Quality-Oriented Design of Software Services in Geographical Information Systems

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

Suad Sejdovic
Campana & Schott
Stuttgart, Germany
suad.sejdovic@campana-schott.com

***Abstract*—Distributed information, such as sensor information, increasingly constitutes the basis for geographical information systems. For that reason, these systems are designed according to services-oriented design principles, which means that they require software services returning necessary information and provide higher-value ones. These services are expected to follow quality attributes, such as loose coupling and autonomy, which have been identified as important in the context of service-oriented architectures. For measuring these quality attributes, metrics have been derived that enable quantifications. They can be directly evaluated on basis of formalized service designs and indicate the extent of quality attributes. This article shows the application of these service design metrics for a quality-oriented design of services in geographical information systems. The considered system is part of the Personalized Environmental Service Configuration and Delivery Orchestration project of the European Commission.**

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

I. INTRODUCTION

A geographic information system (GIS) is a computer system, which is used for capturing, storing, analyzing and also displaying geospatial data, whereas geospatial data is data that is describing characteristics of spatial features on the Earth's surface which are referenced to by a location [1].

In order to access this data in a standardized manner, it is provided by means of software services that base on standardized protocols and interface description languages, such as Simple Object Access Protocol (SOAP) over HyperText Transfer Protocol (HTTP) and Web Services Description Language (WSDL) [2]. Besides the usage of services, the information systems themselves are often required to be integrated in a more complex architecture. This is why the systems are additionally supposed to not only invoke but also to provide services that enable accessing higher-value functionality. As result, geographical information systems apply services as architecture paradigm and follow service-oriented design principles.

In the context of service-oriented architectures (SOA) several quality attributes have been identified as important depending on higher-level quality goals that are associated with the system. In order to easily switch between several data sources, for geographical systems a very important aspect is to build a flexible and maintainable architecture. These higher-level quality goals can be broken down into more fine-grained quality attributes, such as loose coupling and autonomy, affecting the building blocks of the architecture, in this case the services. Accordingly, the used services in the context of the geographical information system have to be designed in a way that these quality attributes can be fulfilled. The design of services can be confined to a service interface and a service component. Whilst the service interface describes the externally visible access point to the service, the service component focuses on the internal behavior of the service itself. In order to formalize the design of a service, the Service oriented architecture Modeling Language (SoaML) as profile for the Unified Modeling Language (UML) can be applied [3]. It represents an emerging standard to describe service designs in a standardized manner and gains increasing tool support, which leads to an increasing acceptance in development processes.

For measuring the quality of software, metrics can be used as quantified values of quality indicators [4], [5], [6], [7], [8]. In the context of service-oriented architectures and in particular for the design of services, Gebhart et al. identified metrics especially evaluating service designs based on the Service oriented architecture Modeling Languages (SoaML) [4]. These ones refer to model elements available within this Unified Modeling Language (UML) profile, which simplifies the evaluation of formalized service designs. Compared to other non-formalized quality indicators, such as textual descriptions, or metrics not designed for SoaML, the usage of these SoaML-oriented metrics avoids interpretation effort with possibly faulty interpretation and accordingly faulty measurement. Finally, the metrics can be automatically calculated as implemented by the QA82 Architecture Analyzer [9].

In order to demonstrate the quality-oriented design of services based on these metrics, this article considers the design of a geographical information system in a service-oriented manner [1]. This means that metrics especially designed for service designs based on SoaML are applied for designing services of a geographical information system with certain quality attributes fulfilled. In this article, the project Personalized Environmental Service Configuration and Delivery Orchestration (PESCaDO) is considered [10].

The article is organized as follows: Section II introduces the service design process, the formalization of service designs using SoaML, and wide-spread quality attributes. The scenario is introduced in Section III and in Section IV the services are designed. Section V concludes this article and introduces future research work.

II. BACKGROUND

This section describes fundamentals for the article. This includes especially the understanding of service designs in the context of software service engineering and its formalization using SoaML.

A. Service Design Process

The service design phase is a primary ingredient of the software service engineering that can be understood as the "discipline for development and maintenance of SOA-enabled applications" [11]. The central purpose of the service design phase is to create a formalized draft of services, so-called service designs, before implementing them. This enables the adaptation and optimization of the entire services architecture without cost-intensive source code changes. That is why analyses of the designs regarding quality attributes, such as loose coupling, are required to be performed within the service design phase. In [12], Gebhart introduces a service design process reusing existing work of Erl, IBM et al. [13], [14], [15], [16], [17], [18] and describes necessary steps within the service design phase for fulfilling this requirement. Figure 1 illustrates this process.

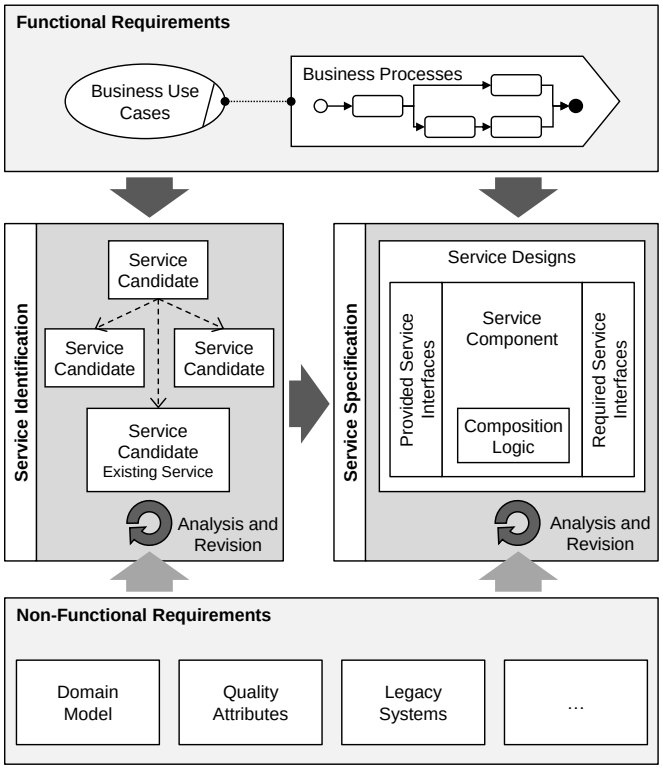

Figure 1. Quality-oriented service design process.

The service design process is a combination of systematic derivations and subsequent analyses and revisions. The systematic derivation especially considers the fulfillment of functional requirements that have been identified within the requirements analysis phase: In a first step, the functional requirements are transferred into so-called service candidates. These represent preliminary services that are not fully specified yet [13]. Especially, when existing services have to be taken into account, there is no necessity to specify new service designs. Instead, the existing specifications can be reused. Otherwise, the service candidates are transferred into elements of service designs. For example, for each service candidate a service interface and implementing service component is created.

The iterative analysis and revision focuses on the fulfillment of non-functional requirements, such as quality attributes. Within each iteration first the current state is analyzed regarding non-functional requirements. For example, the quality attributes are determined using appropriate metrics as demonstrated by Gebhart et al. in [19]. Afterwards, the artifacts are revised for improving the quality attributes or other non-functional requirements. As a result, service designs are created that both fulfill functional requirements that have been determined within the requirements analysis phase and non-functional ones, such as loose coupling, that support higher-level quality goals.

The created service designs can be used to derive web service implementation artifacts in a model-driven way as introduced by Hoyer et al. in [20] and Gebhart et al. in [21].

