2009

A Method for Information Systems Development According to SOA

Philipp Offermann
Deutsche Telekom Laboratories, philipp.offermann@telekom.de

Udo Bub
Deutsche Telekom Laboratories, udo.bub@telekom.de

A Method for Information Systems Development according to SOA

Philipp Offermann
Deutsche Telekom Laboratories
philipp.offermann@telekom.de

Udo Bub
Deutsche Telekom Laboratories
udo.bub@telekom.de

ABSTRACT

While service-oriented architecture (SOA) is gaining momentum in research and industry, we believe the field of methods for designing information systems according to SOA still leaves room for improvement. Nevertheless, the implementation of SOA design principles demands a methodical approach. All existing methods have shortcomings of some kind, but also propose some useful concepts. In this paper, the SOA method (SOAM) is presented. It is based on an understanding of the useful concepts from existing methods and puts them together with new ideas that address the key shortcomings. It has been tested in practice at Vattenfall Europe, one of the largest European energy and heat providers and three other companies. The evaluation has shown that SOAM is well-defined, supports the design of information systems according to the SOA design principles and can be applied in practice.

Keywords

Service-oriented architecture, SOA, method, methodology, information systems development.

INTRODUCTION

Service-oriented architecture (SOA) is an architectural style to design and implement information systems. Its technical standards are well established; agreement on service design principles is growing. Still discussions about how the software architecture according to SOA is defined and how this architecture can be designed continue. When planning to introduce an SOA, a methodical approach is of high importance to design and implement business-aligned, reusable services. Additionally, companies often want to integrate their legacy systems into SOA software systems. This has to be taken into account when constructing a method for SOA.

A number of methods for designing SOA solutions exist. All existing methods have some shortcomings. Based on existing methods for SOA, the SOA method (SOAM) was developed. The goal was to design a method to design SOA solutions. The method should overcome weaknesses in existing methods; its applicability should be proven in practice. The method was evaluated for practical applicability in different companies. By this it has been shown that the method is highly usable and facilitates the design of an SOA solution.

In this article, first, the service-oriented architecture is introduced. This is the foundation on which methods for the SOA are based. Then, weaknesses of existing methods are presented. Based on the existing methods, SOAM was designed. It is explained in detail in this article. Finally, the results of the method evaluation at four companies are given and a conclusion is drawn.

SERVICE-ORIENTED ARCHITECTURE

SOA combines elements of software architecture and enterprise architecture. It is based on the interaction with autonomous and interoperable services that offer reusable business functionality via standardised interfaces. Services can exist on all layers of an application system (business process, presentation, business logic, data management). They may be composed of services from lower layers, wrap parts of legacy application systems or be implemented from scratch. (Erl, 2005; 2007; IBM, 2007; Krafzig, Banke and Slama, 2004)

The underlying principle of SOA is the service. A service is a software component that can be accessed using commonly known communication technologies. In most cases, Web service technologies are used to implement SOA software (Newcomer and Lomow, 2005). Service types can be deduced from application system layers (see figure 1):

- Business process service: A service that orchestrates other services according to a business process. Implemented e.g. by using WS-BPEL.

- Presentation service: A service that provides a user interface to perform a user interaction. Usually supported by a workflow management system.
- Business logic service: Business logic like calculations, data verifications, transformations etc. that make business sense. Usually, business process activities on the finest level of granularity are supported by this type of service.
- Data management service: Data management for business entities (data object types). Usually, operations like create, read, update and delete (CRUD) are offered.

Presentation services access business logic and data management services in order to read or write data needed or initiate business functions. Business logic services can also access data management services for reading or writing data. Business process services can access services from all other layers as required by the process; direct access to data objects is deprecated to ensure a clear separation of the layers.

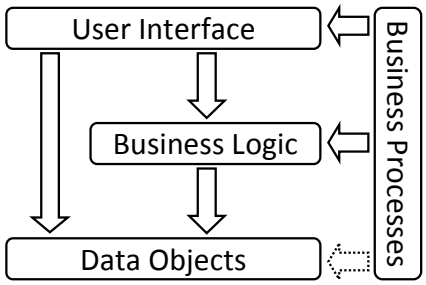

Figure 1. Application System Layers

The aim of SOA is to create reusable, flexible and business-aligned IT systems. Legner and Heutschi have summarised nine publications on design principles (Legner and Heutschi, 2007). They identified four classes of design principles.

