Advanced Metering and Data Access Infrastructures in Smart Grid Environments

Armin Veichtlbauer, Dominik Engel
Josef Ressel Center for User-Centric
Smart Grid Privacy, Security and Control
Salzburg University of Applied Sciences
Puch/Salzburg, Austria
{firstname.lastname}@en-trust.at

Fabian Knirsch, Oliver Langthaler,
Felix Moser
Research & Development, Energy Systems
Cappatec
Salzburg, Austria
{firstname.lastname}@cappatec.com

***Abstract*—For securing the provision with electric energy in smart grids, the ascertainment of energy consumption and production is inevitable in order to be able to apply appropriate control algorithms. As energy management is a highly distributed task, a data network has to exist in parallel to the power network for gathering the required data and applying control strategies by using distributed actuators, e.g., for demand response applications. For the sensing part, the introduction of Advanced Metering Infrastructures is a step ahead to the targeted meter data acquisition. Yet the definition and deployment of intelligent, but nevertheless privacy preserving, distributed control algorithms demands the collection of data from distinct sources and the application of further services like aggregation, anonymization, etc. Thus, future metering infrastructures have to provide more functionalities than those of a simple sensor network. In this paper, we discuss methods to provide such an added value infrastructure.**

***Index Terms*—Advanced Metering Infrastructures; Smart Grids; Role-based Data Access; User Control; Generic Interfaces**

I. Introduction

In order to address the challenges of a modern Smart Grid IT communication infrastructure, it is imperative to incorporate various decentralized data sources representing all kinds of different stakeholders in a Smart Grid communication environment [1]. Data items of varying precision and granularity need to be stored, processed, enriched, transported, transformed and transmitted to serve the requirements of all stakeholders, i.e., customers, energy providers and regulatory authorities.

Data from different sources in a Smart Grid environment may be retrieved in many different representations and formats. For Advanced Metering Infrastructures (AMI), there are a number of (partly proprietary) protocols available, i.e., the need for means to make these data sources interoperable is evident [2]. Furthermore, different data items may have logical relations which require further processing.

An AMI acting isolated from other services is not enough to ensure customer acceptance as well as commercial interests [3]. An all-encompassing approach for generating added value requires high level services which are able to operate across system, network and format boundaries.

While sensor networks typically operate in a limited domain only, significant enhancement of data can be achieved when being integrated in an overall network of information. However, such information does have a specific semantic relationship, e.g., a customer is assigned a meter and has a specific contract, which is not physically represented at all. There are a multitude of data sources that need to be combined to retrieve data as an additional service.

Establishing an architecture which is able to provide this functionality involves all layers of communication and information exchange. Starting from physical data exchange up to the application layer, an entire series of challenges need to be mastered.

For addressing the different layers and zones of a Smart Grid, the Smart Grid Architecture Model (SGAM) [4] is a standardized model [5] authorized by a European Commission mandate M/490 [6].

The main requirements for such an architecture are:

- Retrieving data from multiple and decentralized sources with heterogeneous interfaces and data structures
- Storing and caching data in appropriate resolutions to ensure the delivery of appropriate services
- Adding additional services for enhancement of data and information retrieval as one of the key features to ensure a sustainable metering infrastructure
- Providing data in a standardized format to allow open accessibility for client applications
- Assuring security and privacy of information and personal data by a role-based access model

In this paper, we propose a middleware architecture that allows for the retrieval of data from multiple sources in a standardized format and provides a scalable environment for processes and activities that use, enhance and maintain access to said data and the corresponding sources. We start with an overview of related work in Section II, followed by a description of our proposed architecture in Section III and an outline of internal data structures in Section IV. Section V presents the interfaces for concrete data providers as well as a generic interface for data users. Section VI contains information about the conducted implementation and evaluation activities.

II. Related Work

In [7], we presented a Smart Metering and data access infrastructure for a Smart Home environment. Data is retrieved, processed and provided to using applications. This architecture provides a good basis for our present research activities; yet it had two shortcomings which we had to overcome: First, the access of data sources as well as the scope of the applications were centralized; second it did not provide means for value enhancing services like data aggregation.

For Advanced Metering Infrastructures, there are several approaches dealing with different data sources [8]. As security and privacy issues do have a significant impact on design, implementation and subsequently on customer acceptance [9], such security and privacy issues have been identified and described especially concerning the intended two-way functionality of modern power grids [10].