B. Service Design Formalization

For formalizing a service design, in this article SoaML is applied [3]. In comparison to other proprietary languages, such as the UML Profile for Software Services developed by IBM [22], SoaML is a profile for UML [23] and a metamodel standardized by the Object Management Group (OMG). It provides elements necessary to describe service-oriented architectures and its building blocks, the services. In the meanwhile, SoaML is an emergent standard adopted by several tool vendors. Even IBM has replaced its proprietary UML profile with SoaML [24]. In this article SoaML is applied as UML profile.

In order to model service designs with SoaML, necessary elements of the profile have to be identified. This article uses the elements as introduced by Gebhart et al. in [25]. The service design formalization consists of both the formalization of service candidates and service designs. Thus, for both sub-phases of the service design phase the adequate formalization has to be determined.

According to Erl [26], a service candidate represents a preliminary service on a high level of abstraction. During this phase, only possible operations, called operation candidates, service candidates as grouping of these capabilities, and dependencies between service candidates are determined. In SoaML the Capability element exists, which corresponds to this understanding. The following table shows the mapping of service candidate elements on a conceptual level onto elements within SoaML.

TABLE I. MAPPING BETWEEN SERVICE CANDIDATE ELEMENTS AND SOAML

Service Candidate Element	SoaML Element
Service Candidate	Capability (UML class that is stereotyped with "Capability")
Operation Candidate	Operation within a Capability element
Dependency	Usage Dependency between Capability elements

This table demonstrates that there is a one-to-one mapping between service candidate elements and elements within SoaML possible. Figure 2 illustrates the modeling of service candidates in SoaML.

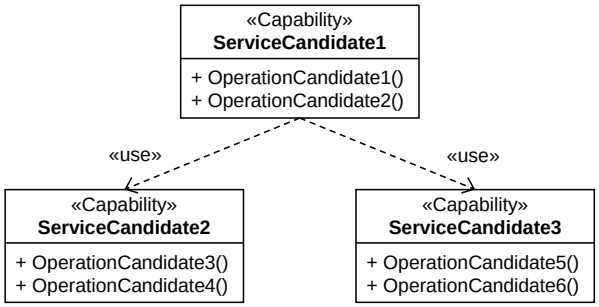

Figure 2. Service candidates in SoaML.

The example includes three service candidates with each of them containing two operation candidates. In this case ServiceCandidate1 requires operations of ServiceCandidate2 and ServiceCandidate3 for fulfilling its functionality.

TABLE II. MAPPING BETWEEN SERVICE DESIGN ELEMENTS AND SOAML

Service Design Element	SoaML Element
Service Interface	ServiceInterface (UML class that is stereotyped with "ServiceInterface")
Provided Operation	Operation within an interface that is realized by the ServiceInterface element
Realized Operation	Operation within an interface that is associated with the ServiceInterface by using a Usage Dependency in UML
Role	Property within the ServiceInterface that is typed by the interface that contains the provided operations or by the interface that contains the required operations
Interaction Protocol	A behavior, such as an UML Activity
Service Component	Participant (UML component that is stereotyped with "Participant")
Provided Service	Service (UML Port that is stereotyped with "Service")
Required Service	Request (UML Port that is stereotyped with "Request")
Internal Behavior	UML Activity that is added as OwnedBehavior to the Participant

A service design represents a full specification of a service [27]. It includes both the service interface as externally visible access point and the service component as realization of the business logic. The service interface has to specify the operations provided by the service and the ones required in order to receive callbacks. Additionally, the participating roles and the interaction protocol have to be determined. Latter describes in which order the operations have to be called for obtaining a valid result.

The service component consists of the services provided by the component and the ones required by the component for fulfilling its functionality. Additionally, the internal behavior is specified by means of a flow of activities that is the composition in case of a composed service. In SoaML there exist elements that directly correspond to the described understanding.

Table II shows the mapping according to [27]. Whilst the original work bases on SoaML in version 1.0 Beta 1, the table was adapted that it corresponds to the standard in the current version 1.0 final.

To illustrate the modeling of service designs, the following figures illustrate the modeling of a service interface and a service component in SoaML. The service interface in Figure 3 assumes two participants interacting, the provider and the consumer. The provider offers two operations the consumer can call. Furthermore, also the consumer has to provide one operation for receiving callbacks. The interaction protocol describes the operation call order for a valid result.

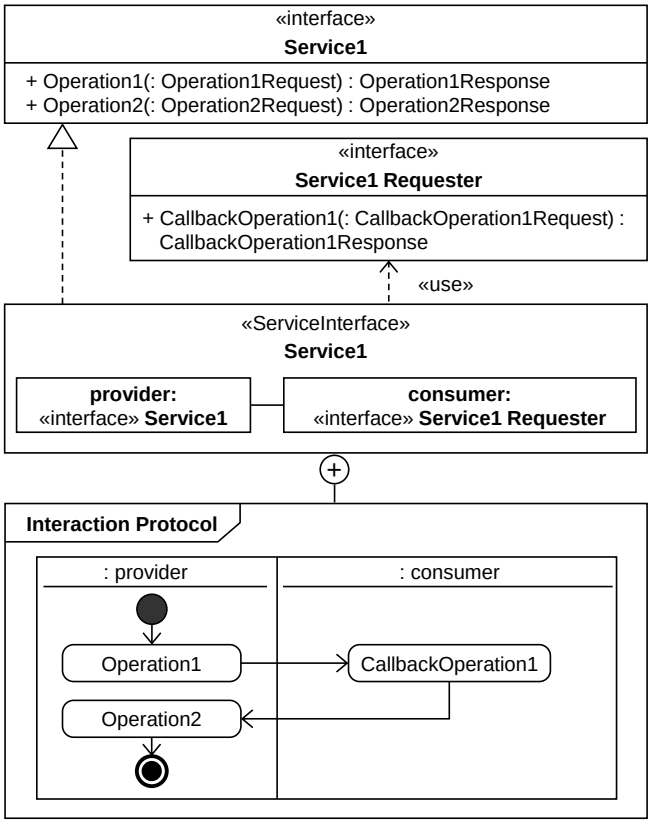

Figure 3. Service interface in SoaML.

The service component illustrated in Figure 4 provides one service and requires one service for fulfilling its functionality. It consists of two internal components, one realizing the composition logic and one implementing further internal logic. The internal behavior can be described by means of an owned behavior in UML. For the sake of simplicity, the internal behavior is not illustrated.

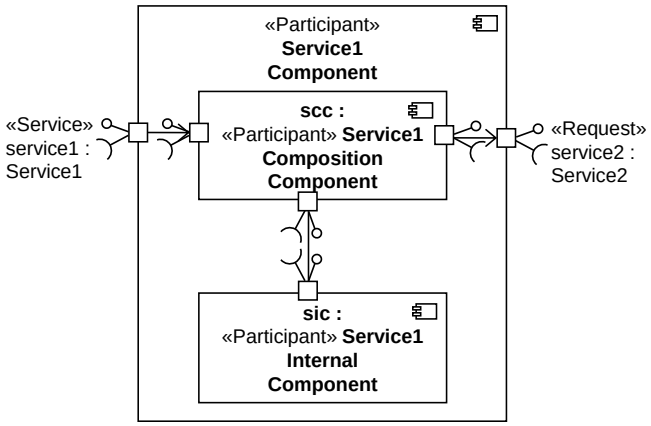

Figure 4. Service component in SoaML.

C. Quality Attributes and Metrics

With the establishment of service-oriented architectures, several strategic goals are associated. Examples are the higher flexibility of the architecture and its easier maintenance [28]. In order to fulfill these strategic goals, quality attributes, such as loose coupling and autonomy, for the building-blocks of the architecture, the services, have been identified. The fulfillment of these quality attributes provides the basis for achieving the strategic goals. As these quality attributes yet are described on an abstract level, they can be further broken down into measurable quality indicators that refer to concrete elements of the services [25]. If these elements are described during design time, the quality indicator can be determined on basis of a service design model. A metric describes the formula for a certain quality indicator and enables its concrete quantification.

