Case Study for a Quality-Oriented Service Design Process

Michael Gebhart, Suad Sejdovic, Sebastian Abeck
Research Group Cooperation & Management
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{gebhart | sejdovic | abeck} @kit.edu

***Abstract*—Due to the usage of distributed information, such as sensor information, geographical information systems are designed according to service-oriented principles. Thus, the development of new solutions within this context requires a design of necessary services. These services have to follow certain quality attributes that have evolved as important for services, such as loose coupling and autonomy. In this paper, a quality-oriented design process is considered and its applicability and effectiveness are shown within the Personalized Environmental Service Configuration and Delivery Orchestration project of the European Commission.**

Keywords-service; design; quality; geographical information system; case study

I. INTRODUCTION

Today, geographical information systems use distributed information, such as sensor information, that measures environmental data, such as air temperature, or presume precipitation. This information is provided by public authorities or private sectors in form of services [8]. The geographical information system acts as service consumer, thus sends requests to the services and receives according responses. Additionally, functionality of the geographical information system can also be provided in form of services in order to enable the realization of systems at a higher level.

Accordingly, the development of such geographical information systems requires a design of necessary services in order to support the usage of distributed information and the provision of functionality that bases on this information. The design of services consists of two elementary phases, the identification and the specification [1, 2, 9, 10, 11, 25]. During the identification phase service candidates as proposals for services and their dependencies are formalized [5, 6]. Each service candidate includes a set of operation candidates that represent preliminary operations. A dependency between service candidates describes that a service requires another service for fulfilling its functionality. Within the specification phase, the final specifications of the services are created. Each specification constitutes a so-called service design and consists of a specification of the service interface and the realizing service component. The service interface describes provided and required operations, message and data types, interacting roles and the interaction protocol [7]. The specification of the service component determines the services provided by the realizing component and the services required for fulfilling the provided functionality. Additionally, the internal behavior in form of a composition of own functionality and functionality provided by other services is formalized.

For services several quality attributes have been identified that should be fulfilled in order to attain goals that are associated with the application of service-orientation, such as an increased flexibility [5, 6, 12, 14, 15, 20, 30], reusability [5, 21], or maintainability [19] of provided functionality. Wide-spread quality attributes that support these goals are a unique categorization, loose coupling, discoverability, and autonomy [2, 6, 13, 16, 17, 18, 19]. Since these goals are also important for geographical information systems, the quality attributes should be considered when designing new services in the context of a geographical information system. This requires a quality-oriented service design process when developing a service-oriented geographical information system.

In the context of the project Personalized Environmental Service Configuration and Delivery Orchestration (PESCaDO) [3, 4] of the European Commission, a service-oriented geographical information system has to be developed in cooperation with the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation. This system enables getting personalized information regarding the personal profile and environmental conditions. Since the services should fulfill quality attributes, such as loose coupling, a quality-oriented service design process has to be applied. For this purpose, the design process created by the authors of this paper as introduced in [1] has been applied. This design process includes a transfer of artifacts of the business analysis phase into artifacts of the design phase and considers a certain set of quality attributes. In this case, the quality attributes of a unique categorization, loose coupling, discoverability and autonomy are regarded using the quality indicators as introduced in [2]. This case study shows how to apply the design process for a geographical information system of a real world project and demonstrates the applicability and effectiveness of the design process.

The paper is structured as follows: Section 2 introduces the PESCaDO project and the considered service design process. In Section 3, the design process is performed in order to design the necessary services for PESCaDO. In this context, the artifacts of the design phase are systematically derived and revised subsequently. Section 4 concludes the paper and offers suggestions for future research.

II. FUNDAMENTALS

In the following, the PESCaDO project and the considered scenario of this project are introduced. Additionally, the quality-oriented service design process that is applied for designing the required services is described.

A. *Personalized Environmental Service Configuration and Delivery Orchestration*

Nowadays, more and more people are aware of the influence that environmental conditions can have on the quality of their life. Since each individual has the need for specific information about the environment that is affecting him and his life, information personalization plays a major role.

