Measuring Design Quality of Service-Oriented Architectures Based on Web Services

Michael Gebhart
Gebhart Quality Analysis (QA) 82 GmbH
Karlsruhe, Germany
michael.gebhart@qa82.de

***Abstract*—For achieving a flexible and maintainable IT, companies increasingly design their IT architecture in a service-oriented manner using web services. As the effectiveness of this transition is influenced by the design of the architecture, patterns and best-practices have been evolved that are expected to be considered during the development process. However, reviewing the architecture regarding these guidelines is complex and time-consuming as a lot of interpretation and calculation has to be performed. This article introduces an approach for efficiently measuring design quality with a focus on the service layer, thus the service interface and service component design. To illustrate the approach, services of an automotive scenario are developed using a product that integrates the introduced concepts.**

Keywords-soa; web service; design; quality; metrics

I. INTRODUCTION

The ability to realize new business requirements within shortest time has become a critical success factor for companies. This requires the IT to be both flexible and maintainable, which constitute main drivers for service-oriented architecture (SOA) projects [1][2]. While SOA does not dictate any technology usage, in most cases web services are applied as their standardization increases the flexibility and maintainability of the architecture from a technical point of view [3]. In this case, the web services are described using the World Wide Web Consortium (W3C) standards Web Services Description Language (WSDL) [4] and XML Schema Definition (XSD) [5]. Furthermore, in some projects the Service Component Architecture (SCA) [6] standardized by the Organization for the Advancement of Structured Information Standards (OASIS) is applied to describe the component model.

In the past, many projects have shown that the success of SOA projects is influenced by the design of the architecture especially its service layer [7]. On a service layer the architecture targets the design of service interfaces, service components, and their dependencies. Decisions, such as the grouping of operations to services and their granularity, impact the achievement of the previously described goals. For that reason in literature many best-practices and patterns have been identified that describe how to design the service layer. Furthermore, companies also establish standards or design guidelines that represent internal experiences and might be company-, industry-, or project-specific. Developers are expected to consider these guidelines during their work. This requires a solid understanding of the guidelines and discipline to not overlook any application. From a project management perspective it is also necessary to ensure a consistent application of the guidelines.

In both cases, the review of developed web services regarding these requirements is complex and time-consuming. Besides the necessary interpretation and solid understanding a manual analysis of every web service and its relations to other services has to be performed. Furthermore, every change requires a new analysis not only of the changed service but – due to interdependencies – of all web services. The necessary effort is costly and mostly cannot be asserted. In addition, with increasing complexity of the architecture measure mistakes become more likely due to the high number of performed calculations. The result is that quality analyses regarding guidelines are often neglected even though they are relevant for the creation of a flexible and maintainable architecture and the success of SOA projects.

This article introduces an approach to simplify those analyses on a service layer by means of appropriate automation or at least semi-automation. For that purpose, existing best-practices and patterns for service interfaces and service components are formalized so that no interpretation effort is necessary and their compliance can be automatically or at least semi-automatically verified. Even though the internal behavior of a service component, such as its implementation using object-oriented languages, influences the quality of the architecture as a whole, in this article the focus is on the service part represented by the service layer. When designing a service-oriented architecture from a strategic point of view, this is the first essential design task that has to be performed. Previous work in the context of service design metrics will serve as basis for this article. In [8], Gebhart et al. introduced metrics for service designs based on the Service oriented architecture Modeling Language (SoaML) that represent design guidelines. These metrics have been demonstrated by a case study in [9]. Combined with work that describes the relation between SoaML and web services [10] service design metrics based on SoaML are transferred to web services based on WSDL, XSD, and SCA. As result, web services can be automatically analyzed regarding wide-spread guidelines. Furthermore, the methodology can be applied on any other company-, industry-, or project-specific design guidelines.

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-304-9

The concept is illustrated using a scenario in the context of automotive manufacturing. In this case, the usage of formalized guidelines helps to systematically design web services and to coordinate several developers. Furthermore, the concepts are integrated into the QA82 Analyzer as product for analyzing software and data. The product enables the automatic measurement of the design quality of the created SOA, thus increases the efficiency.