[1] and [11] each describe a comprehensive approach dealing with communication aspects, incorporating mainly the component and communication layers of the M/490 conceptual model. In both cases, IP-based network infrastructures as well as the ensuing latency issues are discussed. Furthermore, [1] proposes an information middleware to deal issues like security and data storage. Both approaches, however, are omitting any higher level middleware functionality.

Another approach recently presented [12] introduces a Service Oriented Architecture (SOA) for a Smart Grids middleware, provided by individual web services with a centralized policy store/policy decision point. Thus, it becomes in principle possible to provide any service with any data source, yet this work focuses mainly on security and privacy issues.

Handling different types of data is one part of the challenge, finding a generic approach to storing and processing the data is a different issue altogether. An approach for adding an ontology layer for semantically sensitive retrieval is proposed by [13].

III. Architecture

Considering the requirements listed in Section I, it becomes apparent that sensor networks can only be a part of the solution, even if they are combined with classic middleware. To cover the entire set of requirements, a SOA is well suited. The advantage of a SOA over plain middleware is the possibility to define processes as a whole and not just interfaces. Such a layer provides sets of services, incorporating multiple concrete data sources. Services include simple requests expressed in a common language as well as more complex queries, where the intermediary layer provides additional value for any given data. Thus, also a hierarchical structure of abstract services according to the SGAM is made possible.

To achieve centralized data access, we designed an integrated business logic where services are tailored to data sources at a certain organizational or logical unit. This design is fully aligned with the conceptual idea of a SOA. As in [7] we defined an architecture that is designed around three layers: The infrastructure layer includes the data sources (sensors, but also database interfaces), a middleware layer (the “Core Engine”) provides services, and an application layer comprises client applications and service users.

All Core Engines are independent systems, each operating directly with their associated data sources and client applications. However, these Core Engines can communicate and interoperate with each other, as shown in Figure 1. This allows System A client applications to access - in accordance with security policies - services from System B or other connected systems. Supplemented by Single Sign On (SSO) solutions, client applications can gain transparent access to numerous systems without being directly concerned with data origin issues. Client applications themselves may act as service providers and can thus combine data from various sources.

This concept provides scalability at multiple levels: Individual cores can maintain connections to an arbitrary number of data sources. It is even possible to attach a core instance to a single meter, e.g., a Smart Meter installed in a private house. Furthermore, security policies can be defined for individual cores as well as for individual services. Policies are based on a set of rules that are applied for a certain user or a certain group. For a single core with locally assigned security rules, the instance acts both as Policy Decision Point (PDP) and Policy Enforcement Point (PEP). For a grid of connected cores, a hierarchy of trust is implemented. Assume System A is a trust providing party (“Primary

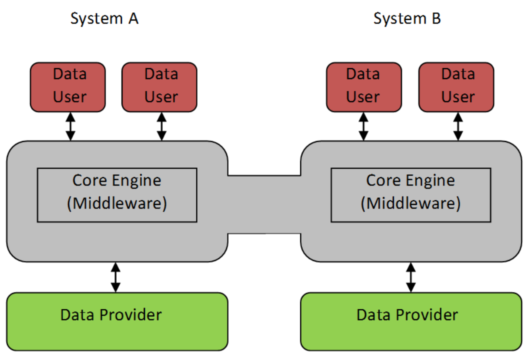

Figure 1. Platform Independent Architecture

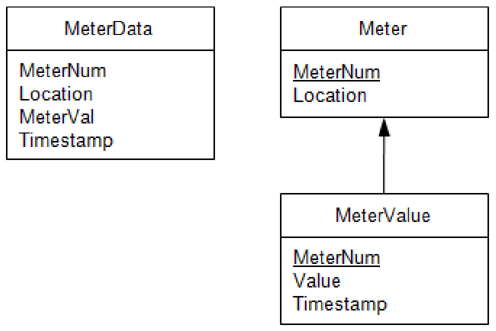

Figure 2. Example of Unnormalized and Normalized Tables

Core") for System B and System C ("Secondary Cores"). For any requests to a Secondary Core, System A acts as the PDP and the Secondary Core as the PEP.