In [27], Gebhart et al. identified quality attributes for services that are considered as important in this context. Quality indicators and metrics that enable their determination on basis of formalized service designs are derived in [4]. Based on this work, this article uses the following quality attributes and quality indicators.

1) Unique Categorization: The first quality attribute is the unique categorization, which is comparable to cohesion. According to its description, a service should provide functionality that belongs together. In literature the categorization is mostly described by means of service categories, such as entity, task, and utility services [29]. The quality attribute can be described in detail by means of quality indicators:

First, technical and business-related functionality should be separated up into two services. As technical functionality is used by a different target group than business-related one this helps to maintain the services. This corresponds to the distinction between entity / task services and utility services as introduced by Erl [13], [30].

In order to further increase the maintainability of services also functionality that can be reused in several contexts, i.e., general one or also known as agnostic, should be separated from specific one [26]. This encourages the reuse of general functionality and avoids the influence of changes concerning specific functionality on the general and highly used one. This results in a distinction between entity services that provide general and entity-based operations and task services with mostly specific operations [29]. However, whether functionality is agnostic or not depends on personal estimation.

According to the data superiority, when a service manages a business entity, it should be the only one. This is important to avoid redundant functionality within various services. For the categorization this means that there are no entity services for the same business entity.

Finally, all operations within one service should work on the same business entities. This means that within all operations the same business entity is used. As result, this quality indicator measures whether an entity service is managing only one business entity as expected for an entity service.

2) Discoverability: The best service cannot be leveraged when it cannot be found. That is why discoverability is an important aspect concerning the reusability of services [30]. The discoverability as quality attribute can be refined by the following quality indicators:

First, services and operations should have functional names. Only in this case a service and the contained operations can be found.

In order to increase this aspect, the naming should follow known naming conventions. This can be both the language of the artifacts and the case sensitivity. Also other rules, such as naming operations by using a verb and a noun, are often applied [31].

Finally, the more information is provided the faster a service can be found. This means that especially when modeling services, as most information as possible should be given.

3) Loose Coupling: One of the most often referred quality attribute is loose coupling. It focuses on the dependencies between services, which influences the flexibility and maintainability of services. The following quality indicators that are measurable on service designs can be identified:

In order to support long-running operations, these operations should be provided asynchronously. This means that if an operation provides a long-running functionality an appropriate callback operation should be provided by the consumer and invoked when the operation is finished. This enables the exchange of service provider and consumer during the operation execution.

The dependency between services is also influenced by commonly used data types. Especially when services commonly use complex data types they are dependent as changing one data type requires changes within all using

services. The loose coupling can be measured by the degree to which complex data types are commonly used. Best, services share only simple types. Of course, services can work on the same business entity, such as a Person entity, however the data types should be only copies. A canonical data schema as part of an enterprise service bus should map similar or identical data.

To further increase the independence between services the operations and parameters should be abstract. This means that no technical background information should be necessary to use a service [31]. Also parameters should not include technical data types. This supports the exchange of services as the implementation details are hidden.

If an operation provides functionality resulting in state changes there should be always a compensating undo operation. This again reduces the dependency between services.

4) Autonomy: Finally, the autonomy is one of the considered quality attributes. It also considers the dependency between services but focuses on the ability of a service to be used without other services.

The first quality indicator considers the direct dependency between services, i.e., how many other services are required for fulfilling the own functionality. Basic services are mostly highly autonomous. Composite services instead are composing existing functionality and are thus not autonomously usable.

The second quality indicator focuses on the functional overlap between services. If the functionality of a service overlaps with the one of other services, in most cases the service can also be only used together with the other ones, because in most scenarios functionality of all these services is required. Thus, even though there is no direct dependency between the services, because of the overlapping functionality the service cannot be used solely.

III. SCENARIO

This section introduces the underlying scenario for the exemplary quality-oriented design of software services in this article, the project Personalized Environmental Service Configuration and Delivery Orchestration (PESCaDO) of the European Commission (EC) [10], [32]. The overall goal of the system is to assist human beings in decision-finding under consideration of the personal profile. For example, a user with a pollen allergy and heart problems at very high temperatures wants to know, whether it is advisable for him to book a bicycle tour within the next few months. As described in [1], one special requirement is the semantic support for accessing environmental data. Thus, the system should be capable to identify any related data sources for a requested phenomenon like pollen. That is, the system has to be able to extend a single requested phenomenon by other more specific related ones, like “Birch Pollen”.

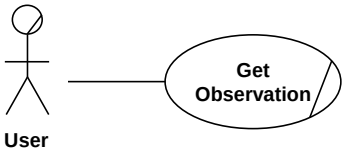

Figure 6. Considered business use case.

Regarding PESCaDO the business use case in Figure 6 can be identified. The business use case describes the requirement to get an observation, which results in a value describing some phenomenon. It is modeled using the adapted notation for use case diagrams by the UML profile for business modeling as introduces by IBM [33], [34]. It is very important to achieve a deep understanding about the business use case, as it is the basic artifact for the identification of service candidates in the service design

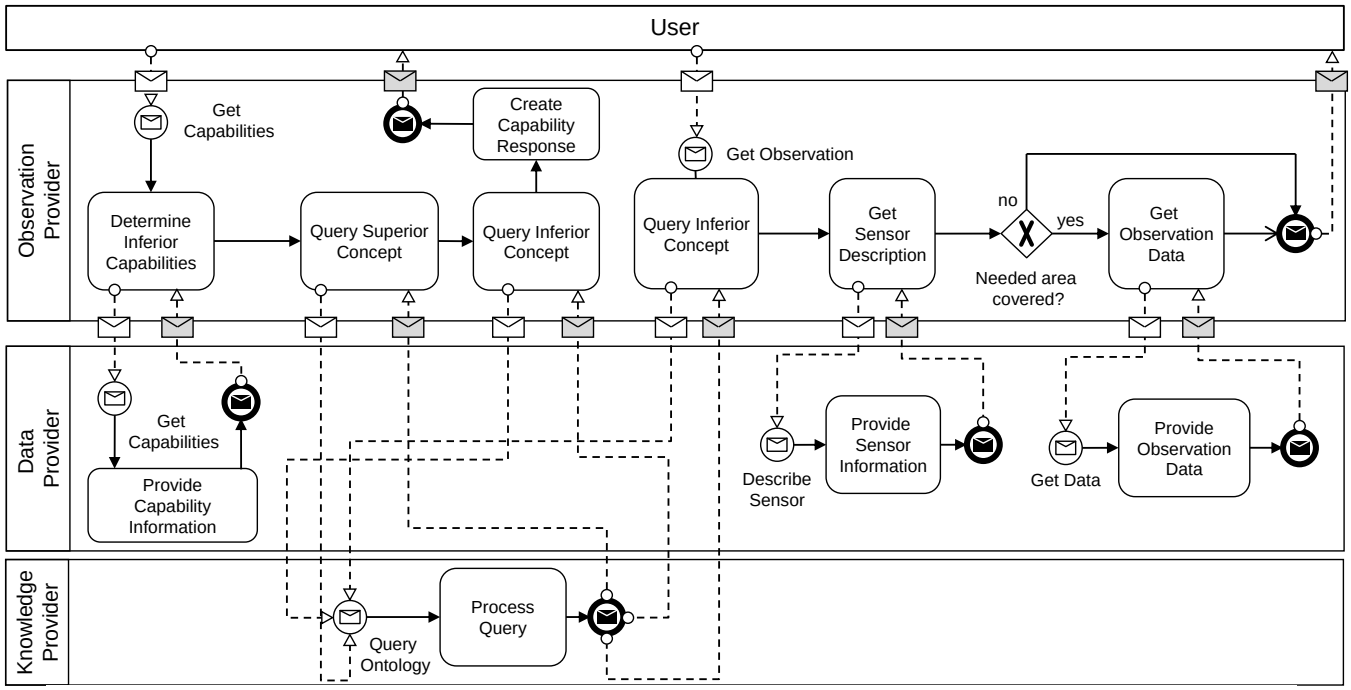

Figure 5. Considered business process.

phase. Thus, knowledge about the internal behavior of the business use case is important within the service design process. This internal behavior can be modeled using the Business Process Model and Notation (BPMN) [35], whereby the modeling concentrates on activities that could be processed automatically.

