



Article

Open Information Architecture for Seamless Integration of Renewable Energy Sources [†]

Armin Veichtlbauer ^{1,2,*} , Oliver Langthaler ², Filip Prösl Andrén ³, Christian Kasberger ⁴
and Thomas I. Strasser ^{3,5,*} 

¹ Department of Mobility and Energy, Campus Hagenberg, University of Applied Sciences Upper Austria, 4232 Hagenberg, Austria

² Center of Secure Energy Informatics, University of Applied Sciences Salzburg, 5412 Salzburg, Austria; oliver.langthaler@fh-salzburg.ac.at

³ Electric Energy Systems—Center for Energy, AIT Austrian Institute of Technology, 1210 Vienna, Austria; filip.proestl-andren@ait.ac.at

⁴ Fronius International GmbH, Solar Energy, 4600 Wels-Thalheim, Austria; kasberger.christian@fronius.com

⁵ Institute of Mechanics and Mechatronics, Faculty of Mechanical and Industrial Engineering, TU Wien, 1060 Vienna, Austria

* Correspondence: armin.veichtlbauer@fh-hagenberg.at (A.V.); thomas.i.strasser@ieee.org (T.I.S.); Tel.: +43-50-804-22825 (A.V.); +43-50-550-6279 (T.I.S.)

[†] Andrén, F.P.; Strasser, T.; Langthaler, O.; Veichtlbauer, A.; Kasberger, C.; Felbauer, G.: Open and Interoperable ICT Solution for Integrating Distributed Energy Resources into Smart Grids. In Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), Berlin, Germany, 6–9 September 2016.



Citation: Veichtlbauer, A.; Prösl Andrén, F.; Langthaler, O.; Kasberger, C.; Strasser, T.I. Open Information Architecture for Seamless Integration of Renewable Energy Sources. *Electronics* **2021**, *10*, 496. <https://doi.org/10.3390/electronics10040496>

Academic Editor: Apel Mahmud

Received: 22 January 2021

Accepted: 16 February 2021

Published: 20 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: Electric power systems are currently confronted with a fundamental paradigm change related to its planning and operation, mainly caused by the massive integration of renewables. To allow higher penetration of them within existing grid infrastructures, the “smart grid” makes more efficient use of existing resources by integrating appropriate information technologies. Utilising the benefits of such smart grids, it is necessary to develop new automation architectures and control strategies, as well as corresponding information and communication solutions. This makes it possible to effectively use and manage a large amount of dispersed generators and to utilise their “smart” capabilities. The scalability and openness of automation systems currently used by energy utilities have to be improved significantly for handling a high amount of distributed generators. This will be needed to meet the challenges of missing common and open interfaces, as well as the large number of different protocols. In the work at hand, these shortcomings have been tackled by a conceptual solution for open and interoperable information exchange and engineering of automation applications. The approach is characterised by remote controllable services, a generic communication concept, and a formal application modelling method for distributed energy resource components. Additionally, the specification of an access management scheme for distributed energy resources, taking into account different user roles in the smart grid, allowed for a fine-grained distinction of access rights for use cases and actors. As a concrete result of this work, a generic and open communication underlay for smart grid components was developed, providing a flexible and adaptable infrastructure and supporting future smart grid requirements and roll-out. A proof-of-concept validation of the remote controllable service concept based on this infrastructure has been conducted in appropriate laboratory environments to confirm the main benefits of this approach.

Keywords: smart grid; information and communication technologies; networking infrastructure; automation; control application; renewable energy; system integration; overlay networks

1. Introduction

In recent years, the electric power grid had to face enormous challenges on the way to a more environmentally friendly infrastructure, supporting decarbonisation and higher

shares of renewables. This is a world-wide political goal; for example, in the year 2009, the European Union defined a set of target values for the energy industries, commonly known as 20-20-20 goals [1]. The objective was to reduce the emission of greenhouse gases, increase the share of renewable energy sources and improve energy efficiency by 20% each by the end of 2020. Renewables for power generation include technologies, like photovoltaics (PV) and wind energy plants. Since these technologies are very volatile and can only be predicted to limited extent, grid control becomes significantly more laborious and complex. Especially for PV, the use of small roof-top generation sites is very common nowadays; thus, they are feeding electricity into low voltage (LV) grids. This causes severe stability problems for LV grids, as they are massively distributed, i.e., control applications are needed for the coordination of a large number of participating nodes using respective communication means [2]. Furthermore, system maintenance (e.g., in case of security audits, technical updates, functional outages) and the need for adaptation to changing and locally different regulatory and legal conditions result in additional effort. Semi-automatic management, which can be done remotely, may alleviate the required amount of effort significantly.

Thus, the work at hand intends to contribute to an information and communication technology (ICT) infrastructure useful for smart grids, i.e., power grids with additional means to exchange, store, interpret, and edit data and to use it for control purposes [3]. This data may be used for control commands (switching consumers or producers on and off, etc.), measurement values (instantaneous power, or energy consumption in a 15-min slot, etc.), additional relevant information (price curves, weather forecasts, etc.), as well as the deployment of software components (for updates or new functions). The large number of different stakeholders (consumers, producers, network operators, facility owners, device manufacturers, maintenance engineers, authorities, certification bodies, sales organisations, etc.) generates high demands regarding openness and interoperability but also regarding the security of this infrastructure. A proof-of-concept prototype has been developed, which provides functionalities, such as a unified application programming interface (API) for control applications, a management interface for integrating new or updating existing functions, an engineering tool for developing control functions, a communication subsystem containing adapters for common smart grid protocols, easy to use integration of physical interfaces (sensors, actuators), and a role-based access control (RBAC) system for users and applications, including a policy enforcement point, providing necessary security and privacy of the operated data.

The basic ideas of this work have been previously published in Reference [4]. There, the concept and the prototype architecture have been outlined. Partial solutions have been also shown in Reference [5], where security by design aspects of the prototype architecture have been explored, or in Reference [6], where extensible messaging and presence protocol (XMPP) communication has been integrated into a distributed, standard-compliant control environment. For the engineering of control applications in such environments, Reference [7] applies a model-based system engineering (MBSE) approach. Anyhow, the present work introduces the whole concept with all corresponding details. It also provides a comprehensive discussion about the achieved results which goes far beyond the content of the aforementioned papers where mainly high-level ideas and first proof-of-concepts have been outlined.

This paper is organised in the following way: Section 2 describes the related work and state-of-the-art in the targeted field of smart grid information architectures. In addition, existing technologies and solutions which are potentially useful for the work at hand are outlined. Section 3 recapitulates our approach to overcome existing shortcomings, including the architecture of the developed prototype. A special focus on interfaces, address formats, and protocols, as well as the applied communication logic, is given in Section 4. After describing the realisation of the proof-of-concept prototype and the validation scenarios in Section 5, the gained results from conducting the respective tests are discussed in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

The massive roll-out of distributed energy resources (DER), like wind turbines, PV systems, small hydro power plants, biomass generators, etc., over the last decades has led to a fundamental paradigm change in terms of operation and planning of the electric power system [8]. As a consequence, the electricity generated from renewable sources has become visible in power transmission and distribution systems, creating additional challenges for the management of the power system, mostly due to the large numbers of systems, the variable power output and uncoordinated response to changing conditions of the power grid [9]. In some regions of the world, the deployment of renewable energy sources (especially PV) has been reaching or even exceeding the hosting capacity of distribution grids. In order to solve this issue, an effective integration into the power system is required, which becomes even more critical for the development of future DER components [7].

2.1. Meta-Models for ICT Architectures

The so-called smart grid [10] constitutes a very promising solution to use existing grid infrastructure more efficiently, which allows a higher degree of deployment of renewables [11]. All components (generators, loads, storage facilities, lines, substations, on-load tap change transformers, DER, measurement devices, smart meters, etc.) in the smart grid are interconnected via a powerful communication network and corresponding automation infrastructure in order to monitor, manage, and optimise the electric infrastructure in a more intelligent manner. To be able to fully utilise the benefits of smart power grids, it will be necessary to develop new control strategies. These strategies will make it possible to manage the large number of dispersed generators more effectively and to better utilise their “smart” capabilities, such as the ones that can be provided by inverter-based DER. For example, an overview of various research and development activities in the area of smart grids and DER integration on European and American level is discussed in Reference [12,13]. Here, ICT plays a key role for the implementation of such a smart and coordinated system approach [14]. In the dynamically evolving field of ICT solutions for smart grids, the development of standards is of crucial importance mainly due to interoperability requirements. Several standardisation organisations and various international projects have analysed this topic already [15,16].

On an international level, the International Electrotechnical Commission (IEC) provides a collection of common rules for the planning and operation of smart grids. Thus, the “IEC Smart Grid Standardization Roadmap” [10] suggests core standards to be used for the realisation of smart grids. This includes standards for data models, like the common information model (CIM), as defined in IEC 61968 and IEC 61970 [17], for power utility automation (e.g., IEC 61850 [18]), and many more. Gungor et al. [19] provide an overview of relevant smart grid standards, as well as roadmaps for further development. An impressive visualisation of standards is given in Reference [20]. The smart grid typical interaction of energy technology and ICT was specifically focused on in Reference [21].

In the smart grid domain, several proposals for organising ICT contributions have been made. The 3-dimensional smart grid architecture model (SGAM) model [22] is probably the most popular among these structuring meta-models. It includes not only interoperability layers but also zones (which refer to the parts of the automation pyramid) and domains (which incorporate the electricity grid’s value-chain from bulk generation to customer premises). An older model for interoperability layers (1-dimensional, but more granular) is the “Gridwise interoperability context-setting framework” [23] from the GridWise architecture council (GWAC). Figure 1 shows a comparison of these models. The lower layers of these models (up to the “Semantic Understanding” GWAC layer) are furthermore covered by the International Organisation for Standardisation (ISO) open systems interconnection (OSI) reference model [24] for computer networks.

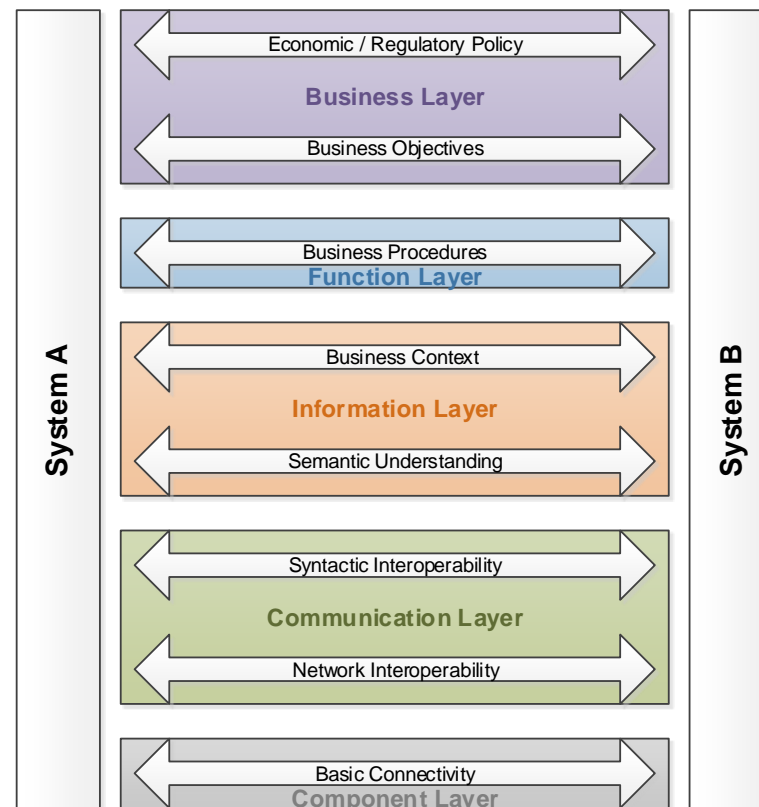


Figure 1. GridWise architecture council (GWAC) and smart grid architecture model (SGAM) layer models (adopted from [22]).

A special focus for smart grid information architectures should be set on the “middleware” layers, i.e., the “Syntactic Interoperability” and “Semantic Understanding” layers of the **GWAC** stack, respectively, the **OSI** layers 4 to 7. However, if the Internet is not used as common infrastructure, lower layer technologies have to be taken into account, as well. An overview of possible technologies for appropriate communication stacks is given in Section 2.3.

2.2. Architectures, Technologies, and Frameworks

Before considering the communication stack, the technologies for realising smart grid applications have to be assessed: First, the applications define the requirements on infrastructure and middleware. Second, the used technologies influence the interface and, thus, indirectly also the functionalities and the quality attributes of smart grid applications. Especially control applications are often realised using technologies, such as IEC 61131-3 [25] or IEC 61499 [26]. IEC 61499 is a promising technology for the smart grid domain, as it allows for distributed and event-driven control logic [27]. In case of IEC 61499, layer 7 functionality (and due to the hierarchical abstraction of the **OSI** scheme also the respective included lower layer services) would be incorporated in so-called service interface function blocks (**SIFBs**).

The question here is which communication services and functionalities can be included into such a function block, and how this can be done (i.e., which architectural approach is taken for integrating middleware and infrastructure services). When looking at adjacent domains, similar activities can be identified. In home and building automation (**HA/BA**), an “X-Architecture” using a convergence middleware layer has, e.g., been deployed in Reference [28]. Tooling support with connection to smart grids has been provided, e.g., by Reference [29]. The automotive domain has come up with a similar middleware architecture named AutoSAR [30].

The existing approaches have varying amounts of impact, yet it can be said that, in the smart grid domain, there are no widely accepted solutions for generic data exchange. Instead, a plethora of data standards and protocols has been defined, each of them tailor-made for a very specific use case, e.g., device language message specification (DLMS)/companion specification for energy metering (COSEM) [31] is widely used for smart metering and advanced metering infrastructures (AMI) [32]. These approaches are very useful but lack the genericity to serve as a basis for a generic interoperable information architecture.

Furthermore, there are numerous approaches for smart grid frameworks, which can be considered as middleware solutions, as they implement an (X-architecture like) intermediate tier between applications and hardware/operating system, like OGEMA [29] or OpenHAB [33]). Many of them deploy Open Services Gateway initiative (OSGi) [34] as basis, as surveyed by Pichler et al. [35]. This may be traced back to the origins of OSGi, which can be located in the areas of HA/BA. However, these installations are intended to work at a local site mainly, although they may be collecting data from remote sources (like sensors in the field). The main problem is their lack of interoperability [36], i.e., the collaboration of instances of different frameworks usually fails. Here, the middleware should embody common notions for data exchange between these instances, i.e., a common protocol stack.

Still, there are numerous approaches for distributed middleware solutions besides smart grid related infrastructure, which could be deployed generically, such as the common object request broker architecture (CORBA) [37]. In the field of industrial Internet of things (IIoT), solutions, such as the data distribution service (DDS) [38] or open process control unified architecture (OPC UA) [39] (sometimes combined with time sensitive network (TSN)), are common. Distributed applications based on these technologies are still limited, as they allow cooperation of distributed partners, but only when these partners are using the same base technology. Thus, commonly agreed protocols can yield enormous benefits for all stakeholders, e.g., user rights management could be based on lightweight directory access protocol (LDAP) [40].

2.3. Communication Concepts and Protocols for Energy Systems

According to the OSI model [24], protocols provide a northbound interface called service access point (SAP). The SAP of OSI layer 7 protocols (application layer, L7), thus, provides a generic API for whatever applications of different vendors want to use the underlying middleware and infrastructure. One of the most widespread realisations of such a generic API is the email system. The series of actions for sending, querying, and delivering emails is given by the L7 protocols post office protocol 3 (POP3), Internet message access protocol (IMAP), and simple mail transfer protocol (SMTP), respectively. Furthermore, the handling of email attachments is described by the multipurpose Internet mail extensions (MIME) definition [41], which gives information about the structure of the data that has to be exchanged. This is a very powerful and flexible system, allowing for data exchange not only between humans but also for machine communication—for instance, it can be used for exchanging calendar updates [42].