- "Interface orientation" implies abstraction from service implementation. To call a service, only the interface description has to be known. To achieve this, interfaces have to be specified comprehensively and uniformly.
- "Interoperability" demands technical and business standardisation for the message format and the interface specification, but also for the messages themselves and the business functionality offered.
- "Autonomy/modularity" means that services should be loosely coupled and stateless, i.e. independent of previous calls. Internally services should have a high cohesion. By this, side effects during usage and maintenance are reduced.
- "Business suitability" requires that service granularity should be oriented toward business activities and services should be generalised to enable reuse.

In summary it can be said that SOA is not so much about the introduction of new concepts, but rather about a combination and integration of existing practices to achieve the architecture's goals.

EXISTING METHODS

Methods describe a way to transform an initial state to a target state. Constituting elements can be seen in figure 2. Activities explain what to do. They may be structured hierarchically e.g. in phases, activities, tasks and steps. The sequential order of the activities is the process model. For each activity, executing roles should be specified. A role can be performed by one or more persons; one person can take on several roles. Activities produce results, but may also use existing results as inputs. Results may be linked, e.g. lanes in a process model to organisation units in an organisational chart. Techniques support the generation of results and, therefore, are used by activities that need to create such results. Finally, a meta-model for the results can be specified for clarity and consistency.

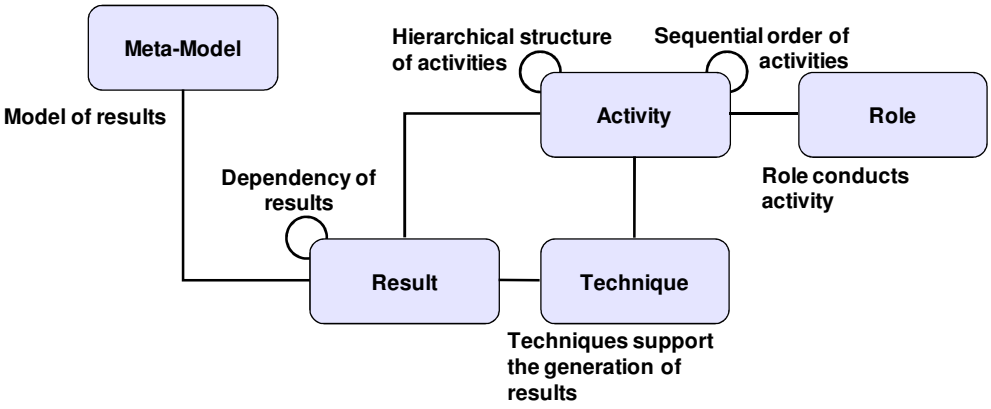

Figure 2. Method components, following (Gutzwiller, 1994)

The construction of SOAM is based on several published methods. When checking all of these methods for eligibility for immediate deployment we have encountered some major or minor shortcomings either with respect to the method description or with respect to SOA design. SOAM is designed to overcome these. An empirical comparison with some methods can be found in (Offermann and Bub, 2009).