The article is organized as follows: Section II introduces existing guidelines for web services and their formalizations. The scenario is introduced in Section III. In Section IV, the services are designed using the formalized guidelines and our product. Section V concludes this article and introduces future research work.

II. BACKGROUND

This section describes guidelines for the design of services in service-oriented architectures that will be considered within the scenario. Furthermore, this work is examined regarding the possibility to be efficiently measured using tool support. The technologies of web services, such as WSDL, XSD, and SCA are not further introduced in this article. They are assumed to be well known.

The service design phase is an essential ingredient of software service engineering that can be defined as the "discipline for development and maintenance of SOA-enabled applications" [11]. The service design phase includes decisions about the interface of a certain service, such as its grouping of operations, and its internal behavior. As services constitute the building blocks of an SOA, they determine its design. For services several best-practices and patterns have been evolved as guidelines.

In [7] and [12], Erl describes numerous patterns for services in particular web services. They have been derived from experiences in real-world projects and provide valuable hints for architects and developers. Nevertheless, all guidelines are only textually describes. This results in ambiguities and requires interpretation before using it in concrete projects. This again may result in faulty applications.

Similar to Erl, also Cohen [13] and Josuttis [14] focus on patterns from a similar point of view. While the guidelines are clearly motivated, their usage in projects requires interpretation. Furthermore, due to the textual description concrete artifacts cannot be checked against these guidelines without manual effort.

A more academic approach is chosen in [15] and [16]. Perepletchikov et al. introduce metrics for quality attributes, such as loose couplings. These metrics consider formalized service designs independent from concrete technologies. The essential benefit of this work is its ability to perform an automatic measurement. However, the motivation of the introduced metrics is not obvious. Work as introduced by Erl and Josuttis is not reflected by the metrics. This is even not possible as Perepletchikov et al. consider an abstract formalization of services. Most of the aspects described by best-practices refer to elements that are not part of this formalization.

Similarly to Perepletchikov et al. also Hirzalla et al. [17] and Choi et al. [18] introduce metrics for services. Also in this work, the metrics are very abstract and cannot be directly applied in projects. They do not represent best-practices as introduced by Erl and Josuttis.

To fill this gap, in previous work we created a quality model that combines best-practices as introduced by Erl et al. with a formalization as used by Perepletchikov et al. [8]. The quality model was aligned with the Service oriented architecture Modeling Language (SoaML) [19] as profile for the Unified Modeling Language (UML) [20] that is meant to replace proprietary UML profiles for services, such as the one developed by IBM [21][22][23]. As result of this work, an SOA formalized using SoaML can be checked against wide-spread guidelines. The usage of SoaML is explained in [24][25] and a case study that applies the metrics is presented in [9]. However, in most cases web services are created or are already existent without a formalization based on SoaML. Furthermore, some guidelines refer to elements that are not part of a SoaML-based description. Thus, an approach is necessary that is applicable on web services directly.

In [10], it is shown how service designs based on SoaML can be transformed into web services using the WSDL, XSD, and SCA. This work was not necessarily created with quality analysis in mind. However, it can be applied to transfer the service design metrics based on SoaML to web services.

The summary of existing work shows, that a lot of good work exists that focuses either on the description of best-practices, patterns, design guidelines etc. for web services or on a formalization of academic metrics. Whilst the former are too abstract to be efficiently measured, the latter are too academic to be comprehensible understandable and motivated. For that reason we use the metrics introduced in [8] that on the one hand represent best-practices and on the other hand are formalized so that they can be automatically measured. They are transformed so that they can be applied on web services using the mapping rules described in [10].

III. SCENARIO

To illustrate the quality analysis of a service-oriented architecture design, a scenario from automotive manufacturing is chosen.