The success of the email system is its simplicity and genericity; the question is whether this concept can be transferred to the smart grid domain. Thus, in the following, potential technologies to be used as basis for such a smart grid information architecture, mainly covering functionalities located in the layers L4 to L7, will be examined. L7 defines an application specific series of actions (e.g., for control tasks); this has to be provided via the aforementioned generic API for the L7 functionality. Just like SMTP provides an L7 SAP to email clients, this API shall provide a similar point of communication service provision for smart grid devices and applications. State information may be held in the application itself, or in the L7 protocol. Application layer protocols, as mentioned, are tailor-made for the concrete use cases; however, standards, such as IEC 60870 [43] and IEC 61850 [18], have gained prominence in many smart grid-related data exchanges, predominantly in the transmission grid infrastructure. IEC 61850 allows for a tree-like data object structure

(thus also defining L6 functionality). Whereas IEC 61850 is connection-oriented (and thus stateful), IEC 60870 uses (stateless) telegrams; this is more lightweight but may lack functionality needed for control applications. As for the semantics of the L7 payload, we seek a unique data modelling. This could be provided by the mentioned CIM [17], which has gained importance in the smart grid world over the last few years [44].

Furthermore, L6 technologies are needed which are able to represent the data models defined by CIM. A good candidate for this is OPC UA [39], which is a common technology to be used for connecting “nodes” (communicating end systems) in distributed industrial environments. It constitutes a message-oriented middleware [45], which already provides service orientation. In addition, the combination of OPC UA and CIM has already been proven to work well [44]. As for L4/L5, the focus will be set on real-time capable end-to-end protocols, preferably stateless protocols, as state information will be needed already for the L7 protocol, and a doubling could be too inefficient for the desired solution. Session handling could be performed by session initiation protocol (SIP) [46] or XMPP [47]. For the transport layer, user datagram protocol (UDP) [48] might be an alternative to the predominant transmission control protocol (TCP) [49], especially for use cases involving strict latency requirements. However, mechanisms to ensure delivery of control commands have to be defined; automatic repeat request (ARQ) could be inappropriate due to latency properties—yet reliability is also an important topic for control applications.

Regarding the communication infrastructure (i.e., the lower part of the SGAM Communication Layer), Internet protocol (IP) should be taken as granted. The only question is, which version is to be used. IPv6 [50] might have advantages over traditional IPv4 [51] for the huge number of possible addresses, which makes it more suitable to IoT environments. This applies especially to the smart grid, due to the numerous devices in the field (e.g., power sensors [52]). However, field devices often face performance problems, which could be worsened by using 128 bit long addresses—6LoWPAN [53] yet provides IPv6 with header compression. Depending on the requirements of L2, this might be an option. As for L2, IP support is strongly desired. Thus, IP capable field buses, like Modbus (in the variants Modbus/TCP and Modbus/UDP) [54], will play an important role. Furthermore, L2 should be real-time capable—so real-time Ethernet variants, such as Ethernet Powerlink [55], could be considered. This would also define L1 (the physical layer).

2.4. Open Issues and Shortcomings

To summarise, there are a couple of open and unsolved issues when it comes to the integration of DER units into smart grid solution. From an automation and ICT-point of view, the most relevant topics for the present work are:

- missing common automation application modelling concepts for power and energy systems,
- scalability and openness of automation and control applications with a high share of DER components is only partly addressed,
- lack of common and open communication interfaces in smart grids impede scalable and distributed automation solutions,
- missing possibilities to update and extend DER services, and
- available proprietary automation concepts in smart grids prevent efficient reuse of control software; thus, the engineering costs exceed admissible costs by far.

In the following sections, a potential approach that overcomes those issues is introduced and discussed in detail.

3. Approach and Architecture

In order to overcome the mentioned shortcomings, a concept for the desired ICT infrastructure has been developed, and a proof-of-concept prototype has been implemented and validated. This section outlines the elicitation of requirements on the solution and the design of an appropriate information architecture, as well as the basic functionalities covered by the prototype components.

3.1. Use Cases and Requirements

Potential application scenarios of the approach at hand have been defined and analysed on the basis of a use case analysis in order to derive respective functional requirements. This use case methodology is widely used in ICT engineering and provides a structured approach to identify requirements for a system under design in a technology independent way. In addition, in the domain and energy systems domain, the usage of such a methodological approach becomes more and more important. The “IntelliGrid Methodology for Developing Requirements for Energy Systems”—which is covered by the IEC 62559 standard [56]—defines a corresponding approach for defining use cases and deriving requirements especially for the smart grid. This very well known methodology usually separates the concepts of user requirements from technical specifications; therefore, user requirements define *what* is needed without taking any specific designs or technologies into account. On the other side, technical specifications define *how* the specified systems have to be realised in order to fulfil the user requirements.

The definition of actors is one of the crucial parts in the process. An actor is an entity having a defined behaviour when interacting with the system under discussion. In principle, actors can be humans, organisations, (technical) components, and systems (or even systems of systems). The identification of actors has already been carried out in various smart grid related activities so far (DKE [57], M/490 [58], etc.). Therefore, a survey of existing projects has been carried out and relevant actors have been collected and evaluated. Useful actor definitions have been included in a use case management repository hosted at DKE [57]. On that basis, 25 relevant use cases in 6 use case clusters as visualised in Figure 2 have been collected and analysed. Hereby, the operational use cases cover some intended control applications, whereas the process-oriented use cases describe the interaction with the proposed system itself.

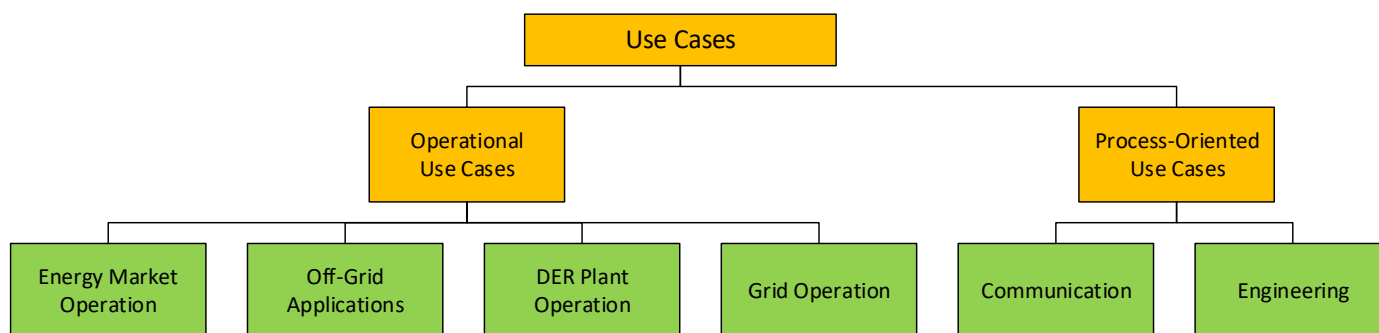


Figure 2. Overview of identified use case clusters.

The process-oriented use cases yielded up to about 40 requirements, categorised into rights management, communication, and application requirements, as depicted in Figure 3. Finally, validation scenarios have been derived (see Section 6), which cover one or more of the mentioned use cases, as well as validation environments, which hosted the tests for these validation scenarios.

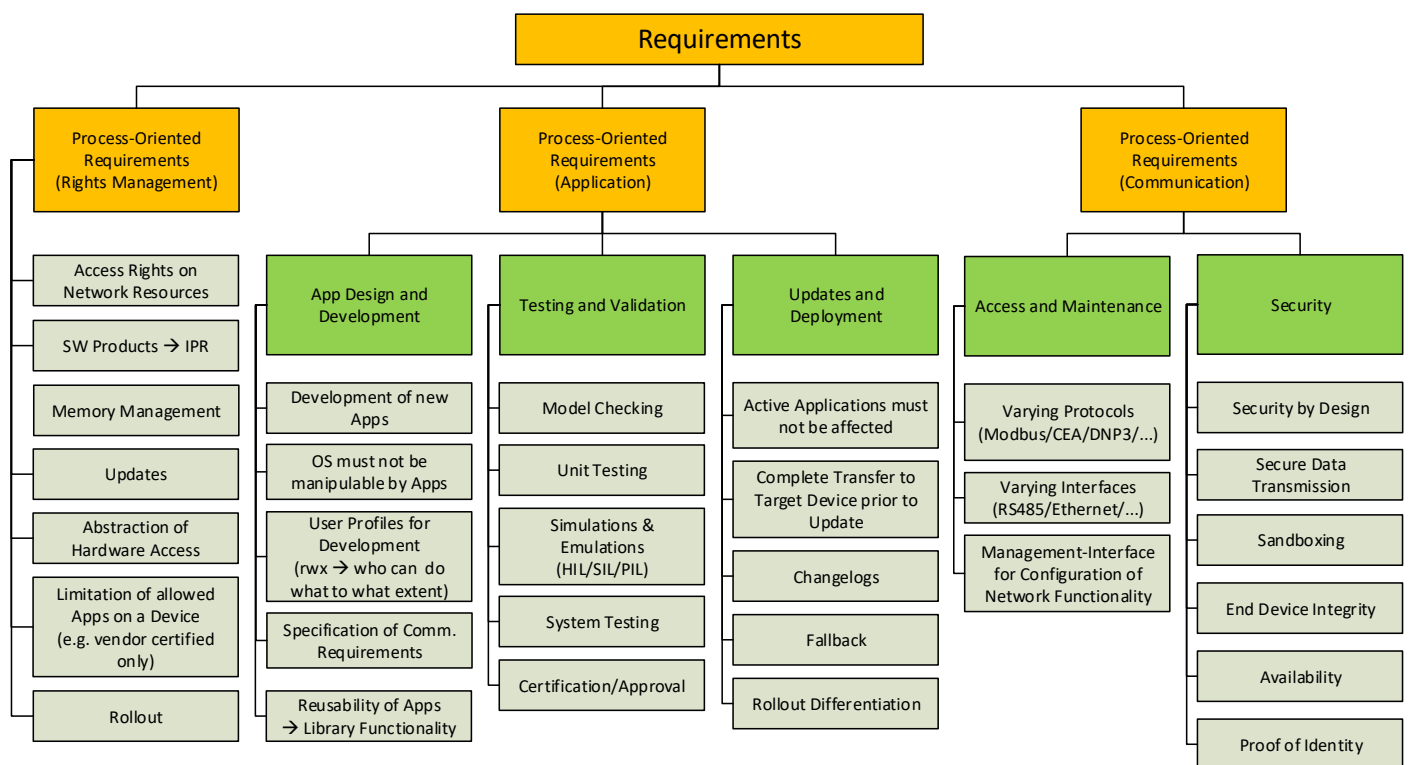


Figure 3. Overview of derived requirements.

3.2. Overall Concept and System Architecture

Based on the aforementioned requirements, a prototypical solution has been designed and implemented in order to serve as a proof-of-concept. The overall system architecture of this prototype is depicted in Figure 4. Hereby, the power flow (solid lines) and the data flow (dotted lines) between different intelligent electronic devices (IEDs)—be it transformers, DERs, remote terminal units (RTUs), or any other kind of IED—are visualised. Each IED comprises hardware and a middleware named “SmartOS”, as well as “remote controllable services (RCSs)”. The SmartOS again contains hardware control, security and access management, as well as connectivity functions. The RCSs access the middleware via a defined API only, and again provide functionality, such as voltage measurements, needed by smart grid applications, like Volt/var control. These services have been designed to run in sandboxes; thus, they can communicate with their environment only via the middleware.

In case such RCSs want to communicate with other RCSs on different IEDs, the SmartOS has to provide the connection to a remote IED via an appropriate connectivity module. Figure 5 shows the respective module for external communication in the context of the overall system architecture depicted above. One of the main goals of the proposed architecture is to provide flexibility with regard to the integration of different smart grid-related protocols since there is currently no unified solution available. Typically, protocols, like Modbus, DNP3, or IEC 61850, are used, and the proposed approach allows to integrate all of them concurrently, as depicted in Figure 5.

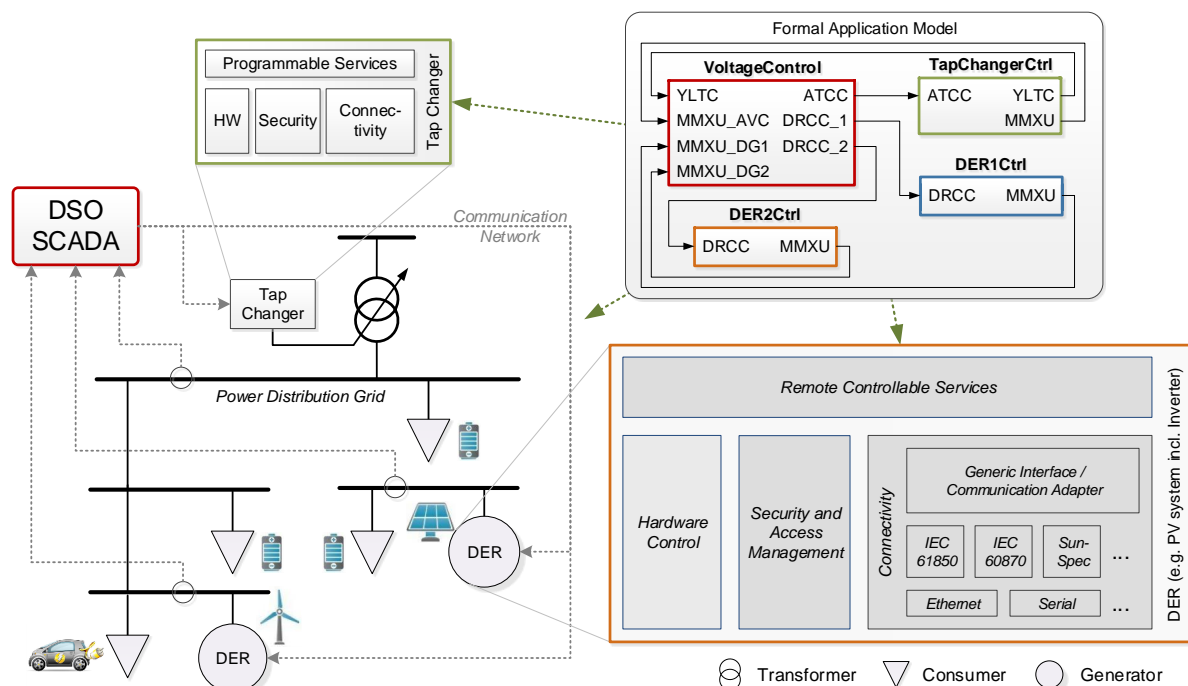


Figure 4. Overview of the overall system architecture (adopted from Reference [4]).

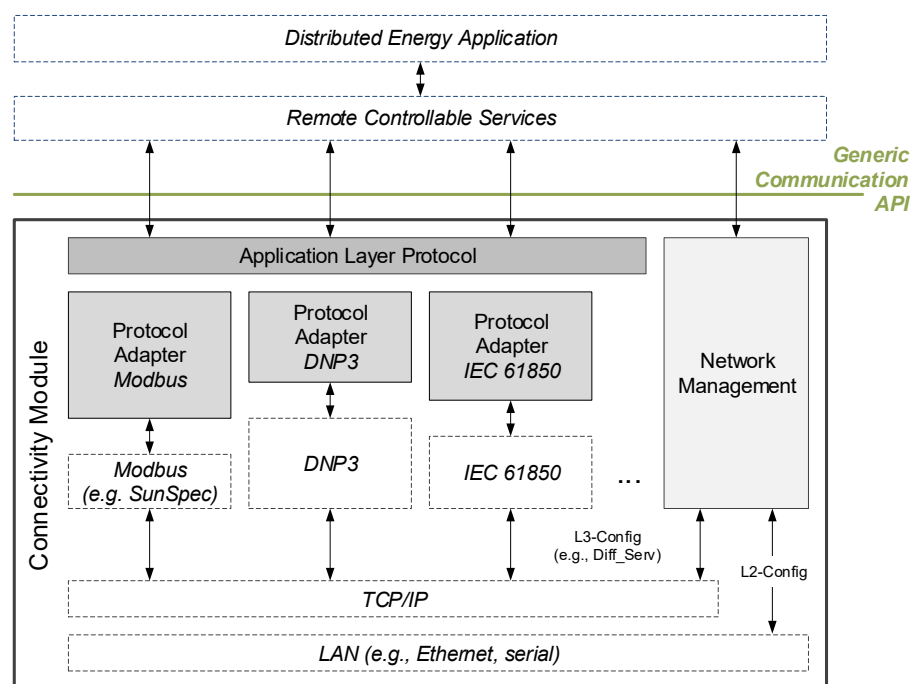


Figure 5. Elements of the connectivity module.