- Erl published a very comprehensive book on SOA (Erl, 2005). Within the book, a method for SOA system development is described. The method combines a top-down with a bottom-up-approach and positions an agile procedure as a compromise. On a coarse level, the method contains the steps "service-oriented analysis", "service-oriented design", „service development", "service testing", "service deployment" and "service administration". Unfortunately, Erl's method does not sufficiently take into account legacy systems. Legacy systems are only used as a source for requirements, not as elements for system integration. Additionally, roles are not specified consistently.
- The method of Papazoglou and van den Heuvel (Papazoglou and Heuvel, 2006) encompasses the phases from planning to execution, but only analysis and design are specified in detail. During planning, it is decided if a Greenfield-approach is taken or if analysis should proceed top-down, bottom-up or out-of-the-middle. Interestingly, processes are not used for analysis but are rather treated like services. The main problem of the method is its rather complicated documentation and missing proves for its practical relevance. There is no integrated example; the application of the method seems difficult. No application in practice has been published yet.
- The method of OASIS (OASIS, 2005) has a somewhat different focus compared to the other methods. It only defines the specification of Web services. Analysis of business processes and legacy systems and the identification of services is not part of the method. Therefore, business processes are not represented in IT systems and legacy systems are not integrated.
- The method published by IBM (Arsanjani, 2004; Endrei, Ang, Arsanjani, Chua, Comte, Kroggdahl and Luo, 2004) is very well specified. It consists of the phases "domain decomposition", "goal-service model creation", "subsystem analysis", "service allocation", "component analysis", "structure components and services using patterns" and "technology realization mapping". As can already be seen from the phases, services are identified from domains and goals and not from business processes. Business processes only play a subordinate role.
- The method of Jones und Morris (Jones and Morris, 2005) is very pragmatic. It explicitly starts with a domain analysis and identifies services based on a domain decomposition. Unfortunately, the method ends after the identification of services. Further analysis and modelling is not provided. Therefore, the process of designing SOA software is not completely specified.
- Erradi et al. define a framework for SOA development (Erradi, Anand and Kulkarni, 2006). The framework consists of the phases "information elicitation", "service identification", "service definition", "service realization" and "roadmap and blanning". Unfortunately, the process description is rather confusing. Roles and modelling notations are not specified. A case study is only described very briefly, many aspects remain unclear.
- Marks and Bell describe services identification, analysis and design (Marks and Bell, 2006). However, no process model is defined. The description is narrative, mixing different aspects of a method. As the description is not systematic and is lacking a consistent chronological order, it is difficult to evaluate and compare the method.

THE SOA-METHOD

Based on the existing methods and their shortcomings, SOAM was developed in the period of September 2006 till October 2008. It specifies all method components and supports the architecture realms "workflow management", "application architecture" and "enterprise application integration". It is vendor-independent and explicitly states the architecture goals as presented above. The six phases of SOAM contain all required activities. Every activity is specified with executing roles, input and output artefacts and supporting tools. All necessary modelling notations are supported by a tool; most notations can be seen in the following examples. The SOAM tool can generate XML schemata (XSD), WSDL files and WS-BPEL process descriptions automatically from graphical diagrams. A detailed description in German can be found in (Offermann, 2008). As with SOA itself, the aim was not to create a new method, but to integrate best practices into a unified approach.

The sequence of phases can be seen in figure 3. The method uses the top-down approach and the bottom-up approach in parallel. Following the top-down approach, the company requirements are analysed. Based on this, required service operations are discovered. Following the bottom-up approach, legacy systems are identified and analysed regarding data and/or functionality that can be wrapped. Top-down requirements and bottom-up findings are then consolidated. Finally, services are designed and service properties ensured. Processes are prepared for execution. In the remainder of the section the method is explained along these phases.

Examples are given from the practical project at Vattenfall Europe. Vattenfall Europe (http://www.vattenfall.de/) is the fourth largest European electricity and heat provider. To evaluate SOAM, the process "address and bus bar management" of Vattenfall, was analysed and implemented as an SOA system. The goal for Vattenfall was to enhance the process flexibility. A qualitative evaluation of the case study is presented in the next section.

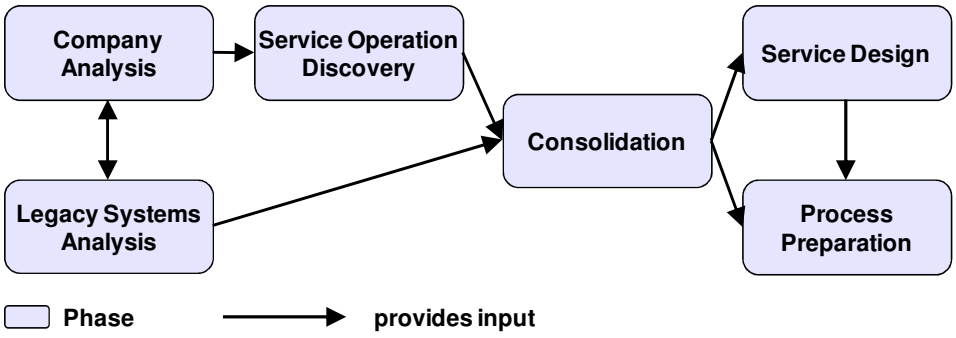

Figure 3. Phases of SOAM

Company Analysis

Company analysis focuses on business strategies, business processes and functional domains to identify company requirements. The aim of SOA is that business processes are supported by services. However, reusability can only be ensured if services are designed to support common functionality, thereby requiring a functional rather than a process perspective. SOAM combines both perspectives and values them equally. All activities in this phase are performed by a business analyst.