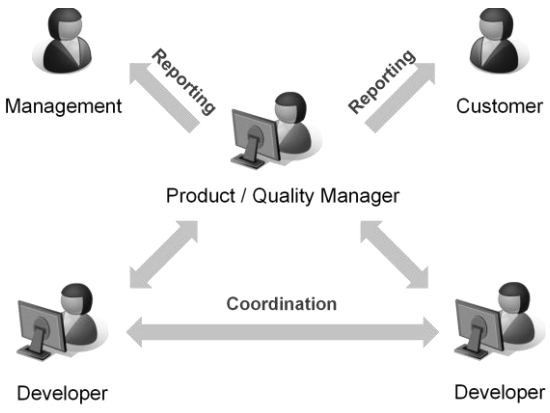

Figure 1. Participants and their relationships.

There is a product and quality manager who coordinates two developers and in addition delivers reports to the management and the customer. In some cases, the role of the product and quality manager might also be fulfilled by an architect, who is responsible for the design of the architecture and its quality. Fig. 1 illustrates the participants and their relationships.

According to this figure, the product and quality manager has an interest in proving the high quality of the created software. In this scenario, besides functional requirements especially the architectural design is considered. So it is necessary that he understands the meaning of high quality in the context of service-oriented architecture design. Furthermore, he is required to analyze software artifacts regarding these quality requirements. To support this quality assurance, this article shows how to analyze artifacts, such as web service interfaces, regarding wide-spread best-practices and guidelines for services.

The scenario begins with the development of a service for the manufacturing of automobiles by the first developer. An SCA Composite is created, which combines a service for manufacturing automobiles and a service for filing manufactured automobiles in the database. The artifacts are filed in a shared Git repository. Fig. 2 illustrates the composite using the graphical representation introduced in the official SCA standard. In the scenario, originally a proprietary tool is used that uses a different visualization.

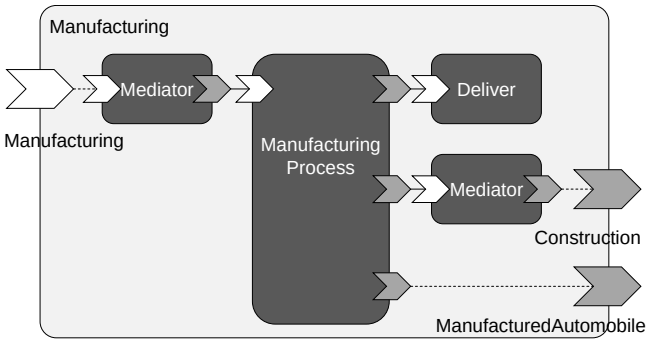

Figure 2. Created SCA composite.

Starting with this SCA composite the product and quality manager determines the quality of the architecture using the approach introduced in the following section.

IV. Measuring Design Quality of Service-Oriented Architectures Based on Web Services

To determine the quality of software, one approach is to refine the term quality until it can be measured. A wide-spread quality model methodology is Factor, Criteria, Metric (FCM) introduced by McCall et al. in [26]. According to this methodology a factor is refined into more fine-grained criteria that again are refined into quantifiable metrics. Similar approaches use the equivalent terms quality characteristics, quality sub-characteristics, and quality indicators.

Correspondingly, applied on the design of service-oriented architectures the term quality from a design perspective has to be broken down into measurable aspects that can be formalized by means of metrics. In [8], a quality model has been created that enables the measurement or at least systematic determination of best-practices and patterns that have been identified as important for service-oriented architectures. However, the quality model has been formalized on basis of Service oriented architecture Modeling Language (SoaML) as language to formalize the architecture. When the product and quality manager of the scenario in Section III tries to apply this quality model, the usage of SoaML hampers the direct. As in the scenario other technologies in particular WSDL, XSD, and SCA are used, the metrics introduced in [8] cannot be applied without additional effort. However, in [10], a mapping between SoaML and web service technologies is described. The combination of this work enables the transformation of metrics onto web services so that they can be directly applied. This application is shown next.