The main parts of this module are:

1. a common application layer (L7) protocol called “open application layer protocol (OpenALP)” (see Section 4.3) providing a unique access point (L7 SAP) to connectivity functionality for RCS,
2. legacy protocols widely used in the power and energy systems domain,
3. adapters for missing functionality of these protocols (e.g., for keeping state information),
4. existing network infrastructures (e.g., on the basis of the TCP/IP/Ethernet stack), and

5. network management functionality to provide the possibility of (re-)configuration during runtime.

3.3. Subsystems and Functionalities

An overview of the software within one IED is given in Figure 6. Each IED contains the SmartOS middleware, which serves as container for hardware control (e.g., access to sensors and actuators), connectivity (communication to other IEDs), and security (especially access control to restrict functionality to users with respective rights), as well as a registry for services (in an OSGi-like manner) and users (based on LDAP [40]). Despite its name, the SmartOS, thus, actually constitutes a middleware rather than an operating system (OS).

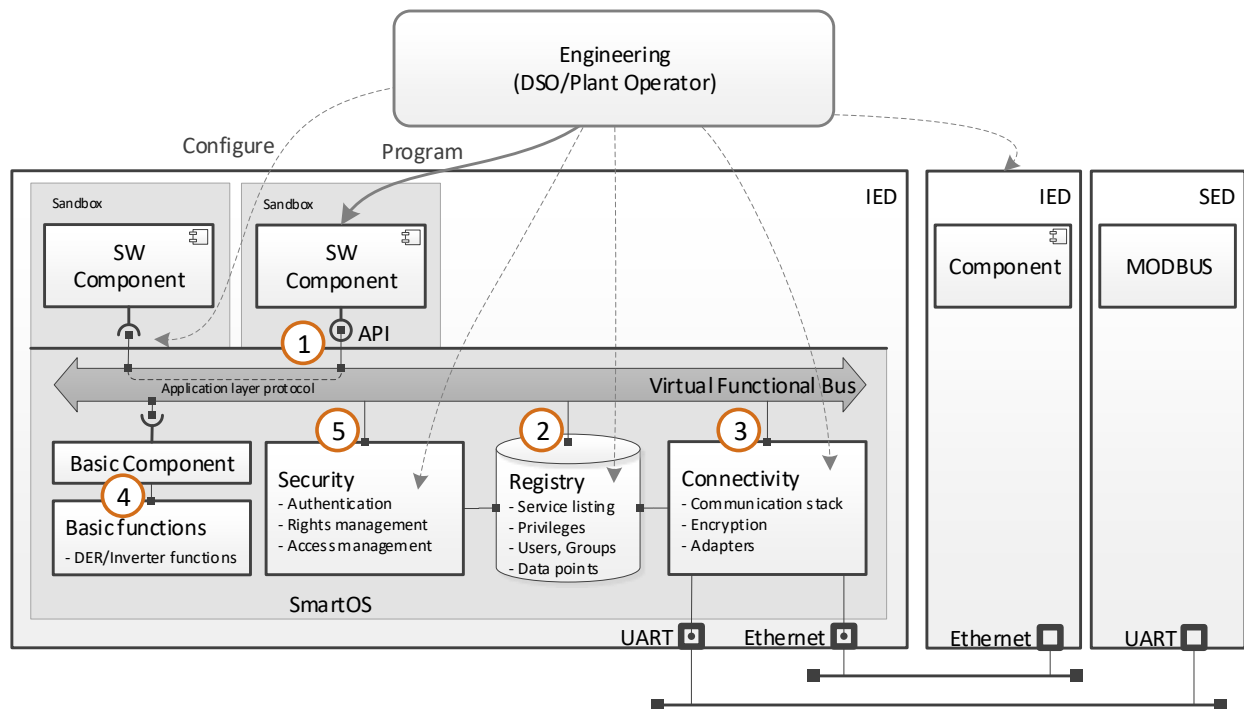


Figure 6. SmartOS and Virtual Functional Bus (VFB) (adopted from Reference [4]).

3.3.1. The Virtual Functional Bus

The core subsystem of the SmartOS is the “virtual functional bus (VFB)”, which provides intra-IED communication for different RCSs denominated as “software components”. This is done via the specifically developed OpenALP (see Section 4.3). Furthermore, the VFB connects the SmartOS modules, which are described in more detail in Section 3.3.2. The numbers shown in Figure 6 relate to the interfaces provided by the VFB. As only exception, interface 4 connects the basic component (which is a pre-defined software component encapsulating direct hardware access) to its respective basic functions, i.e., sensor and actuator access. To fulfil the requirements, the VFB needs to provide an ecosystem which is acting as a message oriented middleware (MOM) designed for a highly distributed environment.

Consequently, IEDs and their software components, as shown in Figure 6, will exclusively be provided with a predefined address scheme (see Section 4.1) for communication. They do not require information about the location of their communication partners or the appropriate protocol and physical adapter needed for communication. Such an infrastructure can be provided by a classical messaging system, which is explained in more detail in the following. Furthermore, the VFB represents the central point for dispatching messages by querying the registry and security modules and finally forwarding the messages to the correct recipient(s).

The messaging system is the basic functionality of the VFB. As such, it must provide a level of functionality similar to message queues, where each possible component is provided its own input and output queue. They need to be based on classical first in first out (FIFO) mechanisms which can be filled with a specific number of messages and ensure sequential delivery. Messages are received and sent by the VFB itself, by software components, or by the connectivity module.

The SmartOS middleware allows adding RCSs on a plugin basis. Ideally, they can be deployed as a package without the need for copying many different files. Third parties should be able to develop such software packages and they should be reviewed and certified by the hardware manufacturer of the device they are intended to run on. When certified for the brand/device, such software components ideally are signed by the owner and by the company reviewing the software component, using a commonly known and established signature methodology. Signed software components finally receive standard permissions on a device to run in the sandboxed context of the VFB, which limits system access to the provided VFB API described in Section 4.

The design of the software component plugin system should allow hot-plug features, such that new applications can be loaded and updated dynamically during runtime. As the VFB is based on a messaging system, a software component needs to be able to access this system. To achieve this, the VFB infrastructure provides the mentioned common API (interface number 1 shown in Figure 6), which is accessible to internal components (e.g., the basic component), as well as to the loaded third-party applications. Furthermore, communication should be restrictable for an individual or a group of communication parties. This has to be conducted according to the permission system handled by the security module and the device registry. However, the VFB itself acts as the respective policy enforcement point (PEP) which processes or discards the messages accordingly.

3.3.2. Modules

The SmartOS architecture contains two main important modules which are:

Connectivity Module

IEDs may communicate with other IEDs or RTUs via the connectivity module, using various kinds of protocols. The connectivity module has the ability to establish new connections or handle in-coming messages from established servers (instantiated by the VFB) via communication technologies, such as Ethernet, serial connections, or field busses, like controller area network (CAN) bus. To do this, the connectivity module is aware of the necessary adapters and protocols; these are determined via registry entries. Furthermore, adapters are implemented in a way to allow non-blocking operation and multiple simultaneous connections, if the respective protocols require such functionalities.

If the connectivity module establishes a connection, some mapping needs to take place to check if there is already an open channel for that particular communication partner to continue communicating on that channel. This is handled via a simple lookup-table. When reconnecting on the other hand, an adapter and a protocol need to be chosen; this is done by translating the system address into an adapter address. Finally, it is possible to execute protocol specific tasks to prepare the open channel for communication. This is achieved by the respective protocol adapters. When receiving new connections, the adapter and the protocol need to be pre-defined because an incoming request is handled by the particular server implementation of the adapter. Thus, an incoming message needs to be verified (if applicable) and forwarded to the VFB.

Security Module and Registry

The security module is one of the central elements ensuring security within the infrastructure at hand. Therefore, it needs to be able to impose a permission strategy on the messaging system, thus serving as a policy decision point (PDP). It has also to restrict loading only to signed software components and to provide the necessary information to

the sandbox in order to allow or deny a specific function call. Additionally, the security module needs to know which functions and configuration parameters are accessible and to which extent. An important pre-condition to provide an appropriate rights and access management is to authenticate requesting modules and their respective users in cooperation with the registry. On each SmartOS capable device, a persistence and configuration layer needs to be maintained by a central authority. This part is referred to as the registry. It needs to contain a variety of information required for the normal operation of the device. These configuration entries include the following types of information:

1. Device: Device name, device interface, application folder,
2. Address: All accepted outbound and inbound addresses,
3. Registered applications: Registered running applications, including their data points,
4. Data points per application: Manually registered data points,
5. Adapter: All adapter implementations which run as a server thread,
6. Users: All registered human or machine users and their respective access rights, and
7. Permission: Permissions based on (at least) a 3-tuple of sender, recipient, and communication type, supporting wildcards.

3.3.3. Internal Interfaces

Here, the most important interfaces between the various components internal to the IED are considered. Internal in this case refers to the fact that under normal circumstances, they will not be particularly relevant to any third parties implementing software components or adapters for the connectivity module. The internal interfaces are accessed by the VFB only, i.e., they connect the VFB to a functional module. The interface between the VFB and the security module (interface 5 in Figure 6) can essentially be limited to a single function call: For any message arriving via the connectivity module, the VFB must query the security module (which in turn must query the local database and/or a central permission store, such as an LDAP server), for existence of the appropriate permission. If a corresponding entry is found, true should be returned and the message can be passed on by the VFB. Otherwise, the message has to be discarded by the VFB.

The VFB and the connectivity module (interface 3 in Figure 6) should be linked in a sequential and non-blocking way, e.g., via a set of message queues. Communication needs to be completely encapsulated and only possible over the messaging system specified above from and to the connectivity module. In order to enable software components to register and de-register data points, the VFB needs to be able to access the registry (interface 2 in Figure 6) as persistent data storage. Furthermore, since the VFB acts as a PEP, it must be able to indirectly access the registry via the security module which, in turn, acts as a PDP relying on the corresponding entries in the permission table.

3.4. Engineering Approach

Besides the prototypical realisation of an IED-based ICT infrastructure, an appropriate tooling was also necessary to provide. This tooling had the task to support engineering of smart grid applications from design over implementation and testing up to deployment and maintenance. Thus, this engineering environment should support the whole product lifecycle, e.g., by hot plug&play of new or modified services and applications in an OSGi-like manner, but tailor-made for the smart grid domain, rather than for HA/BA.

The SmartOS concept of RCS sandboxing (see Figure 6) in combination with access right enforcement allows for using services in the intended flexible way. Thus, access to system resources is granted via a controlled access way only but can yet be modified easily during runtime. However, these modifications have to be performed solely by system services in order to be done again under complete access control (it has no sense to enforce access control rights for applications, when these rights can be manipulated without control), while being easily invoked by legitimate users. This means system services are engineered the same way as energy services.

Energy services can then be utilised to engineer higher order applications, as for instance Volt/var control in LV grids (see Section 6). To enable engineering in an easy to use way, as well as for allowing reuse of existing functionalities, a model-driven architecture (MDA)-based approach has been chosen, as given in Reference [7]. According to MDA, the platform independent model (PIM) is created using a domain specific language (DSL), here called the power system automation language (PSAL). PSAL is intended as a tool for SGAM compatible use case design and at the same time it provides a basis for further code generation. Thus, PSAL is also the starting point for all design activities in the provided engineering methodology. As an example, in Reference [7], IEC 61499 function blocks (FBs) are used to realise a Volt/var control application, engineered with the help of PSAL.

Since SGAM is already divided into different layers, it makes sense to keep this structure in PSAL, as well. When studying the SGAM layers, it is clear that the component layer and the communication layer are more “static” (in the sense of related to existing infrastructures) than the information, function, and business layers. According to this, PSAL was divided into one static and one dynamic model, the latter also called the functional model. A graphical representation of the main parts of PSAL is provided in Figure 7. In the static part, the component layer and the communication layer are represented, whereas, in the dynamic part, the information layer, the function layer, and the business layer are represented.

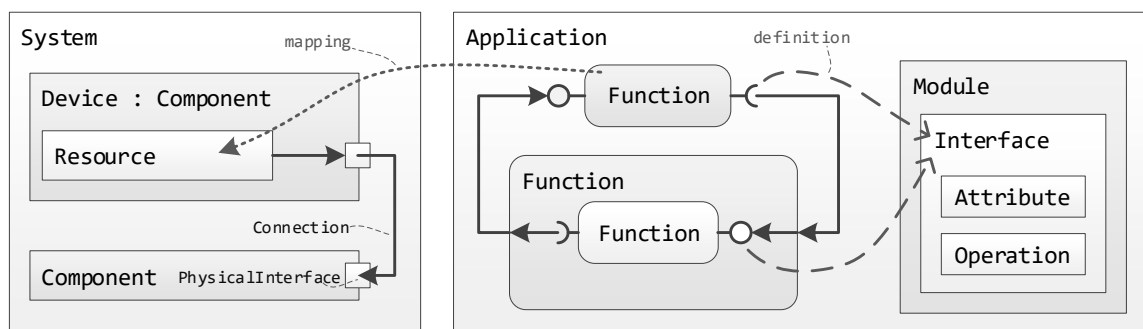


Figure 7. Elements of the Power System Automation Language (PSAL) [4].

Another important requirement is platform independence. An application or functionality should be independent from its possible execution platform. Expressed in SGAM, this would correspond to using the same business, function, and information layers, while using different communication and component layers. From a user perspective, this means that it must be possible to define a functional model independently from the static model.

According to Reference [7], the static model is represented by the *system*, a container for *components* and *connections*. Together, these can be used to describe the ICT system, as well as the electrical power system. In this work, the emphasis is on ICT and automation equipment. Examples of ICT equipment are routers and switches, while automation equipment is mainly represented by IEDs and embedded controllers. Each *device* contains one or more physical communication interfaces. Part of the device is also the *resource*, a function container similar to the logical devices (LDs) used in IEC 61850. By means of multiple resources, it is possible to logically group functionality of a device. To complete the component layer, devices can be connected to each other through connections. Each connection can be defined through a number of *attributes* (e.g., communication protocol and performance requirements). With the system, the component layer and the communication layer of SGAM are modelled.

The *functions* are defined in the *application*, the dynamic model of PSAL as discussed in Reference [7]. Here, the exchanged information is also defined. Using the SGAM methodology, it is possible to roughly describe functionality in different unified modelling language (UML) diagrams. Use case, activity, and sequence diagrams are tools that are often used for this purpose [22]. However, in order to automatically generate code from the

use case description, more details are needed in the descriptions of the functions. In [PSAL](#), this is modelled by the functions, which are defined through a software component model, a containment model that supports different levels of detail. The component model was developed with focus on the function layer rather than on the business layer. Each function is a component that can contain other functions. This allows the developer to choose the level of detail that is necessary for the use case. Furthermore, each function can provide or request services that are defined by an *interface*. The information of an interface is defined using attributes and *operations*. The application represents the business, function, and information layers of [SGAM](#).

According to Reference [7], the connection between the system and the application is realised by a function-resource mapping. By performing such a mapping, all provided and requested services of the respective function are also available on the interface of the resource. Consequently, these services are also available on the physical interface of the device. One of the advantages of using a textual syntax is that comments can be used anywhere in the source code. As a result, the documentation of the constructed use case is simplified. A major part of [SGAM](#) is the placement into domains and zones. However, this placement is mainly intended for documentation purposes and does not directly affect the implementation. For the execution of code, it is not important if a device is located in a station or a field zone. With a textual [DSL](#), the definition of domains and zones can be achieved through the use of annotations (e.g., @DER and @Station).

