Architecture and Quality Standards for the Joint Development of Modular Open Source Software for Power Grid Distribution Management Systems

Andre Goering · Juergen Meister · Sebastian Lehnhoff · Martin Jung · Matthias Rohr · Peter Herdt

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 development of enhancements and new functionality as re-usable modules by 3rd parties, thus allowing the integration of best-of-breed systems in the system landscape at grid operators. A significant 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 common architecture framework, common processes and quality standards.

1. Motivation

The steadily growing integration of decentral renewable energy resources, regulatory effects, business pressure in the unbundled energy sector and the transformation to smart grids foster the need for up-to-date 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 respective upgrade projects are highly complex and cost intensive because of a steadily growing range of functional adaption extensions, or new development of the interfaces.

This situation results in a vendor lock-in of grid operators to their system vendors and requires enormous effort by the vendors 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[3] (oK), which target is, to reduce maintenance costs of their systems landscape by reducing system complexity and vendor dependency as well as increasing software quality and software development efficiency.

The oK consortium brings together German and Netherlands Distribution System Operators (DSOs) supplying over 15 million German inhabitants 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 vendor lock-in 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).

[3] http://www.openkonsequenz.de

Andre Goering · Juergen Meister · Sebastian Lehnhoff
OFFIS, Escherweg 2, DE-26121 Oldenburg
Andre.Goering@offis.de

Martin Jung
develop-group, Am Weichselgarten 4, DE-91058 Erlangen
Martin.Jung@develop-group.de

Matthias Rohr
BTC, Escherweg 5, DE-26121 Oldenburg
Matthias.Rohr@btc-ag.com

Peter Herdt
Main-Donau Netzgesellschaft, Hainstraße 34, DE-90461 Nuernberg
Peter.Herdt@main-donau-netz.de

This paper discusses the architecture and quality standards development by the oK. Chapter 2 describes the related work. Chapter 3 shows the current results and chapter 4 sums the work up and gives an outlook on future projects.

2. Related Work

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.

2.1 CIM Standards – IEC 61970, IEC 61968, IEC 62325

In the energy domain, the Electric Power Research Institute (EPRI) started developing a Common Information Model (CIM) in the 1990s to solve vendor lock-in at Energy Management Systems (EMSs) [2]. Since now, it is developed further by the International Electrotechnical Commission (IEC) as series of Standards: 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 data exchanges 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 but is still growing to meet future requirements, and therefore should be stressed for data exchange instead of proprietary developments. 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. It might be reasonably assumed, that needed future model exchanges between TSOs and DSOs for power grid stability calculations may underlie the same standard.

2.2 Reference Architectures

Reference Architectures are proven, generical Software-Architectures for concrete domains and apply across product and organizational borders.

Enterprise Application Integration (EAI) [3] deals with business-critical systems that are hard to adapt to communicate and share information with more advanced systems, which are long known issues. Point-to-Point connections are not appropriate to facilitate interoperability. A central message broker like an Enterprise Service Bus (ESB) is an adequate solution. It moves messages from any type of application to any other, changing the format of messages according to target systems.

Service Oriented Architecture (SOA) [4] is derived from EAI. It is the combination and recombination of services, which base on existing enterprise components, to a business process choreography. SOA has several layers that match to the energy domain in the following way: Operational Systems are existing IT-Systems and Sensor-Systems of network operators. Enterprise Components build on that first Layer, to enable e.g. DMS, SCADA, ERP, 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 and Security/Management/Monitoring lie vertical to these tiers as cross-cutting issues.

The Open Smart Grid Platform (OSGP)[4] is a platform for open, generic, scalable and independent “Internet of Things” (IoT) services. According to the SOA concept, it provides information on wide spreaded sensors like a SCADA kernel does to a DMS/SCADA and can be used as basis to get data from the field.

Using the CIM’s information model (see previous subchapter) on an ESB as core concepts in the interoperability architecture on top of existing enterprise applications, a set of central energy domain services/modules can be built to provide energy systems data/services and be recombined to high-level decision and optimization functions.

2.3 Quality Standards in Agile Development

Software Quality [5, chapter 10] is fundamental in software engineering and essential in development of long-living and safety/security critical software systems, e.g. critical infrastructure. Such software systems have to guarantee quality attributes, foremost extended maintainability and security goals, commonly termed “CIA Triad” (confidentiality, integrity and availability). Quality standards have been put into place to standardize the meaning of software quality with respect to said quality attributes (e.g. ISMS[5], BSI[6], BDEW[7]). These quality standards mainly focus on the operation of software and not on the development.

Agile methods have become a mainstream in software engineering [5, section 4.4], and are also applied successfully in safety critical environments such as critical

[4] https://smartsocietyservices.com/osgp/

[5] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/BSI-Standard_1001.pdf

[6] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Hilfsmittel/Extern/Diplomarbeiten/Erstellung_IT-Profil_Lefin.pdf

[7] https://www.bdew.de/internet.nsf/id/232E01B4E0C52139C1257A5D00429968/$file/OE-BDEW-Whitepaper_Secure_Systems%20V1.1%202015.pdf

infrastructure. As one of the specific methods, Scrum has proven to deliver fast results with high quality. The low project management overhead of Scrum and the self-organizing team culture of all agile methods make Scrum the method of choice for open source development.

From a safety and security point of view, open source software is considered by the BSI to have significant advantages[8].