The PESCaDO project of the European Commission [3, 4] takes up this issue and aims at developing a platform for getting personalized information regarding the personal profile, such as health status, mode of presentation or language of an individual, and also takes into consideration the intention of the individual. PESCaDO covers the discovery of services providing the data, their orchestration, the processing of the data and the delivery of the gained information. In terms of reusability, technology independence and the flexible usage of existing functionalities, a service-oriented approach should be pursued [5, 6, 12, 14, 15]. The resulting services are expected to consider the quality attributes of a unique categorization, loose coupling, discoverability, and autonomy. These attributes are chosen, because they can be evaluated during design time [1, 2]. Quality attributes, such as statelessness, require implementation information.

Within a first prototype, the data access functionality has to be developed. One special requirement is the semantic support for accessing environmental data. Thus, the system has to be capable to identify all related data sources for a requested phenomenon like temperature. For this purpose, it has to be able to extend a single requested phenomenon by other related ones. For example, if the system has identified the phenomenon “Pollen” as relevant, it also will have to retrieve information about more specific phenomena, like “Birch Pollen”. For achieving this goal, the system uses a knowledge base, which contains a related ontology. The focus in the development of the first prototype lies on the extension of the requested phenomenon and accessing the related data in the background.

B. *Quality-Oriented Service Design Process*

The quality-oriented service design process, which is illustrated in Figure 1, starts with the business analysis phase that yields artifacts that constitute the input for the service design phase. The primary goal of this phase is the identification and modeling of the considered business use cases and the realizing business processes [9, 10]. The artifacts use terms as introduced within the domain model for a common understanding. The business processes can consider already existing services in order to increase the reuse of functionality. This means, that the activities within the business process are aligned with the operations of existing services regarding their granularity and names.

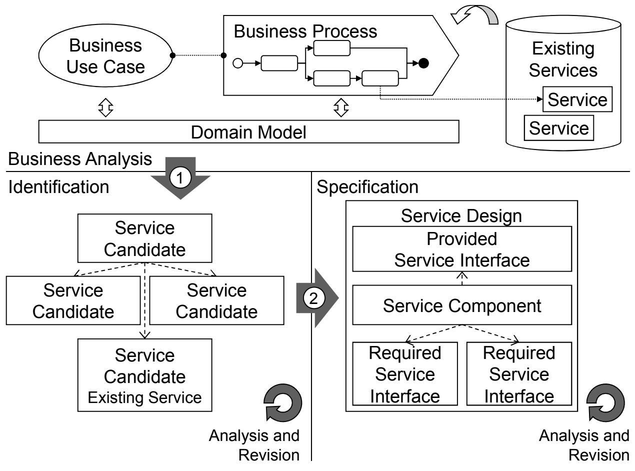

Figure 1. Quality-oriented service design process.

Within the service design phase, two activities have to be performed. In a first step service candidates are systematically identified by using the modeled business processes of the analysis phase. Afterwards, these service candidates are analyzed and revised according to the quality attributes unique categorization, loose coupling, discoverability, and autonomy [1]. In a second step the service specification is performed. The service specification uses the identified and revised service candidates as input and defines service design, i.e., the service interfaces and service components. After a systematic derivation, the service designs are revised with regard to the previously mentioned quality attributes. This additional revision is necessary as service designs include more information than service candidates.

III. CASE STUDY FOR A QUALITY-ORIENTED SERVICE DESIGN PROCESS

Within PESCaDO the business use case for getting an observation has to be considered. The business use case can be modeled using use case diagrams of the Unified Modeling Language (UML) [34]. Furthermore, the UML profile for business modeling as introduced by IBM [22, 23] can be applied with its adapted notation for use case diagrams as shown in the following figure.

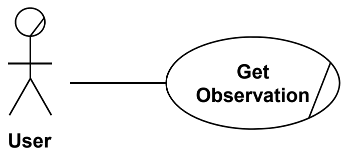

Figure 2. Considered business use case.