4. Data Semantics and Interface Definition

To realise a concept as stated above, some data semantic issues have to be considered. In addition, some formal definitions have to be given, especially when it comes to the integration with external (system context) components. In the concrete prototype environment, addressing objects uniquely within in the whole envisaged system is a crucial precondition to allow for wide area communication. Furthermore, defining the data semantics when exchanging or processing data is inevitable. However, in order to allow for integrating elements of different legacy, adapters to the used mnemonics have to be used. Thus, this section intends to document all the necessary working conditions for the prototype being able to proceed.

To reduce the complexity of software components, a specific “[OpenALP](#)”, along with an address scheme, has been developed. It is intended to enable software components to communicate with other devices or software components with little or no knowledge about the various possible protocols which may be involved in cross-device communication. First of all, the specification of the developed “[OpenAddress](#)” format is provided.

4.1. Address Format

The following definition, which is formulated in extended Backus Naur form ([EBNF](#)), shows how device addresses are formed in the work at hand:

```
OpenAddress = Device "/" Namespace ( "." Namespace)* "$" Variable (" $"
Variable)*
```

Hereby, the terms describe the following semantics:

Device	The name of the target device
Namespace	A namespace within a device—Namespaces are defined by the software components; each namespace typically corresponding to one software component.
Variable	The address of a data point (which can be a simple variable but also a structure) within a namespace—Variables are also defined by the software components.

The following example shows a complete address in OpenAddress format:

```
IED1/Measurements.POC.Voltages$PhaseToNeutral$phsA
```

4.2. Mapping between OpenAddress Format and Legacy Addresses

To be able to translate an OpenAddress into a hardware address which can be used by a protocol adapter, an address mapping needs to be created and stored as an address mapping entry in an address translation table. In theory, several available routes with different adapters may be possible, so each address mapping allows for the declaration of multiple adapters. The corresponding EBNF definition is as follows:

```
AddressMapping = OpenAddress ":" "{" AdapterName "[" AdapterAddress  
"]" ("," AdapterName "[" AdapterAddress "]" )* "}"
```

Again, the meaning of the terms is as follows:

OpenAddress	The OpenAddress of a data point; see above.
AdapterName	The name of the corresponding adapter (e.g., "MODBUS", "IEC61850").
AdapterAddress	The internal address of a data point within the adapter/protocol.

The syntax of AdapterAddress is adapter specific, two examples are specified below:

```
AdapterAddress = ModbusAdapterAddress | Iec61850AdapterAddress | ...
```

The ModbusAdapterAddress is the address scheme for the MODBUS [54] adapter, which is defined as follows:

```
ModbusAdapterAddress = ModbusEntity "," ModbusAddressRange  
ModbusEntity = "Coils" | "DiscreteInputs" | "HoldingRegisters" |  
"InputRegisters"  
ModbusAddressRange = INTEGER (".." INTEGER)?
```

Here, the ModbusEntity specifies the register type of the given address, whereas the ModbusAddressRange is the range of valid addresses.

The Iec61850AdapterAddress denotes the address scheme for the IEC 61850 [18] adapter and is defined as follows:

```
Iec61850AdapterAddress = LDevice "/" LN "." DO "." DA ("." BDA)*
```

Here, the adapter address consists of an IED name, a logical node, and several hierarchical structures of datapoints.

To aid comprehension, the following examples show complete addresses in OpenAddress format and how the contained variables are derived from adapter datapoints:

```
IED1/Measurements.POC.Voltages$PhaseToNeutral$phsA  
: {MODBUS[HoldingRegisters, 4523], IEC61850[DERCtrl/  
MMXU1.PNV.phsA.cVal.mag.f]}  
IED1/Measurements.POC.Voltages$PhaseToNeutral$phsB :  
{MODBUS[HoldingRegisters, 4524..4525], IEC61850[DERCtrl/  
MMXU1.PNV.phsB.cVal.mag.f]}
```

4.3. Open Application Layer Protocol

The prototype specific L7 protocol OpenALP is the second key element to ensure low development overhead within software components. In combination with the OpenAddress format, software components require little or no knowledge about the various possible protocols which may be involved in cross-device communication. Regarding

[OpenALP](#), this property is achieved by abstracting the generic communication patterns from the underlying communication protocols which are considered to be required within the prototype architecture. These communication patterns form the basis for the [API](#), which is provided to the [RCS](#) by means of the SmartOS (see Section 3).

- **Push 1:1:** Refers to a message being transferred to a single destination software component or device. This functionality may be used, e.g., for setting values, thereby influencing the operation of a remote system. A real-world analogy is the shipment of a letter.
- **Push 1:N:** Is very similar to Push 1:1, the only difference being that with Push 1:N, more than one recipient may be specified. This functionality is an abstraction from broadcast or multicast messages, which exist in several protocols. It may be used, e.g., for setting values in multiple devices simultaneously or for transmitting notifications which concern more than one device. A real-world analogy is the shipment of bulk mail.
- **Request/Response:** Describes the generic process of requesting information from a target. The request is sent and a response containing the requested information, or at least an error message, is expected. It may, for example, be used to retrieve data relevant to local control decisions, such as measurement values or price information, from a remote system. The most similar real-world analogy is a registered letter.
- **Publish/Subscribe:** This communication pattern is used whenever automatic updates about a data point are desired, the requesting entity being denoted as subscriber and the entity sending the updates as publisher. The conditions under which messages are published may vary. Typically, a message is sent whenever the subscribed value changes, but minimum and maximum time intervals between messages may be defined. A minimum time interval may be used to prevent excessive traffic, while a maximum time interval may serve as notification of continued availability of the publishing party. The most similar real-world analogy is a newspaper or magazine subscription.

5. Prototype, Testbed, and Validation

Following the definition of the conditions and the system context of the intended prototypical solution, a proof-of-concept implementation has been performed. The prototype has then been used in an appropriate testing environment. This section informs about both the prototypical implementation, as well as the testbed environment in which the prototype has been used and tested.

5.1. Prototypical Implementation

A platform specific prototypical implementation of the SmartOS framework has been performed in order to provide a test environment suitable for validating the approach (see Figure 8). In the scope of the work at hand, this test environment has been developed based on Java, as Java allows for a maximum of platform independence and many useful libraries, for example, for serialisation and queuing functionalities or for legacy protocols, are available. In addition, in combination with [OSGi](#), Java has been used for multiple platforms in [HA/BA](#), as well as in smart grid environments [35].

While the prototype is not a fully-fledged implementation suitable for productive use, it provides all of the functionality required for validating the approach (see Section 5.3). It is designed to conceptually cover all aspects of the presented architecture, thereby ensuring the viability of the envisioned solution.

In order to meet the previously defined requirements and provide as much flexibility as possible, a highly modular design approach has been taken. Figure 8 provides an overview of the different components and the respective interfaces of the implemented prototypical software.

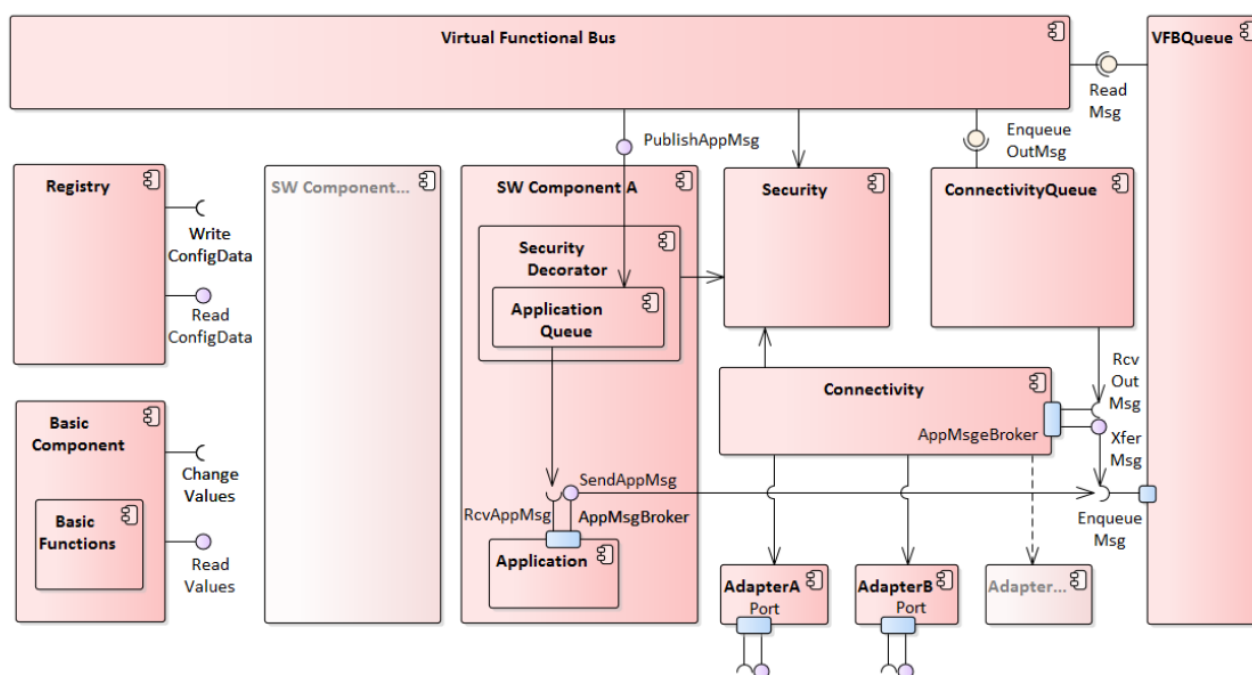


Figure 8. Platform specific implementation of the SmartOS.

5.2. Testing Environment

The environment used for testing, simulation and validation is based on a VMware vSphere hypervisor running various CentOS and Debian virtual machines. The hardware consists of a compact laboratory server which has been assembled using then state-of-the-art hardware (4 Core Intel Xeon CPU, 16 GB RAM, 512 GB SSD storage) and is used exclusively for the work at hand.

Testing, simulation and validation have largely been carried out using a number of virtual machines running the various prototypes as sending and receiving elements (representing the various involved devices and stakeholders). In order to provide a more realistic environment, a network infrastructure, including several virtual routers and switches, has been set up. The network topology, including the various virtual machines, is illustrated in Figure 9. This was complemented by temporary virtually private network (VPN) infrastructure allowing external and/or remote devices from another lab to remotely connect to any of the subnets as needed.

vSphere allows the link speed of virtual switches to be defined in 1 KB/s steps, which makes it possible to simulate connections with low bandwidth, such as legacy powerline communication (PLC) networks. Furthermore, OMNeT++ and Mininet were conceptually considered to simulate more complex intermediate transmission paths and connections with high load (denoted Transmission Path Simulator in Figure 9), but extensive simulation and performance evaluation was ultimately deemed beyond the scope of this work.

On the machine denoted credential store, enterprise JavaBeans certificate authority (EJBCA) has been used in combination with open lightweight directory access protocol (OpenLDAP) to provide a public key infrastructure (PKI), as well as the necessary user and rights management capabilities. Both products are widely used and actively maintained state-of-the-art open source solutions that may be used similarly in a production environment.

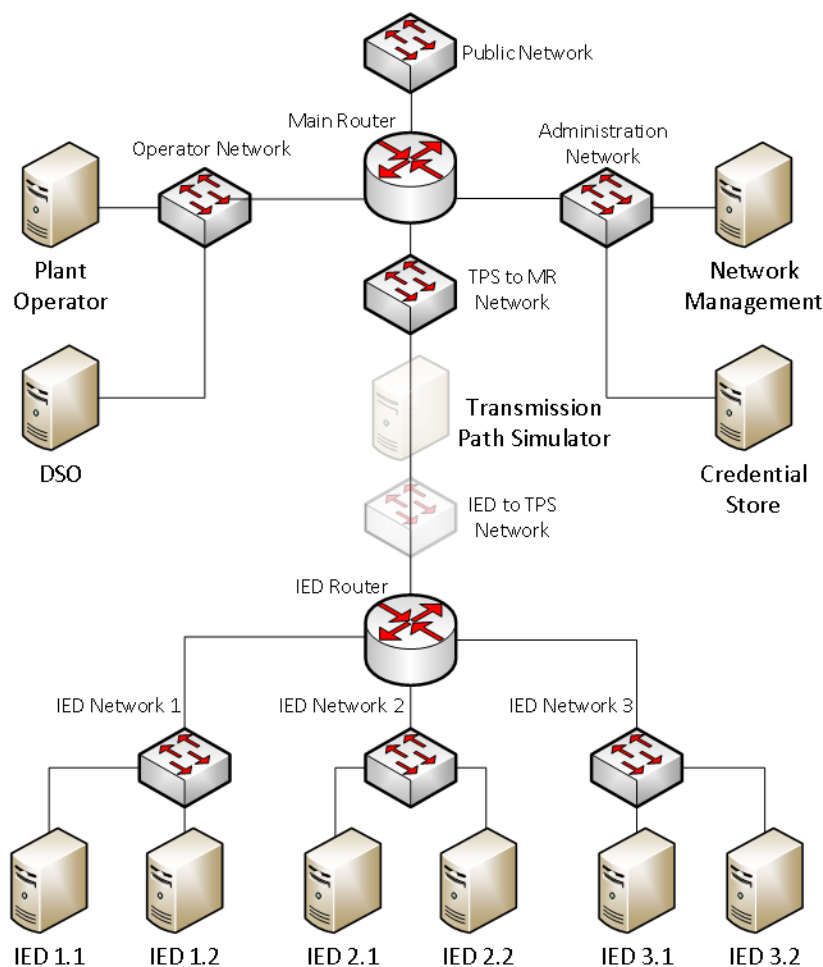


Figure 9. Realised testbed for validation and testing.

5.3. Validation Approach

The validity of the approach finally had to be documented by executing appropriate test scenarios, gathering the respective results of the conducted tests, and contrasting these results with the defined requirements, which is discussed in this section.

First, the requirements regarding the proof-of-concept solution have been derived by analysing the process-oriented and operational use cases outlined in Section 3.1. Operational use cases cover typical control tasks, which have to be performed by operational systems; thus, the developed system had to show via a concrete example, that these use cases can actually be performed. Process-oriented use cases cover functionalities to provide easy-to-access engineering tools and methodologies, as well as flexible utilisation of a reliable, secure, and robust communication infrastructure.

Thereafter, the concrete test cases for the system have been defined on basis of a set of validation goals and requirements, including pre- and post-conditions for successful testing. Accordingly, the respective test environments have been provided. The test cases have been conducted as unit tests, e.g., for the Java-based communication infrastructure, appropriate JUnit tests have been performed in the virtualised network environment described in Section 5.2.

Having successfully executed the envisaged functionalities in the respective unit test environments, an integration test has been defined in order to prove, that the participating building blocks are cooperating as intended. This integration test was performed with real hardware (i.e., an inverter-based DER provided by Fronius) at the AIT SmartEST lab [59] and the Fronius system test laboratories to prove the potential of the developed solution for real world scenarios (see Figure 10). This scenario covered the setting and

getting of inverter data points by an application engineered with the developed engineering toolset and utilising the developed communication infrastructure (including the SmartOS middleware and the [OpenALP](#), as well as the certificate authority (CA) and [LDAP](#)-based security infrastructure).

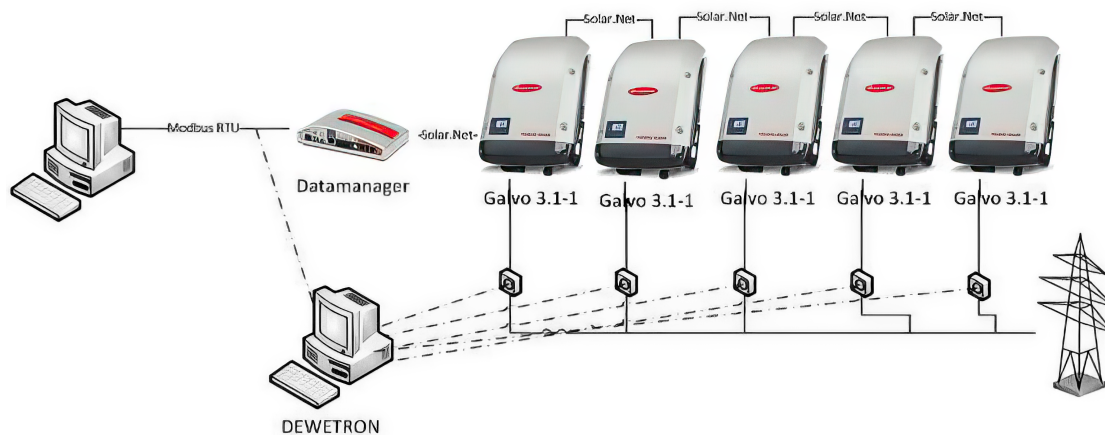


Figure 10. Test setup for remote controllable services.

