communication

The component for internal communication or Inter Process Communication (IPC) is realized in the *Messaging* bundle, which is a central unit of the core system. The bundle comprises two essential service interfaces. The first, the *MessageConsumer* provides the ability for applications or other bundles to register for specific *Topics*, which is comparable with a queue or a channel. The implemented patterns also imply a solution for a messaging topic restriction for incoming messages. To send an internal message to another application or component, the second service interface, the *MessagingService*, can be applied. Having regard to the restriction for outgoing messages, the sending applications rights according to the outgoing topic have to be checked before sending the message. This is done by the OSGi Service Tracker using a Proxy Pattern for the management of the individual rights of applications.

External Communication

The interfaces for external communication have been realized in the *Communication* bundle. The services for the technologies are made available via service injection and are delivered in a map, including all permitted service implementations. They can be accessed by making an application bundle *CommunicationServiceAware*. Each of the supplied services comprises a method for sending messages, which has to be implemented individually for any declared *CommunicationService*. The sending method of the interface expects a specific data structure according to the message to be sent (EventDTO) as well as a data transfer object including the receiver information (AddressDTO). In this respect address management is also part of the *Communication* bundle. It is implemented as *AddressService* and can be accessed via *AddressServiceAware*. The service includes functionality for managing addresses and the associated technologies.

As most external communication services are in fact bidirectional, a topic for incoming messages has to be established. Therefore, the internal messaging service is consulted. Each external communication service in consequence implements the *MessagingServiceAware* interface and possesses a fixed topic for incoming messages, which is a reserved string based on the underlying technology. This pattern has to be complied with any new external communication service implementation.

Besides to the above in detail described functionality, patterns and services, further implemented general core services have to be mentioned in short. In particular, FREDOSAR also includes a *Console Logger*, a *Web Server* implementation, a *Data Store* service interface offering data storage solutions and a *Mapping* service for the mapping of Java objects to defined strings and vice versa. Further aspects to be discussed are the management an the service layer.

4.2 The *Management* Layer

The *Management* layer comprises extension functionality for the FREDOSAR *Core*. More precisely it offers services for configuration, observation and module management. Therefore, the management layer as a whole addresses the configurability.

The *Module Management* bundle monitors the life cycles of all FREDOSAR bundles and provides further services or applications to be downloaded from a central repository. To furthermore achieve an adequate solution for bundle configuration, the OSGi ConfigAdmin service has been extended in the *Configuration* bundle. As a result, any service or application bundle can simply be extended with the *ConfiguredService* interface. In combination with the *ManagedService* from OSGi it can be used for preconfiguring service properties as well es updating service properties during runtime.

4.3 The *Service* Layer

The *Service* layer of FREDOSAR primarily applies to extensibility and autonomy. From the architecture viewpoint, the service layer follows a well-defined process for service extension. As a result, new services always have to implement existing core interfaces to reach autonomy. This is the major concept of FREDOSAR services.

The current implementation includes services for *I/O*, *DataStore*, *Mapping* and *Messaging*, which can be seen as extensions and/or alternatives to existing core services. As FREDOSAR is declared as an open systems architecture, adding new technology-specific services is also a major part of the ongoing implementation process. At this point of time the service layer already contains *I/O* services for RabbitMQ, XMPP, EnOcean, OPC UA, Serial, P2P, KNX, GPIO, Bluetooth and HTTP, *DataStore* services for JSONFile and MySQL, a GSON *Mapping* service and a *Messaging* service based on RabbitMQ.

4.4 Security-related Concepts

Regarding the **Security by Design** feature and the related requirements, the FREDOSAR core system has been designed with a complying architecture, covering several pivotal concepts and services, including authentication and authorization, a bundle signing and verifying procedure, a component for general service restriction management, a concept for FREDOSAR communication service and channel or topic restriction, and finally a service dealing with basic cryptographic issues.

Bundle Signing and Verification

The bundle signing and verification process is a central aspect of the FREDOSAR security concept. Basically OSGi provides a Conditional Permission Admin (CPA). It is applied within the FREDOSAR core *Security* bundle and provides the ability to define *Rules* for any service or application bundle in the OSGi execution environment. As a concrete example, the *Security* bundle of FREDOSAR implements a *Rule* for *Core* bundles. It grants comprehensive access rights for bundles that have been signed by the FREDOSAR core certificate. This can be done for any layer or application.

The fundamental element of this approach is the *OSGi Delegation Model* shown in figure 3. In this approach the platform *Operator* assigns specific rights to a chosen *Enterprise*. The rights can be bound to a certificate. As a result, the *Developer* is able to use all services that are needed for a certain development proposal, still having restrictions to services that are not required.

