Americas Conference on Information Systems (AMCIS)

AMCIS 2009 Proceedings

Association for Information Systems *Year* 2009

Applied Service Engineering for Single Services and Corresponding Service Landscapes

Stefan Schulte[*] Kay Kadner[†]
Nicolas Repp[‡] Ralf Steinmetz[**]

[*]TU Darmstadt, schulte@kom.tu-darmstadt.de
[†]SAP Research, kay.kadner@sap.com
[‡]TU Darmstadt, nrepp@kom.tu-darmstadt.de
[**]Department of Electronic Engineering and Information Technology, Technische Universität Darmstadt, Germany, steinmetz@kom.tu-darmstadt.de

This paper is posted at AIS Electronic Library (AISeL).
http://aisel.aisnet.org/amcis2009/472

Applied Service Engineering for Single Services and Corresponding Service Landscapes

Stefan Schulte
Multimedia Communications Lab, TU Darmstadt
schulte@kom.tu-darmstadt.de

Kay Kadner
SAP Research, Dresden
kay.kadner@sap.com

Nicolas Repp
Multimedia Communications Lab, TU Darmstadt
nrepp@kom.tu-darmstadt.de

Ralf Steinmetz
Multimedia Communications Lab, TU Darmstadt
rst@kom.tu-darmstadt.de

ABSTRACT

Building reusable and autonomous services that possess the proper degree of granularity is critical for the success of a service-oriented architecture (SOA), especially for large and heterogeneous application landscapes. While there are a number of approaches to service engineering, most of these approaches are designed with a special purpose or project context in mind. This paper presents a pragmatic approach to service engineering that can be applied in scenarios where it is particularly necessary to identify service interfaces with the right granularity. The approach is based on a comparison of service engineering methodologies with special regard for their ability to connect different levels of an SOA. We apply concepts from this approach to build single services and their service landscapes. We also suggest the utilization of service inventory techniques to evaluate and assess the outcome of the proposed service engineering methodology.

Keywords

Service engineering, Service-oriented architectures, Service landscapes

INTRODUCTION

Service-orientation is based on well-known concepts such as autonomy and the loose coupling of software components. However, the service-oriented architecture (SOA) paradigm has gained momentum in recent years primarily due to the standardization of Web service technologies. SOA has evolved into one of the "hot" topics in both the software industry and the research community (Bichler and Lin, 2000; Papazoglou, Traverso, Dustdar and Leymann, 2007).

One of the major requirements regarding application landscapes based on the SOA paradigm is the ability to build reusable and autonomous services which possess the correct degree of granularity. This challenge must be met in order to achieve desirable outcomes (i.e., reusability of software components and a clearly arranged application landscape) in the adaption of service-oriented technologies (van den Heuvel, Zimmermann, Leymann, Lago, Schieferdecker, Zdun and Avgeroiu, 2009).

The approaches and techniques used to identify and design appropriate services are often labeled as "Service Design" (Erl, 2007) or "(Software) Service Engineering" (Margaria and Steffen, 2006, van den Heuvel et al., 2009). Many service engineering methodologies are created in response to specific problems, and are therefore only applicable to specific goals, purposes, and technologies. On the other hand, some approaches try to be so universally applicable that they lead to a rather vague course of action which requires extensive adaptation to address the problem at hand.

In this paper, we propose an approach for *applied* service engineering for single services and corresponding service landscapes, i.e., the collectivity of all services in a specified domain. Especially if software development is distributed among different teams, it is not always possible to identify the functionalities offered by components in the whole IT infrastructure. Here, service engineers need an easy to use approach to identify functionalities which could be reused in the own applications respectively service compositions. We want to extend the manifold research done in the field of identifying services on the business level by linking the results of this identification process to the actual implementation of technical services. The aim is to close the gap between the actual identification of single services (and hence, the service landscape) and the implementation of services. A very important success factor if applying a quite new software paradigm as service-oriented computing still is, is that the different parties know which higher steps are needed in the development process and what has to be done in each step. Hence, we center on the desired outcome of each step and what needs to be done in order to accomplish it.

