2009

Empirical comparison of methods 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

EMPIRICAL COMPARISON OF METHODS FOR INFORMATION SYSTEMS DEVELOPMENT ACCORDING TO SOA

Offermann, Philipp, Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany, philipp.offermann@telekom.de

Bub, Udo, Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany, udo.bub@telekom.de

Abstract

While service-oriented architecture (SOA) as an architectural principle for information systems is gaining momentum in research and industry, the field of methods for designing information systems according to SOA is still poorly developed. However, the implementation of SOA design principles, e.g. service reusability, business alignment and autonomy, demands a methodical approach. In order to overcome the shortcomings of current methods, we have developed the SOA method (SOAM). The method is based on existing methods for SOA. Activities are specified along with roles, techniques, modelling notations and a meta-model. A tool supports all necessary modelling notations as well as the generation of XSD, WSDL and WS-BPEL from the models. The newly developed method has been compared to other methods using a laboratory experiment with students. Different methods have been used on different company scenarios; the results have been recorded using a questionnaire. Results show that according to the evaluated criteria, SOAM together with IBM's method gets the best scores. With respect to the alignment of the software architecture with business processes, one of the primary goals of SOA, SOAM received a better rating than IBM's method.

Keywords: Method construction, service-oriented architecture, SOA, design science, software engineering

1 INTRODUCTION

The service-oriented architecture (SOA) provides concepts for designing and implementing information systems. Its widespread adoption has been made possible by generally accepted technical standards such as WSDL (Web Service Description Language), SOAP (formerly Simple Object Access Protocol, now proper name), UDDI (Universal Description, Discovery and Integration) and WS-BPEL (Business Process Execution Language). While technical standards have reached an acceptable level and agreement on SOA design principles is growing, discussions on how the software architecture according to SOA is to be defined and how the architecture can be designed continue. When planning to introduce an SOA, a methodical approach is of great importance if business-aligned, reusable services are to be designed and implemented.

We have developed the SOA method (SOAM) based on existing methods for SOA. The method should overcome weaknesses in existing methods; its applicability should be proven in practice. The method that we developed specifies activities, necessary roles and modelling notations and is supported by a tool. We have evaluated the method in different companies using action research. It was possible to show that the method is highly usable and facilitates the design of an SOA system that adheres to the SOA design principles. In this article, the results of a laboratory experiment comparing SOAM to existing methods are presented.

First, service-oriented architecture is introduced. This is the foundation on which methods for SOA are based. Then the weaknesses of existing methods are presented. Our SOA method is explained in detail in this article. Finally, the results of the laboratory experiment are described.

2 SERVICE-ORIENTED ARCHITECTURE

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.

The underlying principle for 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. Service types can be deduced from application systems layers:

- 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. Data processing is not done automatically, but by the user.
- Business logic service: business logic like calculations, data verifications, transformations etc. that make business sense. Business process activities on the finest level of granularity are usually 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.

The aim of SOA is to create reusable, flexible and business-aligned IT systems. Especially the alignment of software systems with business processes is a step that has not been achieved yet. In order to facilitate the alignment, methods have to take into account business requirements for the design of technical artefacts (Erl 2007). Legner and Heutschi (2007) have summarised nine

publications on design principles. They identified four classes of design principles: interface orientation, interoperability, autonomy/modularity and business suitability.

3 EXISTING METHODS

Methods (sometimes also called methodologies) describe a way to transform an initial state into a target state. (Cronholm and Ågerfalk 1999; Wynekoop and Russo 1997) Activities explain what has to be done. 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. 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 on 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.

The construction of SOAM is based on several published methods. All of these methods have some major or minor weaknesses. SOAM is designed to overcome these.

- Erl published a very comprehensive book on SOA (Erl 2005, pp. 366-370). 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 the confusing 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, Krogdahl, Luo and Newling 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 goal and not from business processes. Business processes only play a subordinate role. We are conducting a laboratory experiment to determine which approach gives better results.
- 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.

4 THE SOA-METHOD

We developed the SOA method (SOAM) based on the existing methods and their shortcomings. It has a high integrity and consistency regarding the constituent elements and supports the architecture realms "workflow management", "application architecture" and "enterprise application integration". It is vendor-independent and explicitly states the architecture goals, which is not the case for with any other method. The six phases of SOAM contain all necessary activities, various activities containing several steps. Every activity is specified with executing roles, input and output artefacts and supporting tools. All necessary modelling notations are supported by a tool. The SOAM tool can generate XML schemata (XSD), WSDL files and WS-BPEL process descriptions directly from the graphical models. Action research on the method has bees published in (Offermann 2008).