Figure 6 shows the business process that covers the data access under consideration of the semantic information that is given within the request. The BPMN model consists of four pools. The first pool, labeled with "User" represents the user and is collapsed, as the contained activities are not relevant for automation. The three remaining pools are expanded, as they contain relevant activities for further steps in the development process. Interactions between the different providers are shown by message flows between the pools, whereas the message flows are representing requests and the resulting answers. As the business process is a fundamental artifact, it has to be clear and unambiguous before entering the service design phase.

The observation provider offers two functionalities to the user. The first functionality "Get Capabilities" refers to the capabilities of the observation provider. It represents the self-description capability of the service. By requesting the capabilities the user initiates a procedure, which dynamically generates the information about the capabilities with regard to the underlying data sources. For this, the capabilities of the underlying data provider have to be requested. The data provider also offers the self-describing functionality "Get Capabilities", which returns information about its functionality and the type of data that is available. The information about the available data is returned as a concept referring to the content in an ontology [36], [37]. The returned data can now be used within the observation provider to generate a semantic hierarchy by gaining details about the inferior and superior concepts of the retrieved concept. All the required data is provided by the knowledge provider, which knows all relevant concepts and relationships between them. An important functionality to enable such hierarchies is the functionality "Query Ontology". Through this functionality it is possible to query the ontology and determine the required information. For instance, a data source may contain information about birch pollen and refers as a consequence to a concept called "BirchPollen" within the ontology. The knowledge provider may now generate a hierarchy, which is presenting the position of this concept within a hierarchy, if one exists. For example, the knowledge provider may return a relationship between the concepts "Pollen" and "BirchPollen". Thus, a request for data containing information about the concept "Pollen" should also take into consideration any data about the inferior concept "BirchPollen". This feature supports the requester to find information for more complex concepts, which are referring to composite phenomena, such as air quality.

After retrieving all necessary information, the observation provider processes all retrieved data and generates the requested reply. Thus, the user gets a structured, hierarchical view on the available data. This dynamic approach ensures that users can always get a current view on all available information.

The second functionality of the observation provider, "Get Observation" realizes the data access, whereas the request is addressed to the knowledge provider to determine the inferior concepts of the user input. Thus, all relevant data is found and returned. The next step is to verify that any relevant data is also available for the given area and/or date before requesting the quality parameters from the data provider. The quality parameters give some indication of the quality and accuracy of the available data. Within the last step, all available information is retrieved and delivered to the user.

IV. QUALITY-ORIENTED SERVICE DESIGN

In this section, the services for the described scenario are designed considering the quality attributes introduced in the Background section. For this purpose, first service candidates are systematically derived from the business requirements. Afterwards, these candidates are analyzed and revised according the quality attributes. The revised service candidates are used to derive service designs as full specifications of the required services. Finally, the service designs are again analyzed and revised. As result, service designs are created that fulfill both the functional requirements and certain quality attributes.

A. Derivation of Service Canditates

In a first step, service candidates have to be derived from the modeled business requirements. This step can be performed systematically, as there exist clear descriptions about which elements are transformed into which ones. For this step especially the business process has to be considered as it describes provided functionality and the dependencies between participating roles. Figure 7 shows the methodology for service candidate derivation.

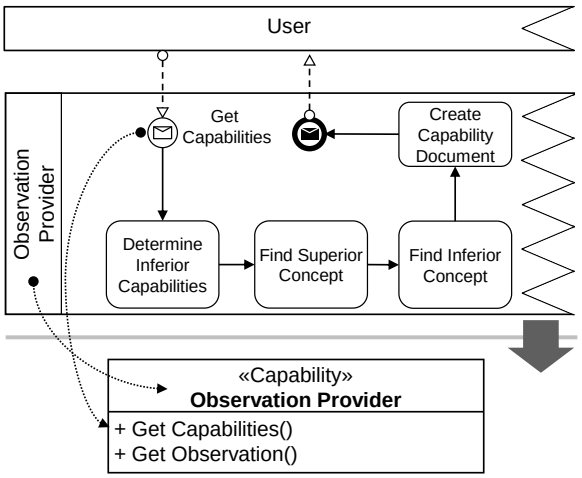

Figure 7. Derivation of service candidates.

Each pool is transformed into a service candidate and each message start event that represents an available operation is transformed into an operation candidate.

The dependencies between the service candidates are derived from the operation calls between the pools. As result three service candidates can be derived as shown in Figure 8.

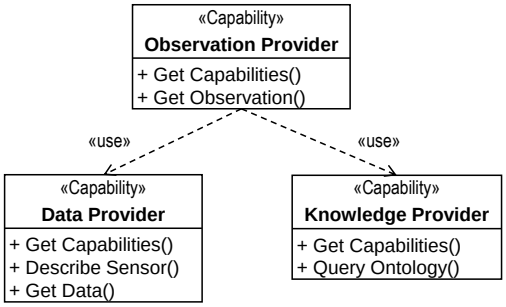

Figure 8. Derived service candidates.

B. Service Candidate Analysis and Revision

In order to assure a high quality of the services already during this phase, a quality analysis is performed. For that purpose, the service candidates are analyzed by measuring the quality indicators introduced in the Background section. During this phase not all quality indicators are applicable as some information might be missing. Based on the available information, the following quality indicators are determined. The used metrics are taken from Gebhart et al. [4].

1) Unique Categorization: In order to measure the separation of technical and business-related functionality, the following metric is applied.

$$DBTF(sc) = \frac{|\,BF(OC(sc))\,|}{|\,OC(sc)\,|}$$

TABLE III. VARIABLES AND FUNCTIONS USED FOR DBTF

Element	Description
DBTF	Division of Business-related and Technical Functionality
sc	service candidate: the considered service candidate
s	service: the considered service that is provided or required, represented by an ServicePoint or RequestPoint in SoaML
BF(oc)	Business-related Functionality: operation candidates providing business-related functionality out of the set of operation candidates oc
BF(o)	Business-related Functionality: operations providing business-related functionality out of the set of operations o
OC(sc)	Operation Candidates: operation candidates of the service candidate sc
SI(s)	Service Interface: service interface of the service s. In SoaML it is the type of the ServicePoint or RequestPoint s
RI(si)	Realized Interfaces: realized interfaces of the service interface si
O(i)	Operations: operations within the interface i
| oc |	Number of operation candidates oc
| o |	Number of operations o

As all service candidates were derived from the business process they provide business-related functionality only. The value of DBTF for all service candidates is 1. The following table shows the interpretation of this value.