For the derivation of service candidates, especially the internal behavior of the business use case is required. This behavior is represented by a business process and can be modeled using the Business Process Model and Notation (BPMN) [31]. Figure 3 shows the business process as main artifact for deriving service candidates as first step of the service design phase.

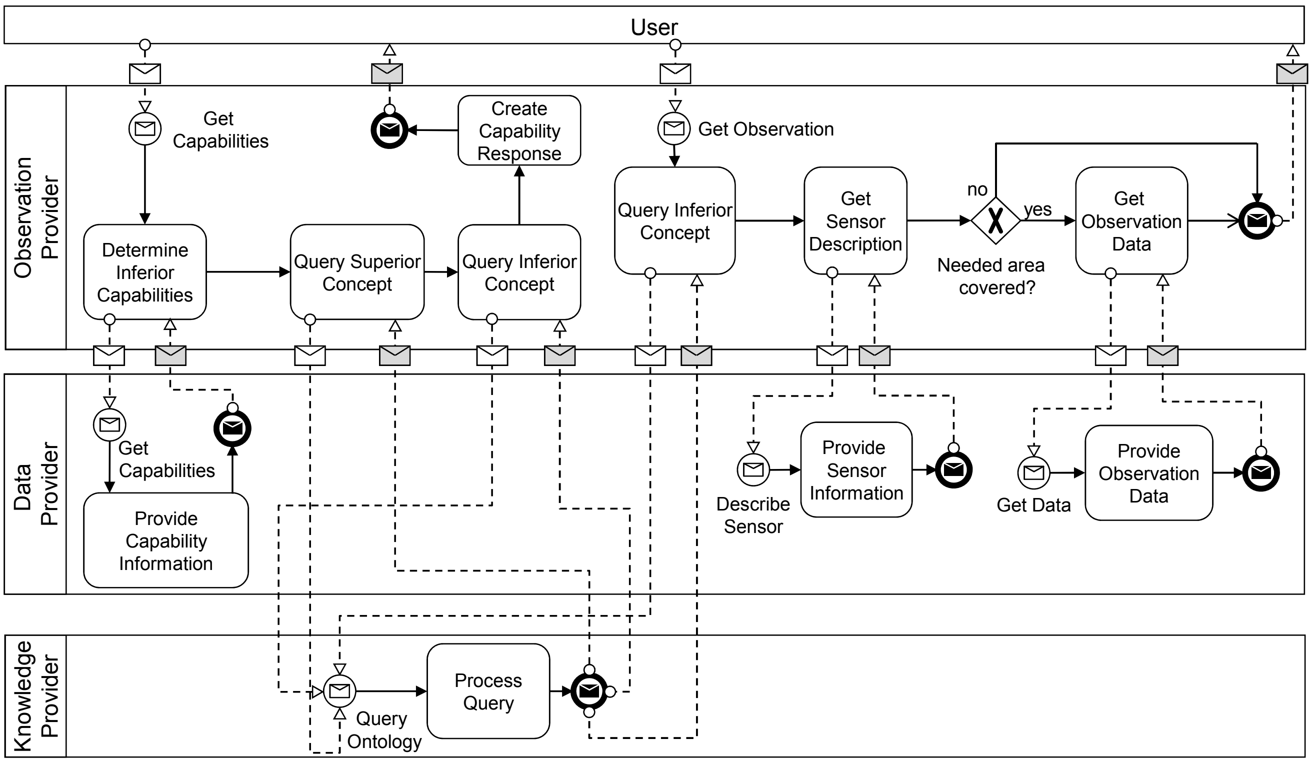

Figure 3. Considered business process.

Each term within the business use case and business process bases on a common domain model for avoiding ambiguity and misunderstandings. This domain model can be described using an ontology based on the OWL 2 Web Ontology Language (OWL) [32, 33]. It determines the concepts and their relations within the considered domain.

A. *Identification*

For the derivation of service candidates each pool within the BPMN business process is transformed into one capability element of the Service oriented architecture modeling language (SoaML), for this element represents a collection of capabilities that corresponds to the understanding of service candidates. Each capability element contains operations that represent operation candidates as preliminary operations of the service [7, 24, 26]. Figure 4 shows the derived service candidates.

