A REFERENCE ARCHITECTURE FOR OPEN, MAINTAINABLE AND SECURE SOFTWARE FOR THE OPERATION OF ENERGY NETWORKS

Andre GOERING
OFFIS – Germany
andre.goering@offis.de

Juergen MEISTER
OFFIS – Germany
juergen.meister@offis.de

Sebastian LEHNHOFF
OFFIS - Germany
Sebastian.lehnhoff@offis.de

Peter HERDT
Main-Donau Netzgesellschaft – Germany
peter.herdt@main-donau-netz.der.de

Martin JUNG
develop-group – Germany
martin.jung@develop-group.de

Matthias ROHR
BTC - Germany
Matthias.rohr@btc-ag.com

ABSTRACT

Regulatory effects, business pressure, and the transformation to smart grids foster the need for up-to-date software systems for managing and operating the grid operators' electric power grids. The complexity of these systems has grown over decades. This makes enhancements and development of new functionalities in existing systems cost intensive, vendor/system specific and often prevents meeting time to market and quality requirements. Public interfaces and open data formats allow the development of enhancements and new functionality as re-usable modules by 3rd parties, thus enable the integration of best-of-breed systems in the system landscape at grid operators. A reduction of system complexity is a precondition to develop such re-usable modules while meeting time to market and quality requirements in critical infrastructure. This is accomplished by defining a reference architecture with a common architecture framework, common processes and quality standards.

INTRODUCTION

The steadily growing integration of decentral renewable energy resources, regulatory effects, business pressure in the unbundled energy sector, and the transformation to smart grids lead to many new requirements for software systems of grid operators for managing and operating their electric power grids. The complexity of existing systems has grown over decades: Each IT-System (e.g. Distribution Management System (DMS)/Supervisory Control and Data Acquisition (SCADA), Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Geographic Information System (GIS)) only holds parts of relevant grid data. Via direct coupling between these systems, data are made accessible specifically for each business process and proprietary at each grid operator/vendor [1]. A net of point-to-point connections leads to dependencies between the systems that are unmanageable. Each of the named systems has a five to 15 years interval of major updates. The upgrade projects are highly complex and cost intensive because of a steadily growing range of (sometimes customer-specific) adaptions and extensions.

This results in a vendor lock-in of grid operators to their system vendors and requires enormous effort for integration, thus binding development capacities needed for new development or updates forced by regulation authorities.

These problems are addressed by a consortium called openKONSEQUENZ[1] (oK), whose goal it is, to significantly reduce maintenance costs of their systems' landscape by decreasing system complexity and vendor dependency as well as increasing software quality, usability and enhancing safety and security in critical infrastructures. According to the Smart Grid Architecture Model (SGAM) [2] the consortium mainly focuses on the development of new functionality in the higher hierarchical zones of operation and enterprise in the domain of power distribution as shown in Figure 1.

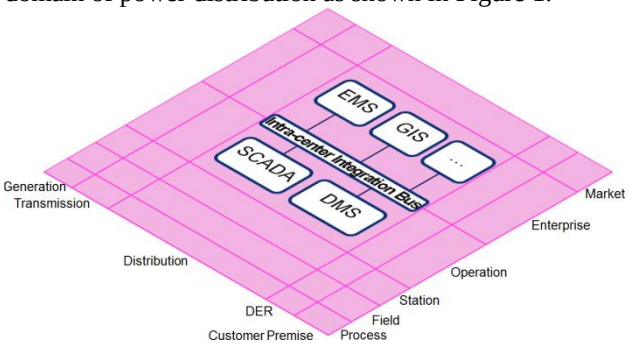

Figure 1: SGAM classification of oK (see also [3]).

The oK consortium brings together German and Dutch Distribution System Operators (DSOs) supplying over 15 million German households with electrical power and 5,7 million Dutch customers with gas and power, software vendors, service providers and researchers. It started up 2013 with the idea of developing open source software to solve the problems explained above. The consortium is organized in the Eclipse Foundation structure as Driver Members (a number of German DSOs), User Members (DSOs with focus on development), Service Providers (including software vendors), and Guest Members (universities and research institutes, interested service providers, a Dutch DSO).

This paper discusses the reference architecture subsuming

1 http://www.openkonsequenz.de

quality and architecture standards, the development process, and a project adoption process developed by oK. Architecture and quality standards have already been published in [4] but are also included in this paper for the sake of conceptual clarity. Chapter 2 describes the related work. Chapter 3 and 4 discuss the reference architecture and its evaluation and evolution while chapter 5 sums up the work and gives an outlook on future projects.

