Open Geospatial Consortium Inc.

Date: 2007-7-10

Reference number of this OGC® project document: **OGC 07-024**

Version: 2.0

Category: OGC® Discussion Paper

Editor: **Thomas Usländer (Ed.)**

Reference Model for the ORCHESTRA Architecture (RM-OA) V2

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type:	OGC® Discussion Paper
Document subtype:	OGC
Document stage:	Draft
Document language:	English

Preamble to the "Reference Model for the Orchestra Architecture (RM-OA)"

This document specifies the Reference Model for the ORCHESTRA Architecture (RM-OA). It is an extension of the OGC Reference Model and contains a specification framework for the design of geospatial service-oriented architectures and service networks. The RM-OA comprises the generic aspects of service-oriented architectures, i.e., those aspects that are independent of the risk management domain and thus applicable to other application domains. The ORCHESTRA Architecture is a platform-neutral (abstract) specification of the informational and functional aspects of service networks taking into account and evolving out of architectural standards and service specifications of ISO, OGC, W3C and OASIS. The target audience of the RM-OA comprise system architects, information modellers and system developers.

The present V2 of the RM-OA extends V1 (OGC 05-107) in the following points:

- inclusion of service meta-modelling into the ORCHESTRA Meta-model leading to a coherent meta-model for feature, interface and service types as an extension of the General Feature Model
- update and refinement of the service descriptions
- preliminary specification of the engineering and technology viewpoint
- conceptual meta-information model (annex A3)
- rules for the specification of meta-information models (annex B1)

For the ORCHESTRA abstract service specifications see http://www.eu-orchestra.org .

FP6-511678

ORCHESTRA

Open Architecture and Spatial Data Infrastructure for Risk Management

Integrated Project

Priority 2.3.2.9 Improving Risk Management

Reference Model for the ORCHESTRA Architecture (RM-OA)

Deliverable D3.2.3 RM-OA Version 2

Date: 2007-01-31

Revision: 2.0

Start date of the ORCHESTRA project:	2004-09-01
Duration of the ORCHESTRA project:	3 years
Organisation name of lead contractor for this deliverable:	Fraunhofer IITB

Document Control Page

Title	Reference Model for the ORCHESTRA Architecture (RM-OA) D3.2.3: RM-OA Version 2 (Rev. 2.0)	
Creator	Thomas Usländer, Fraunhofer IITB (Ed.) e-mail: thomas.uslaender@iitb.fraunhofer.de	
Subject	ORCHESTRA Architecture Design	
Description	This document specifies the Reference Model for the ORCHESTRA Architecture (RM-OA). It contains a platform-neutral specification of the ORCHESTRA Architecture and a specification framework for the design of ORCHESTRA-compliant service networks across all viewpoints.	
Publisher	ORCHESTRA consortium	
Contributor	Bernard, Lars	Joint Research Centre - IES
	Bügel, Ulrich	Fraunhofer IITB
	Corabœuf, Damien	BRGM
	Cooper, Michael	ETH Zürich
	Denzer, Ralf	Environmental Informatics Group
	Dihé, Pascal	Environmental Informatics Group
	Ecker, Severin	ARC Seibersdorf Research
	Fischer, Julian	Environmental Informatics Group
	Friis-Christensen, Anders	Joint Research Centre - IES
	Frysinger, Steve	Environmental Informatics Group
	Goodwin, John	Ordnance Survey
	Güttler, Reiner	Environmental Informatics Group
	Havlik, Denis	ARC Seibersdorf Research
	Hilbring, Désirée	Fraunhofer IITB
	Hofmann, Thomas	Environmental Informatics Group
	Holt, Ian	Ordnance Survey
	Humer, Heinrich	ARC Seibersdorf Research
	Iosifescu Enescu, Ionut	ETH Zürich
	Kunz, Wolfgang	Environmental Informatics Group
	Kutschera, Peter	ARC Seibersdorf Research
	Lorenzo, José	Atos Origin Spain
	Lutz, Michael	Joint Research Centre - IES
	Ma, Wenjie	Environmental Informatics Group
	Pichler, Guenther	Open Geospatial Consortium Europe
	Portele, Clemens	Open Geospatial Consortium Europe
	Robida, Francois	BRGM
	Schimak, Gerald	ARC Seibersdorf Research
	Schlobinski, Sascha	Environmental Informatics Group
	Schmieder, Martin	Fraunhofer IITB
	Serrano, Jean-Jacques	BRGM
	Sykora, Peter	ETH Zürich
	Usländer, Thomas	Fraunhofer IITB

Date	2007-01-31
Type	Text
Format	application/msword
Identifier	ORCHESTRA Portal: SP3 / SP3 Quality Assurance / 09: D3.2.3 / 06: D3.2.3 RM-OA V2 (2.0) – published version
Source	Not applicable
Language	en-GB.
Relation	none
Coverage	Not applicable
Rights	© 2007 ORCHESTRA Consortium The ORCHESTRA project is an Integrated Project (FP6-511678) funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).
Deliverable number	D3.2.3
Audience	☒ public ☐ restricted ☐ internal

Revision History

Revision	Date	Sections Changed	Description
1.0	2004-12-23	all	Final draft of D3.2.1 submitted to the QA process
1.1	2005-01-26	Official sections of D3.2.1	Inclusion of QA comments by SP3 leader on V1.0 Renaming of user roles
1.2	2005-02-04	Official sections of D3.2.1	Resolution of open points Harmonisation of terms
1.3	2005-02-16	Official sections of D3.2.1	Results of WP3.2 meeting on 2005-02-09
1.4	2005-03-24	Official sections of D3.2.1	Inclusion of QA comments by SP3 leader on V1.3, results of discussion within SP3
1.5	2005-04-22	Sections 5.3 and 5.4	Inclusion of QA comments by SP3 leader on V1.3, results of discussion within SP3
1.6	2005-06-28	all	Change of Information and Service Viewpoint description according to discussions within SP3
1.7	2005-07-13	all	Inclusion of partner contributions to Information and Service Viewpoint description and results of discussions within SP3 Move of sections 5.3 and 5.4 to an annex
1.7	2005-07-14	all	QA by SP3 leader
1.8	2005-07-22	all	Update following QA review by SP3 leader
1.9	2005-09-15	all	Update following technical review by the Technical Supervisor
1.10	2005-10-14	all	editorial corrections, update following the ORCHESTRA Annual Technical Review
1.11/1.12	2006-07-21 2006-08-04	all	Major update incorporating the progress, the refinements and the agreements between October 2005 and July 2006 of the ORCHESTRA sub-project 3 List of major changes: • update of the references • new and adapted glossary terms (section 4), e.g. for meta-information, semantics, UAA, service and service type, platform • extended understanding of the engineering viewpoint: inclusion of service characteristics • new and refined design decisions (section 7) - extension of functional domains - OMM partition into information and service part - revision of UAA approach - consideration of meta-information

			- revision of source system integration process - identification of resources • Information Viewpoint (section 8) - removal of source system descriptor - no specific handling of coverages - addition of data types • Service Viewpoint (section 9) - inclusion of service meta-model (OMM-Service) - adapted list of OA/OT Services - update of service description framework - adaptation of nearly all service descriptions • Technology Viewpoint (section 10) - guidelines for the specification of a platform • Engineering Viewpoint (section 11) - description of OSN characteristics - section on identification of OSIs and feature instances • Revision of RM-OA Annex structure
1.13	2006-09-19	all	• updated glossary terms: spatial data infrastructure • update of section 6.2.2.2 “Requirements of the INSPIRE Relationship” • new section 7.2 “The ORCHESTRA Meta-model Approach” • update of the OMM-Service • update of sections about UAA • new section 12.2 “Evolution of the RM-OA” • update of a series of minor architectural decisions taken during the SP3 discussion process • extension of Table 7 to include the relationship to the INSPIRE Network Services
1.14	2006-09-24	all section 11	QA Review by SP3 Leader • minor typing errors corrected
1.15	2006-09-29	all	Corrections made by the editor after the QA Review of SP3 Leader
1.16	2006-10-26	all	Corrections made by the editor after the QA Review of the ORCHESTRA Technical Supervisor
1.17	2007-01-17	all	Incorporation of comments from the 2nd Annual Review and re-structuring of RM-OA for publication
2.0	2007-01-31	all	Editorial changes for publication

Table of Contents

Figures

Tables

1 Executive Summary

Increasing numbers of natural disasters have demonstrated to the European Union the paramount importance of avoiding and mitigating natural hazards in order to protect the environment and citizens. Due to organisational and technological barriers, actors involved in the management of natural or man-made risks cannot cooperate efficiently. In an attempt to solve some of these problems, the European Commission has made “Improving risk management” one of its strategic objectives of the Information Society Technology (IST) research programme. The goal of the integrated project ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) is the design and implementation of an open, service-oriented software architecture as a contribution to overcome the interoperability problems in the domain of multi-risk management.

Public information about the ORCHESTRA project is available under http://www.eu-orchestra.org/.

The present document defines the Reference Model for the ORCHESTRA Architecture (RM-OA). The RM-OA comprises the generic aspects of service-oriented architectures, i.e., those aspects that are independent of the risk management domain and thus applicable to other application domains.

Based on a glossary of architectural terms, the RM-OA provides a specification framework for system architects, information modellers and system developers. The ORCHESTRA Architecture is a platform-neutral (abstract) specification of the informational and functional aspects of service networks taking into account and evolving out of architectural standards and service specifications of ISO, OGC, W3C and OASIS.

The structure of the RM-OA follows the viewpoints of the ISO/IEC 10746-1 Reference Model for Open Distributed Processing (RM-ODP) in the following manner:

- The RM-OA Enterprise Viewpoint provides a business perspective with respect to other European initiatives such as INSPIRE, GMES and other Integrated Projects. It yields the major architectural requirements, namely the rigorous use of standards where applicable, the independence from technology, the demand for loosely-coupled self-describing components based on a generic infrastructure and the design for change.
- The RM-OA Information Viewpoint provides a specification framework of all categories of information including their thematic, spatial, and temporal characteristics as well as their meta-information. The basic unit is the concept of a feature as an abstraction of a real world phenomenon. In principle, it follows ISO 19109 for the meta-model structure and rules of application schemas, but extends it by the pre-definition of the characteristics of eminent feature types (e.g. documents). As meta-information models are considered to be purpose-specific, the ORCHESTRA Meta-Model enables pluggable application schemas for meta-information. Furthermore, it explicitly considers the integration of data and services of existing systems (source systems) as well as the usage of ontologies.
- The RM-OA Service Viewpoint (in RM-ODP called Computational Viewpoint) specifies types of ORCHESTRA Architecture Services that support the syntactic and semantic interoperability between systems as well as the administration of service instances organised in ORCHESTRA Service Networks. The RM-OA provides textual service descriptions according to a common service description framework and contains an initial description of so-called ORCHESTRA Thematic Support services that facilitate the development of thematic functionality. Furthermore, by means of a meta-model for services on a platform-neutral level, the RM-OA provides rules how to formally specify service types based on interface types as the basic unit of re-usability and how to map them to concrete service platforms.
- The RM-OA Engineering and Technology viewpoints yield the mapping of the application schemas and service specifications to service platforms (e.g. W3C Web Services). Here, the RM-OA just provides guidance for the mapping to a given service platform and specifies engineering options for the design of ORCHESTRA Service Networks. The resulting work lead to platform-specific ORCHESTRA Implementation Specifications that are, however, documented outside of the RM-OA.

RM-OA annexes contain more detailed system requirements, a conceptual meta-information model and default application schemas for meta-information for an initial list of “purposes” (e.g. discovery).

2 Document Structure and Links

2.1 Link to the ORCHESTRA Project Structure

The current document presents the results of the work package 3.2 “Architecture Design” of the ORCHESTRA sub-project 3 “Open Architecture” according to the ORCHESTRA Description of Work (DoW) (ORCH-DoW 2006). The DoW is the technical part of the ORCHESTRA contract with the European Commission.

The objectives of the work package “Architecture Design” are as follows:

- To specify requirements which an ORCHESTRA Architecture for risk management needs to address.
- To design a draft ORCHESTRA Architecture, defining which components in the overall systems are needed, what their functionalities and roles are, and how these components collaborate.
- To serve as the design drawing for the detailed specification of services.
- To further refine the RM-OA during the lifetime of the project.

The work package is structured in three tasks whose goals are specified as follows in the DoW:

- Task 3.2.1 “High level requirements specification”:

  In this task, the abstract high-level requirements of the OA are specified. Issues involve user management and authorisation, quality in the information production chain, trust, availability, fault-tolerance, coordination, management of the OA, security and others. The task specifically addresses high level requirements which today prevent inter-operability. One particular issue will be how the OA collaborates in the crisis phase with crisis management systems. Requirements may also come from the WIN project.

  The results of this task have led to the deliverable D3.2.1 “High Level Requirements Specification”.

- Task 3.2.2 Draft architecture design

  In this task, a draft design of the architecture is developed. The design includes a) the clear definition of layers of the OA, b) the definition of required components like registries, catalogues, information and processing services, collaboration components, etc., c) a concept for a systematic approach to how the integration of spatial and non-spatial information and components will work, d) the management view of the overall system, e) the most important interfaces at the conception level (later to be refined in WP3.4).

  The results of this task have led to the deliverable D3.2.2: “Draft Architecture Design”. This deliverable is identical with the version V1.10 of the RM-OA (RM-OA 2005) having being published to the Open Geospatial Consortium in 2005 as project document OGC 05-107.

- Task 3.2.3 Refined architecture design

  Based on feedback from the other subprojects, and in particular in collaboration with information providers within the project, refined versions of the architecture design will be elaborated until month 32. Selected versions (those which are the most appropriate ones) of RM-OA 2.x, 3.x and 4.x will be published to the community, in particular to OGC.

  The results of this task lead to the deliverable D3.2.3, D3.2.4 and D3.2.5 corresponding to further RM-OA versions.

Sources of requirements for the design of the RM-OA are:

- Results from the ORCHESTRA sub-project 2 “User Requirements and Policy Watch”

  The user requirements as specified in (ORCH-D2.1 2006) constitute the major source of requirements for the RM-OA.

- The DoW as a basic legal reference to be fulfilled.

- The extensive experience of the ORCHESTRA consortium partners in sub-project 3 “Open Architecture” in the development of environmental information and risk management systems.
- Implementation experiences and refined requirements from the ORCHESTRA sub-project 4 “Risk Management Services” which delivers pilot applications of the ORCHESTRA Architecture the will be continuously incorporated into the RM-OA and mapped to system requirements according to the iterative process of the RM-OA design.

2.2 Link to the RM-OA Annexes and ORCHESTRA Deliverables

The RM-OA encompasses the results of the ORCHESTRA sub-project 3 and the related deliverables as annexes.

Annex	Name	ORCHESTRA Deliverable/ WP
A	High Level Requirements Specification	
A1	Development dimensions	D3.2.1 of WP3.2
A2	System requirements	D3.2.1 of WP3.2
A3	Conceptual Meta-information model	D3.3.1 of WP3.3
B	Specification of ORCHESTRA Meta-information Models	
B1	RM-OA rules for OAS-MI	D3.3.2 of WP3.3

Table 1: Overview about the RM-OA Annexes

.

3 Introduction

3.1 Scope

This document specifies the Reference Model for the ORCHESTRA Architecture (RM-OA). It contains a specification framework for the design of ORCHESTRA-compliant service networks and provides a platform-neutral specification of the information and service viewpoints.

The RM-OA specification is structured according to the viewpoints of the Reference Model for Open Distributed Processing (RM-ODP) as defined in ISO/IEC 10746-1:1998 (E), with some modifications reflecting both ORCHESTRA needs and the design objective of a service network based on loosely-coupled components.

The RM-OA document is divided into the following sections:

- Section 4 "Glossary" provides a definition of the architectural terms used in the RM-OA.
- Section 5 "Process of the ORCHESTRA Architectural Design" describes the ORCHESTRA Reference Model resulting from the mapping of the ISO/IEC 10746-1 Reference Model for Open Distributed Processing (RM-ODP) to the ORCHESTRA architectural design process.
- Section 6 "Enterprise Viewpoint" provides a business perspective and summarises the architectural requirements for the design of ORCHESTRA-compliant service networks. The architectural requirements are motivated in detail in an argumentation chain in Annex A2 of the RM-OA..
- Section 7 "Design Decisions of the ORCHESTRA Architecture" summarises basic design decisions for the ORCHESTRA Architecture as an introduction to the architecture specification in the following section.
- Section 8 "Information Viewpoint" provides a specification framework of all categories of information dealt with by the ORCHESTRA Architecture, including their thematic, spatial, temporal characteristics as well as their meta-information.
- Section 9 "Service Viewpoint" provides a specification framework for ORCHESTRA Services. Furthermore, it contains descriptions for the services that support the syntactic and semantic interoperability between services, applications and systems as well as the administration of ORCHESTRA service networks. The description distinguishes between ORCHESTRA Architecture services that provide the generic, i.e. application-domain independent part of a service network, and ORCHESTRA Thematic Service that support particular application-domains, in the case of ORCHESTRA the risk management domain.
- Section 10 "Technology Viewpoint" describes general guidelines to be considered when specifying a platform as a service infrastructure upon which the platform-neutral ORCHESTRA Architecture may be mapped.
- Section 11 "Engineering Viewpoint" describes topics to be considered by designers of ORCHESTRA Service Networks, in particular characteristics of ORCHESTRA Service Networks and policies w.r.t. naming of service and feature instances, discovery, user management, access control and authentication and service administration.
- Section 12 "Conclusion" lists the major aspects where the RM-OA deviates from standards. Furthermore, it provides an outlook for issues to be tackled in future RM-OA versions.

The RM-OA core document is associated with a list of annexes that provide more background information and more refined specifications. See Table 1 in section 2.2.

3.2 Intended Audience

System architects, information modellers and system developers when designing service networks taking into account relevant standards from ISO, OGC, W3C and OASIS.

3.3 References

The following references are used as background documents for the RM-OA. They are categorised in normative references (i.e. ISO Standards or respective drafts) and other technical or scientific documents and books.

3.3.1 Normative references

ISO/IEC 10746-1:1998 (E). Information technology - Open Distributed Processing - Reference model

ISO/IEC 10746-2:1996 (E). Information technology - Open Distributed Processing - Foundations

ISO/IEC TR 14252:1996. Information technology - Guide to the POSIX Open System Environment

ISO 19101:2004(E). Geographic information -- Reference model

ISO/PDTS 19103. Geographic information -- Conceptual schema language

ISO 19107:2004(E). Geographic information -- Spatial schema

ISO 19108:2004(E) Geographic information -- Temporal schema

ISO/FDIS 19109:2003. Text for FDIS 19109 Geographic information -- Rules for application schema, as sent to the ISO Central Secretariat for issuing as Final Draft International Standard

ISO 19111:2003(E). Geographic information -- Spatial referencing by coordinates

ISO 19112:2003(E). Geographic information -- Spatial referencing by geographic identifiers

ISO 19115:2004(E). Geographic Information -- Metadata

ISO 19119:2005. Geographic Information -- Services (see also “The OpenGIS Abstract Specification - Topic 12: OpenGIS Service Architecture” under http://www.opengis.org/docs/02-112.pdf)

ISO 19119:2005(E). Geographic Information -- Services

ISO 19123:2005(E). Geographic Information -- Schema for coverage geometry and functions

ISO 19125-1:2004(E). Geographic Information -- Simple feature access -- Part 1: Common architecture

ISO/DIS 19136 Geographic Information -- Geography Markup Language (GML)

3.3.2 Documents and Books

COM (2004) 516 final. Proposal for a DIRECTIVE OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL establishing an infrastructure for spatial information in the Community (INSPIRE). 2004/0175 (COD), http://inspire.jrc.it/proposal/EN.pdf

Dufourmont, H., Annoni, A., De Groof, H. (2004). INSPIRE - work programme Preparatory Phase 2005 – 2006. Publisher: ESTAT-JRC-ENV. Identifier: rhd040705WP4A_v4.5.3.doc, http://inspire.jrc.it/reports/rhd040705WP4A_v4.5.3_final-2.pdf

Egenhofer, M.J. (1989). A Formal Definition of Binary Topological Relationships. 3rd International Conference on Foundations of Data Organization and Algorithms: 457-472

GMES (27004). Global Monitoring for Environment and Security (GMES): Final Report for the GMES Initial Period (2001-2003) http://www.gmes.info/action_plan/index.html

OASIS (2006) OASIS WS-Trust 1.3 Committee Draft 01. 06 September 2006 http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-spec-cd-01.html

OMG (2006). “Software Services Profile and Metamodel”. Request For Proposal OMG Document: soa/2006-09-01

ORCH-D2.1 (2006). D2.1 Final Report on User Requirements V1.4. Restricted Deliverable D2.1 Integrated Project 511678 ORCHESTRA. Editor: BRGM. 5 October 2006

ORCH-D2.3.5 (2006). Knowledge Modelling Final Report. Internal Deliverable D2.3.5 Integrated Project 511678 ORCHESTRA. Editor: Ordnance Survey. Version 1.0. 28 February 2006

ORCH-D2.4.1 (2005). Report on analysis of existing risk management processes. Deliverable D2.4.1 Integrated Project 511678 ORCHESTRA. Editor: DATAMAT. Revision [final]. 13 June 2005

ORCH-D2.4.2 (2005). Report identifying common service requirements. Deliverable 2.4.2 (2005) Integrated Project 511678 ORCHESTRA. Editor: DATAMAT. Revision [final]. 21 December 2005

ORCH-AbstrServ (2007). WP3.4 OA Service Abstract Specifications. Deliverables D3.4.x Integrated Project 511678 ORCHESTRA. Editor: Environmental Informatics Group (EIG). January 2007

ORCH-DoW (2006). Integrated Project 511678 ORCHESTRA: “Annex 1 – Description of Work”. 6th Framework Programme IST Priority 2.3.2.9 Improving Risk Management. 11 May 2006

ORCH-ImplServ (2007). WP3.6 OA Service Implementation Specifications. Deliverables D3.6.x. Integrated Project 511678 ORCHESTRA. Editor: Environmental Informatics Group (EIG). 2007 (to be published)

OGC (2003). Open Geospatial Consortium Doc. No. 03-040. OGC Reference Model, Version 0.1.2 , 2003-03-04 http://portal.opengis.org/files/?artifact_id=3836

OGC (2006) Open Geospatial Consortium Discussion paper 05-087r3 “Observations and Measurements”, 2006-02-24, http://portal.opengeospatial.org/files/?artifact_id=14034

Pollock, J.T., Hodgson, R. (2004). Adaptive Information. ISBN 0-471-48854-2. Wiley 2004

Powell, D. (Ed.) (1991). Delta-4: A Generic Architecture for Dependable Distributed Computing. Research Reports ESPRIT. Project 818/2252 Delta-4 Vol.1. ISBN 3-540-54985-4 Springer-Verlag 1991

RM-OA (2005) Usländer, T. (Ed.) Reference Model for the ORCHESTRA Architecture Version 1.10. Deliverable D3.2.2 of the ORCHESTRA Consortium, OGC Discussion Paper OGC 05-107 - https://portal.opengeospatial.org/files/?artifact_id=12574, October 2005

SOA-RM (2006). OASIS Reference Model for Service Oriented Architecture 1.0. Committee Specification 1, 2 August 2006. http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

Studer, R.; Benjamins, V. R.; Fensel, D.: Knowledge engineering: Principles and methods. Data and Knowledge Engineering (DKE), 25(1-2):161-197, 1998.

Tomlin, C.D. (1990). Geographic Information Systems and Cartographic Modeling (Prentice-Hall)

W3C (2003). QoS for Web Services: Requirements and Possible Approaches. W3C Working Group Note, 25 November 2003, http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

W3C (2004). Web Services Architecture. W3C Working Group Note 11 February 2004. http://www.w3.org/TR/ws-arch/

4 Glossary

The glossary provides the coherent terminological framework used in the RM-OA.

4.1 Abbreviations

AAA	Authentication, Authorisation, and Accounting
ACID	Atomicity, Consistency, Isolation, and Durability
CEN	Comité Européen de Normalisation (European Committee for Standardization)
CSL	Conceptual Schema Language
DIS	Draft International Standard
DoW	ORCHESTRA Description of Work
DRM	Digital Rights Management
EBAC	Expression-based access control
EC	European Commission
ESA	European Space Agency
ESDI	European Spatial Data Infrastructure
GeoDRM	Digital Rights Management related to Geographic Information
GFM	General Feature Model
GMES	Global Monitoring for Environment and Security
HCI	Human-Computer Interaction
INSPIRE	Infrastructure for Spatial Information in Europe
ID	Identifier
IS	International Standard
ISO	International Standardization Organisation
IST	Information Society Technology
LMO	Legally Mandated Organisations
OA	ORCHESTRA Architecture
OA Service	ORCHESTRA Architecture Service
OT Service	ORCHESTRA Thematic Service
OAA	ORCHESTRA Application Architecture
OAS	ORCHESTRA Application Schema
OAS-MI	ORCHESTRA Application Schema for Meta-information
OFS	ORCHESTRA Feature Set
OASIS	1) IST FP-6 project: Open Advanced System for Improved Crisis Management 2) Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OIS	ORCHESTRA Implementation Specification
OMG	Object Management Group
OMM	ORCHESTRA Meta-model

ORCHESTRA	Open Architecture and Spatial Data Infrastructure for Risk Management
OSC	ORCHESTRA Service Component
OSI	ORCHESTRA Service Instance
OSN	ORCHESTRA Service Network
OWL	Web Ontology Language
OWL-S	Web service ontology based on OWL
RBAC	Role-based access control
RDF	Resource Description Framework
RM	Risk Management
RM-OA	Reference Model for the ORCHESTRA Architecture
RM-ODP	Reference Model for Open Distributed Processing
SOA	Service-oriented Architecture
SOA-RM	(OASIS) Reference Model for Service Oriented Architecture
SDI	Spatial Data Infrastructure
SDIC	Spatial Data Interest Communities
UAA	User Management, Authentication and Authorisation
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WIN	Wide Information Network for Risk Management
WP	Work package
WSMO	Web Service Modeling Ontology
XSD	XML Schema Definition

4.2 Terms and definitions

ABox

Set of description logics statements about individuals with reference to a TBox (so-called "extensional" knowledge).

Note: An example is: "Katrina" is-instance-of TropicalCyclone.

Access control

Combination of Authentication and Authorisation.

Accounting

Process of gathering information about the usage of resources by subjects.

Application [derived from http://www.opengeospatial.org/resources/?page=glossary]

Use of capabilities, including hardware, software and data, provided by an information system specific to the satisfaction of a set of user requirements in a given application domain.

Application Domain

Integrated set of problems, terms, information and tasks of a specific thematic domain that an application (e.g. an information system or a set of information systems) has to cope with.

Note: One example of an application domain is risk management.

Application Schema [ISO/FDIS 19109:2003]

Conceptual schema for data required by one or more applications.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Authentication

Process of verifying the principal of a certain subject. In other words authentication indicates whether a subject is allowed to use a certain principal .

Within the authentication process a subject proves that it is allowed to act with the corresponding principal . Generally speaking, this proof can depend on a secret that can be, e.g.

- what somebody has (key, smart card, …)
- what somebody knows (password, …)
- what somebody is (biometrical data, …)
- the place somebody resides (certain computer, …)
- the skills of somebody (handmade signature)

The result of an authentication process is called a session.

Authorisation

Process of determining whether a subject is allowed to have the specified types of access to a particular resource. This is done by evaluating applicable access control information contained in a so called authorisation context.

Usually, authorisation is carried out in the context of authentication. Once a subject is authenticated, it may be authorised to perform different types of access.

Catalogue [derived from http://www.opengeospatial.org/resources/?page=glossary]

Collection of entries, each of which describes and points to a feature collection. Catalogues include indexed listings of feature collections, their contents, their coverages, and of meta-information. A catalogue registers the existence, location, and description of feature collections held by an Information Community. Catalogues provide the capability to add and delete entries. A minimum Catalogue will include the name for the feature collection and the locational handle that specifies where these data may be found. Each catalogue is unique to its Information Community.

Component

Hardware component (device) or Software Component.

Conceptual model [ISO/FDIS 19109:2003(E); ISO 19101]

Model that defines concepts of a universe of discourse.

Conceptual schema [ISO/FDIS 19109:2003(E); ISO 19101]

Formal description of a conceptual model.

Coverage [ISO 19123]

Function from a spatial, temporal or spatiotemporal domain to an attribute range. A coverage associates a position within its domain to a record of values of defined data types. Thus, a coverage is a feature with multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.

Description Logics

Family of logic based knowledge representation languages that are a decidable subset of first order logic with well defined semantics and inferencing (problem decision procedures). In Description Logics, a distinction is made between the terminological knowledge (the so-called TBox) and the assertional knowledge (ABox). This distinction is useful for knowledge base modelling and engineering: for modelling it is just natural to distinguish between concepts and individuals; for engineering it helps by separating key inference problems, e.g. classification is related to the TBox, while instance checking is related to the ABox.

Discovery [derived from W3C: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#discovery]

Act of locating a machine-processable description of a resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions.

Engineering viewpoint

Viewpoint of the ORCHESTRA Reference Model that specifies the mapping of the ORCHESTRA service specifications and information models to the chosen service platform and the characteristics of ORCHESTRA Service Networks.

End user

Members of agencies (e.g. civil or environmental protection agencies) or private companies that are involved in an application domain (e.g. risk management) and that use the applications built by the system users according to the ORCHESTRA Architecture.

External Source System

Source system that does not provide its data and functions through an ORCHESTRA-conformant interface.

Feature [derived from ISO 19101]

Abstraction of a real world phenomenon [ISO 19101] perceived in the context of an ORCHESTRA Application.

Note: The ORCHESTRA understanding of a "real world" explicitly comprises hypothetical worlds.

Features may but need not contain geospatial properties. In this general sense, a feature corresponds to an “object” in analysis and design models.

Framework [http://www.opengeospatial.org/resources/?page=glossary]

An information architecture that comprises, in terms of software design, a reusable software template, or skeleton, from which key enabling and supporting services can be selected, configured and integrated with application code.

Gazetteer [http://www.opengeospatial.org/resources/?page=glossary]

A catalogue of toponyms (place names) assigned with geographic references. A gazetteer service retrieves the geometries for one or more features, given their associated well-known feature identifiers (text strings).

Generic

A service is generic, if it is independent of the application domain. A service infrastructure is generic, if it is independent of the application domain and if it can adapt to different organisational structures at different sites, without programming (ideally).

Geospatial [http://www.opengeospatial.org/resources/?page=glossary]

Referring to a location relative to the Earth's surface. “Geospatial” is more precise in many geographic information system contexts than "geographic," because geospatial information is often used in ways that do not involve a graphic representation, or map, of the information.

Implementation [http://www.opengeospatial.org/resources/?page=glossary]

Software package that conforms to a standard or specification. A specific instance of a more generally defined system.

Information Community [http://www.opengeospatial.org/resources/?page=glossary]

A collection of people (a government agency or group of agencies, a profession, a group of researchers in the same discipline, corporate partners cooperating on a project, etc.) who, at least part of the time, share a common digital geographic information language and common spatial feature definitions.

Information viewpoint

Viewpoint of the ORCHESTRA Reference Model that specifies the modelling approach of all categories of information the ORCHESTRA Architecture deals with including their thematic, spatial, temporal characteristics as well as their meta-information.

Interface [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Named set of operations that characterize the behaviour of an entity.

The aggregation of operations in an interface, and the definition of interface, shall be for the purpose of software reusability. The specification of an interface shall include a static portion that includes definition of the operations. The specification of an interface shall include a dynamic portion that includes any restrictions on the order of invoking the operations.

Interoperability [ISO 19119:2005 or OGC; http://www.opengeospatial.org/resources/?page=glossary]

Capability to communicate, execute programs, or transfer data among various functional units in a manner that require the user to have little or no knowledge of the unique characteristics of those units [ISO 2382-1].

Knowledge Base

Store of formal knowledge about identifiable entities of a real or hypothetical world. The entity descriptions are typically instance knowledge or data, or an ABox in terms of Description Logics. In some cases, the knowledge base additionally provides access to the knowledge schema (the TBox corresponding to the ABox). Generally, a knowledge base does not necessarily need to be described by means of a schema: it basically provides a flexible means for identification, representation and interlinking of entities. Compared to a conventional relational database, a knowledge base is more flexible: it may comprise several identifiable sets of entity relationships ("models"), and new models can dynamically be added without the need for redefining the complete database schema. New entities and relations can be inserted at run time (population of the knowledge base).

Note: Knowledge stored in a knowledge base can be retrieved by means of a query language. Compared to a Catalogue and/or a Feature Access Service (see section 9.6.1), the result of these queries is not necessarily a feature collection, e.g. just a boolean value an extreme case. If the knowledge base contains implicitly represented information, e.g. in terms of rules, the quality of the query results may be improved by automatically inferring new knowledge (TBox and/or ABox reasoning).

Loose coupling [W3C; http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#loosecoupling]

Coupling is the dependency between interacting systems. This dependency can be decomposed into real dependency and artificial dependency: Real dependency is the set of features or services that a system consumes from other systems. The real dependency always exists and cannot be reduced. Artificial dependency is the set of factors that a system has to comply with in order to consume the features or services provided by other systems. Typical artificial dependency factors are language dependency, platform dependency, API dependency, etc. Artificial dependency always exists, but it or its cost can be reduced. Loose coupling describes the configuration in which artificial dependency has been reduced to the minimum.

Meta-information

Descriptive information about resources in the universe of discourse. Its structure is given by a meta-information model depending on a particular purpose.

Note: A resource by itself does not necessarily need meta-information. The need for meta-information arises from additional tasks or a particular purpose (like catalogue organisation), where many different resources (services and data objects) must be handled by common methods and therefore have to have/get common attributes and descriptions (like a location or the classification of a book in a library).

Meta-information model

Implementation of a conceptual model for meta-information. It is represented by an ORCHESTRA Application Schema for Meta-information.

Middleware [http://www.opengeospatial.org/resources/?page=glossary]

Software in a distributed computing environment that mediates between clients and servers.

OA Info-structure Service

OA Service that is required to operate an OSN in the sense that it plays an indispensable role in the operation of an OSN.

OA Support Service

OA Service that facilitates the operation of an OSN , e.g. providing an added-value by combining the usage of OA Info-Structure Services.

Ontology [based on (Studer et al 1998)]

Explicit, formal specification of a shared conceptualisation (Studer et al 1998).

It is formal in order to not only make it readable by humans, but also by machines. It is explicit as it is based on a taxonomy specified in terms of concepts, properties (or relationships) and axioms (the "vocabulary"). It is shared in the sense that these specifications are fixed as an agreement set up and shared by a dedicated user community and that it is associated with a particular subject area (domain) or task. It is a conceptualisation as it defines a conceptual schema by abstracting from a real or hypothetical world. Its ultimate purpose is to enable machine understanding which in turn provides the potential for data and service interoperability.

In Description Logics, an ontology describes a TBox; optionally, it may also describe an ABox. The TBox can then be considered to be the schema of the ABox.

Open Architecture [based on (Powell 1991)]

Architecture whose specifications are published and made freely available to interested vendors and users with a view of widespread adoption of the architecture. An open architecture makes use of existing standards where appropriate and possible and otherwise contributes to the evolution of relevant new standards.

Operation [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Specification of a transformation or query that an object may be called to execute. An operation has a name and a list of parameters.

ORCHESTRA Architecture (OA)

Open architecture that comprises the combined generic and platform-neutral specification of the information and service viewpoint as part of the ORCHESTRA Reference Model.

ORCHESTRA Application

Set of software components that together comprise an application based on the usage of ORCHESTRA Services

ORCHESTRA Application Architecture (OAA)

Instantiation of the ORCHESTRA Architecture by inclusion of those thematic aspects that fulfil the purpose and objectives of a given application. The concepts for such an application stem from a particular application domain (e.g. a risk management application).

ORCHESTRA Architecture Service (OA Service)

ORCHESTRA Service that provides a generic, platform-neutral and application-domain independent functionality.

ORCHESTRA Application Schema (OAS) [extending ISO/FDIS 19109:2003]

Conceptual schema for the data required by one or more ORCHESTRA Applications. As such it provides a formal specification that is compliant to the ORCHESTRA Meta-model of the concepts (e.g. feature types), their properties and associations which are relevant for a specific information model in an ORCHESTRA Service Network.

ORCHESTRA Application Schema for Meta-information (OAS-MI)

Form of an ORCHESTRA Application Schema applied to meta-information.

ORCHESTRA Application Implementation Specification (OAIS)

Extension and restriction of an ORCHESTRA Implementation Specification according to the needs of a particular application domain. An OAIS comprises a platform-specific combined specification of a thematic information model and a set of OT Services.

ORCHESTRA Feature Set (OFS)

Set of feature instances following the information model formally specified in an ORCHESTRA Application Schema.

ORCHESTRA Implementation Specification

Combined platform-specific specification of the engineering and technology viewpoints as a result of the mapping of the ORCHESTRA Architecture to a specific platform.

ORCHESTRA Meta-Model (OMM)

Framework of rules for the specification of an ORCHESTRA Application Schema. It is specified in terms of UML classes stereotyped as <<MetaClass>> and associated rules for their instantiation in an ORCHESTRA Application Schema.

ORCHESTRA Reference Model

The ORCHESTRA Reference Model comprises a specification of all RM-ODP viewpoints for the open architecture for risk management. In particular, it encompasses the specification of the ORCHESTRA Architecture and a specification framework for ORCHESTRA Implementation Specifications which are implemented in ORCHESTRA Service Components and deployed in an ORCHESTRA Service Network as ORCHESTRA Service Instances.

ORCHESTRA Service

Service specified as an ORCHESTRA Service Type, implemented as ORCHESTRA Service Component and offered in an ORCHESTRA Service Network by an ORCHESTRA Service Instance.

ORCHESTRA Service Component

Component that provides an external interface of an ORCHESTRA Service according to an ORCHESTRA Implementation Specification.

ORCHESTRA Service Instance

Executing manifestation of an ORCHESTRA Service Component.

ORCHESTRA Service Network

Set of networked hardware components and ORCHESTRA Service Instances that interact in order to serve the objectives of ORCHESTRA Applications. The basic unit within an OSN for the provision of functions are the OSIs.

ORCHESTRA Service Type

Type of an ORCHESTRA Service specified according to the rules of the ORCHESTRA Reference Model.

ORCHESTRA Service Types are functionally classified in ORCHESTRA Architecture Services (OA Services) and ORCHESTRA Thematic Services (OT Services).

ORCHESTRA Source System

Source system that provides its data and functions through an ORCHESTRA-conformant interface. Each ORCHESTRA Source System is associated to at least one External Source System.

ORCHESTRA Thematic Service (OT Service)

ORCHESTRA Service that provides an application domain-specific functionality built on top and by usage of OA Services and/or other OT Services.

Note: An OT Service may but need not be specified in a platform-neutral way.

Purpose (of meta-information)

A purpose of meta-information describes the goal of the usage of the resources.

(Service) Platform

Set of infrastructural means and rules that describe how to specify service interfaces and related information and how to invoke services in a distributed system.

Examples for platforms are Web Services according to the W3C specifications including a GML profile for the representation of geographic information or a CORBA-based infrastructure with a UML profile according to the OMG specifications.

Principal

A principal represents the identity of a subject in an ORCHESTRA Service Network. A subject may have several identities, and thus several principals. The association between a principal and a subject is established in an authentication process.

Reference Model [ISO Archiving Standards; http://ssdoo.gsfc.nasa.gov/nost/isoas/us04/defn.html]

A reference model is a framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education and explaining standards to a non-specialist.

Resource

Functions (possibly provided through services) or data objects.

Semantic Interoperability (Pollock, Hodgson 2004)

Semantic interoperability emphasizes the importance of information inside enterprise networks and focuses on enabling content, data, and information to interoperate with software systems outside of their origin. Information's meaning is the crucial enabler that allows software to interpret the appropriate context, structure, and format in which the information should reside at any given moment and inside any given system.

Semantic Web [W3C; http://www.w3.org/2001/sw/Overview.html]

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

Note: In ORCHESTRA, such an entity is called ORCHESTRA Service Component when referring to the software component and ORCHESTRA Service Instance when referring to the running instance in an ORCHESTRA Service Network.

Service Mapping

Process of mapping a description of an ORCHESTRA Service Type and the specification of its interfaces on platform-neutral level to an ORCHESTRA Implementation Specification for a given platform.

Service Profile Specification

ORCHESTRA Implementation Specification defining a functional subset of an interface of an ORCHESTRA Service Type as a result of a service mapping. The functional subset is defined in the sense that those operations and parameters that are marked on the abstract level as “mapping not required” may be omitted for the platform-specific specification.

Service Viewpoint

Viewpoint of the ORCHESTRA Reference Model that specifies the ORCHESTRA services supporting the syntactic and semantic interoperability between source systems and the development of ORCHESTRA Applications.

Session

Temporary association between a subject and a principal as a result of an authentication process initiated by the subject. Information about a session is stored in authentication session information.

Software Component [derived from component definition of http://www.opengeospatial.org/resources/?page=glossary]

Software program unit that performs one or more functions and that communicates and interoperates with other components through common interfaces.

Source System

Container of unstructured, semi-structured or structured data and/or a provider of functions in terms of services. The source systems are of very heterogeneous nature and contain information in a variety of types and formats.

Spatial Data Infrastructure [http://www.gsdi.org/pubs/cookbook/chapter01.html#spatial]

Relevant base collection of technologies, policies and institutional arrangements that facilitate the availability of and access to spatial data. The Spatial Data Infrastructure provides a basis for spatial data discovery, evaluation, and application for users and providers within all levels of government, the commercial sector, the non-profit sector, academia and by citizens in general.

Subject

Abstract representation of a user or a software component in an ORCHESTRA Application.

System [ISO/IEC 10746-2:1996]

Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a subsystem.

Note: For modelling purposes, the concept of system is understood in its general, system-theoretic sense. The term "system" can refer to an information processing system but can also be applied more generally.

System User

Provider of services that are used for an application domain as well as IT architects, system developers, integrators and administrators that conceive, develop, deploy and run applications for an application domain.

TBox

Describes relations between concepts (so-called "intensional" knowledge) without regarding concrete individuals.

Note: An example is: Every TropicalCyclone has-exactly 1 hurricaneCategory.

Technology viewpoint

Viewpoint of the ORCHESTRA Reference Model that specifies the technological choices of the service platform and its operational issues.

Thesaurus (Pollock, Hodgson 2004).

Synonym and antonym repository for data vocabulary terminology.

Transaction [W3C, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#transaction]

Transaction is a feature of the architecture that supports the coordination of results or operations on state in a multi-step interaction. The fundamental characteristic of a transaction is the ability to join multiple actions into the same unit of work, such that the actions either succeed or fail as a unit.

User

Human acting in the role of a system user or end user of the ORCHESTRA Architecture.

Viewpoint [RM-ODP]

Subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the design of the system.

Universe of discourse [ISO 19101]

View of the real or hypothetical world that includes everything of interest.

Web Service

Self-contained, self-describing, modular service that can be published, located, and invoked across the Web. A Web service performs functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.

W3C Web Service [W3C, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice]

Software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

4.3 General Remark

This document follows the ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards w.r.t. the usage of the word “shall”**.** The word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

5 Process of the ORCHESTRA Architectural Design

5.1 Overview

The ORCHESTRA Architecture is being designed in an iterative way recognising the fact that the requirements of the system and of the end users as well as the technological progress in the IT market and in IT standardisation have a dynamic nature and cannot be completely caught in a one-shot design. Thus, a global iteration cycle between the analysis, the design, the implementation and the deployment phase of the architecture is foreseen.

Figure 1 illustrates the iteration cycle between the analysis and the design phase which is explained further in the following paragraphs.

A **consolidation process** in-between ensures that, at a defined point in time, there is a common understanding of the system requirements, the user requirements and an assessment of the current technology as a foundation to design the ORCHESTRA Architecture.

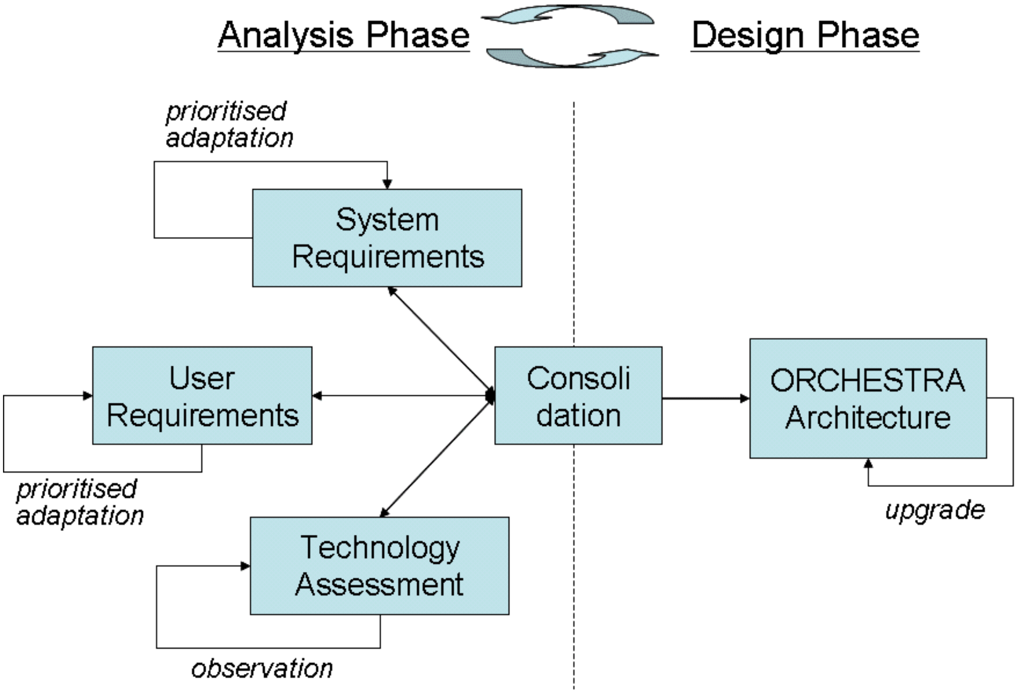


Figure 1: Dynamic ORCHESTRA Analysis and Design Process

System requirements for the ORCHESTRA Architecture encompass all functional and non-functional aspects that need to be considered in order to enable interoperability between systems. Interoperability is understood here according to ISO 19119:2005 as the capability to communicate, execute programs, or transfer data among various functional units in a manner that require the user to have little or no knowledge of the unique characteristics of those units.

Thus, system requirements for the ORCHESTRA Architecture are requirements for the infrastructure. Within the RM-OA, they originate from the combined expertise of the consortium in the area of interoperability as well as from (ORCH-DoW 2006).

Starting from a view oriented at system user roles, the system requirements for the ORCHESTRA Architecture are finally expressed in terms of architectural principals (see section A2.1.4 in the RM-OA Annex A2) that a system should follow. These architectural principals aim at improving the exchange, sharing and using of information and services among various functional units cross system boundaries, i.e. boundaries of existing systems which for some purpose need to collaborate with each other.

System requirements are expressed in generic technical terms, i.e. independent of application domains.

User requirements for the ORCHESTRA Architecture encompass all aspects that users or end-users of the ORCHESTRA Architecture expect to be reflected by a service infrastructure. User requirements are usually expressed in terms that are tailored to the needs of a specific application domain, for ORCHESTRA being the domain of risk management. As such, user requirements for the ORCHESTRA Architecture have to be aligned with and mapped to generic system requirements.

Technology assessment is a continuous process, too. ORCHESTRA aims at building the architecture on top of and abstracting from technologies, tools and products that are either standard approaches or have proven to be successful in solving interoperability problems in deployed use-cases.

The dynamic nature of these three input factors of the ORCHESTRA Architecture naturally leads to an iterative architectural design process. Various but controlled upgrades of the ORCHESTRA Architecture are required to adapt the architecture to the changing needs. Both the system and the user requirements are dynamic in the sense that they will be prioritised and adapted in local iteration cycles. A **consolidation** process is required in order to assess them in the light of time, budget and technological constraints. The consolidation process is determined by the answers to the following questions:

- How can the user requirements be realised by generic concepts such that a re-use for other application domains will be possible ?
- Which user requirements are of utmost importance with respect to the pilot scenarios in which the ORCHESTRA results are to be validated in a first place?
- What is the status of the existing technology in order to realise a given user requirement ?
- What is the effort to realise a user requirement in a given environment ?

As constant factors across the ORCHESTRA architectural design process, ORCHESTRA follows in each iteration step the principles of the Reference Model for Open Distributed Processing (RM-ODP) and the taxonomy of the ORCHESTRA services as described in subsections 5.2 and 5.4.

5.2 Application of the Reference Model of Open Distributed Processing (RM-ODP)

5.2.1 RM-ODP Overview

The Reference Model of Open Distributed Processing (ISO/IEC 10746-1:1998) is an international standard for architecting open, distributed processing systems. It provides an overall conceptual framework for building distributed systems in an incremental manner. The RM-ODP standards have been widely adopted: they constitute the conceptual basis for the ISO 19100 series of geomatics standards (normative references in ISO 19119:2005), and they also have been employed in the OMG object management architecture.

The RM-ODP approach has been used in the design of the OpenGIS Reference Model (OGC 2003) with respect to the following two aspects:

- It constitutes a way of thinking about architectural issues in terms of fundamental patterns or organizing principles, and
- It provides a set of guiding concepts and terminology.

Systems resulting from the RM-ODP approach (called ODP systems) are composed of interacting objects (see section 7.1.1 of ISO/IEC 10746-1:1998) whereby in RM-ODP an object is a representation of an entity in the real world. It contains information and offers services.

Based on this understanding of a system, ISO/IEC 10746 specifies an architectural framework for structuring the specification of ODP systems in terms of the concepts of viewpoints and viewpoint specifications, and distribution transparencies.

The viewpoints identify the top priorities for architectural specifications and provide a minimal set of requirements—plus an object model—to ensure system integrity. They address different aspects of the system and enable the 'separation of concerns'.

Five standard viewpoints are defined:

- The enterprise viewpoint: A viewpoint on the system and its environment that focuses on the purpose, scope and policies for the system.
- The information viewpoint: A viewpoint on the system and its environment that focuses on the semantics of the information and information processing performed.
- The computational viewpoint: A viewpoint on the system and its environment that enables distribution through functional decomposition of the system into objects which interact at interfaces.
- The engineering viewpoint: A viewpoint on the system and its environment that focuses on the mechanisms and functions required to support distributed interaction between objects in the system.
- The technology viewpoint: A viewpoint on the system and its environment that focuses on the choice of technology in that system.

The aspect of a distributed ODP system is handled by the concept of distribution transparency. Distribution transparency relates to the masking from applications the details and the differences in mechanisms used to overcome problems caused by distribution. According to the RM-ODP, application designers simply select which distribution transparencies they wish to assume and where in the design they are to apply. The RM-ODP distinguishes between eight distribution transparency types. These distribution transparencies consider aspects of object access, failure of objects, location of objects, as well as replication, migration, relocation, persistence and transactional behaviour of objects.

5.2.2 Mapping of RM-ODP to the ORCHESTRA Architectural Design Process

An RM-ODP-based approach has been selected for the design of the ORCHESTRA Architecture as the primary objectives of RM-ODP, such as

- support for aspects of distributed processing,
- provision of interoperability across heterogeneous systems, and
- hiding consequences of distribution to systems developers,

are largely coherent with the ORCHESTRA objectives. However, as an ORCHESTRA system will have the characteristic of a loosely-coupled network of systems and services instead of a "distributed processing system based on interacting objects", the RM-ODP concepts are not followed literally. For instance, the ORCHESTRA concepts are not specified in terms of the RM-ODP distribution transparencies as these are specified in terms of interacting objects.

The usage of RM-ODP for the ORCHESTRA Architectural design process focuses on the structuring of ideas and the documentation of the ORCHESTRA Architecture. Thus, a mapping of the RM-ODP viewpoints to the ORCHESTRA needs has been applied and summarised in Table 2:

- The second column of Table 2 provides the original definitions of the viewpoints as given in the OpenGIS Reference Model using the terms of the OpenGIS glossary.
- The third column of Table 2 indicates the mapping of the viewpoints to the ORCHESTRA needs using the terms as defined in the ORCHESTRA glossary (see section 4).

  Note: In order to highlight the fact, that an ORCHESTRA deployment will have the nature of a loosely-coupled distributed system based on networked services rather than a distributed application based on computational objects, the "computational viewpoint" is referred to as "service viewpoint" in ORCHESTRA.
- The fourth column of Table 2 provides examples of what will be defined in the respective viewpoint.

iew oint lam	Definition according to ISO/IEC 10746	Definition according to the OpenGIS Reference Model	Mapping to the ORCHESTRA architectural design process	Examples
Enterprise	Concerned with the purpose, scope and policies governing the activities of the specified system within the organization of which it is a part.	Focuses on the purpose, scope and policies for that system.	Reflects the analysis phase in terms of the system and the user requirements as well as the technology assessment. Includes rules that govern actors and groups of actors, and their roles.	Use case definition for a statistical processing service. Rules for the maintenance and evolution of the architecture.
Information	Concerned with the kinds of information handled by the system and constraints on the use and interpretation of that information.	Focuses on the semantics of information and information processing.	Specifies the modelling approach of all categories of information the ORCHESTRA Architecture deals with including their thematic, spatial, temporal characteristics as well as their meta-data.	Information objects specified in UML class diagrams and referred to by the specification of the processing service (e.g. as parameter types).
Computational	Concerned with the functional decomposition of the system into a set of objects that interact at interfaces – enabling system distribution.	Captures component and interface details without regard to distribution.	Referred to as “Service Viewpoint” Specifies the ORCHESTRA Interface and Service Types that aim at improving the syntactic and semantic interoperability between services, source systems and ORCHESTRA Applications.	Specification of the externally visible behaviour of a service type, e.g. UML specification of the interface types of the processing service including the possibility to perform statistics Service support for service orchestration and choreography.
Technology	Concerned with the choice of technology to support system distribution.	Focuses on the choice of technology.	Specifies the technological choices of the platform, its characteristics and its operational issues.	Specification of the platform “ORCHESTRA Web Services” consisting of W3C Web Services according to (W3C 2004) and a GML profile.

Engineering	Concerned with the infrastructure required to support system distribution.	Focuses on the mechanisms and functions required to support distributed interaction between objects in the system.	Specifies the mapping of the ORCHESTRA service specifications and information models to the chosen platform. Considers the characteristics and principles for service networks.	Provision of the service implementation specification, incl. mapping of the UML specification to WSDL and functional service properties (e.g. persistency). Decision on access control policies.

Table 2: Mapping of the RM-ODP Viewpoints to ORCHESTRA

5.3 The ORCHESTRA Reference Model

A graphical depiction of the relationships between the viewpoints and their mapping to the ORCHESTRA architectural design process, the implementation and deployment phase is provided in Figure 2. The result is called the ORCHESTRA Reference Model that covers the following phases:

- Analysis phase that leads to the specification of the Enterprise Viewpoint (see section 6)
- Design phase that leads to the specification of the ORCHESTRA Architecture (see section 5.3.1)
- Implementation phase that leads to ORCHESTRA Implementation Specifications (see section 5.3.2) implemented as ORCHESTRA Service Components
- Deployment phase that leads to ORCHESTRA Service Networks (see section 5.3.3).

