Query-Based Static Analysis of Web Services in Service-Oriented Architectures

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

Abstract—**The switch to a service-oriented architecture is often associated with strategic goals, such as an increased flexibility and maintainability of the IT architecture. The design of the services as building blocks directly influences the achievement of these goals. For that reason, in recent years best practices and patterns have evolved that describe how to design services and how to implement them by means of web service technologies. However, the best practices and patterns that focus on the architectural issues are often too abstract to be verified on concrete web service artifacts. Previous work describes how these best practices and patterns can be broken down into measurable quality indicators. This article shows a query-based approach for a static analysis to measure these quality indicators on implemented web services. To illustrate the approach, services of an automotive scenario are developed using a product that realizes the introduced concepts.**

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

I. INTRODUCTION

This article is an extended version of [1]. When companies switch to a service-oriented architecture (SOA) as paradigm to structure their IT architecture, in most cases strategic goals are the main drivers. Typical strategic goals are to increase the flexibility and maintainability as the ability to realize new business requirements within shortest time has become a critical success factor for companies [2][3]. In the past, experiences have shown that the success of SOA projects is influenced by the design of the architecture especially its service layer [4]. On a service layer the architecture focuses on 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 several best practices and patterns from a conceptual point of view have evolved that describe how to design services in a way that they support the achievement of these strategic goals. These best practices include hints, such as how to group operations to services and what is important to consider regarding their names. When starting with the implementation, further guidelines provide information about how to implement services using a certain technology. While SOA does not dictate any technology usage, in most cases web services are applied as their standardization supports the flexibility and maintainability of the architecture from a technical point of view [5]. In this case, the web services are described using the World Wide Web Consortium (W3C) standards Web Services Description Language (WSDL) [6] and XML Schema Definition (XSD) [7]. Furthermore, in some projects the Service Component Architecture (SCA) [8] standardized by the Organization for the Advancement of Structured Information Standards (OASIS) is applied to describe the component model.

Though both the best practices and patterns from a conceptual point of view and the guidelines from a technical point of view provide valuable information, there is a gap between both approaches. On the one hand, best practices that focus on architectural issues from a conceptual point of view are too abstract to be verified on concrete web service artifacts. On the other hand, the technology-specific guidelines that describe how to implement services using a certain technology are not related to strategic goals which hampers their motivation. As result, for architects and developers who want to design and implement web services that consider existing architectural best practices it is hard to verify that they have done everything correct.

In previous work, we have shown how to close this gap: In [9], we have described how to break architectural best practices down into measurable quality indicators that can be verified on concrete artifacts, such as web services. The quality indicators can be formalized using metrics that enable an objective and repeatable quality analysis. The metrics already provide the first step to evaluate web services systematically. However, for an efficient application in development processes the metrics have to be measured automatically on concrete technology elements of web services, such as WSDL documents and SCA artifacts. For that reason, this article introduces a query-based static analysis (QSA) approach that includes the mapping of metrics and their constituents onto elements of web services implementation artifacts for an automatic execution.

The concept is illustrated using a scenario in the context of automotive manufacturing. In this case, the usage of formalized metrics 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 web services, thus increases the efficiency.

The article is organized as follows: Section II introduces existing best practices and patterns for web services, their formalizations, and their automatic measurement. The scenario is introduced in Section III. In Section IV, the services for the scenario are developed using a quality model, the QSA approach, and our product. Section V concludes this article and introduces future research work.

II. Background

This section describes best practices for the design of services in service-oriented architectures. Furthermore, this work is examined regarding its possibility to be efficiently measured on web services using tools. In addition, work in the context of evaluating software regarding best practices is considered. 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" [10]. The service design phase includes design 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 the design of the entire architecture. In the last years, for services several best practices and patterns have evolved.

In [4] and [11], 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 [12] and Josuttis [13] focus on patterns from a similar point of view. While the guidelines are clearly motivated, their usage in projects similarly 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 [14] and [15]. 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 that is derived from real-world projects is not reflected by the metrics. This is even not possible as Perepletchikov et al. consider an abstract formalization of services. Most of the best practices introduced by Erl and Josuttis refer to elements that are not part of the formalization used by Perepletchikov et al. Furthermore, the abstract formalization is not mapped onto concrete technologies, such as web services. This hampers the measurement of the introduced metrics in real-world projects and requires additional effort.

Similarly to Perepletchikov et al. [14][15], Hirzalla et al. [16] and Choi et al. [17] introduce metrics for services. Also in this work, the metrics are very abstract and cannot be directly applied in projects. Even though they are formalized which reduces interpretation effort, they do not represent best practices as introduced by Erl and Josuttis which hampers their motivation. These metrics should be associated with best practices of real-world projects. A mapping onto concrete web service technologies would enable their application in concrete projects.

To fill this gap, in previous work [9] we created a quality model that combines best practices as introduced by Erl et al. [4][11] with a formalization as used by Perepletchikov et al. [14][15]. The quality model was aligned with the Service oriented architecture Modeling Language (SoaML) [18] as profile for the Unified Modeling Language (UML) [19] that is meant to replace proprietary UML profiles for services, such as the one developed by IBM [20][21][22]. As result of this work, an SOA formalized using SoaML can be checked against wide-spread best practices. The usage of SoaML is explained in [23][24] and a case study that applies the metrics is presented in [25]. However, in most cases web services are created or are already existent without a formalization based on SoaML. Furthermore, some best practices 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 [26], it is shown how service designs based on SoaML can be transformed into web services using 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 in the context of best practices for web services shows that a lot of good work exists, which 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 imprecise to be efficiently measured as they are only textually described, the latter are too academic to be comprehensible understandable and motivated. For that reason, we use the metrics introduced in [9] 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 [26]. As result, metrics are available that can be directly be measured on web services and their development artifacts. However, there is also a mechanism for the measurement itself necessary.