RELATED WORK

In the domain of the smart grid, the SGAM is a reference architecture for describing main actors and interactions with a presentation schema using the NIST Conceptual Model and the GridWise Architecture Council (GWAC) Stack as basis [2]. The GWAC Stack shows, that interoperability has to be discussed on different levels (e.g. Network, Syntactic, Semantic, Procedures or Objectives). So, ensuring interoperability and making software development vendor independent and faster, while keeping software quality, leads to questions for standards on data exchange and architectures for combination of modules of different vendors. These fields are discussed briefly in the context of the electricity domain in the following subchapters.

CIM Standards – IEC 61970, 61968 & 62325

Interoperability issues in the field of oK are mainly addressed by the Common Information Model (CIM) [3]. The Electric Power Research Institute (EPRI) started developing the CIM in the 1990s to solve vendor lock-in at Energy Management Systems (EMSs) [5]. Since then, it was developed further by the International Electrotechnical Commission (IEC) as a series of standards for information exchange: IEC 61970 for EMS, IEC 61968 for Distribution Management, and IEC 62325 for Energy Markets. Core of the CIM is a sematic data model for information exchange in and between electric utilities. The CIM data model describes all necessary structures/elements of electricity networks, their relationships and multiplicities, their semantical meaning, and their syntactical values from the point of view of the IEC. Although it is still under development to cover the whole (and future) extent of data in electrical information modeling, usage of CIM as a foundation for information interchange is superior to proprietary approaches. The European Network of Transmission System Operators for Electricity (ENTSO-E) uses the CIM as basis for their Common Grid Model Exchange Standard (CGMES) to exchange grid models between different Transmission System Operators (TSOs) and achieving interoperability between TSOs software systems. We expect that future model exchanges between TSOs and DSOs for e.g. power grid stability calculations underlie the same standard.

Reference Architectures

Reference Architectures are proven, generic Software-Architectures for specific application domains and apply across product and organizational borders.

Enterprise Application Integration (EAI) [6] deals with business-critical applications. EAI defines patterns on how to integrate such applications as well as newly developed advanced components in order to create complex systems. In such systems, point-to-point connections are not appropriate to facilitate interoperability and maintainability. A central message broker like an Enterprise Service Bus (ESB) is a more suitable solution. It moves messages from any type of application to any other, changing the format of messages to meet target systems' requirements.

Service Oriented Architecture (SOA) is derived from EAI. SOA has several layers: Operational systems, enterprise components, services, business process choreography, and presentation [7]. SOAs main pattern is the combination and recombination of independent services, which base on existing enterprise components, to a business process choreography. The SOA layers match to the energy domain in the following way: Existing IT-systems and sensor-systems of network operators are operational systems. Enterprise components build on that first Layer, to enable e.g. DMS, SCADA, ERP, and CRM. In a SOA, on top of that layer, services provide single or combined functionality, which can be reorganized in business processes in the next layer and get presented in a top layer. The integration functionality and cross-cutting concerns like security, service management and monitoring are orthogonal to these layers.

The Open Smart Grid Platform[2] (OSGP) is a platform for open, generic, scalable and independent "Internet of Things" (IoT) services. It provides information on wide-spread sensors like a SCADA kernel does to a DMS/SCADA and can be used as basis to get data from the field. According to the SOA concept, OSGP can be located in the enterprise layer.

Quality Standards in Agile Development

Software Quality [8, chapter 10] is essential in the development of long-living and safety/security critical software systems e.g. for critical infrastructure. In particular, maintainability, confidentiality, integrity and availability are important for oK software. Quality standards have been put into place to standardize the meaning of software quality with respect to said quality attributes (e.g. by BSI [9], BDEW [10]). These quality standards mainly focus on the operation of software and not on the development.

Agile methods have become a mainstream in software engineering [8, section 4.4], and are also applied successfully in safety critical environments such as critical infrastructure. As one of the specific methods, Scrum has proven to be capable to deliver fast results with high quality. The low project management overhead of Scrum and the self-organizing team culture of all agile

2 https://smartsocietyservices.com/osgp/

methods make Scrum the method of choice for open source development.
Intensive research has been done in comparison of closed source and open source. There is no verifiable general advantage of one to the other in terms of safety and security [11], but open source has other positive aspects, e.g. white box components, adaptability, extendibility, world-wide peer reviews, easy re-use without legal issues.