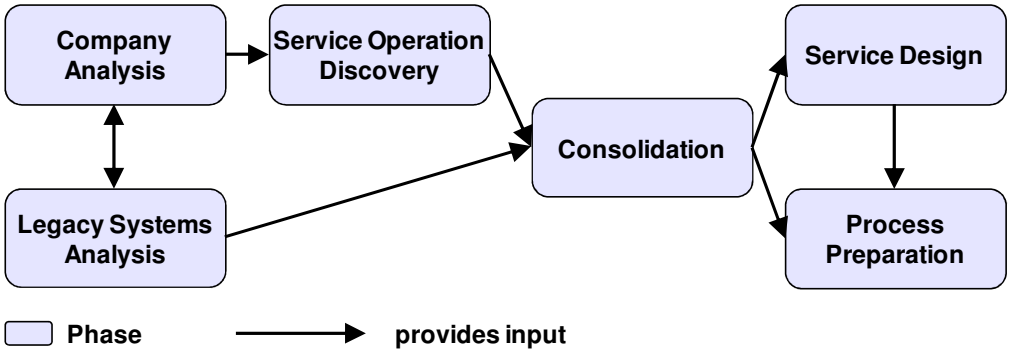

Figure 1. Phases of SOAM

The sequence of phases can be seen in figure 1. The method uses the top-down approach and the bottom-up approach in parallel. The company requirements are analysed following the top-down approach. Required service operations are discovered based on this. 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.

4.1 Company Analysis

Company analysis focuses on business strategies, business processes and functional domains to identify company requirements. The guideline is that business processes have to be supported by services, but reusability can only be ensured by functional specialisation. 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 (OMG 2006a). 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 an SOA system 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 the degree to which the process can be predetermined and is based on division of labour (Heutschi 2007).

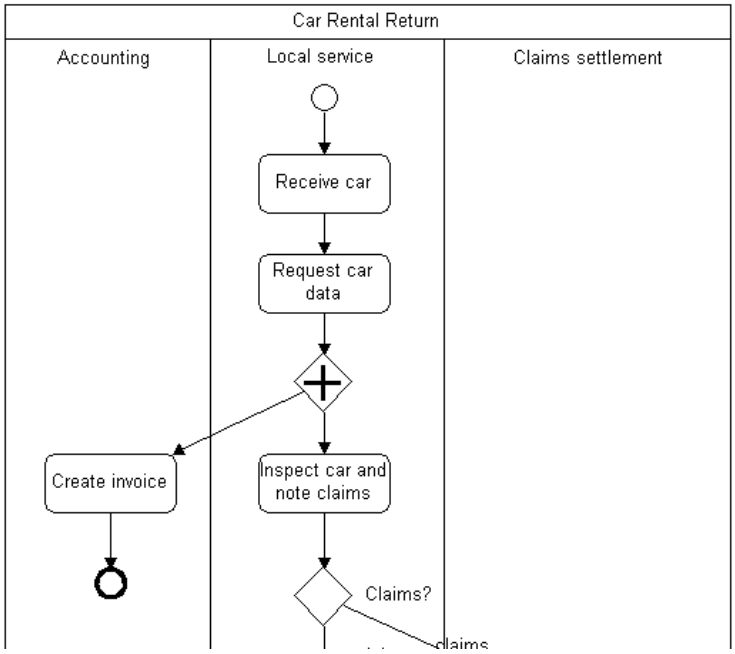

Figure 2. Business process "Car Rental Return", finest granularity

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, non-divisible 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 2006b). 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. Figure 2 shows a business process on the finest granularity level modelled in BPMN.

Figure 3. Functional domain model, coarse 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 organisational 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 seven activities assigned to an FD, the FD has to be refined and the activities reassigned. Business process and FD

modelling are iterative. The link between activities and functional domains is later used to ensure the reusability of services. Figure 3 shows a model of functional domains.

4.2 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 services, we have developed a notation of our own (see figure 6).

Until now, the service operations and UI discovered only support business process activities. This facilitates 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. “business logic”, “data management”). The UML Use Case Diagram is used as a 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 them. When implementing the original service or UI, the service operations derived by the use cases can be called.