Next, existing work to evaluate software artifacts, such as the described web service artifacts, regarding best practices and patterns is examined.

A typical approach to evaluate implementation artifacts regarding a certain architecture specification is the usage of software reflexion models as shown by Murphy et al. in [27]. This approach is helpful to find differences between two models, mostly a specification and a source code model. By this means, inconsistencies between an architecture specification and its implementation can be identified. However, this approach is not applicable to analyze an architecture regarding best practices as the compared models have to be on the same level. Best practices describe rules that refer to elements of the metamodel. Thus, they are not described on the same level as the source code model.

A more applicable approach is shown by Giesecke et al. in [28]. In this work, architecture styles represent the basis for architecture evaluations. Even though this is the only work of the authors in this context and there is no example described, it can be recognized that the basis for the evaluation is an architecture model that is derived from the source code and has to be described in a certain language. The essential disadvantage of this approach is the limiting metamodel the architecture model bases on. Best practices can refer to many different aspects of an architecture that go beyond components and their dependencies. When creating an architecture model from the source code, all these specifics the best practices refer to have to be available in the architecture metamodel and have to be considered when mapping the source code to the architecture model. Furthermore, especially when considering best practices that are more technology-specific, either the metamodel has to be extended in a way that it represents all these technology specifics or information gets lost and the best practices cannot be verified. Another approach could be to check some best practices on a general architecture model and some other best practices on the source code directly. However, our experience is that this results in further complexity: First, again a mapping mechanism is required to get the architecture model from the source code. And second, to check the technology-specific best practices two approaches are necessary: One to verify the architecture model and one to verify the source code.

We suggest to unify the evaluation methodology to reduce complexity. The consideration of the entire wide range of best practices would result in complex mapping rules and a very complex architecture metamodel. This is exactly the reason why we propose not to derive an abstract architecture model from the source artifacts, such as source code, but directly work on the source artifacts using a query-based approach.

III. SCENARIO

To illustrate the query-based static analysis approach for the evaluation of web services, a scenario from automotive manufacturing is chosen. A service landscape has to be created that supports the manufacturing of cars. A new service has to be provided that offers functionality to initialize the manufacturing of a new automobiles. Meta data about the manufactured automobile are expected to be stored in external systems. Furthermore, the construction system has to be triggered.

The project team consists of two developers and one product and quality manager who coordinates the developers and 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.

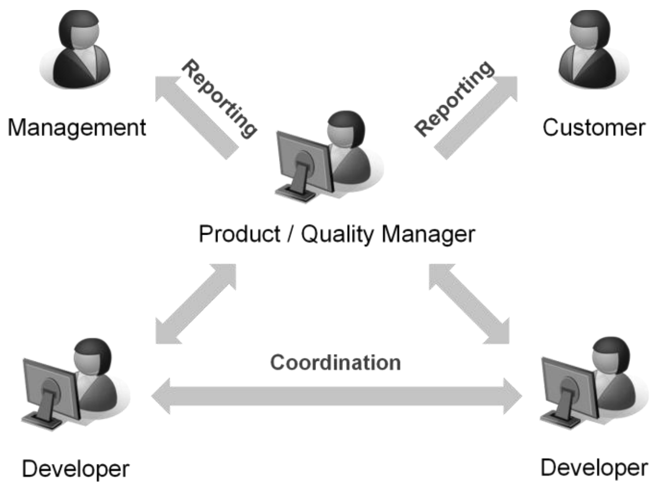


Figure 1. Participants and their relationships.

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 of interest. So it is necessary that developers consider best practices and patterns that support the achievement of a flexible and maintainable architecture. Furthermore, the product and quality manager 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. Furthermore, for all services appropriate web services interfaces using WSDL are developed. 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 applied that uses a different visualization.

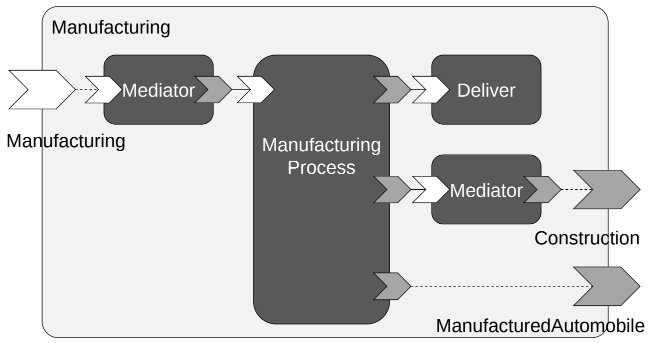


Figure 2. Created SCA composite.

Starting with this SCA composite the product and quality manager determines the quality of the architecture using the approach introduced in the following section. The used WSDL documents are shown later in this article. The results of the quality analysis will help both the product and quality manager and the developers to revise the architecture in a quality-oriented way.

IV. Query-based Static Analysis of Web Services in Service-oriented Architectures

In this section, the query-based static analysis approach is applied to automatically analyze web services in service-oriented architectures regarding best practices. For that purpose, first the applied quality model for web services in service-oriented architectures is shown. As this quality model describes the quality of web services only on a conceptual lever, a mapping onto web service artifacts, such as WSDL documents and SCA artifacts, is described. Finally, based on this mapping the analysis is automated using the query-based static analysis approach.

A. Quality Model for Web Services

To determine the quality of software, one approach is to refine the term quality until it can be measured. A widespread quality model methodology is Factor, Criteria, Metric (FCM) introduced by McCall et al. in [29]. 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 web services in 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 [9], a quality model has been created that enables the measurement or at least systematic evaluation of services regarding best practices and patterns that have evolved as important for service-oriented architectures. The quality model is shown in Fig. 3 in a tree structure.