The high level goal of the proof-of-concept validation was to prove the viability of the overall approach and the developed prototype system via concrete test cases. Prior to the execution of these tests, the test scenarios and the respective test environments to validate the proof-of-concept realisation needed to be defined.

Since the realisation had been split into three major parts, the test cases have accordingly been grouped into three major validation scenarios. An additional validation scenario had been defined in order to test the interoperability of the three parts of the developed system.

In the following sections, the major requirements of these validation scenarios, as well as the scenarios themselves and the results, are outlined. However, the application modelling and engineering aspect of the proof-of-concept validation has already been covered in detail in Reference [7] and is not the focus of this publication; therefore, it is not discussed here.

6. Performed Experiments and Achieved Results

To ensure targeted experimentation and meaningful results, a set of subsequent validation requirements (VRs) based on the functional requirements outlined in Section 3.1 was defined for each validation scenario. They are listed at the beginning of each scenario to give an impression of the intended outcomes.

6.1. Remote Controllable Services

For the basic [DER](#) service part of the proof-of-concept validation, the requirements have been identified as follows:

- [VR1-1](#): Ensure the creation of extended [DER](#) services, and
- [VR1-2](#): Ensure the integration of extended [DER](#) services in inverter-based [DER](#) devices.

The validation of the basic [DER](#) services, as well as the remote programmability, was performed in Fronius's system test laboratories. The test system comprises several network routes to connect inverters to each other, as well as to a remote supervisory control and data acquisition ([SCADA](#)) system.

The following two main validation scenarios have been realised:

- *PowerLimitChanged*: This scenario was used to validate the ability of the [DER](#) to remote control its AC power output. The inverter itself is remotely controlled via Modbus/[TCP](#) command; the power output is rated in steps of 1 percent of the in-

verter's nominal AC output power. The power reduction according to the remote signal was measured via a Dewetron AC network analyser.

- *CosPhiChanged*: This scenario was used to validate the ability of the DER to control its percentage distribution between active and reactive power output. The inverter itself is remotely controlled via Modbus/TCP commands; the power factor $\cos(\phi)$ is adjusted in steps of 10^{-2} between its lower and upper limits. The $\cos(\phi)$ delimitation according to the remote signal was again measured via the Dewetron AC network analyser.

Figure 10 provides an overview of the realised test setup for validating the basic DER services using 5 solar inverters connected to the Fronius datamanager box (DM-Box) controller equipped with the SmartOS-compliant prototype.

The solar inverters have been remotely controlled via Sunspec/Modbus messages. The time difference between Modbus signal and the reaction of the inverters was measured. Remote commands used for this experiment were startup and shutdown commands, as well as power reduction and $\cos(\phi)$ variation. In Figures 11–13 and Tables 1 and 2, the startup and shutdown measurements are provided along with the key characteristics.

Table 1. Measurement results: startup for 3 phases ($P = \max$, $\cos(\phi) = 1$).

Measurement	Time [s]
t (P_1)	56
t (P_3)	52
t (P_5)	35
Average	48

Table 2. Measurement results: shutdown.

Measurement	t (P_1) [ms]	t (P_3) [ms]	t (P_5) [ms]
1	725	677	702
2	839	790	815
3	775	757	778
4	794	745	780
5	832	814	838
Median		780	

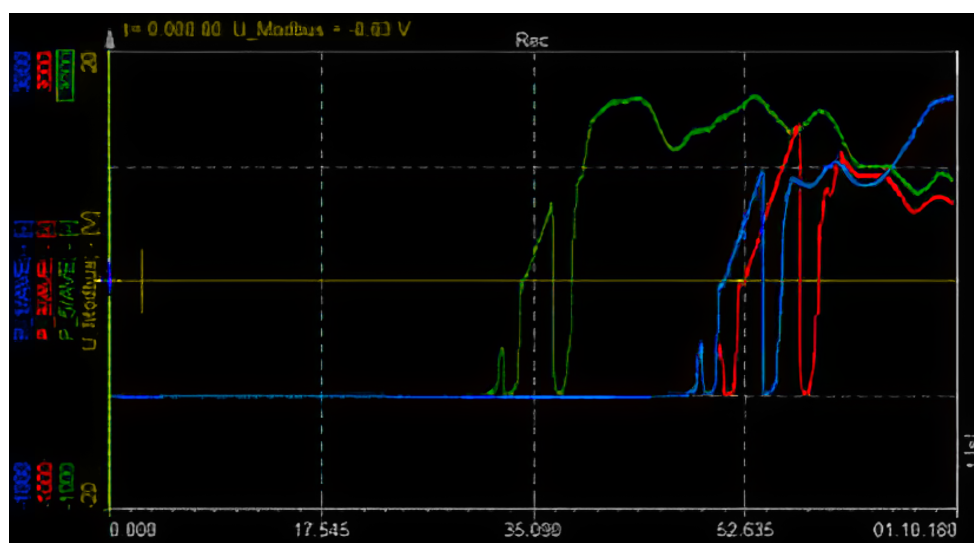


Figure 11. Startup for 3 phases (I).

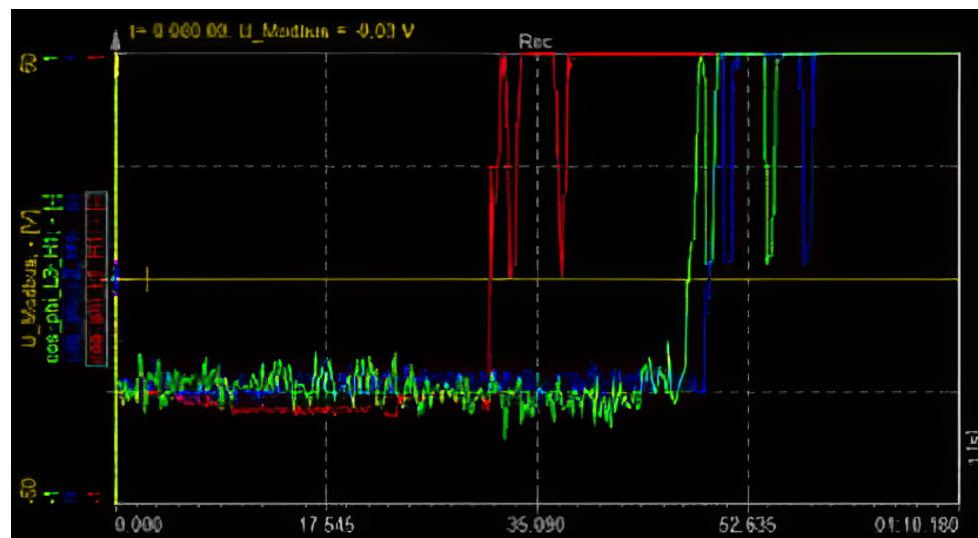


Figure 12. Startup for 3 phases (II).

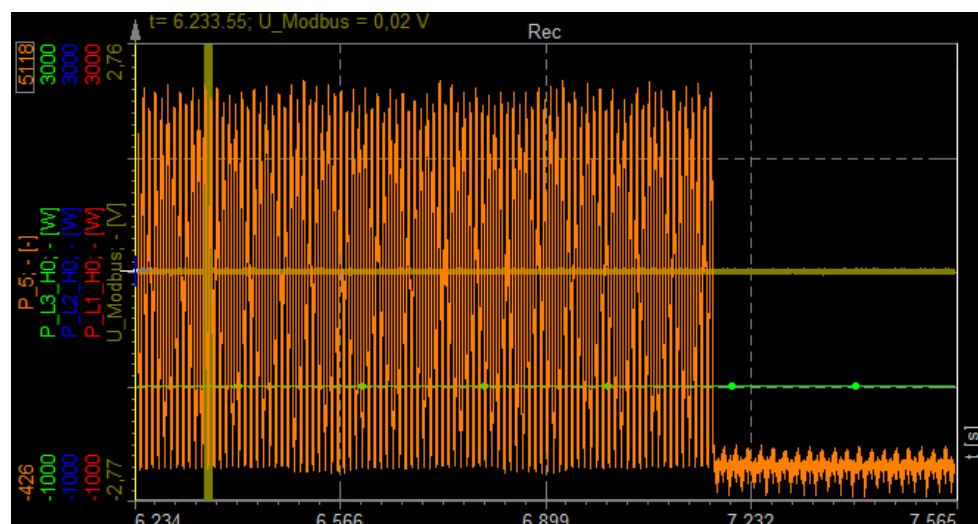


Figure 13. Shutdown scenario.

In summary, it can be said that this validation scenario has successfully proven that the remote control of DER services enables direct influence of the grid environment. It should be noted that inverter-based DER units (in this scenario, conjoined PV inverters) controlled remotely via SCADA can stabilise, but also destabilise, the energy grid. The use of such functionality must, therefore, be strictly controlled and performed with great care in order to avoid disturbances of the energy grid.

6.2. Generic and Interoperable Communication

For the communication part of the proof-of-concept validation, the requirements have been identified as follows:

- VR2-1: Ensure internal communication of applications,
- VR2-2: Ensure communication of applications on different systems (i.e., IEDs),
- VR2-3: Ensure security (especially confidentiality and integrity) of external communication,
- VR2-4: Ensure authentication methods and role-based rights management,
- VR2-5: Ensure interoperability with legacy devices (e.g., over Modbus),
- VR2-6: Ensure the seamless cooperation with high level engineering concepts, and
- VR2-7: Ensure flexibility and configurability of the communication subsystem.

The requirements VR2-3 and VR2-4 show that the request for integrating security means has been fulfilled appropriately. The approach here was to integrate security already by design, as described in Reference [5], to allow for the consideration of security related functionalities in the engineering process. As will be demonstrated, an appropriate realisation thereof has been developed as well.

Configuration of all communication related parameters (VR2-7) had to be done manually throughout the proof-of-concept validation. This concerns all sub-scenarios listed below. For obtaining the necessary test results, this is sufficient. For commercial solutions, a more practicable way of configuring the communication subsystem should of course be chosen.

The goal of this validation scenario is to demonstrate the functionality of the modular communication system which has been previously specified and realised. This necessitates a series of tests in order to involve all relevant components of the SmartOS. An overview of the involved components is given in Figure 14, where directly involved components are coloured orange, and indirectly involved components are coloured light orange.

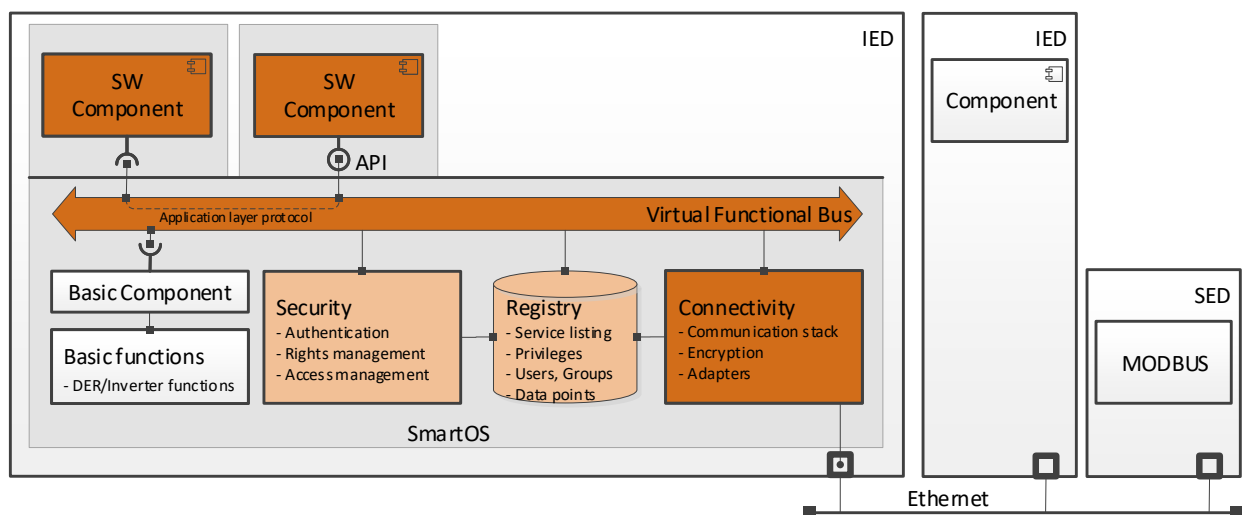


Figure 14. SmartOS components validated in scenario “Generic and Interoperable Communication” (adopted from Reference [4]).

Furthermore, an overview of the tested adapters and protocols, which are implemented as a part of the connectivity module, is provided in Figure 15. In order to allow incoming traffic for TCP/IP, secure TCP/IP and abstract syntax notation one (ASN.1), server modules are used. For Modbus, this has been omitted because the SmartOS only acts as a Modbus master in the conducted tests.

For this scenario, a series of increasingly complex tests has been devised, which have been designed to gradually increase and/or broaden the amount of components involved with each successive test. This keeps the amount of (added) complexity for each test case manageable, yet it allows for a systematic test of the entire communication system and a demonstration of its generic and interoperable nature. The scenarios are as follows:

- Two software components communicating locally: This minimal test demonstrates the basic functionality of the VFB and OpenALP, as well as the security and registry subsystems.
- Two software components on separate IEDs communicating over TCP/IP: Extends the previous scenario to include the connectivity module and its ancillary TCP/IP adapter. In addition, the communication pattern is changed from a push message to a request and the accompanying response.
- Two software components on separate IEDs communicating over ASN.1: This shows the transparency and interchangeability of the underlying protocol.

- (d) Two software components on separate IEDs communicating over secure TCP/IP: This extends the scope of scenarios b) and c) to include encryption and authentication (the latter utilising the trust centre). Furthermore, it can be shown that certificate revocation prevents a device from connecting to other devices.
- (e) A software component communicating with a legacy device via Modbus/TCP: Further demonstrates the transparency and interchangeability of the underlying protocol, as well as the support for legacy devices.

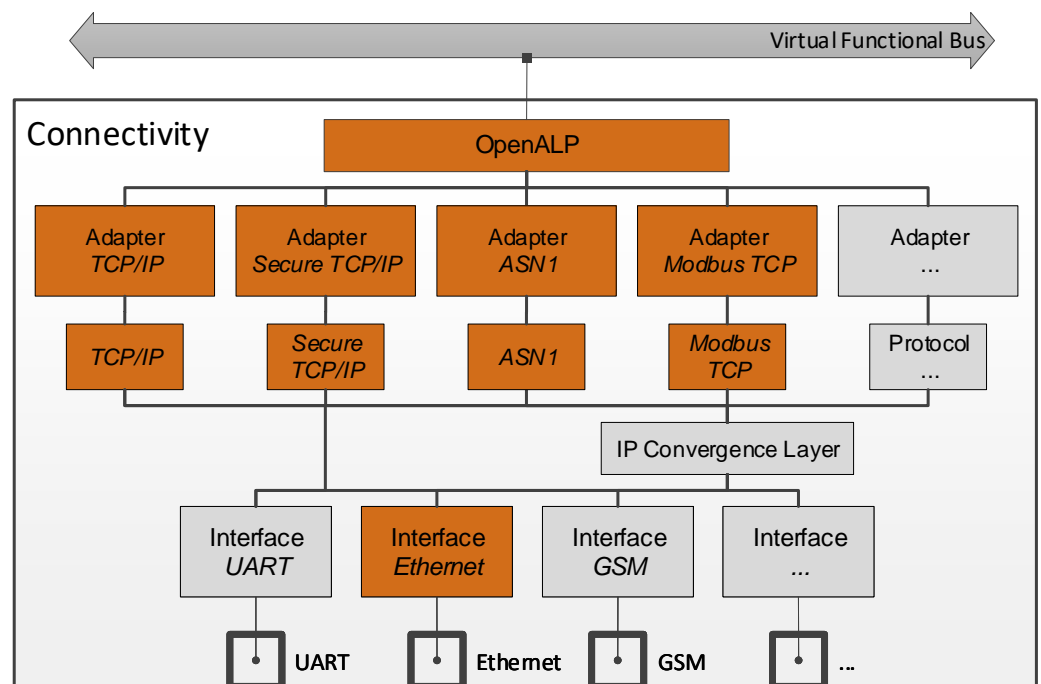


Figure 15. Connectivity module parts tested in scenario “Generic and Interoperable Communication” (adopted from Reference [4]).