The remaining part of this paper is structured as follows. In the next section, we analyze different service engineering approaches with respect to their ability to define single services as well as a service landscape. Then, we introduce an application of service engineering methodologies. To evaluate the outcomes of our approach in service engineering, we propose the use of service inventory taking. Finally, this paper concludes with a summary of our findings and an outlook on our future work.

SERVICE ENGINEERING METHODOLOGIES

The scope of service engineering approaches ranges from the derivation of service functionalities from existing business processes, to the technical implementation of services. To demonstrate this range, we present different service engineering methodologies, compare them, and identify common characteristics of service engineering methodologies. Due to the extensive variety of service engineering methodologies, it is not possible to consider every approach. Thus, the following sample of methodologies has been chosen to represent those approaches who primarily affect our own work.

In 2005, *SeCSE* (Service Centric System Engineering[1]) defined service engineering as "the activity of specifying, designing, implementing, and maintaining services offered by a service provider" (Sawyer, 2005). SeCSE is a long-running EU-sponsored project, which tackles service and system engineering (amongst other topics). Their main focus is the definition of a conceptual model for service engineering, including service description, service discovery, and service composition. The former regards the application of semantic information in order to describe service functionalities and includes Quality of Service aspects in the service description. However, the SeCSE-project focuses on the examination of different aspects of service engineering instead of the deployment of an overall approach to service engineering.

Margaria and Steffen (2006) proposed the application of principles from the field of networks and telecommunication for service engineering. The authors focus on the alignment of the business and IT levels based on services which link the two. However, this happens on a quite coarse level of granularity. By adapting a "divide and conquer" based approach, a taxonomy of services is constructed. Thus, the authors use a top-down approach for service definition. The model is enhanced by formal verification of the entire design process. One particular strength of this approach is the continuous support offered by tools.

Several companies (e.g., BEA, IBM, Oracle, SAP) were involved in the development of the *Service Component Architecture* (SCA), a model used to implement and connect software components based on SOA principles (Chappell, 2007). SCA is independent of any corporation, i.e., there are no restrictions regarding specific tools when adapting the proposed model. SCA is only a model and does not include an implementation. SCA offers both a methodology to develop software components, and a mechanism which addresses the interaction between different components. The definition of components ranges from simple Java classes to complex software artifacts.

A comprehensive approach to the engineering of semantic Web services has been carried out by Bayer, Eisenbarth, Lehner and Petersen (2008). Here, the authors provide a service engineering process that allows for the determination of service landscapes as well as single services. Following principles from requirements engineering, the authors provide a top-down-approach that makes use of information regarding the whole application landscape. Then, the requirements of single services are identified, and services are designed and implemented. The process ends with the testing and registration of the services. One particular strength of the proposed approach is the holistic process that covers all necessary stages of service engineering. However, this methodology works best if information about the service landscape is centrally available. It would be difficult to scope the service landscape of a more distributed approach where a central entity does not exist. Furthermore, testing is only carried out on the implementation level, i.e., a test for the whole service landscape is missing. Nevertheless, this approach comes closest to our requirements as will be presented in the following sections.

There are a number of other approaches, for example by Specht et al., that focus on the technical deployment of services with special attention to the reusability of services (Specht, Spath and Weisbecker, 2005). An extensive model for the deduction of business services, and the modeling of technical services and their implementation is provided by Erl (2005).

The incorporation of semantic information in service engineering has also been observed by some researchers. Tetlow et al. (2006) presents some potential uses of Semantic Web technologies in systems and software engineering. Even though they consider the use of semantic information to enhance the retrieval of Web services, it does not contribute directly to service engineering itself. Hepp and Roman present an ontology framework for semantic business process management which should

[1] European Commission, Information Society and Media Directorate-General, Network and Communication Technologies, Software Technologies. www.secse-project.eu

be incorporated into the work at hand, as it provides possible ontologies to represent business processes (Hepp and Roman, 2007). Once again, the authors have not taken the service engineering process itself into account.