The iteration cycles that permit to adapt the architecture to changing or refined needs as specified in the enterprise viewpoint are not shown in Figure 2.

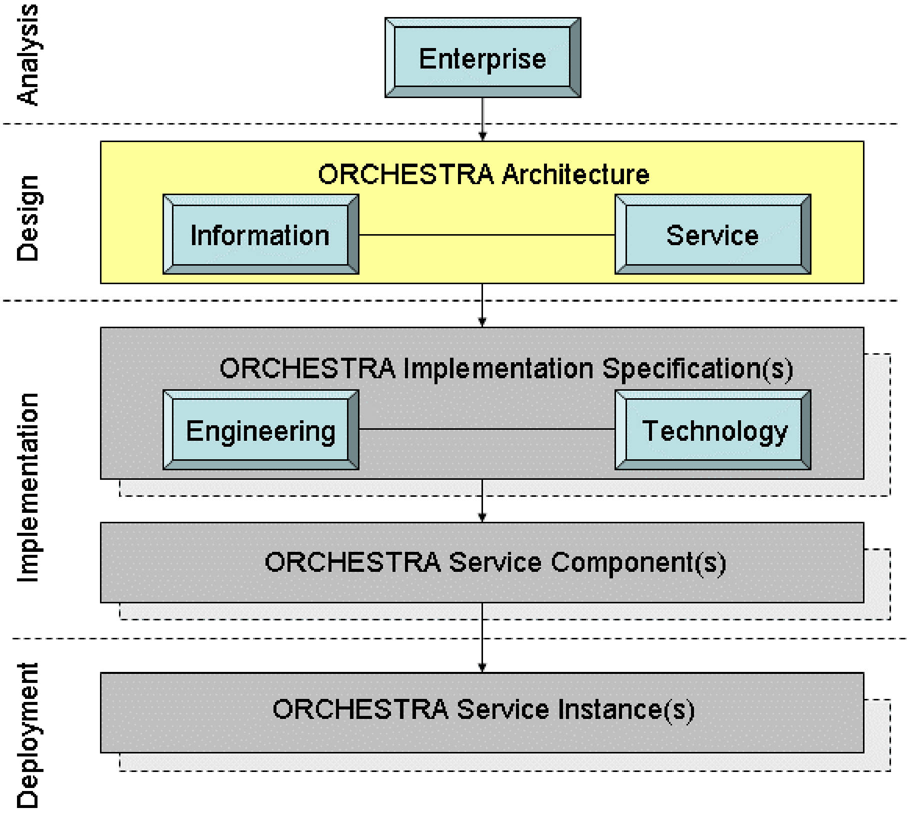


Figure 2: The ORCHESTRA Reference Model

5.3.1 The ORCHESTRA Architecture

The **ORCHESTRA Architecture (OA)** is, by definition, a platform-neutral specification according to the requirements of ISO 19119:2005 (i.e. specification in the conceptual schema language UML). The ORCHESTRA Architecture is specified as part of the design phase. It encompasses the harmonised specification of the Information and Service Viewpoint resulting from requirements of the Enterprise viewpoint. The fact that the specification is platform-neutral means that the ORCHESTRA Architecture does not contain any particular dependencies on the peculiarities of a given platform.

5.3.2 The ORCHESTRA Implementation Specification

The aspects of the Engineering and Technology viewpoints are outside the scope of the ORCHESTRA Architecture. Instead, they are combined in a dedicated specification step that may be carried out multiple times. Each step represents one mapping of the ORCHESTRA Architecture (i.e. the Information and Service Viewpoint specification) to a specific service platform and leads to a platform-specific **ORCHESTRA Implementation Specification (OIS)**.

A **service platform**, or **platform** for short, hereby is defined to be the set of infrastructural means and rules that describe how to specify service interfaces and related information and how to invoke services in a distributed system. Thus, a platform provides a service infrastructure and associated rules for the specification, discovery, composition and invocation of services in a distributed system. Examples of platforms are Web Services according to the W3C specifications or a CORBA-based infrastructure according to the OMG specifications.

An OIS contains platform-specific specifications of ORCHESTRA information models and ORCHESTRA services. This means in concrete terms that the information models expressed in UML have to be mapped to a schema language (e.g. XML/GML or EXPRESS) that fits to the selected platform. Likewise, the abstract specifications of the ORCHESTRA service interfaces expressed in UML have to be mapped to a service description language (e.g. WSDL) that fits to this platform, too. These mapping processes may be done manually or performed (semi-)automatically by a tool.

Note: The iterative design process of the ORCHESTRA Architecture allows designers to re-apply changes in the viewpoint specifications if problems during an OIS specification process occur.

Note that an OIS itself is not part of the RM-OA specification. The RM-OA just provides the architectural framework for an OIS. As a consequence,

- the RM-OA description of the Technology Viewpoint (see section 10) contains guidelines, requirements and rules what has to be considered when specifying a platform, and
- the RM-OA description of the Engineering Viewpoint (see section 11) contains guidelines, requirements and rules what has to be considered when mapping to a chosen platform and in the specification of an OIS. Furthermore, the Engineering Viewpoint also covers engineering principles and guidelines how to design ORCHESTRA Service Networks (see section 5.3.3) and discusses their characteristics.

The implementation phase encompasses the edition of the ORCHESTRA Implementation Specifications and their implementation in **ORCHESTRA Service Components (OSC)** and platform-specific encodings of the information models. An OSC is a component that provides an external interface of an ORCHESTRA Service according to an OIS. Note that the platform-specific encodings of the information models are accessed by means of ORCHESTRA Services, thus they are not explicitly illustrated in the ORCHESTRA Reference Model in Figure 2.

5.3.3 The ORCHESTRA Service Network and ORCHESTRA Applications

An executing manifestation of an OSC is an **ORCHESTRA Service Instance (OSI)**. The deployment phase encompasses the deployment of OSIs on hardware (see Figure 3).

The set of ORCHESTRA Service Instances connected through a communication network is called an **ORCHESTRA Service Network (OSN)**. An OSN thus comprises the set of networked hardware components and ORCHESTRA Service Instances that interact in order to serve the objectives of ORCHESTRA Applications.

Note that the grouping of OSIs into software components and their distribution and deployment on hardware components (e.g. server machines) is not relevant from when specifying the ORCHESTRA Information and Service Viewpoint. The basic unit of an OSN for the provision of functions are the OSIs. One of several OSIs may be deployed as part of one software component.

On a next higher level, software components distributed in a network are grouped together to form **ORCHESTRA Applications**. A software component as part of an ORCHESTRA Application may contain one or more OSIs but, in addition, also other functionality, e.g. functions to built service request messages or to consume response messages.

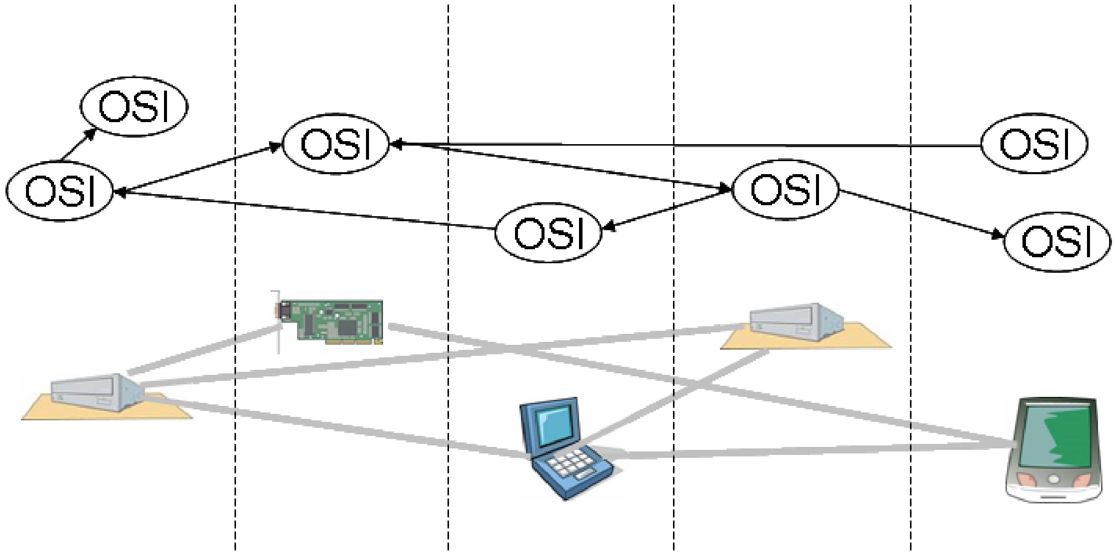


Figure 3: Deployment of ORCHESTRA Service Instance in an ORCHESTRA Service Network

Figure 4 shows the example of two ORCHESTRA Applications that are built out of several interacting software components, some of them containing an OSI and some not. Note that in this example these two ORCHESTRA Applications are sharing the usage of one OSI, i.e., client software components in the respective ORCHESTRA Applications may call operations of this OSI in parallel.

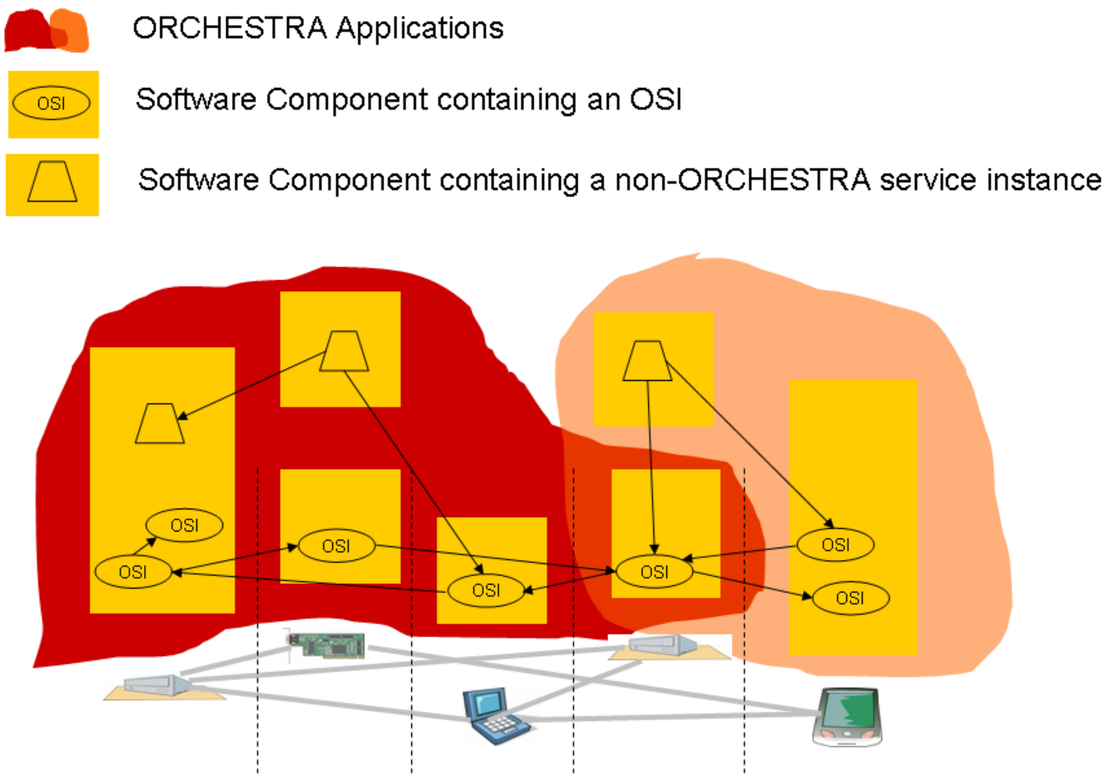

ORCHESTRA Applications
OSI
Software Component containing an OSI
Software Component containing a non-ORCHESTRA service instance
OSI
OSI
OSI
OSI
OSI
OSI
OSI

Figure 4: Example of two ORCHESTRA Applications using the same OSI

5.3.4 The ORCHESTRA Application Architecture

An **ORCHESTRA Application Architecture (OAA)** is an instantiation of the ORCHESTRA Architecture by inclusion of those thematic aspects that fulfil the purpose and objectives of a given application. The concepts for such an application stem from a particular application domain (e.g. a risk management application).

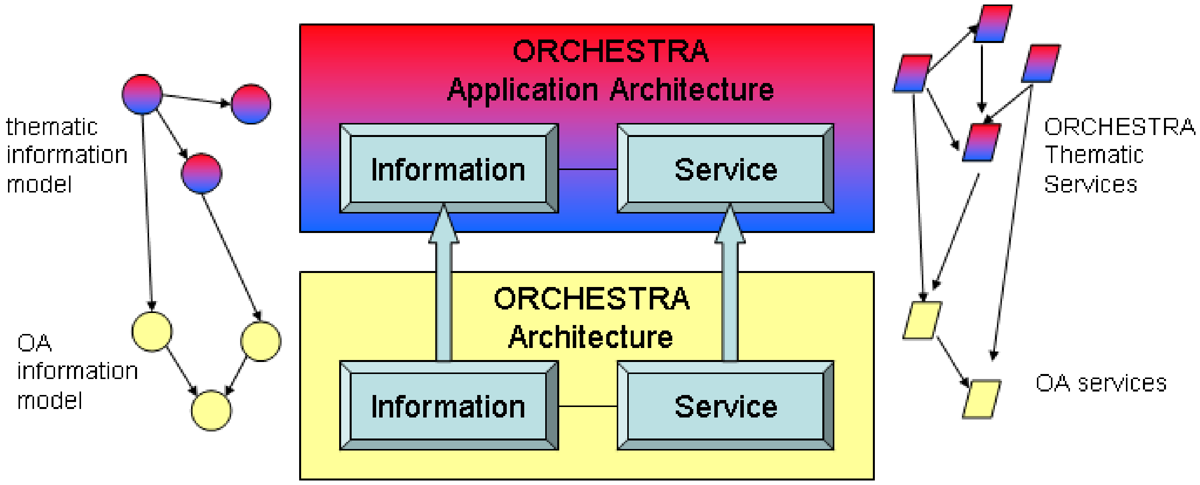


Figure 5: ORCHESTRA Application Architecture

By definition, an OAA is a platform-neutral specification. As such, an OAA covers both the platform-neutral specification of the thematic aspects of the information viewpoint (thematic information model, e.g. a domain-specific ontology) and the service viewpoint (addition of thematic services). It may encompass a specification extension but also a restriction, e.g. omission of optional services or information elements.

The relationship between an ORCHESTRA Application Architecture and the ORCHESTRA Architecture is shown in Figure 5.

Note: The process to identify on the conceptual level the pre-eminent information types and their relationships (leading to a conceptual thematic information model) and the functional requirements (leading to service descriptions on the conceptual level) is outside the scope of the RM-OA. The RM-OA just provides the framework to formally specify information models as well as services in order to integrate them into the OA.

5.3.5 The ORCHESTRA Application Implementation Specification

A platform-neutral specification of an OAA based on a conceptual schema language like UML might not be adequate in all development projects. Sometimes, the platform has been pre-selected and the delivery of a platform-neutral specification that abstracts from the platform specific characteristics is not necessary.

Nevertheless, in order to allow the exploitation and usage of the ORCHESTRA Architecture, the thematic information model and the respective OT Services may also be specified directly on the basis of a chosen ORCHESTRA Implementation Specification. In this case, the resulting platform-specific specification of the thematic extensions and restrictions is called an **ORCHESTRA Application Implementation Specification (OAIS)**.

5.4 The OpenGIS Service Architecture

Topic 12 of the OpenGIS Abstract Specification ("The OpenGIS Service Architecture" - ISO 19119:2005) provides a specification framework for developers to create software that enables users to access and process geographic data from a variety of sources across a generic computing interface within an open IT environment.

It extends the architectural reference model of ISO 19101:2001 defining an Extended Open Systems Environment (EOSE) model for geographic services.

The resulting ISO Architecture for Geospatial Services distinguishes between Information Technology Services (IT services) and Geospatial Information Services (GI services).

- IT Services are general services in a distributed computing environment, such as processing services that perform large-scale computation involving substantial amount of data, system management services for encoding and transfer of data across communication networks etc.
- GI Services are specialized IT services that define capabilities that are specific to the access to, analysis of, transformation of, manipulation of, storage of, or exchange of geographic information.

In the ISO Architecture for Geospatial Services, a GI service is only specified wherever existing IT services of the selected distributed computing platform do not exist or do not meet the specific GI requirement.

In the ORCHESTRA Reference Model the distributed computing platform is referred to as the service infrastructure. However, the distinction between IT and GI services is not applied for the ORCHESTRA service taxonomy because the ORCHESTRA Architecture (and thus the ORCHESTRA services) shall contain an integrated information model that covers thematic, temporal and spatial aspects.

The link of the RM-OA to the technical content of ISO 19119:2005 focuses on the two following aspects:

- the requirements for platform-neutrality (see section 5.4.1)
- the usage of the service taxonomy (see section 5.4.2), and
- the requirements for a simple service architecture (see section 5.4.3).

5.4.1 Platform-neutral and Platform-specific Service Specification

The ORCHESTRA service specifications as part of the ORCHESTRA Architecture shall comply with the requirements of ISO 19119:2005, section 10.3, for "platform-neutrality".

This means that the following points are considered:

- The ORCHESTRA architectural models shall be described in UML according to the rules and guidelines of ISO/PDTS 19103 (conceptual schema language), e.g. for the usage of basic UML data types.
- As part of the service viewpoint, ORCHESTRA services shall be defined as "platform-neutral service specifications". They both define static models (objects including the attributes and operations for each object) and dynamic models (capturing the interaction patterns between objects and state modelling).
- As part of the engineering viewpoint, the ORCHESTRA platform-neutral models are mapped to a specific service infrastructure context. The resulting platform-specific service models may be defined in UML or in terms of the platform-specific language (e.g. WSDL). However, it is required to maintain a description of their mapping from the corresponding platform-neutral models. This mapping shall show how the intentions of the platform-neutral specifications are met in the context of the service platform. In order to support interoperability, the reverse mapping back to the concepts in the platform-neutral model must be defined.

5.4.2 Service Taxonomy

The ORCHESTRA Architecture informally classifies the ORCHESTRA services according to the service taxonomy of ISO 19101 (also referred to in ISO 19119:2005, section 8.3). The service categories are:

- **Human interaction services** are services for management of user interfaces, graphics, multimedia, and for presentation of compound documents.
- **Model/Information management services** are services for management of the development, manipulation, and storage of meta-information (including ontology specifications), conceptual schemas, and datasets.
- **Workflow/Task management services** are services for support of specific tasks or work-related activities conducted by humans or software components with a high degree of autonomy (agents). These services support use of resources and development of products involving a sequence of activities or steps that may be conducted by different persons.
- **Processing services** are services that perform computations. These computations might range from the performance of mathematical equations up to large-scale computations involving substantial amounts of data.
- **Communication services** are services for encoding and transfer of data across communications networks.
- **System management services** are services for the management of system components, applications, and networks. These services also include management of user accounts and user access privileges.

Note: The classification of a particular service in a taxonomy is considered as meta-information for the service. According to the ORCHESTRA handling of meta-information (see section 8.4.1), the adequacy of this service taxonomy is therefore to be considered when defining purpose-oriented meta-information for services (see section 8.4.2).

5.4.3 ORCHESTRA as Simple Service Architecture according to ISO 19119:2005

The ORCHESTRA Architecture is a service-oriented architecture. Furthermore, looking at ISO 19119:2005, section 7.6, the ORCHESTRA Architecture aims at observing the characteristics of a “simple service architecture” in all cases where it is applicable and useful. Exceptions shall be identified in an explicit fashion.

A simple service architecture according to ISO 19119 and interpreted in the context of the ORCHESTRA Architecture is a message-based architecture that supports service chaining and considers the following simplifying assumptions:

- Message-operations

  ORCHESTRA operations shall be modelled as messages. A message operation shall consist of a request and response. Requests and responses contain parameters as the payload, which is transferred in uniform manner independent of content. Simple applications are characterized by message exchange patterns such as one-way (or event), and two-way (or synchronous) request response interactions. A service specification should make such simple exchange applications as easy as possible to create and to use.

- Separation of control and data

  A client controlling an ORCHESTRA service may not want the full results of the service. For example, the user may have no need for the potentially voluminous intermediate products in a service chain. Only the final result of a service chain may be needed by the client. Therefore, an interface should separate the control of a service from the access to the data resulting from the service. A client should have the option of receiving just the status of an operation and the data should be accessible separately through a separate operation.

- Stateful vs. stateless service

  For simplicity it is desired that an ORCHESTRA service be stateless, i.e., that a service invocation be composed of a single request-response pair with no dependence on past or future interactions. This will not always be possible. For some ORCHESTRA services, preconditions must be set and iteration may be required. Then it will be necessary to model the service with a state diagram having multiple states. Transitions between the states are triggered by operations. The state diagram and associated descriptions will be part of the abstract and of the implementation specification of the interfaces of an ORCHESTRA service (see section 9.2.6).

- Known service type

  All ORCHESTRA service instances are of specific service types and the client may access the service type description prior to calling the service. In the ORCHESTRA Reference Model, a “known service type” is a service type with an externally available description.

  Note: The ORCHESTRA Reference Model does not enforce that the “clients shall contain software for accessing the service type prior to encountering service instances of the type in an implemented architecture” as requested by ISO 19119:2005. Although this is useful and a good start in many applications in order to reduce complexity, the ORCHESTRA Architecture aims at providing services that enables the design of generic application code that is controlled by the availability of service meta-information. In a first step, this meta-information will stick to providing syntactic information like the operation signatures, the provider name and a textual service description. However, in RM-OA Version 4 (see section 6.2.3) meta-information will be considered that includes semantic concepts for services.

- Adequate hardware

  ORCHESTRA Services are implemented as software components (OSCs) and deployed and executed on hardware hosts. The ORCHESTRA Reference Model assumes that the issues of hardware hosting of the software are transparent to the user. It is assumed that the service has adequate hardware, i.e. hardware assignment is transparent to user.

6 Enterprise Viewpoint

6.1 Overview

The enterprise viewpoint of the ORCHESTRA Architecture briefly describes its

- business perspective,
- purpose (the core mission of the ORCHESTRA Architecture),
- scope (e.g. intended users),
- policies (e.g. standardisation approach, openness)

In terms of the architectural process described in section 5, it reflects the analysis phase in terms of the high-level and the user requirements as well as the technology assessment.

6.2 Business Perspective

6.2.1 Contribution to the ORCHESTRA Goals

The design of the ORCHESTRA Architecture (OA) is triggered by the main goals of the ORCHESTRA project which have been described as:

- To design an open service-oriented architecture for risk management where special attention will be paid to providing a solution for the combination of spatial and non-spatial data and services. The ORCHESTRA Architecture will contribute to the INSPIRE (COM 2004) (Dufourmont, Annoni, De Groof 2004) and GMES (GMES 2004) infrastructure and thus will assist and support the needed development of INSPIRE technical specifications and guidelines in the INSPIRE preparatory phase.
- To develop a software infrastructure for enabling risk management services.
- To develop services for various multi-risk management applications based on the architecture.
- To validate the ORCHESTRA Architecture and thematic services in a multi-risk scenario.
- To provide software standards for risk management applications, and to provide additional information about these standards. In particular, the de facto standard of OGC and the de-jure standards of ISO and CEN are expected to be influenced.

6.2.2 Collaboration with European Initiatives and Projects

Furthermore, the ORCHESTRA Architecture is meant to provide substantial input to an information infrastructure (info-structure) in the context of the European INSPIRE (Infrastructure for Spatial Information in Europe) and GMES (Global Monitoring for Environment and Security) initiatives, especially but not exclusively for environmental risk management applications. For this task, ORCHESTRA will co-operate with two other European integrated projects:

- OASIS: Open Advanced System for crisIS management (IST IP 4677 http://www.oasis-fp6.org/)
- WIN: Wide Information Network for Risk Management (IST IP 511481 http://www.win-eu.org)

These projects face in common the task of organising risk management systems that are networked across and between organisations with interoperable capabilities.

6.2.2.1 Common Architectural Principles of ORCHESTRA, OASIS and WIN

In June 2004, the European Commission (DG INFSO) has initiated a series of meetings between major stakeholders of the strategic objective “Improving Risk Management”, (i.e. ORCHESTRA, OASIS and WIN), stakeholders of GMES (in particular ESA) and stakeholders of INSPIRE (in particular JRC). These meetings aim at discussing how all on-going initiatives may collaborate in the future.

With respect to the relationship between ORCHESTRA, OASIS and WIN common architectural principles of an open info-structure have been discussed and were finalised in a white paper (see also section 6.2.2.4).

OASIS and ORCHESTRA have agreed to work on a common scenario that will combine the needs across different phases of the risk management cycles, including the response phase. The scenario will be developed as a paper study which aims at evaluating the OA in a disaster management context.

6.2.2.2 Requirements of the INSPIRE Relationship

The acronym INSPIRE stands for “Infrastructure for Spatial Information in Europe”. INSPIRE is a draft European directive establishing the legal framework for setting up and operating an Infrastructure for Spatial Information in Europe. The Directive focuses on spatial data that are held by or on behalf of public authorities. INSPIRE is targeting environmental policies, however, other sectors such as agriculture, transport and energy may benefit, too, once this infrastructure is in place.

The proposal of the INSPIRE directive lays down general rules for the various components of a framework for a European Spatial Data Infrastructure (SDI). Thus it considers rules for metadata to support the discovery and evaluation of spatial data and services; rules to achieve interoperability that allows integration of spatial data of the various themes addressed by INSPIRE; rules for interoperable network services for discovery, viewing, accessing and downloading spatial information; rules for data sharing; necessary coordinating structures; and the development of a European geo-portal to provide a common entry to access all INSPIRE network services. The proposal has been adopted by the European Commission in July 2004 and since then has entered into the co-decision procedure that is a legislative procedure central to the Community's decision-making system.

Whilst this political process continues, the INSPIRE Work Programme published in April 2005 identified a step-wise approach for the definition and preparation of detailed Implementing Rules (Dufourmont, Annoni, De Groof 2004). Clearly, such Implementing Rules cannot be developed in isolation but need to take into account what already exists in the Member States as well as the broader international developments in the field of SDI and e-government services. In addition operational experiences, international agreements and protocols that are already in place across various thematic communities need to be considered.

With these considerations in mind, an open call was launched in spring 2005 for the registration of interest by Spatial Data Interest Communities (SDIC) and Legally Mandated Organisations (LMO). LMO represent those organisations at local, regional, and national levels that have a formal legal mandate giving them the responsibility for specific thematic data resources. As part of the open call it was asked to put forward experts and reference material to support the preparation of the Implementing Rules. More than 180 experts have been proposed, including experts supported by the ORCHESTRA project. The INSPIRE Drafting Teams were then established and started operations in October 2005.

The current time-line for the full implementation of INSPIRE envisages that the Directive will be approved by the end of 2006 or the beginning of 2007, that its provisions will be transposed in national legislation by the Member States in 2008-9, and that implementation will take place in the following years. During the current INSPIRE preparatory phase (Dufourmont, Annoni, De Groof 2004) the ORCHESTRA project will provide input towards the drafting as well as the piloting of the technical INSPIRE Implementing Rules in the risk management domain.

The technical INSPIRE Implementing Rules shall be based on existing standards and specifications if possible. Thus the existing geographic information standards and specifications from ISO, CEN and OGC serve as input into the drafting of the INSPIRE Implementing Rules. If it turns out that these standards do not cover or cannot fully fulfil requirements formulated in the INSPIRE directive adequate extensions and modifications are proposed and respective feedback into the standardisation bodies will be ensured. The current status of the drafting of the INSPIRE Implementing Rules has been reported on the recent 12th EC GI conference in June 2006[1].

The first input of ORCHESTRA into INSPIRE could be expected on the drafting of Implementing Rules for Network Services by providing the ORCHESTRA RM-OA and the developed ORCHESTRA services

[1] See http://www.ec-gis.org/Workshops/12ec-gis/presentations

specifications as reference materials. The requirements on INSPIRE Network Services will therefore be detailed in the following sub-section.

Moreover ORCHESTRA could support the drafting of Implementing Rules for INSPIRE Data Specifications[2] by providing the ORCHESTRA RM-OA and the derived application schemata as reference material.

6.2.2.3 Detailed definitions and requirements of INSPIRE Network Services

In the context of INSPIRE Network Services the current INSPIRE proposal distinguishes the following service types:

- discovery services
- upload services
- view services
- download services
- transformation services
- “invoke spatial data services” services

Following the INSPIRE proposal, the Network Services will be available from each Member State leading to a distributed architecture at the European level. They will be accessible via the European Geo-Portal and potentially via the member states’ own portals. The definition of appropriate technical specifications requires that considered interface specifications are mature and proved by implementation and operational usage, including performance consideration.

As a first task a more detailed description of these network services is developed. The document on *Detailed definitions on the INSPIRE Network Services* [3] proposes a (technical) understanding of the INSPIRE Network Services and tries to identify related issues. This understanding served as a starting point for the work in INSPIRE Drafting Team on Network Services. The Drafting Team is currently updating the document and adding a description of an INSPIRE (service) reference model that includes the concept of horizontal services for DRM, UAA, and e-commerce aspects[4]. The understanding and detailed description of the INSPIRE network services developed so far is summarised in the following paragraphs.

Discovery Services

Discovery services are to search for *spatial data sets* and *spatial data services* on the basis of the content of the corresponding metadata and to display the content of the metadata. As a minimum the following combination of search criteria shall be implemented:

- keywords,
- classification of spatial data and services,
- spatial data quality and accuracy,
- degree of conformity with the harmonised specifications,
- geographical location,
- conditions applying to the access to and use of spatial data sets and services,
- the public authorities responsible for spatial data sets and services.

The related search and response metadata are defined by the INSPIRE Metadata Drafting Team.

The OpenGIS Specification Catalogue Service Web with the ISO application profile (CS-W 2.0 ISO AP

[2] See http://www.ec-gis.org/Workshops/12ec-gis/presentations/Plenary%20room/INSPIRE%20I/portele.pdf

[3] http://inspire.jrc.it/reports/dt/ir_dev_process_network_services.pdf

[4] See http://www.ec-gis.org/Workshops/12ec-gis/presentations/Plenary%20room/INSPIRE%20II/serrano.pdf

19115/19119) has been identified by the Network Services Drafting Team as the most relevant specification for INSPIRE discovery services. This specification would make the definition of a related set of query and response properties, query language, and the desired level of discovery (dataset only, or also feature level) necessary. As a candidate standard for service metadata ISO19119 has been identified but is not considered to be as well developed as the ISO19115 standard is for metadata.

Another open issue on discovery services is whether and how to deal with multiple application profiles for discovery services (e.g. the ebRIM Profile for the CS-W) and whether and how to link to service registries as UDDI.

Upload Services

Currently the upload services are considered to be functionality closely linked to discovery services allowing for the publishing and updating of metadata sets.

View services

The following specifications have been identified by the Network Services Drafting Team as the most relevant specification for INSPIRE view services:

- ISO 19128 Web Map Service
- Draft CEN TC287 profile of ISO 19128 / WMS 1.3

Download services

For INSPIRE Download services it is proposed to distinguish downloading predefined datasets (for instance FTP for downloading files) and downloading features allowing for an appropriate selection of these features (for instance an OpenGIS Web Feature Services). It is envisioned that INSPIRE download services require close links to e-commerce services and the work on INSPIRE metadata and data specification implementing rules.

Transformation services

Services to support coordinate transformation have been identified as an important and thus prioritised instance of INSPIRE transformation services. Within this context the draft OpenGIS Specification for Web Coordinate Transformation Service (OGC WCTS) has been identified as highly relevant. As for the view services questions were raised about the need for and requirements on a (European) CRS Registry.

As further candidates for INSPIRE transformation services, services for schema transformation and services for generalisation have been discussed. Whether these services are required is still under consideration.

“Invoke spatial data services” services

The INSPIRE drafting team proposed invocation services to be understood as the possibility to orchestrate (“chain”) INSPIRE spatial data services in the sense of distributed geo-processing.

The draft INSPIRE Directive requires "invoke spatial data services" to ensure that spatial data services can be invoked in an INSPIRE way fostering harmonisation and interoperability, be it by a user or by other services or applications. If an INSPIRE service reference model includes constraints and characteristics a spatial data service has to fulfil to be effectively invoked inside a framework and the invocation mechanism is unambiguously defined and detailed in an INSPIRE reference model then it could be envisioned that “invoke spatial data services” service implementing rules concentrate on this reference model and the detailed invocation/activation framework.

For defining INSPIRE invocation services or mechanisms it has been realised that orchestration/chaining of geospatial services is in a very early stage. Here, SOAP, WSDL, UDDI, and BPEL are currently considered as relevant technologies and specifications.

6.2.2.4 Requirements of the GMES Relationship

The overall aim of the Global Monitoring for Environment and Security (GMES) initiative is to support Europe’s goals regarding sustainable development and global governance by providing timely and high quality data, information, and knowledge. Access to information has strategic value in the development of nations and regions. GMES will contribute to Europe's ability to fulfil its role as a world player. This entails the capacity to have independent access to reliable and timely information on the status and evolution of the Earth's environment at all scales, from global to regional to local. GMES must also ensure long-term, continuous monitoring on a time-scale of at least decades.

A final report for the GMES initial period (2001-2003) is available (GMES 2004). It proposes a way forward for the GMES period 2004-2008. As part of the strategic requirements specifying how to realise the GMES action plan, this report contains assessments and objectives for Data Integration and Information Management in the GMES service context which could be relevant for ORCHESTRA.

To date, the relationship between ORCHESTRA and GMES is not formally defined. Potential contributions to GMES were discussed in the meetings mentioned in section 6.2.2.1, but no conclusions have been reached so far. Commitments have not been made and can only be made if they are compatible with the work plan and budget of the ORCHESTRA project. This means that at this time ORCHESTRA does not need to take into account specific GMES business requirements which do not overlap with ORCHESTRA requirements in the first place.

6.2.3 Evolution of the ORCHESTRA Architecture

In order to fulfil the business objectives, especially with respect to the GMES and INSPIRE initiative, the ORCHESTRA Architecture considers from the beginning a multi-step approach:

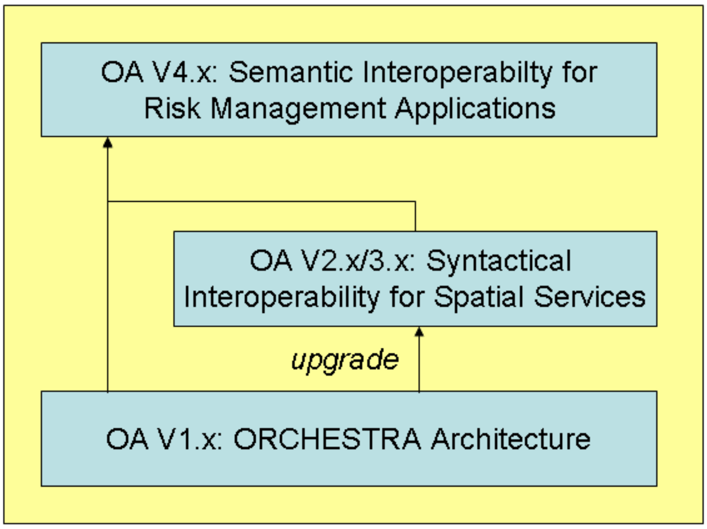

OA V4.x: Semantic Interoperabilty for Risk Management Applications
OA V2.x/3.x: Syntactical Interoperability for Spatial Services
upgrade
OA V1.x: ORCHESTRA Architecture

Figure 6: The Evolution of the ORCHESTRA Architecture

- In OA Version 1.x (RM-OA 2005), the ORCHESTRA Architecture has been conceived. The ORCHESTRA Architecture provides a common view of how to harmonise the requirements for syntactic and semantic service and data interoperability including their thematic, temporal and spatial characteristics.
- In OA Versions 2.x (the present RM-OA version) and V3.x, the focus is on refining the OA V1 in terms of service specifications for syntactic interoperability in the spatial domain. These versions link to the INSPIRE requirements for network services as outlined in section 9.4.
- In OA Version 4.x, the focus is on extending and refining former OA Versions in terms of service specifications for semantic interoperability in the risk management domain.

Note: None of these OA versions includes ORCHESTRA Implementation Specifications (OIS); they all stay on the platform-neutral level. It has not yet been decided for which OA versions a platform mapping will be provided in the form of corresponding OISs.

6.3 Architectural Requirements for the OSN Design

In the following sections, architectural requirements for the ORCHESTRA Architecture and an OSN are specified. They have been derived through a line of argument starting from

1. the different types of *users* of an OSN and their *roles,*
2. connecting these *user roles* with *fundamental challenges* for the ORCHESTRA Architecture,
3. deriving from that the *key system requirements*, and
4. finally developing *architectural principles*.

Here, only the architectural principles are briefly explained in terms of architectural requirements.

6.3.1 Rigorous Definition and Use of Concepts and Standards

The ORCHESTRA Architecture shall make rigorous use of proven concepts and standards in order to decrease dependence on vendor-specific solutions, to help ensure the openness of the OSN and to support the evolutionary development process of the ORCHESTRA Architecture.

6.3.2 Loosely Coupled Components

The components involved in OSN shall be loosely coupled, where loose coupling implies the use of mediation to permit existing components to be interconnected without changes.

Note: An example of an ORCHESTRA Service Type that supports the concept of mediation is the Catalogue Service (see section 9.6.6) that decouples the resources (data and services) from their clients.

6.3.3 Technology Independence

The ORCHESTRA Architecture shall be independent of technologies, their cycles and their changes. It must be possible to accommodate changes in technology (e.g. the lifecycle of middleware technology) without changing the ORCHESTRA Architecture itself. The ORCHESTRA Architecture shall be independent of specific implementation technologies (e.g. middleware, programming language, operating system) and shall not be influenced by or deal with technical limitations of specific implementation technologies.

Note: The ORCHESTRA Architecture follows this architectural requirement by specifying it in a platform-neutral manner in the first place before mapping it to one or more ORCHESTRA Implementation Specifications (see section 5.3.1).

6.3.4 Evolutionary Development - Design for Change

The ORCHESTRA Architecture and an OSN shall be designed to evolve, i.e. it shall be possible to develop and deploy the system in an evolutionary way. The ORCHESTRA Architecture and an OSN shall be able to cope with changes of user requirements, system requirements, organisational structures, information flows and information types in the source systems.

Note: The iterative design approach in ORCHESTRA resulting in the planned evolution of the RM-OA in several versions (see section 6.2.3) is an example of how this architectural requirement is supported.

6.3.5 Component Architecture Independence

The ORCHESTRA Architecture shall be designed such that an OSN and source systems (i.e. existing information systems and information networks) are architecturally decoupled. This means that the ORCHESTRA Architecture shall not impose any architectural patterns on source systems for the purpose of allowing them to collaborate in an OSN, and no source system shall impose architectural patterns (i.e. service interaction patterns as for instance described in section 9.9) on an OSN .

6.3.6 Generic Infrastructure

The OA Services shall be independent of the application domain. This means that the OA Services should be designed in such a flexible and adaptable way that the OA Services can be used across different thematic domains and in different organisational contexts, and that the update of integrated components (e.g. applications, systems, ontologies) causes little or ideally no changes to the users of the OA Services.

Note: The functional classification of the ORCHESTRA Service Types into application domain-independent and dependent service types (see section 9.3) reflects this architectural requirement.

6.3.7 Self-describing Components

OSN components, such as data elements or services, shall include descriptions of their critical characteristics, including sources, assumptions, etc. The usage of self-describing components that provide context-sensitive formal and semantic descriptions of their interfaces can help to realise semantic interoperability.

Note: An example of how the ORCHESTRA Architecture considers the concept of self-describing components is the mandatory support of the service capabilities interface (see section 9.6.1) that allows a service consumer to learn about the capabilities and the characteristics of a service implementation.

7 Design Decisions of the ORCHESTRA Architecture

The ORCHESTRA Architecture is the combined specification of the ORCHESTRA Information and Service Viewpoints. Both of these viewpoints are specified in dedicated sections (see section 8 for the Information Viewpoint and section 9 for the Service Viewpoint).

However, as concepts introduced in one viewpoint are required for the specification of the other viewpoint, a purely sequential description is not possible. Important design decisions that are not specifically allocated to one of these viewpoints have to be presented in advance. Note that sometimes they are just introduced here but further refined in the respective section. In this case, a forward reference is used.

7.1 Functional Domains of the ORCHESTRA Service Network

The ORCHESTRA Architecture has to face the problem of integrating environmental risk management systems that are networked across and between organisations. It's the OSN, as the running instance of an ORCHESTRA Architecture, that contributes to improve the syntactic and the semantic interoperability of these systems.

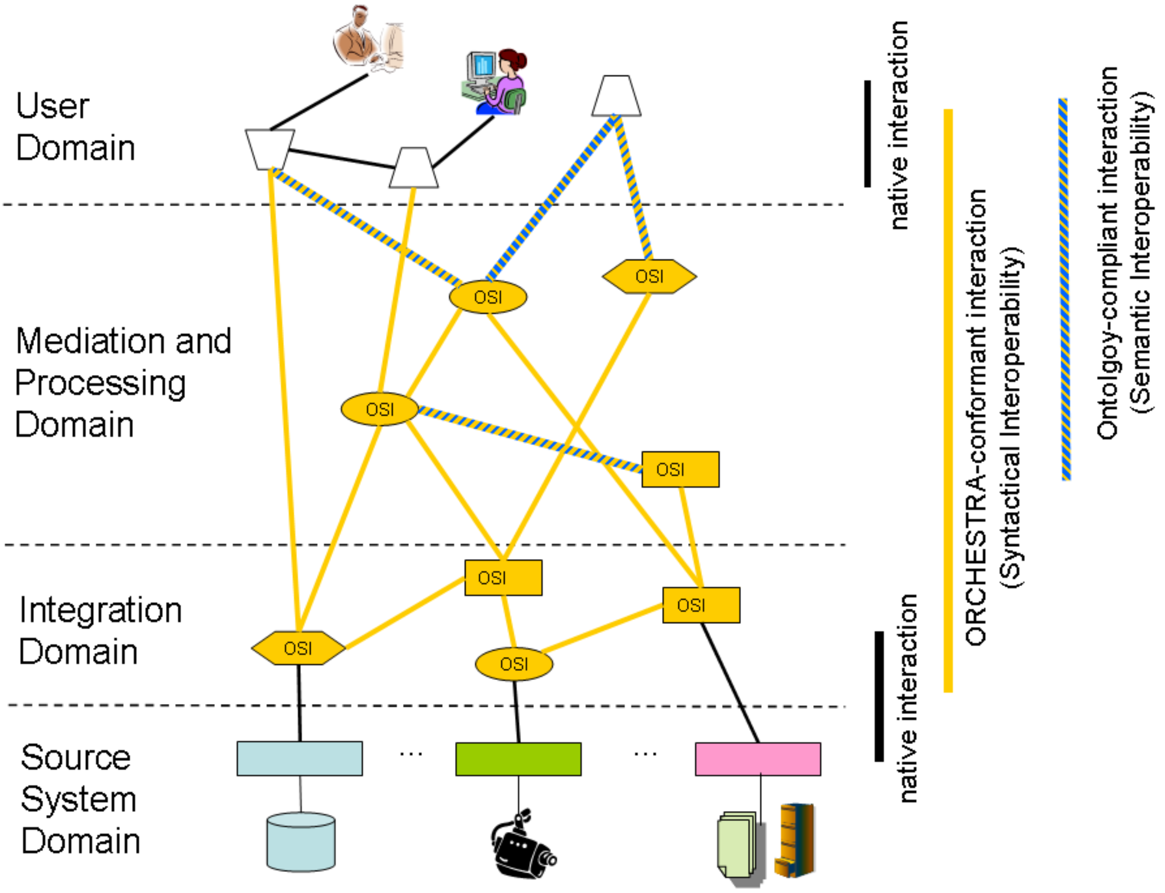

User Domain
Mediation and Processing Domain
Integration Domain
Source System Domain
OSI
OSI
OSI
OSI
OSI
OSI
OSI
OSI
OSI
...
...
native interaction
ORCHESTRA-conformant interaction (Syntactical Interoperability)
Ontolgoy-compliant interaction (Semantic Interoperability)
native interaction

Figure 7: Functional Domains in an ORCHESTRA Service Network

The components of an ORCHESTRA Service Network, i.e. the ORCHESTRA Service Instances (OSIs) are classified according to the following functional domains (see Figure 7):

- User Domain: provides the interface to a user component (a human or a software component) and interacts with the OSIs of the Mediation and Processing Domain according to the rules of the ORCHESTRA Meta-Model. However, user components as part of a (distributed) application may interact among themselves in a native way.
- Mediation and Processing Domain: provides the main functional part of the OSN. It mediates

the service calls from the User Domain to the Integration Domain based on meta-information exchanged with the components of the Integration Domain (e.g. by means of a publishing or a retrieval pattern). Note that the implementation of the services in the Mediation and Processing Domain may be designed themselves as a distributed, possibly functionally-redundant system. The interactions between the OSIs within the Mediation and Processing Domain and with the OSIs in the Integration Domain perform solely according to the rules of the ORCHESTRA Meta-Model. Furthermore, dedicated OSIs in this domain aim at resolving the semantic differences between the information models of the source systems by means of ontologies. Thus, the Mediation and Processing Domain enables semantic interoperability if required by the components of the User Domain.

- Integration Domain: provides support for the source system integration (see below). The OSIs in this domain have two-side interfaces. On the one hand, they interact according to the OMM rules with other OSIs in the Integration Domain and the Mediation and Processing Domain. On the other hand, they interact with the components of the Source System Domain according to their native interface. Thus, the Integration Domain enables syntactic interoperability within an OSN.
- Source System Domain: incorporates the systems and system components of a thematic application area (e.g. risk management) to be coupled. They provide the source of data and functionality and are thus referred to as source systems in the following (see also section 7.6). By means of integration OSIs in the Integration Domains, these source systems are connected to the Mediation and Processing Domain. In practice, this means they need to identify the data and the functionality to be offered in an OSN and to wrap it by respective software components with an ORCHESTRA-compliant service interface. For tightly coupled software systems, this may imply a considerable re-engineering effort.

Note: The platform domain is not visible in Figure 7. It provides the basic communication and encoding mechanisms for the service interactions (the service infrastructure). Its specification is outside the scope of the ORCHESTRA Architecture. The ORCHESTRA Architecture only makes assumptions about the characteristics of the platform (see section 9.2.2.2). Furthermore, in some cases, e.g. in the domain of access control, the platform directly provides support for the implementation of ORCHESTRA Services (see section 10.2).

7.2 The ORCHESTRA Meta-model Approach

7.2.1 Overview

By definition, the ORCHESTRA Architecture shall be generic in the sense that it does not prescribe a specific information model nor an exact configuration of ORCHESTRA Service Instances in an OSN for a given application domain problem. To summarise, the OA is not the specification of a particular information system, but it provides a specification framework for distributed information systems to be used by information modellers and OSN designers. This specification framework provides a set of basic elements to be used and a set of rules to be observed for the purpose of enabling syntactic interoperability between the software components of an ORCHESTRA Application.

Note: Specific rules for semantic interoperability will be added in version 4 of the RM-OA.

These rules and basic elements are summarised in the so-called ORCHESTRA Meta-model (OMM). The OMM consists of two parts:

- The ORCHESTRA Information Meta-model (OMM-Information) that is specified as part of the Information Viewpoint in section 8.7. OMM-Information provides rules about how to specify the application schemas for information models and meta-information models and prescribes the usage of data types.
- The ORCHESTRA Service Meta-model (OMM-Service) that is specified as part of the Service Viewpoint in section 9.2. The OMM-Service provides rules about how to specify interfaces and ORCHESTRA Services and proposes a set of architectural services to be used in an OSN.

Both parts of the OMM are interrelated and depend on each other:

- On the one hand, the structure of the input and output parameters of interface operations have to obey the rules of the OMM-Info.
- On the other hand, built-in operations on feature types have to obey the rules of the OMM-Service.

Note: For convenience, if there is no need to explicitly distinguish between OMM-Information and OMM-Service, the RM-OA simply uses the term OMM to refer to the respective meta-model.

The OMM serves as the basis for checking the conformance of information models and service specifications with respect to the RM-OA. Thus, it has to be specified in detail in a formal and unambiguous way. For convenience, as the OMM is defined in a very formal way as part of the Information and Service Viewpoint the major characteristics are summarised in an informal manner in the following sub-sections.

7.2.2 Major Characteristics of the ORCHESTRA Information Meta-model

In the context of an OSN, information models are specified in order to yield a structure for the information that is potentially being exchanged with an ORCHESTRA Service, i.e. they comprise the structure of the service parameters. The role of the OMM-Information is thus to deliver rules for the specification of such information models (called ORCHESTRA Application Schemas, OAS) with the aim of achieving a harmonised approach for all service specifications and therefore contributing to improved re-usability and interoperability of software components.

The OMM-Information is basically an extension of the General Feature Model (GFM) as defined in ISO 19109. The OMM mandates the usage of UML 2.0 as conceptual schema language.

The central concept in the OMM-Information is that the feature is the basic informational unit as perceived by ORCHESTRA Applications. OMM-Information is a meta-model for feature types. A feature is an abstraction of a real world phenomenon whereby the “real world” explicitly includes hypothetical worlds. Individual features (or feature instances) are grouped into feature types where all instances of a certain type are described by common characteristics.

A feature type contains a set of properties which may be either attributes, operations or associations with other feature types. Furthermore, feature types may be refined by means of inheritance.

The OMM-Information provides rules for the usage of the value domains of attribute type definitions. First of all, for all attribute types it defines a list of basic data types to be used (mostly based on ISO 19103). However, attribute types are further classified into temporal, spatial, location and thematic attribute types with the obligation to use the respective ISO standard definitions (e.g. ISO 19107 and ISO 19125-1 for spatial attribute types).

Attribute types may also represent meta-information about other resources of an OSN. Here, the OMM does not follow the GFM approach of ISO 19109 by strictly requiring the use of ISO 19115. Instead, according to the meta-information approach of ORCHESTRA (see section 7.4), meta-information is always purpose-specific and thus “the” single meta-information model may not be specified. The usage of ISO 19115 in order to define the value domain of meta-information attributes is thus just one of many options.

7.2.3 Major Characteristics of the ORCHESTRA Service Meta-model

The basic structural unit in the ORCHESTRA Architecture as a service-oriented architecture and in an OSN is, of course, the concept of an ORCHESTRA Service. Thus, service modelling plays the predominant role in the specification phase. According to the ORCHESTRA Reference Model, an ORCHESTRA Service is being specified as an ORCHESTRA Service Type, implemented as an ORCHESTRA Service Component (OSC) and executed as an ORCHESTRA Service Instance (OSI).

The OMM-Service provides a meta-model and associated rules for the specification of ORCHESTRA Service Types. Particular emphasis is given to the approach that service modelling is not tied to a particular platform but shall take place on a platform-neutral level (abstract level). The abstraction from platform details improves the mapping from functional user requirements, favours re-use of service specifications for different platforms and enables cross-platform interoperability.

On the abstract level, the purpose and the basic functionality of ORCHESTRA Service Types as seen by the service consumer is described in an abstract description that should be human-readable. The RM-OA proposes the service description framework as introduced in section 9.4 and used later on in the RM-OA for this part. However, there is no formal specification of ORCHESTRA Service Types. Instead, the OMM-Service defines an ORCHESTRA Service Type as a collection of interface types which specify the externally visible behaviour of an ORCHESTRA Service Type. The concept of an interface type aims at aggregating coherent functionality for a particular objective (e.g. rendering of geographic information in a map) such that it may be re-used for other service types. Thus, on the abstract level an interface type is the unit for re-usability. An interface type itself comprises a set of operations which are the individual units of interaction between a service provider and and service consumer. It is specified in an abstract interface specification and uses UML 2.0 as the conceptual schema language. The OMM-Service proposes dedicated stereotypes for UML classes in order to customise UML for this modelling approach.

An operation is specified by its signature, i.e. its name and its request and response (result and exception) parameters. Here the link between the OMM-Service and the OMM-Information becomes visible: The types of the request and response parameters shall be structured as an ORCHESTRA Application Schema (OAS) according to the rules of the OMM-Information. A parameter value may thus be a value of a basic data type (e.g. an integer) but also a collection of feature instances with their attribute values.

On the platform-specific level, an ORCHESTRA Service Type is represented in an implementation specification that is tailored to the needs and capabilities of a given platform. A selected platform shall be specified beforehand in a platform specification.

Derived from the architectural requirement of “rigorous use of standards” (see section 6.3.1) the OMM-Service assumes that the platform properties and especially the conformance guidelines as specified in the OASIS Reference Model for Service Oriented Architecture (SOA-RM 2006) are fulfilled. As an example, the SOA-RM mandates that the SOA approach of a given platform shall describe how visibility is established between service providers and consumers whereby visibility is understood as follows:

- The initiator in a service interaction shall be aware of the other parties (awareness), e.g. effected by means of a discovery mechanism.

  Note: This aspect is supported by the ORCHESTRA Architecture in terms of the Catalogue Service described in section 9.6.6 that shall be available at least in all mediated OSN types (see section 11.2).

- The participants shall be predisposed to interaction (willingness), e.g. a service provider shall respond to an interaction request of a service consumer (except in cases of a denial-of-service attack).

- The participants shall be able to interact (reachability), e.g. it shall be possible to establish a communication path between the participants.

  Note: This aspect is supported by the ORCHESTRA Architecture by the provision of means for OSN administration. See the Service Monitoring Service as described in section 11.2 that shall be available at least in all managed OSN types (see section 11.2).

Such a platform specification shall include a description of the principal way in which the mapping from the abstract level is performed (e.g. how an operation is represented), how synchronous and asynchronous interactions specified on the abstract level are principally implemented and how an OAS is mapped from and to UML to the information model langauge of the platform.

For each service type, the OMM-Service mandates that service mapping from the abstract to the platform-specific level is to be specified as part of the implementation specification. The main rules that control the service specification and the mapping are:

- There may be several implementation specifications for one ORCHESTRA Service Type as the implementation specification is platform-specific. However, the OMM-Service also allows several implementation specifications for the same platform by introducing the concept of a service profile (see below).

- Interface types are not visible on the platform-specific level. Instead, their operations are individually mapped upon the action model (SOA-RM term characterising the permissible actions that may be invoked against a service) of the service.