In a first optional step, a business motivation model is created according to the OMG specification. Based on the company strategies, business processes required to support the company's strategies can be identified. Next, business processes or parts of business processes that should be supported by SOA systems have to be identified. Among the criteria for deciding which processes to choose are the frequency of process changes, the current level of integration of legacy systems along the process and how far the process can be predetermined and requires a division of work.

The processes are modelled down to a level of granularity where each process activity is a single step from the business perspective. A good indicator for this is a clear, indivisible transformation on a data object. Each task is classified according to whether it can be executed automatically (class "business service"), whether it requires an interaction between a user and the software system (class "user interaction") or whether the activity is purely manual (class "manual"). The modelling notation supported by the SOAM tool is the Business Process Modeling Notation (BPMN) (OMG, 2009). The data flow

should be annotated as far as possible in the process models. Each data object (e.g. "customer", "order") is created in a data model (see below). If a domain data model exists in a company, it should be used for annotating the process data flow.

In figure 4, a partial business process of the business process "address and bus bar management" on the finest granularity level from the project at Vattenfall Europe can be seen. The sequence of activities, data objects and the data flow are modelled.

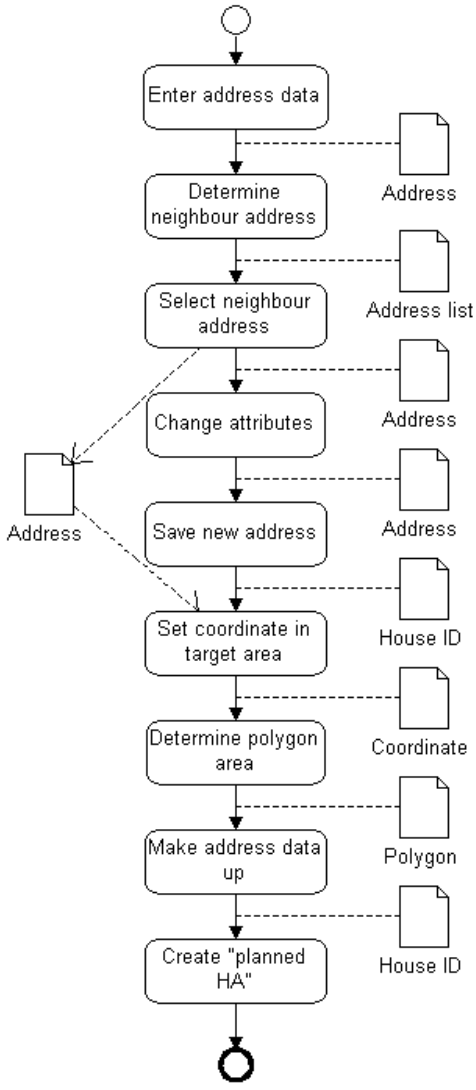

Figure 4. Business process "Address and bus bar management", finest granularity

Parallel to business process modelling, a layered model of functional domains (FD) is created. The top levels of the model can be guided by a functionally organised organisation chart. On lower levels, data objects used in an FD can be used as a guideline to identify refined functional domains. The modelling notation used for the FD model is based on UML Use Case diagrams. Each activity in business process models is linked to a functional domain. If there are more than about seven activities assigned to an FD, the FD should be refined by a detailing FD diagram and the activities reassigned. Business process and FD modelling is iterative. The link between activities and functional domains is later used to ensure the reusability of services. The model of relevant functional domains of Vattenfall Europe can be seen in figure 5.