A. Application of Metrics

According to Gebhart et al. [8] in particularly four quality sub-characteristics or criteria can be considered as relevant for the design quality: Unique categorization, loose coupling, discoverability, and autonomy. Even though this set of quality characteristics is not expected to be complete it is a good starting point to evaluate the design of a service-oriented architecture and to illustrate the approach.

In this section, especially the unique categorization as quality sub-characteristic is considered. This sub-characteristic is comparable to the concept of cohesion in object-oriented systems. It consists of four quality indicators with metrics introduced in [8][27][28]. To illustrate the approach, these metrics are mapped and applied to analyze the service-oriented architecture design.

1) Division of Agnostic and Non-Agnostic Functionality:

TABLE I. Variables and Functions used for DANF

Element	Description and Mapping
DANF	Division of Agnostic and Non-agnostic Functionality
s	service: the considered service that is provided or required It is represented by a SCA Service or Reference element.
SI(s)	Service Interface: service interface of the service s It is represented by the WSDL document that describes the SCA Service or Reference.
RI(si)	Realized Interfaces: realized interfaces of the service interface si. It is represented by the WSDL PortType that includes provided operations of the service.
O(i)	Operations: operations within the interface i The WSDL Operations within the identified WSDL PortType are expected to be returned.
AF(o)	Agnostic Functionality: operations providing agnostic functionality out of the set of operations o This information has to be determined by an IT expert. It cannot be found within the web service technologies.
| o |	Number of operations o

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-304-9

The background of this metric is that generic functionality should be split from specific ones so that changes regarding the specific operations do not affect the highly reused ones. It has its origin in the patterns described by Erl [7].

$$DANF(s) = \frac{\left|AF\left(O\left(RI(SI(s))\right)\right)\right|}{\left|O\left(RI(SI(s))\right)\right|} \quad (1)$$

To apply this metric for the scenario, the functions and variables have to be mapped onto elements within XSD, WSDL, and SCA. Table I shows a brief introduction of the element and afterwards a mapping. This mapping specifies where to find this information.

As result a value of 0 or 1 is desired. These values mean that the service operations provide only agnostic or only non-agnostic functionality.

Based on this mapping information, the metric can be applied for the Manufacturing service that is the SCA Service within the SCA Composite. According to the metric, in a first step the service interface has to be identified. This is the WSDL file Manufacturing.wsdl. Next, the WSDL PortType comprising the provided operations within the WSDL is selected and finally, the operations themselves are returned. Fig. 3 shows the proceeding.

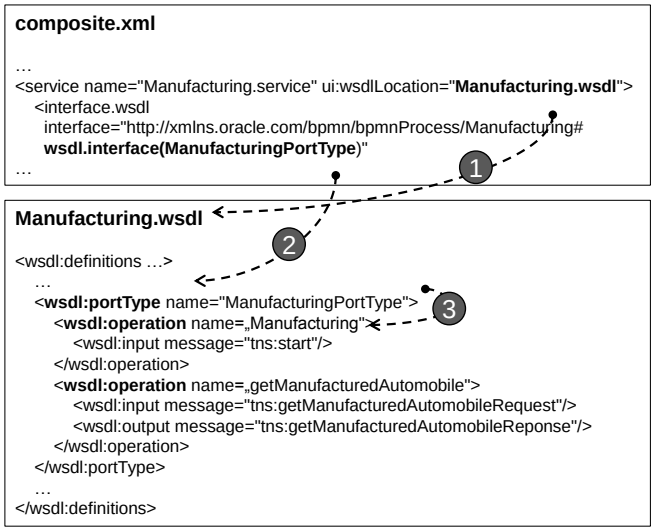

Figure 3. Determination of DANF metric.

After the relevant operations have been identified, the IT quality manager has to decide whether these operations are agnostic or non-agnostic. If he is not capable to answer these questions, he has to ask the developers and estimate the reusability of these operations. In this case, the quality manager comes to the conclusion that the operation "Manufacture" is non-agnostic as it is very specific and cannot be used in other contexts. The operation "getManufacturedAutomobiles" however is agnostic as it provides functionality to request manufactured automobiles, which can be reused in several scenarios. As result the metric returns 0.5, which represents a suboptimal value.