- All ORCHESTRA Service Types shall support the operations of the interface type *ServiceCapabilities* that provide the means to access the meta-information that is associated with a service (e.g. the supported service type, information about the service provider, the set of implemented operations). A recommendation for a capabilities schema is provided in Annex B1 “RM-OA rules for OAS-MI” of the RM-OA.
- Operations and operation parameters that are marked as optional in the respective abstract interface specifications may be omitted in the mapping to implementation specifications. Thus, service profiles of ORCHESTRA Service Types may be defined, even for the same platform. Their action model thus provides a functional subset of the ORCHESTRA Service Type which is, however, syntacticly and semantically compatible such that generic service consumers (application components) may be realised by knowing only the interface types of ORCHESTRA Service Types and the particular platform characteristics.
- ORCHESTRA Service Types that are classified as OA Services (see below) shall first be specified on abstract level and then mapped to the platform level. For all other service types, even if just specified in a platform-specific implementation specification, at least an abstract description of their basic functionality shall be given.

As a consequence of this approach, a community that applies the OMM to specify their services shall maintain a well-defined list of ORCHESTRA Service Types that is consistent between the abstract level and all supported platforms. The RM-OA incorporates as part of its Service Viewpoint in section 9.4 a description of service types that are derived from functional user requirements. This list is further structured into architectural service types (so-called OA Services) that are application-domain independent but indispensable for the operation of an OSN and thematic services (so-called OT Services) that are tailored towards a given application domain. The RM-OA, being a reference model for an application-independent architecture, just provides descriptions of OT Services that support thematic applications across several domains (so-called OT Support Services). Domain-specific services are outside the scope of the RM-OA.

Specifications of the abstract interfaces of the selected ORCHESTRA Service Types are delivered in (ORCH-AbstrServ 2007).

7.3 Resources in an OSN and their identification

There are two fundamental resources in an OSN that need dedicated identification schemes:

- ORCHESTRA Service Instances (OSIs) as the basic functional unit, and
- Feature instances as the basic informational unit.

7.3.1 Identification of OSIs

An OSN comprises a set of interacting ORCHESTRA Service Instances (OSIs) running on top of hardware components connected through a network. In order to be able to search for an OSI and call its operations, a unique identifier of an OSI within an OSN is needed. This unique identifier is also referred to as the name of an OSI in the following.

The name of an OSI is a logical name which may be generated automatically, i.e. it may not directly be meaningful to a human user.

In the case of a dynamic OSN environment that supports the dynamic assignment of an OSI to several OSNs (i.e. the membership of an OSI to one or several OSNs may change during the lifetime of an OSI) an identifier of an OSI that uses an OSN as namespace is not sufficient. In this case, a globally unique identifier is required in order to avoid renaming of OSIs during their lifetime. This means that different OSIs within the same OSN or within different OSNs shall have different names. The OSI name shall be immutable during the lifetime of the OSI.

A recommendation of a naming policy for OSIs that uses the platform as the namespace for an OSI is described in section 11.3.1, however, the usage of this policy is not obligatory. Other naming policies may be defined. The selection of a naming policy for OSIs is one of the characteristics of an OSN as described in the Engineering Viewpoint of the RM-OA (see section 11.1).

Note: It has to be distinguished between:

- name of an OSI
- platform-specific identifier of an OSI (e.g. its URI)
- platform-specific address of an OSI (e.g. its IP-address and port)

The focus here is on OSI names and the mapping between OSI names and their platform-specific identifiers. These tasks are related to an OA Service which is called Name Service and introduced in section 9.6.7. The mapping between platform-specific identifiers and addresses is done by platform-specific mechanisms and is out of scope of this document.

7.3.2 Identification of Features

In the same way as an unambiguous identifier of an OSI is required to refer to that OSI within an OSN, each feature instance needs to be uniquely identifiable within the OSN. This is required in order to enable software components in ORCHESTRA Applications to work with references to feature instances instead of performing a query each time feature information is needed.

Such a feature instance identifier shall be immutable during the lifetime of the feature instance. This means that while the values of attributes of a feature may change over time, the identifier assigned to the feature shall not change.

A proposal of a naming policy for feature instances that uses a Feature Access OSI as namespace is described in section 11.3.2, however, as for OSI names, the usage of this policy is not obligatory and other naming policies for feature instances may be defined. The selection of a naming policy for OSIs is one of the characteristics of an OSN as described in the Engineering Viewpoint of the RM-OA (see section 11.1).

Note that the naming policy of feature instances has to be distinguished from the semantic identity of two feature instances having different names but possibly representing the same real-world phenomenon.

7.4 Meta-information

The terms data, metadata, meta-data, metainformation, information, meta-information, and meta-information are used in different places in the literature, and on the Web.

While most authors clearly distinguish between “data” and “information”, the terms meta-data and meta-information are often used interchangeably. In ORCHESTRA, the meaning of data is only given by the underlying information model, and certain pieces of data may have very different meanings depending on the information model. When referring to certain data in the context of a meta-information model, the RM-OA is actually referring to the meaning given to this data within a model.

In order to avoid confusion, and to account for the fact that all data may have different meanings, the term meta-information shall be used in all the ORCHESTRA documents whenever a datum is seen in the context of a meta-information model (see the RM-OA Annex A3). The related terms, including “metadata”, “meta-data”, and “metainformation” must not be used in the specification parts of ORCHESTRA documents.

The architectural approach to include meta-information in the OA and in the OMM is provided as part of the Information Viewpoint in section 8.4. The argumentation and the foundation for this approach are given in Annex A3 of the RM-OA. A detailed specification of rules and examples is given in Annex B1 of the RM-OA.

7.5 User Management, Authentication and Authorisation

7.5.1 Overview

The access to resources for both feature and service instances is controlled by authentication and authorisation mechanisms. Access encompasses access from human users but also from software components. This is handled by three services: the User Management Service (see section 9.6.7), the Authentication Service (see section 9.6.10) and the Authorisation Service (see section 9.6.9), together referred to as UAA services in the following. An example of their combined usage is described in an OA pattern in section 9.9.1. The general question how many instances of the UAA services are present in an OSN and how they are configured is discussed in the context of UAA policies in the Engineering Viewpoint in section 11.1.5.

The following section just introduces the basic terms and concepts.

7.5.2 User Management based on Subjects, Groups and Principals

The major concepts of the ORCHESTRA User Management are subjects and principals.

A **subject** is an abstract representation of a user or a software component in an ORCHESTRA Application. Subject attributes are intended to store generic information about subjects (e.g. first name, last name, address, e-mail, ...).

Subjects need to be authenticated. However, the concept of a subject itself cannot be used for the authentication process. This is mainly because ORCHESTRA aims at supporting multiple authentication paradigms and mechanisms. Their potentially simultaneous use leads to a number of implications, e.g. different authentication mechanisms use different subject representations. Thus, a single subject representation cannot be chosen for ORCHESTRA.

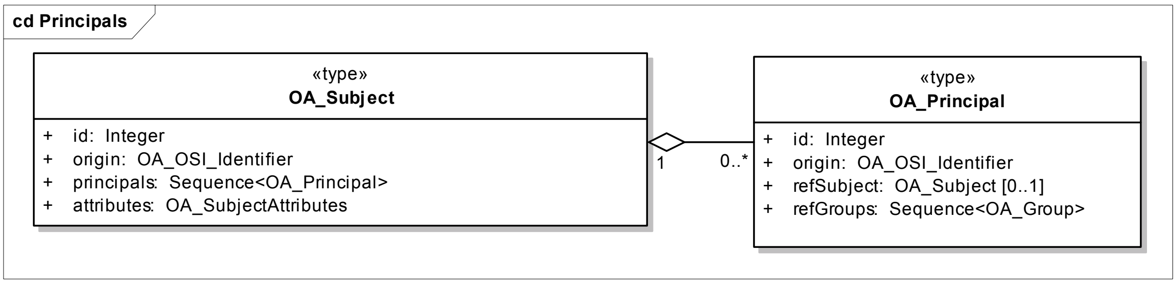

cd Principals
«type»
OA_Subject
+ id: Integer
+ origin: OA_OSI_Identifier
+ principals: Sequence<OA_Principal>
+ attributes: OA_SubjectAttributes
1
0..*
«type»
OA_Principal
+ id: Integer
+ origin: OA_OSI_Identifier
+ refSubject: OA_Subject [0..1]
+ refGroups: Sequence<OA_Group>

Figure 8: Relationship between Subject and Principal

To solve the representation problem, a subject is decoupled from authentication. This decoupling is done by introducing a further concept called a principal. A **principal** is an identity of a subject whereas authentication indicates whether a subject is allowed to use a certain principal. One subject may have multiple principals as illustrated in Figure 8.

Since each authentication mechanism can have its own way of representing a principal, the UAA concept defines a superclass OA_Principal that just contains some attributes used for collaboration purposes (like identifying a principal and referring to the related subject). All attributes that are specific for an authentication mechanism may then be realised by subclasses of OA_Principal.

A **group** is a special subject. A group can have one or more principals (group principals). In addition to principals identifying the group itself a group can have one or more principals as members (member principals). This relationship is illustrated in Figure 9.

Member principals are assigned to a group to define memberships of certain principals.

Based on these concepts, user management is the process of creation and management of subjects, including groups (of principals) as a special kind of subjects. Furthermore, it is up to the User Management Service to associate principals with subjects. The creation and management of principals is up to the Authentication Service.

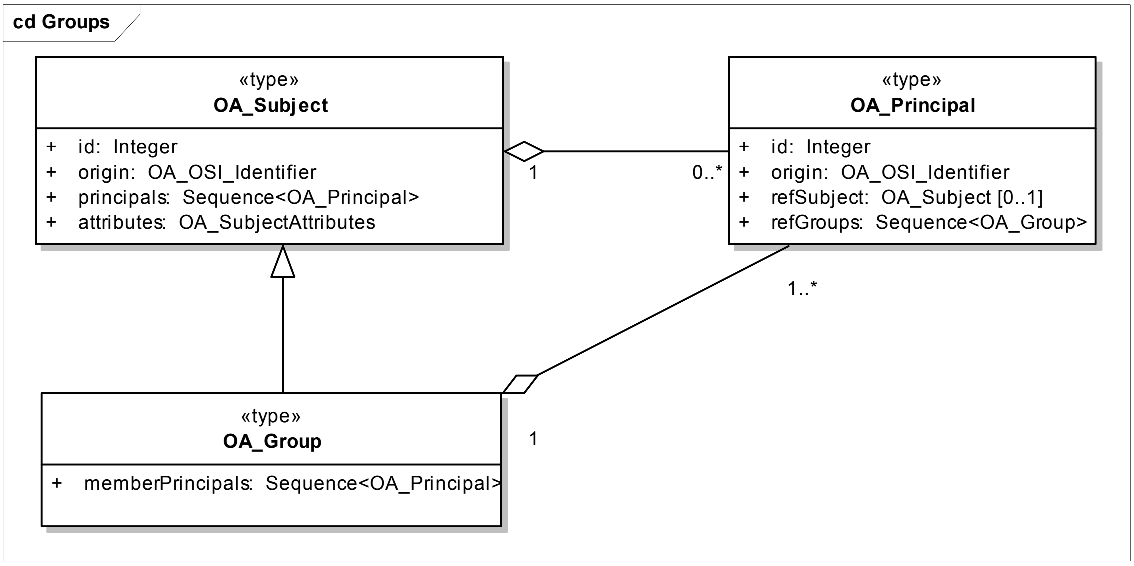

cd Groups
«type»
OA_Subject
+ id: Integer
+ origin: OA_OSI_Identifier
+ principals: Sequence<OA_Principal>
+ attributes: OA_SubjectAttributes
«type»
OA_Principal
+ id: Integer
+ origin: OA_OSI_Identifier
+ refSubject: OA_Subject [0..1]
+ refGroups: Sequence<OA_Group>
1
0..*
1..*
«type»
OA_Group
+ memberPrincipals: Sequence<OA_Principal>
1

Figure 9: Relationship between Subject, Group and Principal

7.5.3 Authentication

Authentication is the process of verifying the principal of a certain subject. In other words, within an authentication process a subject proves that it is allowed to act with the corresponding principal. Generally speaking, this proof can depend on a secret (credentials) that can be, for example:

- what somebody has (key, smart card, …)
- what somebody knows (password, …)
- what somebody is (biometrical data, …)
- the place somebody resides (certain computer, …)
- the skills of somebody (handmade signature)

The result of an authentication process starts a session that is represented by session information (see section 7.5.5).

Note: As the session information represents the state of the session and must be passed in each service interaction request, it is an example where stateful services are required (see the assumptions of a Simple Service Architecture according to ISO 19119 described in section 5.4.3).

Principals are created and managed in instances of Authentication Services. The process of creating a new principal depends on the authentication mechanism used by the corresponding Authentication Service instance.

After authentication has successfully been passed the Authentication Service generates session information containing the information about which principal has been authenticated.

As an example, consider an OSI of an Authentication Service wrapping an existing Kerberos authentication. Usually a Kerberos implementation ships with a solution for user management. A user in the Kerberos user management becomes an ORCHESTRA principal. This principal then will be associated with the corresponding subject using the ORCHESTRA User Management Service.

7.5.4 Authorisation

Authorisation is the process of determining whether a subject is allowed to have the specified types of access to a particular resource (data or services). This is done by evaluating applicable access control information contained in a so-called authorisation context.

Usually, authorisation is carried out in the context of authentication. Once a subject is authenticated through its principal, it may be authorised to perform different types of access. This is carried out through the concept of permissions that are attached to principals.

A service requests an authorisation decision for a given principal and a given authorisation service context. A service requesting an authorisation decision needs to pass session information containing at least one authenticated principal of the service requestor as well as the authorisation context. Since permissions are bound to principals, the Authorisation Service is able to retrieve permissions for a given principal. There is no restriction on how permissions are associated with principals. This might be done directly or indirectly using roles, for example.

The connection of permissions and principals is essential to the UAA concept by enabling the decoupling of authentication and authorisation. An Authorisation Service may assign permissions to every ORCHESTRA principal, regardless of the mechanism that has been used to authenticate it. This possibility is important. If there is a problem with interoperability – maybe because clients do not support a certain authentication mechanism of a foreign authentication service – they can still use every ORCHESTRA service as long as the corresponding service provider is willing to assign permissions to the client principals.

A group (see Figure 9) can be treated as an ordinary subject by Authorisation Service instances. Thus, assigning permissions to a group does not differ from assigning permissions to any other subject.

Authorisation Services may use different authorisation paradigms. These paradigms can be classified into lookup and expression-based access control.

Lookup based paradigms use predefined data structures to retrieve authorisation decisions. The most famous representative is the role-based paradigm.

Example:

A role-based access control (RBAC) system might use the information model illustrated in Figure 10.

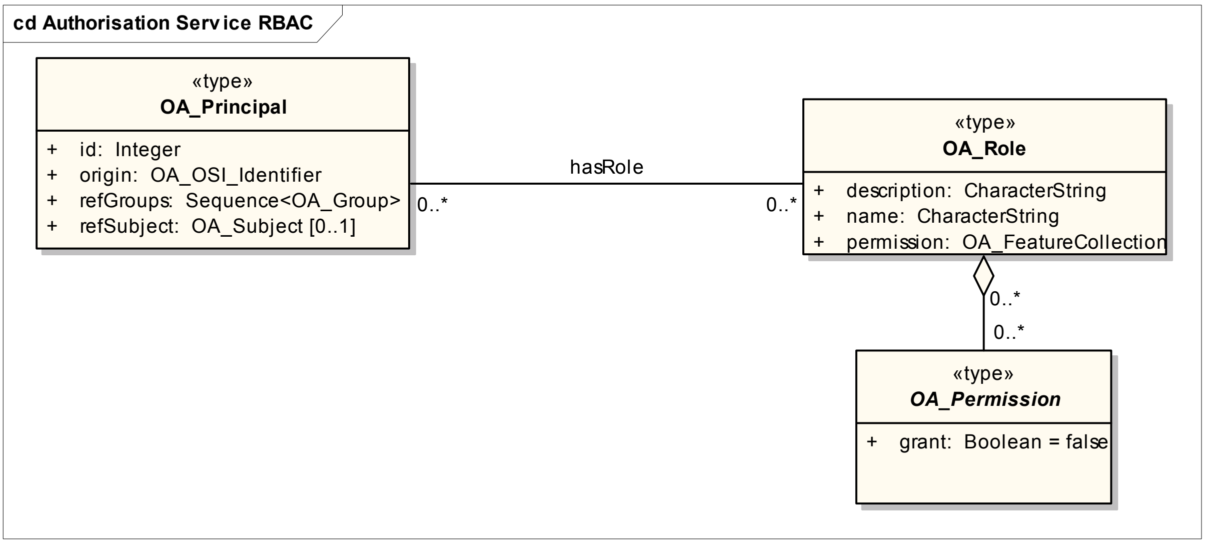

cd Authorisation Service RBAC
«type»
OA_Principal
+ id: Integer
+ origin: OA_OSI_Identifier
+ refGroups: Sequence<OA_Group>
+ refSubject: OA_Subject [0..1]
hasRole
0..*
0..*
«type»
OA_Role
+ description: CharacterString
+ name: CharacterString
+ permission: OA_FeatureCollection
0..*
0..*
«type»
OA_Permission
+ grant: Boolean = false

Figure 10: Schema of Role-based Access Control

Expression-based access control systems (EBAC) do not exclusively rely on predefined lookups. More than that, these systems define a framework to specify authorisation conditions. These conditions are parameterised and evaluated in order to compute authorisation decisions. Evaluation of expressions is done by a separated interpreter. This interpreter contains the computational logic and therefore forms the core of each EBAC.

The most popular representatives of EBAC systems are trust management systems.

7.5.5 Session Information

Session information is created and/or modified by an Authentication Service.

Session information mainly serves as proof that certain principals have been authenticated. Thus, the creation of session information is done by an Authentication Service after successfully authenticating a certain principal.

In order to arrive at an authorisation decision a service needs to know under which principal a service requestor acts. Therefore the requestor of a service has to pass the session information in every interaction with the service instance. Interpretation of the session Information is performed by the invoked service instance.

Verifying and extracting information from session information is a process which is specific to the way session information is treated, e.g. as a session key or as a session envelope. Thus, each service needs to provide a capability, possibly called session handler or session interpreter, which is able to interpret session information as passed from the service requestor.

7.6 Approach to Integration of Source Systems

The OA explicitly takes into account the fact that existing systems and services have to be integrated when designing an OSN. In this respect, it does not matter whether these systems have existed for a long period of time, possibly realised with older technologies, or whether they have been recently designed with modern technology. Thus, the OA uses the term source system to refer to such systems instead of the often-used term legacy system.

A source system is a container of unstructured, semi-structured or structured data and/or a provider of functions in terms of services. The source systems are of a very heterogeneous nature and contain information of a variety of types and and in a variety of formats.

Examples are:

- database containing structured data (e.g. numerical model data), i.e. information that is organised so that it can be easily located, searched, and updated
- database containing semi-structured data (e.g. an XML database)
- database containing unstructured data (e.g. a document archive or image database)
- a system providing services (e.g. a map server)
- Web site, i.e. a provider of a set of html-documents accessible through the W3C http protocol.

For clarification, as illustrated in Figure 11, the OA furthermore distinguishes between an

- External Source System as a source system that does not provide its data and functions through an ORCHESTRA-conformant interface, and
- ORCHESTRA Source System as a source system that provides its data and functions through an ORCHESTRA-conformant interface, in Figure 11 called ORCHESTRA_SourceSystem_IF as an example. This interface shall be built according to the rules as specified in the ORCHESTRA Service Meta-model, in Figure 11 represented by the meta-class OMM_InterfaceType as specified in section 9.2.4.1

Each ORCHESTRA Source System is associated with at least one External Source System.

Thus, the major development process for an OSN designer is the process of transforming an External Source System into an ORCHESTRA Source System which is called source system integration.

The OA approach for source system integration is specified in the RM-OA Service Viewpoint in section 9.9.2 as part of the recommended patterns of OA Service usage. The consideration of source systems for the OMM is specified in the RM-OA Information Viewpoint in section 8.5.

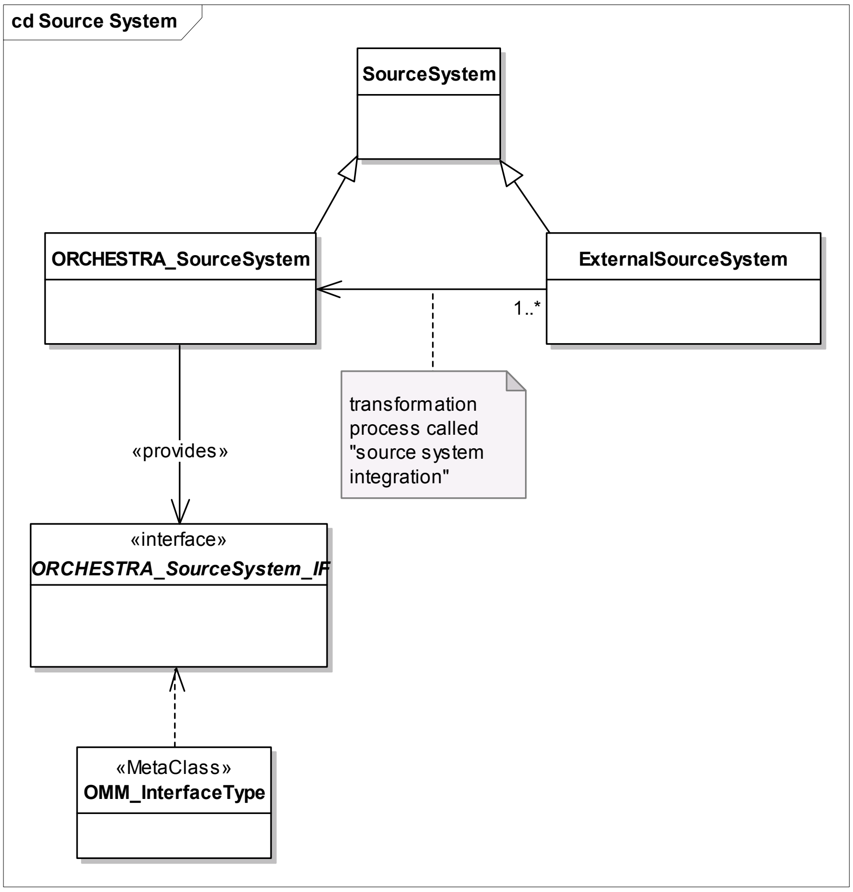


Figure 11: External and ORCHESTRA Source Systems

Note: A future RM-OA version will tackle the particular problem of integrating the (proprietary) UAA solutions that are already implemented in source systems and their environment into the UAA policy of an OSN.

7.7 Service Interaction Modes

ORCHESTRA Services will support at least two interaction modes at the conceptual level for the processing of their operations:

- Synchronous mode: In this mode, the requestor principally waits for the response and the response contains the requested data in its output parameters. This mode is usually applied for all operations with a relatively short response time.
- Asynchronous mode: In this mode, the requestor just issues the request for the operation, continues its work in parallel and is asynchronously informed about the availability and a reference to the results. This mode is usually applied for all operations with a longer response time.

Note 1: In the future, a mixture of these modes and other variants may be investigated.

Note 2: These modes are described on the conceptual level which is reflected in respective interfaces of the abstract specification of the OA Basic Services (see section 9.6.1). It does not imply any constraints on the application programming interface in an implementation. This means that a synchronous operation on the conceptual level may be implemented in an asynchronous way and vice versa.

7.8 Interoperability Between Different Service Platforms

Conceptually, there are the following two possible ways to map an OSN onto service platforms:

1. There is exactly one platform assigned to the OSN. In this case, all interactions between all OSIs that participate in the OSN shall follow the rules of this platform (see Figure 12) with the dotted lines representing the logical interaction relationships between the OSIs.

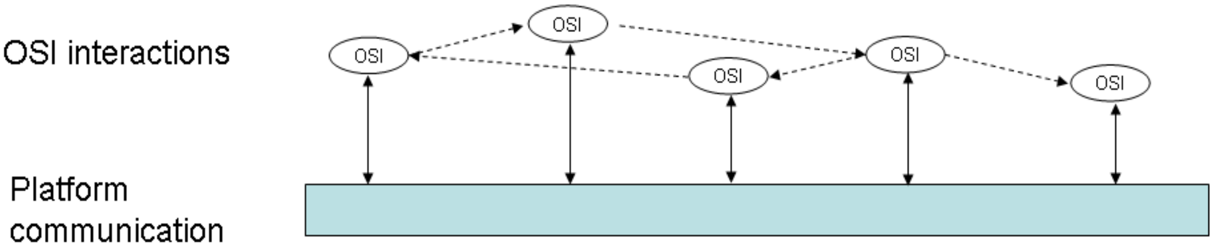


Figure 12: OSI interactions in one platform domain

2. There are several platforms assigned to the OSN sub-dividing the platform into several platform domains. In this case, all interactions between all OSIs that participate in the OSN and belong to the same platform domain shall follow the rules of the respective platform. Furthermore, it must be ensured that all interactions between OSIs that belong to different platform domains are made possible by the provision of respective service platform gateways (see Figure 13). An example for such a situation is a gateway that maps between a CORBA-based platform and W3C Web Services.

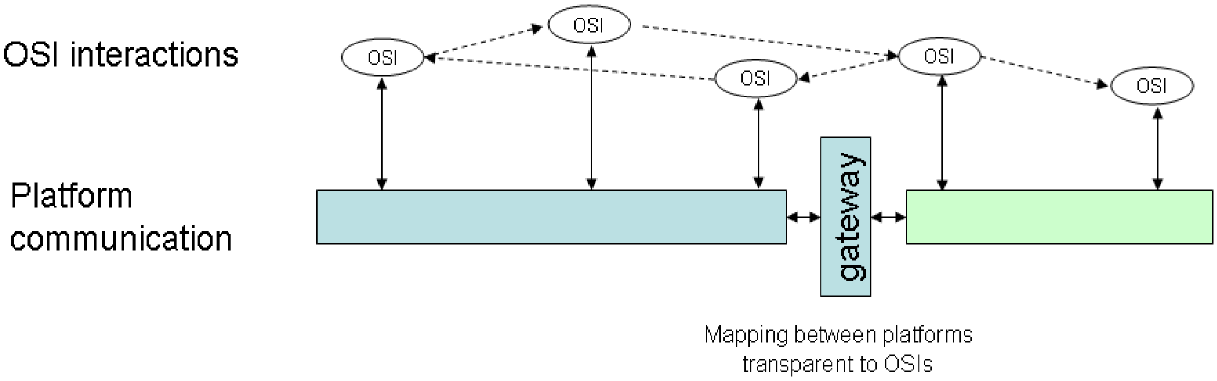


Figure 13: OSI interactions across platform domains

Note: Currently, the RM-OA is restricted to possibility 1, i.e. an OSN may only run on top of one platform that is specified in a given platform specification. Possibility 2 will be considered in version 4 of the RM-OA together with a more detailed discussion about platform gateways.

8 Information Viewpoint

8.1 Overview

The Information Viewpoint of the ORCHESTRA Reference Model specifies the modelling approach for all categories of information the OA deals with, including their thematic, spatial and temporal characteristics as well as their meta-information. The ORCHESTRA Reference Model does not specify an information system. Instead it provides a framework for distributed information systems and ORCHESTRA Applications based on a service-oriented architecture. As such, the Information Viewpoint of the ORCHESTRA Reference Model provides an integrated specification framework in order to support a formal specification of conceptual ORCHESTRA information and meta-information models in the context of ORCHESTRA Applications.

This specification framework encompasses the following levels:

- source system level
- feature level
- schema level
- meta-model level
- semantic level

The source system level comprises all the existing systems that contain relevant data or provide relevant services in order to fulfil a particular objective of an application or end-user task (see also the ORCHESTRA functional domains in section 7.1).

The feature level provides an informational view of the data and services of the source system level according to the rules specified for ORCHESTRA features (see section 8.2). Note that no semantic concepts are considered on this level.

The schema level delivers the structuring of information on the feature level in terms of application schemas. Application schemas provide formal specifications of ORCHESTRA Information Models.

The meta-model level provides rules to define application schemas.

The semantic level provides semantics to the information specified in the other levels through explicit consideration of ontologies defined and shared in user communities.

The following sections describe the framework for ORCHESTRA Information Models in two steps:

- In a first step, just the meta-model, the schema and the feature level aspects are considered. For these levels, a specification framework for information models is specified (see section 8.3) and then extended by the consideration of meta-information (see section 8.4).
- In a second step, the specification framework is enriched by considering aspects of the source system level (see section 8.4.4) and the semantic level (see section 8.6).

8.2 The ORCHESTRA Definition of a Feature

One basic concept of the RM-OA Information Viewpoint is the feature, where a feature is an abstraction of a real world phenomenon perceived in the context of an ORCHESTRA Application. A digital representation of the real world can be thought of as a set of features. These individual features (or feature instances) are grouped into feature types where all instances of a certain type are described by common characteristics. The characterisation of features into feature types typically depends on the particular application and is captured in an application schema. This process is shown in Figure 14.

Note: Features have often been understood just as geographic features, i.e. as a feature associated with a location relative to the Earth. The ORCHESTRA definition of features explicitly goes beyond geographic features. It includes tangible objects of the real world but also abstractions, concepts or software artifacts (e.g. documents, software components of IT systems) that may have a physical representation only in software systems. These features may, but need not, have spatial characteristics. The ORCHESTRA understanding of a “real world” explicitly comprises these hypothetical worlds or worlds of human’s thoughts.

Note: For a future version of the RM-OA, we will investigate whether a distinction between feature types in the real and in the hypothetical world is useful, as the conventional understanding (e.g. within OGC) does not follow the above approach.

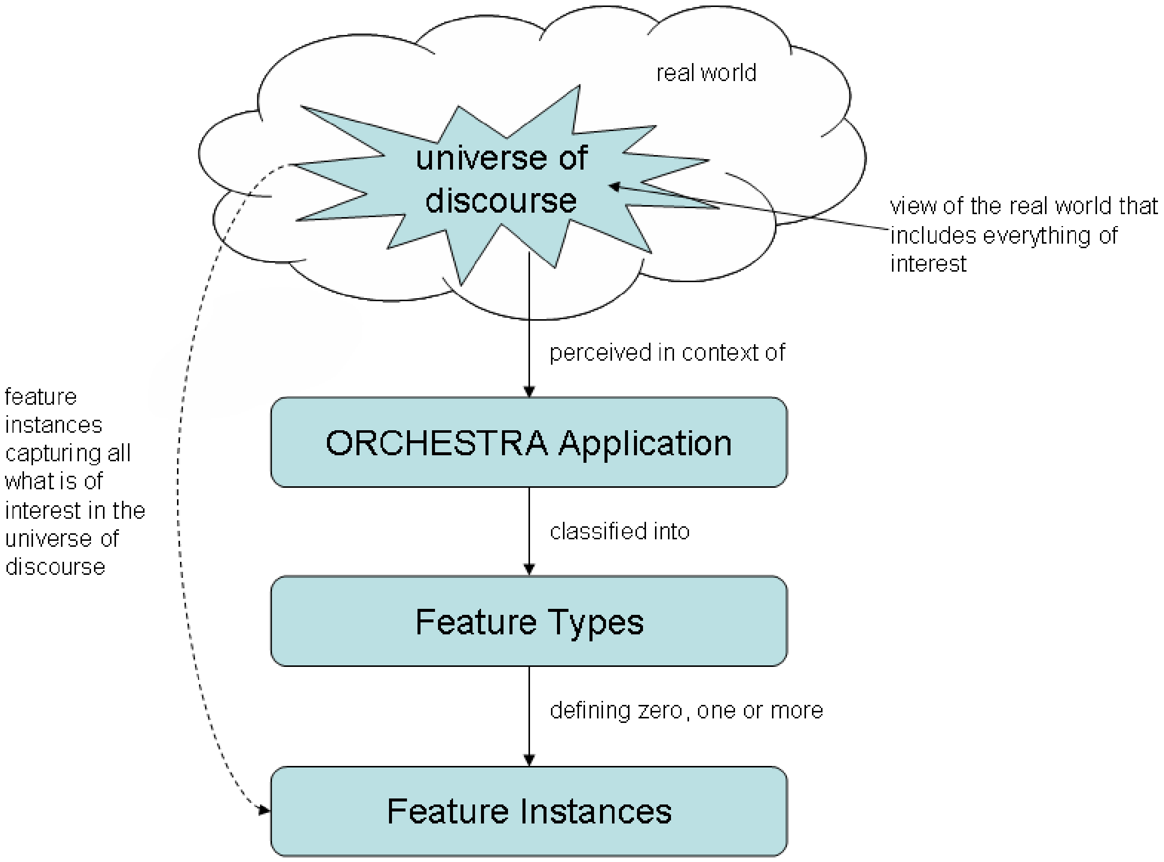

real world
universe of discourse
view of the real world that includes everything of interest
perceived in context of
ORCHESTRA Application
classified into
Feature Types
defining zero, one or more
Feature Instances
feature instances capturing all what is of interest in the universe of discourse

Figure 14: From phenomena to feature instances (derived from ISO 19109)

Common concepts of all application schemas are expressed in the ORCHESTRA feature model as specified in the ORCHESTRA Meta-Model (see section 8.7). Relationships between feature types are feature association types and inheritance. Properties of feature types are feature attributes, feature operations and feature association roles.

Any feature may have a number of such properties. Any feature may have a number of attributes, some of which may be numeric, a spatial geometry, meta-information, temporal information, etc.

Examples of features types are earthquake, forest fire, road, building, water protection area, and monitoring station, but also sensor observation, measurement value, document, and equation.

Examples of feature instances are

- for the feature type “earthquake” the Indian Ocean Tsunami December 26, 2004,
- for the feature type “water protection area” the “Wasserschutzgebiet Seewiesenquellen ID=3463” in the German Federal State of Baden-Württemberg,
- for the feature type “forest fire” the “forest fire near Fréjus in southern France started on July 6, 2005”, or
- for the feature type “document” the “RM-OA Version 1.9 dated July 22, 2005”.

8.3 Framework for ORCHESTRA Information Models

The framework for ORCHESTRA information models distinguishes between

- the ORCHESTRA Meta-Model (OMM) (for information) on the meta-model level,
- ORCHESTRA Application Schemas (OAS) on the schema level and
- ORCHESTRA Feature Sets (OFS) on the feature level.

The OMM specifies the common specification framework for all feature-based application schemas used within ORCHESTRA. It is a meta-model and defines rules for the specification of an OAS. An OAS formally specifies the feature types and their properties which are relevant for a specific information model used in an OSN. It is expressed using the conceptual schema language UML.

The OMM is an evolution of, but it is not a profile of the General Feature Model (GFM) of ISO 19109.

A set of feature instances following the information model formally specified in an OAS is called an ORCHESTRA Feature Set (OFS).

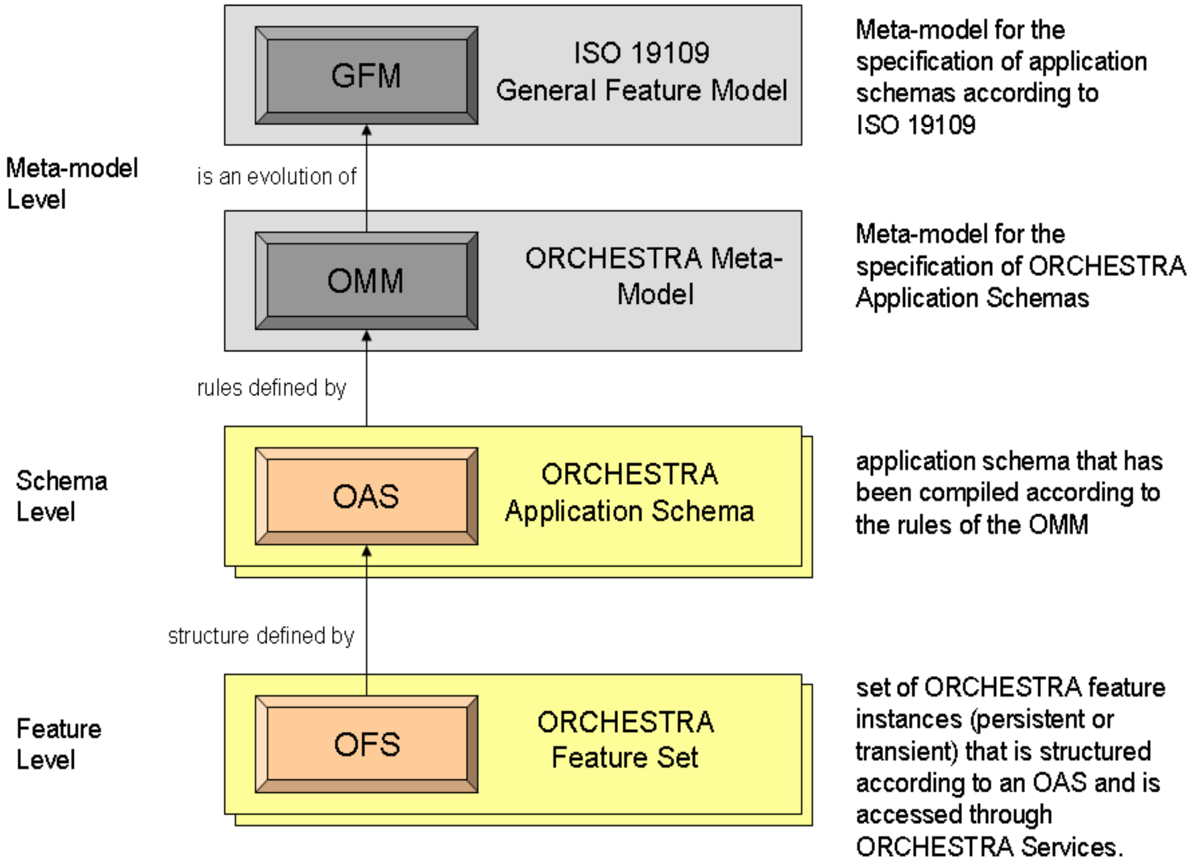

GFM
ISO 19109
General Feature Model
Meta-model for the specification of application schemas according to ISO 19109
Meta-model Level
is an evolution of
OMM
ORCHESTRA Meta-Model
Meta-model for the specification of ORCHESTRA Application Schemas
rules defined by
Schema Level
OAS
ORCHESTRA Application Schema
application schema that has been compiled according to the rules of the OMM
structure defined by
Feature Level
OFS
ORCHESTRA Feature Set
set of ORCHESTRA feature instances (persistent or transient) that is structured according to an OAS and is accessed through ORCHESTRA Services.

Figure 15: Framework for ORCHESTRA Information Models

8.4 Framework for ORCHESTRA Meta-Information Models

8.4.1 Overview

The following definition for meta-information, which is derived from the principle ideas as described in the Annex A3, is applied for the RM-OA:

Meta-information is descriptive information about resources in the universe of discourse. The structure of the meta-information is given by a meta-information model that depends on a particular purpose. The terms used in this definition are used in the following sense:

- Resources are either functions (possibly provided through services) or data objects.
- Universe of discourse: view of the real or hypothetical world that includes everything of interest (see ISO 19101 and also section 8.2).
- Particular purpose: A purpose of meta-information describes the goal of the usage of the resources. The particular purpose also determines the set of resources in the universe of discourse that are to be considered.
- Meta-information model: a meta-information model represents an implementation of a conceptual model for meta-information. It is represented by an ORCHESTRA Application Schema for Meta-information (OAS-MI).

The above definition indicates that a resource by itself does not necessarily need meta-information. The need for meta-information arises from additional tasks or a particular purpose (like catalogue organisation) where many different resources must be handled by common methods.

Common characteristics of resources in the context of a specific purpose are to be described by means of a meta-information model (concrete by an OAS-MI) that shall be suitable and sufficient in order to define respective algorithms. This means:

1. All information needed to fill up the meta-information model is "meta-information" for this particular purpose.
2. Only attributes of the resources that are also specified in a particular meta-information model are candidates to be meta-information attributes. Specific attributes of the resources that are not specified in a meta-information model are consequently not considered as meta-information for this particular purpose.
3. Meta-information may also be implicitly derived from the existence or content of the resources without requiring that this information be explicitly specified as attribute of the resources. Examples here are the results of annotation services for documents or services that generate meta-information according to a given ontology. This process is known as "classification" in the domain of the Semantic Web.

Thus, the ORCHESTRA Architecture does not define "the" single meta-information model which is valid for any purpose. Instead, in the RM-OA Annex B1, ORCHESTRA defines rules which a meta-information modeller will have to apply to build OAS-MIs related to a dedicated ORCHESTRA Application Schema (information model).

The development process of a meta-information model for data and/or services is guided by the fact that it is necessary to know the purpose of the meta-information. The following approach should be taken:

1. Find the purposes (use cases/functions) in the context of users and/or machines like search, retrieve, etc. (see below).
2. Develop the meta-information model(s) for data and/or services in the respective context.
3. Based on the ORCHESTRA meta-information rules specified in Annex B1 and on the above (step 2) developed meta-information model specify your OAS-MI.

In order to simplify the above process for writing OAS-MIs, Annex B1 offers several example OAS-MIs as a recommendation which can be combined in arbitrary ways to cover a great variety of real world needs.

The RM-OA defines a set of rules for specifying OAS-MIs for the following “well-known” particular purposes that are further explained in the subsequent sub-sections:

- discovery (including search and navigation)
- access, storage and service invocation
- integration (collaboration, including orchestration and choreography of services)
- interpretation
- user profiling
- authentication, authorisation, and accounting (AAA)
- quality control/management
- transactions, synchronisation and locking
- OSN configuration and management

8.4.2 Description of Purposes

8.4.2.1 Purpose “Discovery”

The purpose “discovery” encompasses methods to find relevant resources within a set of resources, namely search and navigation.

The procedure of searching starts with formulation of a search query that is submitted to the search engine. The search engine returns a number of resources that it has identified as relevant with respect to the query (the search results). Then, the initiator of the query can select resources from the results and/or refine the query.

Examples of meta-information supporting the search procedure are keyword lists, full text index, bounding areas or gazetteer mapping. Examples of services are the Document Access Service and the Gazetteer Service.

Navigation is the process of finding relevant information by browsing within navigational structures. These are provided either by a static or a dynamic catalogue. Examples of meta-information supporting navigation are catalogue entries or catalogue structures; an example of a service is the “Catalogue Service”.

Discovery of services requires a specific meta-information model and dedicated query languages to access the meta-information entries. The type of meta-information needed depends on the quality of the discovery process: discovery might be user driven and based only on syntactic attributes, or it might be automated and based on semantic descriptions.

8.4.2.2 Purpose “Access, Storage and Service Invocation”

The purposes “access” and “storage” are concerned with meta-information needed to access and store data such as exact location information, access protocol, login information, and access rights (see, for example, the authorisation context of the Authorisation Service as described in section 9.6.9). The storage and retrieval will be handled by a “data access service” (in the case of the RM-OA e.g. the Feature Access Service as described in section 9.6.2), so that data access is a specialisation of a service invocation.

Specific meta-information is needed for the purpose of automated “service invocation” based on semantic service descriptions (e.g. OWL-S or WSMO). This requires mapping (also referred to as grounding) of the abstract specifications to concrete service invocation protocols (e.g. SOAP, the protocol for Web Services).

8.4.2.3 Purpose “Integration”

The purpose “integration” comprises aspects of data integration and service integration.

Meta-information for data integration incorporates the description of data, its location, the mappings between different data representations, and data retrieval.

Meta-information for service integration is needed to support composition and interoperability of services. It comprises the description of the service interfaces and functionality.

As an example for an integration requirement, a simulation service based on a flood forecast model and a database containing meteorological data could be imagined. It should be possible to use the database as input for the simulation model and the model’s output as input for any other integrated service.

Service composition is the process of selecting, combining and executing of services in order to achieve a user objective; from the user point of view, the composition is a new service.

A composition is based on a choreography, which defines the rules to communicate with each service participating in the composition in order to consume its functionality. Compositions of services can be distinguished by the time at which the composition is determined: proactive composition (determined at the design phase) and reactive composition (built dynamically at the time the new service is requested). Meta-information is needed for both patterns.

Service interoperability means mutual usage of open service interfaces and protocols across institutional boundaries. However, internal details of the organisation of an institution should not be made publicly visible. Therefore meta-information is required in order to describe the external behaviour of services such that no information about internal business processes is exposed.

Service mediation resolves incompatibilities that arise when performing tasks concerned with the purpose of discovery, invocation or orchestration of services. For instance, in a discovery scenario, queries (formulated by the requestor) and capabilities of services (formulated by the service provider) may be incompatible because they use different terminologies. Incompatibilities can arise on the data level and/or the process level; at the data level, mediation between different terminologies requires solving the problem of ontology integration. At the process level, mediation between heterogeneous communication patterns is necessary in order to resolve possible mismatches, e.g. by generation of dummy acknowledgements.

8.4.2.4 Purpose “Interpretation”

The purpose interpretation is concerned with the support of explanation and understanding of resources (data and services).

In many cases resources can be interpreted only by investigation of vast amounts of implicitly expressed semantics. Thus, explicit descriptions of the semantics shall be added in order to make data and services self-explanatory and enforce their semantic integration.

A real world example is given by a user needing some information about contaminated sites and their classification according to risk categories. Although he has no access to the database containing all the measurements of toxic substances, in some cases he might have to explain the origin of the category number. Therefore he needs the specific measurement values along with the corresponding critical values that caused this classification.

8.4.2.5 Purpose “User profiling”

It is necessary to provide views on data and services and interaction procedures to support different types of users on a per-user or a per-task basis.

Users and tasks will be described in a way that appropriate views on data and services can be provided for different users and tasks.

The required meta-information relates to the way users are represented in an ORCHESTRA Application as subjects (see section 7.5.2). For example meta-information might be user information (user group, service provider, service/data integrator, administrator, etc.) and a particular language.

8.4.2.6 Purpose “OSN Configuration and Management”

Each OSN has to be monitored and administered.

Meta-information for configuration management of the OSN comprises descriptions of the topology of services of the entire OSN, e.g. which services are available at which sites.

Meta-information for the OSN monitoring comprises information on the actual load, service statistics as well as execution traces of services, which are important especially to document and trace execution of services which have been composed reactively.

In order to be able to fulfil this task, all of the services within the OSN have to provide at least their self-description as meta-information.

Means for monitoring, configuration and administration of the OSN have to be provided in order to facilitate this task.

8.4.2.7 Purpose “Authentication, Authorisation, and Accounting (AAA)”

The purpose “accounting, authentication, and authorisation (AAA)” is concerned with meta-information needed for controlling access to computer resources, enforcing policies, auditing usage, and potentially providing the information necessary to bill for services and/or information. Therefore, AAA requires a special set of meta-information that is directly related to the authorisation paradigm and is of little to no use for anything else. This special set of meta-information makes up the authorisation context.

An authorisation context is a set of information used by the Authorisation Service (see section 9.6.9) to determine the authorisation decision for a given request. The authorisation context can contain, for example, the requesting principal(s), name of the invoked operation, etc.

Authentication is a method for identifying the acting subject (representing users or software components in an ORCHESTRA Application) in an OSN. Authentication systems provide answers to the following questions:

- Who is the subject ?
- Is the subject really who he/she purports to be?

Actual mechanisms used for the authentication can be as simple (and insecure) as a plain-text password challenging system or as complicated as the Kerberos system. All authentication systems rely on at least one of these three factors:

- *Something you know*, such as a password or a personal identification number. This assumes that only the owner of the account knows the password or the personal identification number needed to access the account.
- *Something you have*, such as a smart card, a token, or one end of a quantum key generator. This assumes that only the owner of the account has the necessary smart card or token needed to unlock the account, or that he/she is the only person able to access this end of a quantum key generator.
- *Something you are*, such as fingerprint, voice, retina, or iris characteristics.

The ORCHESTRA Architecture does not impose any limitations on the number and type of authentication systems used within OSNs. Unless such limitations are imposed on the implementation level, every service provider in a typical OSN will be free to use its own authentication system.

Typical authentication-related meta-information includes a principal, which is used by the system for authorisation and accounting purposes and therefore should be uniquely assigned to a well-known subject, and some kind of information that is presumably available only to that subject that attempts to authenticate a principal (e.g. ”password“). Independent of the authentication system, at least one centralised or distributed database with user identifiers must exist. In other words at least one OSI of an Authentication Service shall exist in an OSN that is of type “access-controlled” or “secure” (see the discussion on OSN characteristics in section 11.1). Depending on the authentication system, this database will also contain shared secrets. Subjects must prove their authenticity by supplying the correct secret. Also, more sophisticated authentication-mechanisms (e.g. one-way hashes of a shared secret, actor’s public key, a list of single-use keys, etc.) taking the place of the “username-password mechanism" are imaginable.

In security-critical applications, authentication has to take place before granting access to the requested service operations. As a complex (and perhaps extreme) example, an organisation may wish to implement an authentication mechanism involving a retina- and a fingerprint-scan as a pre-requisite for using their PCs, use quantum key encryption over a quantum channel to secure transmission channels and to assure the end-point's identity, restrict access to specific hosts, and finally use some more classical means of authentication before actually granting access to specific service.

Authorisation protects resources by restricting usage of those resources to those principals that have been authorised to use them. The authorisation process is used to decide if that subject is allowed to make use of a specific resource. In order to identify those subjects the authorisation process makes use of the authentication process.

Apart from a static authorisation list the authorisation-decision might also be based on certain dynamic restrictions like time or date constraints, maximum number of concurrent accesses or location-based restrictions (e.g.: no rights granted to remote accessing actors). The types of permission (operation permissions, time-slice permissions) actually supported depend on the implementation of the Authorisation Service (see section 9.6.9).

Authorisation related meta-information may be as simple as a static authorisation list maintained on a central authorisation server, or as complex as a hierarchical set of dynamic rules involving position in an organisation, time or date constraints, maximum number of concurrent accesses or some other measure for service load, billing, or location based restrictions. Authorisation related meta-information is delivered via or referenced within the authorisation context.

The authorisation context is passed to the authorisation service by the service requesting the authorisation decision.

Note: Authentication and authorisation are critical factors for joining OSNs. Whenever two OSNs are joined, a compromise will have to be made concerning the allowed access levels for actors authenticated by the "other" OSN. In the case of the complex example described above, in-house security policy may completely prevent direct merging of "their" OSN with any other network.

Accounting is the process of gathering information about the usage of resources by subjects. This can, for example, include duration of usage or size of the retrieved resources. Accounting information can further be used to support billing, fair-use, planning and many other purposes. In that sense accounting information can be used by the authorisation process in order to provide a basis for the granting of usage rights. The requirements on the actual implementation define the necessary pieces of information and obviously the implemented logic inside the AAA-related and user management services.

Meta-information related to accounting is usually a combination of the principal identifying a subject (e.g. the login-name), and some measure for resources utilisation, such as "amount of data downloaded from the service", "time required to calculate the answer", "duration the resource was used during working hours", "tons of emitted CO_2", or "m^3 of water used for irrigation". Depending on the business model, accounting information may be connected to some kind of a group identifier ("organisation"), or even be completely anonymous.

Note: Due to a lack of user requirements on accounting, dedicated accounting services and meta-information models are currently out of scope of the planned versions of the RM-OA within the ORCHESTRA project.

8.4.2.8 Purpose "Quality control/management"

The purpose "quality control/management" is concerned with meta-information needed to enhance quality of information and services as well as to increase trust in information, data and services.

Quality control/management is needed when certain criteria need to be fulfilled by data and/or services. Quality usually has different aspects depending on whether services or data are considered. Specifically, quality control is important to every actor in every OSN and highly relevant whenever data and services have to meet certain legal requirements. Therefore working with data that have no quality information may be in some cases just as bad as working with randomly generated data.

Service quality in the ORCHESTRA sense has to deal with infrastructure properties. Examples of these are response time or availability of services. Another aspect that can be considered to be an attribute of service quality is the fee one has to pay to use the service. Quality regarding the output of services, whether it's back to the actor invoking the service, passed on to another service or stored in an internal data repository is considered to be the data quality of the service. This type of quality is important especially in the context of service chaining when accumulation of errors becomes an issue. A valid source of information for this can be found at (W3C 2003).

Data quality becomes an issue when working with data. Quality may refer to many different aspects and only an open list can be given to characterise them in the context of data:

- absolute and relative errors of measurement data
- computational errors of data processing services
- numerical issues
- minimum and maximum degree of detail in the values of a data set on a specific service
- sensitivity to error accumulation
- refresh period of the data (if it's not just a repository for old data)

Obviously the list of criteria for data quality can become quite long but this degree of detail is not always needed in order to classify the quality of data. The meta-information entries required depend on the particular requirements of the ORCHESTRA Application.

Quality management also means trust management. These are tightly coupled. Trust becomes an issue whenever authenticated and authorized but unknown (or not well-known) parties join a network. When providing their data and services to the network they can and must apply meta-information regarding the quality of what they are exposing. But how can an actor be sure if this meta-information really represents the quality of the actual data and services? The actor's only choice is to either trust or distrust the actor that attached the quality meta-information. Besides deciding whether to trust an actor or not, degrees of trust can also exist. Many different information items can be considered important for trust relationships, including

- Information about the actor: e.g.: name
- Certificates the actor has been granted
- The organisation that the actor represents

Note 1: In order to trust an actor, that actor must be identified first, so a trust relationship relies on the authentication process. Trust relationships are not mandatory but are highly recommended to ensure the quality of a network. A network that does not foresee trust management can be seen as a network where every actor is fully trusted by default.

Note 2: For a discussion on trust in a service environment, see also (OASIS 2006).

Examples for data/information-related quality meta-information: This depends on the data or information item itself. It is important that each of them has attached meta-information. For example a measurement value within an air quality monitoring network can have attached meta-information about its verification status (checked/unchecked) and validation status (valid/invalid).

Examples for service-related quality meta-information: The most important type of service-related quality meta-information is the one concerning guaranteed availability of service and guaranteed response times. For example, a single server has far lower guaranteed availability than a redundant server farm, and a huge grid may be able to guarantee answer times (with constant data quality) practically independent of load. Other important aspects of service-related quality meta-information include "guaranteed availability of the service for next N years", "versioning" (which implies availability of all data for long periods – possibly the whole service lifetime), and "transaction safety".

8.4.2.9 Purpose "Transactions, Synchronization and Locking"

The ORCHESTRA Architecture defines a set of services that are built with interoperability in mind. In order to use the ORCHESTRA Architecture to its full extent, different services need to be transparently combined into new “(virtual) compound services”. Using such service chains (combinations) to the full extent requires mechanisms and meta-information that support building transaction-secure composed operations on the OSN level. These mechanisms can be further separated into Transactions, Synchronisation, and Locking.

Transactions are needed when certain tasks that involve resources need to be carried out and it is important to ensure that the resources are not altered during this process.

Synchronisation is needed to secure that data/information are in a consistent state. That means inter-connected data have to be kept synchronous.

Therefore, updating distributed data without transactions is dangerous in two ways:

- First, distributed data will inevitably become “out of sync” during the update procedure. Accessing the data while they are still “out of sync”, can lead to unpredictable outcomes.
- Second, the update procedure may break during execution, leaving the data in an unsynchronised state. Consequently, application programmers have to invest a considerable amount of work in checking the data consistency and assuring that the update is eventually completed.

Neither of these problems occurs if all the changes are encapsulated within a single transaction.