TABLE IV. INTERPRETATION OF VALUES FOR DBTF

Value	Interpretation
0	Only technical functionality is provided
Between 0 and 1	Both business-related and technical functionality is provided
1	Only business-related functionality is provided

This table acknowledges that only business-related functionality is provided. As 0 and 1 are desired values, all service candidates fulfill this aspect optimally. The next quality indicator measures the separation of agnostic and non-agnostic functionality, i.e., the separation of general and highly specific operations. The following metric is applied.

$$DANF(sc) = \frac{|\,AF(OC(sc))\,|}{|\,OC(sc)\,|}$$

TABLE V. VARIABLES AND FUNCTIONS USED FOR DANF

Element	Description
DANF	Division of Agnostic and Non-agnostic Functionality
AF(oc)	Agnostic Functionality: operation candidates providing agnostic functionality out of the set of operation candidates oc
AF(o)	Agnostic Functionality: operations providing agnostic functionality out of the set of operations o

The determination whether an operation provides agnostic functionality or not requires personal estimation. As all operations are generally named and provide functionality that is not specific to a certain scenario, they are assumed as agnostic. As result the metric returns 1 for all service candidates. According to the following table, this represents the case that only agnostic functionality is provided.

TABLE VI. INTERPRETATION OF VALUES FOR DANF

Value	Interpretation
0	Only non-agnostic functionality is provided
Between 0 and 1	Both agnostic and non-agnostic functionality is provided
1	Only agnostic functionality is provided

Also in this case the values 0 and 1 are desired for a unique categorization. Accordingly, a revision regarding this quality indicator is not necessary. For measuring the data superiority the following metric is applied.

$$DS(sc) = 1 - \frac{|\,MBE(OC(sc)) \cap MBE(OC(ALL_{SC} \setminus sc))\,|}{|\,MBE(OC(sc))\,|}$$

TABLE VII. VARIABLES AND FUNCTIONS USED FOR DS

Element	Description
DS	Data Superiority
M1 \ M2	Elements of set M1 without elements of set M2 or the element M2
ALL_{SC}	All existing service candidates
ALL_S	All existing services
MBE(oc)	Managed Business Entities: business entities that are managed by operation candidates oc
MBE(o)	Managed Business Entities: business entities that are managed by operations o

In order to determine the results, the service candidates have to be inspected in detail. All service candidates do not manage business entities as it is known in a typical business environment. In this case, a more data-centric view is required that can be mapped onto the quality indicator.

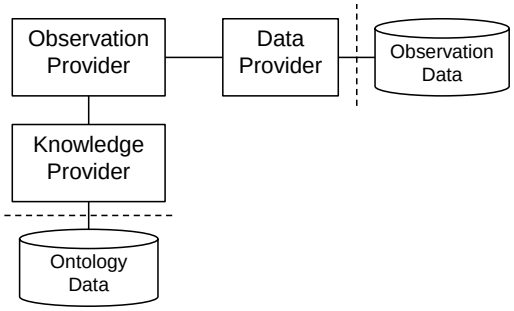

Figure 9. Accessed data storages.

Figure 9 illustrates the services and their access to data storages. This shows that the Data Provider accesses observation data and the Knowledge Provider manages ontology data. The Observation Provider is not responsible for any data directly. As result, for each service candidate but Observation Provider the metric returns 1, which represents the desired value. For Observation Provider this metric is not defined. To exemplify the calculation, the following formula demonstrates it for the Knowledge Provider.

$$DS(Knowledge\ Provider) = 1 - \frac{|\{Ontology\ Data\} \cap \{Observation\ Data\}|}{|\{Ontology\ Data\}|} = 1 - 0 = 1$$

TABLE VIII. TEXT INTERPRETATION OF VALUES FOR DS

Value	Interpretation
Less than 1	No data superiority regarding the managed business entities
1	Data superiority regarding the managed business entities

As result, also in this case there is no revision necessary as all service candidates fulfill the unique categorization concerning this quality indicator optimally. The usage of common business entities can be measured using the following metric.

$$CBEU(sc) = \frac{\left| OCUBE\left(OC(sc), CMP\begin{pmatrix} OC(sc), MOUBE(OC(sc)), \\ UBE(OC(sc)) \end{pmatrix} \right) \right|}{|\,OC(sc)|}$$

TABLE IX. VARIABLES AND FUNCTIONS USED FOR CBEU

Element	Description
CBEU	Common Business Entity Usage
CMP(oc, be1, be2)	Composition: biggest set of business entities managed by operation candidates oc out of be2 that depend on business entitites be1
CMP(o, be1, be2)	Composition: biggest set of business entities managed by operations o out of be2 that depend on business entitites be1
UBE(oc)	Used Business Entities: business entities that are used within operation candidates oc as input
UBE(o)	Used Business Entities: business entities that are used within operations o as input
MOUBE(oc)	Mostly Often Used Business Entities: business entities that are mostly often used within one operation candidate out of operation candidates oc
MOUBE(o)	Mostly Often Used Business Entities: business entities that are mostly often used within one operation out of operations o
OCUBE(oc, be)	Operation Candidates Using Business Entities: operation candidates out of operation candidates oc that only use business entities out of be
OUBE(o, be)	Operations Using Business Entities: operations out of operations o that only use business entities out of be

The calculation of this metric is exemplified for Observation Provider that does not use business entities in any of its operation candidates. In order to comprehend the calculation every function within the formula is calculated separately.

$$OC(Observation\ Provider) = \{Get\ Capabilities, Get\ Observation\}$$

$$MOUBE(OC(ObservationProvier)) = \{\}$$

$$CMP(\ldots) = \{\}$$

$$OCUBE(\ldots) = \{Get\ Capabilities, Get\ Observation\}$$

$$CBEU(Observation\ Provier) = \frac{|\{Get\ Capabilities, Get\ Observation\ \}|}{|\{Get\ Capabilities, Get\ Observation\ \}|} = 1$$

Summarized, every service candidate uses in all of its operation candidates the same business entity and thus is only responsible for one certain business entity or parts of it. Also in this case there is no revision necessary. Thus, the service candidates fulfill the unique categorization optimally.

2) Discoverability: As service candidates describe services and their dependencies in an abstract manner, the discoverability is only important for service designs. Thus, the discoverability will not be measured during this phase but later during the analysis and revision of service designs.

3) Loose coupling: In order to measure the asynchrony for long-running operations details of the service designs are necessary. During the specification of service designs it is determined whether an operation is provided synchronously or asynchronously. Similarly, the complexity of common data types can only be determined when the data types are specified. Thus, also this aspect cannot be measured on service candidates but only on service designs. As provided operations are not final yet and parameters are not defined also the abstraction cannot be measured.

The only quality indicator measurable on basis of service candidates is the compensation. For that purpose, the following metric can be applied.

$$CF(sc) = \frac{\left|CFP\left(SC\left(NC(OC(sc))\right)\right)\right|}{\left|SC\left(NC(OC(sc))\right)\right|}$$

TABLE X. VARIABLES AND FUNCTIONS USED FOR CF

Element	Description
CF	Compensating Functionality
NC(oc)	Non-Compensating: non-compensating operation candidates out of the set of operation candidates oc
NC(o)	Non-Compensating: non-compensating operations out of the set of operations o
SC(oc)	State Changing: operation candidates out of the set of operation candidates oc that provide state-changing functionality
SC(o)	State Changing: operations out of the set of operations o that provide a state-changing functionality
CFP(oc)	Compensating Functionality Provided: operation candidates out of the set of operation candidates oc a compensating operation candidate exists for
CFP(o)	Compensating Functionality Provided: operations out of the set of operations o a compensating operation exists for

TABLE XI. INTERPRETATION OF VALUES FOR CF

Value	Interpretation
Less than 0	There exist state-changing operation candidates respectively operations without compensating operations candidates respectively operations
1	For all operation candidates respectively operations that provide state-changing functionality a compensating operation candidate respectively operation exists

As the Observation Provider only returns information and does not change the state of any artifact, the metric is not defined and there is no revision necessary. Otherwise, the table above lists the values and their interpretation.