OK REFERENCE ARCHITECTURE

The oK consortium drives modularization of DMS functionalities and adopts the benefits of the SOA approach. Applications and components are connected using an ESB and the interfaces of the services use an unified data model called oK-CIM-Profiles which base on the CIM standard. A major goal of oK is to provide an ecosystem in which development of independent modules and their integration in critical infrastructure are standardized. We assume, the enforcement of a reference architecture for software in the electricity domain is essential to enable interoperability of such modules. The reference architecture, as well as basic open source platform services, are developed to ease single-vendor-independent adoption of new modules in an ecosystem and allow long-term support.

Multilayer & Technical Architecture

Existing systems, (possible) externally developed modules, oK User Modules and oK Platform Modules (Core Modules and Domain Modules) interact on the basis of standardized interfaces (the previously mentioned oK-CIM-Profiles) and run on an underlying system following the reference architecture concept.

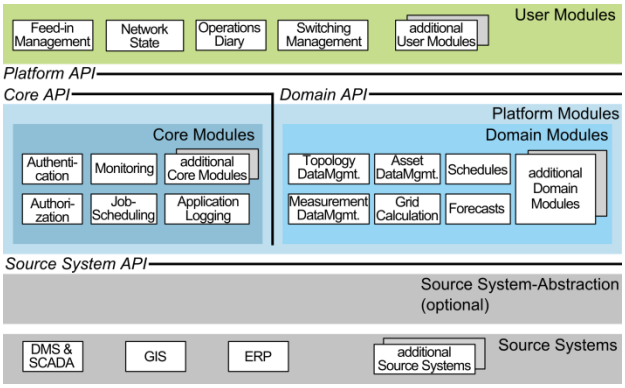

Figure 2: oK Multilayer Architecture.

Figure 2 shows the oK Multilayer Architecture, which provides a general structure to ensure reusability, integratability, modularization, and extendibility. Each module, component, system, and adapter has to be located at some point in this architecture (e.g., shared backend services in the platform layer and GIS, DMS, and ERP in the source system layer). Platform Modules provide reusable basic functionality to multiple User Modules and organize tasks such as source system data access. Platform Modules are distinguished into Domain Modules and Core Modules: Core Modules provide services for cross cutting concerns in a standardized way, while Domain Modules provide specific services to the domain of operating power grid systems. User Modules implement the use cases of end users. The modules communicate using the APIs shown in Figure 2.

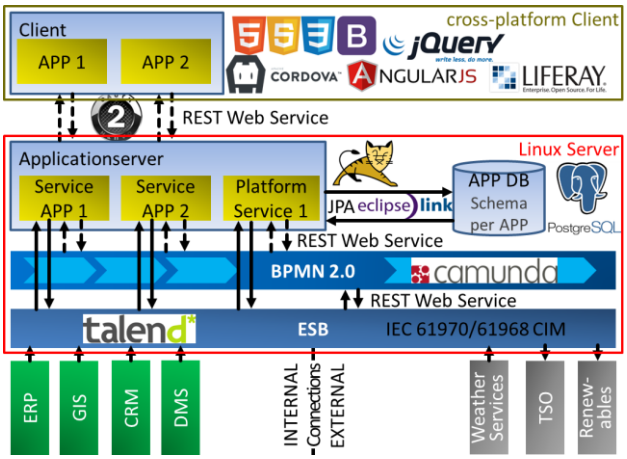

Figure 3: Technical Architecture.

A technical architectural view is shown in Figure 3. This shows that oK makes extensive use of open source technology to implement the Modules. A typical oK application (i.e., a User Module and required Platform Modules) is implemented in Java, has a Web-Interface, and stores own data in a PostgreSQL database. Following the microservice paradigm, a concrete implementation may use other technologies, such as for instance other database management systems.

Quality Standards

The oK platform consists of open source software modules, developed by independent parties using an agile methodology. To engineer and safeguard quality requirements – foremost the security goals and maintainability – of all modules as well as the integrated platform, rigorous quality standards for the software development are mandatory.
oK defines its quality standards in three categories: code quality, design quality, and product quality.
Code quality is maintained by defining

- a set of coding guidelines,
- file naming conventions,
- configuration management conventions,
- build, package and test mechanisms,
- and run-time monitoring and diagnosis functions common for all modules.

