Supporting Service Design Decisions

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

***Abstract*— In the context of service-oriented architectures, services are expected to fulfill certain service characteristics, such as loose coupling or high autonomy. When designing new services, several decisions have to be made, such as how to group capabilities into services, that influence these characteristics. Existing development processes focus on the description of necessary steps to create services and do not explicitly describe detailed design decisions and their impact on the service characteristics. In this paper, an approach is introduced to determine this impact in order to support the design decisions. The approach is applied to design services of a service-oriented surveillance system with comprehensible service characteristics.**

Keywords-service design; design decision; support; soaml

I. INTRODUCTION

Services within service-oriented architectures are expected to fulfill certain service characteristics, such as loose coupling or high autonomy. These characteristics are a prerequisite that the goals that are associated with the shift to a service-oriented architecture, such as increased flexibility of the information technology (IT) and better alignment with the business [1, 2, 19], can be attained.

The development of new services requires their design, which includes their identification and specification. The identification phase focuses on the determination of required services and their capabilities. During this phase, these services are called service candidates and the capabilities are called capability candidates [1, 2, 3, 6]. The subsequent specification phase refines the prior service candidates and results in one final service design for each service. A service design constitutes the basis for the implementation phase and represents the essential information of a service: It describes the provided service interface, the service component that realizes the service logic, and potentially required services in form of required service interfaces [1, 13, 20]. The formalization of this information results in a service design model. When designing services, i.e. when identifying and specifying them, several design decisions have to be made, such as how to group capability candidates into service candidates or how to name a service. Each of these design decisions influences the services and thus their characteristics. Hence, it is necessary to make each of these decisions with care and keep their impact on the service characteristics in mind.

However, existing development processes in the context of service-oriented architectures, as introduced by Erl [1], Engels et al. [2], the Rational Unified Process [14] for Service-Oriented Modeling Architecture (RUP SOMA) [3, 4, 5], and the Service Oriented Architecture Framework (SOAF) [6], only describe the steps that are necessary to create services at a high level of abstraction, such as the identification of capability candidates. More detailed design decisions that have to be made when performing these steps are not explicitly emphasized. Work, as introduced by Erl [1, 7], Engels et al. [2], Reussner et al. [8], Josuttis [9], Maier et al. [10, 11, 12], and SoaML [13], focuses on service characteristics a service should follow. However, the authors of this work do not address which design decisions within the development process impact these characteristics. Thus, the impact of each decision when designing services on the service characteristics has to be presumed. Additionally, if a specific service characteristic is supposed to be fulfilled best, it is unknown, which steps have to be performed with special diligence.

In this paper, we introduce an approach to determine the impact of design decisions on the characteristics of services in order to support making them. For this purpose, design decisions, as they appear in existing development processes, are associated with service attributes that are affected if a decision is made. A service attribute represents a service characteristic without its concrete value, such as coupling or autonomy and a decision influences the correlating service characteristic, i.e. whether the coupling is loose or tight. The association between design decisions and service attributes helps to raise awareness of the impact of design decisions and to keep their impact in mind when making a decision. Also if a service has to be designed with a specific characteristic, the design decisions that affect the correlating attribute and thus have to be performed with special diligence can be identified.

To illustrate our approach, services of a service-oriented surveillance system are designed bearing the surveillance system N.E.S.T. of the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation [21, 22] in mind. The services are designed with respect to loose coupling, high autonomy, unique classification, and high discoverability as desired service characteristics. The design is performed according to existing development processes. Preliminary service candidates and final service designs are created with the Service-oriented architecture Modeling

Language (SoaML) [13], because it is a standardized UML profile [23] and metamodel for describing and formalizing service-oriented architectures. Though SoaML is a very new UML profile and metamodel and still under development, it is becoming increasingly accepted and employed.

The paper is organized as follows: Section 2 presents the related work in the context of development processes, service characteristics, and formalization of service designs. In Section 3, the entire approach is introduced and exemplarily applied to design services of the service-oriented surveillance system. Section 4 concludes the paper and offers suggestions for future research.