4) Autonomy: The dependencies between services can be measured on basis of service candidates using the following metric.

$$SD(sc) = |\,RS(sc)|$$

TABLE XII. VARIABLES AND FUNCTIONS USED FOR SD

Element	Description
SD	Service Dependency
RS(sc)	Required Services: service candidates the service candidate sc depends on
SCT(s)	Service Component: service component of the service s
RS(sct)	Required Services: services the service component sct depends on

For the Observation Provider the metric returns the value 2 as the candidate depends on two other services.

TABLE XIII. INTERPRETATION OF VALUES FOR SD

Value	Interpretation
0	the service candidate or the functionality fulfilling service component depends on no other service candidate respectively service
n (n > 0)	the service candidate or service component requires n other services to fulfill its functionality

Although the value is not optimal, there is no revision possible. The quality indicator shows that there are dependencies, however as the Observation Provider represents a composed service, there is no possibility to improve the quality indicator. Additionally, solving these dependencies would impact other quality indicators, such as those determining the unique categorization.

The functional overlap can be measured using the following metric.

$$FO(sc) = \frac{|\,OF\left(OC(sc), OC(ALL_{sc} \setminus sc)\right)\,|}{|\,OC(sc)\,|}$$

TABLE XIV. VARIABLES AND FUNCTIONS USED FOR FO

Element	Description
FO	Functionality Overlap
OF(oc1, oc2)	Overlapping Functionality: operation candidates out of the set of operation candidates oc1 with overlapping functionality to the operation candidates oc2
OF(o1, o2)	Overlapping Functionality: operations out of the set of operations o1 with overlapping functionality to the operations o2

As in case of the Observation Provider there is no functional overlap, the metric returns 0.

As 0 represents the desired value, there is no revision required. Summarized, the service candidates fulfill nearly all quality indicators optimally. Only the autonomy is not optimal, however this quality indicator cannot be improved

without worsen other quality indicators. Additionally, the composition including the dependencies is intended. Nevertheless, the quality indicators points to the fact that we have dependencies that influence the maintainability and flexibility of the architecture. This has to be kept in mind.

TABLE XV. INTERPRETATION OF VALUES FOR FO

Value	Interpretation
0	The operation candidates respectively operations of the considered service candidate or service do not provide functionality that overlaps with functionality of other service candidates or services
Between 0 and 1	The operation candidates respectively operations of the considered service candidate or service provide functionality that overlaps with functionality of other service candidates or services
1	The operation candidates respectively operations of the considered service candidate or service provide only functionality that overlaps with functionality of other service candidates or services

C. Derivation of Service Designs

Subsequent to the service identification, the service specification can be performed.

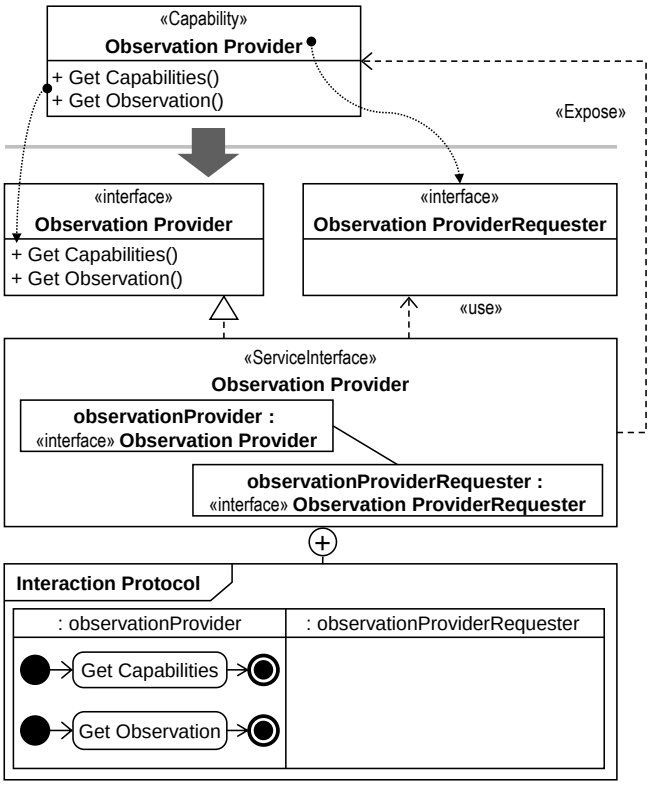

Figure 10. Derivation of service interfaces.

Also in this case, first the service candidates are systematically transformed into service designs. Afterwards, the service designs are iteratively analyzed and revised.

As described in the Background section, a service design consists of a service interface and a service component. Figure 10 illustrates the derivation of a service interface from a service candidate. The service component can be similarly derived as shown in Figure 11.

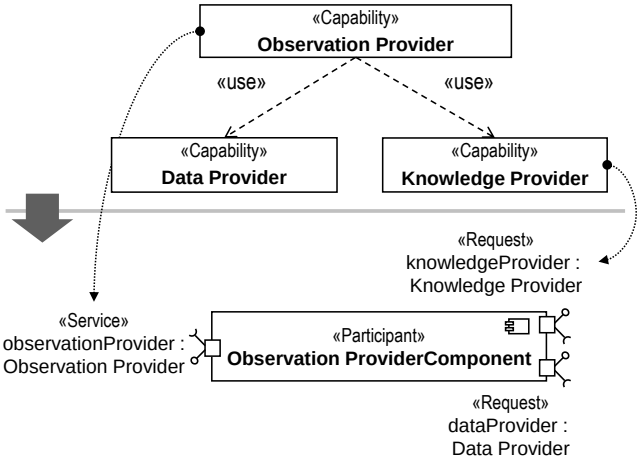

Figure 11. Derivation of service components.

The initial service interfaces and service components are derived from the corresponding capability elements. To create a reference between the service design and the business, the related capability element is attached to the service interface by means of an «Expose» association. The dependencies of the capability elements are reflected with «use» relationships. This relationship information provides the input for the derivation of the ports of the service component. Further details about the systematic derivation are described by Gebhart et al. in [27].

D. Service Design Analysis and Revision

Similarly to the service identification phase, also within the service specification phase an analysis and revision is performed after the systematic derivation of service designs. As the service designs were derived from service candidates with optimized quality indicators, also on basis of service designs most quality indicators will be optimal from the beginning. However, there are some indicators that were not measurable on basis of service candidates. For the sake of completeness, in this section metrics for all quality indicators with focus on service designs are listed. The metrics use the variables and functions introduced above. Also the interpretation of values is identical.

1) Unique Categorization: The quality indicators for the unique categorization can be measured by the following metrics. These metrics focus on the specifics of service designs. The first metric measures the division of business-related and technical functionality.

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

As service within the formula the service described as UML port within the service component, i.e., the Participant in SoaML, has to be chosen.

The division of agnostic and non-agnostic functionality is measured by the following metric.

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

Also the data superiority differs only in the methodology how to determine the relevant operations. Compared to the service candidates, the operations within the realized interface of the service interface have to be chosen.

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

The common business entity usage can be measured using the following metric.

$$CBEU(s) = \frac{\left| OUBE\left(\begin{matrix} O\left(RI\left(SI(s)\right)\right), \\ CMP\left(\begin{matrix} O\left(RI\left(SI(s)\right)\right), MOUBE\left(O\left(RI\left(SI(s)\right)\right)\right), \\ UBE\left(O\left(RI\left(SI(s)\right)\right)\right) \end{matrix} \right) \end{matrix} \right)\right|}{\left| O\left(RI\left(SI(s)\right)\right)\right|}$$

As the service designs were derived from high-quality service candidates, all metrics have the same results as on basis of service candidates. So, there is no revision necessary.