Comparison and Common Characteristics of Service Engineering Approaches

○ – Integrated ● – Partially integrated × – Not integrated	SeSCE	(Margaria and Steffen, 2006)	SCA	(Bayer et al., 2008)
Top-down approach	×	○	×	○
Bottom-up approach	○	×	○	×
Definition of single services	○	○	○	○
Verification/Testing of single services	○	○	×	○
Definition of service landscape	●	×	○	○
Verification/Testing of service landscape	×	○	×	×
Semantic description of services	○	×	×	○
Usage of ontologies	●	×	×	○

Table 1. Comparison of Selected Service Engineering Technologies

Table 1 shows a comparison of the approaches mentioned, and identifies characteristic service engineering elements in each of them.

Service engineering approaches can be divided into top-down or bottom-up approaches. This characteristic normally depends on the scope of the approach. While methodologies based on the business process model often provide a top-down approach, methodologies which primarily address single services logically follow a bottom-up approach. Thus, single services are often not related to the actual business processes (or similar constructs) they are used in and vice versa. This makes the retrieval and usage of services difficult, especially in large, heterogeneous service landscapes.

Another distinguishing feature of a methodology is the definition and verification of either single services or service landscapes. Naturally, the definition of single services is part of almost every service engineering methodology. However, a single service can range from concrete instructions on the implementation of services, to the non-technical identification of services. Furthermore, service landscapes should also be defined using the same service engineering approach used for the identification of single services.

Although the SeSCE project considers the semantic description of services with respect to service retrieval, none of the above-mentioned methodologies has applied this information in order to enhance the service engineering process. In Bayer et al. (2008), semantic information is used to describe pre- and post-conditions of service invocations – however, this approach does not allow the linkage between the service landscape and single services by semantic information.

APPLIED SERVICE ENGINEERING FOR SINGLE SERVICES AND CORRESPONDING SERVICE LANDSCAPES

Even though the abovementioned service engineering approaches provide important means to design and implement services, the realization of services and service landscapes is often constrained by the project context. If a large number of project partners is involved and a project is following a bottom-up approach, it is particularly necessary to follow a pragmatic approach to service engineering and incorporate most of the methods presented in Table 1 at the same time. In the following, we present the simple and pragmatic approach to service engineering which was proposed for the project SoKNOS (www.soknos.de) but can be easily applied to other project contexts as well. Before we present the four steps of this process, we introduce the different service levels we consider in our work.

Service Levels

In our opinion, service engineering should be regarded at four different levels. This classification has been derived from the common understanding of SOA in scientific literature, (e.g., Krafzig, Banke and Slama, 2004; Papazoglou et al., 2007). On the one hand, it is necessary to deploy a technological infrastructure in order to implement elementary functionalities such as communication between services. On the other hand, this infrastructure is predetermined by external conditions and thus technical services have to be implemented according to these circumstances. Last but not least, services are a counterpart for an entity in a real-world system.

Technical services are services on the infrastructure level. They possess a relatively fine granularity and offer functionalities which vary from specific business actions to data services (e.g., connection to a database). While the reusability of this kind of service is very high, technical services are bound to specific IT systems which make them unlikely to be invoked by people unfamiliar with them. Furthermore, the number of technical services can grow very fast in large heterogeneous application landscapes, which further decreases the clarity of the service landscape.

Composite services combine functionalities from two or more technical services. They may possess direct business value and can be directly related to a task or subprocess in a business process; hence, it is necessary that these services are easily found. The reusability of composite services depends on how specific their outcomes are – generally speaking, less specific outcomes are per se more likely to be reusable in different contexts. However, this does not apply to every context and has to be considered when assessing the value of a service.

Business processes can be composed of technical or composite services, and can be very complex. While a business process is usually not a service on its own, it might be reasonable to provide a service interface for a business process, e.g., in order to make a process available to other departments within a company. Business processes possess direct business value and are the most complex of the services mentioned so far.

Public services are all services that are made available to parties outside of the company. They may be technical services, composite services or even complete business processes. While the three aforementioned types of services (including business processes) do not overlap, public services are combinations of other sorts of services, meeting the functional requirements regarding accounting, security, as well as non-functional requirements like availability and other Quality of Service attributes.

