D3.4 - Service Documentation Guidelines

Contents

Table of figures

List of tables

1 Introduction

The bulk of work on this document, has been made as a deliverable for the EfficienSea2 project co-funded by the European Commission.

For clarity it should be stressed that ‘services’ in the context of this document are ‘technical services’. This could for instance be a service that provides the digital information about weather forecasts in a specific geographical area. It would not be the provision of pilotage in an area, which is also a service, but this would be considered an operational service.

1.1 Purpose of the document

This document provides meta-information explaining how services shall be described in the context of the Maritime Cloud (see http://efficiensea2.org/solution/the-maritime-cloud/).

Service specification and design artefacts described in this document include the definition of data models. It has to be mentioned that these data models shall, whenever possible, be strongly related to the S-100 data framework [5]. This means, service descriptions following this guidelines document shall provide references to the appropriate S-100 features and attributes in their data models.

This document, together with the Service Specification Template [1], the Service Design Description Template [3] and the Service Implementation Description Template [4], builds Deliverable D3.4 of the project EfficienSea2.

1.2 Intended readership

This service documentation guidelines document is intended to be read by service architects who shall produce service specifications in the maritime context.

In addition, this guidelines document is useful for system engineers and developers in charge of designing and developing a service instance.

2 Overview

2.1 Service Management Overview

The Conceptual Model (see reference [2]) describes the service management concept as depicted in Figure 1. Both, service specifications as well as information about service instances are published in the service registry. For more details, please refer to section “Service Registry / Service Management” of reference [2].

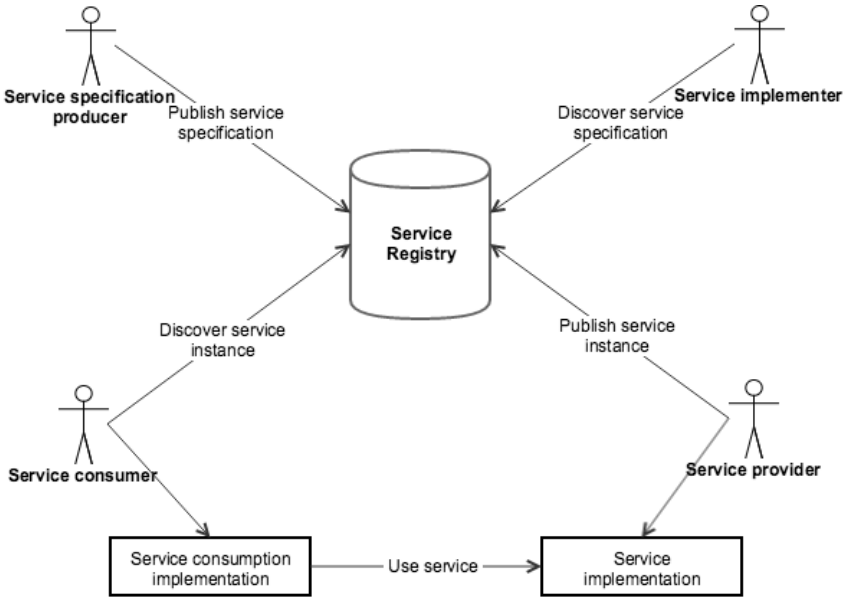

Figure 1: Service Management Concept

Figure 2 provides more insight in the distinction between service specification, service technical design and service implementation. The service specification describes one dedicated service at logical level in a technology-agnostic manner, by providing, for instance:

- the operational context of the service (e.g., requirements, use cases),
- the service interface descriptions (operations, parameters),
- the data structures used by the service (the service data model),
- the dynamic behaviour of the service (sequence of operations),
- author of the service specification (organisation, contact person)

The service specification shall not describe the details of a specific service implementation. For that purpose, a service technical design description has to be provided, where the actual realisation of the service with a dedicated technology shall be described.

It is possible to provide different technical designs (by using same or different technologies), all being compliant with the same service specification. It is also possible to provide one

technical design that conforms to several service specifications, e.g., to allow backward compatibility to older versions of a certain specification.

Each service technical design shall be documented by providing, for instance:

- reference to the service specification,
- description of the chosen technology,
- detailed description of the used data structures (service physical data model),
- mapping of the used data structures to the service specification's service data model,
- author of the technical design (organisation, contact person)

A service implementation (implemented according to a given technical design) may be deployed at different locations by different service providers. For each such service instance a service instance description shall be provided.

Each service instance shall be documented by providing, for instance:

- reference to the service technical design (and thus, implicitly, to the service specification),
- information about service provider,
- coverage information,
- etc.

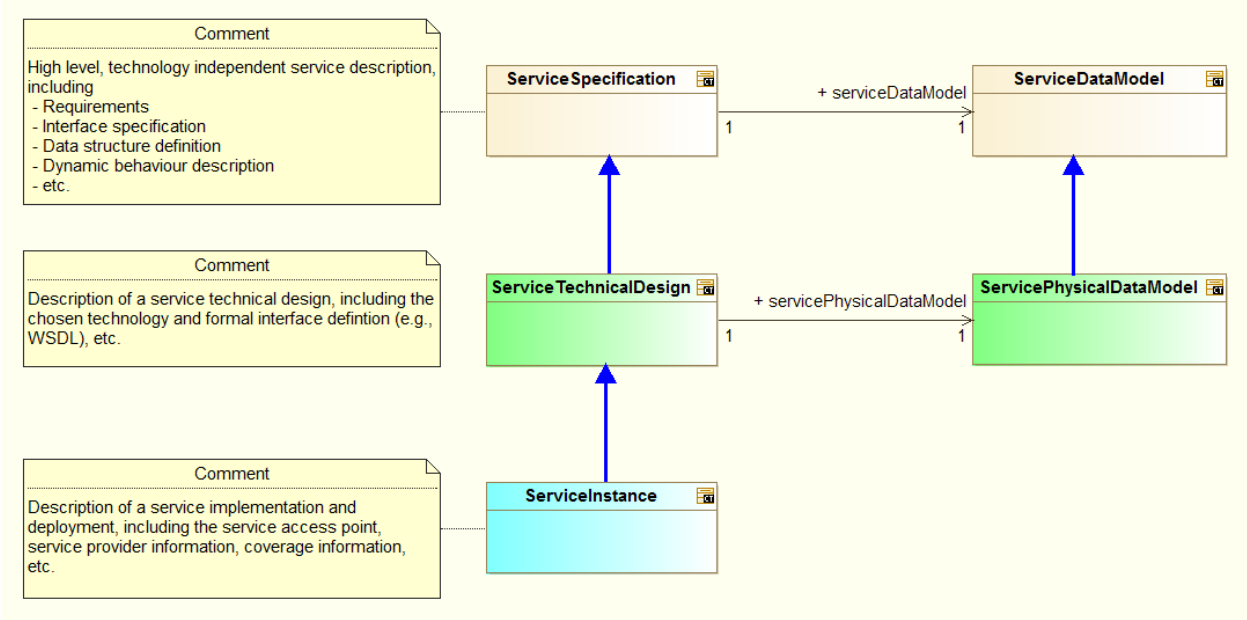

Figure 2: Distinction between Service Specification, Service Technical Design and Service Instance

2.2 Service Documentation Overview

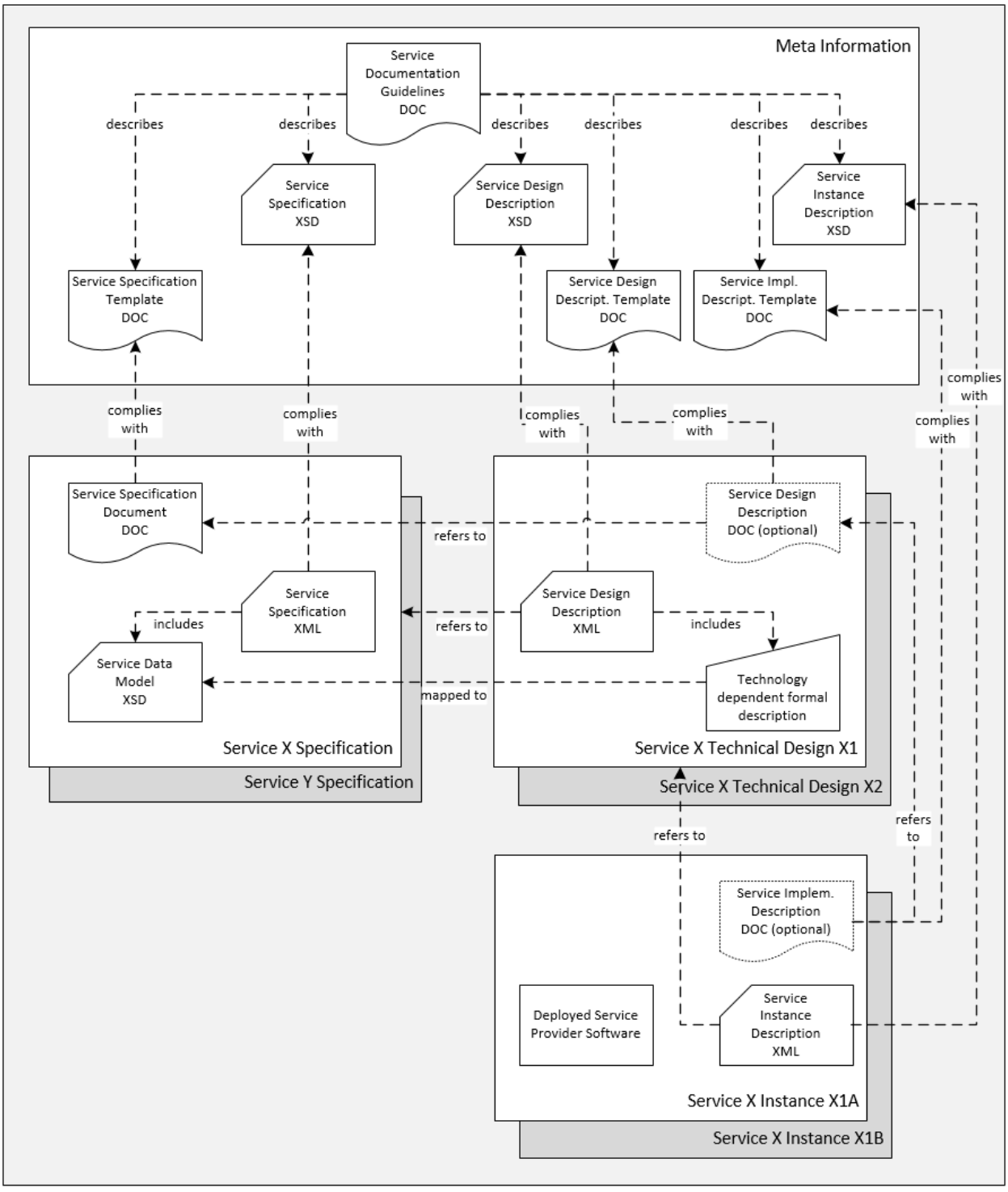

Figure 3: Service Documentation Overview

Figure 3 provides an overview about the service documentation artefacts. The upper rectangle contains meta-information explaining how to describe services. This meta-information consists of:

- service documentation guidelines: this document;

- service specification template: a word document providing the framework for a textual description of a service specification;
- service specification XSD: an XML schema definition for the formal description of a service specification;
- service design description XSD: an XML schema definition for the formal description of the service technical design;
- service design description template: a word document providing the framework for a textual description of the service technical design;
- service instance description XSD: an XML schema definition for the formal description of a service instance;
- service implementation description template: a word document providing the framework for a textual description of the service implementation.