2) Division of Business-Related and Technical Functionality: A metric similar to DANF is DBTF that targets the division of business and technical functionality. It can be mapped in a similar way. To illustrate the approach a more complex metric, the data superiority, is chosen next.

3) Data Superiority: This quality sub-characteristic describes that a service that manages an entity is exclusively responsible for managing it. The metric can be formalized as follows. Most functions have already been described. The others are explained in Table II.

$$DS(s) = 1 - \frac{\left|\begin{matrix} ME\left(O\left(RI(SI(s))\right)\right) \cap \\ ME\left(O\left(RI\left(SI\left((ALL_S \setminus s)\right)\right)\right)\right) \end{matrix}\right|}{\left|ME\left(O\left(RI(SI(s))\right)\right)\right|} \quad (2)$$

TABLE II. VARIABLES AND FUNCTIONS USED FOR DS

Element	Description and Mapping
DS	Data Superiority
M1 \ M2	Elements of set M1 without elements of set M2 or the element M2
ALL_S	All existing services Represented by all SCA Services
ME(o)	Managed Entities: entities that are managed by operations o This information has to be determined by an IT expert. It cannot be found within the web service technologies.

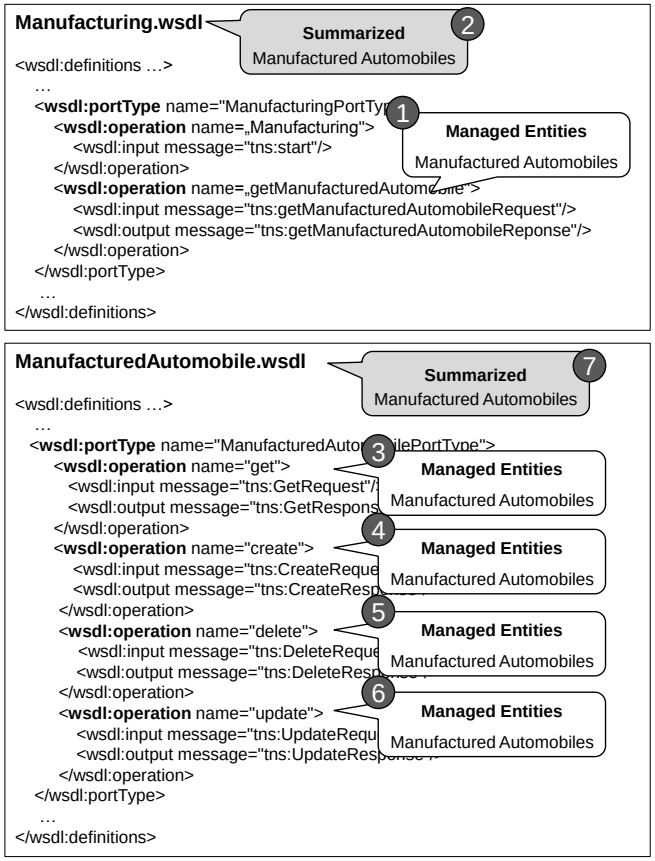

Figure 4. Determination of DS metric.

To illustrate this metric we assume that the ManufacturedAutomobile Reference within the SCA Composite refers to a service described by the ManufacturedAutomobile.wsdl and that no other services are relevant for this metric.

To calculate the metric, the product and quality manager has to consider the provided operations of the Manufacturing service and of all other services, i.e., the ManufacturedAutomobile service in this case. Afterwards, he has to decide for each operation whether an entity is managed by this one. Finally, he has to compare the set of managed entities of the services to identify conflicts. Fig. 4 illustrates the proceeding for the Manufacturing service. According to this figure all entities managed by the Manufacturing service are not exclusively managed. The Manufactured Automobile service that corresponds to an entity service [1][7] manages manufactured automobiles too. So from a data superiority perspective the Manufacturing service is not ideal and should be revised.