II. Related Work

Erl [1], Engels et al. [2], the Rational Unified Process [14] for Service Oriented Modeling and Architecture (RUP SOMA) [3, 4, 5], and the Service Oriented Architecture Framework (SOAF) [6] introduce development processes for the development of services in the context of service-oriented architectures. The processes have the general abstract phases in common: They start with an analysis of the requirements, followed by the design consisting of an identification of the required services in form of service candidates and the subsequent specification of concrete services resulting in a service design for each service [1, 7]. A service design includes a description of the provided service interface, the component realizing the internal service logic, and the required services. According to Service Component Architecture (SCA) [20], the realizing component is defined as service component. A service design is formalized as service design model. However, a common language to formalize a service design is not applied within the development processes. After the design phase, the services are realized, i.e. implemented, tested, and deployed. The processes describe the required tasks that have to be performed during the identification and specification phase, such as analysis of existing assets or identify service candidates. However, concrete design decisions that have to be made or different action alternatives that can be chosen are not provided. Also the impact of the tasks on the characteristics of the resulting services is missing. In [7], Erl describes service principles that have to be considered during the identification and specification. However, how to create services with certain characteristics is not explained. We use the processes as a guideline for building services and see the phases and tasks as valid and useful. Based on these processes, we identify detailed design decisions that have to be made and associate them with service attributes in order to support the design decisions with respect to the resulting service characteristics. Additionally, we agree with the notion of a service design and use it as artifact to associate design decisions with service attributes.

Erl [1, 7], Engels et al. [2], Reussner et al. [8], Josuttis [9], Maier et al. [10, 11, 12], and SoaML [13] introduce service characteristics that a service should follow, such as loose coupling or high discoverability. However, this work focuses on a textual description of these characteristics and does not address how to design services in order to fulfill them. In [1], Erl associates the characteristics with the abstract identification and specification phases. However, the service characteristics are not assigned to detailed design decisions. Thus, design decisions that have to be performed with special diligence during the abstract phases in order to create a service design with certain characteristics are unknown. We see these characteristics as valid and reuse their descriptions in order to derive design attributes that can be used to associate design decisions with service attributes.

The Service-oriented architecture Modeling Language (SoaML) [13] is a standardized UML profile [23] and metamodel from the OMG for describing and formalizing service-oriented architectures. Various modeling elements enable the description of entire service-oriented architectures and single services in detail. There exist elements within SoaML that correlates with service candidates and final service designs. The element ServiceInterface represents a service interface. It realizes a technical interface that describes the provided operations and can use a technical interface that describes the operations the service consumer has to provide to receive callbacks. Additionally, a ServiceInterface can include an OwnedBehavior as interaction protocol that describes interactions between the service consumer and provider for a valid result. The service component is represented as Participant. A Participant provides services as ServicePoints that are typed by the provided ServiceInterface. Required services are added as RequestPoints, each typed by the required ServiceInterface. A Participant can also include an OwnedBehavior that describes the internal logic. Even though SoaML is currently only available as preliminary beta version, due to its compliance to the understanding of service candidates and service designs and its increasing acceptance and employment, we chose SoaML as formalization for service candidates and service designs. Additionally, we use this formalization to associate design decisions with service attributes in order to support design decisions.

III. Supporting Service Design Decisions

The approach to support design decisions is based on the association of design decisions that have to be made with service attributes. A service attribute represents a service characteristic without its value, such as coupling or autonomy. On the one hand, every design decision influences the resulting service design, i.e. elements are added or change. On the other hand, the characteristics of a service are influenced by the service design, i.e. its elements. Thus, a decision impacts a service characteristic and can be associated with a service attribute. To establish this association, the elements of service designs are used as connecting elements. Since a service attribute refers to a service as a whole, first, each service attribute has to be broken down into design attributes that refer to certain elements of a service design. Afterwards, design decisions that have to be made can be analyzed associated with these design attributes using the elements affected within a service design. The approach is illustrated in Figure 1 including one example in the context of high discoverability as service characteristic.

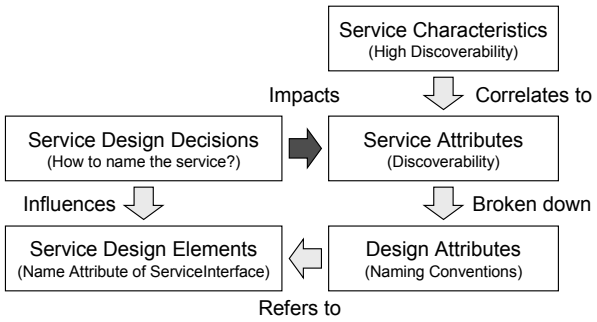

Figure 1. Approach to support service design decisions

A. *Evaluable Design Attributes*

To derive evaluable design attributes that refer to elements of a service design instead to the service as a whole, the textual descriptions of common and widespread service characteristics are analyzed and the evaluable criteria to fulfill these characteristics are identified. To exemplify the approach, the service characteristics loose coupling, high autonomy, unambiguous classification, and high discoverability are chosen. For each of them one design attribute with their preferred characteristic in SoaML and thus the affected elements in SoaML are determined. It is only necessary to identify the affected elements specific for the considered design attributes. The derivation itself is not the focus of the paper and therefore not further explained.