The different kinds of services show the need to interrelate the different levels of services in order to accomplish a clearly arranged service landscape. In the following, we proceed on the assumption that business processes such as workflows are available or can be constituted based on existing information. After that, composite services and service interfaces (i.e., single services) are constructed, and finally enriched with semantic information in order to link the different service levels.

Application of a Domain Model

Instead of "structuring" a heterogeneous application landscape based on monolithic systems, the SOA paradigm implies the composition of an application landscape by services. These services possess clearly defined standardized interfaces which make the communication and use of single services very easy. In this way, the complexity of interfaces is radically reduced. If interfaces are not standardized, a component in the application landscape has to be adapted to each interface in order to use the functionalities offered. In the worst case, each component has to adapt n (number of functionalities in the domain) different interface standards, leading to $n \times k$ (k = number of components in the domain) different adaptations. If applying service-oriented concepts and standards, the number of adaptations is reduced to k (one adaptation per component) if all components use the same service interface standard.

However, in order to achieve the possible advantages of applying service-oriented concepts, it is necessary to base the application landscape (i.e., the service landscape) upon a commonly accepted foundation. If a business process model is available, this could be used as a foundation. Another possibility is the utilization of an existing domain ontology. Despite the possible benefits of industry-specific domain ontologies, a shortage of such ontologies exists. Thus, it is necessary to derive a domain model from an existing business process model or similar sources.

Derivation of Composite Services from a Domain or Process Model

The derivation of composite or business-oriented services from a process model has been regarded in various publications, (e.g., Erl, 2005; Krafzig et al., 2004; Bell, 2008). Hence, we will restrict this section to two major aspects of this process. First, business-oriented services are usually used to map exactly one (major) aspect of the business (Krafzig et al., 2004) and vice versa, making it possible to identify necessary services based on the processes available. Secondly, a service repository needs to be established in order to identify appropriate services for potential consumers (Bell, 2008). Of course, it is necessary to limit the visibility of certain services, as not every service is intended for all users such as external customers.

If a process model is not available, services have to be defined based on other data sources, one method being to list which workflows make use of each software component (Alonso, Casati, Kuno and Machiraju, 2004). While these methods may produce good results, the lack of a process model yields to the loss of an important piece of (semantic) information which is essential in order to understand and model a service landscape.

As a result of the derivation of composite services, an application designer should possess a number of composite services, workflows, or business processes which have been split into singular steps. Now, it is possible to identify the service interfaces required to deliver the functionalities of each process step.

Identification of Service Interfaces

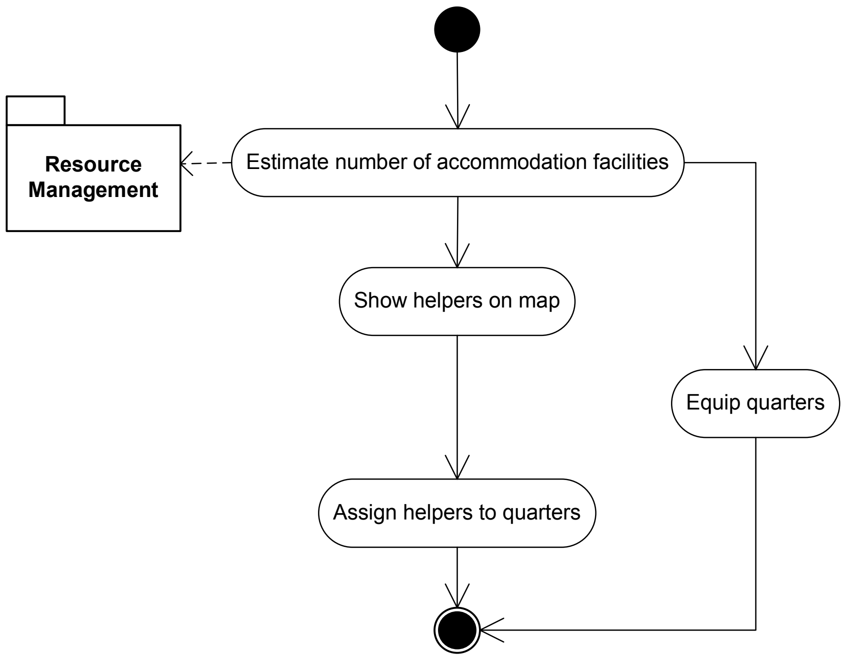
Resource Management
Estimate number of accommodation facilities
Show helpers on map
Equip quarters
Assign helpers to quarters