In recent work, 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 application. As in the scenario other technologies, in particular WSDL, XSD, and SCA are used, the metrics introduced in [9] cannot be applied without additional effort. However, in [26], a mapping between SoaML and web service technologies is described. The combination of this work enables the mapping of metrics onto web services so that they can be directly applied. This mapping is shown next and constitutes the basis to implement the query-based analysis approach.

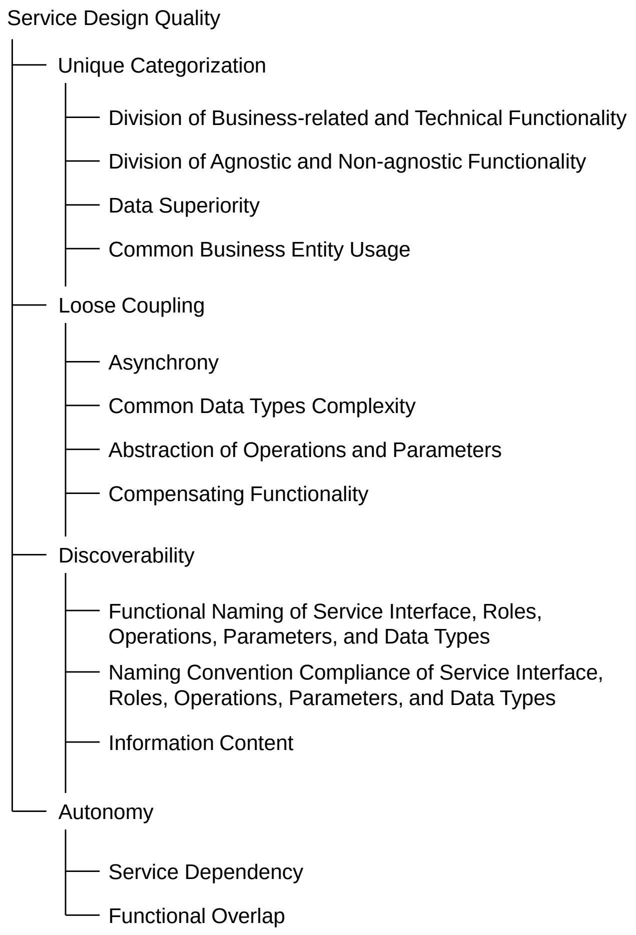

Service Design Quality
Unique Categorization
Division of Business-related and Technical Functionality
Division of Agnostic and Non-agnostic Functionality
Data Superiority
Common Business Entity Usage
Loose Coupling
Asynchrony
Common Data Types Complexity
Abstraction of Operations and Parameters
Compensating Functionality
Discoverability
Functional Naming of Service Interface, Roles, Operations, Parameters, and Data Types
Naming Convention Compliance of Service Interface, Roles, Operations, Parameters, and Data Types
Information Content
Autonomy
Service Dependency
Functional Overlap

Figure 3. Quality model for web services in service-oriented architecture.

B. Application on Web Service Implementation Artifacts

According to Gebhart et al. [9] 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.

To apply these quality sub-characteristics on concrete web service implementation artifacts, a mapping of the quality indicators and their metrics is required first. 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 [9][30][31]. 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: The background of this metric is that generic functionality should be seperated from specific one so that changes regarding the specific operations do not affect the highly reused ones. It has its origin in the patterns described by Erl [4].

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

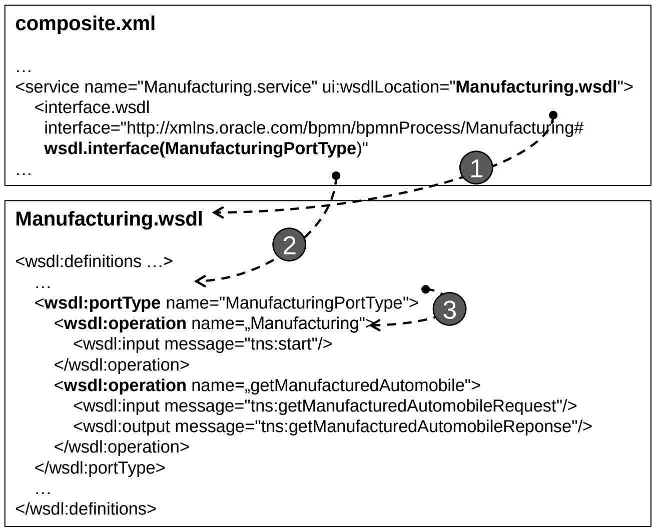
To apply this metric for the scenario, the functions and variables have to be mapped onto elements within XSD, WSDL, and SCA. Table I shows a brief introduction of the element and afterwards a mapping. This mapping specifies where to find this information.

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

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. A value between 0 and 1 means that agnostic and non-agnostic functionality has been mixed. In this case, the participants should revise the design. For example the provided operations can be separated into several services.

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. 4 shows the proceeding.



composite.xml

```
…
<service name="Manufacturing.service" ui:wsdlLocation="Manufacturing.wsdl">
  <interface.wsdl
    interface="http://xmlns.oracle.com/bpmn/bpmnProcess/Manufacturing#
    wsdl.interface(ManufacturingPortType)"
…
```

Manufacturing.wsdl

```
<wsdl:definitions …>
  …
  <wsdl:portType name="ManufacturingPortType">
    <wsdl:operation name=„Manufacturing">
      <wsdl:input message="tns:start"/>
    </wsdl:operation>
    <wsdl:operation name=„getManufacturedAutomobile">
      <wsdl:input message="tns:getManufacturedAutomobileRequest"/>
      <wsdl:output message="tns:getManufacturedAutomobileReponse"/>
    </wsdl:operation>
  </wsdl:portType>
  …
</wsdl:definitions>
```

Figure 4. Determination of DANF metric.