According to SoaML [13], a criterion for loose coupling is that the operations mostly use message style parameters instead of Remote Procedure Call (RPC) style. We define this criterion as design attribute "Parameter Style". It refers to the parameter types of the operations within the provided technical interface of a ServiceInterface.

In [1, 7], Erl describes that high autonomy requires the provided capabilities of a service to be not redundant to capabilities provided by other services. This criterion is defined as design attribute "Capability Redundancy". In SoaML, it refers to operations within Capability elements, that represent service capability candidates.

According to Erl [1, 7], Engels et al. [2], Reussner et al. [8], and Maier et al. [10, 11, 12], a service should be unambiguously classifiable. This includes that all capabilities of a service candidate should either be only responsible for managing the data of business entities that equals entity services in Erl [1, 7], or keep business logic that does not include the management of business entities that equals task services. These different kinds of logic should never be mixed. We define this criterion as design attribute "Entity / Task Classification". It refers to the operations within Capability elements.

Finally, according to Erl [7], a service is more discoverable, if the service itself, its provided operations, and the included parameters and parameter types follow conventions, such as naming conventions. We define this criterion as design attribute "Convention Compliance". It refers to the name attribute of a ServiceInterface in SoaML and the name attributes of operations, parameters, and parameter types within its provided technical interface.

B. *Design Decisions during Service Identification*

In the following, services of a service-oriented surveillance system are designed according to Engels et al. [2] combined with approaches, as introduced in Erl [1], RUP SOMA [3, 4, 5], and SOAF [6]. This example is used to identify a set of design decisions that have to be made. Each decision is associated with affected elements of a service design and subsequently associated with design attribute, thus service attribute. This enables the support of the design decision and thus a systematically design of the services.

The development process starts with the analysis of the business that is supposed to be supported. The scenario starts with a visitor entering a building. The visitor registers at the reception, and the reception takes a photo and asks the personnel administration to check the visitor's identity. Afterwards, it requests the role of the visitor at the personnel administration. Depending on the role, the allowed and forbidden areas are determined and returned to the visitor in form of a map. In a next step, the reception requests the security department to surveil the visitor in order to ensure that the visitor only accesses allowed areas. For this purpose, the security department again requests the role at the personnel administration and determines allowed and forbidden areas. Afterwards, it requests the current position of the person at the facility management based on the photo taken at the reception. The Facility management observes all persons within a building using cameras. In a next step, the current position is compared with the areas. If it is within a forbidden area, the alarm is set off. Otherwise, the surveillance is continued. When the visitor de-registers at the reception, the reception stops the surveillance.

According to this scenario description, the business services as displayed in Figure 2 can be determined. They are modeled as use cases with the performing roles as actors as introduced in Engels et al. [2]. For each business service a realizing business process exists that can also be created considering existing systems if desired [17]. Figure 3 shows the relevant business processes using BPMN [24].

Afterwards, the services have to be identified, i.e. service candidates as preliminary services equipped with capability candidates are created. All tasks and pools except the task "Take Photo" and the visitor pool are expected to be IT-supported. Since the services are expected to reflect the business, each business service that is fully or partially IT-supported is transformed into a service candidate that realizes the business service. Additionally, each activity that is invoked across pools using message flows is added as capability candidate.