Figure 1. Placement of Helpers

With regard to the construction of service landscapes, the internal implementation of services is rather irrelevant as long as it does not affect the ability to provide services with reasonable granularity. Far more important is the identification of functionalities and data that a service needs to provide to other components within an application landscape. Thus, it is first and foremost necessary to identify the dependencies which exist between different components in a landscape.

Our approach for the identification of service interfaces can be applied both bottom-up as well as top-down with regards to the underlying business process model. For an existing application landscape without a corresponding business process model (i.e., the model and the applications have not been mapped to each other), it is necessary to identify interactions in which the applications were used so far. This step is redundant if an explicitly defined business process model exists. In this case, necessary services and their granularity can be identified based on the model.

For either the bottom-up or top-down approach, we use activity diagrams to identify necessary services and the entity in the application landscape which must provide them. It is possible to adapt and apply similar notations, especially if a business process model has already been defined in an appropriate notation. The use of activity diagrams is only recommended in relatively small scenarios. For large application landscapes, the identification of service interfaces needs to be supported by more sophisticated tools.

If services are identified bottom-up, it is necessary to distinguish all the functionalities of an application that this software component needs to provide to other software components. Our methodology is based on the principle of path testing (Howden, 1976), but instead of using control flow graphs, we use activity diagrams. While control flow graphs usually feature a fine granularity, activity diagrams match the comparatively coarse granularity of business processes and workflows much better. Paths are identified by determining the interactions where the applications have been used so far, or will be used in the future.

The principle of path testing implies that instead of constructing all possible paths and activity diagrams, only independent paths have to be considered. When the independent paths are combined, they represent all the functionalities an application must provide to be an equivalent system to what it was before service-oriented principles were applied. In the following, we will show the identification of service interfaces based on a simple example which is taken from the project SoKNOS. This research project aims to develop concepts that are valuable in the support of governmental agencies, private companies, and other organizations active in managing disastrous events in the public security sector. Hence, there are many components involved in the provisioning of particular processes. In a simplified illustration, we use the *Resource management* component and the workflow *Placement of helpers*.

As seen in Figure 1, this workflow contains several different activities. The activity "Estimate number of accommodation facilities" has already been connected to the resource management component. Next, it is necessary to identify and define dependencies on other components of the application landscape. Attention should be paid to the fact that "dependencies" here are not regarded in terms of software engineering as the components are loosely coupled. If an activity is attached to more than one component, or only parts of necessary functionalities are covered by a component, it is necessary to divide the activity. In the example (Fig. 2), the activity "Equip quarters" is assigned to two different components. Hence, it is necessary to split this activity.