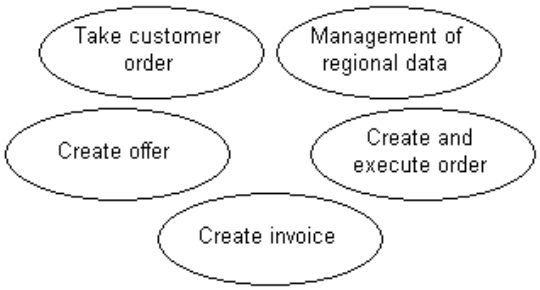

Figure 5. Model of relevant functional domains of Vattenfall Europe

Service Operation Discovery

After having analysed business processes and functional domains, service operations that support the process activities can be discovered. This is done by FD. The activities in this phase are performed by a software architect. For each FD, all activities assigned are analysed per class. For service tasks, service operations that support the activities are modelled in a service model. For user tasks, the user interface (UI) is specified. If two tasks are similar, a service operation that supports both activities can be defined. Each individual service operation and UI is assigned to a functional domain, usually to the same unit as the activity the operation supports. As there is no standardised modelling notation for service, an own notation has been developed (see figure 6).

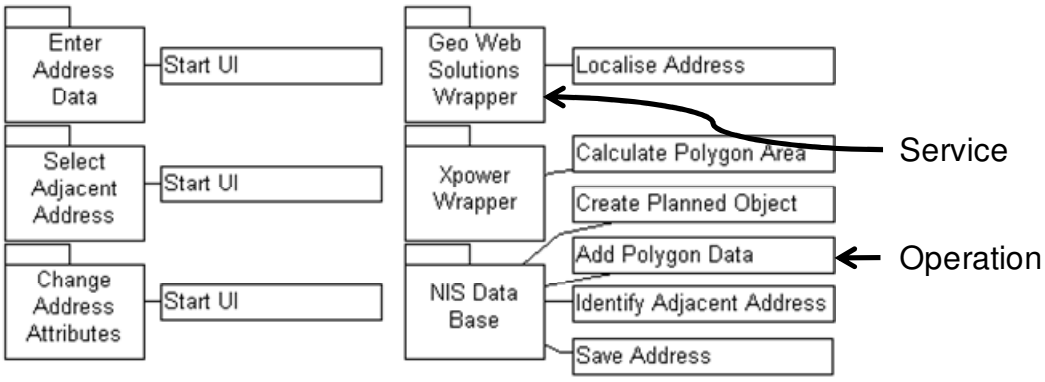

Figure 6. Vattenfall Europe Services as modelled in the SOAM tool

Until now, the discovered service operations and UI only support business process activities. This enables process flexibility. Reusable services could also exist on a more technical level. Therefore, service operations and UI use cases are modelled. These are use cases as seen from an operation or UI point of view (e.g. “get clients” and “store order” for an order creation dialog). The use cases are again assigned to FD and classified (e.g. “technical service”, “database service”). The UML Use Case diagram is used as modelling notation. The use cases are then ordered by FD and class. If there are similar or equal use cases, a service operation can be defined to support these use cases. When implementing the original service or UI, the service operations derived by the use cases can be called.

Legacy Systems Analysis

Usually, legacy systems have to be integrated in the SOA software systems. Therefore, in parallel to the company analysis, legacy systems that are used to support the considered processes are analysed. First, an IT analyst has to identify the relevant systems. Then, these systems are analysed regarding functionality and/or data that can be accessed by a wrapping service. Possibilities are existing API, accessible data bases, software components that can be wrapped or even command line invocation. A modelling notation for legacy systems has been developed to support this activity. The legacy systems model of relevant systems of Vattenfall Europe can be seen in figure 7.

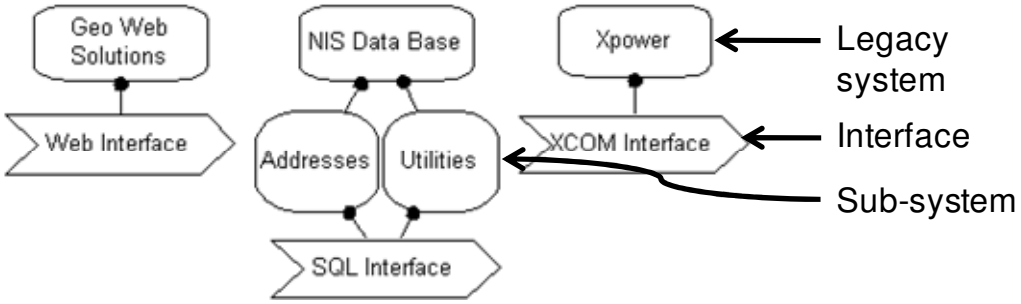

Figure 7. Relevant legacy systems of Vattenfall Europe

Every possibility found to be relevant is then modelled as a service with service operations. Operations are assigned to an FD. Also, used data objects are modelled in the data model. Each object in the data model is assigned to an FD. In the SOAM tool, the UML profile for Core Components is used as modelling notation.