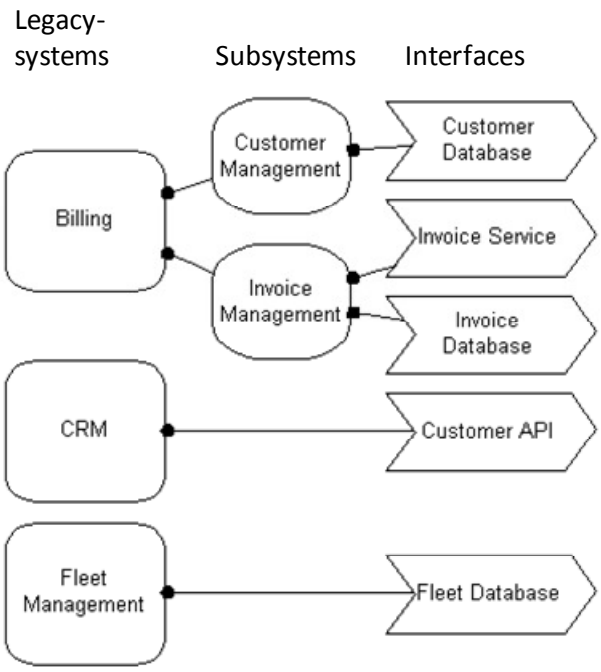

Figure 4. Legacy Systems Model

4.3 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 processes are analysed. First, an IT analyst has to identify the relevant systems. Then, these systems are analysed with regard to 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. Figure 4 shows a legacy systems model.

Every possibility found 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 (UN/CEFACT 2006a; UN/CEFACT 2006b) is used as a modelling notation.

4.4 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. address with country vs. address without country). 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 too big can a change in a legacy system be considered. In case two object types cannot be merged, a mediator has to be created. Figure 5 shows the model of data objects types.

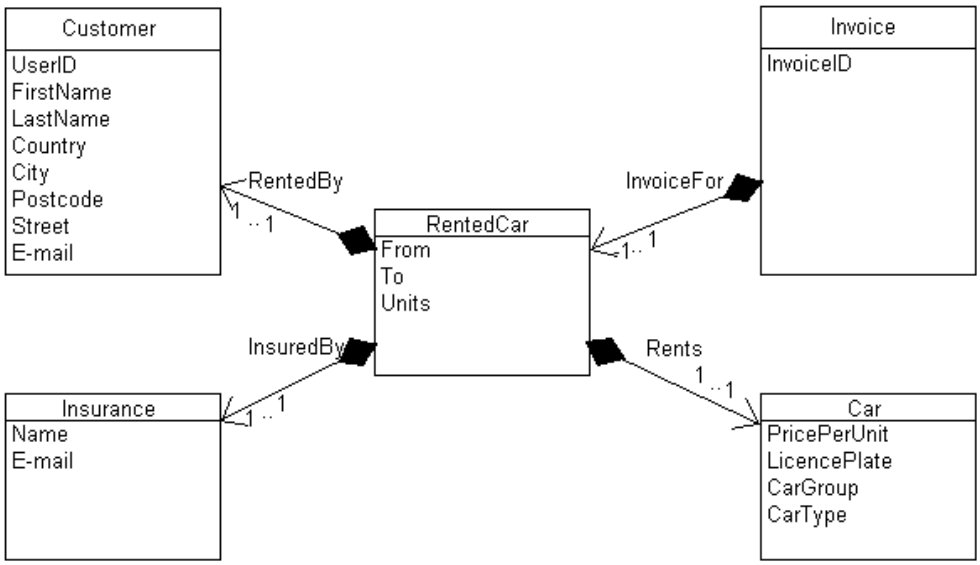

Figure 5. Data object types from Vattenfall Europe

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

4.5 Service Design

Once a uniform model of service operations has been established, services can be designed. Usually, a service consists of several operations. Therefore, service operations have to be grouped by service. The software architect uses rules specific to each service class. Operations of class “business logic” are grouped by business logic. Each operation of class “user interaction” is usually represented by a

single service. Data management 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 that could not be guaranteed 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 et al. 2006, pp. 111-117). If services become too big during this process, they have to be split up and rerun through the steps above. Figure 6 shows a simple service model. Finally, services are implemented and tested. For these steps, known software engineering methods can be used.