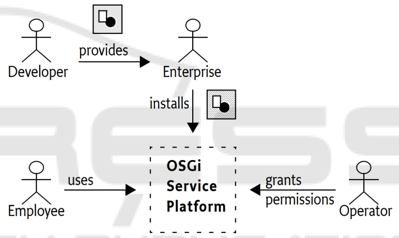

Figure 3: OSGi Delegation Model (OSGi Alliance, 2018).

To accomplish the integrity check, FREDOSAR performs a signature check by default. For this reason, a Certification Authority (CA) has been established at the highest level of a Chain of Trust. The CA is able to sign underlying certificates, to finally prove full integrity for a service in a specific FREDOSAR layer or for applications of any enterprise.

Authentication and Authorization

Based on the framework's deep focus on security, a dedicated bundle for authentication and authorization has been implemented. A service for both is part of the *Auth* bundle. Authentication for now has been applied for basic user authentication in the HTTP web server implementation. However, the user authentication and authorization service is designed to be simple and straightforward. Hence, it can be used for any user interaction purpose and simply be extended to a token based authentication approach for external service interaction. Regarding the internal communication restrictions, the service also implements an interface for module authorization.

Communication Service and Topic Restriction

As already mentioned above, FREDOSAR follows a strictly defined layered architecture with layer-specific security restrictions. Thus, no application has access rights to do a direct service injection. The injection is handled by means of the Whiteboard Pattern explained earlier. This is where the *ModuleAuthorizationService* can be integrated to the *ServiceAwareTracker* of the chosen service to be finally able to restrain the service injection if needed. This approach has been implemented for communication service restriction in FREDOSAR. In the concrete implementation, the *ModuleAuthorizationService* holds a list of authorized applications and the corresponding communication service permissions. To eventually achieve the rights to use the requested communication service, the user has to approve the applications request via the user interface. This is where the application will be authorized for service injection.

The same procedure in a slightly modified way is applied for the internal messaging topic restriction. Thereby, the consuming application requests a specific topic to read from and/or to write to. The approval for the requested topics is again done by the user. With the aid of the *ModuleAuthorizationService*, the core *Messaging* service then handles the permissions of all applications and services. If an application is not permitted to access a topic, the affected message will not be further processed or forwarded.

Cryptography

The *CryptoService* is a core service that provides standard cryptographic methods and security-related procedures like certificate and key store handling. To fulfill the standard encryption protocols, the *Cryptography* bundle also manages secure sessions for HTTPS and TLS. Furthermore, it offers hashing functionality as well as RSA and AES encryption with appropriate key lengths and algorithms. The *CryptoService* API is generally designed with clear and simple interfaces, making the *Cryptography* bundle a powerful core service for security- and privacy-aware applications.

5 DEMAND SIDE MANAGEMENT

For evaluation, the present work refers to a general assessment approach based on a case study and the consequential appraisal of architects, developers, stakeholders and users. During the development process of FREDOSAR, various case studies have been established and implemented as PoC FREDOSAR applications (*FRAPPS*[4]). Smarthome, Ambient Assisted Living, Smart Metering, Demand Side Management and Electric Vehicle Charging are some of the energy domain related *FRAPPS*. For the framework (architecture) evaluation, Demand Side Management (DSM) has been taken into consideration.

One of the major problems of Smart Grids is the unpredictability of the generated energy from renewable energy systems. The goal of DSM therefore is to achieve power grid stability by controlling huge electrical loads (e.g. heat pumps). DSM has been implemented as a PoC FRAPP, including a number of ”real” consumers in a demo world model.

The primary focus during the DSM development was the verification of the general functional requirements of FREDOSAR. Therefore, requirement 3.2 has been tested by making use of two external *CommunicationServices*, GPIO for LEDs simulating the electrical loads and RabbitMQ for the communication between the *Distribution System Operator (DSO)* and the CEMS. External communication in FREDOSAR is usually accompanied by technology- and address management (requirement 3.4). Therefore, the *AddressService* has been implemented and tested. Another aspect of DSM is the individual load control. In this context the DSO sends control commands to the CEMS concerned. This is where the internal *Messaging* has been deployed and requirement 3.1 has been verified. The whole communication procedure has to be built upon clear data structures (requirement 3.3). Hence, a DTO for the control commands has been constructed and tested regarding its simplicity of integration. A final issue was that the *Web Server* service was used to run a RESTful JSON API for the frontend. The whole execution was automatically logged by the *Logging* module.