The coding guidelines and the common conventions ensure conformity of the modules developed by independent organizations. Central elements of quality assurance on code level are static analysis, automated testing, and dynamic analysis. These mechanisms are

implemented by reviews (see below) and by nightly builds on a continuous integration system which includes static analysis (enforcing the coding guidelines), unit-tests, and code coverage analysis.

Design quality is maintained by defining a set of design documents common for all modules. The central design document of each module is its architecture concept. The outline and contents of these documents are defined. A test specification document is also required for each module, defining integration test cases that are also run during the nightly builds. Design quality is maintained by a peer review setup. Design documentation – as well as the code itself – of each module is examined by a third party, typically the architects and developers of another module.

Product quality is maintained by using a reference installation environment ("QA environment"). After each sprint, the module is deployed into an oK platform installation in the QA environment. Each module has to produce a test specification and a validation concept to describe the test steps to be performed on the QA environment. As far as possible, these test steps should be automated, but manual tests will be required on product level. The manual tests are executed by the product owners, at least once at sprint end, before they accept a feature/user story. The product documents are also subject to peer review. The intensity of reviews to ensure quality is determined by the classification of a module in terms of criticality and complexity. The documents will be created and filled according to the agile development method. Availability of the document contents relevant for a feature/user story is part of the "done" criteria. This helps keeping the documents up to date, and also helps to keep the review scope in each sprint small.

The quality rules are independent of technology as far as possible and may be adjusted to technologies applied in oK now or in the future.

UI Style Guide

In critical situations, users need to identify and select their options fast. To get an adequate reaction time, users need to be comfortable with the user interface (UI) even though different modules may have different ones. To address this and further end-user requirements, the EN ISO 9241-110 was chosen as a foundation of oK's UI style guide. The style guide is mandatory, thus ensuring the same user experience and comfortability for different modules and also reducing the need for training on each UI.

Development Process

To fulfil oK's vision and create a reference platform for high-quality grid operator software applications, a vibrant ecosystem of developers and users is necessary, and oK has to exercise some sort of control over the open development. Control is necessary in two ways, to steer the development of the platform features and to control the quality of the platform.

oK issues requests for tenders in order to have vendors develop features (user and platform modules as shown in Figure 2) of the platform. By prioritizing such tenders, the development of the platform can be guided. Also, oK requires vendors to state the dependencies to the reference platform modules. This enables oK to enforce dependency management early on, and to ensure availability of the platform modules.

The oK community relies on three groups of participants:

- *users (and their integrators)* who deploy the oK platform, integrate it in their IT systems, and operate it. Typically, they contribute to the community by reporting issues with the platform and by requesting additional features. An example for users are the grid operators. The tenders mentioned above are driven by the users.
- *contributors* who fix issues or work on features. Typically, they contribute substantial amounts of code, over a limited time. An example for contributors are researchers or software service providers.
- *committers* who exercise QA on the work of the contributor. Additionally, they perform architecture work and maintenance. Typically, they contribute by prioritizing the work on features and issues, over extended periods of time. An example for committers are the oK driver members, operators and vendors alike.

Finally, to centralize all work on the oK platform, a host for the community is required. There are several such hosts, e.g. the apache foundation, github, or the eclipse foundation. They vary in the amount of services they provide, as well as the different open source licenses they support. oK's primary criteria for the selection of a host environment was the availability of an intellectual property (IP) check service and the support of a license that allows the commercial usage of the platform.

oK chose the eclipse foundation as a host. They offer an elaborate IP management process, and its license, the Eclipse Public License (EPL) allows a variety of business models. Also, it offers a development environment including source code repositories and continuous integration services, and an easy integration to many other open source projects, e.g. IoT.

Project Adoption Process

The development as open source enables everybody to contribute new functionality. In existing modules, according to the previous subchapter, the committers are responsible for checking contributions. For contribution of whole modules in the landscape of oK, there are different levels of conformity:

a. *oK-ready* – a vendor labels his software to be compatible to oK-interfaces without further checks,
b. *oK-ready** – the vendors software is tested for API compatiblity by oK,
c. *oK-UI-Styleguide-Conformity*,

d. *oK-Architecture-Conformity*,
e. *oK-Quality-Conformity*,
f. *full-oK* – all above mentioned, approved by oK, published as open source in an oK eclipse project.