The middle part of the figure contains the artefacts for describing the specification and technical design of a dedicated service.

The service specification describes the “What”-aspects of a service, e.g., what are the characteristics of a Weather Forecast Service. This service specification consists of

- service specification document: a word document (complying with the service specification template [1]) detailing the service specification in textual form supporting the readability for human beings;
- service specification XML: an XML file (following the service specification XSD schema), describing the service specification in a more formal manner;
- service data model XSD: an XML schema definition describing the data model used in the service; the service data model XSD is included in the service specification XML.

The service technical design comprises the “How”-aspects of a service, e.g., how in detail is a Weather Forecast service instance implemented and accessible. Several different technical designs may exist at the same time for one service specification. A service technical design consists of:

- service design description: a document describing the technical service design. The provision of this document is optional and it is up to the implementer to choose a suitable format for this document, e.g., by using the service design description template [3]. If provided, the document shall refer to the service specification document;
- service design description XML: an XML file (following the service design description XSD), describing the service technical design in a formal manner, e.g., by providing information needed for the registration in the service registry;
- technology dependent formal description: additional service description files as appropriate for the chosen technology (e.g., WSDL, XSD, YAML, JSON, etc.), describing the details (syntax, protocol, etc.) of the exchanged data. The contents of

this formal description must be mapped to the service data model. The means of how such mapping has to be performed is not prescribed, as they heavily depend on the chosen technology and actual service technical design. In some cases the mapping is implicitly given (e.g., if the service technical design re-uses the service data model in a 1:1 manner). In other cases a mapping table may be provided (e.g., as part of the service design description document), mapping each single data element of the service instance to a corresponding data element of the service data model.

The lower part of the figure contains the artefacts for a dedicated implementation (instance) of a service. The service instance comprises the “Where”-aspects of a service, e.g., the actual access address (URL) of a weather forecast service and the geographical coverage of it. Several service instances may exist at the same time, all implementing the same service technical design. A service instance consists of:

- deployed service provider software implementation: this is the actual service implementation. This is not part of the description, but it is the “subject” that shall be described. It consists of all the software and configuration artefacts needed for providing the service;
- service implementation description: a document describing the actual service implementation. The provision of this document is optional and it is up to the implementer to choose a suitable format for this document, e.g., by using the service implementation description template [4]. If provided, the document shall refer to the service technical design description document;
- service instance description XML: an XML file (following the service instance description XSD), describing the service instance in a formal manner, e.g., by providing information needed for the registration in the service registry.
  Note that one service implementation (described by one service implementation description document) may be deployed several times at different access points. In this case, several service instance description XML files have to be produced – one for each deployed instance.

2.3 Process Considerations

This document describes how the service documentation shall look like in the context of the Maritime Cloud. However, this document intentionally does not prescribe any process to be followed when generating such documentation. In particular, this document does not identify any governance rules for service technical design and implementation (see also Section 6).

This means, the two sub-sections below shall be just seen as proposals, not meaning that these are the only valid approaches.

2.3.1 Top-Down Service Development

In a top-down approach, the necessity of a new service and its basic outline would be first identified and described in an operational requirements document. This step is optional, and out of scope of this service documentation guidelines document.

Once the decision for building a service has been taken, a service architect (in the role of a service specification producer) creates the service specification by producing the service specification document and the service specification XML including the service data model. If an operational requirements document exists, the service specification refers to it; otherwise the requirements are documented in the service specification.

As soon as the service specification has reached sufficient maturity, it is published in the service registry.

An interested service provider or implementer (there could also be more than one) takes the service specification and elaborates a technical design for it. During this step, technology decisions are taken and documented in the service technical design. The service technical design may already be published in the service registry before the service is actually implemented and deployed. This is useful for developers of service consumer software, as they can already base their development on the service design description while the service provider software is still under development.

Having the service technical design in place, service implementers will develop the software required for service provision.

When the service software is sufficiently mature, the service provider deploys it and publishes the access information (URL) and coverage area in the service instance description in the service registry.

Interested service consumers can obtain service specifications and service technical design descriptions from the service registry and build the required client software.

Interested service consumers with existing clients look up the service registry for service instances (complying with their choice of technology) in order to get the access points for the provided services in their respective geographical area.

2.3.2 Bottom-Up Service Documentation

Already existing services may be documented in a bottom-up approach. In order to publish the service in the service registry, all of the following information shall be provided: a service instance description, a service design description and a service specification.

Assuming the service already exists, it should be easy to provide the service design description and service instance description in the structure/format described in this guidelines document. In this case, it is assumed that the technology specific data model already exists and can be directly taken as part of the service design description.

The service data model (part of the service specification) can be derived by abstracting the existing data model from technology-specific details.

The rest of the service specification (interface and operation descriptions as well as requirements) has to be newly created.

Once the service instance description, the service design description and the service specification are sufficiently mature, they shall be published in the service registry and interested consumers may look them up.

3 Service Specification

The purpose of the service specification is to collect the results of service identification and service design activities. The aim is to document the key aspects of a dedicated service at the logical level:

- the operational and business context of the service
  - requirements for the service (e.g., information exchange requirements)
  - involved nodes: which operational components provide/consume the service
  - operational activities supported by the service
  - relation of the service to other services
- the service description
  - service interfaces
  - service interface operations
  - service payload definition
  - service dynamic behaviour description
- service provision and validation aspects

The purpose of the service specification is to provide a holistic overview of a particular service and its building blocks at logical level. The service specification consists of

- (mandatory) a service specification document:
  a human readable documentation of the service key aspects;
- (mandatory) a service specification XML:
  a formal description of the service specification in a more formal manner, including a formal definition of the service data model;
- (optional) a model based description:
  e.g., UML model describing the service interfaces, operations and data structures. The service specification document as well as the service specification XML might re-use artefacts produced in the model based description.

Note that the service specification is intended to be technology-agnostic. The service specification shall not describe the details of a specific service implementation. For that purpose, a service design description has to be provided as well, where the actual realisation of the service with a dedicated technology shall be described.

3.1 Service Specification Document

The purpose of the service specification document is to document in human readable manner all the information comprising a service specification. It should be noted that a service specification document describes one dedicated version of one dedicated service in detail at logical level.

The service specification document describes a well-defined baseline of the service and clearly identifies the service version. In this way, it supports the configuration management process.

The service specification document provides also the foundation material for the future standardisation process.

A template is available in order to assure a certain uniformity of service specification documents produced by different authors.

3.2 Service Specification Template

The service specification template [1] shall support the service architects in creating a document based description of the services at a high level of abstraction. The template prescribes a structure of chapters (to be completed by the author of the service specification), and for each section descriptive instructions for the intended content.

The basic structure of the service specification template is replicated in the subsections below.

3.2.1 Introduction

The introduction section contains the usual basic information, such as purpose of the document, intended readership, etc.

3.2.2 Service Identification

The service identification section provides a tabular overview of mainly administrative attributes needed for identification and lookup of the service. Examples: name, identifier, version, author, key words of the service specification.

The service identifier shall be unique, for example in form of an MRN (Maritime Resource Name) according to [6].

3.2.3 Operational Context

The operational context section describes the context of the service from an operational perspective.

The operational context description should be based on the description of the operational model, consisting of a structure of operational nodes and operational. If such an operational model exists, this section shall provide references to it. If no such operational model exists, then its main aspects shall be described in this section.

Optionally, a simple high level use case, described in layman’s terms, could be provided as an introduction to this section.

The operational context shall be a description of how the service supports interaction among operational nodes. This can be achieved in two different levels of granularity:

1. A description of how the service supports the interaction between operational nodes. This basically consists of an overview about which operational nodes shall provide the service and which operational nodes will consume the service.
2. A more detailed description stating what operational activities the service supports in a process model.

Moreover, the operational context shall describe any requirement the service shall fulfil or adhere to. This refers to functional as well as non-functional requirements at high level (business/regulatory requirements, system requirements, user requirements). Especially, information exchange requirements are of much interest since the major objective of services is to support interaction between operational nodes.

The source material for the operational context description should ideally be provided by operational users and is usually expressed in dedicated requirements documentation. Any applicable documents shall be mentioned in the References section. If no requirements documents are available, then the basic requirements for the service shall be defined in the service specification document in tabular form.

The service shall be linked to at least one requirement.

3.2.4 Service Overview

This chapter aims at providing an overview of the main elements of the service. The elements in this view are all usually created by means of an UML modelling tool.

Architectural elements applicable for this description are:

- Service:
  the element representing the service as a whole.
- Service Interfaces:
  the communication mechanisms of the service, i.e., interaction mechanisms between service provider and service consumer. Defined by allocating service operations to either the provider or the consumer of the service.
- Service Operations:
  describe the logical operations used to access the service.
- Service Operations Parameter Definitions:
  identify data structures being exchanged via Service Operations.

The above elements may be depicted in one or several diagrams. Which and how many diagrams are needed depends on the chosen architecture description framework and complexity of the service.

The service overview may be described by using an UML diagram[1] that illustrates the service interfaces with their operations and their allocation to service provider and service consumer. This information should also be provided in tabular form.

It is also recommended to describe the considerations resulting in the selection of a certain Message Exchange Pattern (MEP) for the service interfaces.

A service interface supports one or several service operations. Depending on the Message Exchange Pattern, service operations are either to be implemented by the service provider (e.g., in a Request/Response MEP, query operations are provided by the service provider – the service consumer uses them in order to submit query requests to the service provider), or by the service consumer (e.g., in a Publish/Subscribe MEP, publication operations are provided by the service consumer – the service provider uses them to submit publications to the service consumer). This distinction shall be clearly visualised. For each service interface it shall be stated whether it is provided or requested by the service. A service provides at least one service interface.

3.2.5 Service Data Model

This section shall describe the data structures to be exchanged between service providers and consumers. The data model shall provide enough information allowing to implement the service based on this information, but on the other hand it should describe the data structures sufficiently abstract; this means, it should be avoided to list all details or to define technology-specific data types. It is recommended to visualise the data structures by means ofUML diagrams. The complete **logical data structure** should be shown using diagram(s) and explanatory tables. It is mandatory to give a description of each entity item (class), its attributes and the relations between entity items after each diagram that shows data items.