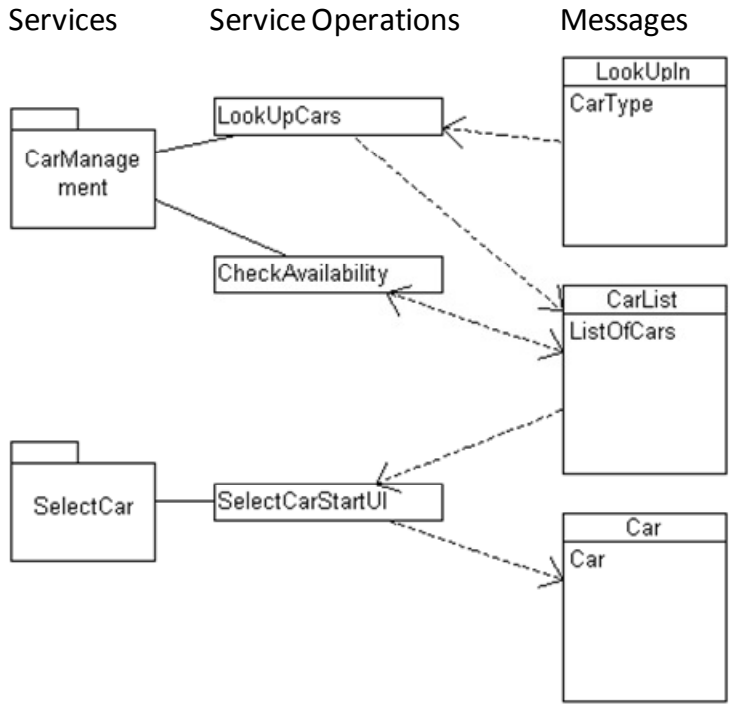

Figure 6. Services as modelled in the SOAM tool

4.6 Process Preparation

After the services have been 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 built. Finally, the tool translates the BPMN models into WS-BPEL. The user interfaces that have been designed earlier have to be included in the process specification. Usually, these steps depend heavily on the middleware infrastructure.

5 EMPIRICAL METHOD COMPARISON

A laboratory experiment was carried out to compare the newly developed SOAM with SOMAM, MSA, SDS, WSIM and SODDM. The comparison was done by using the methods on different company scenarios, rating the experiences of the application using a questionnaire and carrying out a

subsequent statistical analysis of the answers. A company scenario consists of models of a company that are relevant for the application of the methods. The models represent e.g. business processes, organisational charts, data object types and legacy systems. By using predefined company scenarios, it was possible to apply different methods under the same conditions and thereby to retrieve comparable results. This is usually not possible in a company setting. Often, companies are not prepared to provides the resources necessary to apply several methods. Also, because of a possible time shift of method application, general conditions (e.g. legacy systems, staff responsible) might change. On the other side, practical relevance of a laboratory experiment might be limited. A practical application of SOAM is presented in (Offermann 2008).

The first part of the questionnaire is based on the theoretical model for validating information systems design methods by Moody (2003; Moody, Sindre, Brasethvik and Sølvberg 2002). Using a questionnaire, "perceived ease of use", "perceived usefulness" and "intention to use" can be evaluated. Because all methods evaluated were concerned with SOA, for the usefulness, in addition to general questions, the quality of the realisation of SOA design principles and the achievement of goals of enterprise application system design were evaluated. In the second part of the questionnaire, completeness, consistency and applicability of the method's components (Greiffenberg 2004) were evaluated.

When designing the questionnaire, 5-point Likert scales were used to simplify the analysis. As the survey was conducted in German, English translations are given here. For the first part of the questionnaire, a scale of accordance strongly agree (5), agree (4), undecided (3), disagree (2), strongly disagree (1) was used. For the second part, a scale of intensity extremely (5), a great deal (4), moderately (3), a little bit (2), not at all (1) was used. The German version of the scales used corresponds to the scales used by Bortz and Döring (2002). They are equidistant as far as possible. The numerical value used for the statistical analysis is given in brackets.