Closed source commercial modules of a vendor may also be classified as levels b to e by certification authorities that are approved by oK. For level f, the source code needs to be open. When adopting software as *full-oK*, a new form of lock-in could arise, and has to be avoided. This may be by pre-dominance of technical solutions in the case of large components donated by industry organizations, but also by unique know-how in contributions from the scientific community. The quality standards mentioned above serve to mitigate this risk. Also, there are cycles of checks (by oK) and modifications (by module contributors or oK) to discover and correct violations of the reference architecture.

EVALUATION

The evolution of the reference architecture is part of the overall development process, described in the previous chapter. So oK drives the reference architecture development parallel to module development. After an initial draft architecture, created in a pilot project for feed-in-management[3], the architectural development is performed in oK's architectural and quality committees (AC/QC) with respect to future modules. It will be adapted and supplemented to future demands of module development. For example in the current awarding process of the module *Operation Diary*, details on authentication and authorization will be examined by the module developers and discussed by the AC/QC not only for the Operation Diary but for all future modules. The novel architecture concepts are evaluated in sprint reviews, the project reviews and regular discussion, as well as in a research project on integration of big data technologies in the operation of energy networks called *NetzDatenStrom*, and the integration projects, where each newly developed oK module is adopted to individual DSOs system landscapes. Initial evaluation in the pilot project shows a barrier in using CIM by DSOs, mainly due to unfamiliarity with this set of standards. As discussed above, application of the CIM standards is essential.

SUMMARY

The oK consortium drives reference architecture development in their field of electricity network management and operation to overcome the existing vendor lock-in, monolithic systems, and system complexity that hinders development of new, needed functionalities for smart grids. Therefore, the consortium uses CIM standards to ensure interoperability and applies open source development methodology to support both an open multi-vendor ecosystem and quality assurance. The oK specifies multi-layer and technical architecture, the quality assurance, as well as guidelines for UIs and processes for development and adoption of projects. Developed under the EPL, the modules keep being modifiable and usable in DSOs system landscapes. Results of further development of oK modules by vendors are shared with the community. A foundation of Platform Modules is build up in single module projects, while quality assurance is in place over all projects. In the long-term, modules implementing new functionalities for the users can rely on a growing number of Platform Modules, maintained as open source software. Novel functionality can be developed as open source by the consortium as well as closed source by commercial vendors with the possibility of easy integration in the landscape of electricity grid operators reaching more software efficiency and reliability.

3 A German demonstrator can be found on http://demo.openkonsequenz.de

REFERENCES

[1] ARGE KONSEQUENZ, 2013, *Machbarkeitsstudie – Konsortiale Softwareentwicklung auf der Basis von Open-Source-Software*, ARGE KONSEQUENZ
[2] CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012, *Smart Grid Reference Architecture*, CEN CENELEC.
[3] IEC, 2014, *Smart Grid Standards Map*, IEC, Online: http:/smartgridstandardsmap.com/
[4] A. Goering, J. Meister, S. Lehnhoff, M. Jung, M. Rohr, P. Herdt, 2016, "Architecture and Quality Standards for the Joint Development of Modular Open Source Software for Power Grid Distribution Management Systems" , *Proceedings 5th D-A-CH+ Energy Informatics Conference*, OVE, vol. 84, 36-39.
[5] M. Uslar, M. Specht, S. Rohjans, J. Trefke, J.M. González, 2012, *The Common Information Model CIM – IEC 61968/61970 and 62325 – A Practical Introduction to the CIM*. Springer, Berlin, Germany.
[6] D.S. Linthicum, 2000, *Enterprise Application Integration*, Addison-Wesley, Boston, USA.
[7] A. Arsanjani, 2005, *Service-oriented modeling and architecture – How to identify, specify, and realize services for your SOA*, IBM.
[8] P. Bourque, R.E. Fairley, 2014, *Guide to the Software Engineering Body of Knowledge SWEBOK*, Version 3.0, IEEE.
[9] N. Lefin, 2010, *Weiterentwicklung des IT-Grundschutzes – IT-Grundschutz-Profil für Open-Source-Software (GSProOss)*, master thesis, Hochschule Landshut.
[10] BDEW, 2015, *Requirements for Secure Control and Telecommunication Systems*, Version 1.1, BDEW.
[11] G. Schryen, 2011, "Is Open Source Security a Myth?", Communications of the ACM, vol. 54 No. 5, 130-139.

All web links are accessed on 2017-01-13.