2) Discoverability: The discoverability could not be measured on service candidates as they only represent abstract services with non-final names. Thus, new metrics have to be introduced.

The functional naming of service interfaces, roles, operations, parameters, and data types are measured by the following metrics.

$$FNSI(s) = \frac{\left| FN\left(SI(s)\right)\right|}{|SI(s)|}$$

$$FNR(s) = \frac{\left| FN\left(R\left(SI(s)\right)\right)\right|}{\left| R\left(SI(s)\right)\right|}$$

$$FNO(s) = \frac{\left| FN\left(O\left(RI\left(SI(s)\right)\right)\right)\right|}{\left| O\left(RI\left(SI(s)\right)\right)\right|}$$

$$FNP(s) = \frac{\left| FN\left(P\left(O\left(RI\left(SI(s)\right)\right)\right)\right)\right|}{\left| P\left(O\left(RI\left(SI(s)\right)\right)\right)\right|}$$

$$FNDT(s) = \frac{\left| FN\left(DT\left(P\left(O\left(RI\left(SI(s)\right)\right)\right)\right)\right)\right|}{\left| DT\left(P\left(O\left(RI\left(SI(s)\right)\right)\right)\right)\right|}$$

TABLE XVI. VARIABLES AND FUNCTIONS USED FOR FNSI, FNR, FNO, FNP, AND FNDT

Element	Description
FNSI	Functional Naming of Service Interface
FNR	Functional Naming of Roles
FNO	Functional Naming of Operations
FNP	Functional Naming of Parameters
FNDT	Functional Naming of Data Types
FN(me)	Functional Naming: set of functionally named elements out of the set of modelling elements me
P(o)	Parameters: parameters of the operations o and in case of messages the contained parameters
DT(p)	Data Types: used data types (recursively continued) of parameters p
R(si)	Roles: roles of service interface si

As the original service candidates were derived from business requirements the metric always returns 1 with the following interpretation.

TABLE XVII. INTERPRETATION OF VALUES FOR FNSI, FNR, FNO, FNP, AND FNDT

Value	Interpretation
Less than 1	There are elements that are not functionally named
1	All elements are functionally named

The naming convention compliance of service interfaces, roles, operations, parameters, and data types, can be measured as follows:

$$NCCSI(s) = \frac{\left| NCC\left(SI(s)\right)\right|}{|SI(s)|}$$

$$NCCR(s) = \frac{\left| NCC\left(R\left(SI(s)\right)\right)\right|}{\left| R\left(SI(s)\right)\right|}$$

$$NCCO(s) = \frac{\left| NCC\left(O\left(RI\left(SI(s)\right)\right)\right)\right|}{\left| O\left(RI\left(SI(s)\right)\right)\right|}$$

$$NCCP(s) = \frac{\left| NCC\left(P\left(O\left(RI\left(SI(s)\right)\right)\right)\right)\right|}{\left| P\left(O\left(RI\left(SI(s)\right)\right)\right)\right|}$$

$$NCCDT(s) = \frac{\left| NCC\left(DT\left(P\left(O\left(RI\left(SI(s)\right)\right)\right)\right)\right)\right|}{\left| DT\left(P\left(O\left(RI\left(SI(s)\right)\right)\right)\right)\right|}$$

TABLE XVIII. VARIABLES AND FUNCTIONS USED FOR NCCSI, NCCR, NCCO, NCCP, AND NCCDT

Element	Description
NCCSI	Naming Convention Compliance of Service Interface
NCCR	Naming Convention Compliance of Roles
NCCO	Naming Convention Compliance of Operations
NCCP	Naming Convention Compliance of Parameters
NCCDT	Naming Convention Compliance of Data Types
NCC(me)	Naming Convention Compliance: set of elements out of the set of modelling elements me that follow specified naming conventions

The used names do not correspond to naming conventions specified in the project. For example, spaces are not allowed within names, which is why the NCCSI for the Observation Provider service interface returns 0.

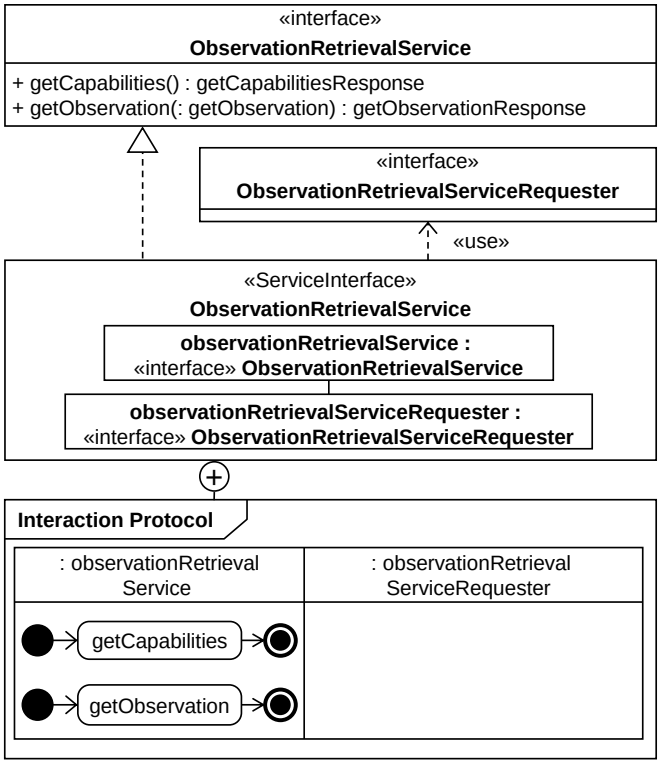

Figure 12. Revised service interface.

TABLE XIX. INTERPRETATION OF VALUES FOR NCCSI, NCCR, NCCO, NCCP, AND NCCDT

Value	Interpretation
Less than 1	There are elements that do not follow naming conventions
1	All elements follow naming conventions

As result, the names of the artifacts have to be revised in order to fulfill the naming conventions and support the discoverability. Figure 12 shows the revised service interface for the Observation Provider.

Whether all possible five information is provided can be measured by the following metric.

$$IC(s) = \frac{EX(SI(s)) + EX\left(RI(SI(s))\right) + EX\left(UI(SI(s))\right) + EX\left(R(SI(s))\right) + EX\left(IP(SI(s))\right)}{5}$$

TABLE XX. VARIABLES AND FUNCTIONS USED FOR IC

Element	Description
IC	Information Content
EX(e)	Exists: returns 1 if the element e exists, else 0
IP(si)	Interaction Protocol: interaction protocol of the service interface si
UI(si)	Used Interfaces: used interface provided by the service consumer

As in this article all information is provided, the metric returns 1 for all services.

TABLE XXI. INTERPRETATION OF VALUES FOR IC

Value	Interpretation
Less than 1	Within the service design not all possible information is available
1	All possible information is available

3) Loose Coupling: Most quality indicators for loose coupling were not measurable on basis of service candidates. Thus, entirely new metrics have to be introduced.

The asynchrony for long-running operations can be determined as follows.

$$ASYNC(s) = \frac{\left|ASO\left(IP(SI(s))\right) \cap LRO\left(O\left(RI(SI(s))\right)\right)\right|}{\left|LRO\left(O\left(RI(SI(s))\right)\right)\right|}$$

TABLE XXII. VARIABLES AND FUNCTIONS USED FOR ASYNC

Element	Description
ASYNC	Asynchrony
ASO(ip)	Asynchronous Operations: asynchronous operations within the interaction protocol ip
LRO(o)	Long Running Operations: long-running operations out of the set of operations o

Whether an operation is provided synchronously or asynchronously can be determined by means of the "synchronous" flag of a UML CallOperationAction within the interaction protocol. As there is no long-running

operation in this scenario, the metric is not defined and cannot be determined. Otherwise the results can be interpreted as follows.