A transaction is a logical group of operations that succeeds or fails as a group. This means that either all tasks within a transaction are carried out or none are. That way a transaction appears to be atomic. A lock is a mechanism to (temporarily) restrict the access rights to a resource for certain actors. Locking is used to guarantee the atomicity of transactions.

Note: Care must be taken when using a locking concept in order to avoid deadlocks.

Examples of meta-information related to Transactions, Synchronization and Locking include “start transaction”, “end transaction” and “abandon transaction” signals, and various exceptions signaling that a service is unable to perform a transaction (e.g. transaction unsafe services), had to abandon a transaction because part of it did not work out (e.g. one service in the chain isn’t transaction safe), or that a service is unable to respond to a request because it is currently busy with an transaction.

In addition, each transaction/synchronization request to a transaction safe service produces a lock that is unique with respect to at least this service and thus also unique with respect to OSN (because service has unique identifier with respect to OSN). In order to minimize problems with deadlocks, it may be advisable to assign an OSN-wide unique identifier to each transaction, maintain a globally accessible list of transactions and locks they are causing, and enforce an OSN-wide policy on maximal acceptable transaction times.

8.4.3 Framework Specification

The framework for ORCHESTRA Meta-Information Models (see Figure 16) is specified according to the general considerations for meta-information as described above. It distinguishes between

- an ORCHESTRA Meta-Model (also used for meta-information) on the meta-level,
- ORCHESTRA Application Schemas for Meta-information (OAS-MI) on the schema level, and
- Meta-Information Bases on the feature level.

The Meta-Information Base is a store for meta-information elements. The store might be persistent or transient, depending on the purpose of the meta-information usage. An example of a persistent store is a catalogue for discovery or navigational purposes. An example of a transient store is the usage of meta-information that is extracted on-the-fly in order to support mediation tasks. The Meta-Information Bases contain information that describes features in the form of an OFS according to a well-defined purpose (e.g. navigation, search). There may be several Meta-Information Bases in an OSN.

The structure of these Meta-Information Bases is defined in dedicated ORCHESTRA Application Schemas for Meta-Information (OAS-MI) as a special variant of OAS applied to meta-information. As the Meta-Information Bases are generated according to some purpose, there may be different OAS-MIs for different purposes. ORCHESTRA does not specify one conceptual schema for meta-information models for all tasks. Instead, the ORCHESTRA Meta-Information Model consists of the set of all OAS-

MIs that are defined according to the purposes identified above.

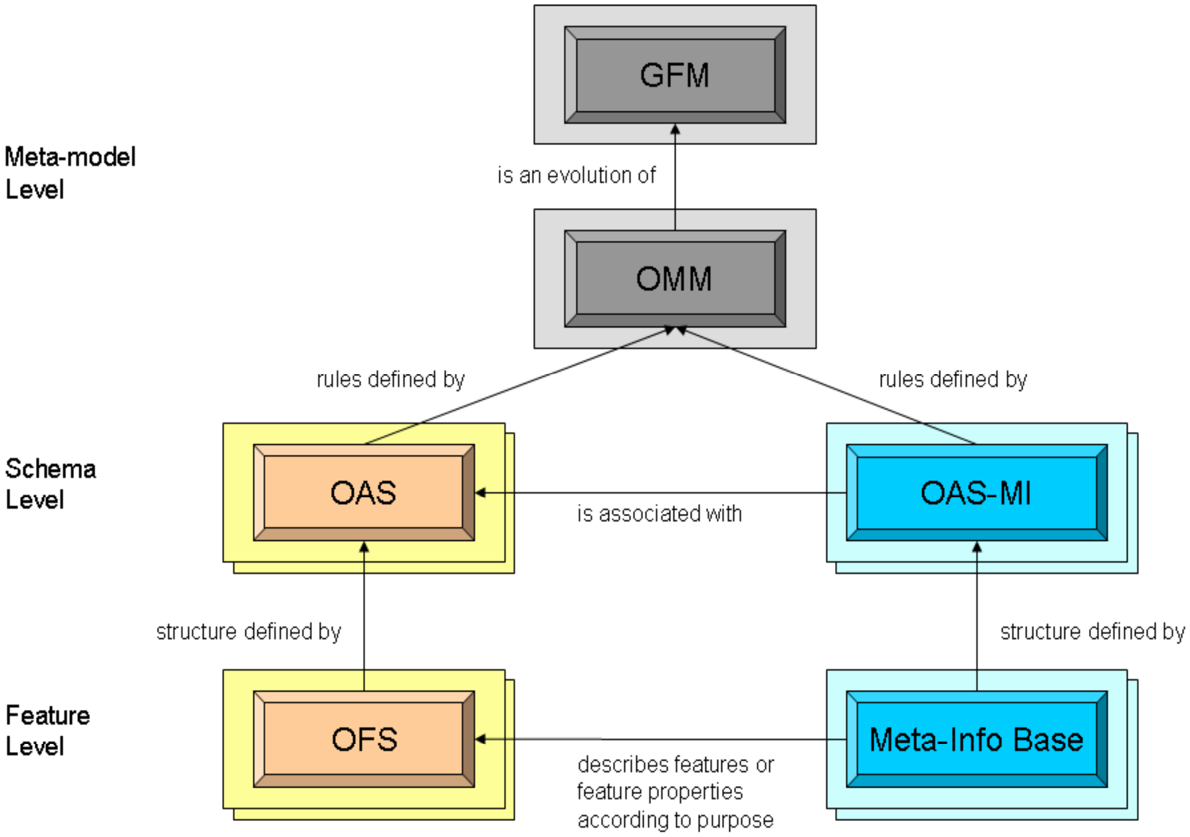


Figure 16: Framework for the ORCHESTRA Meta-Information Model

Depending on the purpose, an OAS-MI may be related to an OAS through some relationships between the two models, e.g. the OAS-MI elements may be attribute types of feature types or they may be feature types themselves that are associated with other feature types.

The meta-model for the OAS-MI is the OMM with dedicated statements on the role of attributes that are considered as meta-information for a particular purpose (see section 8.7.4). Thus, all rules for OAS also apply for OAS-MI.

Dedicated rules for the definition of OAS-MI are defined in Annex B1 of the RM-OA.

8.4.4 OMM Extensions for Meta-information Association Types

In order to allow one OMM_FeatureType instance to serve as meta-information for another OMM_FeatureType instance another subclass, OMM_MetaInfoAssociationType, is added to OMM_AssociationType (see Figure 17). This means that in an OAS, classes marked as feature types can be associated with each other using instances of the OMM_MetaInfoAssociationType.

Note 1: The list of subclasses is not complete in Figure 17 as new or refined classification schemes could be applied, e.g. different variants of aggregation.

Note 2: This approach covers meta-information for Features, Feature Collections and Feature Types as all three terms can be subsumed under the term feature.

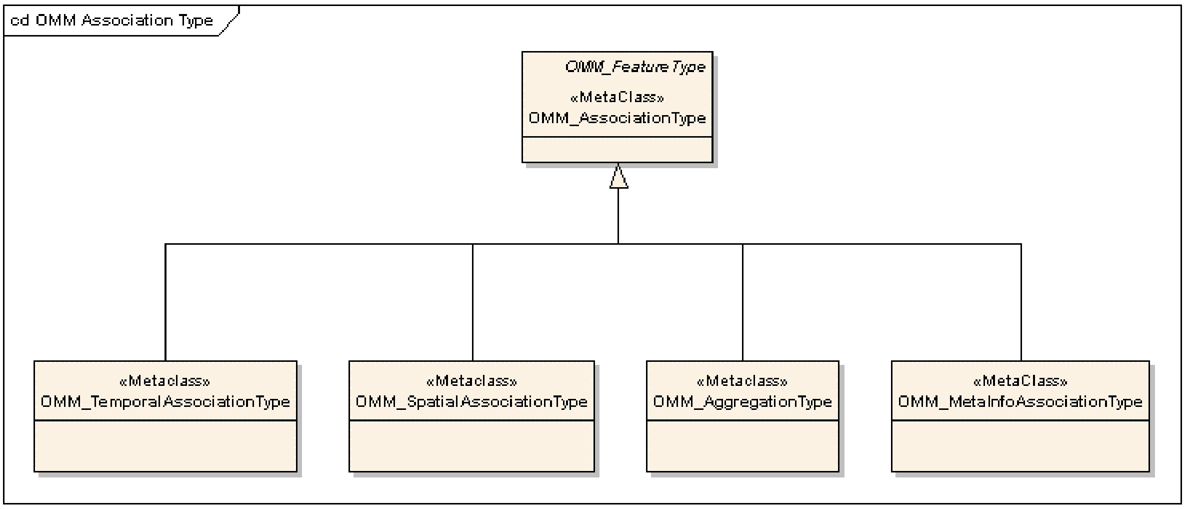


Figure 17: Subclasses of OMM_AssociationType

8.5 Inclusion of the Source System Level

8.5.1 Extension of the Information Model Framework

The RM-OA specifies a service-oriented architecture that is dedicated to the integration of systems providing both information and services (see section 5.4.3). For this purpose, ORCHESTRA offers means and services for syntactic and semantic interoperability. Thus, the RM-OA specifies an architecture for a “system of systems” or “networked systems”. These systems may already exist, whether implemented in older technologies (“legacy systems”) or in more recent technologies, or they may already be built based on ORCHESTRA services.

Regardless of their structure, their technology, their information or their services, these systems are called “source systems” in the sequel. They provide the source of information and services to be integrated into an OSN.

Source systems are of a very heterogeneous nature with respect to their structure and content. Examples of source systems are relational or object-oriented databases, information systems, document archives, map servers, Web sites and sensors. As a consequence, the interfaces to access the information contained in a source system or to call a service offered by a source system are very diverse. Although they are sometimes based on individual de-facto or de-jure standards (e.g. SQL, JDBC/ODBC, CORBA, RMI, Web Services, .NET), there is no standard interface for the integration of source systems as a whole.

Figure 18 illustrates the consequences for the information model framework when explicitly taking the source system level into account.

The majority of source systems do not comply with the ISO, OGC or ORCHESTRA understanding of a feature, nor is their information model specified according to the respective feature models. In order to allow ORCHESTRA services to process this information, data and information of the source systems have to be converted into an OFS according to an OAS. Whether the resulting OFS is persistently stored or just maintained in a transient manner depends on the implementation architecture and the task to be fulfilled. The only requirement on source systems is that (possibly through some software adapter) they may offer their data and/or functions in a way that complies with the OMM.

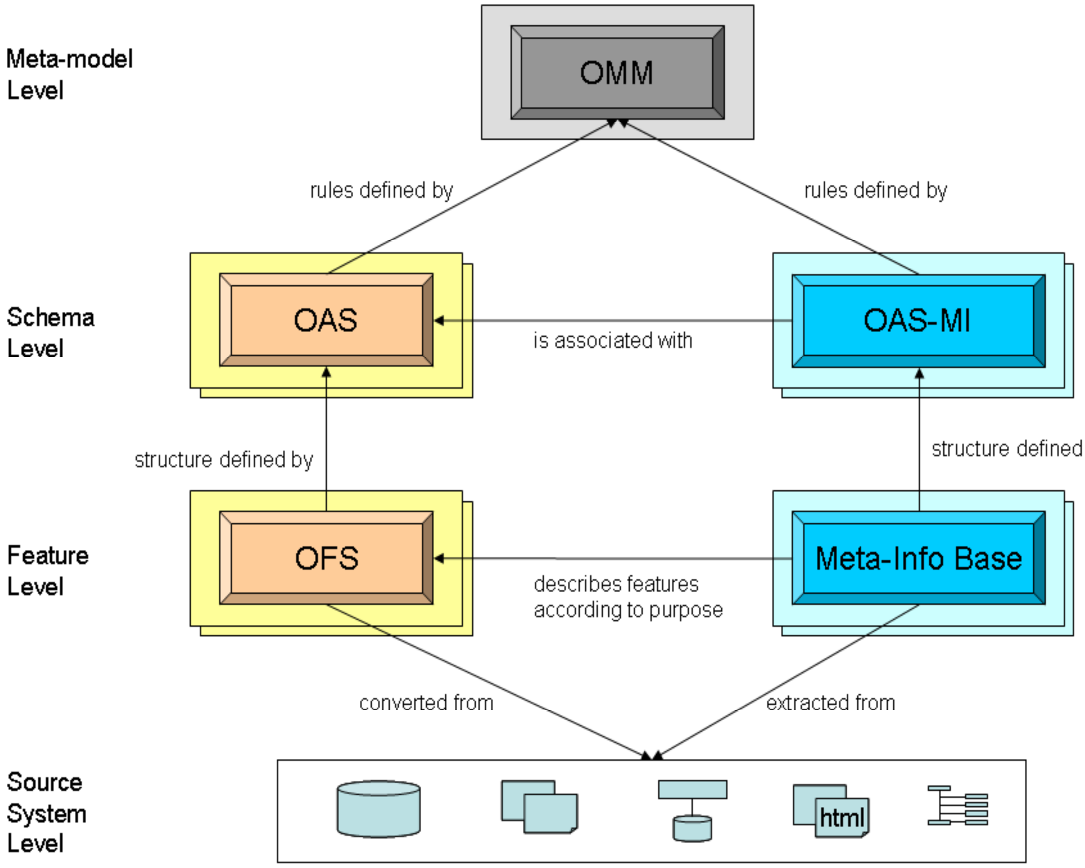


Figure 18: Inclusion of the Source System Level into the ORCHESTRA Information Model Framework

Furthermore, before ORCHESTRA services may access the information of the source systems, they have to be known in an OSN, either by means of an explicit registration step initiated by the source systems or by means of a discovery process initiated by OSN components. For this purpose, meta-information about the source systems, their information and/or their services is required.

This meta-information has to be extracted from the source systems, either by an explicit delivery process initiated by the source systems or their providers, or automatically by some extraction (annotation) process of meta-information initiated by a software component in an OSN. In any case, the extraction of meta-information is guided by the respective OAS-MI specifically designed for this particular purpose.

Note: The process for converting source system information into an OFS and the process for extracting meta-information about source systems for a particular purpose are independent processes. They may be performed in an isolated manner (e.g. just discovery based on provided meta-information), subsequently (e.g. firstly discover the source system using the meta-information provided, and secondly access to the source system information via the OFS) or in parallel (e.g. offline transformation of a source system into an OMM-compliant information system).

8.5.2 Scenario for Data Interchange related to ISO 19109

ISO 19109 specifies two patterns for the interchange of information between systems to be supported:

- Data interchange by transfer: this is the more traditional model where only the data along with the application schema describing its structure are exchanged between the two partners;
- Data interchange by transaction: in this usage pattern, the communication protocol for querying or modifying data is also specified allowing systems to communicate directly.

For the ORCHESTRA Architecture, being a service-oriented architecture, the data-interchange-by-transaction pattern will be used.

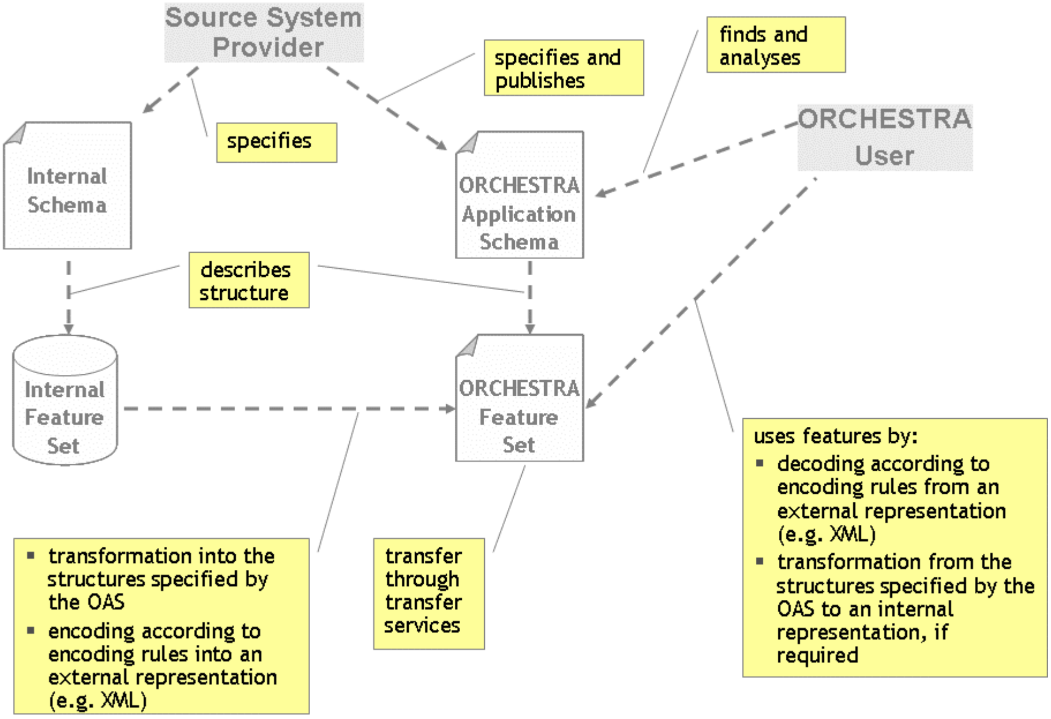

Source System Provider
specifies and publishes
finds and analyses
ORCHESTRA User
Internal Schema
specifies
ORCHESTRA Application Schema
describes structure
Internal Feature Set
ORCHESTRA Feature Set
uses features by:
decoding according to encoding rules from an external representation (e.g. XML)
transformation from the structures specified by the OAS to an internal representation, if required
transformation into the structures specified by the OAS
encoding according to encoding rules into an external representation (e.g. XML)
transfer through transfer services

Figure 19: Ad-hoc use of published feature sets and application schemas

The descriptions in ISO 19109 can be read in a way that data interchange according to that International Standard requires agreement of all parties involved in the interchange over the application schema. Within the ORCHESTRA Architecture a typical usage scenario will be that a source system provider will publish its data (OFS) and the application schema describing it (OAS) without consulting most potential users of the data. If a potential user then discovers the OFS/OAS through catalogues, carries out an assessment of the usability of the feature set for his task and decides to use the data, this is then considered as an agreement (ex-post) over the application schema to be used in the data interchange, too.

This scenario is illustrated in Figure 19

8.6 Inclusion of the Semantic Level

8.6.1 Ontologies

The semantic level provides semantics to the information specified in the other levels, e.g. through explicit consideration of ontologies defined and shared in user communities.

An ontology is an explicit, formal specification of a shared conceptualisation (Studer et al 1998). Ontologies may be thought of as a formal representation of the knowledge associated with a particular subject area (domain) or task. Their ultimate purpose is to enable machine understanding, which in turn provides the potential for data and service interoperability.

8.6.1.1 Ontology Classes

Ontologies may be broadly classified as listed in Table 3 (ORCH-D2.3.5 2006). Domain and task ontologies capture knowledge at a level of abstraction free from implementation concerns – that is, they reflect the pure nature of the domain or task. The application and data ontologies are descriptions of information system implementations, and are only necessary if domain and task ontologies cannot be mapped directly to these implementations. Domain ontologies are intended to provide a source of predefined concepts for use with task ontologies. Task ontologies will typically cross domains and therefore draw concepts from more than one domain ontology.

Ontology Class	Definition
Domain Ontology	A formalisation of the knowledge in a subject area (domain) such as topography, ecology, biology, flooding, etc.
Task Ontology	A formalisation of the knowledge necessary to solve a specific problem or task but abstracted above the level of a specific situation or organisational context, for example performing the task of monitoring fresh water quality.
Application Ontology	Contains knowledge for a specific application designed to complete a task in a specific situation and organisational setting, such as the task of monitoring water quality as performed by the Environment Agency. Such ontologies will contain little knowledge that is directly reusable by other organisations and serve to provide a semantic interface between the domain and task ontologies and the application.
Data or Service Ontology	Describes a service or data source and may be seen as a special type of an application ontology.

Table 3: Ontology Classes (ORCH-D2.3.5 2006)

Within the RM-OA, ontologies of these classes may be taken into account as follows:

- Domain Ontologies may be used in order to provide a semantic reference for ORCHESTRA Information Models and ORCHESTRA Meta-Information Models.
- Task Ontologies may be used in the context of service chaining and workflow modelling and will be considered as part of the RM-OA Service Viewpoint specification.
- Application and Data Ontologies may be used to support the integration of source systems. Here, available application or data ontologies are meta-information for the source systems. Thus, they will be considered as part of the RM-OA Information Viewpoint in the context of
  - the schema mapping between internal schemas of source systems and respective OAS, or
  - the process of converting data from source systems into OFS according to an OAS, or
  - the process of extracting meta-information from source systems.
- Service Ontologies may also be used to support the integration of source systems with a particular focus on the discovery and mediated access to services provided by source systems. Here, service ontologies are meta-information for the services of source systems. Thus, they will be considered as part of the RM-OA Information Viewpoint in the context of the process of extracting meta-information from source systems. Their usage for the service mediation will be specified as part of the RM-OA Service Viewpoint.

Note 1: The RM-OA will start with the consideration of domain ontologies. Domain ontologies are the most advanced ones in the research community of the Semantic Web. Furthermore, they play a major role within the ORCHESTRA project (ORCH-D2.3.5 2006).

Note 2: The current version of the RM-OA has its focus on the support of syntactic interoperability. Thus, this RM-OA version just positions domain ontologies in the framework for ORCHESTRA Information Models. Future versions of the RM-OA will provide more detailed specifications of how ontologies influence the RM-OA Information and Service Viewpoints.

8.6.1.2 Conceptual and Logical Ontologies

Ontologies are formal representations of the knowledge associated with a particular subject area (domain) or task, whose ultimate purpose is to enable machine understanding of the knowledge in a particular domain (ORCH-D2.3.5 2006). Within the RM-OA, ontologies are considered in two appearances according to the following two development stages of ontologies:

- The first stage is the construction of a conceptual ontology by the domain expert. A conceptual ontology is structured knowledge in a domain which a domain expert can understand. Its documentation includes the following:
  - A glossary of concepts, instances, relationships, their natural language definitions, assigned characteristics and values, and additional information assigned to the relationships.
  - Sources of the documents used to create the content of the glossary.
  - Defined rules, assumptions and primitives used to express the definitions.
  - Concept networks and hierarchies (either in a diagrammatic format or in linear notation).
  - Relationship networks and hierarchies (either in a diagrammatic format or in linear notation).
  - Defined rules and assumptions regarding the networks or hierarchies.
- The second stage is the transformation of the structured knowledge base into a machine-readable logical ontology by an ontology expert. The resulting logical ontology is thus defined in a machine-readable notation like e.g. OWL.

8.6.1.3 High-level Ontologies

A high-level ontology could be expected to contain terms of a more abstract nature or coarser level of granularity that can be related (through subsumption relationships) to those concepts in other domain ontologies which capture knowledge at a finer level of granularity (ORCH-D2.3.5 2006). For example in the thematic context of risk management, a “flood risk” domain ontology may include concepts like “flood risk map”, “risk of flood”, and “velocity measurements”, and may need to use their super-ordinate, more generic terms, to effectively describe these concepts. The super-ordinate generic concepts are, however, often out of scope. A high-level ontology serves the purpose of containing these generic terms which are common across several domains. A high-level ontology, which the “flood risk” ontology could reuse, would contain concepts such as “map”, “risk”, and “river data”.

Due to the generic nature of the RM-OA, those generic concepts of high-level ontologies that are not tied to a particular thematic domain have the highest relevance to be considered as basic information elements in the framework of ORCHESTRA information models (see section 8.4).

8.6.2 Extension of the Information Model Framework for Domain Ontologies

The extension of the information model framework after domain ontologies have been taken into account is illustrated in Figure 20.

As mentioned above, the RM-OA distinguishes between conceptual and logical ontologies. This is reflected in the framework on the semantic level whereby the logical ontology is the result of a transformation process from the conceptual ontology.

As the RM-OA specifies a generic ORCHESTRA Architecture, the information viewpoint is not tied to a specific domain ontology either on the conceptual or on the logical level.

Note: The handling of the conceptual model and the transformation process to the logical ontology is out of scope of the RM-OA. The RM-OA Version 4 will cover the aspects of semantic interoperability based on machine-processable logical ontologies.

Examples of relationships to the other levels of the specification framework are illustrated in Figure 20:

ex 1. Generic concepts that are relevant across a multitude of domain ontologies (possibly collected in form of a high-level ontology) are candidates for the specification of additional meta-classes in the OMM. Examples here are documents or maps.

ex 2. An OAS-MI provides an application schema for meta-information for a particular purpose. Usually, the classes and their characteristics in the form of attributes and operations used in the application schema have no formally defined semantics. In order to support mediation tasks using the meta-information, the concepts in a domain ontology including their natural language definition (i.e. the glossary) could be referred to by the classes in the OAS-MI.

ex 3. OAS may be generated from logical ontologies if these have a sufficient level of detail, e.g. if they include typed slot definitions that may be mapped to feature properties types.

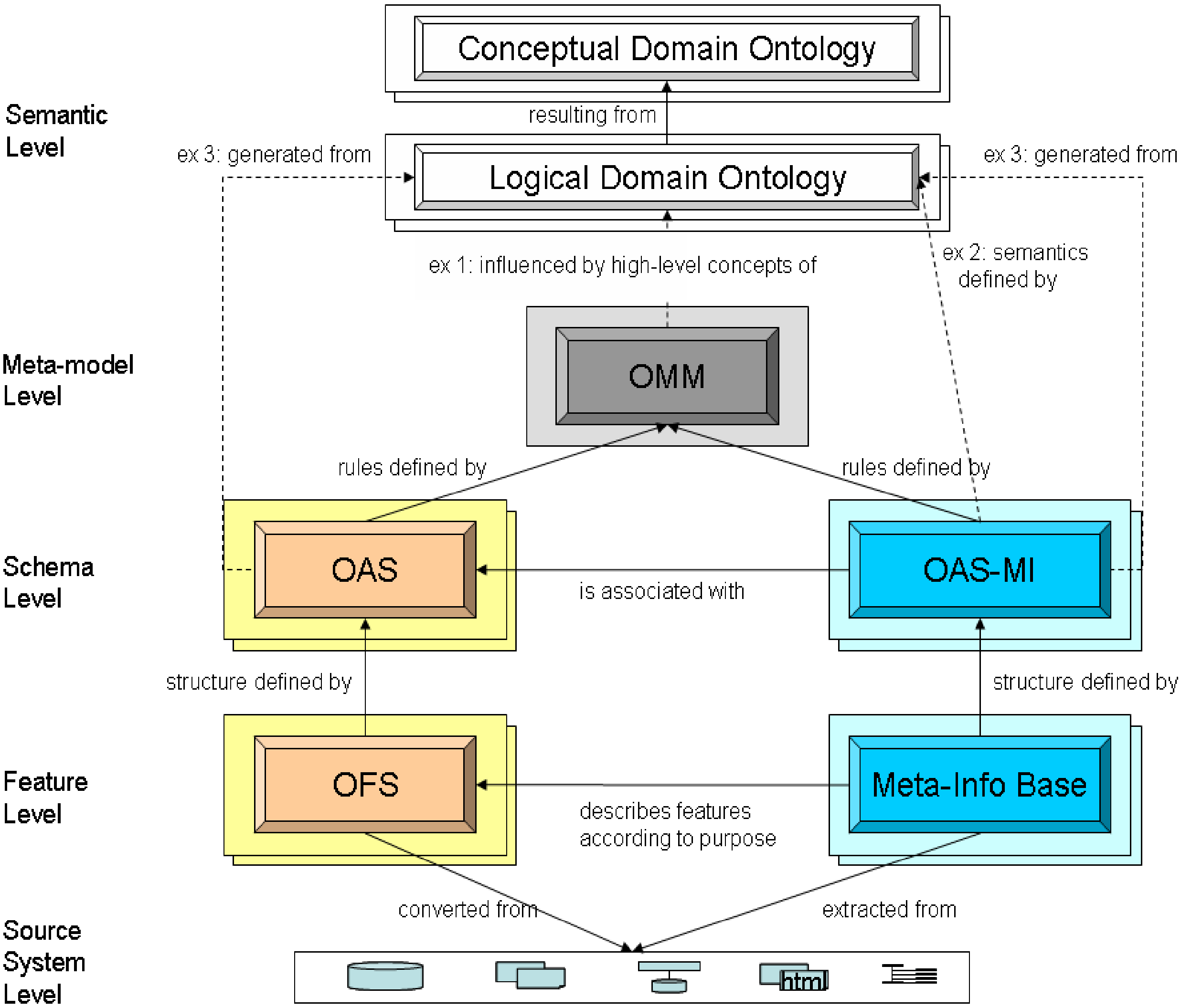


Figure 20: Inclusion of the Semantic Level into the Information Model Framework

8.7 The ORCHESTRA Meta-Model for Information

8.7.1 Overview

As mentioned above, the OMM is derived from the basic ideas of the ISO 19109 GFM, but it is not a true profile of it. In particular, the GFM requires that

- all data quality attribute types are implemented using DQ_Element as specified by ISO 19115,
- all "GFM metadata" attribute types are implemented using "metadata classes" as specified by ISO 19115, and
- a "GFM metadata element" has to be used as a GF_Metadata_AttributeType to carry "metadata" about instances of feature types.

Note: The term "metadata" here refers to its meaning and usage in ISO 19109 and ISO 19115.

While this may be true in a particular OAS, an OAS is *not* required to adhere to these rules. For instance, ORCHESTRA application schemas for meta-information will have to support other standards and other information models. See section 8.4 for additional details.

This is why the OMM is an evolution of the ISO 19109 GFM, taking into account additional, ORCHESTRA-specific requirements. After defining the data types to be used in the OMM and ORCHESTRA application schemas in section 8.7.2, the OMM is specified in two steps:

- the OMM selects the classes and properties of the GFM that are relevant for ORCHESTRA (see sections 8.7.3 and 8.7.4)
- the OMM adds additional meta-classes, namely for additional meta feature and attribute types (see sections 8.7.4 and 8.7.5). Note that the creation of these meta-classes is not strictly required, but shall clearly highlight and list the important information types required by ORCHESTRA applications.

8.7.2 Data Types

8.7.2.1 Introduction

The following section defines the most fundamental data types available in the ORCHESTRA framework. In order to achieve interoperability a common basis is made available and well-defined. ORCHESTRA Basic Data Types (and OA_Types) are part of such a basis.

All data types used and defined in ORCHESTRA shall be built directly and/or indirectly (e.g. OA_Types) using Basic Data Types. This enables ORCHESTRA users to have only one definition for a single type instead of a multitude of definitions (e.g. every service developer and/or every application designer defining its own types for equal purposes). ORCHESTRA basic data types relate and refer to definitions in already accepted standards (like ISO 191xx series) and therefore they are well-known in the software development community.

8.7.2.2 Basic Data Types

Basic Data Types have a standardised definition outside of ORCHESTRA documents (e.g. ISO 191xx series). The names of these types will not be prefixed and refer to standard types. They are defined in Table 4 with the related standard document being referred to in the *Origin* column.

Note: Basic Data Types must not be confused with the UML stereotype called <<DataType>> (see section 8.8.6).

8.7.2.3 OA_Types

OA_Types are predefined types in the OMM which do not have a standardised definition outside of ORCHESTRA documents. They are composed of ORCHESTRA Basic Data Types and other already defined OA_Types. OA_Types might still be rather simple.

8.7.2.4 User-defined types

User-defined types are not predefined within the OMM. They usually refer to types defined for a specific application (e.g. in an OAS) and may only consist of well-known types. These well-known types are Basic Data Types, OA_Types and already specified User-defined types.

Type Names	Origin	Brief Description
Real	ISO19103 section 6.5.2.5	A signed real (floating point) number consisting of a mantissa and an exponent. (not necessarily the exact value as the common implementation of a Real type uses base 2)
Integer	ISO19103 section 6.5.2.3	A signed integer number. Exact with no fractional part.
Decimal	ISO19103 section 6.5.2.4	A number type that represents an exact value as a finite representation of a decimal number. (Unlike real, it can represent 1/10 without error)
Binary	ISO19118 section A.5.2.1.14	Finite-sequence of arbitrary binary data.
Any	ISO19103	The root of all classes. Often not an actual class in the implementation, it essentially is used where the target class of a member name is not known.
CharacterString	ISO19103 section 6.5.2.7	Type representing a simple string. The whole string has a single specific encoding. This encoding is retrievable from the string.
CountryCode	As will be defined in ISO 19139	List of country identifiers.
LanguageCode	As will be defined in ISO 19139	List of language identifiers.
CharacterSetCode	ISO19103 section 6.5.2.7	List of character encodings.
MD_Character SetCode	As defined in ISO 19115	List of character encodings.
PT_Locale	As will be defined in ISO 19139	Type combining language, country and encoding.
Localised CharacterString	As will be defined in ISO 19139	A CharacterString with the addition of a field specifying the language of the string.
Enumeration	ISO19103 section 6.5.4.2	Defined and closed list of valid mnemonic identifiers.
CodeList	ISO19103 section 6.5.4.3	An open Enumeration.
Boolean	ISO19103 section 6.5.2.11	A value specifying TRUE or FALSE
Date	ISO19103 section 6.5.2.8	Type representing a date.
Time	ISO19103 section 6.5.2.9	Type representing a point in time.
DateTime	ISO19103 section 6.5.2.10	Type combining date and time.
Set	ISO19103 section 6.5.3.2	Unordered finite collection of non duplicate objects.
Bag	ISO19103 section 6.5.3.3	Unordered finite collection of possibly duplicate objects.
Sequence	ISO19103 section 6.5.3.4	Ordered 'bag-like' structure.
Dictionary	ISO19103 section 6.5.3.5	Container for key-value pairs where the key and value types are not predefined.

Table 4: Basic Data Types

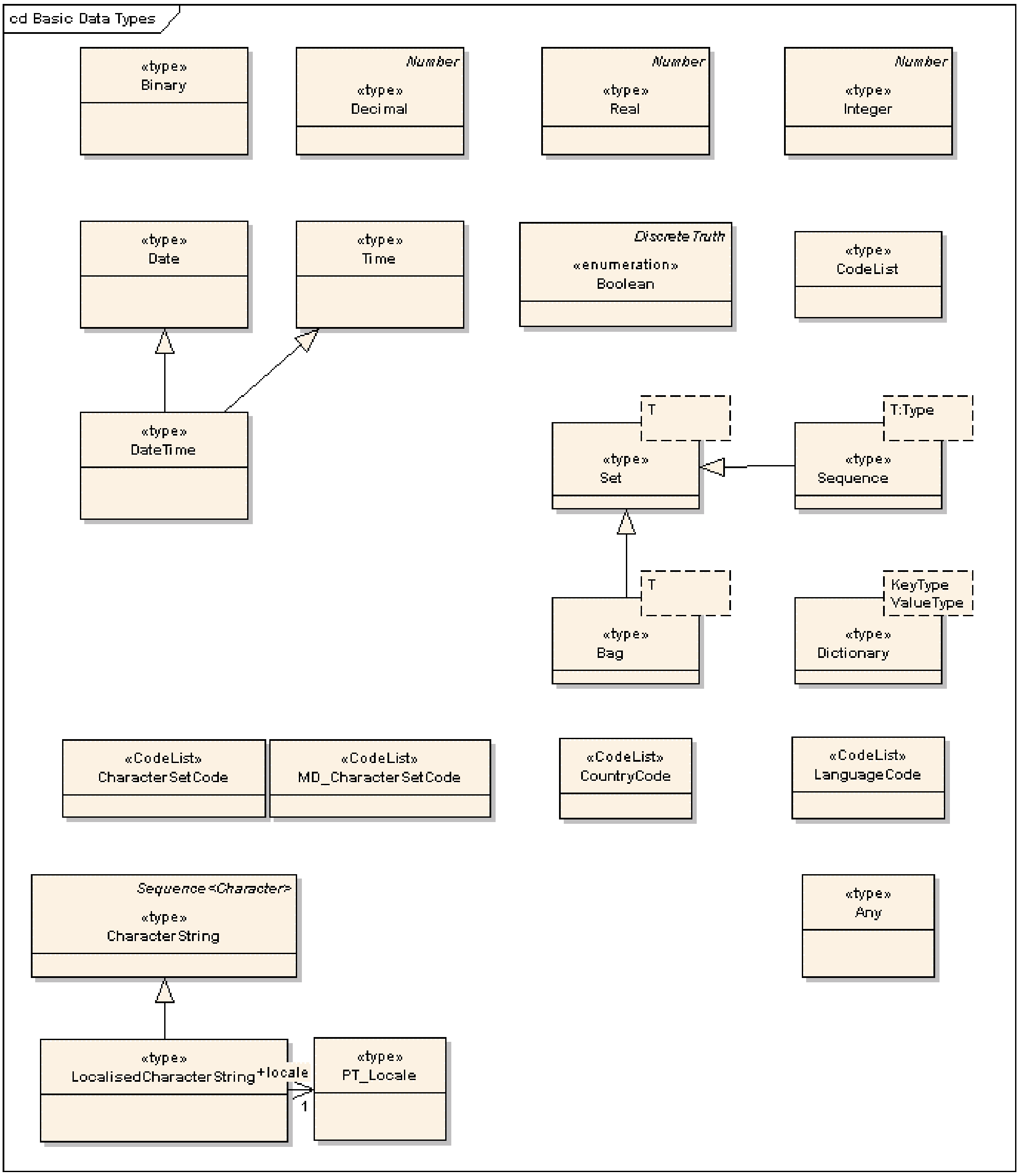


Figure 21: Basic Data Types

8.7.3 OMM Basic Part

The UML class diagrams in Figure 22 show the basic part of the OMM that principally specifies the relationship between OMM_FeatureTypes, OMM_PropertyTypes and OMM_AssociationTypes. It exactly corresponds to the main structure of the GFM as described in the section 7.3.3 (GFM main structure), section 7.3.4 (GF_FeatureType) and section 7.3.5 (GF_PropertyType) and illustrated in Figure 5 of the ISO 19109 GFM document.

The meaning of the respective meta-classes prefixed by OMM_ is the same as the meaning of the meta-classes prefixed by GF_ in ISO 19109 GFM.

The extension of the OMM with respect to the GFM relates to the extended understanding of what a feature type could be in ORCHESTRA as described section 8.2.

Note: Following the architectural principles of “self-describing components” (see section 6.3.7), a future RM-OA version might extend the OMM basic part in order to mandate that a feature instance contains (at least a reference to) the feature type specification, probably as part of its meta-information.

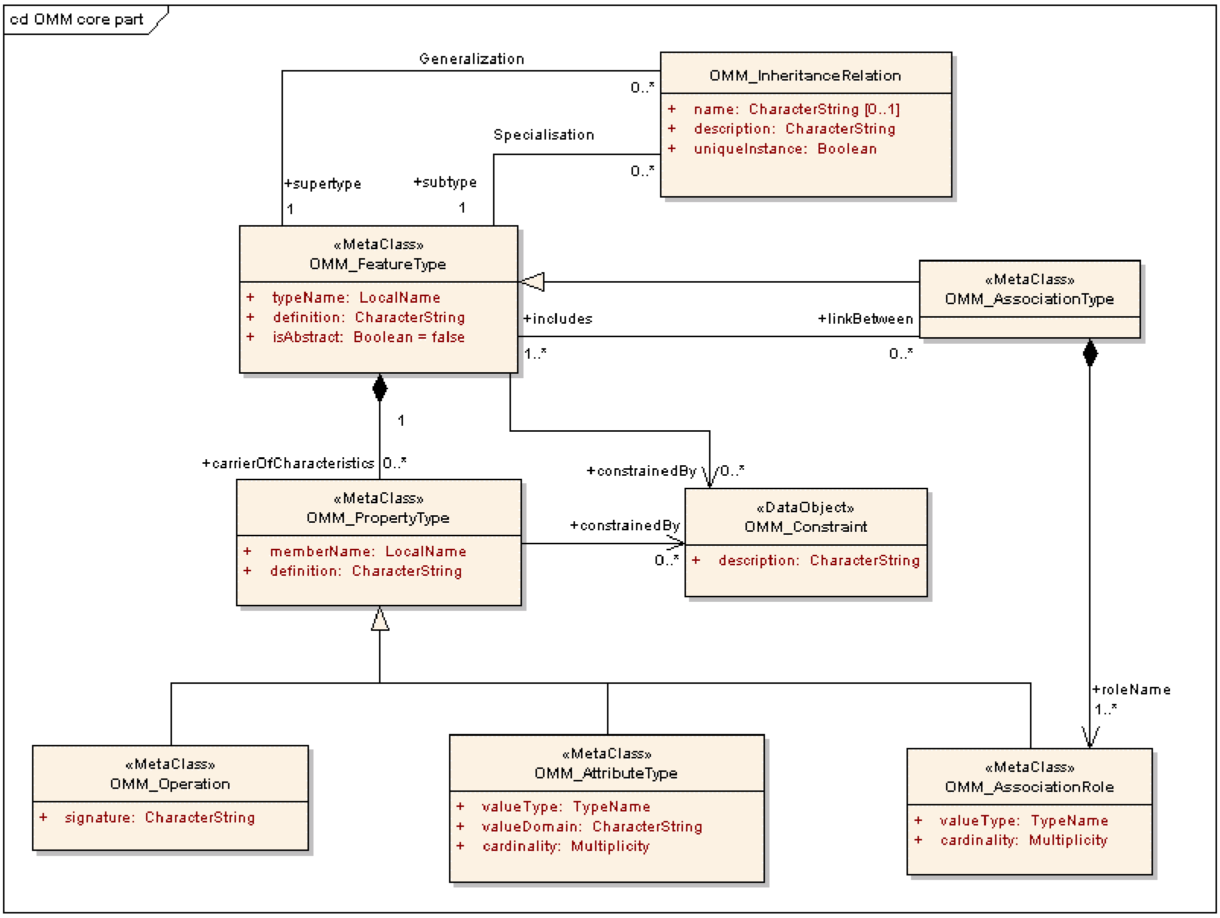


Figure 22: The basic part of the ORCHESTA Meta-model

8.7.4 OMM Attribute Types

The ORCHESTRA Architecture uses the following categories of attribute types and their base class from the ISO 19100 series:

- Spatial Geometry (ISO19107::GM_Object)
- Spatial Topology (ISO19107::TP_Object)
- Temporal Object (ISO19108::TM_Object)
- Geographic Identifier (ISO19112::SI_LocationInstance)
- Data Quality Information (ISO19115::DQ_Element) (see note 1 below)
- Metadata (ISO19115::MD_Metadata) (see note 2 below)

Note 1: The modelling of data quality information or meta-information in the form of attribute types as further specified in ISO 19115 is just one possibility for a meta-information model and the specification of meta-information in the context of an OAS. ORCHESTRA does support further types of meta-information models depending on the particular purpose of the usage of the meta-information (see section 8.4.1).

Note 2: The OMM does not specify meta-information attributes as a prominent high-level attribute type category. Instead, the modelling of meta-information attribute types (OMM_MetaInfoAttributeTypes) as a meta-class that specialises the meta-class OMM_ThematicAttributeType means that a thematic attribute may use type definitions of ISO 19115 as data type values. See also Rule 1 in section 8.8.11

The resulting schema is illustrated in UML in Figure 23.

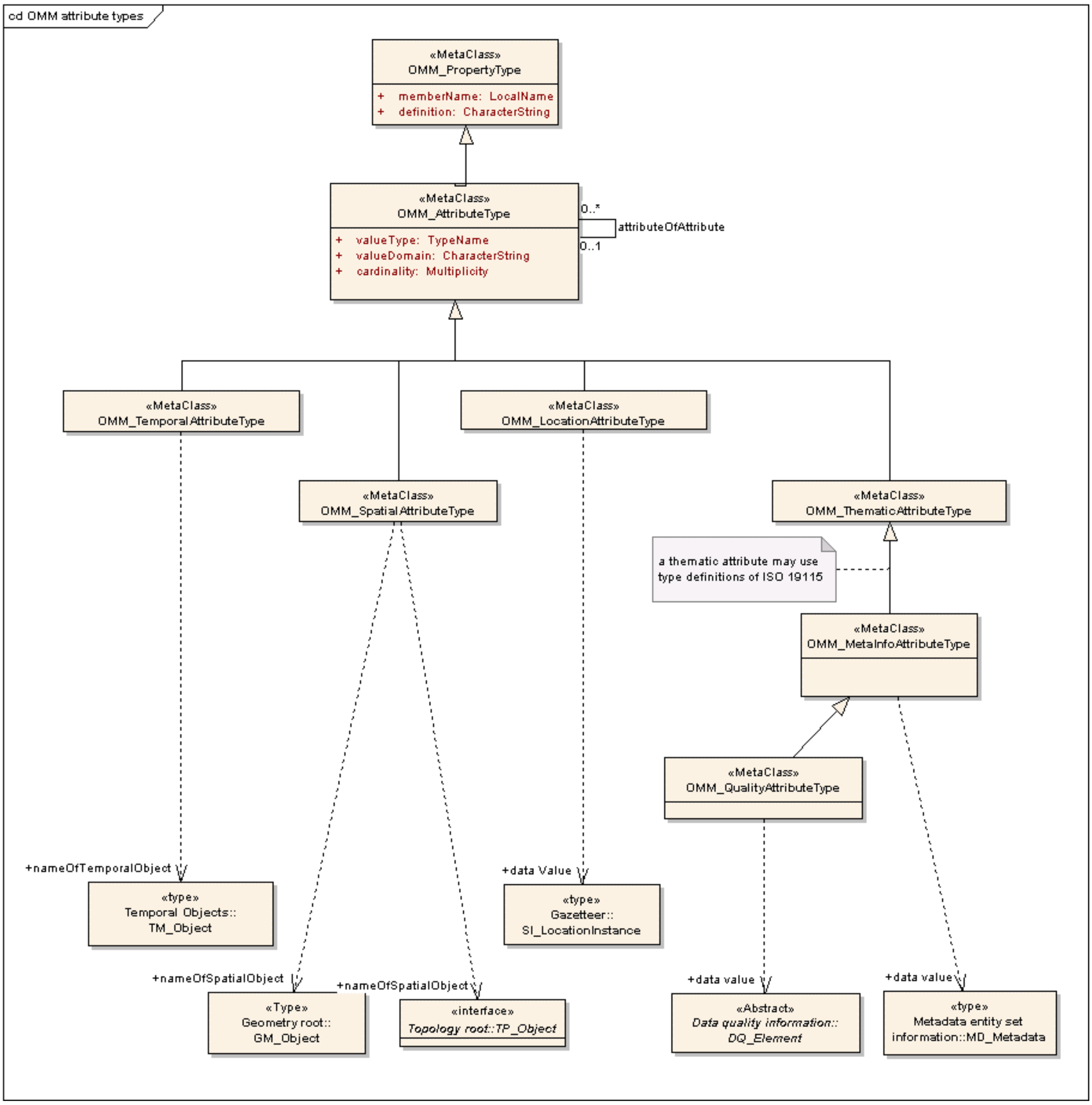


Figure 23: OMM Attribute types

8.7.5 OMM Extensions to Feature Types

8.7.5.1 Overview

As will be defined in the rules below (see section 8.8), an ORCHESTRA Feature Type is defined by a UML class that is part of an OAS as an instance of the OMM meta-class “feature type”. Within an OAS, it has a stereotype “FeatureType”.

Feature types are defined by an information modeller or, in some specific cases, on-the-fly by a software component of an ORCHESTRA Application as part of an OAS and represent "abstractions of real-world phenomena perceived in the context of an ORCHESTRA Application".

Based on the requirements of thematic domains, the OMM extends the OMM_FeatureType definition for additional categories of information types. As a result of an analysis of the requirements of the risk management thematic domain that took place in the ORCHESTRA project, the following eminent but generic information types have been identified:

- Document type (see section 8.7.5.2)
- Schema Descriptor type (see section 8.7.5.3)
- Coverage type (see section 8.7.5.4)

The following list comprises further candidates for OMM information types. Their specification needs further investigation:

- equation/formulae
- model
- observation and measurement (see (OGC 2006))
- dictionary and code list
- action
- meeting/conference/telephone call
- software

Note: These information types are identified out of the user requirements described in the respective ORCHESTRA SP2 deliverables (ORCH-D2.1 2006, ORCH-D2.4.1 2005). Further information types may be added if other user requirements are taken into account.

8.7.5.2 Document Type

Documents are resources that contain recorded information and can be treated as unit. As an ORCHESTRA feature type, a document is represented by a document descriptor that contains identification information (such as name and document type) and a reference to one of more files (the document store) if the document data is stored locally or a reference to a source system if the document data is stored remotely.

An instance of OA_ThematicAttributeType may represent an attribute that carries document information. The value-types of document attributes shall comply with the definition of an OA_DocumentDescriptor as defined below.

Document types may be classified according to the MIME Media Types and include e.g.

- Documents with page layout (e.g. PDF, MS-Word, MS-PowerPoint files, Web pages based on html)
- Audio files
- Video files
- Image files
- XML documents
- tabular data in file format (e.g. an MS-Excel file)

The document schema used in ORCHESTRA is specified in Figure 24.

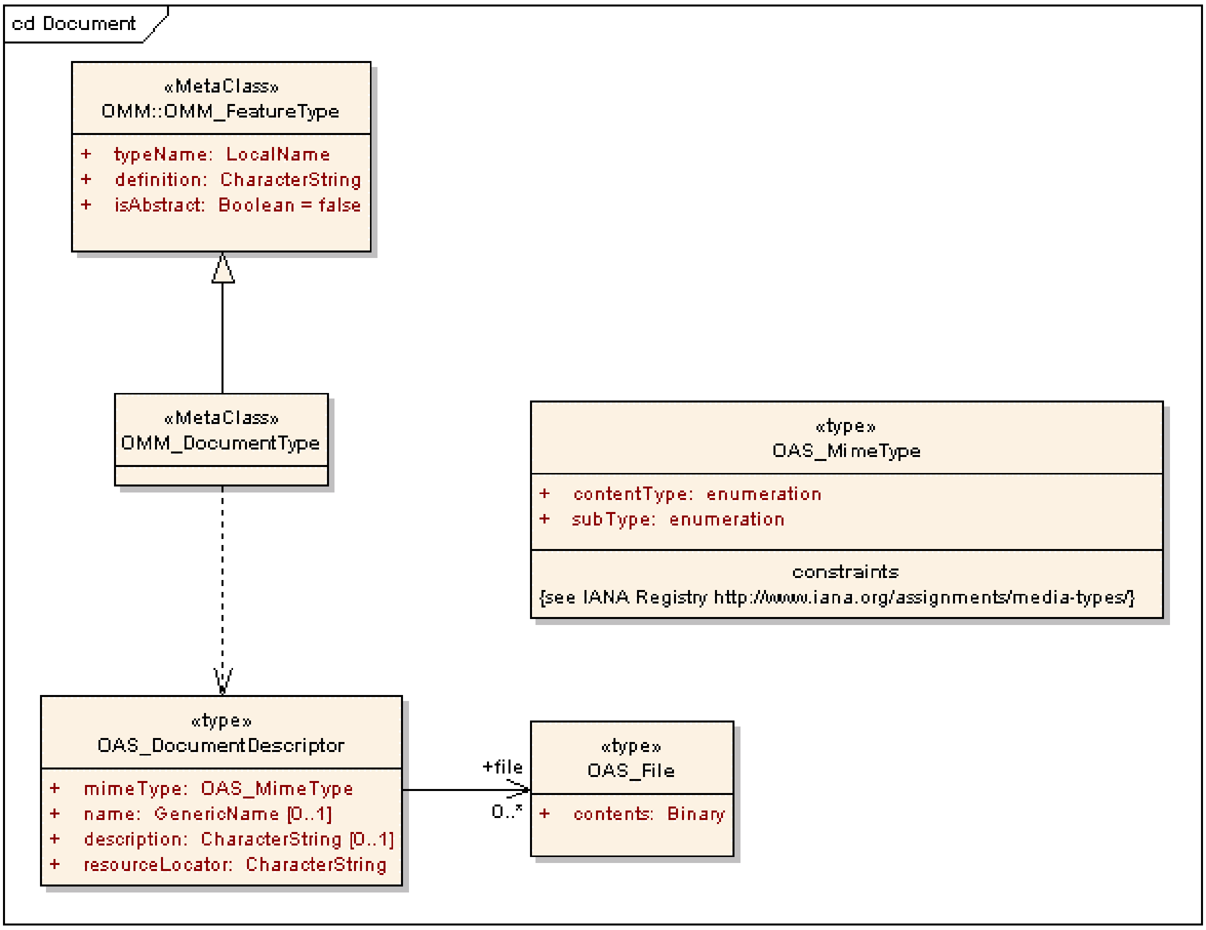

cd Document
«MetaClass»
OMM::OMM_FeatureType
+ typeName: LocalName
+ definition: CharacterString
+ isAbstract: Boolean = false
«MetaClass»
OMM_DocumentType
«type»
OAS_MimeType
+ contentType: enumeration
+ subType: enumeration
constraints
{see IANA Registry http://www.iana.org/assignments/media-types/}
«type»
OAS_DocumentDescriptor
+ mimeType: OAS_MimeType
+ name: GenericName [0..1]
+ description: CharacterString [0..1]
+ resourceLocator: CharacterString
+file
0..*
«type»
OAS_File
+ contents: Binary

Figure 24: Schema of the OMM extension "Document Type"

8.7.5.3 Schema Descriptor Type

A schema is a formal description of a model. Examples are the database schema of a relational data base, an application schema specified in UML or XML, or the table structure of an MS-Excel spreadsheet.

As ORCHESTRA feature type, a schema is represented by a schema descriptor that possesses identification information (such as name, purpose of the schema, encoding and a schema reference). The schema of the OMM "schema type" is specified in Figure 25.

Examples are:

- a schema of a relational data base ("GW", "Groundwater Database Baden-Württemberg", "ORACLE DDL", "SQL")
- a spreadsheet ("EX", "Earthquake Occurrences Naples 2004", "csv", "MS-Excel")
- an XML schema document based on XML Schema Definition (XSD).

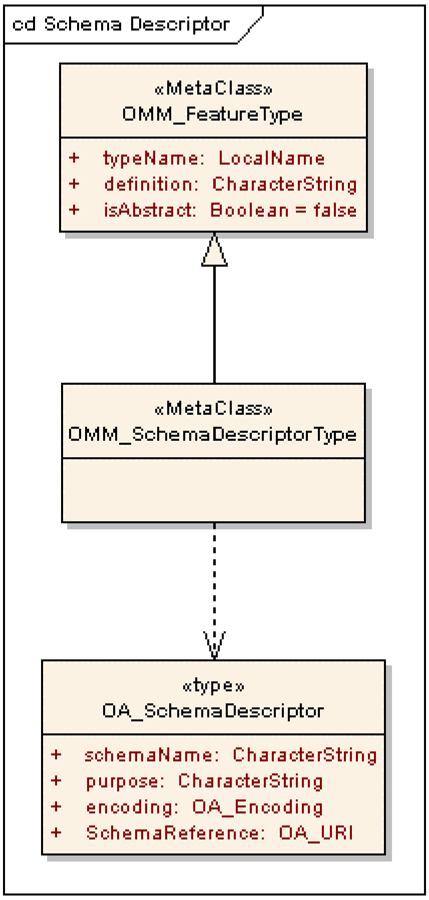


Figure 25: Schema of the OMM extension "Schema Descriptor Type"

8.7.5.4 Coverage Type

A coverage denotes a function from a spatial, temporal or spatiotemporal domain to an attribute range. A coverage associates a position within its domain to a record of values of defined data types. Thus, a coverage is a feature with multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type. Examples include a raster image, polygon overlay, or digital elevation matrix.