IV. Data Structures

As data will be treated in a set oriented way, concepts of relational algebra are applied to the definition of data structures. To facilitate data processing with object oriented programming languages, certain aspects of object oriented models (such as inheritance) further influence the design of the data model. In practice, this concept is directly mapped to relational database management systems (RDBMS). Modern RDBMS (Oracle, DB2, PostgreSQL) include an extensive support of the relational set oriented model and an object oriented layer [14]. The biggest advantages of relational databases are their popularity in the industry and the standardized access via the Structured Query Language (SQL) [15], which is widely supported and implementation independent.

Another interesting approach is to use XML-based databases. It has to be distinguished between native XML databases where data is stored document centered, i.e., as it comes, and XML-based interfaces where data is stored data centered, but accessed via XML. Native XML databases seem not to be reasonable in an environment where we deal with defined data structures and not with documents. A meter will not send document oriented information, but instead a sequence of values strictly adhering to a predefined pattern. Such an approach might therefore be useful for client interfaces that retrieve data for human users, but not necessarily for automated processing facilities.

Relational databases provide the facility to map several kinds of data structures to the database [14]. Normalization levels range from completely unnormalized data structures to highly structured sets of values. The advantage of the latter is that several pieces of information can be retrieved simply by joining tables. With unnormalized tables on the other hand, it is easy to imitate big tables [14]. These databases do not keep structured table layouts but store all the information in lists assigned to a specific key. This is a powerful way to build huge hash maps for data that is always retrieved in the same manner.

Figure 2 shows an example of two different data structures both in use for our solution. They contain the same information, but for different purposes. The unnormalized table on the left is used for temporary storage of large data amounts that need to be accessed quickly and frequently. The normalized structure on the right is more suitable for analysis and storage.

Concerning internal data storage, currently all concrete implementations rely on, but are not limited to, relational databases. This model has been chosen due to freely available and powerful implementations and the ability to natively integrate such data sources into the Core Engine. All possible retrieval scenarios may be combined freely for various requests, i.e., when client level requests reference to multiple data sources, the corresponding activity might use cached data, live data and locally stored resources to service the request.

V. Interfaces

Generally, there are two types of interfaces to be considered: Data needs to be accessed by data users and by data providers, respectively. Either of these two require facilities to actively request data and facilities to be triggered by the middleware to retrieve data.

A. Data Providers

Although a number of interfaces to data sources will need to be incorporated in a Smart Grid / Smart Metering infrastructure, within the scope of this work we focus on test data provided by our company partner Salzburg AG. This comprises the following data sources:

- SAP IS-U: An SAP system providing customer and technical data. Concrete data sets include customer name, contract information and meters assigned to a single customer. Data is retrieved through SAP function modules. Function arguments are a business partner ID. This model is commonly represented as the *SAP IS-U House* [16].
- AMIS: A Meter Data Management system providing actual meter data [17]. This database archives actual data from Smart Meters representing customer usage, meter states and latency values. Data is either retrieved through a web service or directly via native database interfaces. Web service arguments are a meter number and a period of time, returning customer consumption.
- SMIS: A database providing specific meter data for gas and district heating. Data is accessed directly from a relational database through an XML interface provided by the local system.

All these interface provide data in different formats and with different protocols.

B. Data Users

Designing an application interface for data users particularly involves the following functionality:

- An open standard for mark-up that can easily be implemented by the different kinds of client applications
- A protocol for authentication/authorization and persistent operations
- A data exchange protocol that can be defined upon the used open standard

Furthermore, any connections to the system should converge to a single interface definition in order to keep homogeneity throughout all aspects of client accessibility. To meet our previously defined requirements, XML has been chosen due to the vast number of freely available implementations for any client application [18] and its ability to be easily transformed from one scheme to another [19]. The concrete interface has been created by defining XML-based data structures to handle all kinds of data types (such as numbers, strings, timestamps, binary objects) and allowing scalable representations from scalar values to entire sets containing multiple rows. This interface therefore implements a mapping of the concepts used for data structures to actual data representation for external components.