If the service data model is related to an external data model (e.g., being a subset of a standard data model, e.g. typically based on an S-100 specification), then the service data model shall refer to it: each data item of the service data model shall be mapped to a data item defined in the external data model. This mapping may be added in the same table that describes the data items and their attributes and associations. The idea is: when reading the service specification (including the logical service data model), the reader must clearly understand the payload structures. If the service re-uses structures of an external data model, then these structures can be referred to rather than replicated in the service specification. The tabular presentation of the payload allows to provide references to an externally defined model.

An XML schema description of the service data model shall be attached to the service specification document in an annex. The primary reason for the XML schema for the logical data model is not the possibility of data validation. The idea is to provide some kind of formalism already at the logical level, in order to enforce some uniformity across different

[1] e.g., in NATO Architectural Framework (NAF), a NSOV-2 diagram could be used

service specifications. Note that the S-100 specification [5] describes in its Appendix 9-B how S-100 based data models shall be formulated in XML schema format.

In addition to the data model exchanged between service providers and consumers, this section may optionally also contain a description of the internal data model, as it seems appropriate to the service provider and/or the service consumer side. Such a description might be helpful for the understanding as it provides additional information of how the service might be built. However, this internal service data model has to be declared as exemplary only – it is not an authoritative part of the service specification

3.2.6 Service Interface Specification

This chapter describes the details of each service interface. The static interface description provided in this section – together with the service data model provided in previous section – is one of the core parts of the service specification since it describes how the interfaces shall be constructed.

Architectural elements applicable for this description are:

- Service Interfaces
- Service Operations
  Functions or procedures which enable programmatic communication with a Service via a Service interface.
- Parameters
  Constants or variables passed into or out of a service interface as part of the execution of a service operation

A service may have one or more service interfaces. Each of them shall be described in a separate sub-section. The sub-section title shall contain the service interface name.

For each service interface, the purpose, message exchange pattern and architecture of the Interface shall be described.

A service interface supports one or several service operations. Each of them shall be described in a separate sub-section. The sub-section title shall contain the name of the operation. Each service operation sub-section shall contain the following information:

- Functionality:
  shall include a textual description of the operation functionality. In most instances this will be the same as the operation description taken from the UML modelling tool.
- Parameters:
  shall describe the logical data structure of input and output parameters of the operation (payload) by using UML diagrams (which are usually sub-sets of the service data model described in previous section above) and explanatory tables.
  It is mandatory to provide a table with a clear description of each service operation parameter, and the information about which data types defined in the service data

model are used by the service operation in its input and output parameters.
Note: While the descriptions provided in the service data model shall explain the data types in a neutral format, the descriptions provided here shall explicitly explain the purpose of the parameters for the operation.

3.2.7 Service Dynamic Behaviour

This chapter describes the interactive behaviour between service interfaces (interaction specification) and, if required, between different services (orchestration). Architectural elements applicable for this description are:

- Service interaction specifications
- Service state machines
- Service orchestration

Following types of views and UML diagrams can be used to describe the dynamic behaviour[2]:

- Sequence diagrams
- Interaction diagrams
- State machine diagrams

A separate sub-section shall be provided for each service interface. The minimum content of such sub-section is some information about the dynamic aspects of the service interface. Each operation shall be exposed on at least one diagram.

3.2.8 Service Provisioning (optional)

This chapter should describe the way services are planned to be provided and consumed. It is labelled optional since one of the key aspects of service-orientation is to increase flexibility of the overall system by separating the definition of services from their implementation. This means that a service can be provided in several different contexts that are not necessarily known at the time, when the service is designed.

3.2.9 References

The References section contains a list of all documents referred to by the service specification (e.g., requirements documents, if any).

[2] e.g., in NATO Architectural Framework (NAF), state model and interaction specification (NAF3.1) or NSOV-5 Service constraints, state model could be used.

3.3 Service Specification XSD Structure

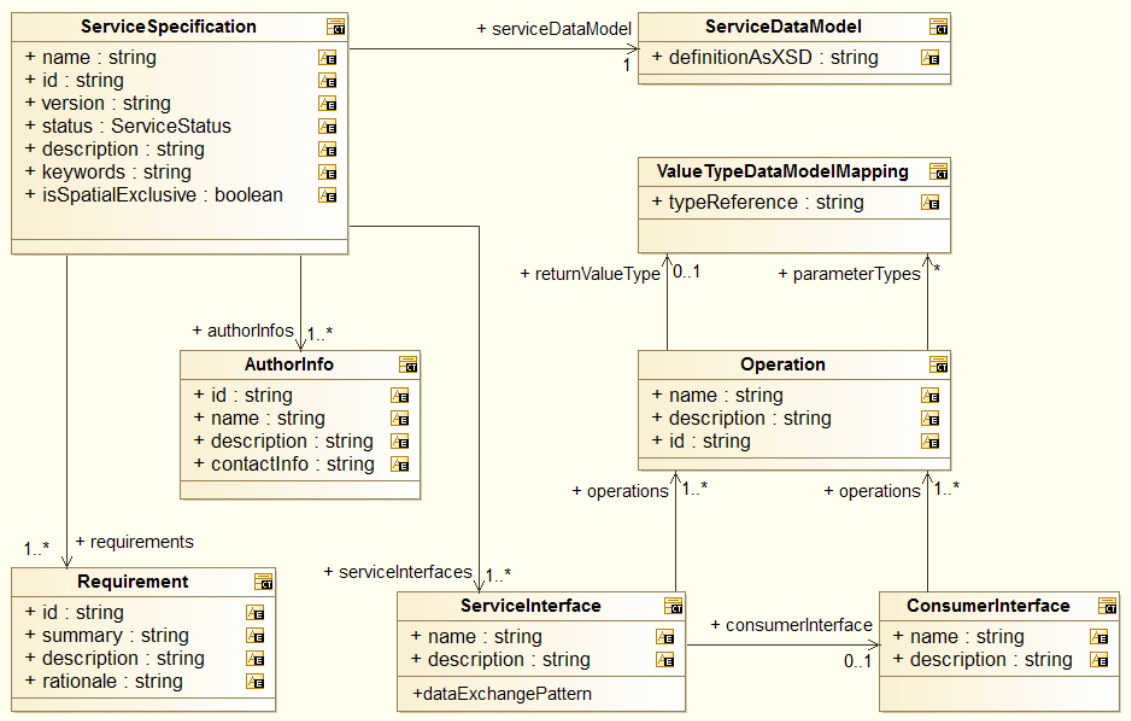

Figure 4: Structure of the Service Specification

Figure 4 gives an overview about the formal description of the service specification. The individual items are described in the table below.

Note that this formal description of a service specification is intentionally kept simple and plain. For most described objects lots of more attributes could be added and standardized, but in order to get as much adoption as possible the entry barrier should be low.

The service specification XSD file is presented in Appendix A.

Table 1: Information elements of the Service Specification

Type Name	Description	
ServiceSpecification	A service specification describes one dedicated service at logical level in a technology-independent way. The service specification identifies a service by its id and version. The service specification refers to requirements for the service, defines a service data model at logical level, defines the service interface(s) and provides information about the author(s).	
Element Name	**Type**	**Description**
Name	string	The human readable service name. The service name should be at maximum a one line brief label for the service. Newer versions of the same service specification should not change the name.
Id	string	Globally unique identification of the service. Newer versions of the same service specification shall not change the id.

version	string	Version of the service specification. A service specification is uniquely identified by its id and version. Any change in the service data model or in the service interface definition requires a new version of the service specification
status	enumeration	Status of the service specification. The status field has one of the following values: • provisional, • released, • deprecated, • deleted.
description	string	A human readable short description of the service. The description shall contain an abstract of what a service implementing this specification would actually do.
keywords	string	A list of keywords associated to the service.
isSpatialExclusive	boolean	Flag to indicate whether the service shall be “spatial exclusive”. “Spatial exclusiveness” means that just one service instance of the same service specification providing the same technical design is allowed to be registered for a certain geographical area.
requirements	Requirement	Refers to requirements specifications for the service. Business requirements, functional and non-functional requirements should be listed here. At least one requirement is mandatory.
authorInfos	AuthorInfo	Refers to administrative information about the authors of the service. It is mandatory to provide at least one author information.
serviceInterfaces	ServiceInterface	Refers to the definition of service interfaces. At least one service interface shall be defined.
serviceDataModel	ServiceDataModel	Mandatory reference to the definition of the logical service data model.

Type Name	**Description**
Requirement	A requirement that the service shall fulfil.

Element Name	**Type**	**Description**
id	string	Globally unique requirement identification
name	string	Human readable requirement name/summary. Shall not be longer than one line.
text	string	The human readable requirement text. Usually formulated in form of a “shall”-statement.
rationale	string	Rationale for this requirement. Textual explanation of why this requirement exists. Provides background information about the need of the service.

	reference	string	Optional information about where the requirement was originally stated. If the requirement comes from external documents, this attribute shall refer to this source.
	author	AuthorInfo	Optional reference(s) to administrative information about the author(s) of the requirement.

Type Name	**Description**
AuthorInfo	Describes an author of a service specification or requirement.

	Element Name	**Type**	**Description**
	id	string	Unique identifier of the author.
	name	string	Human readable name of the author.
	description	string	Human readable description of the author.
	contactInfo	string	Human readable contact information of the author.

Type Name	**Description**
ServiceInterface	Specification of a service interface. One service can offer several interfaces, e.g. both a request/response interface and a publish/subscribe interface at the same time. Different interfaces will usually provide different service operations.

	Element Name	**Type**	**Description**
	name	string	Human readable service interface name. The name shall be no longer than one line.
	description	string	Human readable description of the service interface
	dataExchangePattern	enumeration	Message exchange pattern can be one of • ONE_WAY data are sent in one direction, from service consumer to service provider, without confirmation; • REQUEST_RESPONSE service consumer sends request to service provider and expects to receive a response from the service provider; • REQUEST_CALLBACK (asynchronous REQUEST_RESPONSE) service consumer sends a request to service provider; response is provided asynchronously in an independent call to the service; • PUBLISH_SUBSCRIBE service consumer subscribes at service provider for receiving publications sent out by the service provider; • BROADCAST service provider distributes information independently of any consumers.

	operations	Operation	Refers to the specification of service operations supported by the service interface. At least one operation shall be defined.
	consumerInterfaces	ConsumerInterface	Optional reference to an interface definition that has to be provided by the service consumer in order to complement the service interface. Especially if a publish/subscribe service interface is designed, it is necessary to describe what the service expects to be available on the subscriber side.

Type Name	**Description**
ConsumerInterface	Interface specification that is expected to be provided by the service consumer. For example, if a publish/subscribe service interface is designed, it is necessary to describe what the service expects to be available on the subscriber side.

	Element Name	**Type**	**Description**
	name	string	Human readable interface name. The name shall be no longer than one line.
	description	string	Human readable description of the interface.
	operations	Operation	Refers to the specification of service operations supported by the consumer interface. At least one operation shall be defined.