The environment for these scenarios comprises a series of tests spanning one or more projects in the Eclipse integrated development environment (IDE). It is complemented by VPN access to the virtual testing environment, which hosts required services, such as the credential store. For test (e), a software-based Modbus slave has been created containing a process image from which data can be read and to which data can be written by a software component. Tests with a physical Modbus slave device have also been successfully performed and have yielded similar results. For test (a), a JUnit test invokes two software components (VFBAAppTalk1 and VFBAAppTalk2) running on the same IED. Upon initialisation, VFBAAppTalk1 sends a push message to VFBAAppTalk2. The successful completion of the test is confirmed via JUnit and console output, as can be seen in Figure 16.

For test (b), a JUnit test invokes 2 software components (VFBAAppTalk3 and VFBAAppTalk4) on 2 logically separate IEDs (hosted physically on the same machine). Upon initialisation, VFBAAppTalk3 sends a request message (a simulated meter reading) to VFBAAppTalk4, which then returns the appropriate response (a random integer). The successful completion of the test is confirmed via JUnit and console output, as can be seen in Figure 17.

```

Package ... JUnit
Device2AppsTalking.java VFBAppTalk1.java VFBAppTalk2.java
16 public void test2AppsTalkingSameDevice()

Finished after 2,237 seconds
Runs: 1/ Errors: 0 Failures: 0
test2AppsTalkingSameDevice

<terminated> Rerun at.fhsalzburg.its.opennes.integration.Device2AppsTalking.test2AppsTalkingSameDevice [JUnit]
Sample Software Component VFBAppTalk1 initialized
Sample Software Component VFBAppTalk2 initialized
at.fhsalzburg.its.opennes.servers.TCPIPServer waiting for request on port 8080...
at.fhsalzburg.its.opennes.servers.ASN1Server waiting for request on port 8081...
at.fhsalzburg.its.opennes.servers.SecureTCPIPServer waiting for request on port 8443...
Sample Software Component sending message:
Recipient: IED0/at.freeenergy.devices.apps.VFBAppTalk2$DataPoint1
Message content: Shutdown
Communication type: PUSH
Transaction ID: 1

Sample Software Component received message:
Sender: IED0/at.freeenergy.devices.apps.VFBAppTalk1
Transaction ID: 1
Message content: Shutdown
Communication type: PUSH
Sample Software Component received message:
Sender: IED0/at.freeenergy.devices.apps.VFBAppTalk2
Transaction ID: 2
Message content: Shutdown Talk2 added Text
Communication type: PUSH

```

Figure 16. Test of two software components communicating locally.

```

Package Exp... JUnit
Device2AppsTalking.java
44 public void test2AppsTalkingDifferentDevice() throws InterruptedException
45     final Device IED0 = new Device(new Registry("apps2Test/DeviceIED0/data
46     final Device IED1 = new Device(new Registry("apps2Test/DeviceIED1/data
47
48     (new Thread(new Runnable()
49     {
50     @Override

Finished after 5,196 seconds
Runs: 1/1 Errors: 0 Failures: 0
test2AppsTalkingDifferentDevice

<terminated> Rerun at.fhsalzburg.its.opennes.integration.Device2AppsTalking.test2AppsTalkingDifferentDevice [JUnit]
at.freeenergy.devices.apps.VFBAppTalk4 Sample Software Component initialized
at.fhsalzburg.its.opennes.servers.TCPIPServer waiting for request on port 8081...
at.fhsalzburg.its.opennes.servers.ASN1Server waiting for request on port 9091...
Sample Software Component VFBAppTalk3 initialized
at.fhsalzburg.its.opennes.servers.TCPIPServer waiting for request on port 8080...
at.fhsalzburg.its.opennes.servers.ASN1Server waiting for request on port 9090...
at.freeenergy.devices.apps.VFBAppTalk3 Sample Software Component sending message:
Recipient: IED1/at.freeenergy.devices.apps.VFBAppTalk4$DataPoint1
Transaction ID: 1
Message content: GET METERREADING
Communication type: REQUEST

at.fhsalzburg.its.opennes.servers.TCPIPServer accepted request at: 58504
at.fhsalzburg.its.opennes.servers.TCPIPServer waiting for request on port 8081...
at.freeenergy.devices.apps.VFBAppTalk4 Sample Software Component received message:
Sender: IED0/at.freeenergy.devices.apps.VFBAppTalk3
Transaction ID: 2
Message content: GET METERREADING
Communication type: REQUEST
Endpoint for ID: 2
at.fhsalzburg.its.opennes.servers.TCPIPServer accepted request at: 58505
at.fhsalzburg.its.opennes.servers.TCPIPServer waiting for request on port 8080...
at.freeenergy.devices.apps.VFBAppTalk3 Sample Software Component received message:
Sender: IED1/at.freeenergy.devices.apps.VFBAppTalk4
Transaction ID: 1
Message content: 301763374kWh
Communication type: RESPONSE

```

Figure 17. Test of two software components communicating remotely.

Test (c) uses an identical setup to test (b), but with a modified device configuration which prioritises [ASN.1](#) instead of [TCP/IP](#). The modified mapping tables are provided in [Figures 18 and 19](#) and the successful completion of the test is again confirmed via JUnit and console output.

opennesaddress	priority	adaptername	adapteraddress	datatype	protocol
IED0/at.freeenergy.devices.apps.VFBAppTalk3\$DataPoint1	2	SecureTCPIP	localhost,8443	java.lang.String	VFB
IED1/at.freeenergy.devices.apps.VFBAppTalk4\$DataPoint1	2	SecureTCPIP	localhost,9443	java.lang.String	VFB
IED0/at.freeenergy.devices.apps.VFBAppTalk3\$DataPoint1	1	ASN1	localhost,9090	java.lang.String	ASN1
IED1/at.freeenergy.devices.apps.VFBAppTalk4\$DataPoint1	1	ASN1	localhost,9091	java.lang.String	ASN1

Figure 18. Address mapping on intelligent electronic device (IED)0 during Test (c).

opennesaddress	priority	adaptername	adapteraddress	datatype	protocol
IED0/at.freeenergy.devices.apps.VFBAppTalk3\$DataPoint1	2	SecureTCPIP	localhost,8443	java.lang.String	VFB
IED1/at.freeenergy.devices.apps.VFBAppTalk4\$DataPoint1	2	SecureTCPIP	localhost,9443	java.lang.String	VFB
IED0/at.freeenergy.devices.apps.VFBAppTalk3	2	SecureTCPIP	localhost,8443	java.lang.String	VFB
IED1/at.freeenergy.devices.apps.VFBAppTalk4	2	SecureTCPIP	localhost,9443	java.lang.String	VFB
IED0/at.freeenergy.devices.apps.VFBAppTalk3\$DataPoint1	1	ASN1	localhost,9090	java.lang.String	ASN1
IED1/at.freeenergy.devices.apps.VFBAppTalk4\$DataPoint1	1	ASN1	localhost,9091	java.lang.String	ASN1
IED0/at.freeenergy.devices.apps.VFBAppTalk3	1	ASN1	localhost,9090	java.lang.String	ASN1
IED1/at.freeenergy.devices.apps.VFBAppTalk4	1	ASN1	localhost,9091	java.lang.String	ASN1

Figure 19. Address mapping on IED1 during Test (c).

Test (d) again uses an identical setup to tests (b) and (c) and the device configuration has again been modified to prioritise secure **TCP/IP**. First, valid certificates are used and the test is successfully completed. Thereafter, the certificate used on **IED0** is revoked on the trust centre. As a result, the incoming request is refused by **IED1**, and the unit test fails (see Figure 20), which, in this case, is the expected and desired result.

```

46 final Device IED0 = new Device(new Registry("apps2Test/DeviceIED0/data/regi
47 final Device IED1 = new Device(new Registry("apps2Test/DeviceIED1/data/regi
48
49 (new Thread(new Runnable()
50 {
51     @Override
52     public void run() { IED1.startDevice(); }
53 }).start();

```

```

<terminated> Rerun at fhsalzburg.its.opennes.integration.Device2AppsTalking.test2AppsTalkingDifferentDevice [JUnit]
at.freeenergy.devices.apps.VFBAppTalk4 Sample Software Component initialized
at.fhsalzburg.its.opennes.servers.ASN1Server waiting for request on port 9091...
at.fhsalzburg.its.opennes.servers.SecureTCPIPServer waiting for request on port 9443...
Sample Software Component VFBAppTalk3 initialized
at.fhsalzburg.its.opennes.servers.ASN1Server waiting for request on port 9090...
at.fhsalzburg.its.opennes.servers.SecureTCPIPServer waiting for request on port 8443...
at.freeenergy.devices.apps.VFBAppTalk3 Sample Software Component sending message:
Recipient: IED1/at.freeenergy.devices.apps.VFBAppTalk4$DataPoint1
Transaction ID: 1
Message content: GET METERREADING
Communication type: REQUEST

at.fhsalzburg.its.opennes.servers.SecureTCPIPServer accepted request at: 58374
Connection refused: Invalid certificate.
at.fhsalzburg.its.opennes.servers.SecureTCPIPServer waiting for request on port 9443...

```

Figure 20. Refused request due to invalid certificate.

For test (e), the Modbus library **Modbus4J** has been utilised to provide a Modbus slave containing a process image from which data can be read and to which data can be written. Thereafter, a software component (**SimpleVFBApp3**) has been created to send a series of messages to the Modbus slave, and the device registry on the **IED** running **SimpleVFBApp3** has been configured to provide the address and data point of the Modbus slave as **IED4/sampleNamespace\$SampleModbusValue**. Upon initialisation of the **IED**, **SimpleVFBApp3** is invoked and sequentially performs three tasks: First, a request message is sent to read the value of the aforementioned data point. Second, a push message is sent to set a different value for this data point. Last, a request message is sent once again to confirm that the value has indeed been set. The successful completion of the test is again confirmed via JUnit and console output.

Summarising this scenario, it can be said that all tests could be completed successfully, and the goals regarding generic and interoperable communication have been achieved. Although the tests have certainly not been as rigorous as would be required for a market-ready product, they are sufficient to validate the effectiveness of the solutions that have been developed for the critical challenges. Further work towards the creation of a final product would predominantly involve fine-tuning and optimising and is not expected to encounter any major challenges for which no well-established solutions exist.

6.3. System Integration

The requirements for the system integration test cover in principle all the above outlined requirements. However, especially for the integration testing of the proof-of-concept validation, the following additional requirements have been identified:

- VR3-1: Ensure the safe and secure execution of DER services, and
- VR3-2: Ensure the safe and secure execution of energy/DER applications.

The previous scenarios all resulted in successful validations of the main parts of the proposed approach: (i) the formal application modelling concept (see Reference [7]), (ii) the remote programmable DER services, and (iii) the generic and interoperable communication. With these validations as a basis, the final scenario shows a system integration validation. In other words, the main intention is to use the application modelling approach to create and use remote DER functions with the help of the generic and modular communication system.

This scenario is also intended to be a validation of the overall idea of the approach, as shown in Figure 4 and as it has also been used in Reference [7] for the validation of the application modelling concept. It is assumed that the overall application has been modelled using the formal modelling approach. Furthermore, it is assumed that the remote programmable DER services are available and that the other components of the SmartOS are operational. The last precondition is, that it is possible to communicate between DERs using the Connectivity component (i.e., the generic and modular communication concept tested in Section 6.2). Consequently, what is left to validate is the integration of these parts with each other. First, it has to be shown, how an application can be deployed, configured, and started on a DER device, once it has been modelled. Second, it has to be shown, how the communication between different devices can be implemented and properly configured.

For this validation, the following test case is used: A voltage change in the grid of a distribution system operator (DSO) causes the voltage spread to increase above its allowed threshold. This causes the control algorithm in VoltVArController seen in Figure 21 to calculate a new Volt-var droop curve for the DERController. With the new curve, the DERController will be more sensitive to voltage deviations. This, in turn, will decrease the voltage spread.

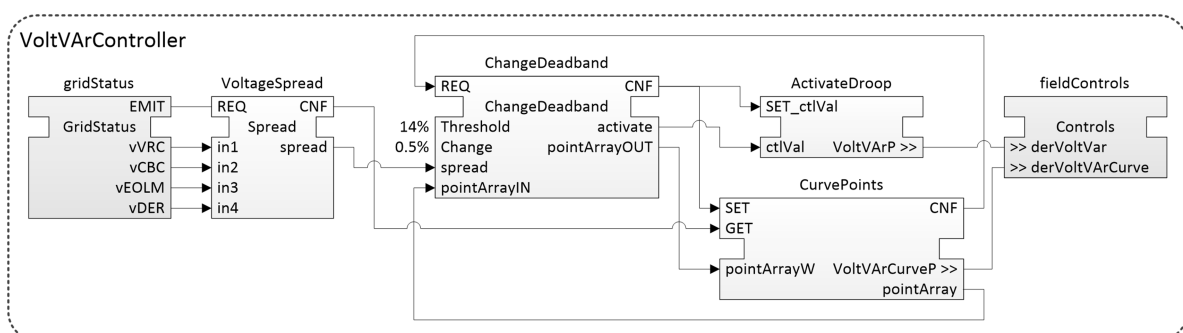


Figure 21. Implementation of the VoltVArController application in IEC 61499 [7].

Compared to the previous scenarios, this test case shows how the envisioned method is used to deploy an application to real devices. Furthermore, it also includes communication

between multiple devices using different communication protocols, which need to be configured. Finally, the goal is also to validate the implementation of the VoltVArController function as depicted in Figure 21.

A scheme of the used parts of the AIT SmartEST lab setup is illustrated in Figure 22. It also shows on which components the different IEC 61499 functions are deployed. The VoltVArController function, as shown in Figure 21, is deployed on the DSOComputer.

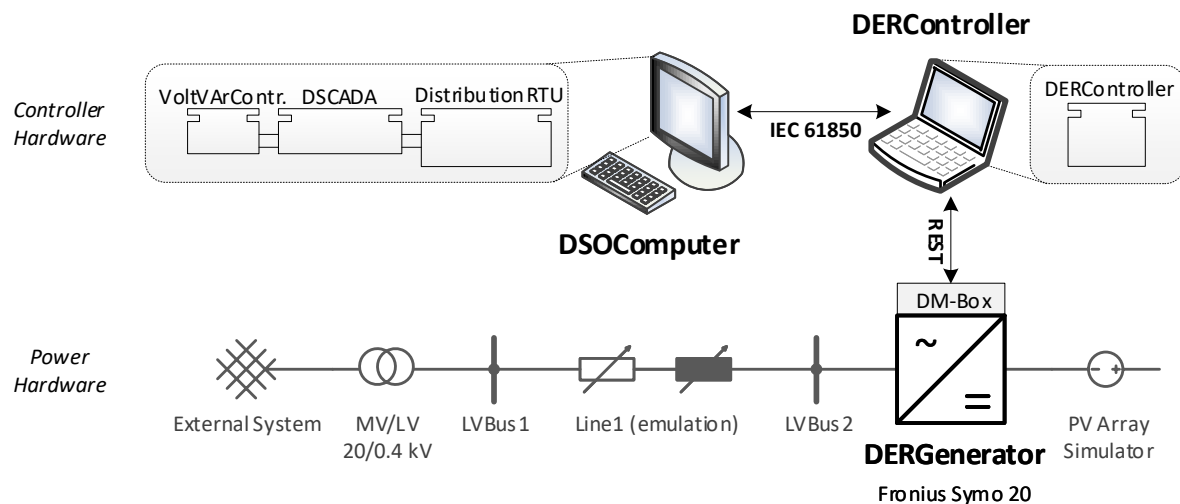


Figure 22. Laboratory setup for the validation use case (adopted from Reference [7]).