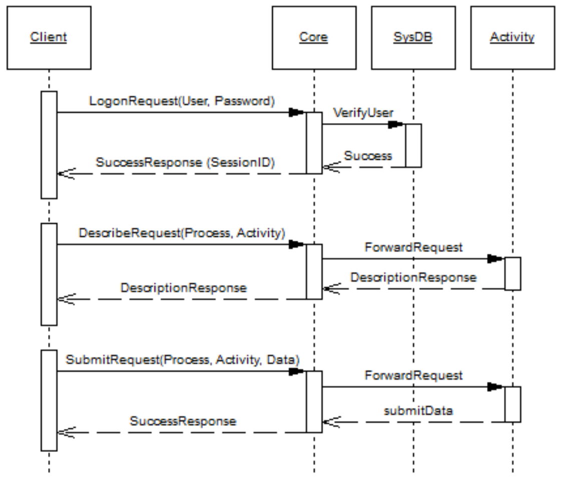

Figure 3. Data Retrieval Process

For our solution we used R×XML [20] as a single interface to connect to external client applications, other instances of the Core Engine and to all data sources. R×XML defines a set of XML schemes that has been designed to provide both an efficient and comprehensive representation of data for transfer or storage and a protocol to exchange basic information about access rules and persistency. In order to support various web-based protocols, such as SOAP or REST, a “SOAP Intermediary” component is applied as a converter to R×XML. This allows multiple cases in which the tasks of the Intermediary can vary from simply transforming data from clients to the Core Engine to providing additional value, such as generating graphical representations for web applications.

This additional interface is not integrated directly into the Core Engine but builds upon the existing R×XML interface. It is designed as a separate and independent service. Clients can communicate with this server by sending web requests via the SOAP protocol. Incoming messages are parsed and their SOAP body is converted to R×XML. Subsequently, the SOAP Intermediary forwards the data to the Core Engine for further processing. The Core Engine in turn regards the SOAP Intermediary as a user with specific authorizations.

In accordance with the aforementioned concept of PDP/PEP, security is governed by the Core Engines solely. A trusted provider (i.e., the owner of the specific Core instance) can label certain processes and activities with access privileges for specific users and roles. When clients log on, they provide a username/password combination to the Core Engine and are assigned a Session ID. Based on this Session ID (a random n bit hex string), further requests are handled. Every Session ID can be mapped directly to a certain user and therefore to a certain role and a set of access rules for that role. Figure 3 shows a sequence diagram illustrating the data retrieval process. In this scenario, a data user (client) logs on, retrieves an interface description and subsequently retrieves data. For securing the transmission of requested data HTTPS [21] is used. As the current implementation is enforcing end-to-end security, no additional encryption is required. However, for specific clients / data sources, data might be custom encrypted and then transferred over other protocols.

VI. Implementation Aspects

For software development, we mapped the architectural outcomes to concrete technologies and design patterns. The general architecture has been developed strictly adhering to the Model-View-Controller (MVC) layout. Data sources, business logic and client representation are conceptually linked via a set of defined interfaces. MVC is a direct mapping of the proposed three-layer architecture [22]. Figure 4 shows the architecture of the implemented prototype.

For process controlling, activity management and role-based user management, the *Cappatec Core Engine 1.x* is used. This Core Engine is enterprise software, developed to run all kinds of applications and services, including business processes and activities, on a common platform. It has been determined that within the scope of the project all the specified requirements and all the use cases developed for an advanced metering and data access infrastructure can be represented as activities on this platform.

The fundamental concept of this approach is the definition of individual modules, which incorporate the business logic by handling certain tasks, such as retrieving and/or aggregating data. These modules may be developed completely independently of each other and then executed on a central Core Engine, which handles runtime behavior and governs security issues.

As an environment for development and testing, a VMware vSphere 5 Update 1 Hypervisor running two instances of CentOS 6.3 as well as one instance of Windows Server 2012 has been set up. This way, easy migration to other, more powerful hardware platforms could be ensured. The first instance of CentOS was used as a host for the Cappatec Core Engine, while the second instance was running PostgreSQL 9.2.3, containing the system and cache databases. Windows Server 2012 was used as host for the SOAP Intermediary.

Development of the Cappatec Core Engine has been performed using Eclipse Juno as IDE and Java 7 as programming language. Furthermore, a lightweight client written in C# and JavaScript, which provides generic access to the Core Engine for debugging purposes, has been developed using Visual Studio 2010. The SOAP Intermediary utilizes Apache Tomcat 4.1.3 and Axis 1.4 and has been written in Java 7.