The coverage model is defined by ISO 19123.

The domain of a coverage is a set of geometric objects described in terms of direct positions, which are associated with a spatial or temporal coordinate reference system. Commonly used domains include point sets, grids, collections of closed rectangles, and other collections of geometric objects. The range of a coverage is a finite or a transfinite set of feature attribute values.

Coverages can be discrete or continuous. A discrete coverage has a domain that consists of a finite collection of geometric objects and the direct positions contained in those geometric objects. A discrete coverage maps each geometric object to a single record of feature attribute values. A continuous coverage has a domain that consists of a set of direct positions in a coordinate space, which it maps to value records. It then returns a distinct record of feature attribute values for any direct position within its domain.

Note: The term coverage may be misleading as it implicitly refers to a 2-dimensional data layer. The term field would be better as it refers to n-dimensional data. However, the term coverage is used in order to conform with ISO 19123.

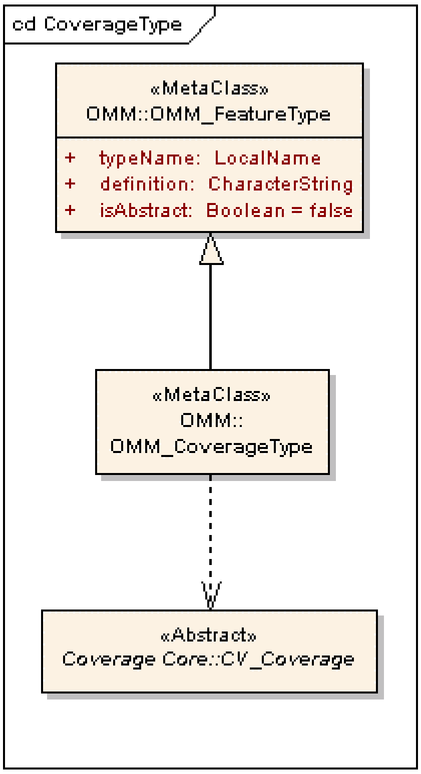


Figure 26: Schema of the OMM Extension "Coverage Type"

8.8 Rules for ORCHESTRA Application Schemas

8.8.1 General Approach

The modelling process for OAS on the platform-neutral level corresponds to the description in ISO 19109, section 8.1. This approach allows automatic derivation of platform-specific application schemas (e.g. GML Application Schemas according to ISO/DIS 19136) from the conceptual application schemas in a normative way. GML Application Schemas can be used to encode ORCHESTRA feature instances in XML. GML is tightly integrated with most OGC Web Service specifications, e.g. the Web Feature Service. In addition, mapping to other platforms is possible from the conceptual UML model.

Note 1: The relationship to the rules for application schemas as specified in ISO 19109, section 8, (conformance, changes and/or extensions) is explicitly indicated in respective notes.

Note 2: Changes during the course of the ISO/DIS 19136 standardization process that influence the rules for the OAS design will be incorporated in future versions of the RM-OA as required.

Rules:

1) The data structures of the application shall be modelled in the OAS.

   Note: Rule conforming to ISO 19109, section 8.2.2, rule 1).

2) An abstract specification of an OAS shall use UML 2.0 as its conceptual schema language following the rules of ISO/PDTS 19103 and ISO 19109. It shall be documented using class diagrams.

   Note: ISO/PDTS 19103. Geographic information - Conceptual schema language is still based on UML 1.3. A potential conflict will have to be resolved in dedicated rules.

3) An OAS shall use the UML extensibility mechanisms "stereotypes" and "tagged values" as described in annex D.8 of ISO/PDTS 19103.

   Note 1: A stereotype is a model element that is used to classify (or mark) other UML elements so that they in some respect behave as if they were instances of new virtual or pseudo meta-model classes whose form is based on existing base meta-model classes. Stereotypes augment the classification mechanisms on the basis of the built-in UML meta-model class hierarchy. Therefore, names of new stereotypes must not clash with predefined meta-model elements or other stereotypes. See section 8.8.6 for the rules how to use stereotypes in an OAS.

   Note 2: A tagged value is a tag-value pair that can be used to add properties to any model element in UML, i.e. it can extend an arbitrary existing element in the UML meta-model or extend a stereotype.

8.8.2 Rules for the Identification of an OAS

Rules:

1) The identification of each application schema shall include a name and a version. The inclusion of a version ensures that a supplier and a user agree on which version of the application schema describes the contents of a particular dataset.

   Note 1: This rule conforms to ISO 19109, section 8.2.3, rule 2).

   Note 2: The agreement between supplier and user also covers the case where there is no explicit bilateral agreement, but where the user is able to discover and understand which version(s) of an application schema are supported by the supplier.

   Note 3: It is recommended that the name of an OAS be globally unique (e.g. an URI) in order to enable unambiguous re-use of its elements in other OAS.

2) In UML, an application schema shall be described within a PACKAGE, which shall be stereotyped with <<Application Schema>> and shall contain the tagged value "OAS" carrying the name of the application schema and the tagged value "version" carrying the version stated in the documentation of the PACKAGE.

   Note 1: This rule extends ISO 19109, section 8.2.3, rule 1).

   Note 2: An OAS may consist of several hierarchically ordered packages. In this case, the OAS name corresponds to the name of the top-level package.

8.8.3 Rules for the Documentation of an OAS

Rules:

1) An OAS shall be documented.

   Note: This rule conforms to ISO 19109, section 8.2.4, rule 1).

2) The documentation of an OAS shall include a reference to the version of the RM-OA that has been used by setting the tagged value "RM-OA" to the version number of the RM-OA document.

3) The documentation of an OAS in UML may utilise the documentation facilities of the software tool that is used to create the application schema, if this information can be exported.

   Note: This rule conforms to ISO 19109, section 8.2.4, rule 2).

4) Documentation of the elements in the UML model shall be stored in tagged values "documentation".

5) If a CLASS or other UML component corresponds to information in a feature catalogue, the reference to the catalogue shall be documented.

Note: This rule conforms to ISO 19109, section 8.2.4, rule 3).

6) Documentation of feature types in an OAS shall be in a catalogue with a structure derived from OMM, for instance in a catalogue in accordance with ISO 19110

Note: This rule conforms to ISO 19109, section 8.2.4, rule 4).

8.8.4 Rule for the Integration of an OAS and other Schemas

Rules:

1) An OAS can be built up of several other application schemas. Each of these schemas can refer to standardised schemas. This organisation can be used to avoid the creation of large and complex schemas (see ISO 19109, section 8.2.6).

2) The dependency mechanism in UML shall be used to describe the integration of the OAS with other application schemas or other standard schemas that are required to form the complete definition of the data structure.

Note: This rule is derived from ISO 19109, section 8.2.5, rule 1).

8.8.5 Rules for the Usage of Types in an OAS

Rules:

1) Basic Data Types as specified in section 8.7.2.2 and OA_Types as specified in section 8.7.2.3 shall be used where applicable.

2) Types defined in OA Services (see section 9.3.2) shall be prefixed by OA_.

Note: An example is the OA_GetCapabilitiesRequest type defined in the *ServiceCapabilities* interface type (see section 9.6.1).

3) Types defined in OT Services (see section 9.3.3) shall be prefixed by OT_.

4) An OAS designer is not enjoined to use prefixes for the specification of user-defined types (e.g. in an OAS), however, OA_ and OT_ are excluded.

8.8.6 Rules for the Usage of Stereotypes in an OAS

Rules:

1) Every class in an application schema must be stereotyped. The stereotype used must be defined either in the standard UML or the stereotypes defined within the OMM. If the stereotype has a name common to the names of those stereotypes already specified, the definition (meaning) has to be the same.

Note: This facilitates the understanding of OAS and supports application development, e.g., to help decide whether a class is a feature type or not.

2) Data types shall be modelled as UML classes with the stereotype <<DataType>>.

Note: According to ISO/PDTS 19103 a <<DataType>> is a descriptor of a set of values that lack identity (independent existence and the possibility of side effects). The primary purpose of a DataType is thus to hold the abstract state of another class (e.g. a class representing a feature type) for transmittal, storage, encoding or persistent storage. An example in the OMM is the aggregation of operation request parameters in one class (see section 9.2.8).

3) Types shall be modelled as UML classes with the stereotype <<Type>>.

Note 1: According to ISO/PDTS 19103, a <<Type>> is a stereotyped class used for specification of a domain of instances (objects), together with the operations applicable to the objects. A type may have attributes and associations.

Note 2: For the definition of the types and their classification see section 8.8.5.

4) Enumerations shall be modelled as UML classes with the stereotype <<Enumeration>>.

Note: See section 8.8.5 for the definition of an enumeration as a basic type in an OAS.

5) Code lists shall be modelled as UML classes with stereotype <<CodeList>>.

Note 1: According to ISO/PDTS 19103, a code list can be used to describe an open enumeration (see rule 4 above). This means that it needs to be represented in such a way that it can be extended during system runtime.

Note 2: See section 8.8.5 for the definition of an enumeration as basic type in an OAS.

6) Interfaces shall be modelled as UML classes with stereotype<<Interface>>.

Note: See the corresponding rule of the OMM-Service in section 9.2.6.

8.8.7 Rules for the Specification of an OAS

Rules:

1) All classes used within an OAS for data transfer shall be instantiable. This implies that the integrated class must not be stereotyped <<interface>>.

Note: This rule conforms to ISO 19109, section 8.2.2, rule 2).

2) All package names used within an OAS shall be unique.

3) Dependencies between packages must be modelled explicitly.

4) If a class is a specialization of another class, then this class shall have one of the stereotypes <<FeatureType>>, <<DataType>>, or <<Type>>. The class shall have zero or one supertype with the same stereotype and zero or more abstract supertypes of the stereotype <<Type>>.

That is, disregarding abstract classes with stereotype <<Type>>, a generalization relationship may be specified only between two classes that are either:

- both feature types (stereotype <<FeatureType>>),
- both types with stereotype <<Type>>, or
- both data types (stereotype <<DataType>>).

For every abstract class <<Type>> all direct or indirect subtypes must be either

- all classes with stereotypes <<FeatureType>>, <<Type>>, or
- all classes with stereotypes <<DataType>> or <<Type>>, where all <<Type>> classes have to be abstract.

All generalization relationships between classes shall have no stereotype. All generalization relationships with other stereotypes will be ignored. The discriminator property of the UML generalization shall be blank.

5) OMM_FeatureType: An instance of OMM_FeatureType shall be implemented as a CLASS stereotyped with <<FeatureType>> except for Rule 6 (see OMM_AssociationType below).

Note: This rule extends ISO 19109, section 8.3.1, rule 1).

6) OMM_AssociationType: An instance of OMM_AssociationType shall not be associated with any instances of OMM_PropertyType. It has the role of linkBetween in associations to those instances of OMM_FeatureType that are being implemented as CLASSes.

   Note 1: This rule conforms to but restricts ISO 19109, section 8.3.1, rule 2).

   Note 2: This rule means that attributed associations between feature types (i.e. associations with own properties) are not supported.

7) OMM_AggregationType: An instance of OMM_AggregationType shall either be implemented as an AGGREGATION (empty diamond) or it shall be implemented as a COMPOSITION (filled diamond). Members of an aggregation can exist independently of the aggregate, and may belong to other aggregates. Members of a composite may not exist independently and may belong to only one composite.

   Note: This rule conforms to ISO 19109, section 8.3.1, rule 3).

8) OMM_AttributeType: An instance of OMM_AttributeType shall be implemented as an ATTRIBUTE, unless it is an attribute of an attribute (see rule 5)

   Note: This rule conforms to ISO 19109, section 8.3.1, rule 4).

9) attributeOfAttribute: An instance of OMM_AttributeType that acts in the role characterizedBy in an attributeOfAttribute association shall be instantiated as a class with a valid stereotype for classes (e.g., <<FeatureType>>). That class shall be used either as the data type of the OMM_AttributeType, or in an association with the class that contains the OMM_AttributeType. Attributes that act in the role characterizes shall be instantiated as attributes of the class that represents the attribute that acts in the role characterizedBy.

   Note 1: This rule extends ISO 19109, section 8.3.1, rule 5).

   Note 2: This means that a class stereotyped as <<FeatureType>> may be used as a data type of an attribute in a class definition

10) OMM_Operation: An instance of OMM_Operation shall be implemented as an OPERATION of the class representing the feature type that it characterizes, which shall have ASSOCIATIONS to other CLASSES from which the operation needs ATTRIBUTE VALUES.

    Note 1: This rule conforms to ISO 19109, section 8.3.1, rule 6).

    Note 2: The relationship between an operation specified in a feature type and operations specified in interface types (i.e. the link to the OMM-Service meta-classes) will be investigated in a further version of the RM-OA.

11) OMM_AssociationRole: An instance of OMM_AssociationRole shall be implemented as a role name at the appropriate end of the ASSOCIATION representing the OMM_AssociationType.

    Note: Rule conforming to ISO 19109, section 8.3.1, rule 7).

12) OMM_InheritanceRelation: An instance of OMM_InheritanceRelation shall be represented by a UML GENERALIZATION relationship, with the following additional characteristics: If uniqueInstance is .TRUE., the {disjoint} constraint shall be attached to the generalization relationship.

    Note: This rule is derived from ISO 19109, section 8.3.1, rule 8).

13) OMM_Constraint: Constraints may be stated in OCL or in plain language and attached to the CLASS, OPERATION or RELATIONSHIP that is constrained. A formal specification of constraints is required when automatic processing is intended.

Note: This rule extends ISO 19109, section 8.3.1, rule 9).

8.8.8 Rules for Adding Information to a Standard Schema

Rule:

1) If it is necessary to extend or restrict a CLASS specified in a standard schema, a new CLASS shall be defined as a SUBTYPE of the CLASS in the standard schema, and ATTRIBUTEs shall be added to this CLASS to carry the additional information.

   Note 1: This rule conforms to ISO 19109, section 8.4.2, rule 1).

   Note 2: For practical reasons the new classes may be collected in a separate PACKAGE.

8.8.9 Rules for restricted Use of Standard Schemas

Rules:

1) Specification of a restricted profile of a standard schema shall be described in a new UML package by copying the actual definitions (classes and relationships) from the standard schema. Attributes and operations within classes may be omitted.

   Note: This rule conforms to ISO 19109, section 8.4.3, rule 1).

2) Reduction of a standard schema shall be in accordance of the conformance clause given for the actual standard.

   Note 1: This rule conforms to ISO 19109, section 8.4.3, rule 2).

   Note 2: The specifications of OMM extension types (see section 8.7.5) are handled like standard schemas. The rules to be considered for a possible reduction are specified in section 8.8.15.

8.8.10 Rules for Adding Information to an OAS

Rule:

1) If it is necessary to extend a CLASS specified in an OAS, a new CLASS shall be defined as a SUBTYPE of the CLASS in the standard schema, and ATTRIBUTEs shall be added to this CLASS to carry the additional information.

8.8.11 Rules for Thematic Attributes

Rule:

1) A thematic attribute may reuse definitions from a package in the ISO 19115 without being considered as meta-information in the application schema.

   Note: This rule conforms to the RM-OA approach to handle meta-information (see section 8.4.1). Whether an attribute is to be considered as meta-information cannot be decided at design time.

8.8.12 Rules for Temporal Attributes

Rules:

1) If a common representation of time across systems is required then it is recommended that any description of temporal aspects be in accordance with the specifications given by ISO 19108.

   Note: This recommendation is still to be validated in the course of the ORCHESTRA specification and implementation process, in particular w.r.t. to the usage of the basic data types “date” and “time” as specified in section 8.7.2.2.

2) The usage of temporal attributes according to ISO 19108 in an OAS shall comply with the specifications and rules of ISO 19109, section 8.6, if not otherwise specified in the RM-OA.

Note: This recommendation is still to be validated in the course of the ORCHESTRA specification and implementation process, in particular in the handling of time-series by the Map and Diagram Service (see section 9.6.3).

8.8.13 Rules for Spatial Attributes

Rules:

1) The value domain of spatial attribute types shall be in accordance with the specifications given by ISO 19107, which provides conceptual schemas for describing the spatial characteristics of features and a set of spatial operators consistent with these schemas. ISO 19125-1 is a profile of 19107 that is widely adopted (see the OGC simple feature specification). If in the process of specifying an OAS there is no explicit need to use other data types than those specified in ISO 19125-1, then ISO 19125-1 shall be used.

   Note: This rule extends ISO 19109, section 8.7, rule 1).

2) The usage of spatial attributes according to ISO 19107 and ISO 19125-1 in an OAS shall comply with the specifications and rules of ISO 19109, section 8.7, if not specified otherwise in the RM-OA.

8.8.14 Rules for Spatial Referencing using Geographic Identifiers

Rules:

1) The value domain of attributes using spatial referencing by geographic identifiers shall be in accordance with the specifications given in ISO 19112.

   Note: This rule conforms to ISO 19109, section 8.9, rule 1).

2) The usage of attributes using spatial referencing by geographic identifiers according to ISO 19112 in an OAS shall comply with the specifications and rules of ISO 19109, section 8.9, if not specified otherwise in the RM-OA.

8.8.15 Rules for Information Types extending the OMM

8.8.15.1 Feature Types vs. Attribute Types

Depending on the semantics, a particular piece of information may be considered either a feature (type) or a value of an attribute (type). When modelling, it is often a judgement call whether to model a particular type one way or the other.

As a general rule, a feature type will be used if the concept is of particular importance for the application, has an identity of its own and can be considered to be an "abstraction of a real world phenomenon."

On the other hand, a concept will be modelled as a data type of an attribute if the concept does not have an identity on its own (i.e. it is just a structured attribute) or if it is just an auxiliary concept and will only be used in the context of a feature (e.g. a geometry or topology object).

8.8.15.2 Rules for Coverages

Coverages are considered in the OMM as instances of ORCHESTRA feature types, see section 8.7.5.2. Their schema is defined in ISO 19123.

Rules:

1) Any description of coverage information shall be in accordance with the specifications given by ISO 19123.

2) A coverage type shall be defined as a coverage feature type which is the appropriate, most specialized type defined in ISO 19123 listed in rule 5 or a subtype of this type.

3) The implementation of a coverage type in UML shall follow the rules (see ISO 19109 8.2.5) for referencing standardised schemas (see RM-OA, section 8.8.4, rule 2).

4) A coverage type shall be represented in an application schema as a UML CLASS that represents a feature (see RM-OA, section 9.2.5.2) and which is derived directly or indirectly from one of the UML classes from rule 5.

5) Valid coverage feature types which shall be applied are::

   - Discrete coverages (CV_DiscreteCoverage)
     - Discrete point coverage (CV_DiscretePointCoverage)
     - Discrete grid point coverage (CV_DiscreteGridPointCoverage)
     - Discrete curve coverage (CV_DiscreteCurveCoverage)
     - Discrete surface coverage (CV_DiscreteSurfaceCoverage)
     - Discrete solid coverage (CV_DiscreteSolidCoverage)
   - Continuous coverages (CV_ContinuousCoverage)
     - Thiessen polygon coverage (CV_ThiessenPolygonCoverage)
     - Hexagonal grid coverage (CV_HexagonalGridCoverage)
     - TIN coverage (CV_TINCoverage)
     - Segmented curve coverage (CV_SegmentedCurveCoverage)
     - Continuous quadrilateral grid coverage (CV_ContinuousQuadrilateralGridCoverage)

   Note: Whether all of these coverage types are required for most of the applications of the RM-OA or if they may be restricted is yet to be determined.

8.8.15.3 Rules for Documents

Documents are considered in the OMM as instances of ORCHESTRA feature types. Their schema is defined in section 8.7.5.2.

Rules:

1) A document type shall be represented in an OAS as an attribute (an instance of OMM_ThematicAttributeType) of a UML CLASS that represents the feature, in which case the attribute shall take OA_DocumentDescriptor as defined in section 8.7.5.2 and Figure 24 or a subtype as the data type for its value.

8.9 A Simple Example

An extremely simplified model of an earthquake feature type is illustrated in Figure 27. In terms of the OMM, the feature type "XE_Earthquake" has the following own properties:

- an optional thematic attribute type with the name "magnitude", the value is a numeric value between 0 and 10 (Richter scale);
- an optional feature association role with the name "officialReport" to a document feature type(see section 8.7.5.2).

Furthermore, by means of multiple inheritance according to the rules specified in section 8.8.7, the XE_Earthquake class inherits the following properties:

- from the feature type “Hazard”: a spatial property type with the name "location", the value type is a spatial point (see ISO 19107).
- from the feature type “Hazard”: a temporal property type with the name "occurredAt", the value type is a temporal instant (see ISO 19108).
- from the type “ObjectWithMetadata”: an optional meta-information property type with the name “metadata”; the value type is a metadata entity (see ISO 19115).

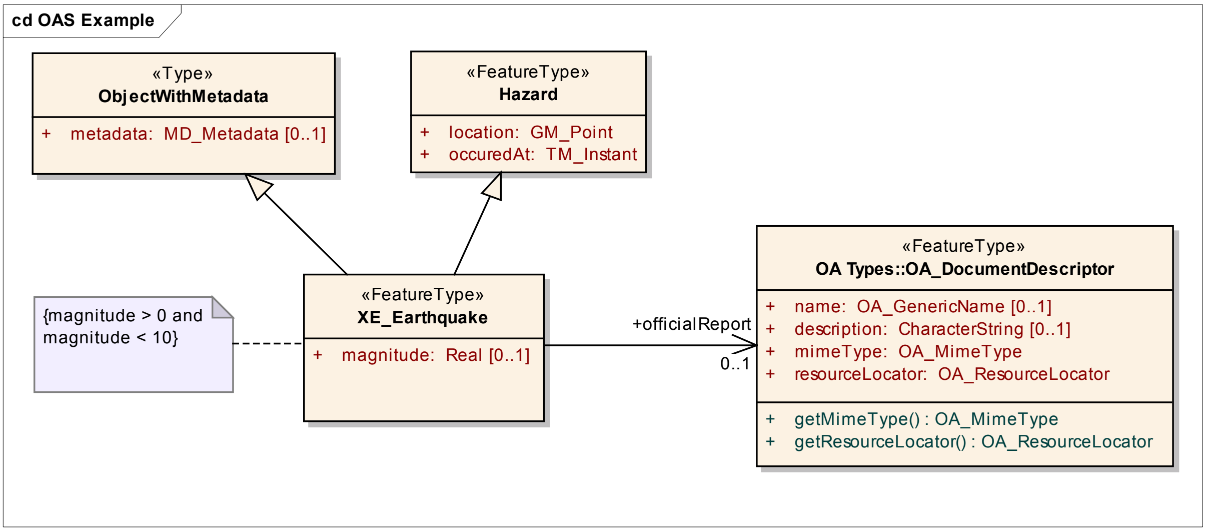

cd OAS Example
«Type»
ObjectWithMetadata
+ metadata: MD_Metadata [0..1]
«FeatureType»
Hazard
+ location: GM_Point
+ occuredAt: TM_Instant
{magnitude > 0 and magnitude < 10}
«FeatureType»
XE_Earthquake
+ magnitude: Real [0..1]
+officialReport
0..1
«FeatureType»
OA Types::OA_DocumentDescriptor
+ name: OA_GenericName [0..1]
+ description: CharacterString [0..1]
+ mimeType: OA_MimeType
+ resourceLocator: OA_ResourceLocator
+ getMimeType() : OA_MimeType
+ getResourceLocator() : OA_ResourceLocator

Figure 27: Earthquake example

9 Service Viewpoint

9.1 Overview

The Service Viewpoint of the RM-OA specifies the specification framework for ORCHESTRA Services. This specification framework is provided by the definition of a Service Meta-Model as given in section 9.2.

Furthermore, the Service Viewpoint of the RM-OA provides abstract specifications for the generic ORCHESTRA Services that support the syntactic and semantic interoperability between ORCHESTRA Source Systems and between services and the development of ORCHESTRA Applications. This includes the management of an OSN as one particular application.

In combination with the specification of the ORCHESTRA Information Viewpoint, this specification provides the ORCHESTRA Architecture. According to RM-OA principles, the abstract description of ORCHESTRA Services and the abstract specification of their interfaces include all properties of the services that may be specified in a platform-neutral way. Their mapping to specific service platforms (e.g. a W3C Web Services environment) is outside the scope of the RM-OA and is specified in ORCHESTRA Implementation Specifications.

Section 9.2 provides a Service Meta-model (OMM-Service) as a complementary part of the OMM Information Meta-model (OMM-Information).

ORCHESTRA Services are functionally classified in section 9.3

The RM-OA specifies the ORCHESTRA Services and their interfaces in two different ways:

- A coarse abstract service description is given for each service in human-readable text format by using a service description framework, see section 9.4.
- A refined abstract specification of the interfaces to be realised by the services is given in (ORCH-AbstrServ 2007) by using UML as the conceptual schema language.

Note: Whereas the OMM-Information is an evolution of the General Feature Model (GFM) of ISO 19109 (see section 8.3), the ISO counterpart for the OMM-Service would be the UML model supplied in section 7.2 of ISO 19119 which is, however, not directly related to the GFM. Furthermore, it does not cover the problem of abstract and implementation specification of services. The meta-model approach of ORCHESTRA aims at a harmonised approach for both the information and the service viewpoint with direct interdependencies and rules about how to handle the problem of platform-neutral and platform-specific service specifications and the mapping between them. A need for such an approach has recently been expressed by the Object Management Group (OMG) in their Request For Proposal for a “Software Services Profile and Metamodel” (OMG 2006).

9.2 The ORCHESTRA Meta-Model for Services

9.2.1 Overview

An ORCHESTRA Service is a service specified according to the rules of the ORCHESTRA Reference Model in an ORCHESTRA Service Specification. As with the Information Viewpoint of the RM-OA, these rules are provided by means of a Service Meta-Model as further part of the ORCHESTRA Meta-Model (OMM).

In the Information Viewpoint, the OMM has been defined as the common specification framework for all feature-based application schemas used within ORCHESTRA. It provides a meta-model and a set of associated rules that control the specification of an OAS. This part of the OMM is called OMM-Information in the following. For the Service Viewpoint the schema level is extended by the concept of ORCHESTRA Service Types. The corresponding rules for their specification are defined in a respective extension of the OMM called OMM-Service in the following.

The framework for ORCHESTRA Services is illustrated in Figure 28. It distinguishes between

- the ORCHESTRA Meta-Model (OMM) on the meta-model level,

- ORCHESTRA Service Specifications on the schema level,
- ORCHESTRA Services on the service level and
- the functionality provided by source systems on the source system level.

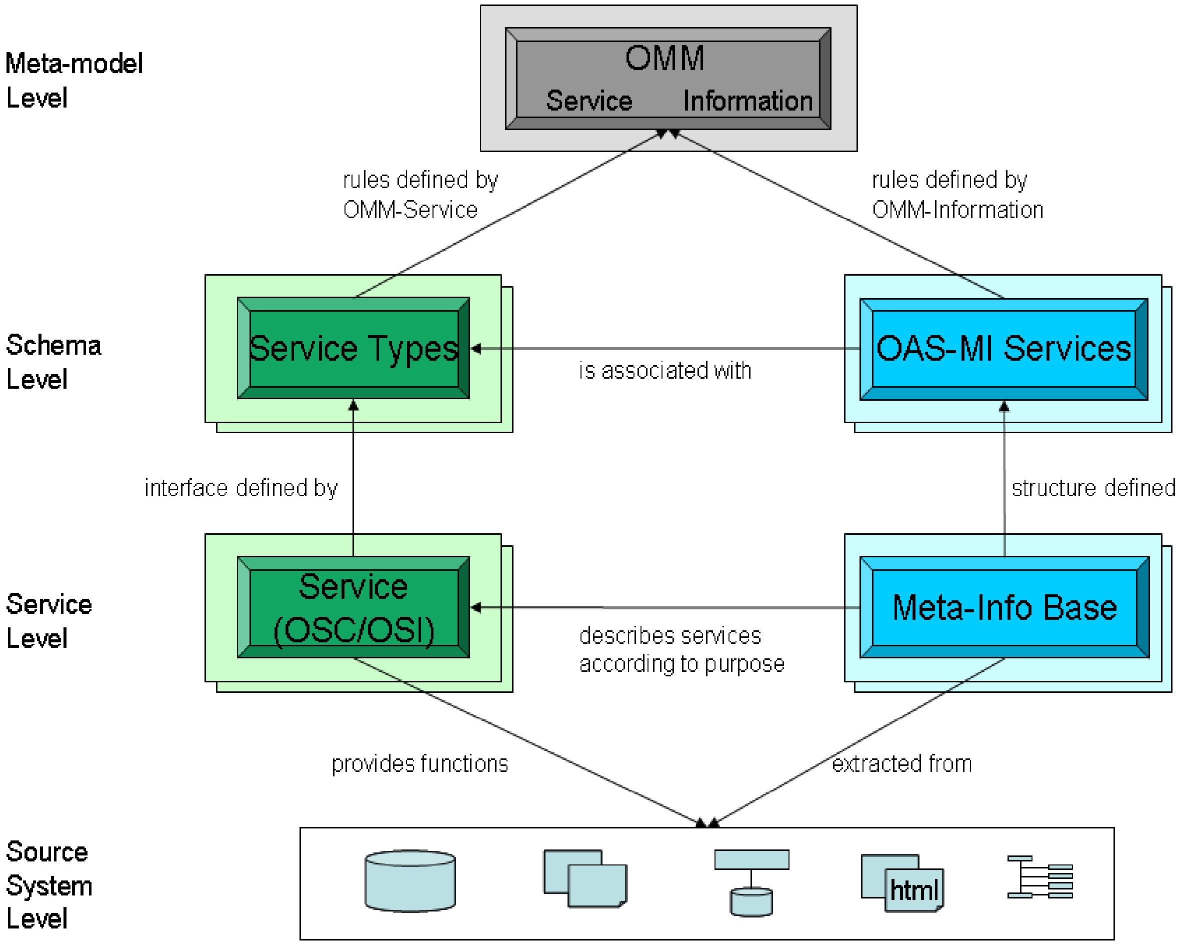


Figure 28: Framework for ORCHESTRA Services

ORCHESTRA Service Types are specified by defining their externally visible behaviour accessible through their service interfaces (see section 9.2.2.3). The service interfaces, including their information models, are expressed using the conceptual schema language UML in the first step (abstract specification), and then mapped to a chosen platform in a second step (implementation specification).

On the schema level, meta-information models are associated to ORCHESTRA Service Types in so-called OAS-MI for Services according to the rules of the Information Viewpoint (OMM-Information) specified in section 8.7. These OAS-MI deliver the schema for the meta-information that is associated with service types in order to serve the various purposes (e.g. discovery of services) as outlined in section 8.4.2.

The service level is built by the set of ORCHESTRA Services and the meta-information base as the logical aggregation of the meta-information that describes the ORCHESTRA Services according to the various purposes. The meta-information base is structured according to the OAS-MI specified on the schema level. ORCHESTRA Services are instances of ORCHESTRA Service Types and have two different appearances:

- as ORCHESTRA Service Components (OSC) when referring to the software component that implements the interfaces defined for the ORCHESTRA Service Types on the schema level, and
- as ORCHESTRA Service Instances (OSI) when referring to deployed and running instances of

OSCs in an OSN.

In the Service Viewpoint, the source system level consists of the set of source systems whose functionality is to be integrated into an OSN. For this purpose, source system-specific service types have to be specified by the system integrator and instantiated as OSIs such that the functions of the source systems may be offered to ORCHESTRA Applications in an ORCHESTRA-compliant way. Note that there is no generic ORCHESTRA Service Type defined for this integration. Instead, the interface types as defined in the RM-OA may be re-used. For a discussion about this integration process, see section 9.9.2.

Furthermore, in order to fill the meta-information bases on the service level, descriptive information about the source systems' functionality is extracted (manually or semi-automatically) from the source systems.

Note: A future RM-OA version will extend the framework for ORCHESTRA Services by the inclusion of the semantic level.

9.2.2 Service Types

9.2.2.1 Overview

According to ISO 19119, a service is defined as a distinct part of the functionality that is provided by an entity through interfaces. If such a service has been being defined according to the rules of the ORCHESTRA Reference Model, it is called ORCHESTRA Service. However, the design and internal behaviour of such entities is outside the scope of the ORCHESTRA Architecture. They are conceived and identified by a designer of an OSN and are called

- ORCHESTRA Service Component when referring to the software component and
- ORCHESTRA Service Instance when referring to an instance in an OSN that has been deployed by a service provider with a dedicated identifier (see section 11.1.2), and whose operations may be called by a service consumer.

Principally, the ORCHESTRA Architecture just deals with types of ORCHESTRA Services. ORCHESTRA Service Types (short: service types) are described on a platform-neutral level in abstract service descriptions which refer to specifications of the interfaces that together provide the externally visible behaviour of the service type. In the ideal case, through a service mapping process, such a service type is mapping to respective implementation specifications for one or more given platforms. When implemented they result in ORCHESTRA Service Components and are later deployed as ORCHESTRA Service Instances in ORCHESTRA Service Networks.

Note, however, that for convenience and readability reasons the RM-OA only distinguishes between ORCHESTRA Service Types, ORCHESTRA Service Components and ORCHESTRA Service Instances when only one is meant. Otherwise, the more general term ORCHESTRA Service is used.

The conceptual schema for the specification of an ORCHESTRA Service Type is provided in the subsequent sections and illustrated in Figure 29. The main ideas are as follows:

- There is a 1:1 relationship between the abstract description of an ORCHESTRA Service Type and an ORCHESTRA Service Type. This means that each abstract service description exactly specifies one service type and vice versa.
- There is a 1:n relationship between an ORCHESTRA Service Type and an implementation specification of an ORCHESTRA Service Type. This means that each implementation specification of an ORCHESTRA Service exactly specifies one service type, and, for each service type there may be one or more corresponding implementation specifications.
- As a consequence, there is a common list of ORCHESTRA Service Types for platform-neutral and platform-specific specifications.

9.2.2.2 Platform Properties

As a general guideline, the platform shall be conformant to the OASIS Reference Model for Service Oriented Architecture 1.0 (SOA-RM, 2006). Thus, when referring in the RM-OA to characteristics of the service platform, the following terms of (SOA-RM, 2006) are used. Note that they are only pre-fixed with SOA-RM in order to distinguish them from RM-OA terms:

- SOA-RM Service: The means by which the needs of a consumer are brought together with the capabilities of a provider.
- SOA-RM Capability: A real-world effect that a service provider is able to provide to a service consumer.
- SOA-RM Action model: The characterization of the permissible actions that may be invoked against a service.

  Note: Interacting with a service involves performing transactions with the service. Usually this is accomplished by sending and receiving messages.
- SOA-RM Service Interface: The means by which the underlying capabilities of a service are accessed.
- SOA-RM Information Model: The characterization of the information that is associated with the use of a service. Only information and data that are potentially exchanged with a service are generally included within that service's information model. The scope of the information model includes the format of information that is exchanged, the structural relationships with the exchanged information and also the definition of terms used.
- SOA-RM Execution Context: The set of technical and business elements that form a path between those with needs and those with capabilities and that permit service providers and consumers to interact.

9.2.2.3 OMM_ServiceType

The conceptual schema for the specification of ORCHESTRA Service Types is illustrated in Figure 29 (see meta-class OMM_ServiceType). The structural refinement of service types in terms of interface types is given in Figure 30 (see meta-class OMM_InterfaceType).

An ORCHESTRA Service Type is modelled by the meta-class OMM_ServiceType with the following properties:

- name: Provides the name of the service type. This name shall indicate the intended behaviour of the service type and may be used in the identification of a service type by a human user.
- abstractDesc: Association role providing a reference to the abstract description of the service type (see OMM_ServiceAbstractDesc).
- implSpec: Association role providing the list of references to service implementation specifications (see OMM_ServiceImplSpec). A reference is provided through the name of the corresponding implementation specification of the service type.
- ifName: Association role providing the list of interface types (see OMM_InterfaceType) that are supported by the service type.

OA_ServiceType is an instance of the meta-class OMM_ServiceType. Rules for ORCHESTRA Service Types are provided in section 9.2.5.2.

The functional classification of ORCHESTRA Service Types is described in section 9.3

9.2.3 Structure of the ORCHESTRA Service Specification Process

The structure of the specification process for ORCHESTRA Services is illustrated by the conceptual models specified in UML in Figure 29. According to the ORCHESTRA Reference Model as described in section 5.3, ORCHESTRA Service Types are specified on a platform-neutral and on a platform-specific level.

The abstract specification level is represented by the meta-classes OMM_ServiceAbstractDesc and OMM_InterfaceAbstractSpec whereas the platform level is represented by the meta-classes OMM_ServiceImplSpec, OMM_ServiceMappingSpec and OMM_PlatformSpec.

9.2.3.1 OMM_ServiceAbstractDesc

OMM_ServiceAbstractDesc represents an abstract description of an ORCHESTRA Service Type that is platform-neutral (i.e. independent of a particular service platform) and may thus be mapped to several service platforms. It provides a summary description of the functionality that the service type offers to a calling client through its external interface. This description may be provided in different forms but in most cases comprises a human-readable text. An example for such a description is the service description framework used in the RM-OA, see section 9.4. However, the abstract description of a service is also considered to be meta-information about the service type. Thus, respective OAS-MI or parts of it may also be used as abstract service descriptions. See Annex A3 of the RM-OA for examples.

OMM_ServiceAbstractDesc has the following properties:

- serviceType: Association role providing the name of the service type that is being described.
- description: Description of the purpose and functionality provided by the service type..
- ifSpec: Association role providing the list of abstract specifications of the interfaces (OMM_InterfaceAbstractSpec) that are supported by the service type that is described in the abstract description.

9.2.3.2 OMM_InterfaceAbstractSpec

OMM_InterfaceAbstractSpec represents an abstract specification of an interface type that is platform-neutral (i.e. independent of a particular service platform). It comprises a collection of operations that together provide a self-contained set of functionality in the sense that its granularity is eligible to be re-usable by other service types.

OMM_InterfaceAbstractSpec has the following properties:

- ifName: Association role providing the name of the interface type that is being specified.
- spec: Specification of the purpose and functionality of the interface type.

9.2.3.3 OMM_ServiceImplSpec

OMM_ServiceImplSpec represents an implementation specification of an ORCHESTRA Service that is specified according to the rules of a particular service platform.

- name: Name of the implementation specification of the service type.
- actionModel: Specification of the permissible actions against the service type, i.e. the SOA-RM Action Model of the service type.
- abstractDesc: Association role providing the reference to the abstract service description upon which the implementation specification is based (see OMM_ServiceAbstractDesc).
- platformSpec: Association role providing the specification of the (service) platform for which the implementation specification is valid (see OMM_PlatformSpec).
- mappingSpec: Association role providing the reference to the specification of the service mapping that links the SOA-RM Action Model of the implementation specification to the operations of the abstract service interfaces (see OMM_ServiceMappingSpec). Such a mapping specification is a mandatory part of the implementation specification of a service.

As the ORCHESTRA Architecture provides the platform-neutral view, the OMM-Service only provides detailed rules for the abstract descriptions and interface specifications of ORCHESTRA Services (see sections 9.2.5.3 and 9.2.6). However, some general rules for implementation service specifications are given in section 9.2.11.

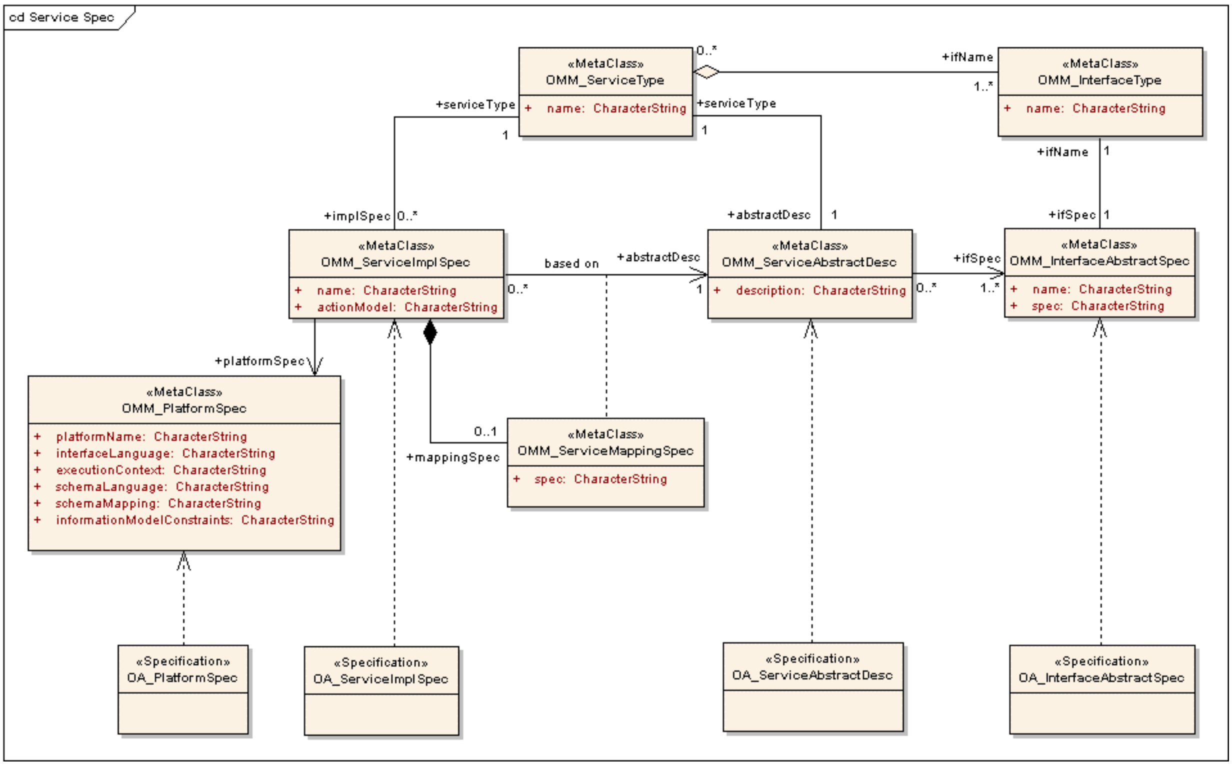


Figure 29: Specification Process for ORCHESTRA Services

9.2.3.4 OMM_ServiceMappingSpec

When purely applying the architectural process of ORCHESTRA, there is a service mapping process between an abstract description and an implementation specification of an ORCHESTRA Service. This process is modelled by the meta-class OMM_ServiceMappingSpec with the properties:

- spec: Specification of how to map from the abstract level to the platform.

The service mapping process shall be carried out according to the rules given in section 9.2.9. Note that one abstract description of an ORCHESTRA Service Type may be mapped to several implementation specifications because

- implementation specifications are platform-specific, i.e. for each platform there is a dedicated implementation specification of service types, or
- the service mapping rules allow the specification of functional subsets or different concretisations of service types even for one platform.

The service mapping process also determines if an operation that is specified for a particular service type is to be called in a synchronous or in an asynchronous interaction. This is handled as part of the rules specified in section 9.2.9.

9.2.3.5 OMM_PlatformSpec

The two-step mapping approach from the abstract to the implementation service specification requires that the (service) platform has been specified beforehand in a platform specification. This is modelled by the meta-class OMM_PlatformSpec in Figure 29.

The OMM_PlatformSpec provides the following properties:

- platformName: Name of the platform. In case of a standard platform, a reference shall be provided.
- interfaceLanguage: Specification of the formal language that is used to define SOA-RM Service Interfaces. In case of a standard language, a reference shall be provided.

- executionContext: Specification of the SOA-RM Execution Context. In case of a standard SOA-RM Execution Context, a reference shall be provided.
- interfaceMapping: Specification of how the interface operations on the abstract level are mapped to actions of the SOA-RM Execution Context. This specification shall cover the following aspects:
  - principle handling of synchronous and the asynchronous interactions,
  - a description of the mechanisms by which “call by value” vs. “call by reference” action parameters are supported,
  - a description of if and how optional actions and optional action parameters are supported and what optionality means for this particular platform,
  - an implementation specification of the abstract interfaces as specified in the OA Basic Service (see section 9.6.1),
  - an implementation specification of the way the UAA concepts (see section 7.5) are realised for the platform, e.g. how session information is handled in the interactions.
- schemaLanguage: Specification of the schema language used to define SOA-RM Information Models.
- schemaMapping: Specification of how to map from the abstract level specified in UML to the schema language used in the platform and vice-versa.
- informationModelConstraints: Specification of the constraints on the SOA-RM Information Model, especially the constraints on the format of the messages that are required to accomplish the SOA-RM Action model.

An example for a platform is the Web Service infrastructure as defined by the W3C specifications (e.g. WSDL, SOAP V1.2) together with further refinements of ORCHESTRA, e.g. the determination of GML 3.2 as schema language and, if required, a specification of a GML schema profile. The corresponding platform mapping rules of how to map from UML to GML and vice versa are given in ISO/DIS 19136 Geography Markup Language (GML).

Rules for platform specifications are provided in section 9.2.10.

9.2.4 Interface Types

9.2.4.1 OMM_InterfaceType

Each ORCHESTRA Service Type shall refer to one or more interface types and each abstract description of a service type shall refer to one or more specifications of interface types. Furthermore, each interface type shall be specified in exactly one abstract specification of an interface.

An interface type is defined as the set of operations that characterize the externally visible behaviour of an entity providing the service. The aggregation of operations in an interface type and the definition of interface types shall be for the purpose of software reusability. The specification of an interface type shall include a static portion that includes a definition of the operations. The specification of an interface type shall include a dynamic portion that includes any restrictions on the order of invocation of the operations.

An interface type is modelled by the meta-class OMM_InterfaceType with the following properties:

- name: Provides the name of the service interface.
- opName: Association role providing the list of operations (see OMM_OperationType) that are defined in the service interface.

OA_Interface is an instance of the meta-class OMM_InterfaceType. The rules for specifying interface types according to the OMM are given in section 9.2.6.

9.2.4.2 OMM_ InterfaceInheritanceRelation

Interface types may be specialised by means of inheritance. Thus, generic interface types may be defined and re-used or refined in other abstract interface specifications. This is modelled by the meta-class OMM_InterfaceInheritanceRelation.

OMM_InterfaceInheritanceRelation is the meta-class that describes a generic relationship between a more general interface type (supertype) and one specialised interface type (subtype). An interface type A being a subtype of another interface type B (that acts as supertype) supports all operations defined in B in addition to the operations defined in A. An interface type may inherit operations from more than one supertype (multiple inheritance).

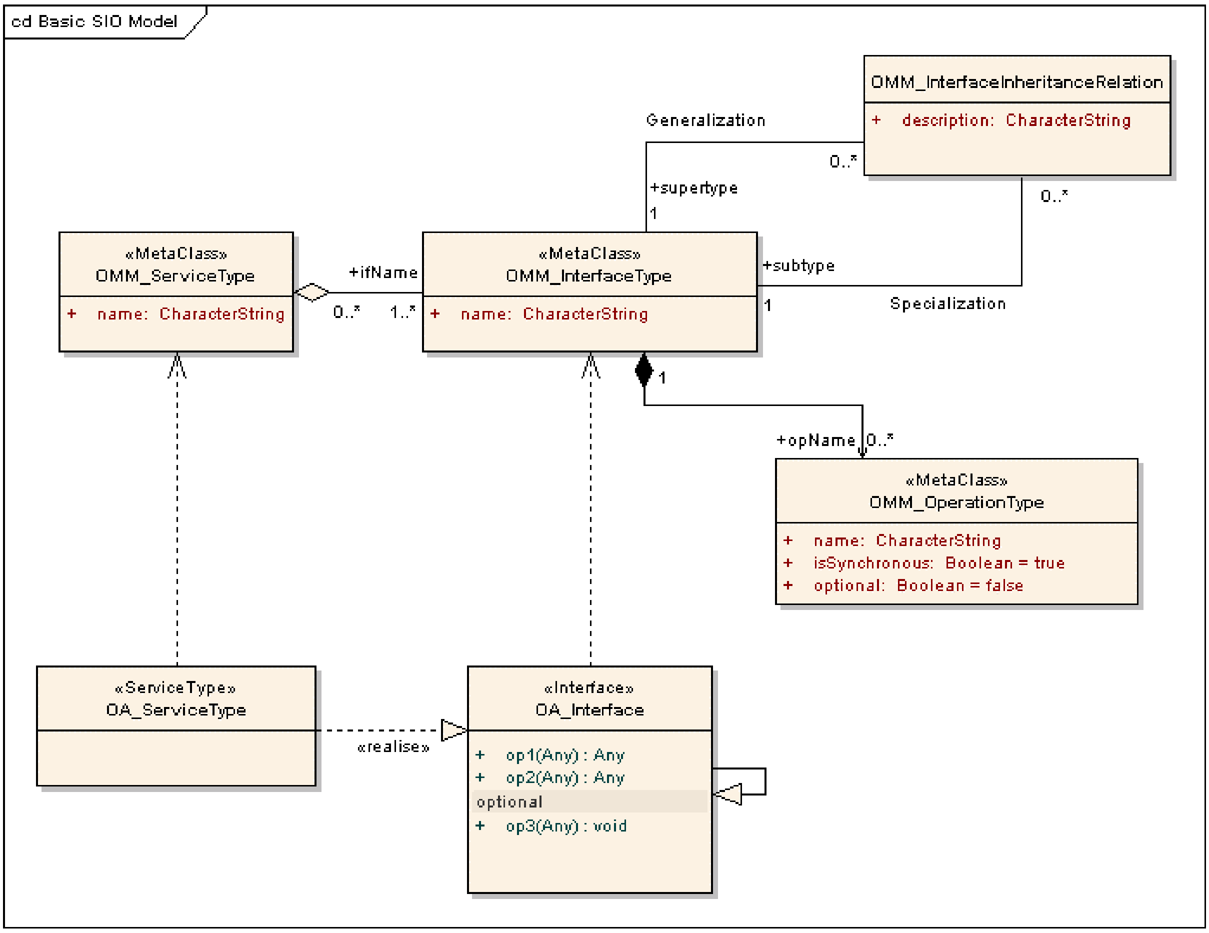


Figure 30: The Service Interface Part of the OMM

OMM_InterfaceInheritanceRelation is defined with the following properties:

- name: Name of the generalization/specialisation (optional).
- description: Explanation of the generalization/specialisation to be provided in the abstract interface specification.
- Generalization: Association specifying that an interface type has the role of being a supertype in an inheritance relationship with another interface type.
- Specialization: Association specifying that an interface type has the role of being a subtype in an inheritance relationship with another interface type.
- supertype: The role of being the more generic interface type of one other or many other interface types.
- subtype: The role of being the more specific interface type of one other or other interface types.

9.2.4.3 OMM_OperationType

The conceptual model for operations is illustrated in Figure 31. An operation type is syntacticly defined through its signature that consists of the name of the operation and the request, result and exception parameters. Operations are modelled in the meta-class OMM_OperationType with the following properties:

- name: Name of the operation type.
- optional: Boolean value indicating if the operation may be omitted in the service mapping from the abstract to the implementation specification (optional = true) or if it shall be supported in the respective SOA_RM Action Model of the an implementation specification (optional = false), in the latter case either as a mandatory action or as an optional action.
- request: Association specifying that an operation type may have zero, one or more request parameter types (OMM_RequestParameterType).
- result: Association specifying that an operation type may have zero or one result parameter types (OMM_ResultParameterType).
- exception: Association specifying that an operation type may have one or more request exception parameter types (OMM_ExceptionParameterType).

Rules for operation types are provided in section 9.2.7.

All parameter types are specified as subtypes of OMM_AttributeTypes. Therefore the rules that are specified for attribute types as part of the Information Viewpoint in section 8.7 are also applied for parameter types. In fact, this means that the totality of the information exchanged in operation requests, results and exceptions is specified as an OAS. Specific rules for parameter types are provided in section 9.2.8.

9.2.4.4 OMM_RequestParameterType

OMM_RequestParameterType is a meta-class representing a parameter to be provided as part of an operation request. It has the following properties:

- name: Name of the request parameter type.
- optional: Boolean value indicating if the request parameter may be omitted in the service mapping from the abstract to the implementation specification (optional = true) or if it shall be supported in the respective operation of the an implementation specification (optional = false), in the latter case either as a mandatory parameter or as an optional parameter.

9.2.4.5 OMM_ResultParameterType

OMM_RequestParameterType is a meta-class representing a parameter to be provided as part of an operation result if the processing of the operation has been successful. It has the following properties:

- name: Name of the result parameter type.

9.2.4.6 OMM_ExceptionParameterType

OMM_ExceptionParameterType is a meta-class representing a parameter to be provided as part of an operation exception if the processing of the operation has not been successful. It has the following properties:

- name: Name of the exception parameter type.

9.2.4.7 OMM_OperationRequest

OMM_OperationRequest is a meta-class representing the set of request parameters to be provided as part of an operation call. It has the following properties:

- opName: Association role representing the name of the corresponding operation.
- paraName: Association role referring to the set of request parameters required for the operation call.

Note: The meta-class OMM_OperationRequest is required in order to model the case where all request parameters are modelled in one UML class with the individual request parameters being attributes of this class. This is, for example, required when the *SynchronousInteraction* or the *AsynchronousInteraction* interface types as specified in the OA Basic Service (see section 9.6.1) are used.