Consolidation

After analysing top-down requirements and identifying existing systems bottom-up, consolidation is necessary. It is performed by the software architect and done for data object types and for service operations. For data objects, redundancies have to be identified and merged (e.g. data object "address with country" vs. data object "address without country" where a merged data object can combine all fields from both data objects). This can be done by FD if functional domains have been annotated to data object types. When merging, data object types from legacy systems have precedence because the systems already exist. Only if the discrepancy with top-down requirements is so big that the department concerned is willing to provide financing, a change of a legacy system can be considered. In case two object types cannot be merged, a mediator has to be specified. The model of relevant data objects types from Vattenfall Europe can be seen in figure 8.

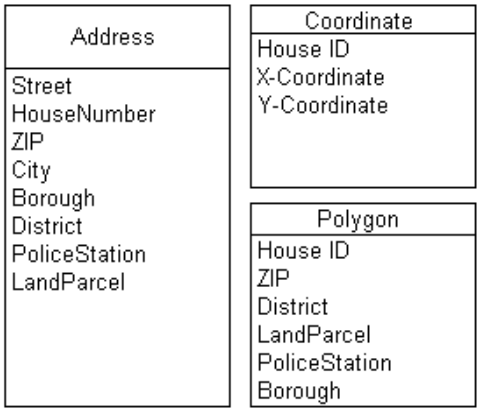

Figure 8. Data object types from Vattenfall Europe

Service operation consolidation is also done by FD and guided by class. Operations that are similar (e.g. if "get customer data" and "get customer address" both return address data) are merged. Again, operations from legacy systems have precedence. A change of a legacy system can be considered, if the discrepancy between requirements and existing system is too big. When the operations are consolidated, data input and data output is added for each operation based on the consolidated data model.

Service Design

Once a uniform model of service operations has been established, services can be designed. Service consists of several service operations. Therefore, service operations have to be grouped by service. The software architect uses rules specific to each service class. Operations of class "business service" are grouped by business logic. Operations of class "technical service" are grouped by shared code. Each operation of class "user interaction" is usually represented by a single service. Database operations are grouped by business entity. Again, only operations from the same FD are considered for grouping. Therefore, there will only be very few services per FD.

Once services are designed, the design principles not already considered during earlier steps have to be enforced. For each service, the software architect has to consider if it is stateless, if rollback operations exist to implement transactional behaviour and if services are loosely coupled. To manipulate the service, the software architect can use operations such as "unification", "intersection", "decomposition", "subset" and "substraction" (Marks and Bell, 2006). If services become too big during this process, they have to be split up and rerun through the steps above. The services designed for Vattenfall Europe can be seen in figure 6. Finally, services are implemented and tested. For these steps, known software engineering methods can be used.

Process Preparation

After the services are defined, the processes have to be prepared for execution by the developers. Service operations have to be assigned to each activity that is supported by a service. Exception handling that has not been modelled before has to be included in the process models. The data flow has to be verified to ensure executability. If necessary, data mediators have to be implemented. The user interfaces that have been designed earlier have to be included in the process. Usually, these steps heavily depend on the middleware infrastructure.

EVALUATION

The evaluate applicability of the method, it was used in four companies: Vattenfall Europe, Bosch und Siemens Hausgeräte (B/S/H), Prevent DEV and Ideal Lebensversicherung. For each of the case studies, the setting is described and key findings are presented.

Vattenfall Europe

Project Setting

Vattenfall Europe AG is one of the leading power suppliers in Germany. It is involved in coal production and power generation from coal, water and nuclear energy and transport and trading of power. The project trying out SOAM was commissioned by the IT strategy department and took place from October 2006 till February 2007. The task was to create a prototypical SOA implementation of the address and bus bar management process.