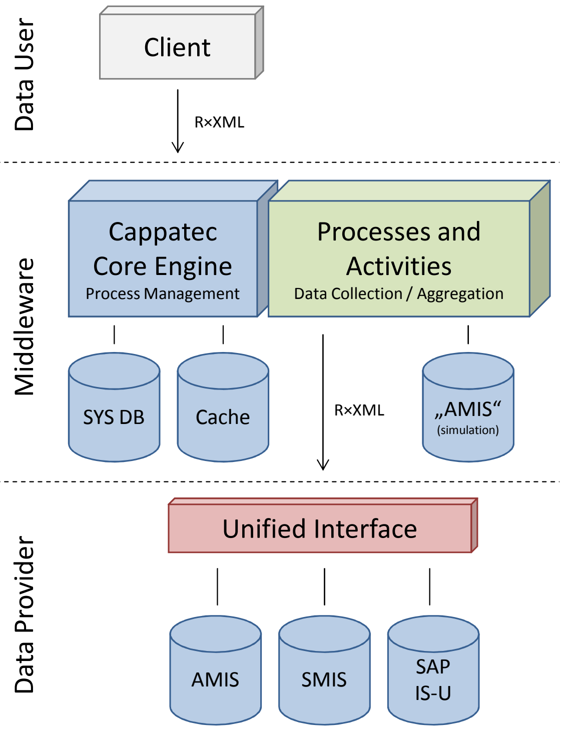

Figure 4. Platform Specific Architecture

VII. Validation and Conclusion

The network setup during development and testing has been kept as simple as possible. All three of the mentioned virtual images have been placed within one local subnet. To achieve R×XML convergence across all aspects of data retrieval (e.g., contract information, meter data), an instance of Inubit 6.1 [23], which connects to all relevant subsystems of the energy provider and acts as an R×XML converter, has been utilized. Conversion is performed by means of XSLT, which is supported natively by this platform.

For evaluating the prototypical implementation, we considered two data sources:

- First, we processed real, but anonymized measurement data from Salzburg AG, connected via Inubit, in order to validate the accessibility of real measurement data.

- Second, we used simulated data in an internal AMIS data base to make it easier to compare multiple scenarios and error conditions, as well as for scalability tests.

Both data sources showed the basic functionality of our solution, i.e., the access to measurement data and the provision to respective applications. Yet the actual strength of the approach is the integration of additional services like data aggregation; here, further validation efforts will be necessary. For instance, our research group has particular interest in the combination of measurement data from the AMIS system and the business data from the SAP IS-U [16]. Also, further tests will be necessary in order to compare our solution with existing ones, e.g., in terms of performance issues.

VIII. Further Work

Besides exhaustive validation, further research might be carried out especially in the domain of security and privacy. The authentication of data user, data providers and distributed Core instances can be performed by certificates. Concerning the architecture itself, the use of a completely generic and semantic data model is considered. Such a data model, however, should integrate seamlessly in the entire architecture and must therefore provide an additional set of semantic interfaces to data providers and users.

Acknowledgments

The work described in this paper was conducted in the “Josef Ressel Center for User-Centric Smart Grid Privacy, Security and Control”, which is supported by the Austrian Federal Ministry of Ecomomy, Family and Youth (BMWFJ) and the Austrian National Foundation for Research, Technology and Development. The authors would also like to thank René Blaschke from the company partner Salzburg AG for providing extensive test data.

References

[1] Y. Kim, M. Thottan, V. Kolesnikov, and W. Lee, “A Secure Dezentralized Data-Centric Information Infrastructure for Smart Grid,” *IEEE Communications Magazine*, vol. Energy Efficiency in Communications, pp. 58–65, Nov. 2010.

[2] Z. Fan, P. Kulkarni, S. Gormus, C. Efthymiou, G. Kalogridis, M. Sooriyabandara, Z. Zhu, S. Lambotharan, and W. Chin, “Smart grid communications: Overview of research challenges, solutions, and standardization activities,” *IEEE Communications Surveys & Tutorials*, vol. PP Issue:99, pp. 1–18, 2012.

[3] C. Müller-Elschner, “Die Rolle von Informations- und Kommunikationstechnologie beim Smart Metering,” in *Smart Metering – Technologische, wirtschaftliche und juristische Aspekte des Smart Metering*, 2nd ed., C. Köhler-Schute, Ed. KS-Energy-Verlag, 2010, pp. 87–95, in German.

[4] *Smart Grid Reference Architecture*, CEN/Cenelec/ETSI Smart Grid Coordination Group Std., Nov. 2012.