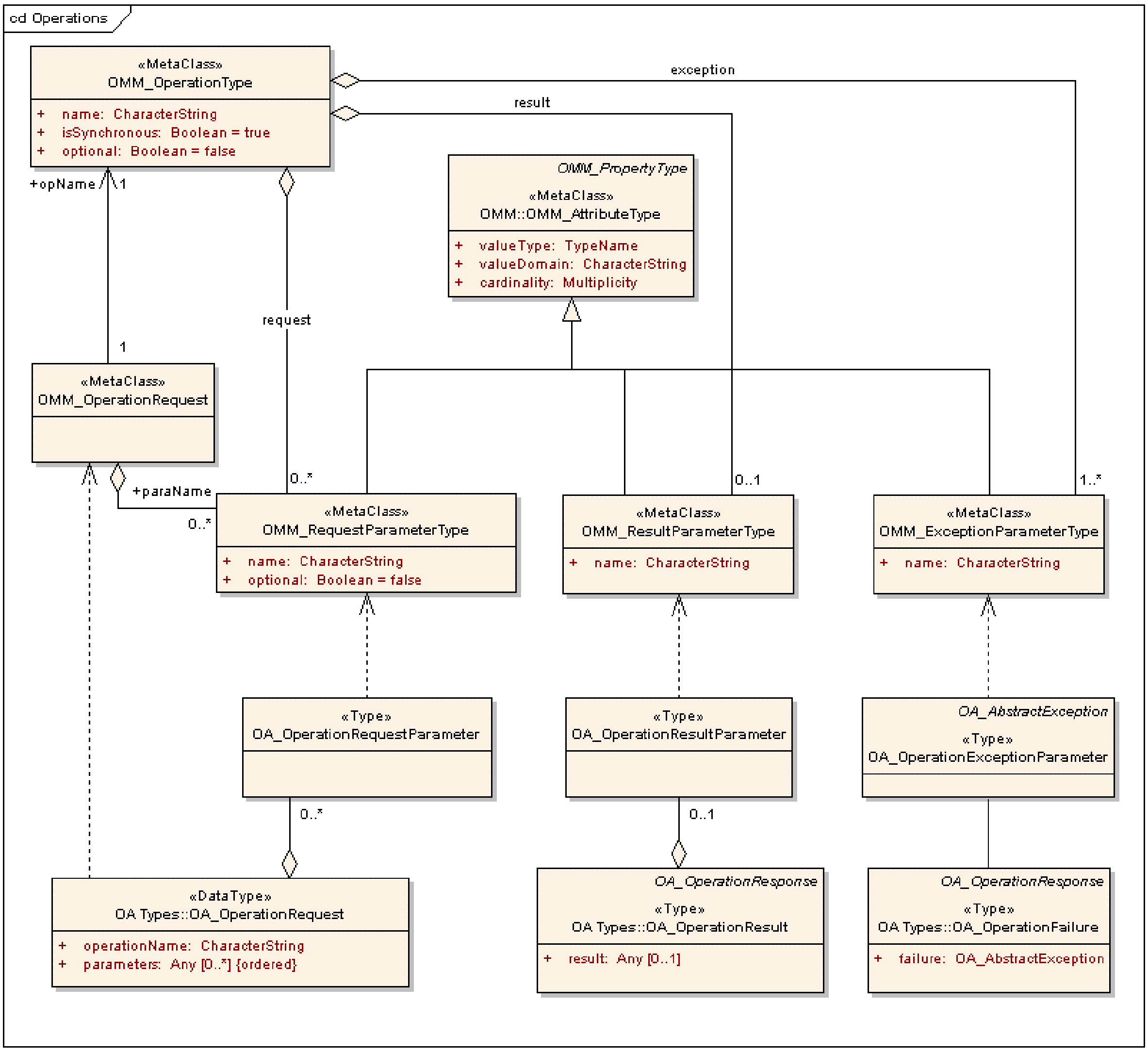


Figure 31: Model of OMM Operations and Parameter Types

9.2.5 Rules for ORCHESTRA Services

9.2.5.1 General Approach

The modelling process for ORCHESTRA Service Types shall obey the rules specified in the following sections. In this process, two cases are to be distinguished:

1. ORCHESTRA Service Types that are in a first step specified on a platform-neutral level, i.e. in addition to the mandatory abstract service description there are abstract specifications of all of their interface types and then, in a second step, are mapped to one or more platforms as specified in corresponding implementation specifications.
2. ORCHESTRA Service Types that are directly specified in an implementation specification without the delivery of abstract specifications of their SOA-RM Action Model in terms of abstract interface types in addition to the mandatory *ServiceCapabilities* interface type.

Note 1: The implementation specification is dependent on the platform specification that contains the mapping rules from and to the abstract level. Thus, it is assured that an ORCHESTRA Service Type, even when just specified on a platform level, is compliant to the OMM.

Note 2: Whether it is possible to automatically derive from a given SOA-RM Action Model of an implementation specification an abstract specification of a corresponding interface such that this distinction is not necessary will be investigated.

Rules:

1) For all ORCHESTRA Service Types an abstract description (i.e. an instance of OMM_ServiceAbstractDesc) shall be provided.

2) For all ORCHESTRA Service Types that are categorised as OA Services an abstract specification of all of their interface types (i.e. an instance of OMM_InterfaceAbstractSpec) is mandatory.

3) For all ORCHESTRA Service Types that are categorised as OT Services and thus are part of an OAA, an abstract specification of all of its interface types is optional. It is strongly recommended to provide abstract interface specifications if

   - it is envisaged to submit the service specification to a standardisation organisation that is not fixed to a particular service platform (e.g. ISO or OGC),
   - parts of the specified functionality of the service type are expected to be re-used by other service types,
   - the foreseen lifetime of the service specification is expected to be above the usual innovation cycle of IT service infrastructure technology (around 5-10 years),
   - it is envisaged to provide at least two different implementation specifications according to the same service requirements (e.g. several service profiles for the same platform or the same service profile for different platforms).

9.2.5.2 Rules for ORCHESTRA Service Types

Rules:

1) An instance of OMM_ServiceType shall be implemented as a CLASS stereotyped as <<ServiceType>> (see OA_ServiceType) that defines an ORCHESTRA Service Type as a realisation of one or more interfaces (OA_Interface). The name of the CLASS corresponds to the service type name and shall be unique for all applications of the ORCHESTRA Architecture.

   Note: RM-OA version 3 will provide rules how the uniqueness of service type names can be achieved.

2) An instance of OMM_ServiceType shall at least realise the interface type *ServiceCapabilities* as specified in the OA Basic Service (see section 9.6.1).

9.2.5.3 Rules for Abstract Descriptions of ORCHESTRA Services

Rules:

1) An instance of OMM_ServiceAbstractDesc shall be implemented as a CLASS stereotyped as <<Specification>> (see OA_ServiceAbstractDesc). It shall describe the purpose and scope of the service type in a human readable form and shall provide an overview about the interface types supported by the service type. If no other form is requested by a project environment, the RM-OA Service Description Framework as introduced in section 9.4 shall be used.

   Note: The link of this description to meta-information models for services (i.e. OAS-MI for services) will be investigated in a future version of the RM-OA.

2) An instance of OMM_ServiceAbstractDesc shall refer to one or more instances of OMM_InterfaceAbstractSpec.

Note: The abstract description of an ORCHESTRA Service Type may also be combined with the abstract specification of the associated interface types (see section 9.2.6) in one “abstract service specification”. The service types that are described in the RM-OA Service Viewpoint are specified like that, see (ORCH-AbstrServ 2007).

9.2.6 Rules for the Specification of Interface Types

Rules:

1) An instance of OMM_InterfaceType shall be implemented as a CLASS stereotyped as <<Interface>> (see OA_Interface) that defines the set of operations implemented as instances of OMM_Operation.

2) An instance of OMM_InterfaceType shall be specified in UML 2.0.

3) An instance of OMM_InterfaceType (acting in the role of a subtype) may only inherit operations from those instances of OMM_InterfaceTypes (acting in the role of supertypes) if these supertypes are marked by the tagged value <<supertype>.

   Note: The supertypes need not be specified in the same abstract specification (an instance of OMM_InterfaceAbstractSpec) as the subtype.

4) An instance of OMM_InterfaceType shall be contained in exactly one abstract specification of an interface type (an instance of OMM_InterfaceAbstractSpec).

5) An instance of OMM_InterfaceAbstractSpec shall be implemented as a CLASS stereotyped as <<Specification>> (see OA_InterfaceAbstractSpec). It shall provide an overview about the interface type both in a human-readable form and in a formal specification (see rule 4) above). If no other form is requested by a project environment, the specification template applied in (ORCH-AbstrServ 2007) shall be used.

6) If an interface type contains stateful operations, i.e. if the service implementing the interface must maintain the value of a state attribute beyond the duration of the processing of an operation request, the interface specification shall contain a state diagram that describes the meaning of each state and the conditions for the transitions between the states.

9.2.7 Rules for the Specification of Operation Types

Rules:

1) An instance of OMM_OperationType shall be implemented as OPERATION of a class stereotyped as <<Interface>> (see OA_Interface) with the following properties:

   - The associated request parameters of an operation type (see instances of OMM_RequestParameterType) shall be implemented as parameter(s) of the interface operations.
   - The associated result parameters of an operation type (see instances of OMM_ResultParameterType) shall be implemented as return type of the interface operations.

2) The set of request parameters of an operation type (i.e. instances of OMM_RequestParameterType) may be summarised in one instance of OMM_OperationRequest

and implemented as a CLASS stereotyped as <<Type>>. This is at least required in the following cases:

- if the operation is to be called by means of the generic *invoke* operation of the *SynchronousInteraction* or *AsynchronousInteraction* interface type specified in section 9.6.1. See also the corresponding rules in section 9.2.9.
- if one of the request parameters has to be specified as optional parameter (see rule 3) of section 9.2.8).

3) If an instance of OMM_OperationType may be omitted in the mapping to the SOA-RM Action model (SOA-RM 2006) of an implementation specification of an ORCHESTRA Service, the corresponding operation shall be marked with a stereotype <<optional>> in the class stereotyped as <<Interface>>.

Note: An instance of OMM_Operation that is not marked with a stereotype <<optional>> is considered to be a mandatory operation. This means it shall be mapped to a corresponding action in the implementation specification. This is the default case.

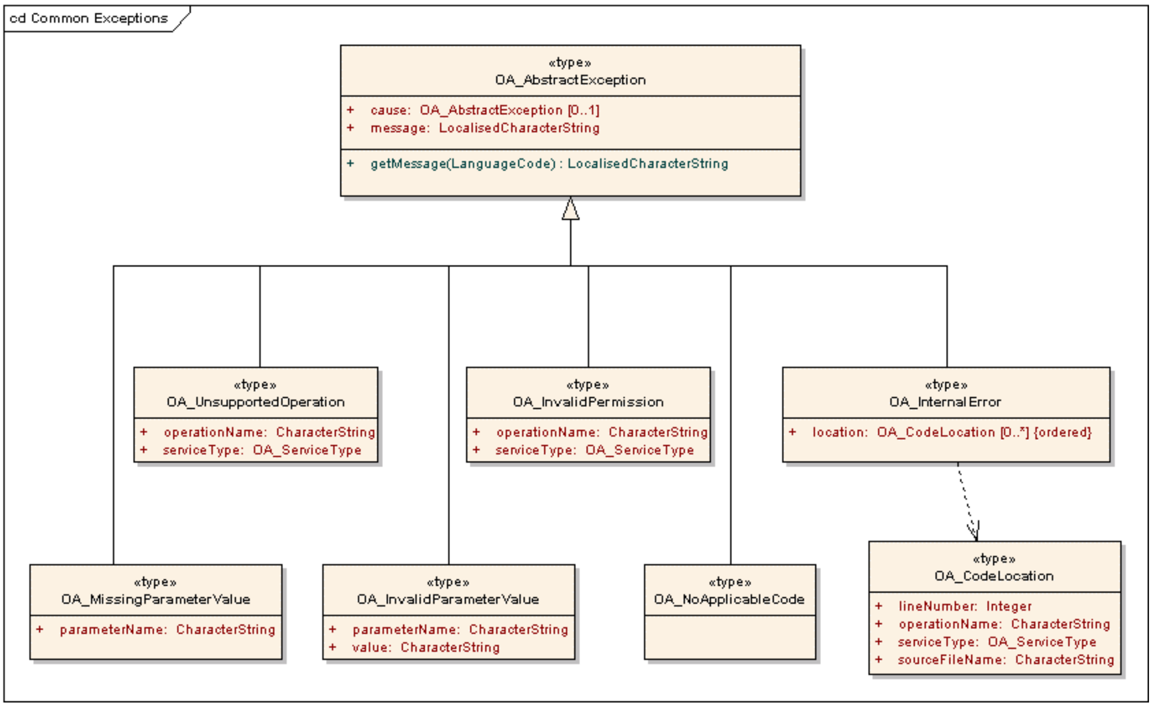

cd Common Exceptions
«type»
OA_AbstractException
+ cause: OA_AbstractException [0..1]
+ message: LocalisedCharacterString
+ getMessage(LanguageCode) : LocalisedCharacterString
«type»
OA_UnsupportedOperation
+ operationName: CharacterString
+ serviceType: OA_ServiceType
«type»
OA_InvalidPermission
+ operationName: CharacterString
+ serviceType: OA_ServiceType
«type»
OA_InternalError
+ location: OA_CodeLocation [0..*] {ordered}
«type»
OA_MissingParameterValue
+ parameterName: CharacterString
«type»
OA_InvalidParameterValue
+ parameterName: CharacterString
+ value: CharacterString
«type»
OA_NoApplicableCode
«type»
OA_CodeLocation
+ lineNumber: Integer
+ operationName: CharacterString
+ serviceType: OA_ServiceType
+ sourceFileName: CharacterString

Figure 32: Specification of Exception Types

9.2.8 Rules for the Specification of Parameter Types

Rules:

1) An instance of OMM_RequestParameterType representing one request parameter of an operation shall be implemented as a CLASS stereotyped as <<Type>> (see OA_OperationRequestParameter in Figure 31).

2) An instance of the OMM_RequestParameterType shall obey the rules for the instances of OMM_AttributeTypes as specified in section 8.8.7.

Note: This rule means that the data type of a request parameter is either a basic data type (see section 8.7.2.2) or a class with a valid stereotype (e.g., <<feature type>>).

3) If at least one instance of OMM_RequestParameterType as part of an operation type is to be specified as optional parameter, an instance of OMM_OperationRequest shall be implemented

as a class stereotyped by <<DataType>> that contains all request parameters as ATTRIBUTE whereby the optional request parameters shall have the cardinality [0..1] or [0..n].

4) An instance of OMM_ResultParameterType representing a result parameter of an operation (i.e. a normal response) shall be implemented as a CLASS stereotyped as <<Type>> (see OA_OperationResultParameter in Figure 31).

5) An instance of OMM_ResultParameterType shall obey the rules for the instances of OMM_AttributeTypes as specified in section 8.8.7.

   Note: This rule means that the data type of a result parameter is either a basic data type or a class with a valid stereotype (e.g., <<feature type>>).

6) An instance of OMM_ExceptionParameterType representing an exception parameter of an operation (i.e. a failure response) shall be implemented as a CLASS stereotyped as <<Type>> (see OA_OperationExceptionParameter in Figure 31). It shall be derived from the CLASS OA_AbstractException as specified in Figure 32.

7) An instance of OMM_ExceptionParameterType shall re-use the exception types that are pre-defined by the OA Basic Service (see section 9.6.1 and the specification of the exception types in UML in (ORCH-AbstrServ 2007)) if the semantics of these exception types fit the needs of the operation type.

8) An instance of OMM_OperationType together with its related instances of OMM_RequestParameterType representing an operation with its request parameters shall be implemented by a CLASS stereotyped as <<DataType> (see OA_OperationRequest in Figure 31). The operation request shall be sent either within a synchronous interaction, which is the default case, or within an asynchronous interaction.

   Note: The interfaces of a synchronous or asynchronous interaction are specified in the OA Basic Service (see section 9.6.1). Rules for their application are given in section 9.2.9.

9) An instance of OMM_ResultParameterType representing an operation result parameter shall be implemented by a CLASS stereotyped as <<Type> (see OA_OperationResult in Figure 31). The operation result is received within a synchronous or asynchronous interaction depending on the interaction mode of the preceding operation request (see rule 8) above).

10) An instance of OMM_ExceptionParameterType representing an operation exception parameter shall be implemented by a CLASS stereotyped as <<Type> (see OA_OperationFailure in Figure 31). The operation exception is received within a synchronous or asynchronous interaction depending on the interaction mode of the preceding operation request (see rule 8) above).

9.2.9 Rules for the Service Mapping to a given Platform

9.2.9.1 General Approach

The process of the service mapping to a given platform is illustrated by the conceptual model in Figure 33.

Rules:

1) For each service type that is considered to be available for a given platform an implementation specification for this platform according to rules of section 9.2.11 shall be available.

2) The process of mapping an abstract specification to an implementation specification shall be documented in a service mapping specification, i.e. an instance of OMM_ServiceMappingSpec (see rule 4) below).

3) The service mapping specification shall be a section in the ORCHESTRA Implementation Specification. Furthermore,
    - It shall define the mapping of each operation type and parameter type specified in abstract interface specifications to the SOA-RM Action Model of the ORCHESTRA service on platform level.
    - The mapping shall comprise both the static part (signature) as well as the behaviour of the operation.

Note: See (ORCH-ImplServ 2006) of an example of such a service mapping specification for the ORCHESTRA Web Services platform.

4) The service mapping specification shall consider the following cases:

    - Case 1: Service Profile, an instance of OMM_ServiceProfile, if the SOA-RM Action Model of the implementation specification comprises a subset of the interface operations specified in the abstract specification of an ORCHESTRA Service Type whereby the structure and the semantics of the interface operations and the SOA-RM Action Model are identical. Rules for a Service Profile are given in section 9.2.9.2.

Note: Other cases (such as ontology-based service mediation) may be considered in future versions of the RM-OA, e.g. if the semantics of the interface operations on the abstract level and the SOA-RM Action Model on the platform level are similar but not identical.

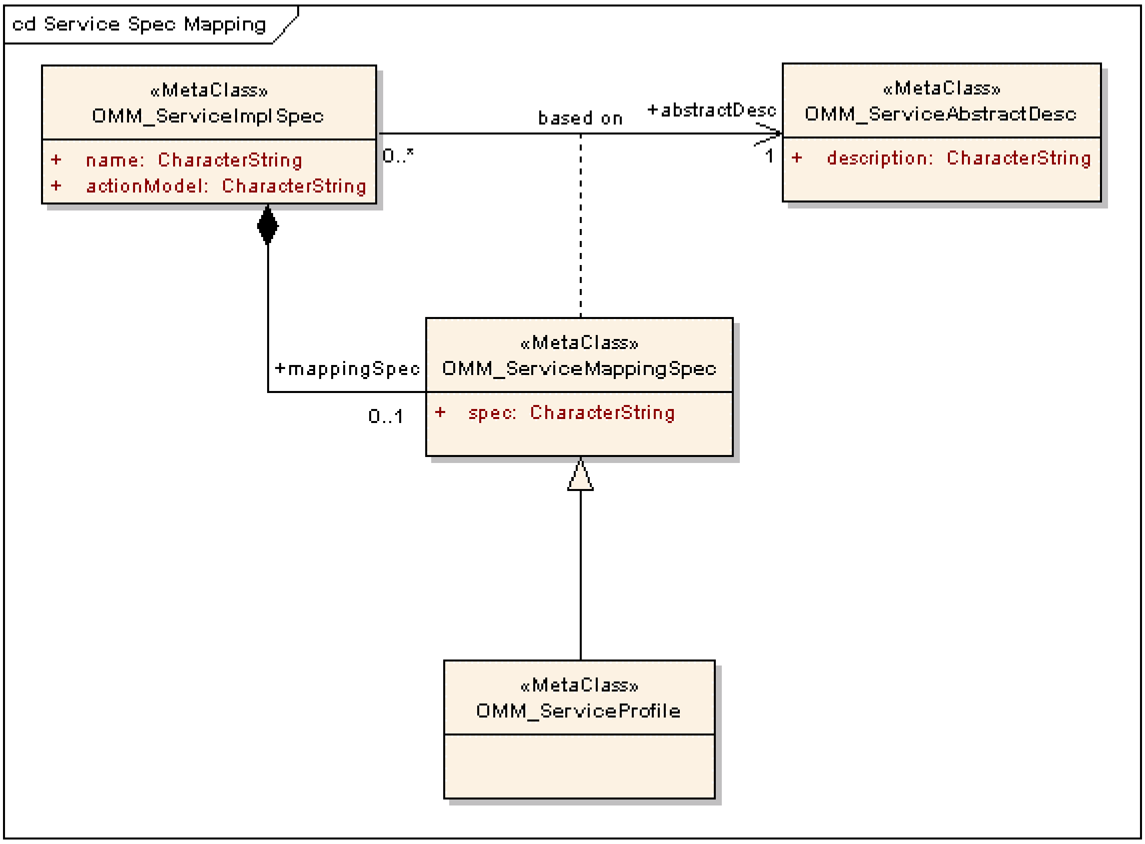


Figure 33: Structure of the Service Mapping in the OMM

9.2.9.2 Rules for Service Profiles

Rules:

1) All operations of all interfaces that are not marked as “optional” (see rule 3) of section 9.2.7) shall be mapped to an implementation specification. An operation shall be represented in the

respective SOA-RM Action Model according to one of the following cases:

- It is mapped to exactly one action invoked against a service specified in an implementation specification. The action invocation is performed in a synchronous interaction and shall be semantically identical to the operation call of the abstract specification.
- It is mapped to the SOA-RM Action Model that provides the *SynchronousInteraction* or *AsynchronousInteraction* interface type for the given platform if the corresponding functionality has been specified for this platform (see rule 2) of section 9.2.10). In this case, the following rules apply respectively for the chosen interaction mode.

2) For all operations of all interfaces that are marked as "optional" (see rule 3) of section 9.2.7) the following cases are possible:
    - They may be omitted in the SOA-RM Action Model of the implementation specification.
    - They may be mapped to optional actions in the SOA-RM Action Model of the implementation specification.
    - They may be mapped to mandatory actions in the SOA-RM Action Model of the implementation specification.

3) A parameter of an operation that is not marked as "optional" in the abstract specification (see rule 3) of section 9.2.8) shall be syntacticly mapped to exactly one parameter of the action invocation. The parameter semantics shall be identical.

4) For all parameters of an operation that are marked as "optional" (see rule 3) of section 9.2.8) the following cases are possible:
    - They may be omitted in the action of the implementation specification.
    - They may be set to a constant value for the action in the implementation specification.
    - They may be mapped to optional action parameters in the implementation specification.
    - They may be mapped to mandatory action parameters in the implementation specification.

Note 1: The meaning of the expression "is semantically identical" is that the "real-world effect" of an action (see OASIS RM-SOA, 2005) is identical.

Note 2: It may turn out that "semantically identical" mappings are not possible in all cases and a weaker definition is required. In this case, a further case in the service mapping rules will be introduced.

9.2.10 Rules for Platform Specifications

Rules:

1) An instance of OMM_PlatformSpec shall be implemented as a CLASS stereotyped as <<Specification>> (see OA_PlatformSpec). It shall describe the basic properties of the platform as specified in section 9.2.3.5.

    Note: A more refined discussion of the platform properties is provided in the RM-OA Technology Viewpoint, see section 10.

2) An instance of OMM_PlatformSpec shall contain or refer to implementation specifications of all interface types specified in the OA Basic Service (see section 9.6.1) for which a respective functionality shall be offered for this platform. The provision of an implementation specification of the *ServiceCapabilities* interface type is mandatory.

3) An instance of OMM_PlatformSpec shall observe the conformance guidelines given in section 4 of (SOA-RM, 2006).

4) The specification of the SOA-RM Information Model constraints for platform services shall include a specification of how the rules of the OMM Service Meta-model for request, result and exception parameters (see section 9.2.8) are fulfilled. This assures that the interactions between service providers and consumers are compliant to the OMM even in cases where the interfaces to ORCHESTRA services are not first specified on an abstract level according to the OMM and then mapped to the SOA-RM action model of a particular platform.

9.2.11 Rules for Implementation Specifications of ORCHESTRA Services

Rules:

1) An ORCHESTRA Implementation Specification of an ORCHESTRA Service Type, i.e. an instance of OMM_ServiceImplSpec, shall be provided according to the rules of the chosen (service) platform (see section 9.2.10).

2) An ORCHESTRA Implementation Specification of an ORCHESTRA Service Type shall be a document that is structured according to a template that fits the chosen platform and is part of an ORCHESTRA Implementation Specification for that platform.

3) If the functionality of the ORCHESTRA Service Type has been specified in terms of abstract interface types (i.e. instances of OMM_InterfaceAbstractSpec) in addition to the mandatory *serviceCapabilities* interface type, there must be an instance of OMM_ServiceMappingSpec (see section 9.2.9) that specifies the mapping process from the abstract to the implementation specification.

9.3 Functional Classification of ORCHESTRA Services

9.3.1 Overview

As part of the ORCHESTRA Architecture, ORCHESTRA Service Types are defined by the collection of the interface types that they support. As an interface type defines the externally visible behaviour, an ORCHESTRA Service Type is in fact defined by the functionality that it provides to the external world. The RM-OA classifies service types into service categories by discussing their functionality. The main service categories are ORCHESTRA Architecture Services (OA Services) and ORCHESTRA Thematic Services (OT Services):

- An OA Service provides a generic, platform-neutral and application-domain independent functionality.
- An OT Service provides an application domain-specific functionality built on top and by usage of OA Services and/or other OT Services.

Note 1: Here and in the following, the term “usage” means that a service may call operations of another service in order to provide the desired functionality. In this sense, the calling service depends on the other service. In the service specification it is stated if such a usage is mandatory or just recommended.

Note 2: The list of OA Services and OT Services as presented in the following section is the result of an intense analysis of the functional user requirements within the ORCHESTRA project.

Note 3: The granularity for the services is oriented at the functional coherency of the service operations and the type of information (e.g. feature types, meta-information) that is managed by the service.

9.3.2 OA Services

OA Services are further classified into two sub-categories:

- OA Info-Structure Service: These are OA Services that are required to operate an OSN in the sense that these services play an indispensable role in the operation of an OSN depending on its required characteristics (see section 11.1). An example of such a role may be that at least one OSI of such a service must exist in one OSN environment (e.g. for the Catalogue Service, see section 9.6.6). Other examples are the various access services which shall be used when a feature of the respective type is accessed in an OSN (e.g. a document shall be accessed by usage of the Document Access Service, see section 9.6.4).
- OA Support Service: These are OA Services that support the provision of OA Info-Structure

Service functionality (as an implementation option) or facilitate the operation of an OSN, e.g. providing an added value by combining them with the usage of OA Info-Structure Services.

These together comprise the generic information infrastructure (info-structure) of the RM-OA. The OA Services thus provide the functional basis for application domain-specific functionality. OA Services themselves do not address any specific thematic application domain, nor do they impose any structure on the OT Services.

Note that OA Services may themselves use other OA Services. Furthermore, OT Services may use both OA Info-Structure Services and OA Support Services in order to fulfil a given functionality.

This functional classification is illustrated in Figure 34.

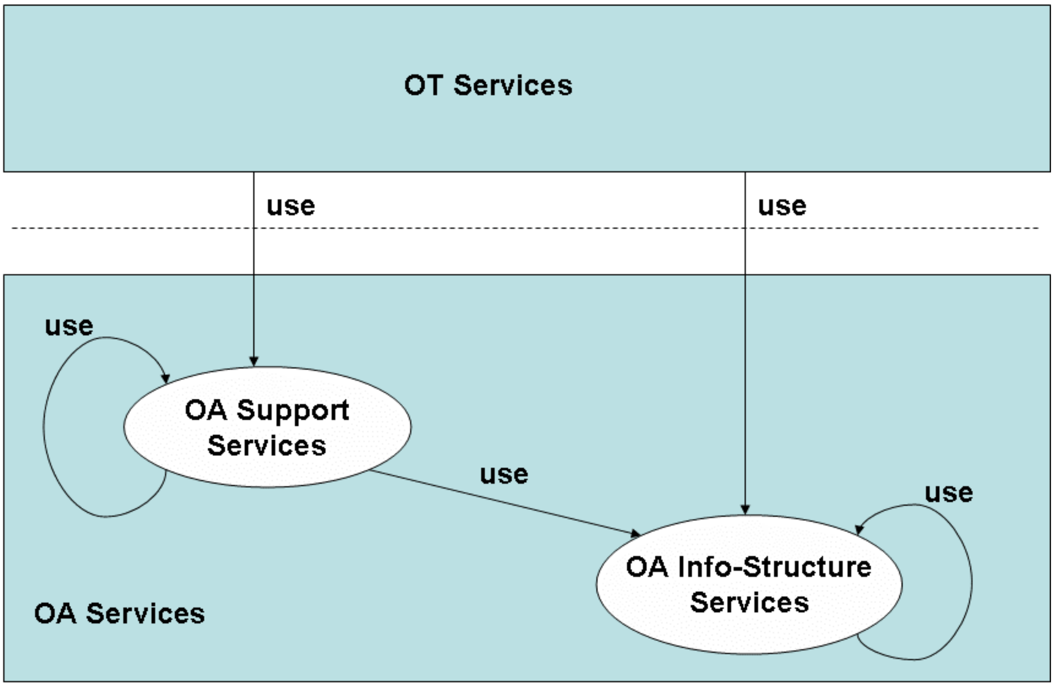


Figure 34: Functional classification of ORCHESTRA Services

Table 5 shows the current list of service types categorised as OA Services in alphabetic order within the sub-categories. The last column indicates if a corresponding abstract specification of the service type and its containing interface type is currently available in (ORCH-AbstrServ 2007).

Note 1: Basic functionality that may, or even shall, be offered by all OA and OT Services with well-defined interfaces is collected in an "abstract service type" called OA Basic Service. This service type is abstract as there is no meaningful instance of such a service type. However, it is kept in the table as the same description and specification techniques are used in order to describe its functionality.

Note 2: The categorisation of an OA Service as either an OA Info-Structure service or an OA Support service is derived from the idea that essential characteristics of an OSN are discovery and access to resources residing in source systems, whereby access means read and/or write access, and, in addition, a possibility of monitoring the running services. The rationale for this selection is a compromise between, on the one hand, keeping the requirements for a service network to be "OSN-compliant" as small as possible and, on the other hand, providing a powerful service infrastructure for a broad range of ORCHESTRA Applications. In this sense, support for transformations of any kind or automatic generation of meta-information is considered to be "OA Support" as it is not required for all ORCHESTRA Applications running in a rather homogeneous environment. See a more refined discussion about OSN characteristics in section 11.1.

Note 3: The column "ISO 19119 Service Taxonomy" provides just a hint of the position of the OA Service in the ISO 19119 Service Taxonomy. Note that GeoModel/InfoManagement here stands for Geographic Model/Information Management Services.

Service Type Name	Service Category	ISO 19119 Service Taxonomy	Section	Abstract Service Specification (ORCH-AbstrServ 2007)
Authentication Service	OA Info-Structure	GeoModel/InfoManagement	9.6.10	yes
Authorisation Service	OA Info-Structure	GeoModel/InfoManagement	9.6.9	yes
Catalogue Service	OA Info-Structure	GeoModel/InfoManagement	9.6.6	yes
Document Access Service	OA Info-Structure	GeoModel/InfoManagement	9.6.4	yes
Feature Access Service	OA Info-Structure	GeoModel/InfoManagement	9.6.2	yes
Map and Diagram Service	OA Info-Structure	GeoModel/InfoManagement	9.6.3	yes
Name Service	OA Info-Structure	GeoModel/InfoManagement	9.6.7	to be provided
OA Basic Service	OA Info-Structure	---	9.6.1	yes
Sensor Access Service	OA Info-Structure	GeoModel/InfoManagement	9.6.5	yes
Service Monitoring Service	OA Info-Structure	GeoModel/InfoManagement	9.6.11	to be provided
User Management Service	OA Info-Structure	GeoModel/InfoManagement	9.6.8	yes
Annotation Service	OA Support	GeoModel/InfoManagement	9.7.3	yes
Coordinate Operation Service	OA Support	Geographic Processing Services	9.7.1	yes
Document Indexing Service	OA Support	GeoModel/InfoManagement	9.7.4	to be provided
Format Conversion Service	OA Support	GeoModel/InfoManagement	9.7.5	yes
Gazetteer Service	OA Support	GeoModel/InfoManagement	9.7.2	yes
Knowledge Base Service	OA Support	GeoModel/InfoManagement	9.7.10	yes
Ontology Access Service	OA Support	GeoModel/InfoManagement	9.7.7	to be provided
Query Mediation Service	OA Support	GeoModel/InfoManagement	9.7.9	to be provided
Schema Mapping Service	OA Support	GeoModel/InfoManagement	9.7.6	yes
Service Chain Access Service	OA Support	Workflow/Task Management Services	9.7.11	to be provided
Thesaurus Access Service	OA Support	GeoModel/InfoManagement	9.7.8	to be provided

Table 5: List of OA Services

9.3.3 OT Services

OT Services provide application domain-specific functionality. However, both within and between different application domains, high-level functions that have a generic nature may be identified. These services are inside the scope of the RM-OA as a generic architecture and area defined as follows:

- OT Support Service: generic service that facilitates the development or interactive composition of thematic functionality.

The application domain of environmental risk management is taken as an informative example of further sub-categories of OT Services, although outside the scope of the RM-OA. Here, the ORCHESTRA project provides dedicated OT Services according to the following structure:

- OT Risk-neutral Service: service specific to the risk management domain that facilitates the development or interactive composition of risk-neutral risk management functionality.
- OT Risk-specific Service: service specific to a specific risk management domain (e.g. earthquakes, forest fires, flood, systemic risks) that facilitates the development or interactive composition of risk-specific risk management functionality.

All OT Services may use and combine the OA Services in order to fulfil their thematic function. As an example, the service sub-categories for the application domain of environmental risk management are illustrated in Figure 35.

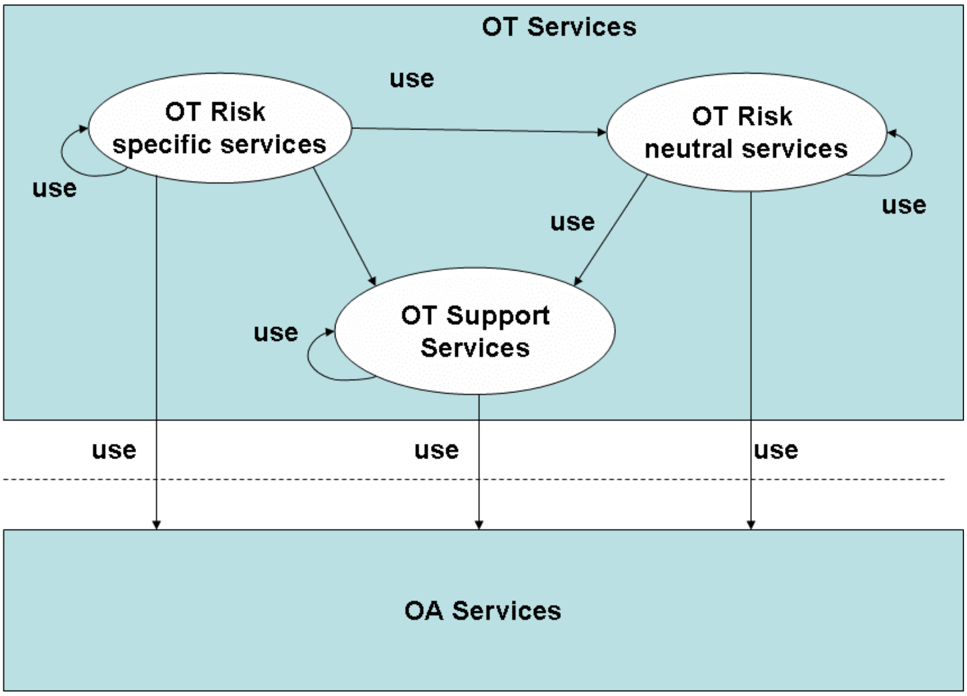


Figure 35: Example of OT Service sub-categories for the application domain of Environmental Risk Management

As an example, Table 6 shows the current list of OT Support Services for the application domain of Environmental Risk Management. The column “ISO 19119 Service Taxonomy” provides a hint of the position of the OA Service in the ISO 19119 Service Taxonomy.

A candidate list of required OT Services in the domain of risk management may be found in (ORCH-D2.4.2 2005).

Note: The current list of OT Support Services is a result of functional user requirements although these service types are not yet specified on a detailed level. However, they are kept for documentation and traceability purposes. They will be redfined once there are clear requirements from a pilot or a customer.

Service Name	Service Category	ISO 19119 Service Taxonomy	Section
Processing Service	OT Support	Geographic Processing Services	9.8.1
Simulation Management Services	OT Support	Geographic Processing Services	9.8.2
Calendar Service	OT Support	Workflow/Task Management Services	9.8.6
Communication Service	OT Support	Workflow/Task Management Services	9.8.5
Project Management Support Service	OT Support	Workflow/Task Management Services	9.8.4
Reporting Service	OT Support	Workflow/Task Management Services	9.8.7
Sensor Planning Service	OT Support	Workflow/Task Management Services	9.8.3

Table 6: List of OT Support Services for Environmental Risk Management

9.3.4 Human Interaction Components

The ORCHESTRA Services as categorized above do not provide an interface to a human user but rather to a software component requesting an operation at the service interface. The provision of such user interfaces is to be provided by so-called Human Interaction Components.

Human Interaction Components are software components that provide the (usually graphical) user interface (GUI) of an OA Service or OT Service. As such, the specification of such components is outside the scope of the RM-OA, i.e. no service description will be provided.

9.4 Relationship of the ORCHESTRA Service Types to INSPIRE

The ORCHESTRA Architecture follows an iterative design approach. The major iteration cycles that are currently foreseen are described in section 6.2.3. The focus of the current version 2 of the OA is to support syntactic interoperability, in particular but not exclusively for spatial services, such that the OA may contribute to the specification of the INSPIRE network services as outlined in section 6.2.2.3.

The following table provides an overview of which of the ORCHESTRA Interface and Service Types may contribute to which INSPIRE network services. This linkage to the INSPIRE requirements is preliminary as the work of the INSPIRE drafting team for network services has not yet been finalised and a detailed definition on the INSPIRE Network Services is not yet available.

INSPIRE Network Services	ORCHESTRA Interface Type	Specified in ORCHESTRA Service Type	Comment
Discovery Services	CatalogueSearchInterface	Catalogue Service (see section 9.6.6)	The ORCHESTRA Catalogue Service is generic w.r.t. the usage of a specific meta-information model. The CS-W 2.0 ISO AP 19115/19119 as currently investigated by INSPIRE could be chosen as one example.
Upload Services	CataloguePublication and CatalogueCollection Interface	Catalogue Service (see section 9.6.6)	
View Services	MapService	Map and Diagram Service (see section 9.6.3)	INSPIRE just requires rendering in maps
Download Services	FeatureAccessService	Feature Access Service (see section 9.6.2)	To support the download of feature instances
	DocumentAccess	Document Access Service (see section 9.6.4)	To support the download of predefined datasets
Transformation Services	CoordinateOperation	Coordinate Operation Service (see section 9.7.1)	
	SchemaMapping SchemaMappingRepository	Schema Mapping Service (see section 9.7.6)	In case schema mapping remains in the scope of the INSPIRE Transformation Services.
“Invoke spatial data services” services	ProcessingService	Processing Service (see section 9.8.1)	OMM-Service (see section 9.2) may provide input to the specification of the INSPIRE service reference model mentioned in the INSPIRE description
	ServiceChainAccessService	Service Chain Access Service (see section 9.7.11)	

Table 7: Possible Contribution of ORCHESTRA Service Types to INSPIRE Network Services

9.5 Service Description Framework

A coarse description of the ORCHESTRA Services is provided in a textual format according to the following template. The detailed abstract specifications of the services are provided in (ORCH-AbstrServ 2007). These documents contain formal specification of the information objects that are referred to in the interface operations (e.g. parameter types).

Name	Name of the ORCHESTRA Service Type Convention: All individual words in the service type name are capitalized.
Standard Specifications	Reference to an abstract or a platform-specific service specification according to a standardisation organisation (e.g. ISO, CEN, W3C, OGC,…) or to important reference material that has been taken into account when describing the service, its interfaces or operations. In case there is no adequate reference the field is set to "no corresponding standard known"
Description	Human understandable description of the functionality provided by the ORCHESTRA service. The end of the description shall provide the following text: The <*name*> Service provides its functionality through the following interfaces: • *Interface1*: human understandable description of the purpose of interface 1 • … • *InterfaceN*: human understandable description of the purpose of interface N Note: If an interface is re-used from another ORCHESTRA Service Type description, the name of this service type shall be indicated in brackets in the interface definition below. The description of the used interface operations shall be adapted to the context of the using service. Convention: All words in the interface name are written together in italics without a blank in between. The first letter of the first word and all other words are written in upper case letters.
Interface *Interface1 (from << Name of an ORCHESTRA Service>*	
oper1	Human understandable description of the operation 1 of the interface. Only major input and output information shall be described, no individual request and result parameters. Note: All words in the service operation name are written together in italics without a blank in between. The first letter of the first word is lower case, all other words upper case.
…	
operN (optional)	Human understandable description of the operation n of the interface. Optional operations are to be marked by suffix (optional) after the operation name.
…	
Interface *InterfaceN*	
…	
Example usage	Description of an example usage scenario of the service, e.g. by the combination of several operation calls of the service or in combination with another ORCHESTRA Service.
Comments	Description of current restrictions or possible extensions and enhancements in future versions of the RM-OA.

Table 8: Service Description Framework

9.6 OA Info-Structure Service Descriptions

9.6.1 OA Basic Service

Name	OA Basic Service
Standard Specifications	• OASIS UDDI Version 3.0.2 Specification (http://uddi.org/pubs/uddi_v3.htm) • OASIS Web Services Notification (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn) • OASIS Business Transaction Protocol (BTP) 1.0, Committee Specification (http://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf) • OGC 05-008c1 Web Services Common Specification V1.0 • WS-I Basic Profile V1 (http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html)
Description	The OA Basic Service provides descriptions of those behaviours for which a common architectural approach is required for all ORCHESTRA Services. It does so by defining abstract interfaces which may be extended or adapted according to the context of a specific ORCHESTRA Service. In addition the OA Basic Service specifies predefined exception types to be used by all ORCHESTRA Services. As the OA Basic Service only provides abstract interfaces for all ORCHESTRA Services, there will be no distinguished OSI of the OA Basic Service, but an implementation of the specified interfaces as part of other OSIs. Note that only the *ServiceCapabilities* interface is mandatory for all ORCHESTRA Service Types. All other interfaces are optional and shall be used by a service type if a corresponding behaviour is required. The OA Basic Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Definition of a uniform way to get a self-description of an OSI by means of so-called capabilities. The capabilities form service meta-information which can be used for various purposes like, for example, service discovery and service invocation. This interface is a mandatory interface and shall be implemented by all ORCHESTRA Services. • *SynchronousInteraction:* Definition of a uniform way to request synchronous execution of a service operation. Synchronous execution of an operation means that the client requests operation execution and then waits until the operation provider has finished operation execution and returns a response. Such a response may either contain an operation result value (which also may be empty) or may be an indication of a failure which is modeled as exception. • *AsynchronousInteraction:* Definition of a uniform way to request asynchronous execution of a service operation, e.g., for operations which are time-consuming or deliver results periodically. Asynchronous execution of an operation means that the client requests operation execution but does not wait until the operation has finished. Instead, the client may execute other tasks while the operation is running. However, in most cases the client wants to be notified when the operation terminates in order to get its results. In addition, when executing an operation asynchronously the client should be able to abort operation execution. • *TransactionInterface:* In a system that supports multiple users,

	synchronization of access to resources must be assured. This is an especially important requirement in the context of changing resources (write access), otherwise the consistency of the state of the system and its data cannot be guaranteed. Obviously not all services need to support transactions but if they do care must be taken. In order to guarantee a great amount of flexibility, the *TransactionInterface* allows numerous different types of transactions, e.g. transactions that support the properties of atomicity, consistency, isolation, and durability (ACID), OASIS business transactions that relax some of the ACID properties, operation batching, 'best try' transactions and sub-transactions.
Interface *ServiceCapabilities*	
get Capabilities	Informs the client about the capabilities of an OSI. This operation takes into account that in addition to capabilities that are common to all ORCHESTRA Services (referred to as common capabilities) an ORCHESTRA Service may provide a specific set of capabilities (referred to as specific capabilities). Furthermore, this operation allows the capabilities to be delivered according to different service meta-information schemas.
Interface *SynchronousInteraction*	
invoke	Executes an operation synchronously and returns the operation response.
Interface *AsynchronousInteraction*	
invokeAsync	Starts asynchronous execution of an operation. The *invokeAsync* operation returns immediately with an identifier (invocation ID) representing the asynchronous execution. In order to receive notifications a reference to a callback interface can be provided.
abort	Aborts execution of a previously invoked asynchronous operation identified by its invocation ID.
notify	Passes a notification to the callback interface provider.
Interface *TransactionInterface (from OA Basic Service)*	
createAcid Transaction	Creates a new ACID transaction at the service
create Business Transaction	Creates a new business transaction at the service
createSubAcid Transaction	Creates a new sub ACID transaction at the service.
createSub Business Transaction	Creates a new sub business transaction at the service.
setImplicit Commit	Sets the implicit timeout action for the specified transaction.
setRollback OnFailure	Sets the default failure action for the specified transaction
setLockOwner	Sets the resource lock owner for resources allocated by this transaction.
start Transaction	Starts an existing transaction at the service
tryCommit	Tries to commit the transaction without rolling back if the commit failed.
commit Transaction	Makes all changes made during the transaction permanent. Also releases all locks that have been acquired (if any) during the transaction.

abort Transaction	Revokes all chances made during the transaction
suspend Transaction	Suspends the transaction environment. All operations that are invoked at the service are carried out outside the transaction environment. This does not free any acquired locks.
resume Transaction	Set the specified transaction as the currently active transaction. This does not free any acquired locks.
getActive Transaction	Retrieves the transaction ID of the (most inner, if sub transactions are supported) currently active transaction.
add Transactions	Adds a number of transactions as children to the specified transaction.
remove Transactions	Removes a number of child-transactions from the specified transaction.
Example usage	The OA Basic Service contributes to a consistent description of the same or similar functionality of ORCHESTRA Services. It helps the developer of ORCHESTRA Applications to provide generic functions to the end-users or system users. Furthermore, it will help in defining a common framework for service discovery and access.
Comments	The contents of the service meta-information are defined as part of the specification of the OAS-MI for services in Annex B1 of the RM-OA.

Table 9: Description of the OA Basic Service

9.6.2 Feature Access Service

Name	Feature Access Service
Standard Specifications	• ISO/IEC 9075 Information technology -- Database languages -- SQL • ISO 19109:2005 Geographic information -- Rules for application schema • ISO 19125-1:2004 Geographic information -- Simple feature access -- Part 1: Common architecture • ISO 19125-2:2004 Geographic information -- Simple feature access -- Part 2: SQL option • ISO/DIS 19136 Geographic information -- Geography Markup Language (GML) • OGC 99-050 Simple Features Implementation Specification for OLE/COM V1.1 • OGC 99-054 Simple Features Implementation Specification for CORBA V1.0 • OGC 03-105r1 Geography Markup Language (GML) Encoding Specification V3.1.1 • OGC 04-094 Web Feature Service (WFS) Implementation Specification) V1.1 • OGC 04-095 Filter Encoding Implementation Specification V1.1 • OGC 05-076 Web Coverage Service (WCS) Implementation Specification (Corrigendum) V1.0.0 • OGC 05-126 Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture V1.1.0 • OGC 05-134 Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option V1.1.0

Description	The Feature Access Service allows interoperable read and write access on feature instances available in an OSN. Furthermore, the Feature Access Service provides an interface that may be inherited by more specific access services (e.g., sensor access service) using interface inheritance. The Feature Access Service offers information about: • The feature types it is capable to provide. • The supported encoding(s) to transfer requested or submitted feature data. • The query language and mechanism for filtered feature access. Features provided by the Feature Access Service are instances of a certain feature type defined in an ORCHESTRA Application Schema (OAS), which again is an instantiation of an OMM_FeatureType (see section 8.7.2). This means that the Feature Access Service only permits access to information which is represented through feature types according to the rules of the ORCHESTRA Meta-Model (OMM). Whether information is remodelled on-the-fly by a software component or whether the features are actually stored in a feature store is not crucial for the Feature Access Service. Seen from the interface, the feature representation is a black box and is not visible for clients. The Feature Access Service allows queries to select certain features based on their type, certain attribute values and their spatial and temporal extent. The selection statement is encoded using a query language that supports all these functionalities (e.g., SQL including spatio-temporal statements). By selecting and retrieving features, access to their attributes and operations is provided. Any Feature Access Service (and its possible profiles or possible inheriting interfaces) may support the update of existing feature instances, the creation of new feature and the deletion of existing features, and hence, in this case, it should also be transactional. It can also allow the creation, updates, and deletions of feature types. Feature instances and feature types are identifiable by a Unique Identifier (UID) that is unique with respect to at least one OSN (section 11.1.2). If a Feature Access Service is used to create a new feature instance or feature types it will also create an appropriate UID for this feature type or instance. Additionally, it is important to emphasize the requirements for Authorisation and authentication in order to support creation, deletion, and modification of feature and feature types (see section 7.5). The Feature Access Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *FeatureAccessService*: selection, creation, update and deletion of feature instances and feature types.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the requestor about the common and specific capabilities of a Feature Access Service instance. Examples of specific capabilities are the supported feature types, the encoding of feature type requests, the encoding of returned feature collections as well as the supported query language.
Interface *FeatureAccessService*	
getFeature Types	Gets a description (the schema) of given feature types serviced by an Feature Access Service instance in a specific encoding based on a query.
setFeature Types	Updates existing Feature Types matching a given query.
createFeature	Creates new Feature Types based on feature type descriptions.

Types	
deleteFeature Types	Deletes existing Feature Types matching a given query.
getFeatures	Retrieves features and their attributes matching a given query.
setFeatures	Updates existing features matching a given query.
createFeatures	Creates new features based on a feature collection and a given query.
deleteFeatures	Deletes existing features matching a given query.
Example usage	A client accessing this service wants to retrieve all feature instances of roads for a particular region. The Feature Access Service is passed a *getFeatures* request for the specified area and feature type. A response is generated containing all valid features. The features may be modified and submitted to the Feature Access Service as an update transaction (via the *setFeatures* operation).
Comments	As the RM-OA, in accordance with ISO 19123, considers coverages as subtypes of features, the Feature Access Service can also be used to access coverages.

Table 10: Description of the Feature Access Service

9.6.3 Map and Diagram Service

Name	Map and Diagram Service
Standard Specifications	• ISO/DIS 19128:2005 - Geographic information -- Web Map Server Interface • ISO/DIS 19136 Geographic information -- Geography Markup Language (GML) • OGC 02-070 Styled Layer Descriptor (SLD) Implementation Specification V1.0 • OGC 04-094 Web Feature Service (WFS) Implementation Specification) V1.1 • OGC 04-095 Filter Encoding Implementation Specification V1.1 • OGC 06-042 Web Map Service (WMS) Implementation Specification V1.3.0
Description	The Map and Diagram Service is a service that visualizes, symbolizes and enables geographic clients to interactively visualise geographic and statistical data. Its main task is to transforms geographic data (vector or raster) and/or numerical tabular data (e.g. census data, result of a statistical analysis) into a graphical representation using symbolization rules. The main output of this service is an image document, which can be either in raster (e.g. jpeg, png) or symbolized-vector format (e.g. SVG). The meaning of the image document (the output of this service) is a general reference map (visualization of geographic information), a diagram (visualization of statistical data) or a thematic map (visualization of the spatial distribution of one or more statistical data themes). This service enables the integration of extended Style Layer Descriptor (SLD) documents, which allows the definition of symbologies and symbolization rules at the feature level and allows also the integration of user data and remotely available data from other OA Services like the Feature Access Service (see section 9.6.1) The Map and Diagram Service provides the functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *MapDiagramService*: This interface allows a client to request and receive maps, diagrams and, optionally, information about the visualized features according to specifications, as well as to put/remove data and styles on the

	server for visualization.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the capabilities of a Map and Diagram Service instance. Examples of specific capabilities are a document containing, among others, a list of supported operations and predefined data layers available on the server with the corresponding layer information.
Interface *MapDiagramService*	
getMap	Returns a map of spatially referenced geographic and thematic information as an image document with the characteristics specified by the client application. The characteristics of the output image are specified by the *outputAttributes* parameter (image format, width, height, transparency, etc…) as well as the *mapAttributes* parameter (list of layers and their corresponding styles, coordinate reference system, global bounding box). Optionally, the map parameters can be provided using an SLD document.
getDiagram (optional)	Returns a diagram representation of numerical data as an image document with the characteristics specified by the client application. The characteristics of the output image are specified by the *outputAttributes* parameter (image format, width, height, transparency, etc…) as well as the *diagramAttributes* parameter (list of tabular data layers and their corresponding styles – diagram type, diagram characteristics). Optionally, the diagram parameters can be provided using an SLD document. This operation expects that the data to be rendered is in tabular format.
getLayerDescription (optional)	Returns a layer description document containing schema information for a layer: attribute names, types, units, statistical information when applicable (like value ranges, max, min etc.). This information is needed by clients in order to create their own styles and symbolization rules based on attribute values.
getLayerLegend (optional)	Returns a legend symbol (corresponding to a layer) as an image document with the characteristics specified by the client application. The characteristics of the output image are specified by the *outputAttributes* parameter (image format, width, height, transparency, etc…) as well as the *styledLayer* parameter (name of the layer for which the legend should be generated and its corresponding styles). If the styles corresponding to the layer are not available on the server, then the styles have to be defined and sent again by the client (optionally, also as a SLD document).
getFeatureInfo (optional)	Returns information about the features rendered in a certain point of a map or diagram layer as a document. The request must specify the attributes of the query point (x and y coordinates of the point in the image coordinate system, the layer name, and the number of features for which is expected to receive information) as well as a copy of the request that generated the image.
setLayer (optional)	Stores a new data layer on the server if the format of the sent layer data is supported (the supported formats for data input are advertised in the service capabilities). For this operation the following information must be defined: the layer (name, data, data format, minimum and maximum scale, etc…), the duration for which the layer will be stored and also if it will be visible or not for other users. The operation confirms the success of the request by sending back to the client a Boolean “TRUE”.
deleteLayer (optional)	Removes an existing data layer from the server. The operation confirms the success of the request by sending back to the client a Boolean “TRUE”.
setStyle (optional)	Stores a new style layer on the server. For this operation the style must be defined either by sending the symbology or by referencing a remotely available symbology. Furthermore, the duration for which the style will be stored and also if it will be visible or not for other users must be defined. The operation confirms the success of the request by sending back to the client a Boolean “TRUE”.
deleteStyle	Removes an existing style from the server. The operation confirms the success of

(optional)	the request by sending back to the client a Boolean "TRUE".
Example usage	A requestor accessing this service wants to create a map that shows the spatial distribution of the forest fire hazard zones (classified by the susceptibility level) with different colours. On top of this layer the requestor is interested to have the road network, the hydrological network, the urban areas and a diagram layer with bar charts showing the number of historical forest fire cases. The hazard zones and the historical forest fire data are accessible by means of a Feature Access Service and other layers are available on the server. The requestor now invokes a *getMap* operation by passing a styled layer descriptor document, which defines the location of the data and the symbolization corresponding for each layer. The response of the service will be a map provided in the requested format.
Comments	It is beyond of the scope of this service to provide a human interface like the geographic viewer in the human interaction services. On the other side, other map service instances, a geographic viewer or even a Web browser could act as a client to this service.