TABLE XXIII. INTERPRETATION OF VALUES FOR ASYNC

Value	Interpretation
Less than 1	There are long-running operations that are not provided asynchronously
1	All long-running operations are provided asynchronously

The common data type complexity is measured by the following metric.

$$CDTC(s) = \frac{\left| SDT\left(\begin{array}{c} DT\left(P\left(O\left(RI(SI(s))\right)\right)\right) \cap \\ DT\left(P\left(O\left(RI(SI(ALL_s \setminus s))\right)\right)\right) \end{array} \right) \right|}{\left| DT\left(P\left(O\left(RI(SI(s))\right)\right)\right) \right|}$$

TABLE XXIV. VARIABLES AND FUNCTIONS USED FOR CDTC

Element	Description
CDTC	Common Data Types Complexity
SDT(p)	Simple Data Types: simple data types within the parameters pt

The service designs in the considered scenario use own packages for own data types, i.e., they do not have common complex data types. Within the numerator the intersection is empty, which is why the metric returns 0 for all services. As the values 0 or 1 represent desired ones, there is no revision necessary.

TABLE XXV. INTERPRETATION OF VALUES FOR CDTC

Value	Interpretation
0	There are no common data types used
Between 0 and 1	There are common and complex data types used
1	The commonly used data types are simple

The following metrics measure the abstraction of operations and parameters.

$$AO(s) = \frac{\left| A\left(O\left(RI(SI(s))\right)\right) \right|}{\left| O\left(RI(SI(s))\right) \right|}$$

$$AP(s) = \frac{\left| A\left(P\left(O\left(RI(SI(s))\right)\right)\right) \right|}{\left| P\left(O\left(RI(SI(s))\right)\right) \right|}$$

TABLE XXVI. VARIABLES AND FUNCTIONS USED FOR AO AND AP

Element	Description
AO	Abstraction of Operations
AP	Abstraction of Parameters
A(o)	Abstract: set out of operations o that are abstract
A(p)	Abstract: set out of parameters p that are abstract

As the operations and parameters are derived from business requirements, they are abstract by nature and do not contain any technical details. The metrics return 1 for all services. This again represents the desired value, which is why there is no further revision required.

TABLE XXVII. INTERPRETATION OF VALUES FOR AO AND AP

Value	Interpretation
Less than 0	There exist operations respectively parameters that are not abstract
1	All operations respectively parameters are abstract

Determining the compensation is similar to the one on basis of service candidates.

$$CF(s) = \frac{\left| CFP\left(SC\left(NC\left(O\left(RI(SI(s))\right)\right)\right)\right) \right|}{\left| SC\left(NC\left(O\left(RI(SI(s))\right)\right)\right) \right|}$$

As there have been no changes on service designs, the results are the same as on basis of service candidates.

4) Autonomy: Instead of using the dependencies between service candidates the required services of a service component can be considered to determine the dependencies to other services.

$$SD(s) = \left| RS\left(SCT(s)\right) \right|$$

The values for the metric are the same as on basis of service candidates, i.e., the metric returns 2 for the Observation Provider and 0 for the other services.

The functional overlap is determined by the following metric, which returns 0 for the Observation Provider.

$$FO(s) = \frac{\left| OF\left(O\left(RI(SI(s))\right), O\left(RI(SI(ALL_s \setminus s))\right)\right) \right|}{\left| O\left(RI(SI(s))\right) \right|}$$

In a next step, the analysis and revision phase is iteratively repeated until there is no further revision necessary. This is why the service design phase ends at this step for the considered scenario. As result, the analysis and revision phase enabled to create service designs with verifiable fulfilled quality indicators. This will support common and wide-spread quality attributes and strategic goals, such as a high maintainability, flexibility and cost-efficiency.

V. Conclusion and Outlook

In this article, the creation of a service-oriented system with quality attributes kept in mind was demonstrated by a geographical information system. As these kinds of systems access distributed information and are expected to be accessible from other systems, an architecture with service-oriented design principles is necessary. Since strategic goals are associated with this decision, the services within the system have to follow certain quality attributes, such as loose coupling and autonomy. As concrete scenario a system of the PESCaDO project of the European Commission was chosen.

After an introduction and the definition of relevant terms in the Background section, the scenario and the artifacts of the business analysis phase were presented. The considered business use case described the requirement to get an observation by using different other services to ensure that all relevant information for the user are found and retrieved. The resulting business process served as the input for the second phase in the service development process, the service design phase, which was performed afterwards.

The service design phase consists of the combination of a systematic derivation of artifacts and the subsequent analysis and revision. The first enables the fulfillment of functional requirements and the latter ensures the compliance with non-functional ones, such as the quality attributes. As result, formalized service designs based on SoaML were created for the PESCaDO scenario that consider quality attributes and thus support the achievement of strategic goals.

With this systematic approach, the IT architect is assisted with performing the complex service design task. The application of this approach on a real world scenario exemplifies its usage and shows its benefits. On the one hand, the methodology enables the creation of service designs in an engineering manner. On the other hand, the quality indicators provide a catalog of criteria an IT architect has to consider during this task. This ensures that important quality aspects are not overseen. Additionally, the metrics help with analyzing models and improving them cost-efficiently.

The usage of SoaML as emerging standard for modeling service-oriented architectures and service designs enables the embedding of this approach into existing tools and entire tool chains. As SoaML provides a UML profile, any tool supporting UML can be used. However, there exist also several tools supporting SoaML natively. The possibility to apply this approach with common and wide-spread tools increases its practical applicability.

In the future, we plan to enhance the analysis methodology. There are some terms that are not concretely defined within existing work. For example, when is a service agnostic and when specific? In order to avoid ambiguity these terms have to be specified in detail. Additionally, the quality analysis is supposed to consider further quality attributes especially with regards to the internal component-oriented architecture that implements the service components. Also specifics of paradigms for realizing services, such as the resource-centric approach used in RESTful Web services, will be considered.

Finally, to further increase the cost-efficiency and productivity of the service design task we have implemented the metrics within our QA82 Architecture Analyzer tool [9] for an automatic analysis of service designs. Thus, in the future, IT architects, developers, executive board, or customers will be able to automatically evaluate the quality of developed or acquired products and provided services. This simplifies the analysis whether services increase the flexibility, maintainability, and cost-efficiency of the IT.

References

[1] 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.

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

[3] OMG, “Service oriented architecture modeling language (SoaML) – specification for the uml profile and metamodel for services (UPMS)”, Version 1.0, 2012.

[4] 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.

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

[6] 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.

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

[8] 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.

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

[10] The PESCaDO Consortium, “Service-based infrastructure for user-oriented environmental information delivery”, EnviroInfo, 2010.

[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] 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.

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

[14] IBM, “RUP for service-oriented modeling and architecture”, IBM Developer Works, http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/, 2006. [accessed: July 11, 2012]

[15] 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.

[16] 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]

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

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

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

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

[21] 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.

[22] 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]

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

[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, 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.

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

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

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

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

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

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

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

[33] S. Johnston, "Rational uml profile for business modeling", IBM Developer Works, http://www.ibm.com/developerworks/rational/library/5167.html, 2004. [accessed: July 11, 2012]

[34] J. Heumann, "Introduction to business modeling using the unified modeling language (UML)", IBM Developer Works, http://www.ibm.com/developerworks/rational/library/360.html, 2003. [accessed: July 11, 2012]

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

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

[37] M. Horridge, "A practical guide to building owl ontologies using protégé 4 and co-ode tools", http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/, Version 1.2, 2009. [accessed: July 11, 2012]