Type Name	**Description**
Operation	Definition of a service operation. Operations allow a service consumer to interact with the service. An operation describes a dedicated function of the service or the consumer.

	Element Name	**Type**	**Description**
	name	string	Human readable operation name. The name shall be no longer than one line.
	description	string	Human readable description of the operation.
	returnValueType	ValueTypeData-ModelMapping	Optional definition of the return value for the operation. The return value could be a business object or a simple status code. The return value data type has to be defined in the logical service data model.
	parameterTypes	ValueTypeData-ModelMapping	Definition of one or more parameters for the operation. This could be business objects or simple types. Parameters have to be defined in the logical service data model.

Type Name	**Description**
ValueTypeDataModel-Mapping	Definition of a data type by providing a reference into the logical service data model. A value type data model mapping is used either in a service operation parameter or return value.

	Element Name	**Type**	**Description**
	typeReference	string	Reference to the logical service data model. It references a type (or element, though type is preferred) in the logical service model by the type's name attribute. References are enforced using XSD unique key and keyrefs.

Type Name	**Description**

ServiceDataModel		The serviceDataModel is a logical model. It is formally described in XSD to achieve interoperability and decouple it from implementing physical data models described in e.g. SOAP or REST. The model can either be described in-line, or existing schemata can be imported. One service specification has one logical service model.	
	Element Name	**Type**	**Description**
	definitionAsXSD	String	The definition of the service data model described in XSD.

4 Service Technical Design

4.1 Service Design Description Document

The purpose of the service design description is to document in human readable manner all the information comprising the technical design of a service. This document shall provide a detailed description of how a service shall be realized with a certain technology. For the technology-independent information this document shall refer to the service specification document, rather than replicating any information.

Note that in theory one service technical design may describe several different kinds of services. In this case, all service specifications shall be referenced in the service design description. On the other hand it is obvious that one service specification may be referenced by several different service design descriptions. This is the case when a service shall be implemented/provided by using different technologies.

In order to assure a certain uniformity of service technical design description documents produced by different authors, the document shall be aligned with the service design description template.

4.2 Service Design Description Template

The service design description template [3] shall support the service architects/designers in creating a document based description of the service technical design. The template prescribes a structure of chapters (to be completed by the author of the service technical design), and for each section descriptive instructions for the intended content.

The basic structure of the service design description template is replicated in the subsections below.

4.2.1 Introduction

The introduction section contains the usual basic information, such as purpose of the document, intended readership, etc.

4.2.2 Service Design Identification

The service design identification section provides a tabular overview of mainly administrative attributes needed for identification and lookup of the service design. Example content of this section: reference to service specification; name, identifier and version of the technical design; author (vendor information), key words, etc.

4.2.3 Technology Introduction

The technology introduction section contains a basic background about the chosen technology. In most cases this will be a short description of basic technology aspects

accompanied with appropriate references to standard documents and best practice descriptions.

4.2.4 Service Design Overview

This chapter aims at providing an overview of the main elements of the service design and a mapping of the design elements to the service specification elements. The elements in this view are all usually created by means of an UML modelling tool.

Architectural elements applicable for this description are:

- Service:
  the element representing the service as a whole.
- Service Interfaces:
  the communication mechanisms of the service, i.e., interaction mechanisms between service provider and service consumer. Defined by allocating service operations to either the provider or the consumer of the service.
- Service Operations:
  describe the operations used to access the service.
- Service Operations Parameter Definitions:
  identify data structures being exchanged via Service Operations.

The above elements may be depicted in one or several diagrams. Which and how many diagrams are needed depends on the chosen architecture description framework , the chosen technology, and the complexity of the service.

If the structure of the service design follows the service specification to a great extent, then it is not necessary to replicate identical diagrams here in this section; in this case, this section shall contain references to the service specification document. However, it is assumed that in many cases, depending on the chosen technology, the actual interface and/or operation names (and structuring) are not fully identical to the abstract definition given in the service specification.

The service design overview may be described by using an UML diagram[3] that illustrates the service interfaces with their operations and their allocation to service provider and service consumer. This information should also be provided in tabular form. Furthermore, it shall be described how the specified Message Exchange Patterns (MEP) are realised with the chosen technology.

4.2.5 Physical Data Model

This chapter provides a detailed description of the data structures exchanged between service provider and service consumer. This description shall also include a mapping of the data structures to the service data model provided in the service specification.

[3] e.g., in NATO Architectural Framework (NAF), a NSOV-2 diagram could be used

The service design description template does not prescribe a detailed format for this section. Allowed presentations of the physical data model include

- UML diagrams representing the data structures including detailed physical data type descriptions at attribute level;
- XML/XSD files describing the data structures;
- Tabular presentations.

Any mixture of the above formats is allowed.

If the physical service data model is related to an external data model (e.g., being a subset of a standard data model, e.g. based on an S-100 specification), then this section shall refer to it: each data item of the physical data model shall be mapped to a data item defined in the external data model. This mapping may be added in the same table that describes the data items and their attributes and associations.

4.2.6 Service Interface Design

The structure of the service interface design section is identical to the structure of the service interface specifications section in the service specification document (see 3.2.6 over). This section may be limited to references to the service specification document, if all of the following conditions are fulfilled:

- the service design reflects the service interfaces in a 1:1 manner,
- the service interfaces are sufficiently described in the service specification,
- the physical data model (section 4.2.5) contains an unambiguous mapping of all payload data items of the service specification to the detailed physical data items.

4.2.7 Service Dynamic Behaviour

This chapter describes the interactive behaviour between service interfaces (interaction specification) and, if required, between different services (orchestration).

Following types of views and UML diagrams can be used to describe the dynamic behaviour:

- Sequence diagrams
- Interaction diagrams
- State machine diagrams

This section is especially relevant, if the service design structure (see section 4.2.4) differs from the service structure introduced in the service specification. If designed service interfaces and operations are equivalent to those of the service specification, and if the dynamic behaviour is sufficiently described in the service specification, then this section may be limited to references to the service specification document.

4.2.8 References

The References section contains a list of all documents referred to by the service design description (e.g., service specification, requirements documents, etc.). At least the service specification document needs to be referenced.

4.3 Service Design Description XSD Structure

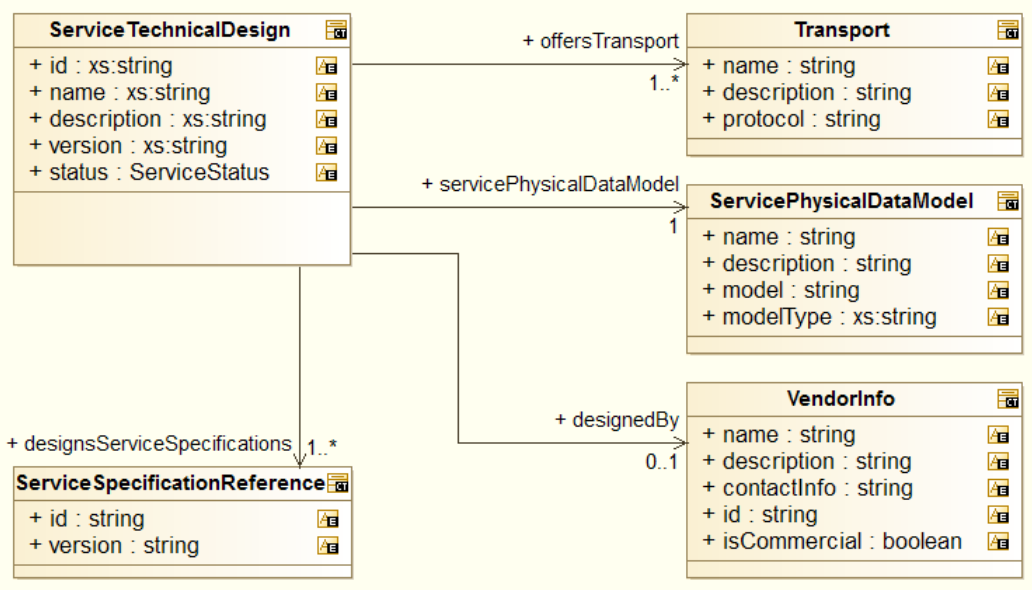

Figure 5: Structure of the Service Technical Design Description

Figure 5 gives an overview about the formal description of the service design description. A description of each item is given in the table below.

Note that this formal description of a service technical design is intentionally kept simple and plain. For all described objects many more attributes and related objects could be added and standardized, but in order to get as much adoption as possible the entry barrier should be kept low, and therefore quite some aspects have been intentionally left out.

The service instance description schema id presented in Appendix B.

Table 2 : Information Elements of the Service Design Description

Type Name		Description	
ServiceDesign		A service design description.	
	Element Name	**Type**	**Description**
	name	string	The human readable name of the service design. The service name should be at maximum a one line brief label. Newer versions of the same service design should adopt the same name.

id	string	Globally unique identification of the service design. Newer versions of the same service design shall adopt the same id.
version	string	Version of the service design. A service design is uniquely identified by its id and version. Any change in the service physical data model or in the service specification reference requires a new version of the service design.
status	enumeration	Status of the service design. The status field has one of the following values: • provisional, • released, • deprecated, • deleted.
description	string	A human readable short description of the service design. The description shall contain an abstract of what a service implementation actually does.
designsService-Specifications	ServiceSpecificationReference	Refers to service specification(s) that is/are realised by this service design. As a minimum, one service specification shall be referenced. One service design may realise several service specifications (either different versions of one specification, or even different specifications).
offersTransport	Transport	Refers to transport technologies offered by the service design. At least one reference shall be provided.
designedBy	VendorInfo	Mandatory reference to information about the author of the service design.
servicePhysicalData-Model	ServicePhysical-DataModel	Mandatory reference to the service physical data model description.

Type Name	**Description**
ServiceSpecification-Reference	A reference to the service specification that is realised by the service design. - Has the id and the version of the respective service specification.

Element Name	**Type**	**Description**
id	string	Identification of the service specification realised by the service design.
version	string	Version of the service specification realised by the service design.

Type Name	**Description**
Transport	Definition of the transport protocol used by the service design.

Element Name	**Type**	**Description**
name	string	Human readable name.
description	string	Human readable description of the transport protocol used by the service design.
protocol	string	A non-formal string representation of the transport (e.g. http/rest, http/soap,..) that provides enough information to a service consumer to be able to connect.

Type Name	**Description**

VendorInfo		Describes the vendor providing the service design.	
	Element Name	**Type**	**Description**
	id	string	Unique identification of the vendor.
	name	string	Human readable vendor name. The name shall be no longer than one line.
	description	string	Human readable description of the vendor.
	contactInfo	string	Human readable contact information of the vendor.
	isCommercial	boolean	Optional indication on the commercial status of the vendor.
Type Name		**Description**	
ServicePhysicalDataModel		The ServicePhysicalDataModel describes the data model for the service design. The ServicePhysicalDataModel describes in detail all the data structures being actually exchanged when service consumers interact with a service instance that implements this design.	
	Element Name	**Type**	**Description**
	name	string	Human readable model name. The name shall be no longer than one line.
	description	string	Human readable description of the model.
	model	string	The model can e.g. be a WSDL file, a JSON API, or the like. It is recommended to wrap the model in a CDATA section, and provide enough information in the name and description to make clear how to deal with the content in model.
	modelType	string	The modelType should contain e.g. an abbreviation that indicates what technology is used to describe the model. E.g. WSDL, JSON.