[5] *Framework Document*, CEN/Cenelec/ETSI Smart Grid Coordination Group Std., Nov. 2012.

[6] M. S. Jimenez, *Smart Grid Mandate*, European Commission Directorate-General for Energy Std., 2012.

[7] A. Veichtlbauer, T. Pfeiffenberger, and U. Schrittesser, “Generic control architecture for heterogeneous building automation applications,” in *Proceedings of the 6th International Conference on Sensor Technologies and Applications (SensorComm 2012)*, Rome, August 2012, pp. 148–153.

[8] ITU-T, “Applications of ITU-T G.9960, ITU-T G.9961 transceivers for Smart Grid applications: Advanced metering infrastructure, energy management in the home and electric vehicles 2010,” ITU-T technical paper, Series G: Transmission Systems and Media, Digital Systems and Networks.

[9] D. Engel, “Wavelet-based load profile representation for smart meter privacy,” in *Proceedings of the Fourth IEEE Conference Innovative Smart Grid Technologies (ISGT'13)*, Washington, D.C., USA, Feb. 2013, pp. 1–6.

[10] A. Barenghi and G. Pelosi, “Security and privacy in smart grid infrastructures,” in *Proc. 22nd Int Database and Expert Systems Applications (DEXA) Workshop*, 2011, pp. 102–108.

[11] V. Sood, D. Fischer, J. Eklund, and T. Brown, “Developing a communication infrastructure for the Smart Grid,” in *Electrical Power Energy Conference (EPEC), 2009 IEEE*, Montreal, QC, Canada, Oct. 2009, pp. 1 –7.

[12] M. Jung, T. Hofer, S. Döbelt, G. Kienesberger, F. Judex, and W. Kastner, “Access control for a Smart Grid SOA,” in *Proceedings of the 7th IEEE Conference for Internet Technology and Secured Transactions*, London, UK, Dec. 2012, pp. 281–287.

[13] S. Rohjans, “(S2)In – Semantic Service Integration for Smart Grids,” Ph.D. dissertation, Carl von Ossietzky University, Oldenburg, Sep. 2012.

[14] G. Vossen, *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*, 5th ed. Oldenbourg Verlag, 2008, in German.

[15] ISO/IEC, *ISO/IEC 9075-1:2011 – Information technology – Database languages – SQL*, International Organization for Standardization and the International Electrotechnical Commission Std., 2011.

[16] Ramamoorthy, K. (2012) SAP for Utilities: SAP IS-U data model - Business and Technical Master Data. Accessed: 2013-06-06. [Online]. Available: http://scn.sap.com/community/utilities/blog/2012/03/26/sap-is-u -data-model–business-and-technical-master-data

[17] Siemens AG. (2013, Mar.) Information mit System – Das automatisierte Verbrauchsdatenerfassungs- und Informationssystem AMIS. In German. Accessed: 2013-06-06. [Online]. Available: http://www.siemens.com/sustainability/pool/de/umw eltportfolio/produkte-loesungen/energieuebertragung-energievert eilung/amis-broschuere-de.pdf

[18] H. Vonhoegen, *Einstieg in XML – Grundlagen, Praxis, Referenz*, 5th ed. Galileo Computing, 2009, in German.

[19] J. Clark. (1999) XSL Transformations (XSLT) Version 1.0 – W3C Recommendation. Accessed: 2013-06-06. [Online]. Available: http://www.w3.org/TR/1999/REC-xslt-19991116

[20] F. Knirsch and O. Langthaler. (2013, Feb.) Cappatec Core Engine RxXML Specification. Accessed: 2013-06-06. [Online]. Available: http://www.cappatec.com/home/res/Cappa tec_RxXML_02_13.pdf

[21] E. Rescorla, “HTTP Over TLS,” RFC 2818 (Informational), Internet Engineering Task Force, May 2000, accessed: 2013-06-06. [Online]. Available: http://www.ietf.org/rfc/rfc2818.txt

[22] I. Sommerville, *Software Engineering*, 6th ed. Pearson Education, 2001.

[23] Bosch Software Innovations. (2013, Feb.) Inubit for Business Process Management. Accessed: 2013-06-06. [Online]. Available: http://www.bosch-si.com/technology/business-process-management-bpm/business-process-management.html