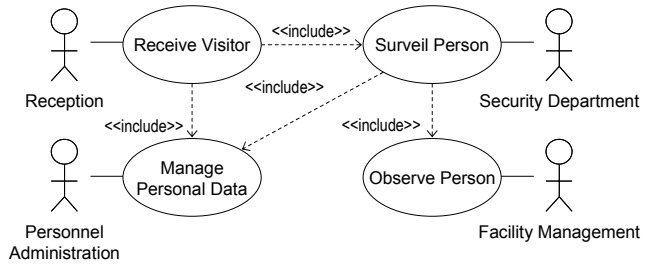

Figure 2. Business services and performing roles

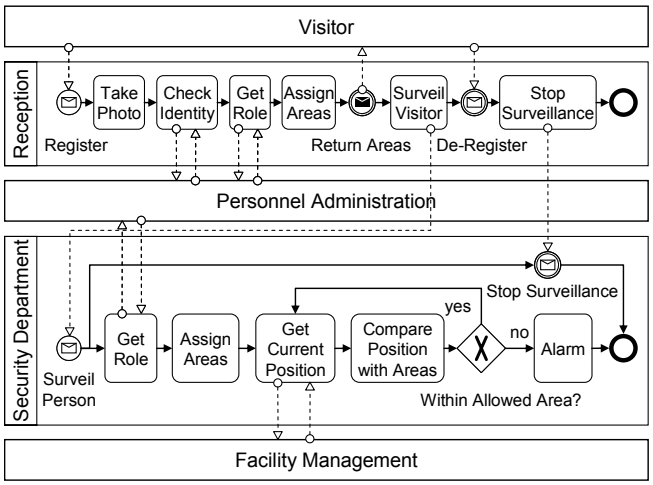

Figure 3. Business processes

In SoaML a service candidate is represented as a Capability element with operations representing the capabilities. The derived services are displayed in Figure 4.

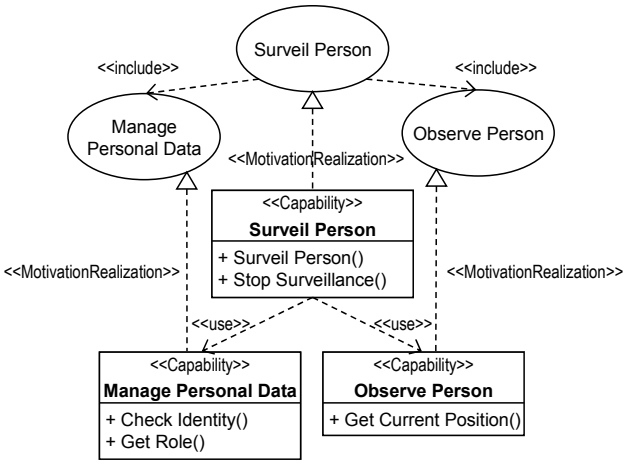

Figure 4. Derived service candidates

Now, the IT architect decides to revise these service candidates. Here, the first design decisions have to be made or revised.

How to group the capability candidates into service candidates?

Since this design decision influences the operations of Capability elements in SoaML, the design attributes "Capability Redundancy", and "Entity / Task Classification" are affected by this decision. Thus, the IT architect is made aware of the impact of this decision. He knows that this decision influences the classification and coupling of the services. In this case, the design decision can even be refined in more detailed design decisions. For these design decisions a decision tree can be specified that shows the different action alternatives.

Should a specific capability candidate be moved into another specific service candidate?

The following action alternatives can be identified: No move, a move into a new service candidate, or a move into an existing service candidate. For the latter case, each service candidate can be seen as action alternative. It is important to notice that each action alternative that is identified has to result in a valid state, i.e. in a set of service candidates that is valid and functional correct. For example, a decision "remove capability candidate?" may result in a state with incomplete functionality. The decision tree for the exemplarily decision, whether to move the capability candidate "Get Role" is illustrated in Figure 5.

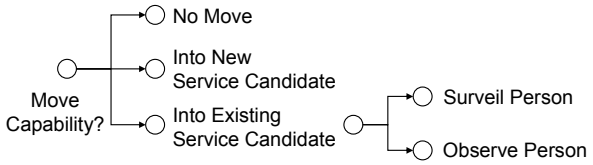

Figure 5. Decision tree

To make this decision, the different action alternatives can be evaluated. The impact of the alternatives on each service candidate is displayed in the following table. For each design attribute, it is displayed whether it improves, gets worse or does not change.

TABLE I. IMPACT ON DESIGN ATTRIBUTES

Service Candidate	*Capability Redundancy*	*Task / Entity Classification*
Move into new service candidate		
Surveil Person	→	→
Observe Person	→	→
Manage Personal Data	→	↗
Move into Surveil Person		
Surveil Person	→	↘
Observe Person	→	→
Manage Personal Data	→	↗
Move into Observe Person		
Surveil Person	→	→
Observe Person	→	↘
Manage Personal Data	→	↗

According to these results, the IT architect decides for the alternative to move the capability candidate "Get Role" into a new service candidate. This move improves the classification, because now, the service candidate "Manage Personal Data" is split into two services: One that provides complex tasks, such as the check of identity (task service), and one that provides data access (entity service). This decision was comprehensibly made by the association of design decisions with service attributes. The added service candidate is named "Manage Personal Data 2". The naming of service candidates is not important at the identification phase. Similarly, another design decisions during the identification phase can be identified and made using the same approach:

Should a specific IT-Supported activity be added as capability candidate into a specific service candidate?