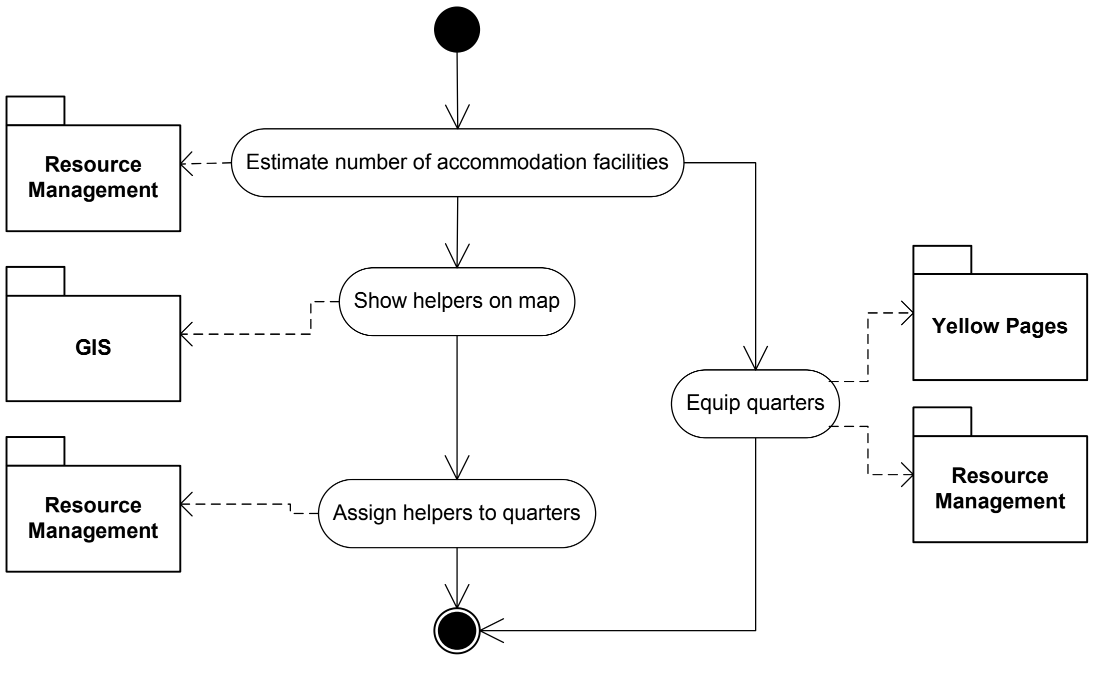

Figure 2. Annotation with Supporting Components

The result of the annotation with supporting components is that a number of interfaces are needed to model a business process. Based on this information, it is possible to develop services with the right degree of granularity, i.e., the functionalities desired have to be written down in a standardized form and need to incorporate further requirements, e.g., security standards. Service interfaces should be enriched with semantic information in order to ease their retrieval and integrate single services into the service landscape, which represents the next step in our service engineering approach.

Enrichment of Service Interfaces with Semantic Information

While different authors have proposed the use of automatic annotation mechanisms in order to enrich service descriptions with semantic information (e.g., Patil, Oundhakar, Sheth and Verma, 2004), we suggest that above all the data from the process model and domain ontology (if at hand) be used to annotate services. Thus, it is possible to interrelate services on different levels with each other. However, if there is no link between services on the technical and business process level, for example, it is not possible to reuse semantic information from one level on another level. The actual integration of services on different levels using semantic information will be a subject of our future work. Currently, the plan calls for services (e.g., technical and composite services, and business processes) to be made available through a service repository. Through references to semantic data (i.e., an ontology) in the description of service interfaces and corresponding entities, it is possible to identify which services are related to each other, are part of a composite service, or constitute a business process.

The result of this step is a detailed draft of service interfaces and the service landscape. Now, the services need to be implemented with consideration to predetermined technologies.