4) Common Entity Usage: Finally the last quality indicator of the unique categorization quality sub-characteristic can be measured. According to the common entity usage metric, all operations within a service should work on the same entities. This guarantees that entities that do not belong together are managed by different services. In turn, the prior described data superiority ensures that operations that manage the same entities are part of one service.

$$CEU(s) = \frac{\left| OUE\left(CMP\left(\begin{matrix} O\left(RI(SI(s))\right), \\ O\left(RI(SI(s))\right), MOUE\left(O\left(RI(SI(s))\right)\right), \\ UE\left(O\left(RI(SI(s))\right)\right) \end{matrix} \right) \right) \right|}{\left| O\left(RI(SI(s))\right) \right|} \quad (3)$$

TABLE III. VARIABLES AND FUNCTIONS USED FOR CEU

Element	Description and Mapping
CEU	Common Entity Usage
CMP(o, e1, e2)	Composition: biggest set of entities managed by operations o out of e2 that depend on entitites e1
UE(o)	Used Entities: entities that are used within operations o as input
MOUE(o)	Mostly Often Used Entities: entities that are mostly often used within one operation out of operations o
OUE(o, be)	Operations Using Entities: operations out of operations o that only use entities out of be

This table shows that there is no explicit mapping to web services necessary. All functions that refer to certain elements within a technology have already been mapped by the functions described in Table I and Table II.

Applied on the Manufacturing service the metric returns the value 1 as all operations that manage entities manage the same. This is also the case for the Manufactured Automobile service. As this entity service provides Create, Read, Update, Delete (CRUD) operations for the same entity, this metric is also ideal for this service. If the Manufactured Automobile service would also manage another entity, the CEU metric would return a suboptimal value.

B. Integration into Scenario

Back in our scenario, the quality manager can use the results to inform developers about the design weaknesses. The usage of these metrics in a quality-oriented service design process is illustrated in [29].

For example, the result of DANF shows that the two provided service operations "Manufacture" and "getManufacturedAutomobiles" should be separated into two services. In addition, the result of the DS metric shows the conflict between the operations provided by the Manufactured Automobile service and the operation "getManufacturedAutomobile" of the Manufacturing service. Summarized, the operation "getManufacturedAutomobile" should be deleted as it provides functionality that is also offered by the Manufactured Automobile service. Service consumers using this operation should switch to the Manufactured Automobile Service.

In addition to the revision hints, the results of the metrics can be used to deliver reports to the management and the customer. For example the product and quality manager can justify cost and investments into quality assurances. Furthermore, he can prove the quality of the software by means of objective criteria.

V. CONCLUSION AND OUTLOOK

In this article, an approach was illustrated to measure the design quality of service-oriented architectures regarding wide-spread best-practices and guidelines. For that purpose an existing quality model that refers to SoaML as formalization of a service-oriented architecture design was chosen. By use of another work that describes the mapping between SoaML and web service technologies, this quality model was transferred onto WSDL, XSD, and SCA. By this means the resulting quality model can be directly applied on service-oriented architectures based on web services. The approach demonstrated that for an efficient quality assurance existing quality models should be mapped onto the used technologies.

After an examination of existing work, a scenario from automotive manufacturing was introduced. In this scenario, a product and quality manager is responsible to ensure the quality of the resulting architecture. Next, the mapped quality model was applied to measure the design quality of services in this scenario. The metrics mapped onto web services enable the product and quality manager to identify weaknesses in the current design and thus give the developers hints about possible improvements. In addition, the results can be used to deliver reports to the management and the customer. The reports help to prove the high quality and to justify investments in additional quality assurance projects. Furthermore, developers can perform analyses by themselves. The metrics reduce the additional effort to interpret the textual descriptions. Furthermore, they directly refer to concrete elements within the used technologies.