After the relevant operations have been identified, the product and 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.

$$DBTF(s) = \frac{\left|BF\left(O\left(RI\left(SI(s)\right)\right)\right)\right|}{\left|O\left(RI\left(SI(s)\right)\right)\right|} \qquad (2)$$

TABLE II. VARIABLES AND FUNCTIONS USED FOR DANF

Element	Description and Mapping
DBTF	Division of Business-related and Technical Functionality
BF(o)	Business-related Functionality: operations providing business-related 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.

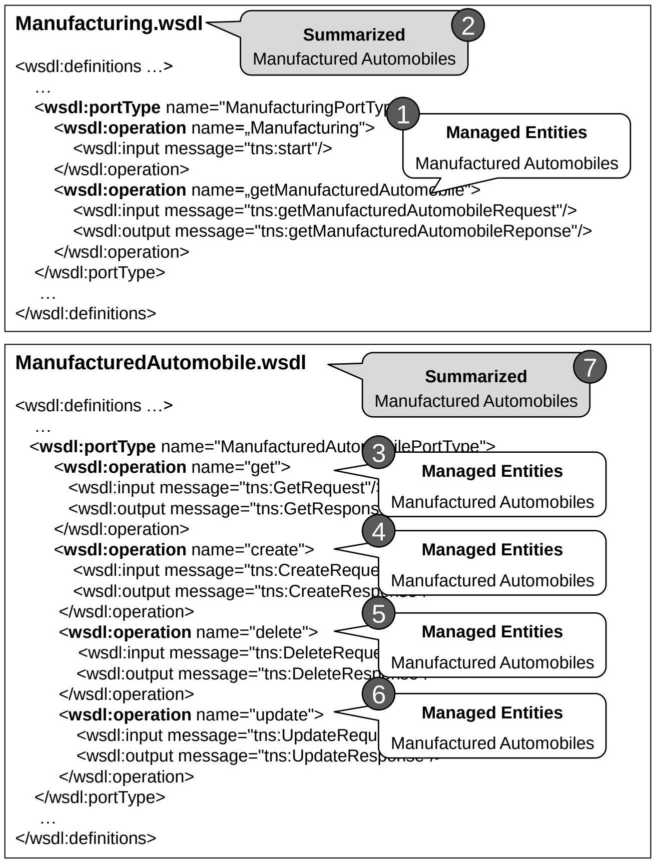
Also in this case, a value of 0 or 1 is desired. These values represent the case that a service provides either only business-related or only technical functionality. In our scenario, all functionality is business-related.

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 III.

$$DS(s) = 1 - \frac{\left| \begin{matrix} ME\Big(O\big(RI(SI(s))\big)\Big) \cap \\ ME\Big(O\big(RI(SI((ALL_S \setminus s)))\big)\Big) \end{matrix} \right|}{\left| ME\Big(O\big(RI(SI(s))\big)\Big) \right|} \tag{3}$$

TABLE III. 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.


Manufacturing.wsdl
Summarized Manufactured Automobiles
2
1
Managed Entities Manufactured Automobiles
ManufacturedAutomobile.wsdl
Summarized Manufactured Automobiles
7
3
Managed Entities Manufactured Automobiles
4
Managed Entities Manufactured Automobiles
5
Managed Entities Manufactured Automobiles
6
Managed Entities Manufactured Automobiles

```
<wsdl:definitions …>
  …
  <wsdl:portType name="ManufacturingPortTyp
    <wsdl:operation name=„Manufacturing">
      <wsdl:input message="tns:start"/>
    </wsdl:operation>
    <wsdl:operation name=„getManufacturedAutomobile">
      <wsdl:input message="tns:getManufacturedAutomobileRequest"/>
      <wsdl:output message="tns:getManufacturedAutomobileReponse"/>
    </wsdl:operation>
  </wsdl:portType>
  …
</wsdl:definitions>
```

```
<wsdl:definitions …>
  …
  <wsdl:portType name="ManufacturedAutomobilePortType">
    <wsdl:operation name="get">
      <wsdl:input message="tns:GetRequest"/>
      <wsdl:output message="tns:GetRespons
    </wsdl:operation>
    <wsdl:operation name="create">
      <wsdl:input message="tns:CreateReque
      <wsdl:output message="tns:CreateResponse"/>
    </wsdl:operation>
    <wsdl:operation name="delete">
      <wsdl:input message="tns:DeleteReque
      <wsdl:output message="tns:DeleteResponse"/>
    </wsdl:operation>
    <wsdl:operation name="update">
      <wsdl:input message="tns:UpdateRequ
      <wsdl:output message="tns:UpdateResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  …
</wsdl:definitions>
```

Figure 5. 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 developed services, i.e., the ManufacturedAutomobile service in this case. Afterwards, the product and quality manager 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. 5 illustrates the proceeding for the Manufacturing service. According to this figure all entities managed by the Manufacturing service are not exclusively managed. The ManufacturedAutomobile service that corresponds to an entity service [1][4] 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.

$$CEU(s) = \frac{\left| OUE\left(\begin{matrix} O\Big(RI\big(SI(s)\big)\Big), \\ CMP\left(\begin{matrix} O\Big(RI\big(SI(s)\big)\Big), MOUE\Big(O\Big(RI\big(SI(s)\big)\Big)\Big), \\ UE\Big(O\Big(RI\big(SI(s)\big)\Big)\Big) \end{matrix} \right) \end{matrix} \right) \right|}{\left| O\Big(RI\big(SI(s)\big)\Big) \right|} \tag{4}$$

TABLE IV. 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 III.

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 ManufacturedAutomobile 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 ManufacturedAutomobile service would also manage another entity, the CEU metric would return a suboptimal value.