EVALUATION APPROACH: SERVICE INVENTORY

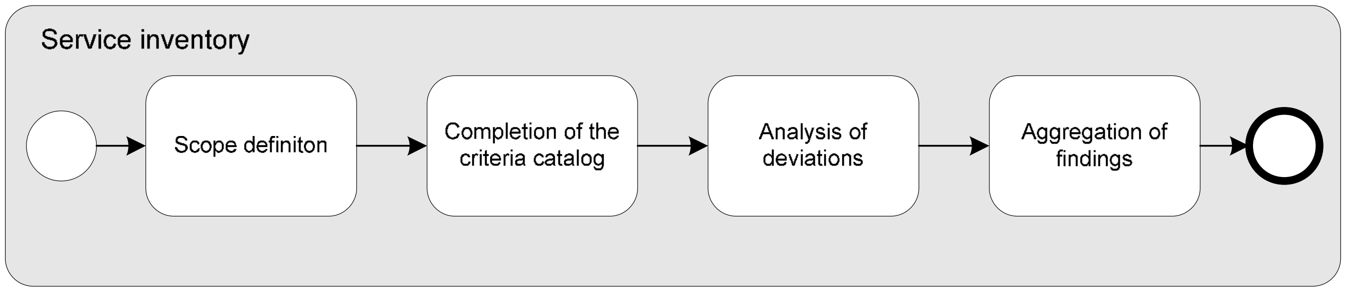

Figure 3. Overview of the Service Inventory Process

In our previous work (Repp, Schulte, Eckert, Berbner and Steinmetz, 2007)[2], we presented the concept of service inventory taking being used to evaluate existing enterprise service ecosystems and their services. As our service inventory process can be used to enhance SOA and service development methodologies, it is possible to apply this process to the service engineering approach as well.

It is important to distinguish between two facets of the "service inventory" concept. On the one hand, this term describes the actual service inventory, which is the result of a service inventory taking. This could be, for example, a service repository enriched with information about the value of the services. On the other hand, the stocktaking of existing services, their analysis, and evaluation is also called "service inventory" (see Fig. 3). Contrary to the inventory process in financial accounting, no monetary value is assigned to a service during service inventory. Instead, the value of a service is measured based on its ability to be integrated into a service landscape.

The actual assessment of single services has to incorporate the type of the service as mentioned above and is achieved by a questionnaire (Repp et al., 2007). The questionnaire or criteria catalog is based on the definition of a scope which could be, e.g., to estimate which services could be offered to external partners. After the criteria catalog is completed, deviations from the required values need to be identified and further analyzed. If necessary, counteractions need to be taken in order to fix identified flaws.

Depending on the scope of the service inventory process, there are different service aspects that should be considered. These aspects have been derived from different sources, such as the common principles of service-orientation presented by Erl (2005), the rules for architectural design of enterprise IT by Engels, Hess, Humm, Juwig, Lohmann, Richter, Voß, and Willkomm, (2008), the standardized specification of business components by Ackermann et al. (2002), and our experience in various SOA projects.

Using these sources as guides, we evaluate services based on the following service aspects (Repp et al., 2007):

---

[2] In order to make this paper self-contained, we include a summary of some materials already presented in (Repp et al., 2007) in here.

- Reusability
- Granularity
- Autonomy
- Degree of coupling
- Information hiding
- Discoverability
- Context independence

The results of the service inventory process show which services do not possess the needed degree of granularity, are not detectable, or are not reusable in a different context. Thus, the outcome of a service inventory process can be used for the evaluation and design of a service landscape, as it evaluates single services and their importance in an overall service landscape.

CONCLUSION

In this paper, we have presented a conceptual model for service engineering which has been applied in a research project. Through the analysis of different service engineering methodologies it was possible to identify characteristic elements of service engineering. Compared with Table 1, the applied service engineering process involves all aspects mentioned. When necessary, we incorporated semantic information in the service engineering process (e.g., to arrange the service landscape). In addition, we applied service inventory practices in order to evaluate the service landscape and assess the outcome of the proposed service engineering methodology.

Even though the proposed approach to service engineering has been applied in the context of SoKNOS, the research presented here has been largely carried out on a conceptual level. We are currently working on an implementation of our approach to service engineering which will be presented in a publication in the near future. This implementation specifically addresses the support provided by tools concerning the identification of service interfaces and the linking of services at different service levels.

ACKNOWLEDGEMENTS

This work is supported in part by the BMBF[3]-sponsored project SoKNOS (www.soknos.de) and the E-Finance Lab e. V. (www.efinancelab.de).

REFERENCES

1. Ackermann, J., Brinkop, F., Conrad, S., Fettke, P., Frick, A., Glistau, E., Jaekel, H., Kotlar, O., Loos, P., Mrech, H., Ortner, E., Raape, U., Overhage, S., Sahm, S., Schmieten, A., Teschke, T. and Turowski, K. (2002) Standardized specification of business components. *Memorandum of the working group 5.10.3*, Component Oriented Business Application Systems.
2. Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004) Web Services – Concepts, Architectures and Applications, Springer-Verlag, Berlin, Heidelberg, New York.
3. Bayer, J., Eisenbarth, M., Lehner, T. and Petersen, K. (2008) Service Engineering Methodology, in Dominik Kuropka, Peter Tröger, Steffen Staab and Mathias Weske (Eds.) *Semantic Service Provisioning*, Springer, Berlin/Heidelberg, 185-201.
4. Bell, M. (2008) Service-Oriented Modeling - Service Analysis, Design, and Architecture. John Wiley & Sons, Inc., Hoboken, NJ, USA.
5. Bichler, M. and Lin, K.J. (2006) Service-oriented computing, *IEEE Computer Magazine*, 39, 3, 99-101.