The controller hardware used in the laboratory consists of a desktop PC representing the DSOComputer and a laptop PC running the SmartOS and the DERController App. The DERGenerator is a commercial off-the-shelf Fronius Symo 20 PV inverter; on the DC side, it is connected to a PV array simulator, and, on the AC side, to the LV grid. A line impedance had to be emulated by the laboratory equipment in order to create a dependency between the voltage and the inverter output. Natively, the inverter offers a Modbus/TCP control and measurement interface. However, for this test, a representational state transfer (REST) interface was added to facilitate communication between the inverter and the SmartOS App. On the laptop, the SmartOS was executed together with 4diac [60]. The setup of the SmartOS is shown in Figure 23.

In order to use the programmability of 4diac, it was integrated with the VFB of the SmartOS. First, an ASN.1 adapter was created for the Connectivity component. This adapter allows communication between software components and the 4diac runtime environment (FORTE). This communication is shown in Figure 23 as a dashed red line. However, in order to change the reactive power, a command must be sent to the basic functions component running on the Fronius DM-Box. For this, the next step was to create a dedicated software component that forwards messages from the FORTE to the DM-Box. This means that, if the DERController function in the FORTE wants to send a new reactive power setpoint to the inverter, it is first sent to the 4diac-App using ASN.1 prior to being forwarded to the basic functions component using the REST protocol (represented as a dashed blue line in Figure 23). Finally, the DM-Box makes sure that the reactive power setting is applied by the inverter-based DER.

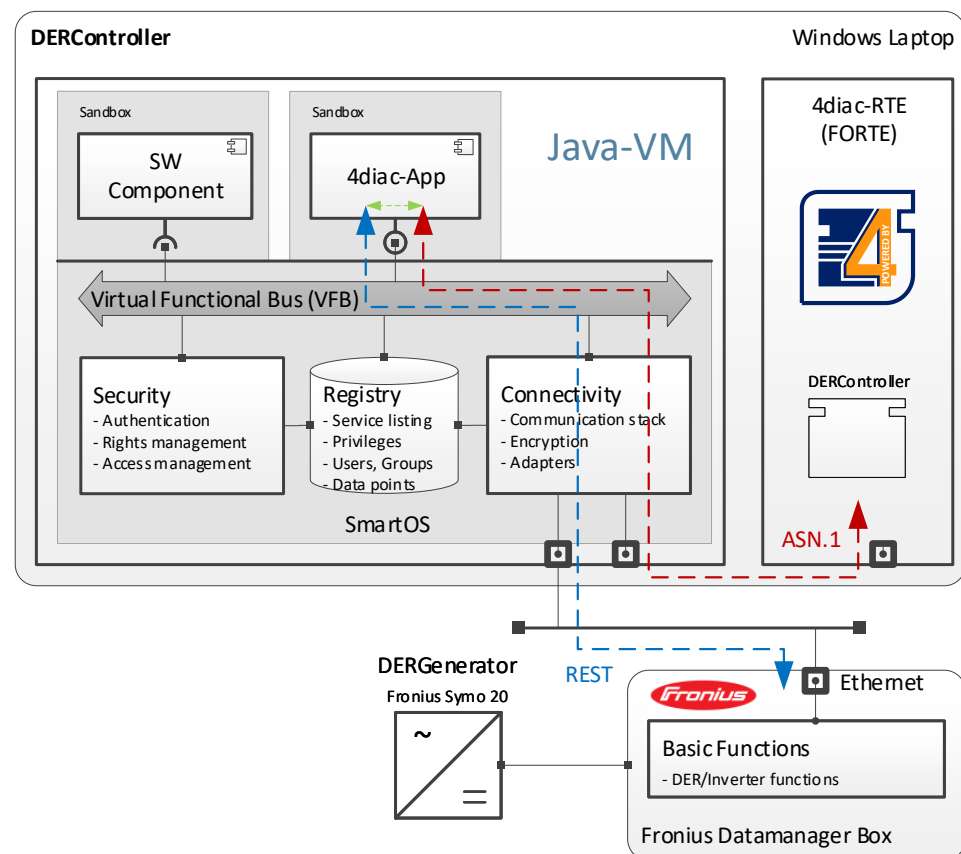


Figure 23. Setup of the SmartOS and 4diac on the distributed energy resources (DER) Controller laptop.

Before the laboratory validation could be started, the application had to be deployed. The first precondition for this is that the inverter is connected and feeding power to the grid. Second, the FORTE must be executed on all the components and awaiting a deployment from the 4diac-IDE. The next step was to deploy the communication configurations.

Next, the communication infrastructure was generated for connections between IEC 61499 functions running on different devices. Generated were both communication FBs, as well as communication configurations. For this test case, all the interfaces are derived from IEC 61850, which results in generation of substation configuration description language (SCL) files for the communication configuration. The resulting IEC 61850 configuration for the DERController is seen in Figure 24. As is seen, the IEC 61850 SCL configuration adopts information from the IEC 61499 application (e.g., the IEC 61850 IED is named DERController after the IEC 61499 device).

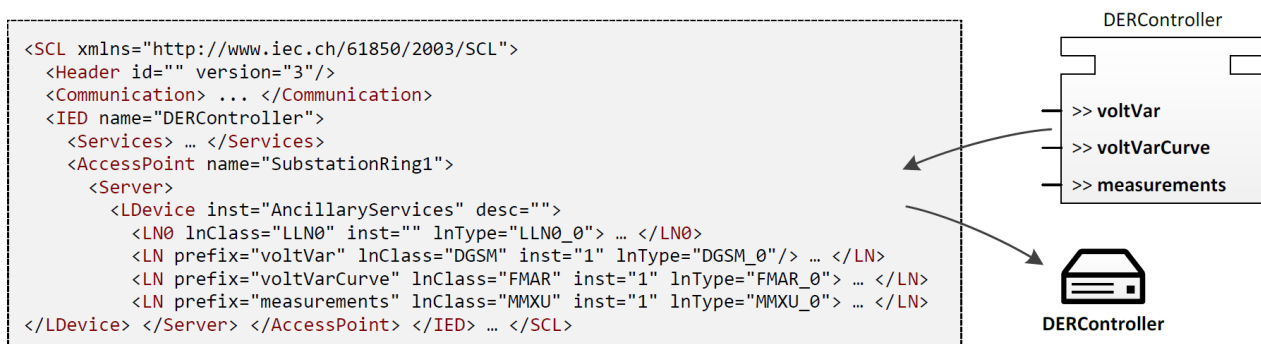


Figure 24. Generated substation configuration description language (SCL) file for configuration of the IEC 61850 server in DER Controller [7].

After this, the SCL files and the IEC 61499 applications are deployed to their components according to Figure 22. This is a built-in feature of 4diac and utilises standard IEC 61499 methods; thus, no new code has to be generated from the IEC 61499 model. After deployment completion, the application is automatically started; hereby, the SCL files are loaded and the IEC 61850 communication is initialised. For example, the IEC 61850 client in DistributionRTU connects to the IEC 61850 server on the DERController.

Subsequently, the VoltVarController algorithm can be validated. For this validation, the inverter is configured to an active power output of 18 kW and a reactive power output of 0 kVAr. The voltage measured by the inverter (i.e., the DERGenerator) is forwarded to the VoltVarController at the vDER data output of the gridStatus FB (see Figure 21). The other voltage measurements (i.e., vVRC, vCBC, and cEOLM) are all emulated and fixed to 0.9 per unit (p.u.). After stabilisation, this results in a vDER voltage of around 1.004 p.u., and no extra reactive power. This is seen at the beginning of the time series in the top graph of Figure 25, where Q is the reactive power of the inverter, and U is the measured voltage by the inverter (i.e., vDER).

An increased voltage spread was emulated to trigger the VoltVarController algorithm, seen as the first event (i) in Figure 25. At this event, the fixed voltage of vVRC was changed from 0.9 p.u. to 0.86 p.u. This increases the voltage spread ($U_{\max} - U_{\min}$, see Figure 25) above the allowed threshold. This is detected by the VoltVarController's algorithm. New Volt/var curve parameters are calculated by the ChangeDeadband FB in Figure 21. The new curve parameters are sent to the DERController, which results in a new reactive power set point for the DERGenerator. With the new Volt/var parameters, the inverter starts producing reactive power. This is shown as event (ii) in Figure 25. However, since the voltage spread is still too high, a second correction of the Volt/var parameters is performed by the VoltVarController after 185 s, which is visualised in Figure 25 as event (iii).

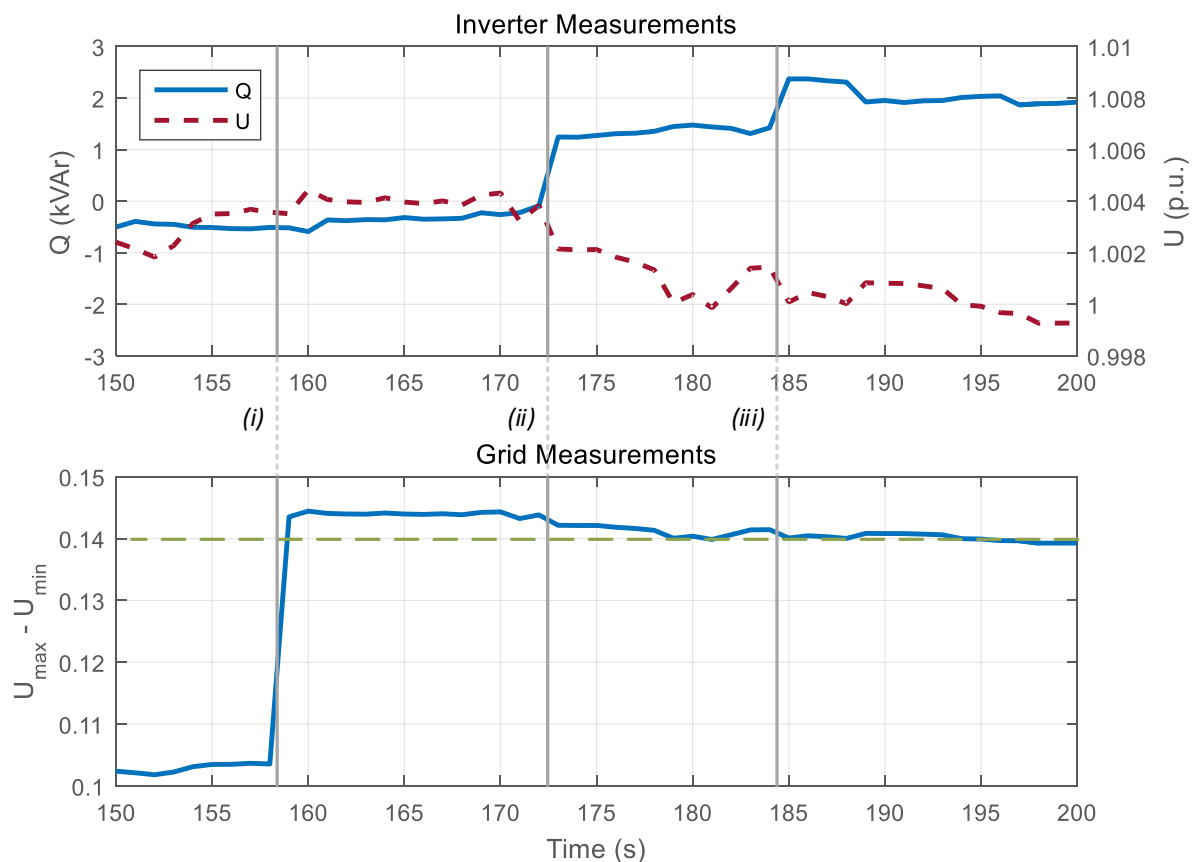


Figure 25. Measurements from the laboratory validation [7].

In general, the results of this experiment indicate a successful integration of the different parts of the proposed approach. This scenario shows that even with the prototypical implementation, it is possible to achieve an open and interoperable **ICT** solution for the integration of **DERs**. For productive use, many adjustments would of course have to be made to the setup shown in Figure 22 (such as running the SmartOS on the **DM-Box** and integrating the **FORTE** into the 4diac App).

Regarding the results of the test, especially the measurements shown in Figure 25, more evidence of the prototypical nature of the implementation is seen. In an industrial implementation, these changes would have to be in the range of milliseconds, whereas they are in the range of seconds in this case. This can be explained by the suboptimal test setup shown in Figure 22, as well as a complete absence of code optimisation in that regard. Nevertheless, the results demonstrate the basic usefulness of the presented concept. For industrial adoption, the reaction time has yet to be improved to the required level.

6.4. Reflection of Results

The achieved results show that the proposed architecture helps to fill the gaps of current solutions and approaches, as identified in Section 2.4, in the following way:

- As outlined in Section 3.4 and further explained in Reference [7], this work provides an approach for a common application modelling concept for power and energy systems automation applications.
- By supporting a wide range of communication protocols, the developed architecture provides a unifying method of integrating renewable energy sources into smart grids. It allows to build open and scalable smart grid automation applications in a hardware and platform independent way.
- The introduction of **RCSs** makes it possible to reuse common functionalities and to update and extend **DER** services across the entire lifecycle of **DER** components.

The specific system performance and level of compliance to timing requirements will depend on code optimisation, as well as the underlying protocols. It will, therefore, be need to be analysed on a case-by-case basis, if given requirements can be met using given protocols.

For real-world solutions, safe handling of potential controller conflicts is an essential requirement. An approach to a potential solution is provided in Reference [61].

Although cyber-security issues have not been the primary focus of the work at hand, security-by-design principles as outlined in Reference [5] have been applied during the development of the architecture, resulting in a fundamentally robust system. As our solution aims for a maximum of protocol independence, the integration of a more comprehensive security architecture is easily possible if needed.

The proposed architecture and proof-of-concept prototype is currently being further developed into an industry grade solution. This entails, among other things, the addition of support for additional protocols and the realisation of a fully-fledged trust centre/credential store, as well as code optimisation. Prior to its roll-out and use in real-world scenarios, it will further have to undergo extensive testing (conformity, security/penetration, deployment and acceptance tests, etc.).

7. Conclusions

As a summary, it could be demonstrated that the presented approach of an open **ICT** infrastructure for the integration of renewable energy sources, especially **DER**, into a smart power grid may be used for producing feasible working prototypes. The presented solution is able to provide necessary means of communication to smart grid applications. The central aspects here include the definition of a unique addressing scheme, as well as the provision of an application layer protocol, for control services (**OpenALP**), which may again be used by smart grid applications. Hereby, legacy protocols are used as much as possible; only functionality necessary for the correct working of **OpenALP** is additionally provided in the form of protocol wrappers. Furthermore, a tool (**PSAL**) to facilitate the

engineering of remote programmable services to be used in the presented ecosystem has also been provided. To test the approach, the developed prototype has been validated in a number of concrete validation cases. It was not a goal of the validation to provide a fully-fledged communication middleware—thus, a prototype containing the minimum required functions has been realised.

As has been shown, with the open definition of the address scheme in combination with [OpenALP](#), it is easily possible for vendors of grid related middleware frameworks to integrate the demonstrated functionalities. Thus, the solution at hand does not intend to provide yet another framework but to suggest an open ecosystem which may be used by several vendors. Consequently, the realised testbed was used in order to prove the concept at hand, rather than to provide an already optimised solution. Nevertheless, the results demonstrate the feasibility of the approach, even when not optimised regarding performance. Thus, it can be expected that a fully engineered version of the presented functionalities would clearly contribute to an efficient and easy to use tooling for integrating renewables into smart grids, especially dedicated to [LV](#) grids. Even more, as the concept has been set up in a very generic way, without clear bindings on the power and energy systems domain, it can be expected that the presented solution may provide benefits to other sectors, as well, especially for those industry domains, where massively distributed control will play an important role in the near future.