5 Service Instance

5.1 Service Implementation Description Document

The purpose of the service implementation description is to document in human readable manner all the information specific to a certain implementation of a service. This document shall provide a detailed description of how a service is actually realised. In most cases, this document will be rather short, since it is expected that the implementation follows the technical design and it is not supposed to replicate any information from the service design description document.

Note that one service implementation may be deployed several times at different access points. In this case, several service instance description XML files have to be produced – one for each deployed instance – but only **one** service implementation description document is required.

In order to assure a certain uniformity of service implementation description documents produced by different authors, the document shall be aligned with the service implementation description template.

5.2 Service Implementation Description Template

The service implementation description template [3] shall support the service developers in creating a document based description of the service implementation. The template prescribes a structure of chapters (to be completed by the service implementer), and for each section descriptive instructions for the intended content.

The basic structure of the service implementation description template is replicated in the subsections below.

5.2.1 Introduction

The introduction section contains the usual basic information, such as purpose of the document, intended readership, etc.

5.2.2 Service Implementation Identification

The service identification section provides a tabular overview of mainly administrative attributes about the service implementation. Example content of this section: reference to service technical design; name, identifier and version of the implementation; author (vendor information), key words, etc.

5.2.3 Service Implementation Details

This section describes any information that appears useful for the understanding of the service implementation. This may include internal design decisions, required configuration data, deployment pre-requisites, etc.

The service implementation description template does not prescribe a detailed format for this section.

5.2.4 Release Notes

This section describes the release notes of the service implementation. It shall contain at least the following set of information:

- Release identification and date
- Feature list
  - added features
  - changed features
  - removed features
- Bug list
  - known open bugs
  - resolved bugs

The service design implementation template does not prescribe a detailed format for this section.

5.2.5 References

The References section contains a list of all documents referred to by the service implementation description (e.g., service specification, service design, requirements documents, etc.). At least service specification and service design documents need to be referenced.

5.3 Service Instance Description XSD Structure

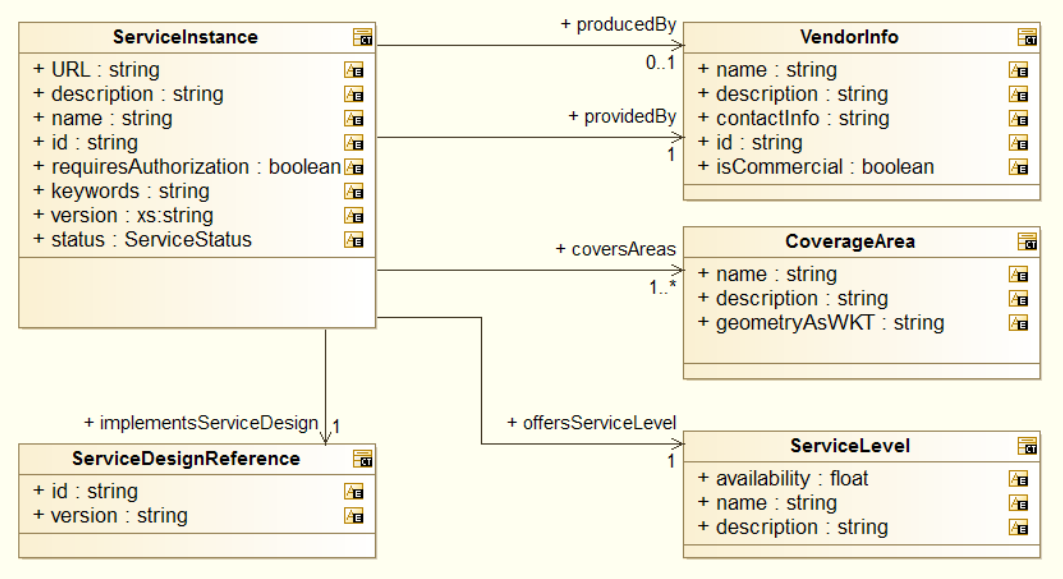

Figure 6: Structure of the Service Instance Description

Figure 6 gives an overview about the formal description of the service instance description. The individual items are described in the table below.

Note that this formal description of a service instance is intentionally kept simple and plain. For all described objects lots of more attributes and related objects could be added and standardized, but in order to get as much adoption as possible the entry barrier should be low and therefore quite some aspects (like e.g. when a service would be available) have been left out.

The service instance description schema id presented in Appendix C.

Table 3 : Information Elements of the Service Instance Description

Type Name		Description	
ServiceInstance		A service instance description. One service implementation may be deployed at several places by same or different service providers; each such deployment represents a different service instance.	
	Element Name	**Type**	**Description**
	name	string	The human readable name of the service instance. The service name should be at maximum a one line brief label. Newer versions of the same service specification should adopt the same name.
	id	string	Globally unique identification of the service instance. Newer versions of the same service instance shall adopt the same id.
	version	string	Version of the service instance. A service instance is uniquely identified by its id and version. Any change in the service design reference requires a new version of the service instance.
	status	enumeration	Status of the service instance. The status field has one of the following values: • provisional, • released, • deprecated, • deleted.
	description	string	A human readable short description of the service instance. The description shall contain an abstract of what a service implementation actually does.
	keywords	string	A list of keywords associated to the service.
	URL	string	URL that describes where the service endpoint is located
	requiresAuthorization	boolean	Indicates whether authorization is required or not.
	implementsService-Design	ServiceDesignReference	Refers to the service design that is implemented by this service instance. Exactly one service design shall be referenced.

	coversAreas	CoverageArea	Mandatory reference to the geographical area covered by the service instance. Note that the content of the CoverageArea defaults to the whole world, if its content is empty.
	offersServiceLevel	ServiceLevel	Refers to the definition of the service level fulfilled by the service instance. Exactly one service level definition shall be provided.
	producedBy	VendorInfo	Optional reference to information about the producer of the service implementation.
	providedBy	VendorInfo	Mandatory reference to information about the service provider of the service instance.

Type Name	**Description**
ServiceDesignReference	A reference to the service design that is implemented by the service instance. - Has the id and the version of the respective service design.

	Element Name	**Type**	**Description**
	id	string	Identification of the service design implemented by the service instance.
	version	string	Version of the service design implemented by the service instance.

Type Name	**Description**
CoverageArea	Defines a geographical area from which the service instance is accessible.

	Element Name	**Type**	**Description**
	name	string	Human readable name of the coverage area, e.g. a well-known name like "Bermuda Triangle". The name shall be no longer than one line.
	description	string	Human readable description of the coverage area.
	geometryAsWKT	string	A polygon described in WKT (Well Known Text) with coordinates in coordinate reference system EPSG:4326, e.g. POLYGON(LON1 LAT1, LON2 LAT2, LON3, LAT3, LON1 LAT1). If the element is empty, the default is the whole world.

Type Name	**Description**
ServiceLevel	Defines the service availability level.

	Element Name	**Type**	**Description**
	name	string	Human readable service level name. The name shall be no longer than one line.
	description	string	Human readable description of the service level
	availability	float	Indicates the guaranteed availability of the service in %, (e.g. 99.9).

Type Name	**Description**
VendorInfo	Describes the vendor producing and/or providing the service instance.

	Element Name	**Type**	**Description**
	id	string	Unique identification of the vendor.
	name	string	Human readable vendor name. The name shall be no longer than one line.

	description	string	Human readable description of the vendor.
	contactInfo	string	Human readable contact information of the vendor.
	isCommercial	boolean	Optional indication on the commercial status of the vendor.

6 Governance

It is anticipated that some kind of governance will be needed in the area of service management. This includes questions about the process to

- decide about maturity of service specifications,
- decide about the scope of service specifications,
- decide about the evolution of service specifications,
- decide about the life cycle of service specifications and service instances,
- decide about conformance of service instances to specifications.

The definition and description of governance structures and procedures is out of scope of this document and also not covered by WP3 of the EfficienSea2 project.

7 References

Nr.	Version	Reference
[1] Service Specification Template	01.01	SG_Annex_B_Service_Specification_Template
[2] Conceptual Model	01.00	Maritime Cloud – Conceptual Model
[3] Service Design Description Template	01.00	SG_Annex_C_Service_Design_Description_Template
[4] Service Implementation Description Template	01.00	SG_Annex_D_Service_Implementation_Description_Template
[5] S-100 Universal Hydrographic Data Model	2.0.0	S-100 – UNIVERSAL HYDROGRAPHIC DATA MODEL http://www.iho.int/iho_pubs/standard/S-100/S-100_Ed_2/S_100_V2.0.0_June-2015.pdf
[6] Maritime Resource Name		Maritime Resource Name, ENAV17-n.n.n

8 Acronyms and Terminology

8.1 Acronyms

Term	Definition
API	Application Programming Interface
MC	Maritime Cloud
MEP	Message Exchange Pattern
MRN	Maritime Resource Name
NAF	NATO Architectural Framework
REST	Representational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSD	Service Specification Document
UML	Unified Modelling Language
URL	Uniform Resource Locator
VTS	Vessel Traffic Service
WSDL	Web Service Definition Language
XML	Extendible Mark-up Language
XSD	XML Schema Definition

8.2 Terminology

Term	Definition
External Data Model	Describes the semantics of the “maritime world” (or a significant part thereof) by defining data structures and their relations. This could be at logical level (e.g., in UML) or at physical level (e.g., in XSD schema definitions), as for example standard data models, or S-100 based data produce specifications.
Message Exchange Pattern	Describes the principles two different parts of a message passing system (in our case: the service provider and the service consumer) interact and communicate with each other. Examples: In the Request/Response MEP, the service consumer sends a request to the service provider in order to obtain certain information; the service provider provides the requested information in a dedicated response. In the Publish/Subscribe MEP, the service consumer establishes a subscription with the service provider in order to obtain certain information; the service provider publishes information (either in regular intervals or upon change) to all subscribed service consumers.
Operational Activity	An activity performed by an operational node. Examples of operational activities in the maritime context are: Route Planning, Route Optimization, Logistics, Safety, Weather Forecast Provision, …
Operational Model	A structure of operational nodes and associated operational activities and their inter-relations in a process model.