C. Query-Based Static Analysis Approach

The previous section illustrated the mapping of conceptual metrics onto web service artifacts. In this section, the QSA approach is introduced that is afterwards applied to automate the web service evaluation.

As mentioned in the Background section, one central disadvantage of existing architecture evaluation approaches is the usage of an architecture model that is derived from the source code. This architecture model is a representation of a source, however it is limited to the elements defined in a metamodel. This means that either information is lost or that the metamodel and the mapping mechanism has to be enhanced in a way that all necessary information is considered. However, as best practices cover a wide range of information this approach is not practicable. As some best practices refer to technology specifics, the metamodel and the mapping mechanism would escalate. The approach is illustrated in Fig. 6.

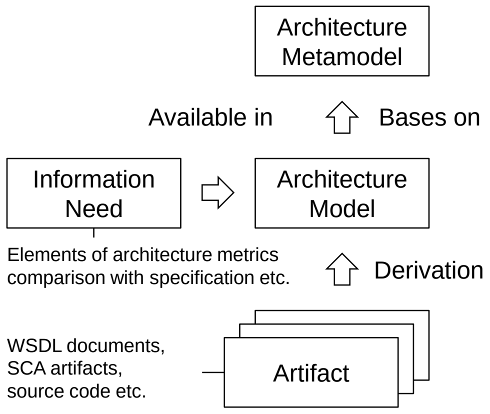

Architecture Metamodel
Available in
Bases on
Information Need
Architecture Model
Elements of architecture metrics comparison with specification etc.
Derivation
WSDL documents, SCA artifacts, source code etc.
Artifact

Figure 6. Usual architecture evaluation approaches

The alternative approach as proposed in this article is to query necessary information from artifacts when they are needed. This means that there is no architecture model derived from the source code. Instead we use mechanisms to find information directly in the web service artifacts when they are required. Fig. 7 shows the query-based approach.

Similar to the architecture evaluation approach in Fig. 6, we start with an information need. For example, metrics or their elements, such as the number of available services or the operations of a certain operations, are expected to be determined. Also, the comparison of the architecture to a certain specification might be an information need. Compared to the approach in Fig. 6, the query-based approach does not work on an architecture model that is derived from the artifacts, such as WSDL documents, SCA artifacts, or source code. Instead, a central component, in this case called Analyzer, receives the information need and tries to satisfy it. For that, the Analyzer component has a repository of so-called information providers.

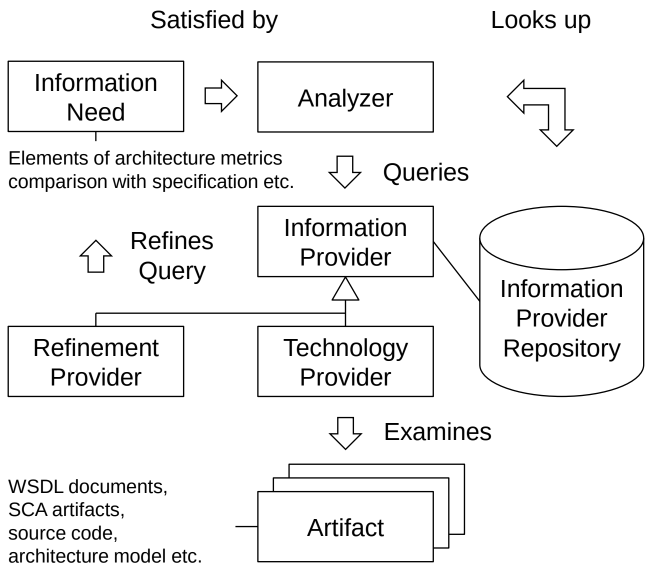

Satisfied by
Looks up
Information Need
Analyzer
Elements of architecture metrics comparison with specification etc.
Queries
Refines Query
Information Provider
Information Provider Repository
Refinement Provider
Technology Provider
Examines
WSDL documents, SCA artifacts, source code, architecture model etc.
Artifact

Figure 7. Query-based analysis approach.

An information provider is able to receive a certain query and answer it depending on the expected result. For example, if the information provider is requested to return the number of services within a service-oriented architecture as Integer, the information provider is able to understand this information need and return it in the expected format. We distinguish between Technology Providers and Refinement Providers. Technology Providers are able to answer a query on basis of information contained in certain artifacts, such as WSDL documents, SCA artifacts, source code, models, databases, and so on. For example, if the query is to get all provided services in a service-oriented architecture, a WSDL Technology Provider and a SCA Technology Provider will be called. These providers access WSDL documents and SCA artifacts within the architecture and determine the services in the architecture and return them to the Analyzer component. A Refinement Provider receives a query, refines it into several information needs, and creates the result for the original query depending on the results for these refined information needs. For example, if the query is to get all operations within the service-oriented architecture, a SOA Refinement Provider refines this query into 1) an information need to get all services in the architecture and 2) to get the operations for each of these services. The refined information needs are answered in the same way, i.e., they are satisfied by the Analyzer component that uses information providers. The SOA Refinement Provider uses these results to generate the result for the original query and returns it to the analyzer.

Besides the flexibility and reduced complexity, another advantage of this approach is that for each information provider a different implementation language can be chosen. When deriving a central architecture model from implementation artifacts, often the same transformation language has to be used for all artifacts. In our approach, an information provider that is expected to analyze XML artifacts, such as WSDL documents and SCA artifacts, can be implemented in another language than an information provider that is expected to work on databases or on hardware information, such as card readers. This has the big advantage that for every purpose the most suitable language can be chosen and that all information that can be requested by any technology can be actually requested and reused in the analysis. Furthermore, languages most people are used to, such as Java and C#, can be applied and no proprietary languages have to be learned. In our case, most of the information providers are directly implemented using Java. This reduces the development time for new information providers and increases the adoption of this approach in real-world projects.