The survey was conducted amongst graduate students of computer sciences and industrial engineering in the time from October 2007 till February 2008 who, alone or in pairs, used different methods on a scenario. According to (King and He 2006), results from students of relevant subjects are comparable to results from professionals. Because the documentations were sufficiently well specified, the methods SOAM, SOMAM, MSA, SDS, WSIM and SODDM were used. To reduce falsification due to the properties of a specific scenario, every team used a different scenario. The scenarios used were "order processing of a mail order snowboard store", "material management and provisioning of an automotive manufacturer", "conduction of an acquisition event of a consulting company", "order processing in industry" and "address and bus bar management of a power utility company". The student were handed all available written documentation for each method. No questions related to methods were answered.

There were 43 usable responses to the questionnaire. The focus of the method comparison was, in accordance with a prior theoretical comparison, on SOAM and SOMAM. These methods were used most often. For analysing the first part of the questionnaire, first of all the validity is checked using factor analysis. The analysis of the questions concerning usability shows that one component can explain 70% of the variance. This is also the only component with an eigenvalue bigger than 1. For the questions concerning usefulness, only 62% of the variance can be explained by one component. By leaving out questions concerning "clearly specified service contracts" and "abstraction from implementation", the value is increased to 66% so that only one component has an eigenvalue bigger than 1. For the questions concerning the intention of use, one component explains 94% of the variance.

The second part of the questionnaire evaluates the completeness, consistency and applicability of a method components' specification. The questions relating to "roles" and "techniques and tools" had to be excluded from the analysis as not all methods specify these components and therefore there are only a limited number of answers. By using an inter-item correlation analysis, it can be shown that the answers for completeness, consistency and applicability are highly correlated (see table 1). All

correlations are highly significant and have a value bigger than 0.6. Therefore, the answers to the three parts of the questions are combined. A factor analysis of the combined questions shows that 88% of the variance can be explained by one component. Hence the quality of the method components can be interpreted as one construct.

Question	Part 1	Part 2	Correlation	Significance
Kom1				
	Completeness	Consistency	0.696	0.000
	Completeness	Applicability	0.725	0.000
	Consistency	Applicability	0.760	0.000
Kom4				
	Completeness	Consistency	0.728	0.000
	Completeness	Applicability	0.655	0.000
	Consistency	Applicability	0.801	0.000
Kom7				
	Completeness	Consistency	0.836	0.000
	Completeness	Applicability	0.762	0.000
	Consistency	Applicability	0.748	0.000
Kom8				
	Completeness	Consistency	0.885	0.000
	Completeness	Applicability	0.884	0.000
	Consistency	Applicability	0.927	0.000

Table 1. Correlation of the method components' completeness, consistency and applicability

Finally, the reliability of the constructs is verified by calculating Cronbach's Alpha for the four constructs "usability", "usefulness", "intention to use" and "quality of method components". The results are summarised in table 2. All values are bigger than 0.8. This means that more then 80% of the variance is systematic and must be attributed to errors in measurement.

Construct	Cronbach's Alpha
Usability	0.889
Usefulness	0.947
Intention to use	0.939
Quality of method components	0.956

Table 2. Reliability of constructs

Table 3 contains an overview of the results of the method comparison. In the categories usefulness, intention to use and quality of method components, SOAM received the best ranking, followed by SOMAM. For usability, MSA scored best.

Construct	SOAM	SOMAM	MSA	SDS	WSIM	SODDM
Usability	3.84	3.60	*3.88*	3.17	2.67	1.65
Usefulness	*4.17*	3.85	2.91	3.47	3.07	1.64
Intention to use	*4.27*	4.05	3.30	3.50	2.25	1.00
Quality of method components	*3.99*	3.69	3.33	3.30	2.94	1.48

Table 3. Summary of results of method comparison (Likert scale 1-5)

By using a t-test against the value 3, the significance of a deviation from a neutral ranking was determined (table 4). Values smaller than 0.05 are counted as being significant. "+" means that the value is significantly bigger than 3. "-" means that the value is significantly smaller than 3. "o" means that no decision can be made. The results for SOAM and SOMAM are significantly bigger than 3. For

SODDM, the results are significantly smaller than 3. The significance of the value for the intention to use of SODDM could not be calculated because the standard deviation was 0.