Operational Node	A logical entity that performs activities. Note: nodes are specified independently of any physical realisation. Examples of operational nodes in the maritime context are: Maritime Control Center, Maritime Authority, Ship, Port, Weather Information Provider, …
Service	The provision of something (a non-physical object), by one, for the use of one or more others, regulated by formal definitions and mutual agreements. Services involve interactions between providers and consumers, which may be performed in a digital form (data exchanges) or through voice communication or written processes and procedures.
Service Consumer	A service consumer uses service instances provided by service providers. All users within the maritime domain can be service customers, e.g., ships and their crew, authorities, VTS stations, organizations (e.g., meteorological), commercial service providers, etc.
Service Data Model	Formal description of one dedicated service at logical level. The service data model is part of the service specification. Is typically defined in UML and/or XSD. If an external data model exists (e.g., a standard data model), then the service data model shall refer to it: each data item of the service data model shall be mapped to a data item defined in the external data model.
Service Design Description	Documents the details of a service technical design (most likely documented by the service implementer). The service design description includes (but is not limited to) a service physical data model and describes the used technology, transport mechanism, quality of service, etc.
Service Implementation	The provider side implementation of a dedicated service technical design (i.e., implementation of a dedicated service in a dedicated technology).
Service Implementer	Implementers of services from the service provider side and/or the service consumer side. Anybody can be a service implementer but mainly this will be commercial companies implementing solutions for shore and ship.
Service Instance	One service implementation may be deployed at several places by same or different service providers; each such deployment represents a different service instance, being accessible via different URLs.
Service Instance Description	Documents the details of a service implementation (most likely documented by the service implementer) and deployment (most likely documented by the service provider). The service instance description includes (but is not limited to) service technical design reference, service provider reference, service access information, service coverage information, etc.
Service Interface	The communication mechanism of the service, i.e., interaction mechanism between service provider and service consumer. A service interface is characterised by a message exchange pattern and consists of service operations that are either allocated to the provider or the consumer of the service.
Service Operation	Functions or procedure which enables programmatic communication with a service via a service interface.
Service Physical	Describes the realisation of a dedicated service data model in a

Data Model	dedicated technology. This includes a detailed description of the data payload to be exchanged using the chosen technology. The actual format of the service physical data model depends on the chosen technology. Examples may be WSDL and XSD files (e.g., for SOAP services) or swagger (Open API) specifications (e.g., for REST services). If an external data model exists (e.g., a standard data model), then the service physical data model shall refer to it: each data item of the service physical data model shall be mapped to a data item defined in the external data model. In order to prove correct implementation of the service specification, there shall exist a mapping between the service physical data model and the service data model. This means, each data item used in the service physical data model shall be mapped to a corresponding data item of the service data model. (In case of existing mappings to a common external (standard) data model from both the service data model and the service physical data model, such a mapping is implicitly given.)
Service Provider	A service provider provides instances of services according to a service specification and service instance description. All users within the maritime domain can be service providers, e.g., authorities, VTS stations, organizations (e.g., meteorological), commercial service providers, etc.
Service Specification	Describes one dedicated service at logical level. The Service Specification is technology-agnostic. The Service Specification includes (but is not limited to) a description of the Service Interfaces and Service Operations with their data payload. The data payload description may be formally defined by a Service Data Model.
Service Specification Producer	Producers of service specifications in accordance with the service documentation guidelines.
Service Technical Design	The technical design of a dedicated service in a dedicated technology. One service specification may result in several technical service designs, realising the service with different or same technologies.
Service Technology Catalogue	List and specifications of allowed technologies for service implementations. Currently, SOAP and REST are envisaged to be allowed service technologies. The service technology catalogue shall describe in detail the allowed service profiles, e.g., by listing communication standards, security standards, stacks, bindings, etc.
Spatial Exclusiveness	A service specification is characterised as “spatially exclusive”, if in any geographical region just one service instance of that specification is allowed to be registered per technology. The decision, which service instance (out of a number of available spatially exclusive services) shall be registered for a certain geographical region, is a governance issue.

Appendix A Service Specification Schema

ServiceBaseTypesSchema.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ServiceSpecificationSchema="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceSpecificationSchema.xsd" targetNamespace="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceSpecificationSchema.xsd" version="1.0" xml:lang="EN">
  <annotation>
    <documentation>
                Authors:
                    EfficienSea2 WP 3 Partners
                        Thomas Lutz
                        Christoph Rihacek
                        Josef Jahn
                        Hubert Künig
    </documentation>
  </annotation>
  <annotation>
    <documentation>
                This file contains basic data type definitions for service specification, design and instance
                descriptions.
    </documentation>
  </annotation>

  <complexType name="AuthorInfo">
    <annotation>
      <documentation>
                Describes an author of a service specification or requirement.

                Elements of an authorInfo are:

                - id          Unique identifier of the author.

                - name        Human readable name of the author.

                - description Human readable description of the author.

                - contactInfo Human readable contact information of the author.
      </documentation>
    </annotation>
    <sequence>
      <element name="id" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="contactInfo" type="string" minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>

  <complexType name="VendorInfo">
    <annotation>
      <documentation>
              Describes the vendor producing and/or providing the service instance..

              - id            Unique identification of the vendor.

              - name          Human readable vendor name. The name shall be no longer than one line.

              - description   Human readable description of the vendor.

              - contactInfo   Human readable contact information of the vendor.

              - isCommercial  Optional indication on the commercial status of the vendor.
      </documentation>
    </annotation>
    <sequence>
      <element name="id" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="contactInfo" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="isCommercial" type="boolean" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>

  <simpleType name="ServiceIdentifier">
    <annotation>
      <documentation>
        Service identifier type to be used by service specifications, designs, instances.
        Currently, the identifier is defined as a string.
      </documentation>
```

```
    </annotation>
    <restriction base="string"/>
  </simpleType>

  <simpleType name="ServiceVersion">
    <annotation>
      <documentation>
        Service version indicator type to be used by service specifications, designs, instances.
        Currently, the version indicator is defined as a string.
      </documentation>
    </annotation>
    <restriction base="string"/>
  </simpleType>