Furthermore, when there is a new artifact that is expected to be considered during the analysis only a new information provider has to be added and the existing logic does not have to be changed. For example, when in the future besides WSDL and SCA also the Business Process Model and Notation (BPMN) 2.0 language is expected to be considered, we only have to add a new technology provider for BPMN that is able to answer queries related to this language. This increases the flexibility and maintainability of this analysis methodology.

D. Query-Based Static Analysis for Evaluation Automation

In this section, the QSA approach is used to automatically evaluate web service artifacts. For that purpose, the mapping knowledge introduced before is implemented as information providers. As illustrated in Fig. 7, an information need can be a metric or elements of a metric. This information need is then sent to the Analyzer component so that it can be satisfied. For that purpose, the Analyzer component uses one or several information providers.

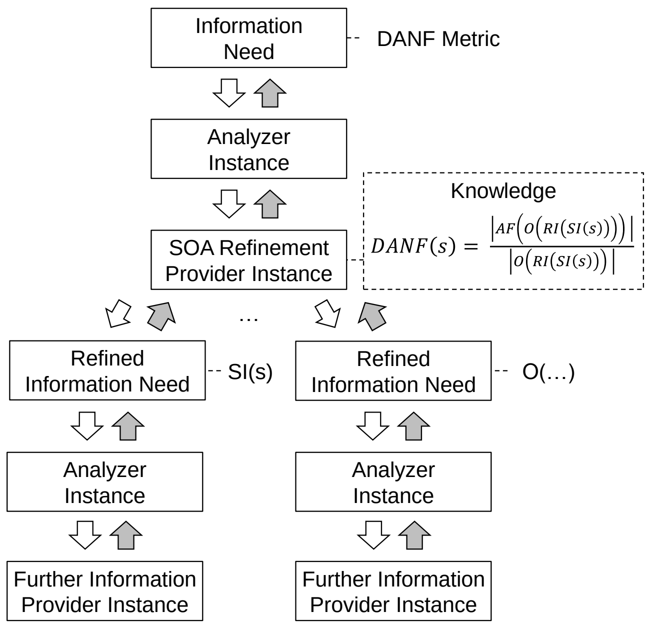

Information Need
DANF Metric
Analyzer Instance
Knowledge
$DANF(s) = \frac{|AF(O(RI(SI(s))))|}{|O(RI(SI(s)))|}$
SOA Refinement Provider Instance
…
Refined Information Need
SI(s)
Refined Information Need
O(…)
Analyzer Instance
Analyzer Instance
Further Information Provider Instance
Further Information Provider Instance

Figure 8. Interaction between Analyzer and SOA refinement provider.

These are able to answer the query. In the previous section, the metrics were introduced and mapped onto web service artifacts. This mapping has shown that 1) the metrics mostly consist of several elements that have to be requested separately and 2) the metrics refer to information kept in WSDL documents and SCA artifacts. Thus, to automate the metrics the following information providers are required:

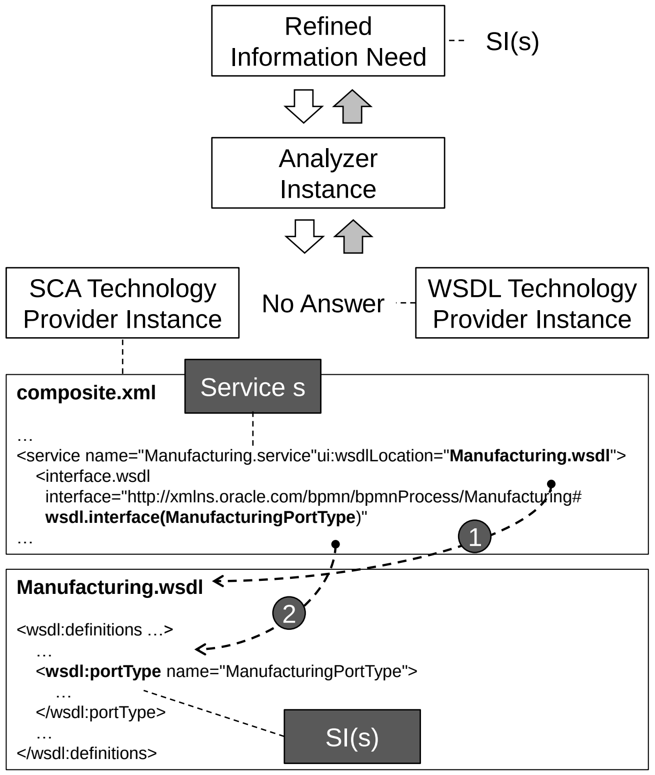
1) SOA Refinement Provider: The SOA Refinement Provider contains the metrics and describes their refinement. For example, the SOA Refinement Provider knows how to calculate the value*s* for *the* DANF metric. It breaks this metric down into the *metric* elements, requests their result from the Analyzer component, and generates the result for the original query. The interaction between the SOA Refinement Provider and the Analyzer component is shown in Fig. 8.

WSDL Technology Provider: The WSDL Technology Provider examines WSDL artifacts regarding certain information needs. For example, when the information need is to get all services within the architecture, the WSDL Technology Provider checks all WSDL files in the architecture and examines them regarding the service XML element. Aftwards, an element as representer for this service is returned to the Analyzer component. In our case, the WSDL Technology Provider is implemented using Java as the Analyzer component is implemented in Java too. However, in our implementation any other language based on the Java runtime can be chosen. It is also possible to switch from operation calls within the Java Virtual Machine to external web service calls etc. In this case, any other language would be possible too. Fig. 9 illustrates how the WSDL Technology Provider interacts with the Analyzer component.