Table 11: Description of the Map and Diagram Service

9.6.4 Document Access Service

Name	Document Access Service
Standard Specifications	no corresponding standard known
Description	The Document Access Service supports access to documents of any type (textual documents, images,). A document is regarded as a specific kind of a feature type, therefore the Document Access Service is a specialisation of the Feature Access Service (see section 9.6.1) which inherits only feature-specific operations. Operations that manipulate feature types are not supported by this service, since the only feature type this service supports is OA_DocumentDescriptor. Compared with the Feature Access Service this service enables the conversion of documents and it guarantees that the returned feature instances are of type OA_DocumentDescriptor. Thus the Document Access Service acts as a specialisation of a Feature Access Service restricted to documents. The Document Access Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *DocumentAccessService*: Selection, creation, update and deletion of documents.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilites	Informs the client about the common and specific capabilities of a Document Access Service OSI. Examples of specific capabilities: a) the specific capabilities inherited from the Feature Access Service, b) information about supported document-encodings and MIME types.
Interface *DocumentAccess (from FeatureAccessService)*	
get Documents	Returns and optionally converts documents. This operation is an extension of the *getFeatures* operation of the *FeatureAccessService* interface. In addition to the *getFeatures* operation it supports the conversion of a document. The *getDocuments* operation retrieves features of the feature type

	OA_DocumentDescriptor. A query can be specified to retrieve certain documents that meet specific requirements.
create Documents	Creates new documents of type OA_DocumentDescriptor. This method is an extension of the *createFeatures* operation of the *FeatureAccessService* interface. Since this operation provides no additional functionality, the detailed abstract specification is omitted.
set Documents	Updates existing documents. This method is an extension of the *setFeatures* operation of the *FeatureAccessService* interface. Since this operation provides no additional functionality, the detailed abstract specification is omitted.
delete Documents	Removes existing documents. A query identifies which document to be deleted. This method is an extension of the *deleteFeatures* operation of the *FeatureAccessService* interface. Since this operation provides no additional functionality, the detailed abstract specification is omitted.
Example usage	After a search in a catalogue-service a found document can be retrieved by call of the *getDocuments* operation.
Comments	The currently provided interaction mode is synchronous interaction, in the future also asynchronous interaction will be supported if required.

Table 12: Description of the Document Access Service

9.6.5 Sensor Access Service

Name	Sensor Access Service
Standard Specifications	• OGC 06-009r1 – Sensor Observation Service Implementation Specification V0.1.5 (Request for Comments) • OGC 05-086r2 - Sensor Model Language (SensorML) Implementation Specification V1.0 (Draft proposed version)
Description	This service provides a basic interface for accessing sensor data, configuring a sensor and publishing sensor data. While the configuration and data publishing interfaces of the Sensor Access Service are optional, the ability to find a certain sensor and retrieve its values is mandatory. The Sensor Access Service is strongly related to the OGC Sensor Observation Service and therefore provides similar functionality. The Sensor Access Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *SensorAdministration:* Allows the client to add or remove sensors at the service and also change the descriptions of already existing sensors. • *SensorConfiguration:* Provides functionality that allows the client to configure a specified sensor (e.g.: adjust measurement range, position) • *SensorData:* Allows the client to query for sensors that provide a specific functionality/type of measurement and retrieve these measurements.

Interface *ServiceCapabilities*	
getCapabilities	Informs the client about the common and specific capabilities of a Sensor Access OSI. Examples of the specific capabilities are: - *configurationSupported*: Flag whether the *SensorConfigurationInterface* is implemented - *administrationSupported*: Flag whether the *SensorAdministrationInterface* is implemented - *configurationCacheSupported*: Flag whether the checkSensorConfiguration operation caches valid configurations. - *cacheTimeout*: Defines the duration of time after which a cached configuration will be deleted and the associated *OA_SensorConfigurationID* is invalid
Interface *SensorAdministration*	
addSensor	Add a new sensor with its specified description to the services.
updateSensorDescription	This operation can be used to change the description of an already existing sensor.
removeSensor	Removes the specified sensor from the service.
setSensorData	Publishes new sensor data at the service so that clients may retrieve it through an invocation of the *getSensorData* operation.
Interface *SensorConfiguration*	
getConfigurationSchema	Retrieves the configuration schema of the specified sensor. The schema describes format, mandatory and optional parts of a valid configuration for the specified sensor.
getSensorConfiguration	Retrieves the currently active configuration for the specified sensor.
setSensorConfiguration	Sets the configuration for the specified sensor.
Interface *SensorData*	
getSensor	Retrieves a list of identifiers of those sensors that match the specified requirements. These requirements are formulated in a query language. The query language is indicated in the service's capabilities.
getSensorData	Retrieves actual data (real measured or calculated/simulated data) of the specified sensor.
getSensorDataTypes	This operation returns the schemas for the data types that can be retrieved at this service.
Example usage	A sensor administrator wants to publish ozone measurement values so that an environmental authority can retrieve it and produce a report.
Comments	The Sensor Access Service is a very basic service that does not include planning of series of measurements or notifications. Notifications can be supported by implementing the *notify* operation of the *AsynchronousInteraction* interface of the OA Basic Service on the client side (see section 9.6.1).

Table 13: Description of the Sensor Access Service

9.6.6 Catalogue Service

Name	Catalogue Service
Standard Specifications	• OASIS UDDI Version 3.0.2 Specification (http://uddi.org/pubs/uddi_v3.htm) • OGC 04-021-r3 Catalogue Service Implementation Specification V2.0.1 (Class: Abstract Specification) • OGC 04-017r1 Catalogue Services – ebRIM (ISO/TS 15000-3) profile of CSW (CAT2 AP ebRIM) V0.9.1 (Class: Engineering Specification) • OGC 04-038r2 ISO19115/ISO19119 Application Profile for CSW 2.0 ((CAT2 AP ISO19115/19)) V0.9.3 (Status: Best Practices) • OGC 06-079r2 EO Application Profile for CSW 2.0 (Status: Pending) • OGC 06-131 EO Extension Package for ebRIM (ISO/TS 15000-3) Profile of CSW 2.0 (Status: Discussion Paper) Note: ORCHESTRA specifies a Catalogue Service that has been derived from the approach how meta-information is being handled in the OA (see section 8.4). Thus, the above standards have been considered, but the goal has not been to specify another variant of the OGC Catalogue Specification. The ORCHESTRA Catalogue Service does not define a meta-information schema by itself. The intention of the ORCHESTRA Catalogue is to provide a flexible service type which can be adapted to the particular purposes of the application environment.
Description	The Catalogue Service supports the ability to publish, query and retrieve descriptive information (meta-information) for resources (i.e. data and services), meta-information about ORCHESTRA Source Systems (just like meta-information for other ORCHESTRA services) and instances of feature types that are referred to by extensions of the OMM_FeatureType, such as documents, schemas, dictionaries, equations and models. The Catalogue Service is not tied to a particular schema of a meta-information standard (e.g. ISO 19115); instead it supports application schemas for meta-information (OAS-MI) that are designed according to the rules of the OMM. Due to independence from a specific meta-information standard the catalogue can be used to store meta-information about services and data according to the meta-information schema used in the catalogue. Therefore a catalogue instance can be used as a data catalogue, service registry or both if multiple meta-information types are used in the catalogue instance. The multilinguality of the catalogue is dependent on the multilingual capabilities of the meta-information schema used inside the catalogue. Meta-information entries in catalogues represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software. The Catalogue Service supports the discovery of registered resources within an information community and returns binding information that allows a user to locate and access the resource (e.g. an URI). The Catalogue Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *CatalogueSearchInterface:* The interface for search provides a means for searching information in the catalogue. The client asks the catalogue capabilities for the available catalogue entry types. Each entry type is associated with a meta-information type and its corresponding query languages. With this information the client can query the catalogue entry type with the appropriate query language. • *CataloguePublicationInterface:* The interface for publication is responsible for

	including, updating and deleting meta-information in the catalogue. It is pushing information into the catalogue. It provides operations for filling the catalogue. The needed meta-information could be created with some kind of meta-information editor, in which the user is specifying the meta-information about resources to be registered in the catalogue, or it could be collected through the collection interface. • *CatalogueCollectionInterface:* The collection interface provides operations, which are helpful for the automatic update of catalogue content in difference to the publication interface, which just fills the catalogue with given content. It is pulling meta-information into the catalogue. The operations in this interface should be able to be triggered from the outside of the catalogue and it should be possible to define a periodic update from the catalogue content. • *CatalogueNavigationInterface:* With the means of this interface, the user is looking for meta-information records managed by the catalogue by navigating from node to node. The search is driven by the catalogue itself: no query is performed. Note that the implementation of this interface makes the Catalogue Service a stateful service. • *AsynchronousInteraction (OA Basic Service):* Definition of a uniform way to request asynchronous execution of a service operation, e.g., for operations which are time-consuming or deliver results periodically. This interface is used by the *collectMetaInformationPeriodic* operation of the *CatalogueCollectionInterface*.
Interface *ServiceCapabilities*	
getCapabilities	Informs the requestor about the common and specific capabilities of a Catalogue Service instance. Examples of specific capabilities are the information about query languages and meta-information types used in the Catalogue Service instance.
Interface CatalogueSearchInterface	
search	Returns a list of identifiers for corresponding features, given a request expressed in a given query language.
getMeta Information	Returns associated meta-information instances, given some identifiers of features managed by the catalogue as returned by a previous search operation call.
getQuery Domain	Returns the domain of values that are applicable to a property of the meta-information type. This is used by catalogue clients. Using this operation by giving the parameters of interest, the client shall know what values (e.g. list of values, range of values) are allowed for a meta-information property.
getMeta Information Type	Returns the associated meta-information type, given a list of catalogue entry types managed by the catalogue.
Interface *CataloguePublicationInterface*	
createMeta Information	Pushes information into the catalogue. The task of this operation is to insert catalogue content into the catalogue. The operation receives the meta-information to be stored and returns information about the update of the catalogue.
setMeta Information	Updates the catalogue content. The operation receives the meta-information types to be stored and returns information about the update of the catalogue.
deleteMeta Information	Deletes catalogue content from the catalogue. The input is a constraint to identify the catalogue content, which needs to be deleted. The operation returns information about the update of the catalogue.
Interface *CatalogueCollectionInterface*	
collectMeta	Pulls meta-information into the catalogue. The operation receives one reference of

Information	a source of meta-information and a catalogue entry type. This catalogue entry type is the type in which the meta-information is going to be stored in the catalogue. The operation returns information about the update of the catalogue.
collectMetaInformationPeriodic (optional)	Receives one reference of a source of meta-information, the catalogue entry type and the time interval between two collections and a date to stop the collect. The catalogue entry type is the type in which the meta-information is going to be stored into the catalogue. The operation is processed periodically according to the given intervals and stores the resulting meta-information into the catalogue. The operation should be called asynchronously using the *AsynchronousInteraction* interface. The operation returns information about the update of the catalogue.
Interface *CatalogueNavigationInterface*	
getNavigationRoots	Returns the catalogue entries that can be used to start navigation inside the catalogue. If none is returned, no navigation will be possible.
getNavigationEdges	Returns all relationships that start from this node to other ones given an existing node in the catalogue. Each relationship is annotated by the kind of relationship, which adds some semantic information (e.g. broader, narrower, similar) to the link.
Interface *AsynchronousInteraction (from OA Basic Service)*	
invokeAsync	Starts asynchronous execution of the *collectMetaInformationPeriodic* operation of the *CatalogueCollectionInterface*. The *invokeAsync* operation returns immediately with an identifier (invocation ID) representing the asynchronous execution.
abort	Aborts execution of the previously invoked asynchronous *collectMetaInformationPeriodic* operation identified by its invocation ID.
notify	Passes a notification to the callback interface provider of the *CatalogueCollectionInterface*.
Example usage	A possible usage scenario of the catalogue is the usage of a catalogue for discovering maps and displaying them in a map viewer. The following steps need to be accomplished for this scenario: 1. The catalogue needs to be initialized with meta-information about the maps and a service capable of displaying the maps. The meta-information can be written into the catalogue using operation *createMetaInformation*. 2. The user performes a search for available maps on the catalogue using the *search* and *getMetaInformation* operations. 3. The user performes a search for an available map viewer, again using the *search* and *getMetaInformation* operations. 4. The user displays the maps in the map viewer, using the retrieved meta-information about the maps and the map viewer.
Comments	The abstract specification leaves the question of the meta-information creation open. It could be created by the user with the help of a meta-information editor or automatically either within the catalogue inside *collectMetaInformation* or with the usage of other means and services inside *collectMetaInformation*. The support of multi-linguality depends on the meta-information schema used in the catalogue. Meta-Information about data and services inside the scope of an OSN will be described with the help of the service capabilities.

Table 14: Description of the Catalogue Service

9.6.7 Name Service

Name	Name Service
Standard Specifications	• IETF RFC 1034 Domain Names - Concepts and Facilities • IETF RFC 1035 Domain Names - Implementation and Specification
Description	The objective of the Name Service is to encapsulate the implemented naming policy for service instances in an OSN. It is responsible for creating globally unique OSI names using a defined naming policy, e.g. by mapping between OSI names and corresponding platform-specific service identifiers. If the naming policy requires additional information to ensure uniqueness of names, e.g. an OSN name, then such information may be provided by configuration and shall be hidden at the service interface. A central Name Service instance for all OSNs is not required. Instead, there may be multiple Name Service instances, and each one may use a different naming policy, as long as global uniqueness of created names is guaranteed. If multiple Name Service instances are available within an OSN, they shall be related, i.e. each one can be used for name resolving within the OSN. It is possible to share a Name Service instance among multiple OSNs. Within an OSN that is based on multiple service platforms, a Name Service instance is available for each service platform and shall be used for name resolving within that platform. The Name Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *NameCreationAndResolution*: provides operations to create names and to resolve names given a platform-specific identifier (PSI) or vice-versa. • *NamingServiceLinkage*: provides operations to support the linkage between several Name Service instances.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of a Name Service instance. An example of a specific capability is the naming policy that is applied in the Name Service instance.
Interface *NameCreationAndResolution*	
registerService	An OSI is made known to the Name Service. The OSI is specified by its platform-specific service identifier (PSI). It is related to the current service platform, i.e. the platform on which the Name Service is based. The operation returns a globally unique name for the OSI according to the implemented naming policy. From that point on, name resolution is possible for that OSI name and PSI. If a PSI is not provided as input parameter, an OSI is registered which has not yet an assigned PSI. In that case, it is assumed that the Name Service itself assigns a PSI to the OSI This PSI can be retrieved later by means of the *getPSI* operation.
getPSI	Given an OSI name, the PSI of that OSI is returned if known to the Name Service. The PSI is used to access the OSI within the current service platform. It may therefore be a PSI of a service gateway, if the OSI is based on a different platform.
getName	Given the PSI of an OSI, the name of that OSI is returned if known to the Name Service.

Interface *NamingServiceLinkage*	
linkName Service	This operation establishes a linkage between this Name Service instance and another one which is specified by its PSI within the current service platform. The linkage is used to allow for cascading name resolving. This means if this Name Service instance has no information to map an OSI name to a PSI, or vice versa, it can redirect the request to all linked Name Service instances.
unlinkName Service	This operation removes a linkage between this Name Service instance and another one which is specified by its PSI within the current service platform.
Example usage	An instance of a Name Service is useful in the case of OSNs that span multiple service platforms connected through an OSN gateway.
Comments	none

Table 15: Description of the Name Service

9.6.8 User Management Service

Name	User Management Service
Standard Specifications	• IETF RFC 2251 Lightweight Directory Access Protocol (v3) • Java.sun.com Java Authentication and Authorisation Service (JAAS) (part of Java 2 SDK 1.4). http://java.sun.com/products/jaas/
Description	The User Management Service is used to create and maintain subjects including groups (of principals) as a special kind of subjects. In general, subjects represent entities that need to be authenticated. They are not authenticated themselves but rather represent a point of contact and management feature for authentication and authorisation purposes. A subject is decoupled from authentication. This decoupling is done by separating principals from subjects. A principal is an identity of a subject and is defined in an Authentication Service instance. Management of subjects includes the association to principals as well as storage of subject attributes. Group management includes definition of principal memberships. The User Management Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *UserManagementService*: Management of subjects and group subjects.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of a User Management Service instance. Examples of specific capabilities are structural information on subject attributes specialised with respect to the different types of subjects: • for human users, e.g. first name, surname as well as contact information • for groups, e.g. administrative contact. • for services, e.g. administrative contact. Additional pieces of information may be defined by a policy provided by the respective OSN.
Interface *UserManagementService*	
createSubject	Creates a subject. After a subject has been created, at least one principal has to be created and associated with the subject.

deleteSubject	Deletes a subject including the deletion of all associated principals and subject attributes.
updateSubject	Updates the subject itself. Can be used to change subject related information, e.g. subject attributes.
createGroup	Creates a group. Groups contain principals, not subjects. After creation a group has no members. Since a group is a special subject, principals have to be added. These can be managed using the *addPrincipalToSubject* and *removePrincipalFromSubject* operations. Group principals represent the identities of the group not group members. Group members can be managed using the operations *addPrincipalToGroup* and *removePrincipalFromGroup*.
deleteGroup	Deletes a group without deleting group member principals. Principals of the group are deleted if not specified otherwise.
updateGroup	Updates the group. Can be used to change group related information, e.g. group attributes. In order to manage group memberships use the operations *addPrincipalToGroup* and *removePrincipalFromGroup*.
getGroups	Retrieves an enumeration of existing groups.
addPrincipalToSubject	Associates an existing principal to an existing subject. This operation can also be used for the assignment of principals to group subjects (not group members).
removePrincipalFromSubject	Removes a prior assigned principal from a subject. This operation can also be used to remove principals from group subjects (not group members).
getSubjects	Enumerates all subjects of the current service instance. Use the operation getGroups to exclusively retrieve group subjects. There is no operation to retrieve an enumeration of non-group subjects. This can be done by simply removing group subjects from the result.
removePrincipalFromGroup	Removes the association between a given principal and a given group. The removed principal is not deleted in the corresponding Authentication Service.
addPrincipalToGroup	Associates an existing group with an existing principal. The principal may reside in another User Management Service instance.
Example usage	A group of users concerned with forest fires manages maps describing fire damage. Another group of users working on flood risk analysis would like to access the maps because they are relevant for their planning. Therefore, read access is granted to the flood analysis group for all maps and features contained in the map layers managed by the forest fire group.
Comments	none

Table 16: Description of the User Management Service

9.6.9 Authorisation Service

Name	Authorisation Service
Standard Specifications	• IETF RFC 2704 The KeyNote Trust-Management System Version 2 (September 1999) http://www.ietf.org/rfc/rfc2704.txt?number=270 • Ferraiolo David F. et. al: Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and System Security, Vol. 4, No. 3, August 2001, Pages 224–274. http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf
Description	The Authorisation Service gives a compliance value as response to a service requesting an authorisation decision for a given authorisation context. The Authorisation Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *AuthorisationService*: Includes all operations which are common to all Authorisation Service implementations regardless to their underlying paradigms. • *XAuthorisationAdministration* (where *X* could be e.g. *Rbac* or *Principal*): The administration interface is specific to the underlying paradigm, e.g. supporting role management and thus may vary for different Authorisation Service implementations. In the following a representative administration interface for a role based Authorisation Service is presented.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of an Authorisation Service instance. Examples of specific capabilities are the supported authorisation paradigms (e.g. principal permissions, or role-based access control). These paradigms are accompanied by specialised by dedicated administrative interfaces.
Interface *AuthorisationService*	
authorise	Requests an authorisation decision for a given authorisation context. An authorisation context is required as an input parameter. An authorisation context is a set of information used by the authorisation service to determine the authorisation decision for a given request. The authorisation context can contain, for example, the requesting principal(s), name of the invoked operation, etc. A compliance value representing the advice how to treat a certain service request is delivered as an output parameter. Authorisation contexts and compliance values need to be agreed upon between a service and its Authorisation Service.
Interface *Administration*	
createRole	Creates a new role. Newly created roles are empty. Neither permission nor principals are assigned, yet.
deleteRole	Deletes an existing role. Permission and principal assignments are deleted as well.
getRoles	Retrieves an enumeration of existing roles.
updateRole	Updates an existing role, e.g. description, etc.
assign Permission ToRole	Assigns permission to a certain role. Permission and role have to exist already.

unassign Permission FromRole	Removes permission from a certain role.
assignRole ToPrincipal	Assigns an existing role to an existing principal. This indirectly assigns permissions associated with the role to the principal.
unassignRole FromPrincipal	Removes the given role from a certain principal. This indirectly removes permissions associated with the role from the principal.
Example usage	For a Format Conversion Service it may be necessary to restrict access to certain principals. The service provider might use an Authorisation Service to assign these principals' permissions to perform conversions. This could be done with a service type independent Authorisation Service implementation supporting operation level authorisation. The authorisation context of such a service needs to include at least requesting principal(s) as well as the requested operation. An Authorisation Service implementation which is specific to Format Conversion Services might additionally restrict the size of files to be converted depending on the requesting principal. The authorisation context for such a scenario would need to include the size of the file to be processed. In the domain of Risk and Crisis Management, another example is the following: Access rights like read, write, access, execute services, compose services or feature collections, modify rights etc. are granted to principals of a Civil Protection Agency for all resources that relate to the responsibility domain of the agency. In case of a hazard event, read access rights are extended to all resources related to the hazard, independent of their organisational assignment.
Comments	none

Table 17: Description of the Authorisation Service

9.6.10 Authentication Service

Name	Authentication Service
Standard Specifications	• IETF RFC 4120 - The Kerberos Network Authentication Service (V5) • IETF RFC 4158: Internet X.509 Public Key Infrastructure: Certification Path Building • IETF RFC 4210: Internet X.509 Public Key Infrastructure Certificate Management Protocols • IETF RFC 4211: Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF) • IETF RFC 4325: Internet X.509 Public Key Infrastructure Authority Information Access Certificate Revocation List (CRL) Extension • IETF RFC 4386: Internet X.509 Public Key Infrastructure Repository Locator Service • IETF RFC 4387: Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP • Java.sun.com Java Authentication and Authorisation Service (JAAS) (part of Java 2 SDK 1.4). http://java.sun.com/products/jaas/ • OASIS Digital Signature Services (DSS) TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss

	- OASIS eXtensible Access Control Markup Language (XACML) TC http://www.oasis-open.org/committees/workgroup.php?wg_abbrev=xacml - OASIS Public Key Infrastructure (PKI) TC http://www.oasis-open.org/committees/workgroup.php?wg_abbrev=pki - OASIS Web Services Secure Exchange (WS-SX) TC http://www.oasis-open.org/committees/workgroup.php?wg_abbrev=ws-sx - OASIS Web Services Security (WSS) TC http://www.oasis-open.org/committees/workgroup.php?wg_abbrev=wss
Description	The Authentication Service verifies genuineness of principals using a set of given credentials. The authentication mechanism, which means the way authentication is performed, is up to the service implementation. Which credentials an Authentication Service needs as well as the way they are passed is specific to the authentication mechanism used. Session information returned after a successful authentication can be used to invoke services demanding authenticated principals. A service might use this information to perform authorisation requests. The Authorisation Service provides its functionality through the following interfaces: - *ServiceCapabilities*: Informs about the common and specific capabilities. - *AuthenticationService*: Includes all operations which are common to all authentication mechanisms. - *UsernamePasswordMechanism*: Contains operations which are specific to the authentication based on a username/password authentication mechanism. This interface should specify credentials as well as the way they are passed.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of an Authentication Service instance. Examples of specific capabilities are the supported authentication-mechanisms (e.g. username-password authentication, public-key authentication).
Interface *AuthenticationService*	
login	Initiates the validation of a certain principal for given credentials. Credentials have to be passed using the *AuthenticationMechanism* interface before calling the login operation. This needs to be done within a transaction. As an output parameter, the session information that can be used to invoke services demanding authenticated principals is provided.
addPrincipal	Creates a new principal. The principal representation is specific to the authentication mechanism used. For a username/password authentication the principal contains at least a username.
remove Principal	Deletes an existing principal. Removal of principals should not be done without updating corresponding User Management OSIs (see section 9.6.8) as well as updating services having permissions associated to the principal to be deleted. A solution to this could be the use of administration tools to keep track of consistency.
update Principal	Updates an existing principal. The principal to be updated as well as information to be changed, e.g. new username, shall be provided as input.

add Credentials	Adds credentials to a certain principals. Credentials are specific to the authentication mechanism used. For a username/password authentication credentials is a password.
Update Credentials	Updates credentials for a certain principal. The principal (username) for whom the credentials (password) should be changed as well as changed credentials shall be provided as input.
deactivate Principal	Deactivates a principal without removing it. The principal, e.g. username to be deactivated and additional information, e.g. a time period for deactivation, shall be provided as input.
activate Principal	Activates an existing principal. The principal, e.g. username to be activated and additional information, e.g. a point of time for activation, shall be provided as input.
Interface *UsernamePasswordMechanism*	
setUsername	Used to pass the principal to be authenticated. In a username/password authentication the username represents the principal.
setPassword	Used to pass the credentials to verify authenticity. In a username/password authentication the password represents credentials.
Example usage	A Format Conversion Service demands authorisation based on principals. Therefore each service requestor has to pass session information including at least one authenticated principal. In order to invoke a service a subject needs to authenticate a principal having appropriate permissions. The resulting session information can be passed to the service. The service uses – among others - the session information to build the authorisation context which is passed to the Authorisation Service.
Comments	It is part of the characteritsics of an OSN to determine if user authentication is necessary and if so, by using which authentication mechanism.

Table 18: Description of the Authentication Service

9.6.11 Service Monitoring Service

Name	Service Monitoring Service
Standard Specifications	• Web Notification Service 03-008r2
Description	The Service Monitoring Service provides an overview about ORCHESTRA Service Instances (OSIs) currently running within an OSN. OSIs can either be monitored using a push or pull model, that is, the status information is actively retrieved from an OSI by a service (this could be any service but preferably the Service Monitoring Service) or they are sent to the Service Monitoring Service. There is also the possibility to register an alert service and bind information of a specific monitoring status to that alert service. That way every time such information is received the alert operation of the alert service will be invoked. The Service Monitoring Service provides the functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *ServiceMonitoringService*: Implements a push model monitoring and alert service binding

	- *Monitorable Interface:* A service must implement this interface in order to use the pull model monitoring. - *Alert Interface:* Used when monitoring values of a certain status are provided. This can for example be used to contact the service administrator via email or Short Message Service.
Interface *ServiceCapabilities*	
getCapabilities	Informs the client about the common and specific capabilities of a Service Monitoring Service instance. Examples for specific capabilities are the supported statistics about the usage of a service in an OSN.
Interface *ServiceMonitoringService*	
putStatus	Gives any service the possibility to send monitoring information to the monitoring service.
getConfiguration	Retrieves the current configuration of the monitoring service.
setConfiguration	Sets the current configuration of the monitoring service. This includes information such as which services should be monitored, the binding between status information and alert services.
getConfigurationSchema	Retrieves the schema that describes the format of the configuration.
getStatistics	Retrieves statistical information about the monitored OSN or single services. These statistical values are features in order to enable easy usage with other feature processing services.
Interface *Monitorable*	
getStatus	Retrieves the status of a specific monitored property of the implementing service.
getConfiguration	Retrieves the currently active configuration of the monitored service.
setConfiguration	Sets the current configuration of the monitored service (e.g., interval that must be between getStatus calls in order to have new values available)
getConfigurationScheme	Retrieves the schema that describes the format of the configuration.
Interface *Alert*	
alert	This operation does not have a predefined functionality. It can either be sending an email or a Short Message Service or do some other mandatory processing.
Example usage	A service provider has her FeatureAccessService monitored by the ServiceMonitoringService. Whenever the hard disk usage exceeds 90% of the storage available a monitoring value of status CRITICAL is produced. This value is retrieved by the ServiceMonitoringService and since the status has been bound to an alert service, it is sent there invoking the alert operation. This OSI that implements the Alert Interface then sends a ShortMessageService to the service operator who can react to this situation.
Comments	Since the concrete procedure of reaction to an alert is application and most likely company dependant the semantic meaning of the alert operation can't be given. In some cases a simple email or other message will be passed to a responsible person, in other cases some complex automatic reaction will take place in case of an alert.

Table 19: Description of the Service Monitoring Service

9.7 OA Support Service Descriptions

9.7.1 Coordinate Operation Service

Name	Coordinate Operation Service
Standard Specifications	• ISO 19107:2003 Geographic information -- Spatial schema • ISO 19111:2003 Geographic information -- Spatial referencing by coordinates • OGC 05-008c1 Web Services Common Specification V1.0 • OGC 05-013 Web Coordinate Transformation Service (WCTS) draft Implementation Specification (Discussion Paper)
Description	The Coordinate Operation Service changes coordinates on features from one coordinate reference system to another (based on a 1-1 relationship). This includes operations on datum and projection. A Datum is used as a basis for defining a coordinate reference system and it specifies how the coordinate system is related to the earth. Examples are WGS84 and NAD1950. A projection is a method for depicting 3-dimensional data (the shape of the earth) in 2 dimensions. There are two principal variants of coordinate operations: • Coordinate conversion: An operation on coordinates that does not include any change of Datum. Examples of a coordinate conversion are a map projection between projected coordinates and geographic coordinates, or change of units such as from radians to degrees or feet to meters. • Coordinate transformation. An operation on coordinates that usually includes a change of Datum. The parameters of a coordinate transformation are empirically derived from data containing the coordinates of a series of points in both coordinate reference systems. This operation introduces errors, hence allowing derivation of error (or accuracy) estimates for the transformation. The Coordinate Operation Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *CoordinateOperation:* Request to change coordinates of features, either by a coordinate conversion or a coordinate transformation.
Interface *ServiceCapabilities from (OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of Coordinate Operation Service instance. Examples of specific capabilities are the supported conversions and transformations.
Interface *CoordinateOperation*	
check Operation	Reports if an operation between two Coordinate Reference Systems is supported by the service implementation and, if so, if it is a conversion or a transformation.
convert Coordinates	Convert coordinates without any change of Datum.
transform Coordinates	Transform coordinates usually including a change of Datum.

Example usage	Coordinate conversion: A user wants to convert coordinates from UTM Zone 33, Euref89 to Geographic coordinates, Euref89. Coordinate transformation: A user wants to change coordinates from UTM Zone 33, ED50 to Geographic coordinates, Euref89
Comments	none

Table 20: Description of the Coordinate Operation Service

9.7.2 Gazetteer Service

Name	Gazetteer Service
Standard Specifications	• ISO 19111:2003 Geographic information -- Spatial referencing by coordinates • ISO 19112:2003 Geographic information -- Spatial referencing by geographic identifiers • OGC 05-035r2 Gazetteer Service - Application Profile of the Web Feature Service Implementation Specification V0.9.3 (Best Practices Paper)
Description	The Gazetteer Service allows a user to relate a geographic location instance identified by geographic names (e.g. city, lake, region, street) with an instance identified by coordinates (e.g. a point, line, polygon or sets of these). A client delivers geographic names or describes them indirectly by means of a query (e.g. all cities in Bavaria) and receives geographic objects with their corresponding coordinates or vice versa. The Gazetteer Service usually provides this functionality by accessing a directory of geographic identifiers that describes location instances, called a gazetteer. The conceptual model of the gazetteer is taken from ISO 19112:2003. Here, location instances contain both geographic identifiers and the geographic positions. Access to the gazetteer is performed through operations of the *FeatureAccessService* interface (see section 9.6.1). Thus, by the selection of location instances using the query mechanisms of the Feature Access Service the relationship between names (indirect spatial reference) and coordinates (direct spatial reference) is carried out. For the purpose of gazetteer maintenance, the Gazetteer Service supports changes and updates of a gazetteer, too. A sequence of these operations may, if required, be secured by a transactional interface. The Gazetteer Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *FeatureAccessService*: provides read and write access to a gazetteer. • *TransactionInterface:* Secures sequences of change requests to a gazetteer.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the capabilities of a Gazetteer Sesrvice instance. Examples of specific capabilities are the provider organisation, the version and the geographic scope of the gazetteer.
Interface *FeatureAccessService (from Feature Access Service)*	
	The operations of the *FeatureAccessService* interface are used to access to the location types and instances of a gazetteer.

Interface *TransactionInterface* (*from OA Basic Service)*	
	The operations of the *TransactionInterface* are used when a synchronised access to the gazetteer must be assured, especially in the case of the *setFeature, createFeature and deleteFeature* operations.
Example usage	The Gazetteer Service may be used to integrate information in a risk assessment process if one of the source information items is geo-referenced by a geographic identifier (e.g. a statistical result based on a departmental area) and another by a geographic coordinate (e.g. measurement values at monitoring locations). In this scenario, the Gazetteer Service helps to generate comparable information that may be commonly processed.
Comments	A future version may consider a combination of a gazetteer with a thesaurus. Thus, the Gazetteer Service may use the operations of the Thesaurus Access Service (see section 9.7.7) in order to support multi-lingual gazetteers and fuzzy queries based on synonyms, quasi-synonyms or related terms, like “give me the coordinates of the city by the riverside of the Rhine that is close to Wiesbaden”. Further enhancements may cover distributed gazetteers, possibly across borders i.e. in combination with the gazetteer-thesaurus combination discussed above.

Table 21: Description of the Gazetteer Service

9.7.3 Annotation Service

Name	Annotation Service
Standard Specifications	• W3C OWL Web Ontology Language Overview http://www.w3.org/TR/owl-features/ • W3C-Resource Description Language http://www.w3.org/RDF/ • W3C RDF-Schema http://www.w3.org/TR/rdf-schema/
Description	The Annotation Service automatically generates specific meta-information from various sources and relates it to semantic descriptions. Semantic descriptions are to be specified as elements of an ontology (e.g. concepts, properties, instances). Sources to be annotated can contain unstructured information (e.g. documents, texts) or structured information (e.g. databases, applications). Annotations refer to the concepts of an ontology, which is specified in an ontology language such as OWL and RDF-Schema (a subset of OWL). The content of an annotation can be stored as a simple string. In order to provide references to concepts, instances and relation types stored in either a knowledge repository or a data ontology, the RDF syntax can be used. The generation of annotations of *unstructured sources* is based on automatic Information Extraction, by means of which named entities occurring in documents and texts can be identified and normalized by means of Natural Language Processing. The process of extracting information and its assignment to ontological elements is based on background knowledge held in a repository, the (pre-populated) knowledge base. In an OSN, such a knowledge base is accessible by means of the Knowledge Base Service. In addition to named entity identification, the service can automatically discover and formalize new knowledge by analyzing the texts. In a certain application scenario, this knowledge can be used to populate a knowledge base, from where it can be queried by means of query languages. The semantic annotation of documents and texts enables applications such as highlighting and document viewing to be supported by automatically generated links to semantic descriptions, or semantic indexing and retrieval (as described in the

	Document Index Service in section 9.7.4). The Annotation Service can automatically generate meta-information for *structured sources* such as databases, applications, etc. As a pre-requisite of the annotation service, the structure and content of such a resource is to be transformed into a data ontology which is compliant with the ontology containing the semantic descriptions. An annotation is a mapping of an element of this ontology to an element of the data ontology. The semantic annotation of databases and applications enables applications such as exploration of the database structure and content by means of ontology query languages, or interpretation of query results by means of domain knowledge. The Annotation Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *AnnotationService*: For sources to be annotated (e.g. documents, databases), an additional document - called a "semantic document" - is established which contains the annotations. Another operation of the service allows annotation of texts; here, the annotations are delivered directly in the operation result; a semantic document is not generated.
Interface *ServiceCapabilities (from OA Basic Service)*	
getCapabilities	Informs the client about the common and specific capabilities of an Annotation Service instance. Examples of specific capabilities are the supported annotation strategies (identification, population of new knowledge etc.), a list of mime types of documents which can be annotated, and a list of supported data and domain ontology formats.
Interface *AnnotationService*	
createSemanticDocument	In a first step prior to annotation, a “semantic document” is associated with the base document. A semantic document contains the content of the base document, the annotations and links to the base document and the corresponding domain ontology. After creation, a semantic document only contains the content of the base document; the annotations and the links are entered through the *annotateDocument* resp. *annotateDataOntology* operations.
annotateDocument	Generates annotations for a given semantic document for an unstructured source. The generated annotations are inserted into the semantic document.
annotateText	Generates annotations for a given text “on the fly”, i.e. they are not stored in a semantic document.
annotateDataOntology	Generates annotations for a given semantic document for a structured source. The semantic document has previously been generated from its data ontology by means of a *createSemanticDocument* operation. The generated annotations are inserted into the semantic document.
Example usage	Risk maps usually can display various thematic layers. The graphical representation in the risk map is explained in an attached legend. In many cases, the user needs more textual explanation about what the values in a legend exactly mean. With a growing number of layers and legends, a map can contain a considerable amount of attached text; new layers, legends and texts can be added dynamically. Moreover, the text itself could contain technical terms that make it difficult to read, or users might only be interested in getting further information on items occurring in the text. In this scenario, the attached text could be processed in an *annotateText* operation, which automatically sets up links of the terms occurring in the text to elements (concepts, instances) described in a domain ontology. The user can navigate to the respective ontology element and start browsing the ontology, thus getting help for interpretation of the text.

Comments	The service does not maintain the set of sources that are to be annotated; this functionality is expected to be provided elsewhere. For instance, annotation could be performed on a regular basis by means of a background job triggered at times of low load. The job checks the set of sources for changes that have been performed since the last run. Documents which have been changed are annotated again and old annotations are deleted.

Table 22: Description of the Annotation Service

9.7.4 Document Indexing Service

Name	Document Indexing Service
Standard Specifications	No corresponding standard known.
Description	The Document Indexing Service supports the automatic generation of document search indexes used to achieve a good and efficient "Boolean Retrieval" of documents. A document search index is meta-information for the purpose of discovery of documents (see section 8.4.2.1) "Boolean Retrieval" is a set of search methods that allows a user to search for information in the following way: - The user formulates a query inaccurately, i.e. they cannot formulate an exact query, but just give a vague description by means of some search terms, - Then, the user refines the search based on results of previous searches. - Finally, the user retrieves the complete document wherein the search term is contained (not just meta-information about it). The Document Indexing Service extracts all terms contained in a document and stores them in an inverted list which is the basis for the document search index. The document search index additionally stores for each term a reference to the document that contains it. Not all of the terms found in a document are stored in the index: for instance, stop-words can be eliminated, stemming and truncation algorithms are applied and so on. Term-based search has well-known weaknesses, which result from the fact that only boolean pattern matching is performed: very large result lists are offered to the user containing many unwanted hits, or documents containing the search term in the wrong context are part of the result list. Relevant documents are often not found despite the fact that they contain valuable content, because they do not contain the exact search term. The Document Indexing Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *Document Indexing*: This interface provides operations to start and stop index generation.
Interface *ServiceCapabilities* (from *OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of a Document Indexing Service instance. Examples of specific capabilities are the supported MIME-types for which this service can automatically extract meta-information.

Interface *DocumentIndexing*	
startGenerate Index	Generates a document search index from a collection of documents, organised as a list or a tree (e.g. a file directory). It may be requested that the generation be updated according to a given cycle time. The collection of documents must have been created by the Document Access Service, the MIME Type must be supported by the service implementation
stopGenerate Index	Stops the generation of the document search index.
Example usage	A user needs an efficient search mechanism for all documents that may be accessed within the entire OSN. Therefore, they need a simple interface for typing in search terms, e.g. like in Google. The Document Indexing Service creates and periodically updates a document search index which holds the effective search structure. See section 9.9.3 for a usage of the Document Indexing Service in the context of the discussion of how to generate meta-information.
Comments	An advanced version of the Document Indexing Service generates an index for smaller and more precise hit lists based on semantic information generated by the Annotation Service (see section 9.7.3). Such an index can be used not only to display the search hits but to embed them into their semantic context (identify search hits as resources which can be related to concepts specified in the ontology and display relationships to other resources as well).

Table 23: Description of the Document Indexing Service

9.7.5 Format Conversion Service

Name	Format Conversion Service
Standard Specifications	• MIME Media Types (http://www.iana.org/assignments/media-types/)
Description	The Format Conversion Service allows the conversion of data given in one format to the corresponding data given in another format. Each conversion between a pair of formats requires a conversion algorithm. The problem we face is how two organisations are able to exchange their data (e.g. documents) without caring about the format the other side uses. This is the reason why the Format Conversion Service is needed. It allows the conversion from one data format (in case of documents e.g. MS-Word, OpenDocument, pdf,) to another one in order to easily exchange data between different organisations. Data could be text based, like a word document or a pdf, or it could be binary data like JPEG or WMF. The Format Conversion Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *FormatConversion*: Provides the conversion operations.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilites	Informs the client about the common and specific capabilities of a Format Conversion Service instance. Examples of specific capabilities are the supported source and target formats and the conversion functionality between these formats.

Interface *FormatConversion*	
convert	Performs the conversion given by input and output MIME type.
Example usage	A time series of measurement values is available as an MS-Excel sheet and shall be converted into an XML file for further processing in an RM application.
Comments	It will be possible to build chains of format conversions. Example: If the conversion functionality png2gif, gif2jp and jpg2pdf are available, the call *convert*(doc1, png, pdf) will directly convert form a png to a pdf format.

Table 24: Description of the Format Conversion Service

9.7.6 Schema Mapping Service

Name	Schema Mapping Service
Standard Specifications	• W3C OWL Web Ontology Language Overview http://www.w3.org/TR/owl-features/ • W3C RDF-Schema http://www.w3.org/TR/rdf-schema/ • W3C SPARQL Query Language for RDF (Candidate Recommendation) http://www.w3.org/TR/rdf-sparql-query/ • W3C XML Path Language (XPath) 2.0 (Candidate Recommendation) http://www.w3.org/TR/xpath20/ • W3C XML Query (XQuery) (Candidate Recommendations) http://www.w3.org/XML/Query/ • W3C XSL Transformations (XSLT) http://www.w3.org/TR/xslt/
Description	The Schema Mapping Service provides functionality that is related to the mapping of features from a source into a target schema. It provides this functionality through two interfaces. The main functionality of the *SchemaMapping* interface is to execute a schema mapping. A schema mapping is considered to be “the definition of an automated transformation of each instance of a data structure A into an instance of a data structure B that preserves the intended meaning of the original information”. The service takes a feature collection and a description of the mapping from the source to the target schema as input and returns the features in the target schema. A schema mapping is described by • an identifier that is unique to the Schema Mapping Service instance; • descriptions of the source and target feature types; • the schema mapping language used to describe the mapping; and • a reference to the actual mapping. The Schema Mapping Service can be used to (1) directly map from one application schema to another one, or (2) to map from an application schema to a common (or community) schema (or vice versa). The latter can be used to perform an indirect mapping between two application schemas through the community schema. The mapping of features might also require that several feature collections be combined. In order to support this, an optional concatenation operation is also included in the interface. The description of the schema mapping is required as an input. It is outside the

	scope of the Schema Mapping Service to automatically derive a mapping between two application schemas. The *SchemaMappingRepository* interface supports repository functionality for mappings between source and target feature types. Service can also serve as a repository for mappings between source and target feature types. For this, operations for the creation (registration), retrieval, updating and deletion of schema mapping descriptions are foreseen. The Schema Mapping Service provides its functionality through the following interfaces: • *ServiceCapabilities:* Informs about the common and specific capabilities. • *SchemaMapping:* Execution of schema mappings and concatenation of feature collections. • *SchemaMappingRepository*: Creation, deletion, update and selection of schema mappings.
Interface *ServiceCapabilities*	
getCapabilities	Informs the requestor about the common and specific capabilities of a Schema Mapping Service instance. Examples of specific capabilities are the supported schema mapping language (for the *Schema Mapping* interface) and a list of the mappings registered with the service (for the *Schema Mapping Repository* interface).
Interface *SchemaMapping*	
mapFeatures	Maps a feature collection to a target schema.
concat	Concatenates several feature collections.
Interface *SchemaMappingRepository*	
createMapping	Registers a new mapping with this instance of the Schema Mapping Service.
getMapping	Returns a (list of) mapping(s) matching a given query.
setMapping	Updates a specific mapping.
deleteMapping	Deletes all mapping matching a given query.
Example usage	A client wants to transform a data source in a local schema into a common agreed global schema. The client submits a feature collection and mapping rules specifying how to map the features into the required feature type.
Comments	The described interfaces can be used in service implementations in different ways: • A service that only implements the *SchemaMapping* interface can be used to map feature collections in arbitrary schemas to a target schema using a mapping description that is provided by the requester. • A service that implements both interfaces can be used in the same way. In this scenario, the requester does not necessarily have to provide the mapping description themselves but can query the Schema Mapping Service for an appropriate mapping description. • A service that implements the *SchemaMappingRepository* interface and another interface for creating or accessing feature collections (e.g. the interfaces of the Feature Access Service or the Processing Service) can be used to provide the output feature collections in different schemas.

Table 25: Description of the Schema Mapping Service

9.7.7 Ontology Access Service

Name	Ontology Access Service
Standard Specifications	• W3C OWL Web Ontology Language http://www.w3.org/TR/owl-features/
Description	The Ontology Access Service supports the read access to the specification of a logical ontology (see section 8.6.1.2) and to export or import a complete specification of a logical ontology into an ontology store. It provides a high-level view to the content of the ontology, allowing the client to get information about the taxonomy (classes and properties) defined by any stored ontology and to extract TBox and ABox vocabulary statements for human/machine interpretation. The Ontology Access Service is independent of any ontology technology, like the ontology language (e.g. OWL). However, the current version of the Ontology Access Service ignores ontological classes that are implicitly defined by rules of description logics (and only the explicit taxonomy is considered). Some typical usages of this service are: • Getting a list of the ontologies this service is providing access to; • Storing, updating or deleting available ontology entries; • Retrieving a partially or fully a stored ontology; • Getting high-level information about ontology, such as the list of concepts or the list of supported properties for a given concept and TBox (optionally ABox) Vocabulary statements. The Ontology Access Service provides the functionality through the following interfaces: • *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Ontology Access Service. • *OntologyAccess*: Supports the storage, retrieval, and deletion of ontologies as well as providing a high-level view on ontologies.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the client about the common and specific capabilities of an Ontology Access Service instance. Examples of specific capabilities are the names of the ontologies available at the servers and the supported ontologies (e.g. OWL).
Interface *OntologyAccess*	
parse Ontology	Given an ontology or a part (selection) of an ontology, it returns the hierarchy of classes (concepts) and properties that are defined by this ontology (a high-level view of the ontology). The format of the result could be, for example basic XHTML (without CSS) that is suitable for both direct display or further machine processing.
getTBox Vocabulary	Given an ontology or a part (selection) of an ontology, it returns a list of TBox statements ready to be used for creating a Knowledge Base.
getABox Vocabulary (optional)	Given an ontology or a part (selection) of an ontology, this optional operation returns a list of ABox statements ready to be used for creating a Knowledge Base.
setOntology	Stores a new ontology in the ontology store, if the ontology format is supported. The operation confirms the success of the operation by sending back to the client a Boolean “TRUE”.

getOntology	Retrieves an existing ontology or a part (selection) of an ontology from the ontology store.
Delete Ontology	Removes an existing ontology from the ontology store. The operation confirms the success of the operation by sending back to the client a Boolean "TRUE".
Example usage	A party is having an ontology about forest fires and decides to share it with other parties. By invoking the setOntology operation, the ontology can be stored in the ontology store of the Ontology Access Service. The stored ontology can then be made accessible to other services. For example the Inferencing Service can retrieve the ontology for inferencing tasks, or the Knowledge Base Service can use the ontology to expand the knowledge base with information about forest fires. Finally, if a client possesses an ontology and wants to present its structure directly on the Web, the parseOntology operation can be called giving the ontology as a parameter. The response is a high level-view of the ontology hierarchy, classes and properties that can be immediately displayed or it can be further processed. Additionally assuming that a client requires on ontology about forest fires and assuming that there are already some ontologies in the ontology store, a client shall call the parseOntology operation for each of the stored ontology in order to get a high-level view of the available ontologies and decide if one of the ontologies is adequate for the purpose. Then the client could retrieve the full ontology or only a part of the required forest fire ontology and pass it to other services for further processing tasks.
Comments	The following are out of the scope of the Ontology Access Service: • Creating ontologies – this service manages the storage and the access to ontologies, but doesn't provide any tool to create ontology structures. This is the purpose of dedicated tools (like Protégé) and methodologies (as the one defined in deliverable D2.3.2). • Inferencing – this is the responsibility for the Inferencing Service. However ontologies used for the inferencing process may be accessed using the Ontology Access Service. • Management of knowledge bases – this is the purpose of the Knowledge Base Service (see section 9.7.10). The Ontology Access Service may be used to help for the creation of a knowledge base (using "getTBoxVocabulary" and "getABoxVocabulary" operations to extract statements), but is not able to store, search and manage knowledge bases. The ABoxVocabulary statements represents the knowledge extracted from only one ontology in the form of statements. In this respect, the getABoxVocabulary operation could be used to populate a knowledge base containing multiple statements (extracted from multiple ontologies or/and other data sources), that can be searched and that can provide answers to certain questions users may have. • Remote editing of ontologies – it is assumed that the client, once it is getting the ontology from this service, will use specialized tools or API (like Protégé or Jena API) to deal with the ontology structure and editing. Calling operations on a remote service to work with ontologies doesn't seem reasonable in terms of architecture or usability. A high-level structure can however be provided for clients that do not need any details but just overall information about the ontology (using "parseOntology operation").

Table 26: Description of the Ontology Access Service

9.7.8 Thesaurus Access Service

Name	Thesaurus Access Service
Standard Specifications	• ISO-2788 standard for monolingual thesauri • ISO 5964:1985 Documentation - Guidelines for the establishment and development of multilingual thesauri. • W3C Quick Guide to Publishing a Thesaurus on the Semantic Web (http://www.w3.org/TR/2005/WD-swbp-thesaurus-pubguide-20050517)
Description	The Thesaurus Access Service supports read and write access to a thesaurus that may be multi-lingual. A thesaurus can be thought of as a synonym and antonym repository for data vocabulary terminology (Pollock, Hodgson 2004). As such, a thesaurus is a variant of an ontology restricting the relations used to a priori relationships between terms, e.g. questioning whether the meaning of two terms is similar, broader, or narrower. In a multi-lingual thesaurus these a priori relationships are not restricted to one natural language, e.g. a term A may be a synonym to term B even if term A is available in English and term B in French. The Thesaurus Access Service is a run time service that provides on-the-fly insight into data meaning by cross-referencing the included terms and providing a human readable description. In this capacity the Thesaurus Access Service provides crucial links in the resolution of unknown data semantics for requestors that are attempting to resolve new schema relationships in newly discovered models. The requestor may choose the language in which the terms requested shall be provided. The Thesaurus Access Service provides its functionality through the following interfaces: • *ServiceCapabilities:* Informs about the common and specific capabilities. • *ThesaurusAccessService:* Includes the operations for the read and write access to a thesaurus.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilites	Informs the requestor about the common and specific capabilities of a Thesaurus Access Service instance. Examples of specific capabilities are the supported languages and relations. Note: The reason to provide these capabilities is less to reflect the services logic capabilities than to reflect the available data.
Interface *ThesaurusAccessService*	
getScope	Gets a note attached to a term to indicate its meaning within an indexing language (i.e. a controlled set of terms selected from natural language and used to represent, in summary form, the subjects of documents; see ISO 2788).
getPreferred Term	Gets the preferred term when a choice between synonyms or quasi-synonyms exists.
getSynonyms	Gets the synonyms of a given term in a given language.
getAntonyms	Gets the antonyms of a given term in a given language.
getTopTerm	Gets the broadest class to which the specific concept belongs; sometimes used in the alphabetical section of a thesaurus (e.g. The concept African elephant would return animal in case of a biological thesaurus)

getBroader Term	Gets a concept having a wider meaning than the given term has.
getNarrower Terms	Gets a concept with a more specific meaning than the given term has.
getRelated Term	Gets an associated term, but that term is not a synonym, a quasi-synonym, a broader term or a narrower term.
setScope	Sets a note attached to a term to indicate its meaning within an indexing language
setPreferred Term	Sets the preferred term for another term
setSynonyms	Sets a synonym for a term in a given language.
setAntonyms	Sets an antonym for a given term in a given language.
setTopTerm	Sets the broadest class to which a term belongs
setBroader Term	Sets a broader term for a term.
setNarrower Terms	Sets a narrower term for a term.
setRelated Term	Sets an associated term for a term; that associated term is neither a narrower nor a broader nor a top term, nor is it a synonym, quasi synonym or antonym.
Example usage	An end-user can use the Thesaurus Access Service to determine synonym terms, which can subsequently be used to broaden a search.
Comments	none

Table 27: Description of the Thesaurus Access Service

9.7.9 Query Mediation Service

Name	Query Mediation Service
Standard Specifications	• Joint US/EU ad hoc Agent Markup Language Committee OWL-QL http://www.ksl.stanford.edu/projects/owl-ql/ • OGC 04-021-r3 Catalogue Service Implementation Specification V2.0.1 (section 6.2.2 Common Catalogue Query Language CQL) • OGC 04-095 Filter Encoding Implementation Specification V1.1 • W3C OWL Web Ontology Language Overview http://www.w3.org/TR/owl-features/ • W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/ • W3C RDQL - A Query Language for RDF (Member Submission) http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/ • W3C SPARQL Query Language for RDF (Candidate Recommendation) http://www.w3.org/TR/rdf-sparql-query/ • W3C XML Path Language (XPath) 2.0 (Candidate Recommendation) http://www.w3.org/TR/xpath20/ • W3C XQuery 1.0 (Candidate Recommendation) http://www.w3.org/TR/xquery/

Description	The Query Mediation Service supports other services in the processing of select queries against heterogeneous source systems. A typical example will be the case of a cascaded catalogue, where the task will be to query other catalogues (even non-ORCHESTRA ones) contained in the root catalogue. One task of the Query Mediation Service is to mediate between different query languages. Thus, a Query Mediation Service instance is related to the query languages it supports. This includes well-known query languages like SQL but also query languages used for semantic mediation. The Query Mediation Service follows a 3-tier mediation model: 1) User queries specified by requestors (software components or humans) are relayed to a mediator component, i.e. an OSC offering a Query Mediation Service interface. 2) The mediator parses the query and rewrites it as necessary, formulates sub-queries against individual source systems, and 3) assembles query results into a single result set (also referred to as content mediation). The source systems might be of a very heterogeneous nature (e.g. catalogues, ontologies, XML repositories, relational data bases) that require different query mechanisms (e.g. CQL, X-Query, SQL, ORCHESTRA Feature Access Service). Note that this heterogeneity of interfaces goes far beyond the common ODBC/JDBC mediator models applied in the context of relational databases. But all source systems handled by this specification of the QMS handle structured information outlined in an according schema. The efficiency of the query mediation procedure is significantly improved if meta information about the source systems is available in a catalogue (see section 9.6.6). If more characteristics of a source system (e.g. schema information, available feature types) are registered in a catalogue, then better-tuned sub-queries may be generated. The Query Mediation Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *QueryMediationService*: Includes the operations for mapping queries from one query language to another.
Interface *ServiceCapabilities*	
getCapabilities	Informs the requestor about the common and specific capabilities of a Query Mediation Service instance. Examples of specific capabilities are the supported query languages and information about types used in services, which can be assembled by means of the Query Mediation Service.
Interface *QueryMediationService*	

createSub Queries	Creates sub-queries to individual source systems out of a given “select”-query and given information about the source system. The sub-queries are built according to the type of source systems against which the sub-queries will be performed. For each given source system one sub-query will be created. While creating a sub-query two cases could arise: 1. The original query can be mapped directly to one source system resulting in one source query, delivering the similar information as the input-query. 2. The original query cannot be mapped directly. Several source systems are necessary to answer parts of the question and the results need to be assembled. Thus *createSubQueries* allows to create several sub-queries for one original query.
performSub Query	Performs an individual given sub-query for a given source system. The sub-query must fit the specified source system type. The result of the query will be returned. Note that this operation just performs one sub-query. In order to avoid loops in case of distributed query performance, source systems that have already been queried for a specific query request shall be remembered.
assemble Results	Given data and information about services for the assembling of the data the *assembleResults* operation merges query results from individual source systems into a composite response. Depending on the type of the source systems, dedicated assembly services must be used. The assembled result set will be returned.
Example usage	One goal of the usage of the Query Mediation Service is to support the Catalogue Service for cascaded querying. A query is directed to an ORCHESTRA Catalogue. The ORCHESTRA Catalogue searches for results in the ORCHESTRA Catalogue itself and other catalogues: e.g. an OGC Catalogue or a UDDI registry. The task of the QMS is to mediate the queries from the query language and schema of the ORCHESTRA Catalogue to the query languages and schemas of the OGC Catalogue and the UDDI registry, receive the result set and assemble (e.g. with the Schema Mapping Service) the containing meta information into the meta information schema of the ORCHESTRA Catalogue instance, which can then deliver the result to the requestor. The operations of the QMS are to be used as follows: 1. The client (e.g. ORCHESTRA Catalogue) calls *createSubQueries* with the input select query. The result contains several sub-queries according to the cascaded catalogues (e.g. OGC Catalogue or UDDI registry). 2. For each sub-query the client (e.g. ORCHESTRA Catalogue) calls *performSubQuery* and receives a result. 3. Then the client (e.g. ORCHESTRA Catalogue) calls *assembleResults* with information about services used for assembling and all results obtained from the different *performSubQuery* calls and receives a composite result set.
Comments	The current abstract specification is oriented towards the usage of the Query Mediation Service for a cascaded catalogue scenario. It may be extended in future versions of the RM-OA for a more generic usage.