For example, in this case, the activity "Assign Areas" could be provided as own capability candidate as proposed in [15]. Then, this functionality could be shared across several service consumers and only needs to be implemented once. However, this would also result in a new service that has to be provided on an organizational level. This results in the following conclusion: Since each change concerning the service candidates and its capability candidates influences the services that have to be provided and maintained by the business, decisions that are made during the service identification phase directly influence the business, i.e. the provided services, its roles or responsibilities. Thus, each decision has to be made after consultation with the business analyst. The revised service candidates are illustrated in the Figure 6.

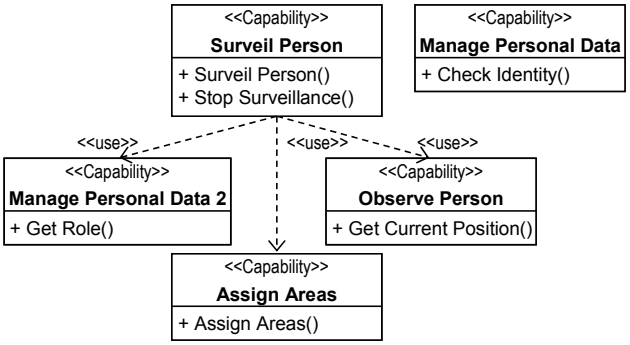

Figure 6. Revised service candidates

Further design decisions during the identification phase can be supported in a similar way and are therefore not considered.

C. Design Decisions during Service Specification

In a next step, the identified service candidates are specified. This includes the specification of the provided service interface and – if the service has to be implemented and is not reused by existing applications or service providers – the service component and required services are specified. Also during the specification phase, several design decisions have to be made that can be supported by the association with service attributes. We exemplify a subset of design decisions by specifying the service candidate "Surveil Person". In a first step, a ServiceInterface is created that exposes the service candidate and realizes an interface containing the capability candidates as operations. The first design decision is:

How to name the service?

In this case, the name attribute of a ServiceInterface is affected. Also the design attribute "Convention Compliance" refers to the name attribute of a ServiceInterface. With this association, the IT architect is made aware that this decision influences the discoverability of the service and that this task has to be performed with special diligence if he targets a high discoverability. Hence, the IT architect decides to name the service as noun without spaces, "PersonSurveillance". In a next step, the provided operations that were derived from capability candidates are revised:

How to name the operations?

Similarly to the decision before, this decision influences the name attribute of an operation and thus can be associated with the design attribute "Convention Compliance". The IT architect decides to rename the operations to "surveilPerson" and "stopSurveillance" in order to increase the discoverability of the service. The next decision is:

How to design the parameter types of the operations?

Also this decision influences the name attribute of parameter types and thus the design attribute "Convention Compliance". Additionally, the choose between message style and RPC style influences the design attribute "Parameter Style". With the association to design attributes, the IT architect is made aware that this decision influences the discoverability and coupling of a service. The IT architect names the parameter types compliant to a particular scheme and uses message styles in order to create services with looser coupling.

Further design decisions during the specification phase can be supported in a similar way and are therefore not considered. Figure 7 shows the resulting service interface.