3. openKONSEQUENZ Approach

The oK drives modularization of DMS functionalities with a SOA like communication over an ESB with oK-CIM-Profiles. The oK develops open source and agile with the goal to establish a reference architecture in the electricity domain to allow an independent development of modules by vendors and to integrate these efficiently in critical infrastructure.

3.1 oK Multilayer 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 oK APIs) and run on an underlying system following a reference architecture concept.

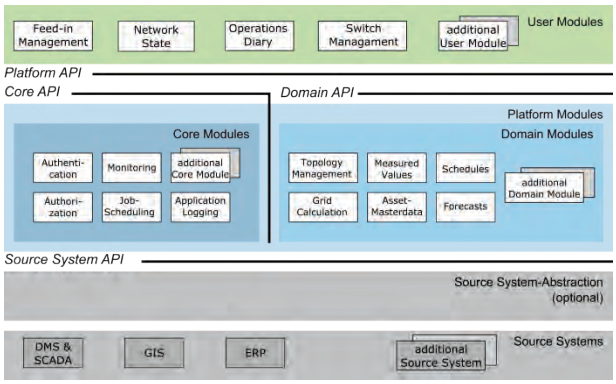

Figure 3: oK Multilayer Architecture.

Figure 1 shows the oK Multilayer Architecture, which provides a general structure to ensure reusability, integratability, modularization and extendibility. Each module (i.e., components, systems and adapters) 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 higher level functions for operating power systems. User Modules implement the use cases of end users. They contain business logic and may have an own private data storage and own user interfaces. The modules communicate using the APIs shown in Figure 1.

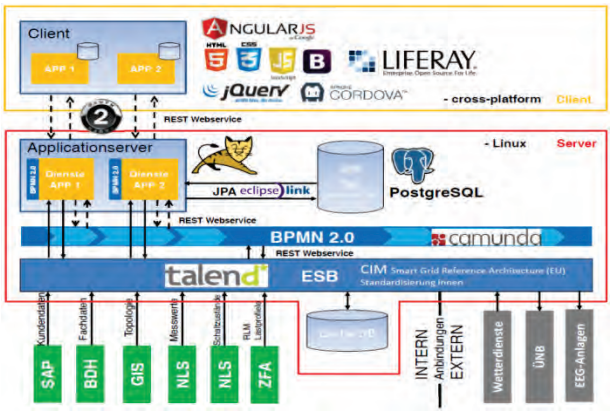

Figure 4: Technical Architecture.

A technical architectural view is shown in Figure 4. 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. A concrete implementation may use other technologies, such as for instance other database management systems.

3.2 oK 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[9]: 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 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

[8] https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/FreieSoftware/freiesoftware_node.html

[9] https://wiki.eclipse.org/images/0/08/OK-QualityCommitteeHandbook-Current.pdf

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[10]. 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 at least once at sprint end, before a feature/user story will be accepted. The product documents are also subject to peer review. Depth and rigor of the review methods used to ensure quality are 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 rules defined in the quality handbook are independent of technology as far as possible and may be comfortably adjusted to technologies applied in oK now or in the future. At the time this paper is written, the technologies shown in Figure 4 are included.

4. Summary

The oK consortium drives architecture and quality standards in their field of electricity network management and operation to overcome the existing vendor lock-in and system complexity that hinders development of new, needed functionalities for smart grids. Therefore, the consortium uses CIM standards intensively for ensuring interoperability and quality assurance approaches from the open source development. The work does not fall in the category of IoT. Wide spreaded sensors and actors are not directly integrated, as a SCADA kernel is also not directly integrated, but interconnections can be established via the current SCADA.

The oK already developed a pilot for feed-in-management. This pilot is divided in two different modules: (i) a platform module for the work with and caching of topologies of the energy grid – an existing DMS/SCADA/GIS must not be queried permanently for this data – yielding higher stability. (ii) a user module for the feed-in-management, using the mentioned platform module for topology management.

With a number of core modules of oK maintained as open source software, in the long-term modules implementing new functionalities for the users can be developed as open source by the consortium as well as closed source by interested vendors with the possibility of easy integration in the landscape of electricity grid operators.

References

1. ARGE KONSEQUENZ (2013) Machbarkeitsstudie – Konsortiale Softwareentwicklung auf der Basis von Open-Source-Software. https://wiki.eclipse.org/images/3/3f/2013_Okt_KSE_Studie_gesamt_final.pdf
2. Uslar M, Specht M, Rohjans S, Trefke J, González J M (2012) The Common Information Model CIM – IEC 61968/61970 and 62325 – A Practical Introduction to the CIM. Springer Berlin Heidelberg
3. Linthicum D S (2000) Enterprise Application Integration. Addison-Wesley, Boston
4. Arsanjani A (2005) Service-oriented modeling and architecture – How to identify, specify, and realize services for your SOA, Technical Report, IBM
5. IEEE (2014) Guide to the Software Engineering Body of Knowledge SWEBOK, Version 3.0

All web links in references and foot notes are accessed on 2016-09-09.

Andre Goering studied computer science at Technical University Dortmund in Germany and finished his diploma in 2012. Since then, he is working for OFFIS - Institute for Information Technology in Oldenburg, Lower Saxony, Germany in the R&D Division Energy in terms of architecture development and interoperability.

[10] https://wiki.eclipse.org/images/3/3d/OK-ArchitectureCommitteeHandbook-Current.pdf