[3] German Federal Ministry of Education and Research

6. Chappell, D. (2007) Introducing SCA. Whitepaper, http://www.davidchappell.com/articles/Introducing_SCA.pdf, accessed at 2008-09-18.
7. Erl, T. (2005) Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA.
8. Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J.-P., Voß, M. and Willkomm, J. (2008) A Method for Engineering a true Service-Oriented Architecture, in José Cordeiro, Joaquim Filipe (Eds.): Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS 2008), Volume 2 Information Systems Analysis and Specification, Barcelona, Spain, 272-281.
9. Erl, T. (2007) SOA – Principles of Service Design. Prentice Hall PTR, Upper Saddle River, NJ, USA.
10. Hepp, M. and Roman, D. (2007) An ontology framework for semantic business process management, in Andreas Oberweis, Christof Weinhardt, Henner Gimpel, Agnes Koschmider, Victor Pankratius and Björn Schnizler (Eds.) *eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik (WI 2007),* February 28 – March 2, Karlsruhe, Germany, Universitätsverlag Karlsruhe, 423-440.
11. Howden, W.E. (1976) Reliability of the path analysis testing strategy, *IEEE Transactions on Software Engineering*, 2, 3, 208-215.
12. Krafzig, D., Banke, K. and Slama, D. (2004) Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series). Prentice Hall PTR, Upper Saddle River, NJ, USA.
13. Margaria, T. and Steffen, B. (2006) Service engineering: Linking business and IT, *IEEE Computer Magazine*, 39, 10, 45-55.
14. Papazoglou, M.P., Traverso, P., Dustdar, S. and Leymann, F. (2007) Service-oriented computing: State of the art and research challenges, *IEEE Computer Magazine*, 40, 11, 38-45.
15. Patil, A., Oundhakar, S., Sheth, A. and Verma, K. (2004) METEOR-S web service annotation framework, in Stuart I. Feldman, Mike Uretsky, Marc Najork and Craig E. Wills (Eds.) *Proceedings of the 13th International Conference on World Wide Web. (WWW2004)*, May 17-20, New York, NY, USA, ACM Press, 553-562.
16. Repp, N., Schulte, S., Eckert, J., Berbner, R. and Steinmetz, R. (2007) An approach to the analysis and evaluation of an enterprise service ecosystem, in *Proceedings of the ICSOFT'07 Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC)*, Barcelona, Spain, INSTICC Press, 42-51.
17. Sawyer, P. (2005) Service specification state of the art, WP1.1. Report, *Service Centric System Engineering (SeCSE).*
18. Specht, T., Spath, D. and Weisbecker, A. (2005) Framework-based IT service engineering, in Peter Kokol (Ed.) *IASTED International Conference on Software Engineering, part of the 23rd Multi-Conference on Applied Informatics*, February 15 – 17, Innsbruck, Austria, IASTED/ACTA Press, 202-207.
19. Tetlow, P., Pan, J.Z., Oberle, D., Wallace, E., Uschold, M. and Kendall, E. (2006) Ontology driven architectures and potential uses of the semantic web in systems and software engineering. *W3C Working Draft Working Group Note 2006/02/11*.
20. van den Heuvel, W.-J., Zimmermann, O., Leymann, F., Lago, P., Schieferdecker, I., Zdun, U. and Avgeriou, P. (2009) Software Service Engineering: Tenets and Challenges, in Proceedings of the ICSE 2009 Workshop "Principles of Engineering Service Oriented Systems (PESOS), IEEE Computer Society Los Alamitos, CA, USA.