  <simpleType name="ServiceStatus" final="restriction">
    <annotation>
      <documentation>
         Service status may be one of the values listed below. Service specifications,
         service designs and service instances each have their own status value.
            provisional    the service specification/design is not officially released, the service instance is
                           available, but not in official operation
            released       the service specification/design/instance is officially released / in operation
            deprecated     the service specification/design/instance is still available, but end of life is already
                           envisaged.
            deleted        the service specification/design/instance is not available any more.
      </documentation>
    </annotation>
    <restriction base="string">
      <enumeration value="provisional"/>
      <enumeration value="released"/>
      <enumeration value="deprecated"/>
      <enumeration value="deleted"/>
    </restriction>
  </simpleType>
</schema>
```

ServiceSpecificationSchema.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ServiceSpecificationSchema="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceSpecificationSchema.xsd" targetNamespace="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceSpecificationSchema.xsd" version="1.0" xml:lang="EN">
  <include schemaLocation="ServiceBaseTypesSchema.xsd"/>
  <annotation>
    <documentation>
                Authors:
                    EfficienSea2 WP 3 Partners
                        Thomas Lutz
                        Christoph Rihacek
                        Josef Jahn
                        Hubert Künig
    </documentation>
  </annotation>
  <annotation>
    <documentation>
                This formal description of a service specification is intentionally kept
                simple and plain.
                For all described objects lots of more attributes and related objects could
                be added and standardized, but in order to get as much adoption as possible
                the entry barrier should be low and therefore quite some aspects e.g. like
                when a service would be available have been left out.
                </documentation>
  </annotation>
  <element name="serviceSpecification" type="ServiceSpecificationSchema:ServiceSpecification">
    <annotation>
      <documentation>
                        The root element of a service specification.
                        Please refer to the type serviceSpecification for details.

                        The serviceSpecification element enforces the tracing of service
                        operation parameters and return values.
      </documentation>
    </annotation>
    <!-- asp parsers seem to have an issue with the unique key being not on the same level -->
    <unique name="serviceDataModelTypeKey">
      <selector xpath=".//xs:*"/>
      <field xpath="@name"/>
    </unique>
    <keyref name="serviceDataModelReturnValueTypeKeyRef" refer="ServiceSpecificationSchema:serviceDataModelTypeKey">
      <selector xpath=".//ServiceSpecificationSchema:returnValueType"/>
```

```
      <field xpath="ServiceSpecificationSchema:typeReference"/>
    </keyref>
    <keyref name="serviceDataModelParameterTypeTypeKeyRef" refer="ServiceSpecificationSchema:serviceDataModelTypeKey">
      <selector xpath=".//ServiceSpecificationSchema:parameterType"/>
      <field xpath="ServiceSpecificationSchema:typeReference"/>
    </keyref>
  </element>
  <complexType name="ServiceSpecification">
    <annotation>
      <documentation>
        A service specification describes one dedicated service at logical level in a technology-independent
        way.
        The service specification identifies a service by its id and version. The service specification
        refers to requirements for the service, defines a service data model at logical level,
        defines the service interface(s) and provides information about the author(s).

        Elements of a service specification are:

        - name           The human readable service name. The service name should be at maximum a one
                         line brief label for the service. Newer versions of the same service
                         specification should not change the name.

        - id             Globally unique identification of the service. Newer versions of the same
                         service specification shall not change the id.

        - version        Version of the service specification. A service specification is uniquely
                         identified by its id and version. Any change in the service data model or in the
                         service interface definition requires a new version of the service specification.

        - status         Status of the service specification. One of the values 'provisional', 'released',
                         'deprecated', 'deleted'.

        - description    A human readable short description of the service. The description shall contain
                         an abstract of what a service implementing this specification would actually do.

        - keywords       A list of keywords associated to the service.

        - isSpatialExclusive Flag to indicate whether the service shall be "spatial exclusive". "Spatial
                         exclusiveness" means that at most one service instance of the same service
                         specification and providing the same technical specification is allowed to be
                         registered for any geographical area.

        - requirements   Refers to requirements specifications for the service. Business requirements,
                         functional and non-functional requirements should be listed here.
                         At least one requirement is mandatory.

        - authorInfos    Refers to administrative information about the authors of the service. At least
                         one author information is mandatory.

        - serviceInterfaces Refers to the definition of service interfaces. At least one service interface
                         shall be defined.

        - serviceDataModel  Mandatory reference to the definition of the logical service data model.
      </documentation>
    </annotation>
    <all>
      <element name="id" type="ServiceSpecificationSchema:ServiceIdentifier" minOccurs="1" maxOccurs="1"/>
      <element name="version" type="ServiceSpecificationSchema:ServiceVersion" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="status" type="ServiceSpecificationSchema:ServiceStatus" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="keywords" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="isSpatialExclusive" type="boolean" default="false" minOccurs="0" maxOccurs="1"/>
      <element name="requirements" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="requirement" type="ServiceSpecificationSchema:Requirement" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="authorInfos" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="authorInfo" type="ServiceSpecificationSchema:AuthorInfo" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="serviceDataModel" type="ServiceSpecificationSchema:ServiceDataModel" minOccurs="1"
maxOccurs="1"/>
      <element name="serviceInterfaces" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
```

```
            <element name="serviceInterface" type="ServiceSpecificationSchema:ServiceInterface" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </all>
  </complexType>
  <complexType name="Requirement">
    <annotation>
      <documentation>
        A requirement that the service specification fulfils.

        Elements of a requirement are:

        - id        Globally unique requirement identification

        - name      Human readable requirement name/summary. Shall not be longer than one line.

        - text      The human readable requirement text. Usually formulated in form of a "shall"-statement.

        - rationale Rationale for this requirement. Textual explanation of why this requirement exists.
                    Provides background information about the need of the service.

        - reference Optional information about where the requirement was originally stated. If the
                    requirement comes from external documents, this attribute shall refer to this source.

        - author    Optional reference(s) to administrative information about the author(s) of the requirement.

      </documentation>
    </annotation>
    <all>
      <element name="id" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="text" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="rationale" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="reference" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="authorInfos" minOccurs="0" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="authorInfo" type="ServiceSpecificationSchema:AuthorInfo" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </all>
  </complexType>
  <complexType name="ServiceInterface">
    <annotation>
      <documentation>
        Specification of a service interface. One service can offer several interfaces,
        e.g. both a request/response interface and a publish/subscribe interface at the same time.
        Different interfaces will usually provide different service operations.

        Elements of a serviceInterface are:

        name                Human readable service interface name. The name shall be no longer than one line.

        description         Human readable description of the service interface

        dataExchangePattern Message exchange pattern can be one of
                                ONE_WAY,
                                REQUEST_RESPONSE,
                                REQUEST CALLBACK,
                                PUBLISH SUBSCRIBE,
                                BROADCAST.

        operations          Refers to the specification of service operations supported by the service
                                interface. At least one operation shall be defined.

        consumerInterfaces  Optional reference to an interface definition that has to be provided by the
                                service consumer in order to complement the service interface.
                                Especially if a publish/subscribe service interface is designed, it is
                                necessary to describe what the service expects to be available on the
                                subscriber side.

      </documentation>
    </annotation>
    <all>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="dataExchangePattern" minOccurs="1" maxOccurs="1">
        <simpleType>
          <restriction base="string">
            <enumeration value="REQUEST RESPONSE"/>
            <enumeration value="PUBLISH_SUBSCRIBE"/>
```

```
            <enumeration value="BROADCAST"/>
          </restriction>
        </simpleType>
      </element>
      <element name="operations" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="operation" type="ServiceSpecificationSchema:Operation" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="consumerInterface" type="ServiceSpecificationSchema:ConsumerInterface" minOccurs="0"
maxOccurs="1"/>
    </all>
  </complexType>
  <complexType name="ConsumerInterface">
    <annotation>
      <documentation>
        Interface specification that is expected to be provided by the service consumer. For example,
        if a publish/subscribe service interface is designed, it is necessary to describe what the service
        expects to be available on the subscriber side.

        Elements of a consumerInterface are:

        - name        Human readable interface name. The name shall be no longer than one line.

        - description Human readable description of the interface.

        - operations  Refers to the specification of service operations supported by the consumer interface.
                      At least one operation shall be defined.
      </documentation>
    </annotation>
    <all>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="operations" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="operation" type="ServiceSpecificationSchema:Operation" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </all>
  </complexType>
  <complexType name="Operation">
    <annotation>
      <documentation>
        Definition of a service operation. Operations allow a service consumer to interact with
        the service. An operation describes a dedicated function of the service or the consumer.

        Elements of an operation are:

        - name            Human readable operation name. The name shall be no longer than one line.

        - description     Human readable description of the operation.

        - returnValueType Optional definition of the return value for the operation. The return value
                          could be a business object or a simple status code. The return value data type
                          has to be defined in the logical service data model.

        - parameterTypes  Definition of one or more parameters for the operation. This could be business
                          objects or simple types. Parameters have to be defined in the logical
                          service data model.
      </documentation>
    </annotation>
    <all>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="returnValueType" type="ServiceSpecificationSchema:ValueTypeDataModelMapping" minOccurs="0"
maxOccurs="1"/>
      <element name="parameterTypes" minOccurs="0" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="parameterType" type="ServiceSpecificationSchema:ValueTypeDataModelMapping" minOccurs="1"
maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </all>
  </complexType>
  <complexType name="ValueTypeDataModelMapping">
    <annotation>
      <documentation>
```

```
        Definition of a data type by providing a reference into the logical service data model.
        A value type data model mapping is used either in a service operation parameter or return value.

        Elements of a valueTypeDataModelMapping are:

        - typeReference  Reference to the logical service data model.
                         It references a type (or element, though type is preferred) in the logical
                         service model by the type's name attribute. References are enforced using
                         XSD unique key and keyrefs.
      </documentation>
    </annotation>
    <all>
      <element name="typeReference" type="string" minOccurs="1" maxOccurs="1"/>
    </all>
  </complexType>
  <complexType name="ServiceDataModel">
    <annotation>
      <documentation>
        The serviceDataModel is a logical model. It is formally described in XSD to achieve interoperability
        and decouple it from implementing physical data models described in e.g. SOAP or REST.
        The model can either be described in-line, or existing schemata can be imported.

        Elements of a serviceDataModel are:

         - definitionAsXSD  The definition of the service data model described in XSD.
      </documentation>
    </annotation>
    <all>
      <element name="definitionAsXSD" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <any namespace="http://www.w3.org/2001/XMLSchema" processContents="lax"/>
          </sequence>
        </complexType>
      </element>
    </all>
  </complexType>
</schema>
```

Appendix B Service Design Description Schema

For the ServiceBaseTypesSchema.xsd see Appendix A.

ServiceDesignSchema.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ServiceDesignSchema="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceDesignSchema.xsd"
xmlns:ServiceSpecificationSchema="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceSpecificationSchema.xsd" targetNamespace="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceDesignSchema.xsd" version="1.0" xml:lang="EN">
  <import namespace="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceSpecificationSchema.xsd"
schemaLocation="ServiceBaseTypesSchema.xsd"/>
  <annotation>
    <documentation>
                Authors:
                    EfficienSea2 WP 3 Partners
                        Thomas Lutz
                        Christoph Rihacek
                        Josef Jahn
                        Hubert Künig
    </documentation>
  </annotation>
  <annotation>
    <documentation>
                This formal description of a service technical design is intentionally kept
                simple and plain.
                For all described objects lots of more attributes and related objects could
                be added and standardized, but in order to get as much adoption as possible,
                the entry barrier should be low, and therefore quite some aspects have been left out.
    </documentation>
  </annotation>
  <element name="serviceDesign" type="ServiceDesignSchema:ServiceDesign">
    <annotation>
      <documentation>
        The root element of a service technical design.
        Please refer to the type serviceDesign for details.
      </documentation>
    </annotation>
  </element>
  <complexType name="ServiceDesign">
    <annotation>
      <documentation>
          A service technical design description.

          Elements of a serviceDesign are:

          - name        The human readable name of the service design The name should be
                         at maximum a one line brief label. Newer versions of the same service
                        design should adopt the same name.

          - id           Globally unique identification of the service design Newer versions of
                         the same service design shall adopt the same id.

          - version     Version of the service design. A service design is uniquely identified
                          by its id and version. Any change in the service physical data model
                          or in the service specification reference requires a new version of the
                          service design.

          - status         Status of the service design. One of the values 'provisional', 'released',
                          'deprecated', 'deleted'.

          - description    A human readable short description of the service design. The
                          description shall contain an abstract of what a service implementation
                          actually does..

          - designsServiceSpecifications
                          Refers to service specification(s) that is/are realised
                          by this service design. As a minimum, one service specification shall
                          be referenced. One service design may realise several service
                          (either different versions of one specification, or even
                          different specifications)..

          - offersTransport    Refers to transport technologies offered by the service design. At
                          least one reference shall be provided.

          - designedBy     Mandatory reference to information about the author of the service design.
```

```
          - servicePhysicalDataModel
                          Mandatory reference to the service physical data model description.
      </documentation>
    </annotation>
    <all>
      <element name="id" type="ServiceSpecificationSchema:ServiceIdentifier" minOccurs="1" maxOccurs="1"/>
      <element name="version" type="ServiceSpecificationSchema:ServiceVersion" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="status" type="ServiceSpecificationSchema:ServiceStatus" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="offersTransport" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="offersTransport" type="ServiceDesignSchema:Transport" minOccurs="1" maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="designsServiceSpecifications" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="designsServiceSpecifications" type="ServiceDesignSchema:ServiceSpecificationReference"
minOccurs="1" maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="designedBy" type="ServiceDesignSchema:VendorInfo" minOccurs="1" maxOccurs="1"/>
      <element name="servicePhysicalDataModel" type="ServiceDesignSchema:ServicePhysicalDataModel" minOccurs="1"
maxOccurs="1"/>
    </all>
  </complexType>
  <complexType name="ServiceSpecificationReference">
    <annotation>
      <documentation>
                A reference to the service specification that  is realised by  the service design
                Has the id and the version of the respective service specification.

                Elements of a ServiceSpecificationReference are:

                - id      Identification of the service specification realised by the service design.

                - version Version of the service specification realised by the service design.
      </documentation>
    </annotation>
    <all>
      <element name="id" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="version" type="string" minOccurs="1" maxOccurs="1"/>
    </all>
  </complexType>
  <complexType name="Transport">
    <annotation>
      <documentation>
                Definition of the transport protocol used by the service design.

                Elements of a transport are:

                - name        Human readable name.

                - description Human readable description of the transport protocol used by the service design.

                - protocol    A non-formal string representation of the transport (e.g. http/rest, http/soap,.. )
                              that provides enough information to a service consumer to be able to connect.
      </documentation>
    </annotation>
    <sequence>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="protocol" type="string" minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>
  <complexType name="VendorInfo">
    <annotation>
      <documentation>
                Describes the vendor providing the service design.

                - id            Unique identification of the vendor.

                - name          Human readable vendor name. The name shall be no longer than one line.

                - description   Human readable description of the vendor.

                - contactInfo   Human readable contact information of the vendor.

                - isCommercial  Optional indication on the commercial status of the vendor.
            </documentation>
```

```
    </annotation>
    <sequence>
      <element name="id" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="contactInfo" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="isCommercial" type="boolean" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
  <complexType name="ServicePhysicalDataModel">
    <annotation>
      <documentation>
                The ServicePhysicalDataModel describes the data model for the service design.
                The ServicePhysicalDataModel describes in detail all the data structures being
                actually exchanged when service consumers interact with a service instance that
                implements this design.

                - name        Human readable model name. The name shall be no longer than one line.

                - description Human readable description of the model.

                - model       The model can e.g. be a WSDL file, a JSON API, or the like. It is recommended to
                              wrap the model in a CDATA section, and provide enough information in the name and
                              description to make clear how to deal with the content in model.