2) SCA Technology Provider: Similar to the WSDL Technology Provider, the SCA Technology Provider examines SCA artifacts, such as SCA composites. This means that when the Analyzer component needs to satisfy an information need, such as the available services or the service interfaces for a certain service, the SCA Technology Provider examines the SCA artifacts regarding this information. Fig. 9 shows how this information provider works and how it interacts with the Analyzer component.

It is important to mention that several information providers can answer the same type of query. For example, both WSDL documents and SCA artifacts can provide information about available services in the architecture. Also, BPMN processes can provide information about available processes and provided services. So when BPMN is used, also this information has to be considered.

The QSA approach allows to query information independent from how it can be answered. The information provider simply receive the query and try to answer it based on their knowledge. Afterwards, they return the result if possible. The Analyzer component however is responsible to merge the results. So when several available services are returned, both from the SCA Technology Provider and the WSDL Technology Provider, the Analyzer component tries to merge the result so that a holistic view is guaranteed.



```
composite.xml
…
<service name="Manufacturing.service"ui:wsdlLocation="Manufacturing.wsdl">
  <interface.wsdl
   interface="http://xmlns.oracle.com/bpmn/bpmnProcess/Manufacturing#
   wsdl.interface(ManufacturingPortType)"
…
```

```
Manufacturing.wsdl
<wsdl:definitions …>
  …
  <wsdl:portType name="ManufacturingPortType">
    …
  </wsdl:portType>
  …
</wsdl:definitions>
```

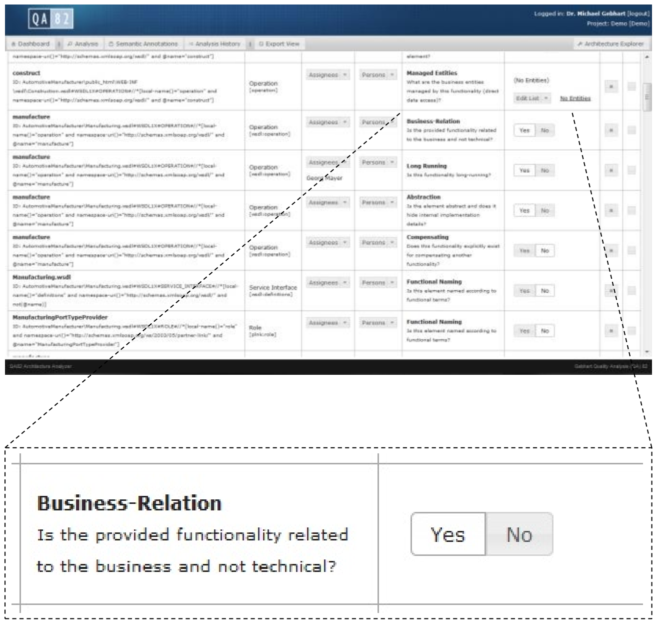
Figure 9. Interaction between Analyzer and technology providers.

E. Manual Information

In Section IV.B the mapping of the conceptual metrics onto web service artifacts has been explained. As part of this mapping it was shown that there is some information that cannot be found in the implementation. For example: What about the business-relation of a service operation? Or what about its agnosticity? There is some information that cannot be determined automatically on basis of existing artifacts. To solve this issue, we have created an additional information provider, the Custom Function Results Provider that represents manual information that can be added by experts. For example, when it is not known if a certain operation is business-related, this knowledge can be added by experts and then it will be considered during the analysis process.

F. Tool Support

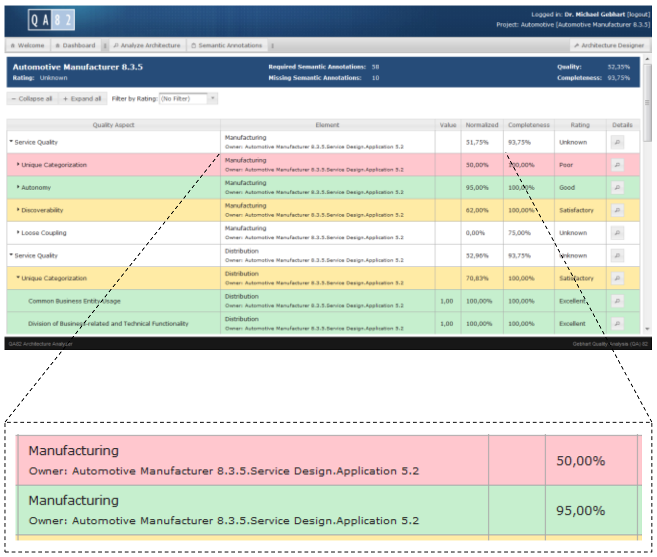
Based on these concepts, we have developed a product that realizes the QSA approach and enables the evaluation of web services regarding the quality model introduced in [9]. The QA82 Analyzer is a generic quality management platform that implements these concepts. Based on the QA82 Analyzer we have developed the QA82 SOA Compliance Center, which implements the SOA quality model introduced before and provides the described information providers. As result, the product and quality manager can automatically perform quality analyses of the developed web services. To demonstrate the behavior of the application we have used it to analyze the introduced scenario. Fig. 10 shows how knowledge can be added by experts.



Business-Relation
Is the provided functionality related to the business and not technical?

Figure 10. Questions in QA82 Analyzer to add expert knowledge.

The QA82 Analyzer or the QA82 SOA Compliance Center creates questions for every information that is not available by any information provider. These questions can be answered by experts and the answer is stored in an internal storage, the Custom Function Result Storage. The Custom Function Result Provider will request this information when it is necessary so that it can be included in the next analysis. In Fig. 10, the expert is asked if the operation "manufacture" is business-related or not. The expert can answer this question by selecting the button "Yes" or the button "No".