Method Application

For applying SOAM at Vattenfall Europe, the process and relevant functional domains as well as the legacy systems involved were modelled as shown in the examples in the previous section. Data objects and service operations have been derived according to the method. Resulting services can be seen in figure 6. The services and the process were implemented using Sun Java CAPS. This shows that SOAM is compatible with third party frameworks.

Findings

The case study showed that the method can be applied in practice and that the services and the process model generated by the method can be developed using off-the-shelf SOA middleware. Method quality attributes like flexibility of the software architecture, applicability of the method and adequate activity descriptions could be shown. The flexibility of the new software architecture was shown by example process changes on the implemented scenario. The successful application by practitioners in a real-world scenario and additional inquiries during application showed that the method is adequate to solve real-world problems.

The following key findings were incorporated in the method after the evaluation:

- During service identification, questions about the right granularity level for process descriptions arose. The following guideline was developed: Activities in business process models should be fine-grained enough to represent every

processing step of a business data object. Thereby, it is ensured that service operations identified from activities support business functions and at the same time are detailed enough for technical specification.

- Due to the common confusion of "service" and "operation", originally it was planned to identify services directly from business process activities. During the case study it became evident that it is better to identify service operations rather than services. Grouping of service operations into services is performed after identification and consolidation.

During service design it became clear that there are certain principles that apply to every type of service. Based on this, rules have been developed for every service type to enforce service design principles during service design.

Prevent DEV

Project Setting

The Prevent Group (http://www.preventgroup.com) is a globally operating company in the automotive industry. The company has over 10,000 employees distributed over 35 sites. Prevent DEV, the part of the company SOAM was applied in, is responsible for development and distribution in Germany. The aim of the project of Prevent DEV was to design SOA software support for project planning. No legacy systems were taken into account. The project took place from October 2007 till March 2008.

Method Application

Schooling for employees took one day. Further questions were answered by phone. At the time, there was no example integrated in the method description. This was seen as a major shortcoming during schooling. Models of the business motivation, processes, data objects and functional domains were created using the ARIS Toolset (http://www.ids-scheer.com/en/ARIS/). Thereby, it was shown that SOAM can be supported by different modelling tools.

Findings

The business processes to be supported by an SOA system were focused on user interactions. For these processes it has to be ensured that every activity makes sense from a business point of view. Control logic of user interfaces should not be included in business process models. Once these points had been clarified and taken into account, the method was applied without problems.

IDEAL Lebensversicherung

Project Setting

The IDEAL Lebensversicherung (IDEAL, http://www.ideal-versicherung.de) sells insurance products for a target group of 50 years or older. They are the only insurance company in the German market focusing on this segment. A project trying out SOAM to redesign software support for tariffing and offerings generation took place from October 20087 till August 2007.

Method Application

During method application, all models as described above were created using the MetaEdit+ tool. It was tried to integrate reference process and data models from the industry standardisation initiative BiPRO (http://www.bipro.net). The activity was stopped at some point as it was seen too complicated for the small-scale project.

Findings

The case study was especially interesting with respect to what not to do. When applying the method, the researcher involved did not communicate enough with relevant professionals from the company. While method application was successful from an exterior point of view, professionals from the company identified considerable differences between processes modelled and processes actually used. As a result, designed services were also reconsidered. The key finding was that during company analysis and legacy systems analysis, a close communication between method user and professionals has to be assured. This is especially true as every problem in the business process models will be looped through to final service design.

BSH

Project Setting

Bosch und Siemens Hausgeräte (BSH, http://www.bsh-group.de) is a group of companies with worldwide activities. Annual turnover in 2007 was more than 8.8 billion Euros. BSH has about 39,000 employees in over 40 countries. BSH is planning to introduce a new system architecture for the product lifecycle management. SOAM was used in the period from February 2008 till September 2008 to design parts of the system.

Method Application

For applying the method, business process models that already had been created in the ARIS Toolset where used as a basis. For SOA software support, the models had to be refined in some areas. Additional models were created in Microsoft Visio 2007. All diagram types could be modelled without problems. While applying the method, every method activity was evaluated in respect to the following criteria: usefulness of activity, activity description, understandability of modelling notations, relevance of activity in context, granularity of activities and tool support. Thereby, critical points could be identified and addressed in the next version of the method documentation.