Another important aspect about the DSM development was the verification of security-related requirements. The goal was to develop DSM in a security-aware manner using the given functionality and the suggested practice of FREDOSAR. Following the OSGi delegation model, a certificate of the enterprise DSM was signed by the FREDOSAR CA, giving DSM related applications access to the needed services and allowing them to be executed in the FREDOSAR execution environment. Both, the integrity check (requirement 2.1) and the system function restriction (requirement 2.3) turned out to be fulfilled, as the execution was denied in cases where the application tried to circumvent one of the restrictions. The requirements 2.2 and 2.3 furthermore are a fundamental demand for any FREDOSAR application. Thus, the entitlement to confidentiality has been im-

[4] https://ressel.fh-salzburg.ac.at/FREDOSAR-Frapps

plemented using pre-defined, dedicated communication topics for DSM applications (requirement 2.2). Besides, the communication service restriction (requirement 2.3) has been added to the DSM application as prescribed by the FREDOSAR convention. In terms of requirement 2.4 and the standard encryption protocols support, the abovementioned RESTful JSON API fulfills the conditions of HTTPS, and the RabbitMQ communication implemented meets the requirements of TLS. Hence, the whole range of the security-related requirements has been implemented, verified and validated regarding its applicability.

Regarding the evaluation process carried out, it turned out that **Security by Design** led to a quite complex system architecture and a sophisticated implementation. Even though most of the security mechanisms have been simplified as effectively as possible, the additional complexity could act as a deterrent for potential developers, especially in cases where security plays a secondary role. However, because of the general criticality of applications in the energy domain, security and privacy will definitely remain relevant for FREDOSAR and its further research topics.

6 CONCLUSION AND FUTURE WORK

The present work has made it a goal to accomplish a trustable, versatile OSGi-based open system architecture for application in complex systems and SoS. Based on previous research in the fields of security in Smart Grids and MBSE and the analysis of related work, a number of features and requirements have been established for the architecture development. The primary focus was on **Security by Design** and the **Model Based Approach**. The results of the architecture and some insights into the development have been presented.

Basically, the continuing research will focus on further attempts regarding MBSE and the corresponding MDD process for the toolchain integration. In the first step an automated process for code generation in the style of a FREDOSAR application will be integrated. Combined with a co-simulation approach for complex systems, thereby a sustainable, automated life-cycle management process for Smart Grid systems and other SoS should be achieved. Another security-related aspect will be the integration of End-to-End security for applications including a Hardware Security Module (HSM) for encryption. FREDOSAR therefore will finally serve as a base framework to do these further steps.

ACKNOWLEDGEMENTS

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

REFERENCES

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33.

Binder, C., Neureiter, C., Lastro, G., Uslar, M., and Lieber, P. (2019). Towards a standards-based domain specific language for industry 4.0 architectures. In *Complex Systems Design & Management*, pages 44–55, Paris, France. Springer International Publishing.

Fürst, S., Mössinger, J., Bunzel, S., Weber, T., Kirschke-Biller, F., Heitkämper, P., Kinkelin, G., Nishikawa, K., and Lange, K. (2009). Autosar–a worldwide standard is on the road. In *14th International VDI Congress Electronic Systems for Vehicles*, volume 62, Baden-Baden, Germany.

Haberfellner, R., de Weck,, O., Fricke, E., and Vössner, S. (2015). *Systems Engineering - Grundlagen und Anwendung*. Orell Füssli, 13 edition.

Hafner, M. and Breu, R. (2008). *Security engineering for service-oriented architectures*. Springer Science & Business Media. Berlin Heidelberg, Germany.

Knirsch, F., Engel, D., Neureiter, C., Frincu, M., and Prasanna, V. (2015). Model-driven privacy assessment in the smart grid. In *2015 International Conference on Information Systems Security and Privacy (ICISSP)*, pages 173–181, Angers, France. SciTePress.

Kreuzer, K. (2014). openHAB 2.0 and Eclipse SmartHome. http://kaikreuzer.blogspot.de/2014/06/openhab-20-and-eclipse-smarthome.html.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 1(4):267–284.

Neureiter, C. (2017). *A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid*. MBSE4U - Tim Weilkiens.

Neureiter, C., Eibl, G., Engel, D., Schlegel, S., and Uslar, M. (2016). A concept for engineering smart grid security requirements based on sgam models. *Computer Science - Research and Development*, 31(1):65–71.

OSGi Alliance (2004). Listener pattern considered harmful: The whiteboard pattern. https://www.osgi.org/wp-content/uploads/whiteboard1.pdf.

OSGi Alliance (2018). OSGi service platform, core specification. https://osgi.org/specification/osgi.core/7.0.0/.

Pichler, M., Veichtlbauer, A., and Engel, D. (2015). Evaluation of OSGi-based architectures for customer energy management systems. In *Proceedings of IEEE International Conference on Industrial Technology (ICIT) 2015*, pages 2455–2460, Seville, Spain. IEEE.