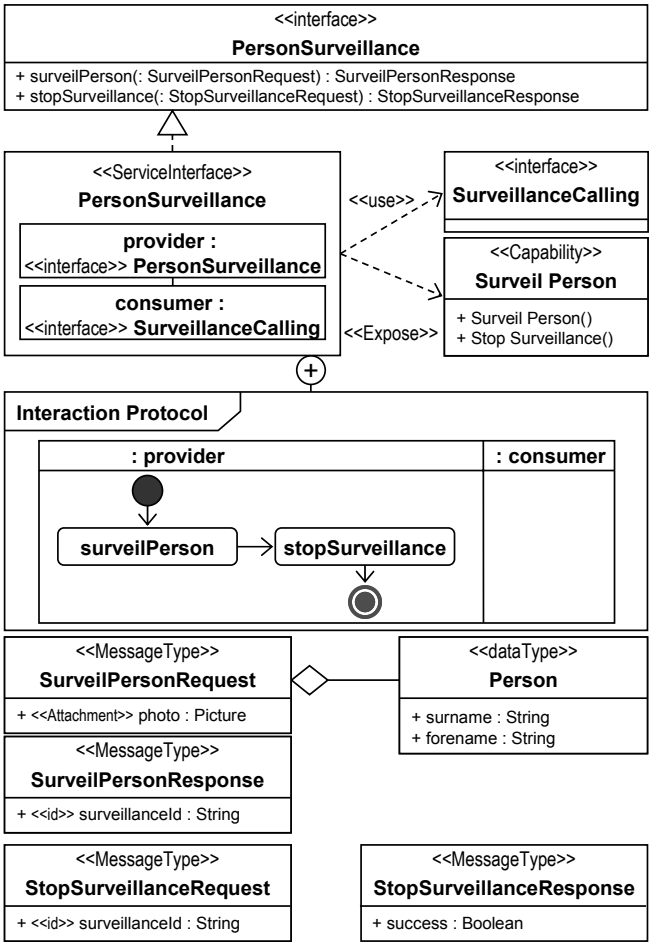

Figure 7. Resulting service interface and parameter types

IV. Conclusion and Outlook

In this paper, we presented an approach to support design decisions that have to be made when creating a service design. The approach is based on the association of design decisions with service attributes over the service design model. Since each design decision influences elements within a service design model, the impact on service characteristics can be determined. For this purpose, a exemplarily set of service attributes was broken down into design attributes that refer to elements within a service design model. Afterwards, a subset of the design decisions that have to be made during the creation of a service design were identified. To model the service designs and to create the association between design decisions and design attributes, thus service attributes, SoaML from the OMG was chosen, because SoaML represents an emerging standard for modeling service-oriented architectures. Due to the usage of SoaML as UML profile, our approach can be embedded in existing and UML-capable development tools, because SoaML is available as XMI [25].

The association of design decisions with design attributes, thus service attributes helps IT architects to design services more systematically. For each decision that has to be made, now the IT architect knows which service attributes are affected. On the one hand this raises awareness of the influence a design decision takes on the final characteristic of a service. So, the IT architect will make a design decision with care. On the other hand, if two contrarily service attributes are affected, the IT architect can directly decide which attribute should be preferably optimized. Finally, when a service should be created with certain characteristics, the design decisions that have to be made with special diligence are known.

To illustrate our approach, services of a service-oriented surveillance system were designed. A subset of design decisions that had to be made was associated with design attributes, thus service attributes. This enabled us to design the services systematically. Before a design decision was made, the impact of this decision could be determined and the awareness of the impact of this decision was raised. Afterwards, the decision was made with respect to the affected service attribute. Thus, each design decision was comprehensibly made.

In our future work we plan to embed our approach into existing development tools in order to further utilize the support of service design decisions. Affected design attributes should be automatically determined and the impact of design decisions should be visualized for the IT architect. The design decisions identified in this paper and their association to design attributes will be used to list action alternatives that may improve the service design. Our goal is to support the entire design process of services based on common and widespread service characteristics. The entire approach will be applied to design services in the context of campus management and for a currently developed human-centered environmental observation system.

References

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

[2] 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.

[3] IBM, "RUP for service-oriented modeling and architecture", IBM Developer Works, http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/, 2006. [accessed: May 10, 2010]

[4] 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.

[5] 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: May 10, 2010]

[6] A. Erradi, S. Anand, and N. Kulkarni, "SOAF: An Architectural Framework for Service Definition and Realization", 2006.

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

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

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

[10] B. Maier, H. Normann, B. Trops, C. Utschig-Utschig, and T. Winterberg, „Lose kopplung – warum das loslassen verbindet", SOA-Spezial, Software & Support Verlag, 2009.

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

[12] 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.

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

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

[15] M. Gebhart and S. Abeck, "Rule-based service modeling", 2009.

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

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

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

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

[20] Open SOA (OSOA), "Service component architecture (SCA), sca assembly model V1.00", http://osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf, 2009. [accessed: May 10, 2010]

[21] A. Bauer, S. Eckel, T. Emter, A. Laubenheimer, E. Monari, J. Moßgraber, and F. Reinert, "N.E.S.T. – network enabled surveillance and tracking", Future Security 3rd Security Research Conference Karlsruhe, 2008.

[22] J. Moßgraber, F. Reinert, and H. Vagts, "An architecture for a task-oriented surveillance system", 2009.

[23] OMG, "Unified modeling language, superstructure", Version 2.2, 2009.

[24] OMG, "Business Process Model and Notation (BPMN)", Version 2.0 Beta 1, 2009.

[25] OMG, "XML metadata interchange (XMI) specification", Version 2.0, 2003.