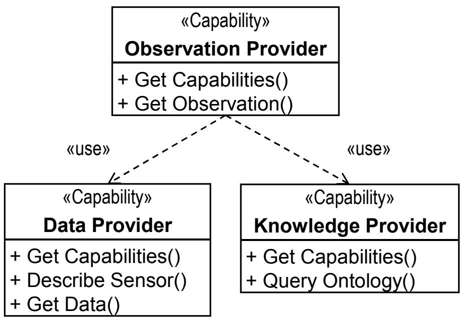

Figure 4. Derived service candidates.

The operation candidates within the service candidates, i.e., capability elements, are derived from the business process and its contained message start events. The usage dependencies are determined by means of the interaction between the pools. The names of the service candidates and operation candidates are taken from the business process.

In a next step, the service candidates have to be analyzed and revised with regard to the quality attribute unique categorization, loose coupling, autonomy and discoverability using the quality indicators introduced in [2].

1) Unique Categorization: According to Erl [5, 6, 28, 29], business-related and technical functionality should be divided. This quality indicator is fulfilled because all services only provide business-related functionality. Similarly, agnostic and non-agnostic functionality should be separated. Also this quality indicator is fulfilled, for all services only provide agnostic functionality, which is not specific for certain business proesses. Another quality indicator for the unique categorization addresses the sovereignity of data. If a service manages a business entity, it should be explicitly managing this business entity for ensuring consistent and clear responsibility [5, 6, 12]. Within the busines process there are two types of data: ontology data and observation data. The former are accessed by the knowledge provider and the latter by the data provider, which is why this quality indicator is fulfilled optimally. The last quality indicator for a unique categorization describes that the operations within one service should use common business entities. The data provider and knowledge provider only operate on ontology data or observation data. However, the observation provider uses both observation data and ontology data, which may result in a split of these two operation candidates into two seperate service candidates. Since the ontology data describes the observation data in more detailed, the ontology data does not represent an own business entity. Thus, the operation candidates can be grouped within one service candidate. As result, the derived service candidates best fulfill the quality indicators for a unique categorization.

2) Loose Coupling: According to Josuttis [15], long-running operations should be able to be invoked asynchonously. Since there are no long-running operations, respectively operations candidates, within the derived service candidates, this quality indicator does not have to be considered. Additionally, the parameters within the operations should be preferably simple types if they are used across several services. Complex types that are used within several services should be avoided. Since during the identification phase the parameters are not defined, this quality indicator can not be determined. Instead, this quality indicator will be considered during the specification phase. A further quality indicator describes that the operations should be abstract [5, 6, 15, 17]. This means that they should hide implementation details. The operation candidates are on a high-level of abstraction, which is why this quality indicator is fulfilled. Also if there is an state-changing operation, a compensating operation should be provided [17]. Since there is no data written or created, there is no state-changing operation.

3) Discoverability: The discoverability is only of interest during the specification phase, when the names of services and operations are finally determined. During the identification phase the artifacts are only preliminarily named.

4) Autonomy: One quality indicator for the autonomy of services focuses on the direct dependencies between services [5], which should be minimal for a maximum autonomy. Within the derived service candidates, the only service candidate with dependencies is the observation provider. However, due to the requirement of using distributed functionality, this quality indicator can not be improved. Another quality indicator addresses the overlapping of functionality [5, 28]. Services should have a certain functional scope. Since the service candidates do not have any overlapping functionality.

As result, the derived service candidates optimally fulfill the quality indicators for the considered quality attributes and thus do not have to be further revised.

B. Specification

The subsequent phase, the specification phase, focuses on the creation of service designs. A service design consists of a service interface, which describes the service from an external point of view, and a service component, which performs the provided functionality [2]. First, the service candidates of the identification phase are used to generate preliminary service designs that can be further revised in order to fulfill the desired quality attributes. Figure 5 shows the derived service interface for the Observation Provider.