Manufacturing Owner: Automotive Manufacturer 8.3.5.Service Design.Application 5.2		50,00%
Manufacturing Owner: Automotive Manufacturer 8.3.5.Service Design.Application 5.2		95,00%

Figure 11. Analysis result in detail.

knowledge can be added by experts.

When an analysis has been performed, it is archived. The user can take a look at the analysis and the entire calculation trace. The analysis represents the prior defined quality model. Fig. 11 shows an analysis result in detail. According to Fig. 11, the Manufacturing service has a design quality of 50%. This means, that the best practices described by the quality model are only partially fulfilled. In our application, the user can select this entry to get more information about how this value has been measured and how it can be improved.

Finally, a quality dashboard shows the recent analysis results and some further information, such as the distribution of certain quality attributes and the number of open questions. Fig. 12 shows the quality of time that is displayed as part of the dashboard.

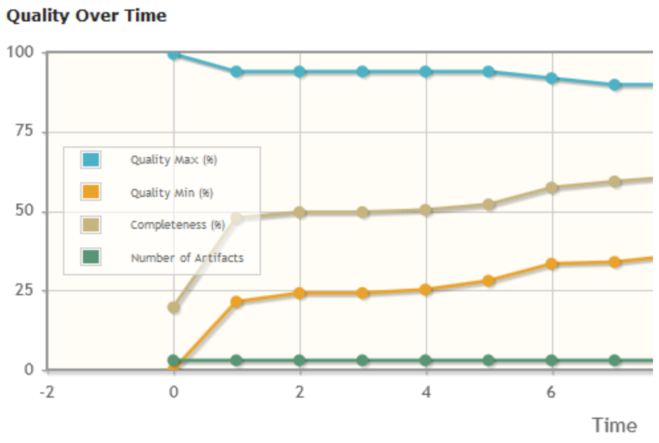


Figure 12. Quality results over time as part of the dashboard.

To sum up, the QA82 Analyzer or the QA82 SOA Compliance Center implement the concepts described in this article and enable their automation. Different views, such as the analysis details view and the dashboard help the user, i.e., the product and quality manager to calculate the quality and to ensure that the developed artifacts comply with wide-spread best practices.

Furthermore, our product is not only for managers or architects. Also developers can now directly take a look at the quality of their artifacts from a design perspective. I.e., they get an impression about how they have considered certain best practices and patterns. If the quality is not optimal, they can interact independently and without being informed by any responsible person. This makes them aware of quality aspects and increases the development speed.

G. *Integration into Scenario*

Back in our scenario, the quality manager can use the results of the QA82 SOA Compliance Center to inform developers about the design weaknesses. The usage of these metrics in a quality-oriented service design process is illustrated in [32]. Furthermore, as described before, developers can already independently get an insight into the quality of their artifacts.

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 ManufacturedAutomobile 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 ManufacturedAutomobile service. Service consumers using this operation should switch to the ManufacturedAutomobile Service. This and further information are given to all participating persons so that they can improve the artifacts with quality goals in mind.

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, the manager 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 web services in service-oriented architectures regarding wide-spread best practices. 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. For an automation of the web service evaluation the quality-based static analysis approach was introduced. Compared to existing architecture evaluation approaches, the query-based static analysis approach does not derive an abstract architecture model but works directly on developed artifacts. Finally, a software product was shown that implements the approach and enables an automatic quality analysis of developed web service implementation artifacts regarding wide-spread best practices.

To demonstrate the approach 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. Examples for reports are the current quality, the characteristic of certain quality attributes, such as the coupling or autonomy, the number of open questions, and the quality over time. The reports help to prove the high quality and to justify investments in additional quality assurance projects.

Furthermore, developers can perform quality analyses by their own. The metrics reduce the additional effort to interpret the textual descriptions. They directly refer to concrete elements within the used technologies.

As part of our research work, we have created a mapping for all metrics introduced in [9]. We also implemented this quality model as part of the QA82 Analyzer and QA82 SOA Compliance Center [33]. Through this, both product and quality managers and developers can automatically measure their web services regarding the quality model. This further increases the efficiency of the quality assurance process and makes the entire quality topic transparent. All participants are made aware of what quality means and how it can be influenced by developed artifacts. Furthermore, all participants can directly get an insight into the current design quality of developed artifacts.

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 the Representational State Transfer (REST) paradigm 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, such as Java artifacts based on JAX-RS. Second, in collaboration with partners we work on a quality model in the context of 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 [34]. 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 [35] or XQuery [36] 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] M. Gebhart, “Measuring design quality of service-oriented architectures based on web services,” Eighth International Conference on Software Engineering Advances (ICSEA 2013), Venice, Italy, October 2013, pp. 504-509.

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

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

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

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

[6] W3C, “Web Services Description Language (WSDL)”, Version 1.1, 2001.

[7] W3C, “XML Schema Part 0: Primer Second Edition”, 2004.

[8] 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]

[9] 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.

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

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

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

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

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

[15] 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.

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

[17] 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.

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

[19] OMG, “Unified modeling language (UML), superstructure,” Version 2.2, 2009.

[20] 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]

[21] 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.

[22] 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]

[23] 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]

[24] 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.

[25] 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.

[26] 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.

[27] G. C. Murphy, D. Notkin, and K. J. Sullivan, “Software Reflexion Models: Bridging the Gap between Design and Implementation,” IEEE Trans. Softw. Eng., vol. 27, 2001.

[28] S. Giesecke, M. Gottschalk, and W. Hasselbring, “The ArchMapper Approach to Architectural Conformance Checks: An Eclipse-based Tool for Style-oriented Architecture to Code Mappings,” 2012, pp. 71–80..

[29] J. A. McCall, P. K. Richards, and G. F. Walters, “Factors in software quality,” 1977.

[30] 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.

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

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

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

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

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

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