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-304-9

As part of our research work, we have created a mapping for all metrics introduced in [8]. We also implemented this quality model as part of the QA82 Analyzer [30]. Through this both product and quality managers and developers can automatically measure their service-oriented architecture regarding the quality model. This further increases the efficiency of the quality assurance process.

For the future, we plan to include further quality characteristics both regarding service-oriented architectures and related fields. First, we plan to adapt the approach to analyze services based on REST as it is often applied today. As REST does not prescribe certain interface formalization, we assume that the adaptation will require using more implementation-specific information. Second, we work on a quality model for business process management (BPM) that enables the determination of quality characteristics regarding the functional quality of modeled business processes based on the Business Process Model and Notation (BPMN) 2.0 [31]. This quality model is expected to be linked with the experiences we gained with the quality model introduced in this article. The results of this BPM quality model will be published as well. Furthermore, it will be supported by our quality analysis product. Finally, we aim to formalize the described metrics in a technology-independent but executable way. With languages, such as OCL [32] or XQuery [33] it is possible to describe queries that refer to a certain technology, such as UML or XML. We will examine the applicability of these languages for our purposes.

REFERENCES

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

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

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

[4] W3C, "Web Services Description Language (WSDL)", Version 1.1, 2001.

[5] W3C, "XML Schema Part 0: Primer Second Edition", 2004.

[6] Open SOA (OSOA), "Service component architecture (SCA), sca assembly model V1.00", http://osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf, 2009. [accessed: January 04, 2011]

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

[8] M. Gebhart and S. Abeck, "Metrics for evaluating service designs based on soaml", International Journal on Advances in Software, 4(1&2), 2011, pp. 61-75.

[9] M. Gebhart and S. Sejdovic, "Quality-oriented design of software services in geographical information systems", International Journal on Advances in Software, 5(3&4), 2012, pp. 293-307.

[10] M. Gebhart and J. Bouras, "Mapping between service designs based on soaml and web service implementation artifacts", Seventh International Conference on Software Engineering Advances (ICSEA 2012), Lisbon, Portugal, November 2012, pp. 260-266.

[11] W. van den Heuvel, O. Zimmermann, F. Leymann, P. Lago, I. Schieferdecker, U. Zdun, and P. Avgeriou, „Software Service Engineering: Tenets and Challenges", 2009.

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

[13] S. Cohen, "Ontology and Taxonomy of Services in a Service-Oriented Architecture", Microsoft Architecture Journal, 2007.

[14] N. Josuttis, SOA in Practice, O'Reilly Media, 2007. ISBN 978-0-59-652955-0.

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

[16] 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.

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

[18] 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.

[19] OMG, "Service oriented architecture modeling language (SoaML) – specification for the uml profile and metamodel for services (UPMS)", Version 1.1, 2012.

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

[21] S. Johnston, "UML 2.0 profile for software services", IBM Developer Works, http://www.ibm.com/developerworks/rational/library/05/419_soa/, 2005. [accessed: July 11, 2012]

[22] 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.

[23] 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: July 11, 2012]

[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: July 11, 2012]

[25] M. Gebhart, "Service Identification and Specification with SoaML", in Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments, Vol. I, A. D. Ionita, M. Litoiu, and G. Lewis, Eds. 2012. IGI Global. ISBN 978-1-46662488-7.

[26] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in software quality", 1977.

[27] 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.

[28] M. Gebhart, S. Sejdovic, and S. Abeck, "Case study for a quality-oriented service design process", Sixth Internation Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October 2011, pp. 92-97.

[29] M. Gebhart and S. Abeck, "Quality-oriented design of services", International Journal on Advances in Software, 4(1&2), 2011, pp. 144-157.

[30] Gebhart Quality Analysis (QA) 82, QA82 Architecture Analyzer, http://www.qa82.de. [accessed: July 11, 2012]

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

[32] Object Management Group, "Object constraint language", Version 2.0, 2006.

[33] W3C, "XQuery 1.0: an XML query language (second edition)", Version 1.0, 2010.

Copyright (c) IARIA, 2013. ISBN: 978-1-61208-304-9