Findings

From a project management point of view, steps were lacking within the method to create and adjust project schedules. Another problem arose from the fact that SOAM requires some kind of business process thinking. Within BSH, processes are managed by boards that are constituted by members from the organisational units involved. This provided a weak basis for a process oriented architecture as proposed by SOAM. Generally, it is helpful to introduce business process management before or together with an SOA solution.

CONCLUSION

SOA promises an increase in flexibility and efficiency of enterprise software systems. In order to implement information systems according to the service-oriented architecture – in addition to standardised technologies – the application of a well-defined method is recommended to ensure service design principles as well as an efficient, uniform software design process. The newly developed SOAM is complete regarding the constituting elements. The method defines activities in six phases and supports the top-down approach as well as the bottom-up approach. For every activity, executing roles, input and output models, supporting tools and detailed steps are defined. The six modelling notations that are used are as far as possible based on standards.

The method was evaluated using case-studies in four companies. By this it was shown that SOA solutions can be designed using the method and that in practice the detailed definition and the tool support create advantages over prior company practices. Because standards are applied for modelling and technical artefact generation, the integration with existing middleware frameworks can be realised. Also, the method is independent of the modelling tool used.

In future, the method will be kept up-to-date with the development of SOA middleware frameworks as well as with developments of the SOA concepts. Furthermore the application of the method in various companies is ongoing to improve the method.

REFERENCES

Arsanjani, A. (2004) Service-oriented modeling and architecture, http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/.

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Kroggdahl, P. and Luo, M. (2004) Patterns: Service-Oriented Architecture and Web Services, IBM International Technical Support Organization.

Erl, T. (2005) Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, Upper Saddle River, NJ.

Erl, T. (2007) SOA: Principles of Service Design, Prentice Hall, Upper Saddle River, NJ.

Erradi, A., Anand, S. and Kulkarni, N. (2006) An Architectural Framework for Service Definition and Realization, in IEEE Computer Society (Ed.) *Conference on Services Computing* Chicago, IL, USA, 151-158.

Gutzwiller, T.A. (1994) Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen, Physica, Heidelberg.

IBM (2007) Service Oriented Architecture — SOA, http://www-306.ibm.com/software/solutions/soa/offerings.html?S_TACT=107AG01W&S_CMP=campaign.

Jones, S. and Morris, M. (2005) A Methodology for Service Architectures, Capgemini, http://www.oasis-open.org/committees/download.php/15071/A%20methodology%20for%20Service%20Architectures%201%202%204%20-%20OASIS%20Contribution.pdf.

Krafzig, D., Banke, K. and Slama, D. (2004) Enterprise SOA - Service-Orientied Architecture Best Practices, Prentice Hall, New Jersey.

Legner, C. and Heutschi, R. (2007) SOA Adoption in Practice - Findings from Early SOA Implementations, in H. Österle, J. Schelp and R. Winter (Eds.) *European Conference on Information Systems (ECIS 2007)*, St. Gallen.

Marks, E.A. and Bell, M. (2006) Executive's Guide to Service-Oriented Architecture, John Wiley & Sons, Hoboken, New Jersey.

Newcomer, E. and Lomow, G. (2005) Understanding SOA with Web Services, Addison-Wesley, Upper Saddle River, NJ.

OASIS (2005) Web Service Implementation Methodology, http://www.oasis-open.org/committees/download.php/13420/fwsi-im-1.0-guidlines-doc-wd-publicReviewDraft.htm#_Toc105485380.

Offermann, P. (2008) SOAM - Eine Methode zur Konzeption betrieblicher Software mit einer Serviceorientierten Architektur, *Wirtschaftsinformatik*, 50, 6, 461-471.

Offermann, P. and Bub, U. (2009) Empirical Comparison of Methods for Information Systems Development According to SOA, *17th European Conference on Information Systems*, Verona, Italy.

OMG (2009) Business Process Modeling Notation (BPMN), http://www.omg.org/spec/BPMN/1.2.

Papazoglou, M.P. and Heuvel, W.-J. (2006) Service-Oriented Design and Development Methodology, *International Journal of Web Engineering and Technology*, 2, 4, 412-442.