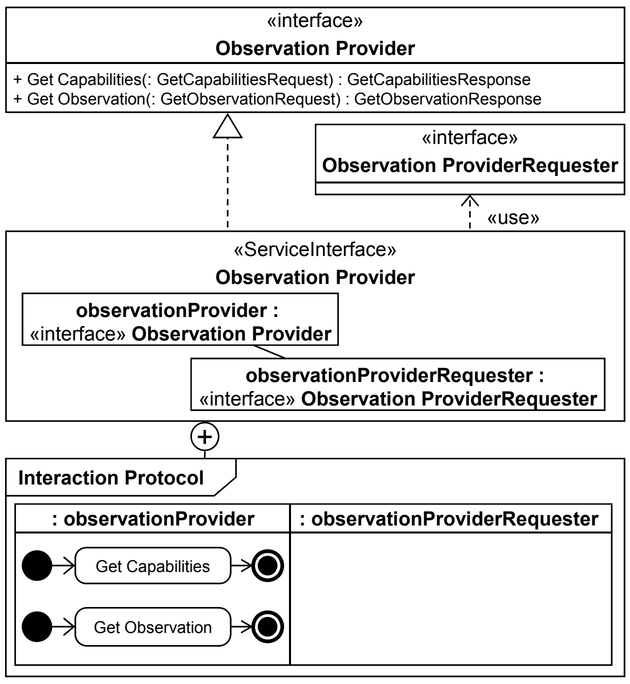
«interface»
Observation Provider
+ Get Capabilities(: GetCapabilitiesRequest) : GetCapabilitiesResponse
+ Get Observation(: GetObservationRequest) : GetObservationResponse
«interface»
Observation ProviderRequester
«use»
«ServiceInterface»
Observation Provider
observationProvider : «interface» Observation Provider
observationProviderRequester : «interface» Observation ProviderRequester
Interaction Protocol
: observationProvider
: observationProviderRequester
Get Capabilities
Get Observation

Figure 5. Derived service interface.

The service interface is formalized using the ServiceInterface modeling element of SoaML [7]. A service interface includes operations provided by the service and operations that have to be provided by the service consumer in order to receive callbacks. In SoaML these aspects are modeled using UML interfaces that are associated with the ServiceInterface element by generalizations and usage dependencies. Additionally, it defines the participating roles and an interaction protocol, which determines the possible orders of operation calls that result in valid results. Latter is modeled using a UML Activity that is added as ownedBehavior. The derivation of a service interface from service candidates transforms the operation candidates into provided operations. Also the name of the service candidate is used for the name of the service interface. Additionally, messages, roles and the interaction protocol are added systematically.

The service component includes provided services, services that are required to fulfill the functionality, and the internal behavior of the component in form of a flow of operation calls. The service component is represented by a Participant in SoaML. A Participant can be an organization, a system or a component within a system. It contains ServicePoints for provided services and RequestPoints for required services. Each ServicePoint and RequestPoint is typed by the describing ServiceInterface element. In Figure 6, the service component for the Observation Provider is shown. The name of the service component is directly derived from the name of the service candidate. The internal behavior is added as ownedBehavior in form of a UML Activity. It will be illustrated in context of the subsequent revision phase.

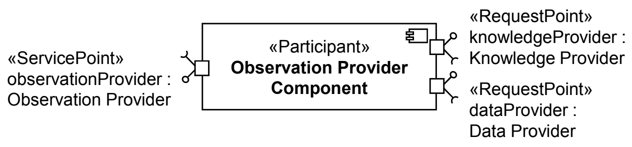

Figure 6. Derived service component.

In a next step, the subsequent analysis and revision phases can be performed considering the quality attributes unique categorization, loose coupling, discoverability, and autonomy.

1) Unique Categorization: Since the quality indicators that influence the unique categorization have already been optimal on basis of service candidates and the service designs were derived from these service candidates, the unique categorization is also optimal on basis of service designs. Thus, there is no revision required.