Funding: This research work received funding by the Austrian Ministry for Transport, Innovation and Technology (bmvit) and the Austrian Research Promotion Agency (FFG) under the “[ICT of the Future](#)” program in the OpenNES project (FFG No. 845632). Contributions have also been performed within the project InterGrid, which is funded by the State of Upper Austria via the FFG under the contract number 881296.

Author Contributions: All authors jointly developed the overall approach, including the prototype implementation and the validation of it. Moreover, A.V. defined the overall paper structure and provided the main parts of Sections 1–4, as well as the abstract and the conclusion. O.L. wrote the main parts of Sections 5 and 6 and contributed to the readability of the entire paper. F.P.A. edited the description of the engineering part of the prototype, especially but not only, in Section 3. C.K. brought in his experience on practical uses of [ICT](#) infrastructures and integrated information on the relevant requirements. T.I.S. contributed to the overall structure, validated the content of the relevant sections, and provided a proof reading of the total paper. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AMI	advanced metering infrastructures
API	application programming interface
ARQ	automatic repeat request
ASN.1	abstract syntax notation one
CA	certificate authority
CAN	controller area network
CIM	common information model
CORBA	common object request broker architecture
COSEM	companion specification for energy metering
DDS	data distribution service
DER	distributed energy resources

DLMS	device language message specification
DM-Box	datamanager box
DSL	domain specific language
DSO	distribution system operator
EBNF	extended Backus Naur form
EJBCA	enterprise JavaBeans certificate authority
FB	function block
FIFO	first in first out
FORTE	4diac runtime environment
GWAC	GridWise architecture council
HA/BA	home and building automation
ICT	information and communication technology
IDE	integrated development environment
IEC	International Electrotechnical Commission
IED	intelligent electronic device
IMAP	Internet message access protocol
IoT	Internet of things
IP	Internet protocol
ISO	International Organisation for Standardisation
LD	logical device
LDAP	lightweight directory access protocol
LV	low voltage
MBSE	model-based system engineering
MDA	model-driven architecture
MIME	multipurpose Internet mail extensions
MOM	message oriented middleware
OPC UA	open process control unified architecture
OpenALP	open application layer protocol
OpenLDAP	open lightweight directory access protocol
OS	operating system
OSGi	Open Services Gateway initiative
OSI	open systems interconnection
PDP	policy decision point
PEP	policy enforcement point
PIM	platform independent model
PKI	public key infrastructure

PLC	powerline communication
POP3	post office protocol 3
PSAL	power system automation language
PV	photovoltaics
RBAC	role-based access control
RCS	remote controllable service
REST	representational state transfer
RTU	remote terminal unit
SAP	service access point
SCADA	supervisory control and data acquisition
SCL	substation configuration description language
SGAM	smart grid architecture model
SIFB	service interface function block
SIP	session initiation protocol
SMTP	simple mail transfer protocol
TCP	transmission control protocol
TSN	time sensitive network
UDP	user datagram protocol
UML	unified modelling language
VFB	virtual functional bus
VPN	virtually private network
VR	validation requirement
XMPP	extensible messaging and presence protocol

References

1. The European Parliament and the Council of the European Union. Directive 2009/28/EC. Available online: <https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32009L0028> (accessed on 25 January 2021).
2. Kupzog, F.; Dimitriou, P.; Faschang, M.; Mosshammer, R.; Stifter, M.; Andrén, F. Co-Simulation of Power- and Communication-Networks for Low Voltage Smart Grid Control. In Proceedings of the 1st D-A-CH Energieinformatik 2012, Oldenburg, Germany, 5–6 July 2012.
3. Budka, K.C.; Deshpande, J.G.; Thottan, M. *Communication Networks for Smart Grids*; Springer: London, UK, 2016.
4. Prörtl Andrén, F.; Strasser, T.; Langthaler, O.; Veichtlbauer, A.; Kasberger, C.; Felbauer, G. Open and Interoperable ICT Solution for Integrating Distributed Energy Resources into Smart Grids. In Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), Berlin, Germany, 6–9 September 2016.
5. Veichtlbauer, A.; Langthaler, O.; Engel, D.; Kasberger, C.; Prörtl Andrén, F.; Strasser, T. Towards Applied Security-by-Design for DER Units. In Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), Berlin, Germany, 6–9 September 2016.
6. Veichtlbauer, A.; Parfante, M.; Langthaler, O.; Prörtl Andrén, F.; Strasser, T. Evaluating XMPP Communication in IEC 61499-based Distributed Energy Applications. In Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), Berlin, Germany, 6–9 September 2016.
7. Prörtl Andrén, F.; Strasser, T.I.; Kastner, W. Engineering Smart Grids: Applying Model-driven Development from Use Case Design to Deployment. *Energies* **2017**, *10*, 374.

8. Bründlinger, R.; Strasser, T.; Lauss, G.; Hoke, A.; Chakraborty, S.; Martin, G.; Kroposki, B.; Johnson, J.; de Jong, E. Lab Tests: Verifying That Smart Grid Power Converters Are Truly Smart. *IEEE Power Energy Mag.* **2015**, *13*, 30–42, doi:10.1109/MPE.2014.2379935.
9. Kupzog, F.; Veichtlbauer, A.; Heinisch, A.; von Tüllenburger, F.; Langthaler, O.; Pache, U.; Jung, O.; Frank, R.; Dorfinger, P. The Impact of Virtualisation Techniques on Power System Control Networks. *Electronics* **2020**, *9*, 1433–1454, doi:10.3390/electronics9091433.
10. SMB Smart Grid Strategic Group (SG3). *IEC Smart Grid Standardization Roadmap*; Technical Report; International Electrotechnical Commission (IEC): Geneva, Switzerland, 2010.
11. Huang, A.Q.; Crow, M.L.; Heydt, G.T.; Zheng, J.P.; Dale, S.J. The Future Renewable Electric Energy Delivery and Management (FREEDM) System: The Energy Internet. *Proc. IEEE* **2011**, *99*, 133–148, doi:10.1109/JPROC.2010.2081330.
12. Iqtiyanillham, N.; Hasanuzzaman, M.; Hosenuzzaman, M. European smart grid prospects, policies, and challenges. *Renew. Sustain. Energy Rev.* **2017**, *67*, 776–790.
13. Romero Aguero, J.; Khodaei, A. Grid Modernization, DER Integration & Utility Business Models—Trends Challenges. *IEEE Power Energy Mag.* **2018**, *16*, 112–121.
14. Rohjans, S.; Danekas, C.; Uslar, M. Requirements for Smart Grid ICT-architectures. In Proceedings of the 2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe), Berlin, Germany, 14–17 October 2012; pp. 1–8, doi:10.1109/ISGTEurope.2012.6465617.
15. Kanabar, M.G.; Voloh, I.; McGinn, D. Reviewing smart grid standards for protection, control, and monitoring applications. In Proceedings of the 2012 IEEE PES Innovative Smart Grid Technologies (ISGT), Washington, DC, USA, 16–20 January 2012; pp. 1–8.
16. Nafi, N.S.; Ahmed, K.; Gregory, M.A.; Datta, M. A survey of smart grid architectures, applications, benefits and standardization. *J. Netw. Comput. Appl.* **2016**, *76*, 23–36.
17. International Electrotechnical Commission. *IEC 61970-301: Energy Management System Application Program Interface (EMS-API)—Part 301*; IEC: Geneva, Switzerland, 2016.
18. International Electrotechnical Commission. *IEC 61850-1: Communication Networks and Systems for Power Utility Automation—Part 1*; IEC: Geneva, Switzerland, 2013.
19. Gungor, V.C.; Sahin, D.; Kocak, T.; Ergut, S.; Buccella, C.; Cecati, C.; Hancke, G.P. Smart Grid Technologies: Communication Technologies and Standards. *IEEE Trans. Ind. Inform.* **2011**, *7*, 529–539, doi:10.1109/TII.2011.2166794.
20. IEC Central Office. *The Smart Grid Standards Map*; IEC: Geneva, Switzerland, 2019.
21. Institute of Electrical and Electronics Engineers (IEEE). *IEEE Guide for Smart Grid Interoperability of Energy Technology and Information Technology Operation with the Electric Power System (EPS), End-Use Applications, and Loads*; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2011.
22. Smart Grid Coordination Group. *Smart Grid Reference Architecture*; Technical Report; CEN/Cenelec/ETSI Smart Grid Coordination Group: Brussels, Belgium, 2012.
23. The GridWise Architecture Council. *GridWise Interoperability Context-Setting Framework*; Technical Report; The GridWise Architecture Council: Richland, WA, USA, 2008.
24. International Organization for Standardization. *ISO/IEC 7498-1:1994 Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*; ISO: Geneva, Switzerland, 1994.
25. International Electrotechnical Commission. *IEC 61131-3: Programmable Controllers—Part 3: Programming Languages*; IEC: Geneva, Switzerland, 2012.
26. International Electrotechnical Commission. *IEC 61499: Function Blocks*; IEC: Geneva, Switzerland, 2012.
27. Strasser, T.; Zoitl, A.; Christensen, J.; Sünder, C. Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems. *IEEE Trans. Syst. Man, Cybern. Part C Appl. Rev.* **2010**, *41*, 41–51, doi:10.1109/TSMCC.2010.2067210.
28. Veichtlbauer, A.; Pfeiffenberger, T. Generic Middleware for User-friendly Control Systems in Home and Building Automation. *Int. J. Adv. Networks Serv.* **2013**, *6*, 51–67.
29. Fraunhofer. *OGEMA: Open Gateway Energy Management*; Technical Report; Fraunhofer IWES: Bremerhaven, Germany, 2012.
30. Broy, M.; Gleirscher, M.; Merenda, S.; Wild, D.; Kluge, P.; Krenzer, W. Toward a Holistic and Standardized Automotive Architecture Description. *Computer* **2009**, *42*, 98–101, doi:10.1109/MC.2009.413.
31. International Electrotechnical Commission. *IEC 62056: Electricity Metering Data Exchange—The DLMS/COSEM Suite*; IEC: Geneva, Switzerland, 2014.
32. Veichtlbauer, A.; Engel, D.; Knirsch, F.; Langthaler, O.; Moser, F. Advanced Metering and Data Access Infrastructures in Smart Grid Environments. In Proceedings of the 7th International Conference on Sensor Technologies and Applications (SensorComm 2013), Barcelona, Spain, 25–31 August 2013.
33. The openHAB Community and the openHAB Foundation e.V. *Welcome to openHAB*; Technical Report; The openHAB Community and the openHAB Foundation e.V.: Ober-Ramstadt, Germany, 2019.
34. The OSGi Alliance. *OSGi Core Release 5*; OSGi Alliance: San Ramon, CA, USA, 2012.
35. Pichler, M.; Veichtlbauer, A.; Engel, D. Evaluation of OSGi-based Architectures for Customer Energy Management Systems. In Proceedings of the 2015 IEEE International Conference on Industrial Technology (ICIT2015), Seville, Spain, 17–19 March 2015.
36. Zeadally, S.; Kubher, P. Internet access to heterogeneous home area network devices with an OSGi-based residential gateway. *Int. J. Ad Hoc Ubiquitous Comput.* **2008**, *3*, 48–56.

37. Object Management Group. *Common Object Request Broker Architecture (CORBA) Specification*; Technical Report; Object Management Group: Milford, MA, USA, 2015.
38. Object Management Group. *Data Distribution Service (DDS)*; Technical Report; Object Management Group: Milford, MA, USA, 2015.
39. OPC Foundation. *OPC—The Interoperability Standard for Industrial Automation & Other*; OPC Foundation: Scottsdale, AZ, USA, 2012.
40. Sermersheim, J. Lightweight Directory Access Protocol (LDAP): The Protocol, RFC4511. Available online: <https://www.rfc-editor.org/info/rfc4511> (accessed on 19 February 2021).
41. Freed, N.; Borenstein, N. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC 2045. Available online: <https://www.rfc-editor.org/info/rfc2045> (accessed on 19 February 2021).
42. Desruisseaux, B. Internet Calendaring and Scheduling Core Object Specification (iCalendar), RFC5545. Available online: <https://www.rfc-editor.org/info/rfc5545> (accessed on 19 February 2021).
43. International Electrotechnical Commission. *IEC 60870-1: Telecontrol Equipment and Systems—Part 1: General Considerations*; IEC: Geneva, Switzerland, 1988.
44. Rohjans, S. (S2)In—Semantic Service Integration for Smart Grids. Ph.D. Thesis, Carl von Ossietzky University, Oldenburg, Germany, 2012.
45. Alkhawaja, A.R.; Ferreira, L.L.; Albano, M. Message Oriented Middleware with QoS Support for Smart Grids. In Proceedings of the INForum 2012 Conference on Embedded Systems and Real Time, Caparica, Portugal, 6–7 October 2012.
46. Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; Johnston, A.; Peterson, J.; Sparks, R.; Handley, M.; Schooler, E. SIP: Session Initiation Protocol, RFC3261. Available online: <https://www.rfc-editor.org/info/rfc3261> (accessed on 19 February 2021).
47. Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Core, RFC6120. Available online: <https://www.rfc-editor.org/info/rfc6120> (accessed on 19 February 2021).
48. Postel, J. User Datagram Protocol, RFC 768. Available online: <https://www.rfc-editor.org/info/rfc768> (accessed on 19 February 2021).
49. Postel, J. Transmission Control Protocol—DARPA Internet Program Protocol Specification, RFC 793. Available online: <https://www.rfc-editor.org/info/rfc793> (accessed on 19 February 2021).
50. Deering, S.; Hinden, R. Internet Protocol, Version 6 (IPv6) Specification, RFC8200. Available online: <https://www.rfc-editor.org/info/rfc8200> (accessed on 19 February 2021).
51. Postel, J. Internet Protocol—DARPA Internet Program Protocol Specification, RFC 791. Available online: <https://www.rfc-editor.org/info/rfc791> (accessed on 19 February 2021).
52. Morello, R.; Capua, C.D.; Fulco, G.; Mukhopadhyay, S.C. A Smart Power Meter to Monitor Energy Flow in Smart Grids: The Role of Advanced Sensing and IoT in the Electric Grid of the Future. *IEEE Sens. J.* **2017**, *17*, 7828–7837.
53. Shelby, Z.; Bormann, C. *6LoWPAN: The Wireless Embedded Internet*; John Wiley & Sons: Hoboken, NJ, USA, 2011; Volume 43.
54. Schneider Automation. *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*; Technical Report; Modbus Organization, Inc.: Hopkinton, MA, USA, 2006.
55. Ethernet POWERLINK Standardisation Group. *Ethernet POWERLINK Communication Profile Specification DS301*; EPSG: Fredersdorf, Germany, 2016.
56. International Electrotechnical Commission. *IEC PAS62559: IntelliGrid Methodology for Developing Requirements for Energy Systems*; Technical Report; International Electrotechnical Commission (IEC): Geneva, Switzerland, 2008.
57. Gottschalk, M.; Göring, A.; Uslar, M. Applying the Use Case Methodology to Smart Cities. In Proceedings of the VDE-Kongress, Frankfurt am Main, Germany, 20–21 October 2014.
58. Smart Grid Coordination Group. *Sustainable Processes*; Technical Report; CEN/Cenelec/ETSI Smart Grid Coordination Group: Brussels, Belgium, 2012.
59. Strasser, T.; Lauss, G.; Andren, F.; Stifter, M.; Bründlinger, R.; Fechner, H.; Knöbl, K. Smart Grid Research Infrastructures in Austria—Examples of available laboratories and their possibilities. In Proceedings of the 13th IEEE International Conference on Industrial Informatics (INDIN 2015), Cambridge, UK, 22–24 July 2015; pp. 1539–1545.
60. Zoitl, A.; Strasser, T.; Valentini, A. Open source initiatives as basis for the establishment of new technologies in industrial automation: 4DIAC a case study. In Proceedings of the 2010 IEEE International Symposium on Industrial Electronics (ISIE), Bari, Italy, 4–7 July 2010.
61. Zanabria, C.; Tayyebi, A.; Pröbstl Andrén, F.; Kathan, J.; Strasser, T. Engineering support for handling controller conflicts in energy storage systems applications. *Energies* **2017**, *10*, 1595.