                - modelType   The modelType should contain e.g. an abbreviation that indicates what technology is
                              used to describe the model. E.g. WSDL, JSON.
            </documentation>
    </annotation>
    <all>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="modelType" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="model" type="string" minOccurs="1" maxOccurs="1"/>
    </all>
  </complexType>
</schema>
```

Appendix C Service Instance Description Schema

For the ServiceBaseTypesSchema.xsd see Appendix A.

ServiceInstanceSchema.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ServiceInstanceSchema="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceInstanceSchema.xsd"
xmlns:ServiceSpecificationSchema="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceSpecificationSchema.xsd" targetNamespace="http://efficiensea2.org/maritime-cloud/service-
registry/v1/ServiceInstanceSchema.xsd" version="1.0" xml:lang="EN">
  <import namespace="http://efficiensea2.org/maritime-cloud/service-registry/v1/ServiceSpecificationSchema.xsd"
schemaLocation="ServiceBaseTypesSchema.xsd"/>
  <annotation>
    <documentation>
                Authors:
                    EfficienSea2 WP 3 Partners
                        Thomas Lutz
                        Christoph Rihacek
                        Josef Jahn
                        Hubert Künig
    </documentation>
  </annotation>
  <annotation>
    <documentation>
        This formal description of a service instance is intentionally kept
        simple and plain.
        For all described objects lots of more attributes and related objects could
        be added and standardized, but in order to get as much adoption as possible
        the entry barrier should be low and therefore quite some aspects like
        e.g. when a service instance would be available have been left out.
     </documentation>
  </annotation>
  <element name="serviceInstance" type="ServiceInstanceSchema:ServiceInstance">
    <annotation>
      <documentation>
          The root element of a service instance.
          Please refer to the type serviceInstance for details.
        </documentation>
    </annotation>
  </element>
  <complexType name="ServiceInstance">
    <annotation>
      <documentation>
        A service instance description. One service implementation may be deployed
        at several places by same or different service providers; each such deployment
        represents a different service instance.

          Elements of a serviceInstance are:

                - name              The human readable name of the service instance. The service name should be
                                        at maximum a one line brief label. Newer versions of the same service
                                        specification should adopt the same name.

                - id                    Globally unique identification of the service instance. Newer versions of
                                        the same service instance shall adopt the same id.

                - version            Version of the service instance. A service specification is uniquely
                                        identified by its id and version. Any change in the service instance data model
                                        or in the service specification reference requires a new version of the
                                        service instance.

                - status       Status of the service instance. One of the values 'provisional', 'released',
                                      'deprecated', 'deleted'.

                - description      A human readable short description of the service instance. The description
                                        shall contain an abstract of what a service implementation actually does.

                - keywords        A list of keywords associated to the service.

                - URL                URL that describes where the service endpoint is located

                - requiresAuthorization
                                        Indicates whether authorization is required or not.

                - implementsServiceDesign
                                        Refers to the service design that is implemented by this service instance.
                                        Exactly one service design shall be referenced..
```

```
                - serviceLevel    Refers to the definition of the service level fulfilled by the service instance.
                                  Exactly one service level definition shall be provided.

                - coversAreas     Mandatory reference to the geographical area covered by the service instance.
                                  Note that the content of the CoverageArea defaults to the whole world, if
                                  its content is empty.

                - producedBy      Optional reference to information about the producer of the service
                                                                                implementation

                - providedBy      Mandatory reference to information about the service provider of the service
                                  instance.
            </documentation>
    </annotation>
    <all>
      <element name="id" type="ServiceSpecificationSchema:ServiceIdentifier" minOccurs="1" maxOccurs="1"/>
      <element name="version" type="ServiceSpecificationSchema:ServiceVersion" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="status" type="ServiceSpecificationSchema:ServiceStatus" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="keywords" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="URL" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="requiresAuthorization" type="boolean" minOccurs="1" maxOccurs="1"/>
      <element name="offersServiceLevel" type="ServiceInstanceSchema:ServiceLevel" minOccurs="1" maxOccurs="1"/>
      <element name="coversAreas" minOccurs="1" maxOccurs="1">
        <complexType>
          <sequence>
            <element name="coversArea" type="ServiceInstanceSchema:CoverageArea" minOccurs="1" maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="implementsServiceDesign" type="ServiceInstanceSchema:ServiceDesignReference" minOccurs="1"
maxOccurs="1"/>
      <element name="producedBy" type="ServiceSpecificationSchema:VendorInfo" minOccurs="1" maxOccurs="1"/>
      <element name="providedBy" type="ServiceSpecificationSchema:VendorInfo" minOccurs="1" maxOccurs="1"/>
    </all>
  </complexType>
  <complexType name="ServiceDesignReference">
    <annotation>
      <documentation>
                A reference to the service design that is implemented by the service instance.
                Has the id and the version of the respective service design.

                Elements of a ServiceDesignReference are:

                - id      Identification of the service design implemented by the service instance.

                - version Version of the service design implemented by the service instance.
            </documentation>
    </annotation>
    <all>
      <element name="id" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="version" type="string" minOccurs="1" maxOccurs="1"/>
    </all>
  </complexType>
  <complexType name="ServiceLevel">
    <annotation>
      <documentation>
                Defines the service availability level.

                Elements of a serviceLevel are:

                - name         Human readable service level name. The name shall be no longer than one line.

                - description  Human readable description of the service level

                - availability Indicates the guaranteed availability of the service in %, (e.g. 99.9).
            </documentation>
    </annotation>
    <sequence>
      <element name="availability" type="float" minOccurs="1" maxOccurs="1"/>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>
  <complexType name="CoverageArea">
    <annotation>
      <documentation>
                Defines a geographical area from which the service instance is accessible.

                Elements of a coverage area are:

                - name          Human readable name of the coverage area, e.g. a well-known name
                                like "Bermuda Triangle". The name shall be no longer than one line.
```

```
                - description   Human readable description of the coverage area.

                - geometryAsWKT A polygon described in WKT (Well Known Text) with coordinates in
                                coordinate reference system EPSG:4326,
                                e.g. POLYGON(LON1 LAT1, LON2 LAT2, LON3, LAT3, LON1 LAT1).
                                If the element is empty, the default is the whole world.
            </documentation>
    </annotation>
    <all>
      <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="description" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="geometryAsWKT" type="string" default="POLYGON(-180 -90, 180 -90, 180 90, -180 90, -180 -90"
minOccurs="0" maxOccurs="1"/>
    </all>
  </complexType>
</schema>
```

Appendix D Related Work

This annex provides an overview of standards and products in the field of service registration and service specification. When producing this document, the standards/technologies listed below have been studied in order to check whether and to which extent they could be adopted for the EfficienSea project.

D.1 Summary

Several standards and products have been investigated as candidates for the EfficienSea service specification and service registry. Results show that some of them are quite old and outdated; others look promising by providing appropriate features, but come with high complexity.

A major goal for the EfficienSea service specification framework is to foster interoperability by supporting a standardized way of specifying services and publishing service specifications. This goal requires more than “just a service registry” that allows to register any kinds of service instances or service instance access points. On the other hand, the intended solution aimed at keeping entry barriers very low for all stake holders in the EfficienSea and maritime cloud context – therefore a basic framework with low complexity was preferred.

Consequently, none of the below listed standards/technologies/products was deemed suitable to be used out of the box for fulfilling the needs of EfficienSea. In all cases a significant amount of work would be required to bring these technologies/products to a level appropriate for EfficienSea. This necessary work (including re-interpretation and adaptation of specifications, customizing of products, addition of missing features, reduction of superfluous features, etc.) was considered too complex and it was expected that the results would not reach the above mentioned low entry barriers.

As a result, a mini-framework for service specification has been elaborated, as described in this document. This framework is inspired by the below mentioned technologies/products, but does not explicitly make use of them. The basic framework described in this document is very open, allowing the specification of any kind of services (not only software services), including the description of high level specifications down to individual instances – at the same time the framework remains on a reasonable simple level. Future evolution and fine-tuning of the framework is expected to happen as soon as it is going to be used in practice.

D.2 UDDI

UDDI (Universal Description, Discovery and Integration) is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet. UDDI has its origins in the early 2000 years.

However, UDDI has never been widely adopted and seems to be quite outdated today. Leading companies (e.g., IBM, Microsoft, SAP) who were using UDDI in their product portfolios have dropped this already some years ago.

For these reasons, UDDI was not an option for the service registry work in this project.

D.3 ebXML

While UDDI was developed by an enterprise consortium, ebXML (Electronic Business XML) was developed nearly at the same time, sponsored by United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and OASIS. The ebXML Registry specification is actually comprised of two specifications – ebXML Registry Information Model and ebXML Registry Services.

The latest "news" on the ebXML web page (http://www.ebxml.org/) date back to the year 2006 – this means that also ebXML is already outdated and has therefore not been further analysed for this project.

D.4 JAXR

JAXR (Java API for XML Registries) provides an API for a set of distributed registry services that enables business-to-business integration between business enterprises. JAXR enables Java software programmers to use a single, easy-to-use abstraction API to access a variety of XML registries (e.g., UDDI, ebXML). A unified JAXR information model describes content and metadata within XML registries.

Also JAXR is already outdated and, e.g., already deprecated from Java Enterprise Edition 6.

D.5 jUDDI

jUDDI is an open source Java implementation of the OASIS Universal Description, Discovery, and Integration (UDDI) specification for (Web) Services. The Apache jUDDI project (see https://juddi.apache.org/) includes Scout. Scout is an implementation of the JSR 93 - JavaTM API for XML Registries 1.0 (JAXR).

D.6 S-RAMP

The OASIS S-RAMP (SOA Repository Artifact Model and Protocol) specification defines a SOA repository artefact data model together with bindings that describe the syntax for interacting with a SOA repository (see https://www.oasis-open.org/committees/s-ramp/charter.php).

The S-RAMP specification defines a quite complex framework for supporting the various activities across the life cycle of SOA artefacts. The purpose of S-RAMP is to define a common data model for SOA repositories as well as an interaction protocol to facilitate the use of common tooling and sharing of data. In addition, it defines the interaction with a compliant repository for create, read, update, delete and query operations.

The S-RAMP specification is not just a registry, but also contains a repository for the various SOA artefacts.

The 0.9 release of the S-RAMP specification dates back to 2010, while S-RAMP 1.0 was released in 2013. This means, compared to the specifications described so far, S-RAMP is quite new and it looks fairly promising. Nevertheless, due to the complexity of the S-RAMP specification and due to the fact that S-RAMP is concentrated on SOAP style services and does not directly support REST style services, S-RAMP has not been adopted for the EfficienSea Service Specification and Service Registry specification.

D.7 WSO2 Governance Registry

The WSO2 Governance Registry (see http://wso2.com/products/governance-registry/) provides features like a “Registry & Repository for Anything”, “Lifecycle management”, a “Framework for Governing Anything”, “SOA Governance”, “Design-time Governance”, “Run-time Governance”.

All in all, the product seems to provide much more than what is needed in the context of the EfficienSea procject – leading to quite high complexity.