2) Loose Coupling: In contrast to the identification phase, during the specification phase, the parameters are formalized. For geographical information systems, standard data types, such as the Keyhole markup language (KML) [35], exist. Also within PESCaDO, standardized data types are expected to be used. Since complex types that are used across several services should be avoided, the data types are modeled within single UML packages for each service design. This ensures that changing data types does not necessarily affect other services. The infrastructure, for instance in form of an enterprise service bus, can handle the transformation between similar data types. The other quality indicators are still optimal, for the affecting artifacts have not changed during the specification phase.

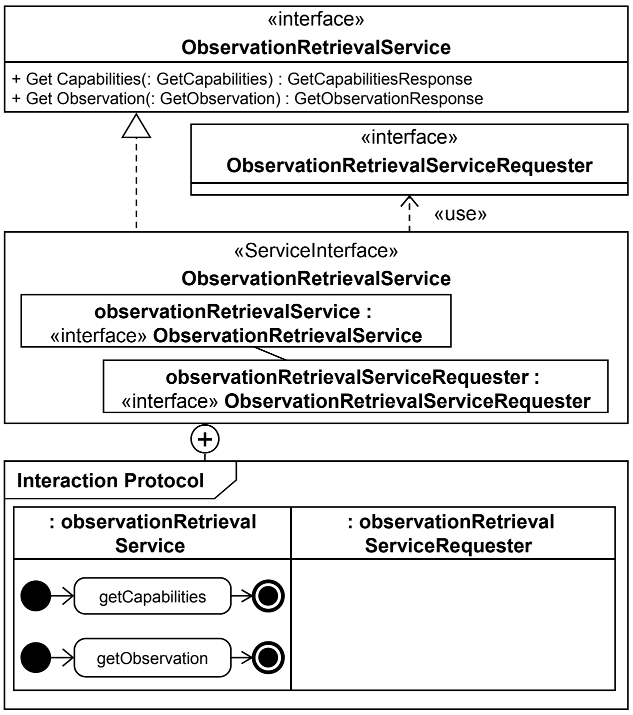
«interface»
ObservationRetrievalService
+ Get Capabilities(: GetCapabilities) : GetCapabilitiesResponse
+ Get Observation(: GetObservation) : GetObservationResponse
«interface»
ObservationRetrievalServiceRequester
«use»
«ServiceInterface»
ObservationRetrievalService
observationRetrievalService :
«interface» ObservationRetrievalService
observationRetrievalServiceRequester :
«interface» ObservationRetrievalServiceRequester
Interaction Protocol
: observationRetrieval Service
: observationRetrieval ServiceRequester
getCapabilities
getObservation

Figure 7. Revised service interface.

3) Discoverability: During the specification phase, the final names of the services and data types are determined. According to Josuttis [15] and Maier et al. [17], the names of the visible artifacts should be functionally named. Additionally, the names should follow naming conventions. Thus, during the specification phase, the names of the artifacts should be inspected in detail. Exemplarily naming conventions are the usage of the english language and beginning operation names with a lower-case character. In Figure 7, a revised service interface is shown that considers the naming conventions of the PESCaDO project. Additionally, the service has been renamed regarding its actual functionality for improving its discoverability.

This revision also affects the service component that uses this service interface. Figure 8 shows the revised service component of the Observation Provider. The service component and the ServicePoints and RequestPoints have been adapted to the revised service interfaces and the naming conventions for PESCaDO. Additionally, the internal behavior of the service component for one of the provided operations is shown.

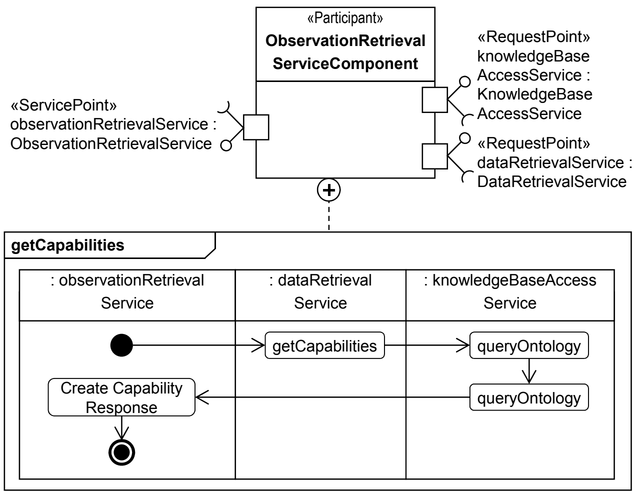

Figure 8. Revised service component.