Construct	SOAM	SOMAM	MSA	SDS	WSIM	SODDM
Usability	+ (0,002)	+ (0,009)	+ (0,006)	o (0,510)	o (0,374)	- (0,009)
Usefulness	+ (0,000)	+ (0,000)	o (0,589)	o (0,060)	o (0,793)	- (0,019)
Intention to use	+ (0,000)	+ (0,012)	o (0,374)	o (0,345)	o (0,076)	- (-)
Quality of method components	+ (0,000)	+ (0,000)	o (0,139)	+ (0,045)	o (0,842)	- (0,012)

Table 4. Significance of deviation of Results from median value 3

A t-test on mean equality between SOAM and SOMAM shows that no significant decision can be taken if the values are actually different (table 5). Therefore, it is not clear which of the two methods is better.

Construct	Equal variances	Unequal variances
Usability	0.390	0.387
Usefulness	0.087	0.085
Intention to use	0.551	0.565
Quality of method components	0.086	0.083

Table 5. Significances of t-test on mean equality between SOAM and SOMAM

One of the architecture goals of enterprise application systems is the alignment of IT functions to business processes (Heutschi, (Heutschi 2007)). Because of the similarity of the results between SOAM and SOMAM, the answers to "The method ensures a good representation of processes in the software architecture." were analysed separately. For SOAM, the mean value was 4.09 (agree) with a high significance (0.000) of being more than 3 (t-test on neutral value). For SOMAM, the mean value was 3.40 with a significance of 0.269. Therefore, for SOMAM it can't be excluded that the real ranking is "undecided". This is probably due to the fact that SOMAM doesn't analyse business processes directly. It can be deduced that by using SOAM, business processes are better represented in the resulting SOA-system then by using SOMAM.

6 CONCLUSION

The field of methods for SOA is still poorly developed. In this article, the SOA-method (SOAM) is presented. It aims at overcoming the weaknesses of existing methods. SOAM was developed using action research. To compare SOAM to existing methods, a laboratory experiment was performed. Evaluating the results of the experiment, it can be said that the ranking of all methods is relatively close to neutral. It is important to investigate whether there is a general problem in the way methods are being described. The laboratory experiment nonetheless shows significant differences in quality between the methods. SOAM and SOMAM received the best results. MSA has high usability but comparatively low usefulness. The goal to align IT functions, especially services, to business processes, can be better reached using SOAM rather than SOMAM.

References

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

Bortz, J. and Döring, N. (2002) *Forschungsmethoden und Evaluation,* Springer, Berlin.

Cronholm, S. and Ågerfalk, P. J. (1999) *On the Concept of Method in Information Systems Development,* Linköping Electronic Articles in Computer and Information Science, 4 (19).

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, M. and Newling, T. (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) *SOAF: An Architectural Framework for Service Definition and Realization*, IEEE International Conference on Services Computing (SCC 2006), Chicago, IL, pp. 151-158.

Greiffenberg, S. (2004) *Methodenentwicklung in Wirtschaft und Verwaltung,* Dr. Kovac.

Heutschi, R. (2007) *Serviceorientierte Architektur,* Springer, Berlin.

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

King, W. R. and He, J. (2006) *A meta-analysis of the technology acceptance model,* Information & Management, 43 pp. 740-755.

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

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

Moody, D. L. (2003) *The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods*, European Conference on Information Systems (ECIS 2003), Naples, Italy.

Moody, D. L., Sindre, G., Brasethvik, T. and Sølvberg, A. (2002) *Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework*, Proceedings of the 21st International Conference on Conceptual Modeling Springer, pp. 380-396.

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), pp. 461-471.

OMG (2006a) *Business Motivation Model (BMM) Specification,* http://www.omg.org/cgi-bin/doc?dtc/2006-08-03.

OMG (2006b) *Business Process Modeling Notation Specification,* http://www.omg.org/cgi-bin/doc?dtc/2006-02-01.

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

UN/CEFACT (2006a) *ISO\DTS 15000-5: 2006 Core Components Technical Specification 2nd Edition UN/CEFACT Version 2.2,* http://www.untmg.org/index.php?option=com_docman&task=docclick&Itemid=137&bid=43&limitstart=0&limit=5.

UN/CEFACT (2006b) *UML Profile for Core Components (BCSS),* http://www.untmg.org/index.php?option=com_docman&task=view_category&Itemid=137&subcat=2&catid=63&limitstart=0&limit=5.

Wynekoop, J. L. and Russo, N. L. (1997) *Studying system development methodologies: an examination of research methods,* Information Systems Journal, 7 pp. 47-65.