4) Autonomy: Since the autonomy has already been optimized during the identification phase, there is no revision necessary regarding this quality attribute.

By finishing the revision of the initial service designs, the specification phase ends. The results for developing a prototype for the PESCaDO project are three revised service specifications, which now can serve as an input for the implementation phase [27]. The service designs have been revised that the resulting services optimally fulfill the chosen quality attributes of a unique categorization, loose coupling, discoverability and autonomy.

IV. CONCLUSION AND OUTLOOK

In this paper, we applied a quality-oriented service design process to the Personalized Environmental Service Configuration and Delivery Orchestration project of the European Commission. The design process enabled the systematic derivation and revision in order to gain service designs that fulfill both the functional requirements and the quality attributes of a unique categorization, loose coupling, discoverability and autonomy. The service designs result in services that support the strategic goals that are associated with service-oriented architectures, such as an increased flexibility and maintainability. Due to the application on a concrete scenario, the usage of the design process in terms of its applicability and effectiveness for real-world projects is demonstrated.

The case study also showed shortcomings of the service design process that are expected to be solved in the future: The used quality indicators that were derived from common and wide-spread descriptions of quality attributes use terms that are not exactly defined. For example, the meaning of agnostic functionality is not clear. The IT architect has to interpret these terms in order to determine the quality indicators and the quality attributes. This may result in wrong measures.

Thus, this case study showed the applicability and effectiveness of the service design process. However, in the future, we plan to further refine the definitions of terms used within the quality indicators and quality attributes to reduce ambiguities, thus increase the correctness of the results. Additionally, we plan to apply the design process on further scenarios.

References

[1] M. Gebhart, M. Baumgartner, and S. Abeck, "Supporting service design decisions", Fifth International Conference on Software Engineering Advances (ICSEA 2010), Nice, France, August 2010, pp. 76-81.

[2] M. Gebhart, M. Baumgartner, S. Oehlert, M. Blersch, and S. Abeck, "Evaluation of service designs based on soaml", Fifth International Conference on Software Engineering Advances (ICSEA 2010), Nice, France, August 2010, pp. 7-13.

[3] The PESCaDO Consortium, "Service-based infrastructure for user-oriented environmental information delivery", EnviroInfo, 2010.

[4] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, "D8.3 Specification of the pescado architecture", Version 1.0, 2010.

[5] T. Erl, Service-Oriented Architecture – Concepts, Technology, and Design, Pearson Education, 2006. ISBN 0-13-185858-0.

[6] T. Erl, SOA – Principles of Service Design, Prentice Hall, 2008. ISBN 978-0-13-234482-1.

[7] OMG, "Service oriented architecture modeling language (SoaML) – specification for the uml profile and metamodel for services (UPMS)", Version 1.0 Beta 1, 2009.

[8] The European Parliament and the Council of the European Union, "INSPIRE", Directive 2007/2/EC, 2007.

[9] IBM, "RUP for service-oriented modeling and architecture", IBM Developer Works, http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/, 2006. [accessed: June 04, 2011]

[10] U. Wahli, L. Ackerman, A. Di Bari, G. Hodgkinson, A. Kesterton, L. Olson, and B. Portier, "Building soa solutions using the rational sdp", IBM Redbook, 2007.

[11] A. Arsanjani, "Service-oriented modeling and architecture – how to identify, specify, and realize services for your soa", IBM Developer Works, http://www.ibm.com/developerworks/library/ws-soa-design1, 2004. [accessed: June 04, 2011]

[12] G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J.-P. Richter, M. Voß, and J. Willkomm, Quasar Enteprise, dpunkt.verlag, 2008. ISBN 978-3-89864-506-5.

[13] A. Erradi, S. Anand, and N. Kulkarni, "SOAF: An architectural framework for service definition and realization", 2006.

[14] R. Reussner and W. Hasselbring, Handbuch der Software-Architektur, dpunkt.verlag, 2006. ISBN 978-3898643726.

[15] N. Josuttis, SOA in der Praxis – System-Design für verteilte Geschäftsprozesse, dpunkt.verlag, 2008. ISBN 978-3898644761.

[16] B. Maier, H. Normann, B. Trops, C. Utschig-Utschig, and T. Winterberg, „Die soa-service-kategorienmatrix", SOA-Spezial, Software & Support Verlag, 2009.

[17] B. Maier, H. Normann, B. Trops, C. Utschig-Utschig, and T. Winterberg, „Was macht einen guten public service aus?", SOA-Spezial, Software & Support Verlag, 2009.

[18] M. Perepletchikov, C. Ryan, K. Frampton, and H. Schmidt, "Formalising service-oriented design", Journal of Software, Volume 3, February 2008.

[19] M. Perepletchikov, C. Ryan, K. Frampton, and Z. Tari, "Coupling metrics for predicting maintainability in service-Oriented design", Australian Software Engineering Conference (ASWEC 2007), 2007.

[20] M. Hirzalla, J. Cleland-Huang, and A. Arsanjani, "A metrics suite for evaluating flexibility and complexity in service oriented architecture", ICSOC 2008, 2008.

[21] S. W. Choi and S. D. Kimi, "A quality model for evaluating reusability of services in soa", 10th IEEE Conference on E-Commerce Technology and the Fifth Conference on Enterprise Computing, E-Commerce and E-Services, 2008.

[22] S. Johnston, "Rational uml profile for business modeling", IBM Developer Works, http://www.ibm.com/developerworks/rational/library/5167.html, 2004. [accessed: June 04, 2011]

[23] J. Heumann, "Introduction to business modeling using the unified modeling language (UML)", IBM Developer Works, http://www.ibm.com/developerworks/rational/library/360.html, 2003. [accessed: June 04, 2011]

[24] J. Amsden, "Modeling with soaml, the service-oriented architecture modeling language – part 1 – service identification", IBM Developer Works, http://www.ibm.com/developerworks/rational/library/09/modelingwithsoaml-1/index.html, 2010. [accessed: January 04, 2011]

[25] P. Kroll and P. Kruchten, The Rational Unified Process Made Easy, a Practitioner's Guide to the RUP, Addison-Wesley, 2003.

[26] M. Gebhart and S. Abeck, "Rule-based service modeling", The Fourth International Conference on Software Engineering Advances (ICSEA 2009), Porto, Portugal, September 2009, pp. 271-276.

[27] P. Hoyer, M. Gebhart, I. Pansa, S. Link, A. Dikanski, and S. Abeck, "A model-driven development approach for service-oriented integration scenarios", 2009.

[28] T. Erl, SOA – Design Patterns, Prentice Hall, 2008. ISBN 978-0-13-613516-6.

[29] T. Erl, Web Service Contract Design & Versioning for SOA, Prentice Hall, 2008. ISBN 978-0-13-613517-3.

[30] D. Krafzig, K. Banke, and D. Slama, Enterprise SOA – Service-Oriented Architecture Best Practices, 2005. ISBN 0-13-146575-9.

[31] OMG, "Business process model and notation (BPMN)", Version 2.0 Beta 1, 2009.

[32] W3C, "OWL 2 web ontology language (OWL)", W3C Recommendation, 2009.

[33] M. Horridge, "A practical guide to building owl ontologies using protégé 4 and co-ode tools", http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/, Version 1.2, 2009. [accessed: January 04, 2011]

[34] OMG, "Unified modeling language (UML), superstructure", Version 2.2, 2009.

[35] OGC, "Keyhole markup language (KML)", http://www.opengeospatial.org/standards/kml/, Version 2.2, 2008. [accessed: June 04, 2011]