Table 28: Description of the Query Mediation Service

9.7.10 Knowledge Base Service

Name	Knowledge Base Service
Standard Specifications	• W3C RDF-Schema http://www.w3.org/TR/rdf-schema/ • W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/ • W3C SPARQL Query Language for RDF (Candidate Recommendation) http://www.w3.org/TR/rdf-sparql-query/
Description	The Knowledge Base Service provides access to a knowledge base in an OSN. The knowledge base can store identifiable units of knowledge, in the sequel referred to as “models”. A model has a uniform resource identifier (URI). The Knowledge Base Service conveys query requests to models received via the OSN to the knowledge base’s local processing engine and returns the results to the OSI that requested them. The Knowledge Base Service abstracts from existing languages for knowledge representation and querying, but it assumes that some concepts are common to most of them: • Knowledge is represented as a graph, i.e. a number of nodes and edges. • The knowledge graph is divided into a number of sub-graphs, so called “models”. • Models are described by a number of basic elements constituting the model graph; these elements describe the nodes and the edges. Updates of a model can be performed by adding/deleting basic elements. RDF is an example for a standard which fulfils these assumptions. In RDF, for instance, “statements” are the basic elements. SPARQL is a query language for RDF models. The SPARQL Protocol uses WSDL 2.0 to describe a means for conveying SPARQL queries to a SPARQL query processing service and returning the query results to the entity that requested them. The Knowledge Base Service can partly be implemented by means of RDF storage and SPARQL queries, but other implementations are possible. The main difference between a knowledge base approach and conventional *SQL databases* is that a knowledge base is more flexible: models can be added or removed during run time and there is no fixed database schema. A knowledge base can have a schema defined by means of ontology (e.g. RDF-Schema or OWL as schema of an RDF knowledge base), but it does not necessarily need one. The Knowledge Base Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *KnowledgeBaseInterface*: This interface provides operations to query and update models contained in the knowledge base. Queries are to be formulated in a query language that is compatible with the queried model. As opposed to the Feature Access Service, the result of such a request does not necessarily need to be a feature set: the service may deliver results of any format, from complete models down to boolean values. Update requests to a model contain the new elements, which are to be added to the model, and the elements to be deleted.

	• *TransactionInterface:* As update requests change the knowledge base, the Knowledge Base Service inherits the operations from the Transaction Interface of the OA Basic Service (see section 9.6.1).
Interface *ServiceCapabilities* (*from OA Basic Service)*	
get Capabilities	Informs the requestor about the common and specific capabilities of a Knowledge Base Service instance. Examples of specific capabilities comprise: • The possible representation formats of query results which can be requested by clients. • The types of the models supported by the Knowledge Base Service (e.g. references to standards such as RDF, RDFS, OWL). • The query languages that can be used in knowledge base queries. • The inferencing capabilities of the knowledge base applied when computing query results.
Interface *KnowledgeBaseService*	
queryModel	Submits a query to a model stored in the knowledge base. The model to which the request is to be sent is referenced by a URI. The query is formulated in a query language which must be compatible with the knowledge representation model used by the knowledge base. The service conveys the request to the knowledge base, which executes the query and composes the result in the required result format (parameter resultFormat). If the resultFormat parameter is not present, the result is delivered in a default format.
updateModel	Submits an update request to a model stored in the knowledge base. The model to which the request is to be sent is referenced by a URI. The request contains the set of basic elements to be added and the set of elements to be deleted. The service conveys the request to the knowledge base, which executes the update request.
Interface *TransactionInterface* (*from OA Basic Service)*	
	The operations of the *TransactionInterface* are used when a synchronised access to the knowledge base must be assured, especially in the case of the *updateModel* operation of the *KnowledgeBaseService* interface.
Example usage	Pre-population and automatic population: In a scenario, the knowledge base can hold so-called “named entity” definitions (e.g. mountains, rivers) and relationships between them. A named entity can be inserted into the knowledge base in two ways: • Pre-population – the named entities are imported or acquired otherwise from trusted sources. • Automatic discovery and population – the named entities are discovered in the process of automatic semantic annotation (or by usage of other knowledge discovery and acquisition methods) and are then populated into the knowledge base by means of the *updateModel* operation.
Comments	The RM-OA distinguishes on the abstract specification level between the Knowledge Base Service and the Ontology Access Service (see section 9.7.7) in order to support layered approaches in which ontologies and knowledge bases are clearly separated and jointly used. However, the RM-OA approach does not prevent interaction with non-layered knowledge organisation systems, and also does not force implementation of the services in separate components, so there is total flexibility on the implementation specification level. In its current specification, the Knowledge Base Service provides means for model

	update, but it does not provide means for adding and removing complete models. It is assumed that these tasks are performed via local, non-ORCHESTRA interfaces of the knowledge base (e.g. import). Nevertheless, implementation should allow adding and removing new models dynamically at runtime.

Table 29: Description of the Knowledge Base Service

9.7.11 Service Chain Access Service

Name	Service Chain Access Service
Standard Specifications	• DAML OWL-S Web Service Ontology version 1.1 (http://www.daml.org/services/owl-s/) • ESSI Web Service Modelling Ontology (WSMO) and Web Service Modelling Language (WSML) (http://www.wsmo.org) • ISO 19119:2005 Geographic information – Services • OASIS Web Services Business Process Execution Language (WSBPEL) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel • OGC 05-007r4 Web Processing Service (WPS), version 0.4.0 (discussion paper) • OMG Business Modelling and Integration DTF (http://bmi.omg.org/) • W3C Web Service Choreography Interface (WSCI) 1.0 (http://www.w3.org/TR/wsci/) • W3C Web Services Description Language (WSDL) 1.1 (http://www.w3.org/TR/wsdl)
Description	The Service Chain Access Service supports the creation of an executable service instance based on an explicit description of a service chain. The chain can then be executed as a single service. However, the execution of the service is outside the scope of the Service Chain Access Service (see comment below). Based on the Reference Model of Open Distributed Processing (ISO/IEC 10746-1 RM-ODP) definition of chain of actions, a service chain is defined in ISO 19119 as a sequence of services in which, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action. For the scope of this specification, it is important to distinguish between the description of a service chain (i.e. a document in some workflow language, e.g. BPEL), a deployed instance of a chain (i.e. an executable piece of code), and the actual process of executing the chain. The service specification is based on the aggregate service pattern where services appear as a single service which handles all coordination of the individual services that are part of the chain. The *createServiceChain* operation supports a service provider in creating an executable instance of an aggregate service based on an explicit service chain description, and optionally registering that service instance with a catalogue service. The Service Chain Access Service provides its functionality through the following interfaces: • *ServiceCapabilities:* Informs about the common and specific capabilities. • *ServiceChainAccessService*: Selection of service chain descriptions and creation and deletion of aggregate services based on such descriptions.
Interface *ServiceCapabilities (from OA Basic Service)*	

get Capabilites	Informs the requestor about the common and specific capabilities of a Service Chain Access Service. An examples of a specific capability is the supported workflow language in which the service chain description can be specified
Interface *ServiceChainAccessService*	
createService Chain	Deploys the service chain instance (an aggregated service) specified in a workflow document
getService Chain	Gets a descriptor of the service chain which includes meta-information (id, address, description, and workflow language) and the workflow description itself.
deleteService Chain	Deletes a service chain instance.
Example usage	A client creates an aggregate service which can access features and perform schema transformations. This service can now be accessed as one single service from a client.
Comments	In a service implementation the *Service Chain Access Service* and *Processing Service* interfaces can be combined. The workflow language can then be used to define combinations of several processing operations of this service instance. Thus, a combination of related processing operations can be executed with one call without having to send the same data repeatedly to the service.

Table 30: Description of the Service Chain Access Service

9.8 OT Support Services

Note: Some of the OT Support Services do not (yet) comprise descriptions of the service operations as the functionality of these services still needs further discussion within the ORCHESTRA project. The result of this discussion will include the list of OA Services and other OT Support Services that may be used by a given OT Support Service in order to provide its functionality according to the functional classification of the ORCHESTRA Services (see section 9.3).

9.8.1 Processing Service

Name	Processing Service
Standard Specifications	• OGC 05-007r4 Web Processing Service (WPS), version 0.4.0 (discussion paper)
Description	The Processing Service describes a common interface for services offering processing operations on spatial (vector as well as raster) and non-spatial data. Examples of processing operations are statistical or geospatial calculations, image processing and analysis or, in general, computer algebra operations. The Processing Service provides mechanisms to identify the data required by the calculation, initiate the calculation, and manage the output so that it can be accessed by the client. The Processing Service provides its functionality through the following interface: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *ProcessingService*: provides the means to get information on and to invoke a specific processing operation.
Interface *ServiceCapabilities (from OA Basic Service)*	
get Capabilities	Informs the requestor about the common and specific capabilities of a Processing Service instance. Examples of specific capabilities are the supported processing

	operations (name and abstract).
Interface *ProcessingService*	
getProcess Description	This operation allows a client to request and receive detailed information about one or more processing operation(s) that can be executed by an execute operation, including the input parameters and formats, and the outputs.
execute	This operation allows a client to execute a specified processing operation implemented by the Processing Service, using provided input parameter values and returning the outputs produced.
Example usage	A client wants to create a buffer zone around a forest during a fire and calculate the total area that is included in the buffer. The client queries the Processing Service for a description of the buffer processing operation (including its input and output types) using the *getProcessDescription* operation and then calls the buffer processing operation using the execute operation. The Processing Service returns the result of the buffer processing operation either directly or as a reference (that can be used by the client to access the result).
Comments	In order to avoid having to send the same data repeatedly to the same instance of a processing service to execute several related operations, it should be possible to invoke a combination of related processing operations with one call to the service. This can be achieved by a service instance that implements both the Processing Service and the Service Chain Access Service (SCAS) interface. Thus, a SCAS workflow language can be used to define combinations of processing operations. The optimisation of "local" operation calls is an issue that should be addressed at the implementation level. For the implementation of GIS functionalities, several (Open Source) GIS libraries exists, both for vector and raster data processing: • GRASS http://mpa.itc.it/markus/grass50progman/node98.html, (including OGC-conformal (Open Geospatial Consortium) Simple Features for interoperability with other GIS) • Terralib http://www.terralib.org/ • GeoTools http://www.geotools.org/display/GEOTOOLS/Overview • GMT http://gmt.soest.hawaii.edu/ • Map window http://www.mapwindow.com/ • OpenEV http://openev.sourceforge.net/ • Jump http://www.jump-project.org/ • STARS: Space-Time Analysis of Regional Systems, http://stars-py.sourceforge.net/whatisstars.html For the implementation of statistical functionalities, many tools and libraries are available. The mathematical algorithms used by the service operations could be taken from existing libraries, e.g: • OCTAVE http://www.gnu.org/software/octave/ (Free, Opensource) • Statistical analysis libraries such as R (http://www.r-project.org/) or Matlab (http://www.mathworks.com). • List of free software available at http://members.aol.com/johnp71/javasta2.html • A complete Statistical Analysis Software Survey available at http://www.lionhrtpub.com/orms/surveys/sa/sa1.html • See http://mathworld.wolfram.com/ for terminology and operator

	explanation An alternative architectural approach could be taken such that no Processing Service interface is described on the abstract level. Instead, the OMM would contain detailed rules about how processing service interfaces may be described by service providers. These descriptions should then include a process description, the input and output of the service and binding information, i.e. all information that is currently described in the Processing Service's *getProcessDescription* operation. In both cases and for a common understanding of processing operations, (basic) operations should be grouped and described in an operation taxonomy to be referenced in the service specific capabilities. Guidelines could be e.g. the Map Algebra operations (Tomlin 1990) or the Egenhofer Operators (Egenhofer 1989).

Table 31: Description of the Processing Service

9.8.2 Simulation Management Services

Name	Simulation Management Service
Standard Specifications	• OGC 05-007r4 Web Processing Service (WPS), version 0.4.0 (discussion paper)
Description	The Simulation Management Service allows the user to discover, specify input for, and control execution of a variety of simulation models. A simulation could be anything from a simple service which combines two numbers to a large simulation based on complicated mathematical models predicting the weather. The Simulation Management Service allows the implementer to allow others to discover, execute and control their model in a simple and generic fashion. The Simulation Management Service allows the model to initially support multiple simulations (which also could be derivatives of a particular model). The user can then ascertain the specifics of what the model requires to run (including additional input services and a description of the parameters required). The Simulation Management Service then provides the user the ability to execute and check on the models progress. They can also modify the currently executing model to dynamically modify the scenario. The Simulation Management Service provides its functionality through the following interfaces: • *ServiceCapabilities:* Informs about the common and specific capabilities. • *AsynchronousInteraction:* Exploits the OA Basic Service to provide a mechanism to invoke a simulation and obtain an ID for the simulation such that subsequent modification and query requests for that simulation can be made. • *ProcessingService:* Provides the operation to call the simulation run. • *SimulationManager:* Provides the interface to describe in detail the inputs required to invoke a supported simulation, as well as its outputs. The interface also provides operations to modify, suspend or resume an executing simulation, and to query its status.
Interface *ServiceCapabilities (from OA Basic Service)*	
getCapabilities	Informs the requestor about the common and specific capabilities of a Simulation Management Service instance. Examples of specific capabilities are the abilities of the simulation manager to include the types and versions of simulations supported by the simulation service

Interface *AsynchronousInteraction (from OA Basic Service)*	
invokeAsync	Starts asynchronous execution of a simulation. The *invokeAsync* operation returns immediately with an identifier (invocation ID) representing the asynchronous execution. In order to receive notifications a reference to a callback interface can be provided.
abort	Aborts execution of a simulation identified by its invocation ID.
notify	Passes a notification to the callback interface provider (to be implemented by the SimMS client).
Interface *ProcessingService*	
getProcess Description	Requests and receives detailed information about one or more processing operation(s) that can be executed by an execute operation, including the input parameters and formats, and the outputs.
execute	Executes a specified processing operation implemented by the Processing Service, using provided input parameter values and returning the outputs produced.
Interface *SimulationManager*	
modify Process	Applies a change to one or more simulation parameters during the execution of a simulation, to take effect from a defined point within the simulation. The simulation to be modified is identified by its invocation ID obtained by the *invokeAsync* operation. This operation also allows requests to the simulation state to be made to either suspend or resume execution.
query Process	Queries the state of a simulation identified by its invocation ID, to determine information such as whether the simulation has been suspended, is executing or has completed. As an option, this operation also provides the percentage complete.
Example usage	The caller wishes to execute a model. – Through *getCapabilities* the caller can discover what simulations can be executed. – On choosing a particular simulation the caller can then invoke *describeProcess* which reveals the requirements of the simulation. – The simulation is then invoked by *invokeAsync* which will execute the simulation. If the input to the simulation is ill-formed or invalid the execution will be aborted and the caller will have to re-specify. – The calling system can poll via *queryProcess* to find out the status of the simulation. – The caller may make dynamic modifications of the active scenario via *modifyProcess* (e.g. moving the position of a spill or adding extra wind). – When the simulation has completed, the SimMS returns the simulation results through the client's notify operation.
Comments	none

Table 32: Description of the Simulation Management Service

9.8.3 Sensor Planning Service

Name	Sensor Planning Service
Standard Specifications	• NASA/JPL Sensor Webs Project (http://sensorwebs.jpl.nasa.gov/). • OGC 05-086r2 - Sensor Model Language (SensorML) Implementation Specification V1.0 (Draft proposed version) • OGC 05-089r3 – Sensor Planning Service Implementation Specification V0.0.30 (Request for Comments)
Description	Following the OGC Sensor Planning Service Discussion Paper: “The Sensor Planning Service is intended to provide a standard interface to collection assets (i.e., sensors, and other information gathering assets) and to the support systems that surround them. Not only must different kinds of assets with differing capabilities be supported, but also different kinds of request processing systems, which may or may not provide access to the different stages of planning, scheduling, tasking, collection, processing, archiving, and distribution of requests and the resulting observation data and information that is the result of the requests. The Sensor Planning Service is designed to be flexible enough to handle such a wide variety of configurations.”
Example usage	A client wants to gather a satellite scene of a certain sensor for a certain region. The Sensor Planning Service offers the client a way to define the required parameters and to set up the respective notification mechanisms.
Comments	The specification of this service shall be aligned to the ongoing specification work within the OGC working group dealing with “Sensor Web Enablement”.

Table 33: Description of the Sensor Planning Service

9.8.4 Project Management Support Service

Name	Project Management Support Service
Standard Specifications	• ISO 10006:2003 Quality management systems -- Guidelines for quality management in projects • ISO 10007:2003 Quality management systems -- Guidelines for configuration management • PMI Project Management Body of Knowledge (PMBOK) (http://www.pmi.org/) • Project Management XML Schema (PMXML) (http://xml.coverpages.org/projectManageSchema.html) • dotProject - the Open Source Project Management tool (http://www.dotproject.net/index.php)
Description	The Project Management Support Service supports the planning and performance of operations (projects) in a cooperative distributed environment in cases where a desktop project management tool is not sufficient. Its purpose is to specify a project based on definitions according to the following dimensions of project management: - the structure of a project into project elements, i.e. the division of a project into sub-projects, work packages and tasks, the identification of logical dependencies between the project elements, the assignment of costs and priorities to the project elements and the identification of project results and partial results.

	- the structure of the resources, i.e. the identification of the type and number of resources (human resources, organisation units, machines, tools, computation resources, network bandwidth, ORCHESTRA features, ORCHESTRA services, meeting resources…), their characteristics (e.g. competences in case of human resources), their relationships (e.g. tool is part of a machine, person belongs to a organisation unit) and their location. - the time horizon, structured into units of, for example, months, weeks, days, hours, minutes in accordance with the plan horizon and the level of plan detail. Time oriented attributes include start and end dates of project elements, the identification of milestones and delivery dates for project results, the time dependencies between project results, the (estimated and actual) duration of project elements and the availability of resources during a given plan horizon. - the spatial dimension describing the location and movement of resources and where the project elements are to be executed. This service comprises the operations in the following operation groups: - to specify the project according to the three dimensions illustrated above with a close interlink to resources in an OSN. - to support queries about a project, like e.g. “Which resources are assigned to which task ?”, “What is the pre-requisite to deliver project result A ?”, “Which document is required to carry out task B ?” - to specify and optimise the allocation of resources to different tasks based on, for example, their importance, their order in which they must be undertaken and competition for the same resources. - to optimise the timely delivery and to calculate and optimise the cost of the project results - to specify and evaluate project scenarios based on multi-criteria optimisations The Project Management Support Service provides the following capabilities: list of supported project management techniques and their options, list of supported operations structured according to operation groups
Example usage	The service may be used in the risk management domain to support the development and evaluation of emergency plans in case of a natural hazard in a given area, e.g. the evacuation of a settlement in case of a threatening forest fire.
Comments	The service operations are based upon known project management techniques such as Gantt diagrams, PERT (Program Evaluation and Review Technique), CPM (Critical Path Method), PSP (Project Structure Plans) or Critical Chain Method. The applicability of more recent techniques such as that of the Business Communication Engineering tool Communigram® will be investigated (http://www.communigram.com/).

Table 34: Description of the Project Management Support Service

9.8.5 Communication Service

Name	Communication Service
Standard Specifications	• IETF 3261 SIP: Session Initiation Protocol, June 2002 • ITU T.120 Data protocols for multimedia conferencing • ITU H.323 Packet-based multimedia communications systems

	• OGC 03-029 OWS Messaging Framework (OMF) V0.0.3
Description	The objective of the Communication Service is to provide harmonised access to direct user-to-user communication means based on multi-media technologies and data exchange between users. Harmonised access is required as these services are most often associated with collaboration within a user community according to a common community objective (e.g. a project) which is not supported by the existing tools and standards in a common approach. The service will directly support users and provide them with the support to conduct interactive collaboration. Examples include: - Presence Awareness: ability to determine who is on line at a given instant - Chat: ability for multiple users to type text data onto their local device and the text can be seen by other chat session participants - Instant Messaging: combining Presence Awareness and Chat - Polling / Surveying: providing the ability for a user to request a vote from other collaboration participants - White boards: to interactively manipulate graphical objects with other users - Application Sharing / Desktop Sharing / File Sharing: provides users with the ability to control a shared application remaining running on the sharers computer (for example to allow multiple users to update a single document interactively) - Shared Storage: provides multiple users with a common place to upload and download files - File Transfer: to transfer a file to another user or set of users - Shared Calendars / Scheduling: provides a group of users with a common calendar that all may directly interact with - Teleconference (audio and/or video) - Audio and/or Video Broadcast The Communication Service indicates the following capabilities to the requestor: the interactive collaboration services supported together with the operations and options related to each of them.
Example usage	Usage through OA Services e.g. 1. Building of user communities and assigning access rights or 2. News registration and communication service Potential uses of collaborative communication services include, e-learning, workflow management, decision support, mission planning and logistics.
Comments	It is to be decided if parts, at least, of this service are better classified as Human Interaction Components than as Workflow/Task Management Services. The component could be a community portal integrating different communication services like e-mail, newsgroups or Internet Relay Chat.

Table 35: Description of the Communication Service

9.8.6 Calendar Service

Name	Calendar Service
Standard Specifications	• ISO 8601: 2004 Data elements and interchange formats -- Information interchange -- Representation of dates and times • ISO 19108:2002 Geographic information - Temporal schema.
Description	The Calendar Service performs arithmetical date/time functions, comparisons and format conversions. As most information in thematic domains has a temporal dimension with a reference to a calendar date (e.g. a measurement value), there is

	a need to support calculations using these dates (e.g. for time series analysis in case of measurement series). The service provides operations to convert between different representations and the usual one using year, month, day, hour, minute and second, - to compare two dates and to perform simple arithmetical functions like adding/subtracting a number of days or seconds and computing the difference between two dates, - to create a calendar for any month, past, present and future, for easy use with other services, - to perform calculations between dates, reducing time computations to simple arithmetic. The Calendar Service indicates the following capabilities to the requestor: list of operations supported, including the parameters and their expected format
Example usage	To try to recreate history or project the future one might need to know just what day was the first Sunday of November 1963 or what day of the week May 12, 2034 will be. The service allows a client to enter a date, to specify a number of days to be added (to check a future date) or subtracted from (to check a past date) and to get the new date. Or, it allows a client to specify a pair of dates in order to calculate the number of days between these.
Comments	none

Table 36: Description of the Calendar Service

9.8.7 Reporting Service

Name	Reporting Service
Standard Specifications	• OASIS Open Document Format for Office Applications (OpenDocument) (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office)
Description	The Reporting Service supports the creation of reports using actual information from other services according to a given template. The process to create a report can be of very high complexity. Thus, instead of providing a generic report generator, this service offers a wrapper interface to existing products and tools for report generation. While many report formats are imaginable, for practical reasons only standardised formats are supported.
Example usage	The result of a seismic risk assessment has to be publicised regularly in a format that has been standardised by a civil protection agency. The Reporting Service supports this task by allowing a template to be provided once according to the report standard and filling the template based on the actual data.
Comments	For reporting there might be more than one source for input data. For simple reports a configurable service may be provided, for special cases subclasses of this service can be created.

Table 37: Description of the Reporting Service

9.9 OA Service Interaction Patterns

The combined usage of the OA Services and the ORCHESTRA Information Models is illustrated by means of OA Service interaction patterns. Note that these interaction patterns are informative and just provide examples. It is not claimed that this is the only way of using and combining the OA Services nor that this way is complete.

The following OA patterns are currently described:

- Controlled user access to resources
- Integration of source system data into an OSN
- Registration of resources in a catalogue
- Generation of meta-information
- Semantic catalogue component

Note: Further OA patterns will be added in the RM-OA version 3, e.g. feature rendering in maps and diagrams.

9.9.1 Controlled User Access to Resources

For the description of the following service interaction pattern, the User Management Service, the Authorisation Service and the Authentication Service (UAA services) are intended to work together in the following way. This use case assumes the following context and OSN characteristics:

- Two departments of one organisation are attached to the same OSN and share a common UAA policy (see section 11.1.5).
- The OSN comprises OSIs of a Format Conversion Service, a Document Access Service and a Feature Access Service that use one User Management OSI, one Authentication OSI and one Authorisation OSI in the following way:
  - The Format Conversion OSI is owned by Department 1. The Feature Access OSI and the Document Access OSI belong to Department 2.
  - Department 1 has an administrator “admin 1”. Department 2 has an administrator “admin 2”.
  - Each administrator is responsible for the services of his department.
  - Department 1 has an employee “user 1”.
  - Department 2 has an employees “user 2”.
  - The Authentication OSI implements a username/password authentication mechanism.
  - The Authorisation OSI implements a role based authorisation paradigm.

In the following, some scenarios are described in order to illustrate the combined usage and the interaction of the UAA services.

9.9.1.1 Scenario “UAA Setup”

This scenario cannot be described in detail because the setup procedure of each service depends on its implementation. Nevertheless, we can describe in principle how such a setup could look.

1. The Authentication OSI is set up. During the setup the first principal called root principal is created. The root principal can be authenticated and the resulting session information is used during the setup of the Authorisation and User Management OSIs.
2. Each UAA service OSI has a simple built-in authorisation component which grants all available permissions to the root principal until the actual Authorisation OSI has been configured.
3. The root principal creates the admin principals “admin 1” and “admin 2”.

4. The next step is to register the User Management and Authentication OSIs as well as the Feature Access, the Format Conversion and the Document Access OSIs in the Authorisation Service. How this is done is specific to the Authorisation Service implementation.
5. After the Services have been registered the root principal creates admin permission for principals “admin 1” and “admin 2” for the corresponding services.
6. For security reasons the root principal will be deactivated.

From now on “admin 1” and “admin 2” are able to administer their services.

9.9.1.2 Scenario “Create new User”

1. The human “John Doe” behind “user 1” has demanded a user account.
2. Admin1 creates a principal “user 1” in the Authentication OSI.
3. Admin1 creates a subject with John Doe’s personal information as subject attributes in the User Management OSI.
4. Admin1 assigns principal “user1” the newly created subject using the *addPrincipaltoSubject()* operation of the User Management OSI.

User1 is now a valid system user but cannot access any service due to the lack of corresponding service permissions.

9.9.1.3 Scenario “Permission Assignment”

1. “User 1” has requested permissions to access the Format Conversion OSI.
2. “Admin 1” assigns an operation permissions for the *convert* operation of the Format Conversion Service to the principal “user1”.

“User 1” is now able to able to invoke the Format Conversion Service.

9.9.1.4 Scenario “Service Request”

1. “User 1” wants to invoke operation *convert* against the Format Conversion OSI.
2. In order to receive session information “user 1” (the client software of “user 1” respectively) uses the Authentication OSI to authenticate his “user 1” principal using his password.
3. “User 1” attaches the session information to the *convert* operation of the Format Conversion Service.
4. The Format Conversion OSI parses the session information and extracts the reference to the Authentication OSI of the authenticated principal(s).
5. The Format Conversion OSI makes a request to the Authentication OSI to verify session information. Verification of session information is implementation specific and might use session keys, for example.
6. The Format Conversion OSI creates an authorisation context and passes it to the *authorise* operation of the Authorisation Service. The structure of the authorisation context is known to the application and specific to the permission types supported by the Authorisation Service. For a operation permission type, for example, the operation context includes the name of the operation to be invoked.
7. The Authorisation OSI receives the authorisation context. It checks whether the given principal (included in the authorisation context) has sufficient permission to invoke the requested operation. This is done within an implementation and permission type-specific decision process. Evaluating an operation permission means, for instance, to check whether the given operation may be invoked. An evaluation of a time coverage permission might require a comparison between the current timestamp and a time coverage given in the permission associated with the current principal.
8. The Authorisation OSI returns a compliance value representing the authorisation decision.
9. The Format Conversion OSI interprets the compliance value. It throws an

OA_PermissionDeniedException for a negative compliance value and performs the operation for a positive one.

9.9.2 Integration of Source Systems into an OSN

Source System Integration has been defined in section 7.6 as the process of transforming an External Source System into an ORCHESTRA Source System. Thus, it starts in a native (i.e. non-ORCHESTRA) environment and results in a running OSI that represents the access point to the data and functionality of an External Source System within an OSN. This OSI must be built according to the rules that are defined in the ORCHESTRA Service Meta-model (OMM-Service as described in section 9.2).

Integration of one or more External Source Systems into an OSN means creating (at least) one new OSI. This instance is created by defining and implementing an ORCHESTRA conformant interface resulting in a service that is able to interact with the External Source System. For the description of this OA pattern, the resulting OSI is called Source System Integration Service in order to have a single name for the entirety of services of this kind. It is a surrogate name since Source System Integration Services needn't share any predefined interface type (apart from the mandatory *ServiceCapabilities* interface of the OA Basic Service) that could be used as a name instead.

Note: The name Source System Integration Service neither states that any specific interface is implemented, nor does it create a new service type, since a Source System Integration Service might as well be just an implementation of the service type Feature Access Service.

Editorial Note: In order to avoid the misunderstanding that a generic service type is being defined, the name "Source System Integration Service" will be replaced in the next release of the RM-OA.

Starting on an abstract level, the integration process of source systems can be described in the following steps:

1. Check the available interfaces types of the defined ORCHESTRA Service Types and select (if any) the interfaces that are suitable to represent the External Source System. (e.g.: a database might be best represented through a *FeatureAccessService* interface as specified in section 9.6.2). This step is not restricted to selecting only one interface type, therefore it's valid for a Source System Integration Service to realise multiple interface types as defined in abstract specifications.
   According to the OMM-Service, at least the *ServiceCapabilities* interface of the OA Basic Service must be selected in this step.
   If the External Source System provides operations that the integrated ORCHESTRA Source System shall offer to the OSN, continue with step 2. If there aren't any further operations continue with step 3.

2. If the collection of selected interface types does not completely fit a predefined ORCHESTRA Service Type, a new service type shall be defined.

3. There are two possible ways to integrate any operations that the External Source System provides. One of these must be chosen as illustrated in Figure 36.

   a. Extend the Source System Integration Service's interface with a new operation for every operation of the External Source System that should be integrated (and therefore visible in the OSN).

   b. Implement the *SynchronousInvocation* interface of the OA Basic Service and add the external operations as possible parameters to the *invoke()* operation.

4. Transform the native meta-information that will be needed within the OSN into ORCHESTRA meta-information according to the rules defined in Annex B1 of the RM-OA (or define such meta-information from scratch if it is not available yet). A non-exhaustive list of such meta-information that would be the contents of the service capabilities (e.g.: provider information, interface self-description…), the OAS, the feature type descriptions, ontologies, parameter types…

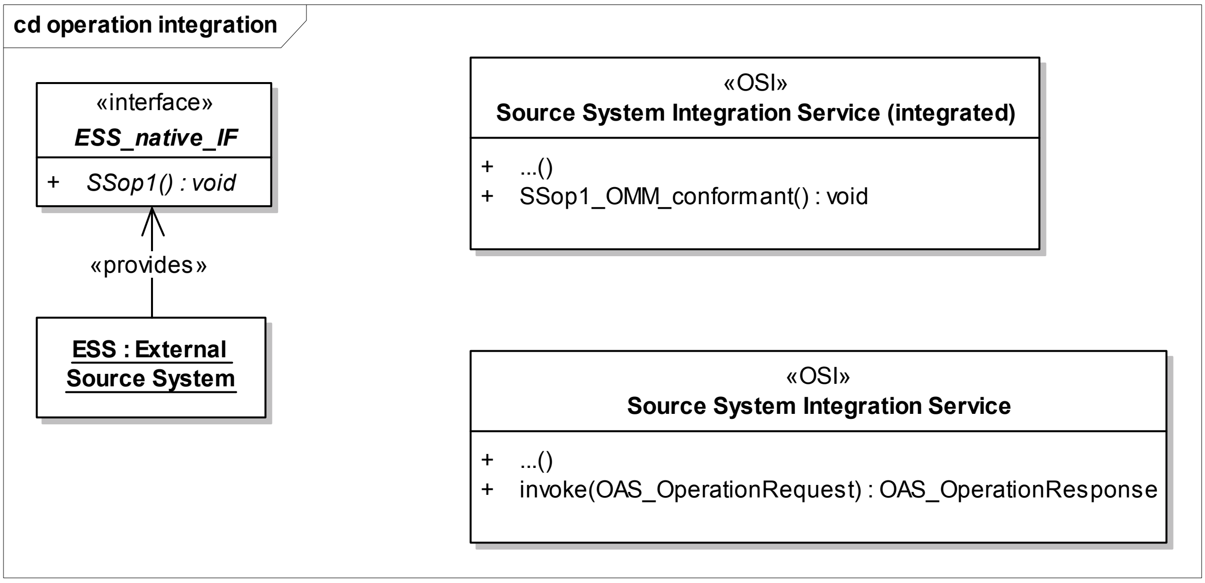

cd operation integration
«interface»
ESS_native_IF
+ SSop1() : void
«provides»
ESS : External Source System
«OSI»
Source System Integration Service (integrated)
+ ...()
+ SSop1_OMM_conformant() : void
«OSI»
Source System Integration Service
+ ...()
+ invoke(OAS_OperationRequest) : OAS_OperationResponse

Figure 36: Operation Integration (upper right: SSI step 2a, lower right: SSI step 2b)

5. For a given platform, provide an implementation specification for the interface types of the Source System Integration Service.
6. Develop an OSC that corresponds to the implementation specification. This can be done either by mapping the interface operations to the native interface operation of the External Source System or by implementing the functionality from scratch.
7. Create and start an instance of the Source System Integration Service (a respective OSI) within the OSN.

Note 1: These steps are the tasks a source system provider must perform in order to integrate his External Source System into an OSN when starting on abstract level. Of course, these steps can be supported by tools in order to result in a (semi-) automatic integration process.

Note 2: A corresponding integration process could be defined when directly starting on platform-specific level.

Note 3: During all of those steps existing interface types of OA/OT-Services and also implementations of OA/OT-Services might be used to facilitate the tasks that need to be performed in the integration process (e.g.: a Schema Mapping Service might be used to transform a database schema into an OAS). But implementations of the OA/OT-Services are not required to support the integration process in any way, since this would mean that those services have to operate outside the specified boundaries of ORCHESTRA.

Figure 37 shows the basic and common interfaces among all integrated source system integration services. Since the type of the External Source System is unknown, it is impossible to know the interfaces needed for all possible External Source Systems. Therefore a generic and also extendable interface must be given as a base.

All that is predefined is the required service self-describing operation *getCapabilities()*.

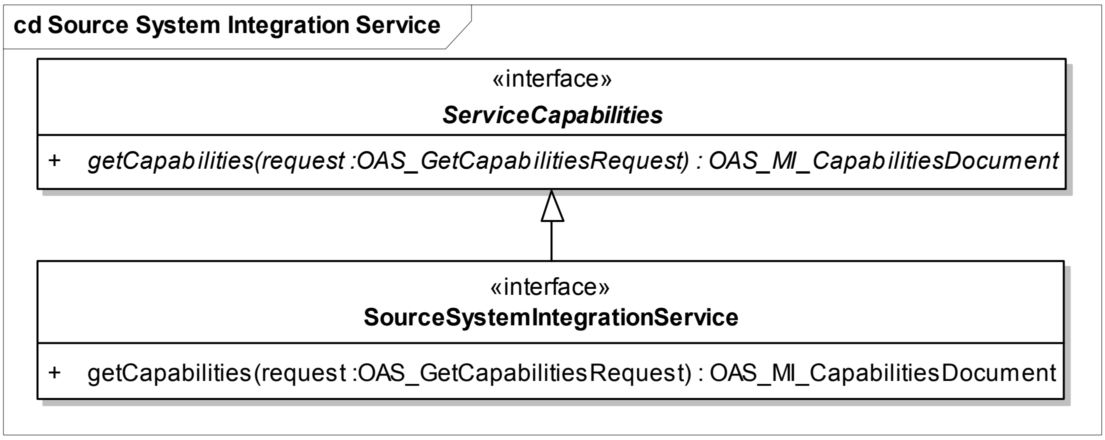

cd Source System Integration Service
«interface»
ServiceCapabilities
+ getCapabilities(request :OAS_GetCapabilitiesRequest) : OAS_MI_CapabilitiesDocument
«interface»
SourceSystemIntegrationService
+ getCapabilities(request :OAS_GetCapabilitiesRequest) : OAS_MI_CapabilitiesDocument

Figure 37: Source System Integration Service

In order to be able to support the wide heterogeneity of available External Source Systems, the Interface of the Source System Integration Service can be extended as the integrator desires. This includes inheriting and implementing interfaces of predefined OT/OA-Services as described in the RM-OA as well as adding new operations unrelated to any predefined interface type. Of course the meta-information, especially the interface description in the service capabilities, must reflect this. Thus, it contains all operations that are available at the service, having in mind that there might not be a hand-written specification of the service in case of a fully automated source system integration process.

9.9.3 Generation of Meta-information

Several OA Services provide the means for the generation of meta-information. Figure 38 outlines known methods for that purpose and assigns the respective OA Service to each method.

Meta-information is generated for various types of resources, being feature or service instances, according to a well-defined purpose (see section 8.4). The main criteria for the classification of methods for the generation of meta-information is the distinction between manual and automatic (or semi-automatic) approaches.

Manual generation of meta-information is usually carried out by a human user, who inserts values into certain fields of meta-information of an input mask. On the one hand, meta-information may consist of simple attributes, such as keywords for discovery purposes, which can be used to find resources by applying a boolean match. The attributes may then be defined according to a meta-information standard such as Dublin Core, ISO 19115 or ISO 19119 in case of service meta-information. On the other hand, meta-information may be schema information in order to support the mapping of information between several schemata. The Catalogue Service (see section 9.6.6) can be used for the access to meta-information for discovery purposes (see method ① in Figure 38).

A more advanced method for describing resources is to edit statements which can be added to a knowledge base by means of the Knowledge Base Service (see section 9.7.10), where they are stored as a knowledge graph. An implementation example of such a knowledge base is an RDF (Resource Description Framework) Triple Store. The statements describe the relationship from resources to concepts of an ontology and their relationship to other resources as well. Thus, this kind of meta-information is on a semantic level, as it can be interpreted by an ontology. However, there is currently no dedicated ORCHESTRA Service for the manual generation of ontology-based knowledge (see method ② in Figure 38).

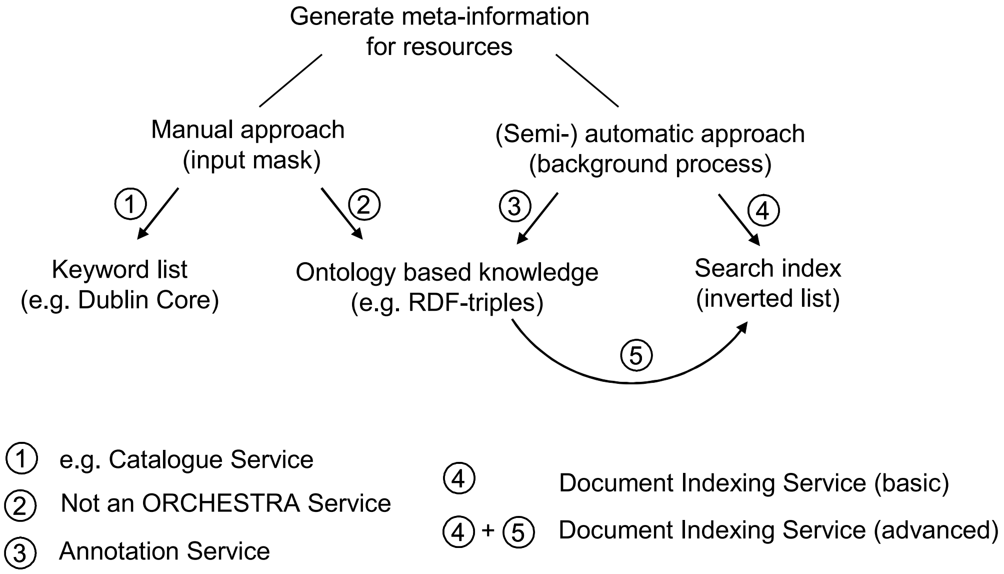

Generate meta-information for resources
Manual approach (input mask)
(Semi-) automatic approach (background process)
①
②
③
④
Keyword list (e.g. Dublin Core)
Ontology based knowledge (e.g. RDF-triples)
Search index (inverted list)
⑤
① e.g. Catalogue Service
② Not an ORCHESTRA Service
③ Annotation Service
④ Document Indexing Service (basic)
④ + ⑤ Document Indexing Service (advanced)

Figure 38: Services for generation of resource meta-information

The OA currently aims at supporting an automatic approach by means of the Annotation Service (see method ③ in Figure 38 and the service description in section 9.7.3). The Annotation Service identifies named entities in texts and automatically establishes relationships between resources and concepts and between resources among each other. The information in such a knowledge base can be explored by browsing the ontology using dedicated navigation tools or by formulating exact queries in an ontology query language.

In many cases, users do not want to retrieve knowledge about resources, but search for and retrieve the resources themselves. The search is not formulated in exact queries, but based on some vague information which the searcher can just describe by means of keywords. Such keywords can be typed in manually for each resource, as outlined above. A more advanced method, especially for documents or Web sites, is to automatically establish an index of all terms contained in the text and corresponding references to the occurrence of the term (such an index is called an inverted list). The Document Indexing Service (see section 9.7.4) provides such a facility (see method ④ in Figure 38) for all occurrences of documents (i.e. features of type OA_DocumentDescriptor, see section 8.7.5.2) in an OSN.

A more advanced approach to the Document Indexing Service is to combine it with the Annotation Service, i.e. to take advantage of the existence of knowledge generated by the Annotation Service. The advanced Document Indexing Service exploits this knowledge in order to achieve better search results (see method ⑤ in Figure 38).

Note: Due to a lack of user requirements, the advanced form of the Document Indexing Service will not be specified during the course of the ORCHESTRA project.

9.9.4 Registration of Resources in a Catalogue

Registration means the creation of an associated meta-information entry of a resource (data or service) in a catalogue in order that a user in an OSN may discover the resource. The registration of the resources can be achieved via the *CataloguePublicationInterface* and the *CatalogueCollectionInterface* of the Catalogue Service (see section 9.6.6), which provides means for including, updating and deleting catalogue entries. The *CataloguePublicationInterface* provides a push paradigm and the *CatalogueCollectionInterface* provides a pull paradigm.

The meta-information in a catalogue is structured according to an OAS-MI that the catalogue is able to handle. The following figures illustrate an example in which an OAS-MI is structured according to the capabilities of ORCHESTRA services which need to be described in the catalogue. In this example, the meta-information is extracted from an OSI by calling the operation *getCapabilities()* contained as part of the *ServiceCapabilities* interface of any OSI.

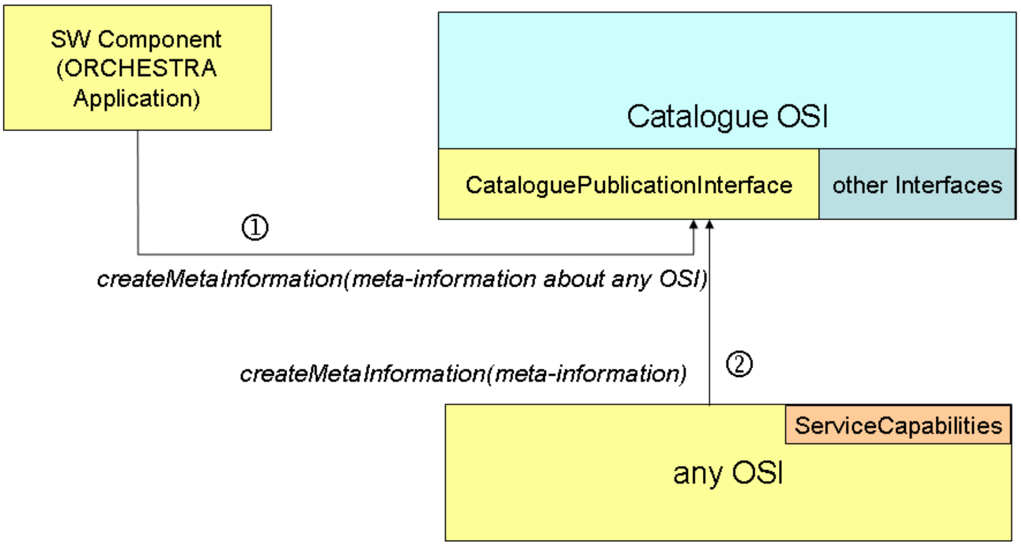

SW Component (ORCHESTRA Application)
Catalogue OSI
CataloguePublicationInterface
other Interfaces
①
createMetaInformation(meta-information about any OSI)
createMetaInformation(meta-information)
②
ServiceCapabilities
any OSI

Figure 39: Generation of meta-information entries (push paradigm)

Now, the push paradigm is supported by the operations *createMetaInformation()* and *setMetaInformation()*. By calling these operations software components of ORCHESTRA Applications (see case ① in Figure 39) or any OSI itself (see case ② in Figure 39) can directly store meta-information in the catalogue.

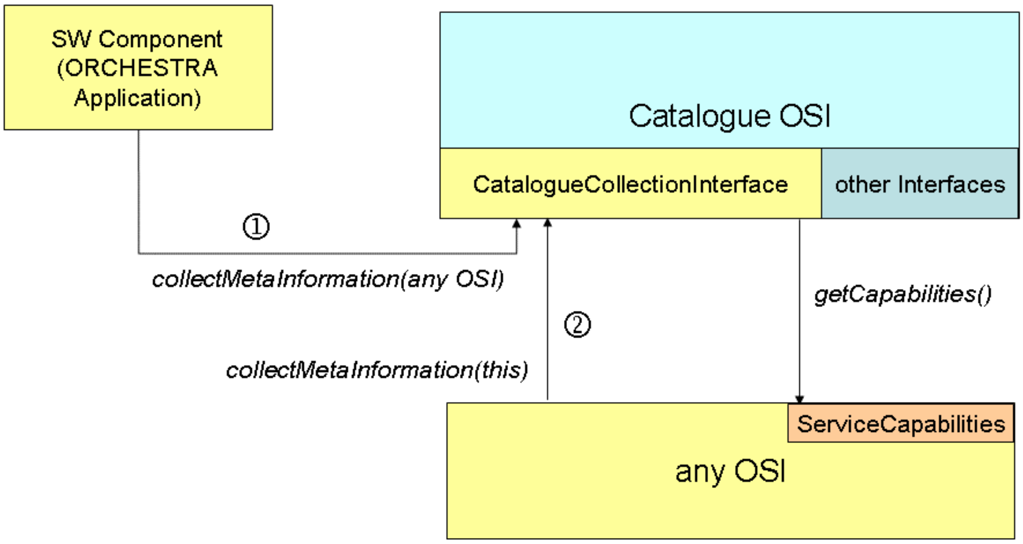

SW Component (ORCHESTRA Application)
Catalogue OSI
CatalogueCollectionInterface
other Interfaces
①
collectMetaInformation(any OSI)
②
collectMetaInformation(this)
getCapabilities()
ServiceCapabilities
any OSI

Figure 40: Generation of meta-information entries (pull paradigm)

The pull paradigm is supported by the operations *collectMetaInformation()* and *collectMetaInformationPeriodic()*. By calling these operations software components of ORCHESTRA Applications can trigger the catalogue to pull meta-information from an OSI (see case ① in Figure 40) or an OSI itself can trigger the catalogue to pull the meta-information (see case ② in Figure 40). *CollectMetaInformation()* is used for a single pull, while *collectMetaInformationPeriodic()* is used for periodic updates of the resources.

9.9.5 Semantic Catalogue Component

An OSC called "Semantic Catalogue" may be built by combining the ORCHESTRA Catalogue Service and the Query Mediation Service as illustrated in Figure 41. A Semantic Catalogue supports the ability to publish and search resources by means of semantic resource descriptions. A resource may be a data element (feature) as well as a service. A resource is described by meta-information which is structured in accordance with an ontology (domain ontology, service ontology). The Semantic Catalogue thus manages a repository of resource descriptions and allows its clients (human users, agents) to find, browse and access resources using semantic queries.

An important variant of a Semantic Catalogue is one that provides, on the front-end to a client application, an interface in form of the ORCHESTRA Catalogue Service based on a CQL or a semantic query language and, on the back-end, access to more than one catalogue service, possibly with different associated meta-information models, e.g. OGC Catalogue Services in the form of the ebRIM and ISO application profiles or any other non-OGC compliant catalogue service. However, this structural diversity should be transparent to the user of the Semantic Catalogue component. By means of query mediation a query to the Semantic Catalogue is directed to the appropriate catalogue service. The response (meta-information in the catalogue's own structure) is then transformed by means of result assembly (content mediation) to the global meta-information structure which is returned as query response to the user.

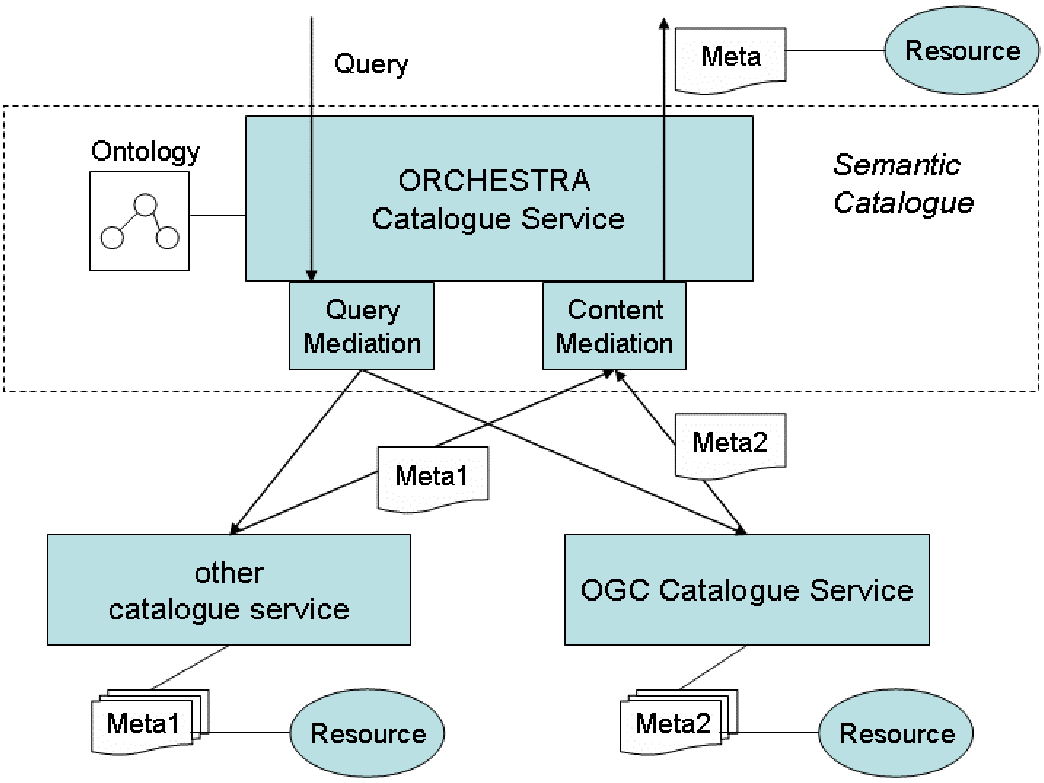


Figure 41: Example of a semantic catalogue

9.9.6 Naming in Dynamic OSN Environments

In the following, a usage of the Name Service (see section 9.6.7) in the case of dynamic OSN environments is described. Dynamic OSN environments are characterised by the fact that the assignment of OSIs to one or more OSNs may change during the lifetime of an OSI (e.g. due to a central OSN administrative decision or due to an autonomous decision of an OSI).

Note: Version 3 of the RM-OA will investigate how an OSI knows about the used naming policy for its own name and its (current) membership in an OSN.

In order to support a dynamic OSN environment, an interaction of Name Service instances is required. Consider the following cases:

- An OSI is added to OSN A and is not already registered at any Name Service instance. In this case, the OSI can be registered at the Name Service instance of OSN A. The Name Service creates a globally unique name for the OSI and can then be used to resolve the name.
- One or more OSIs are added to OSN A and these OSIs are already registered at a Name Service instance of OSN B. As these OSIs already have names, the Name Service instance of OSN A is not used to create OSI names. Instead, a mechanism is needed to create a linkage between the Name Services instances of OSN A and OSN B. Such a mechanism is further described below.
- An OSI is removed from an OSN. If the OSI is not member of another OSN, it may be deregistered from the Name Service instance of the OSN, which means that it will lose its name. However, it may also be useful to keep its name and registration in order to use the OSI in another OSN.
- A new OSN is created and OSIs are added as described above. The new OSN may establish a new Name Service instance or may reuse an existing one of another OSN.
- An OSN is removed which implies that all its OSIs are removed from that OSN. The Name Service instance of the OSN may still be used by another OSN.

The following figure illustrates a linkage between two Name Service instances.

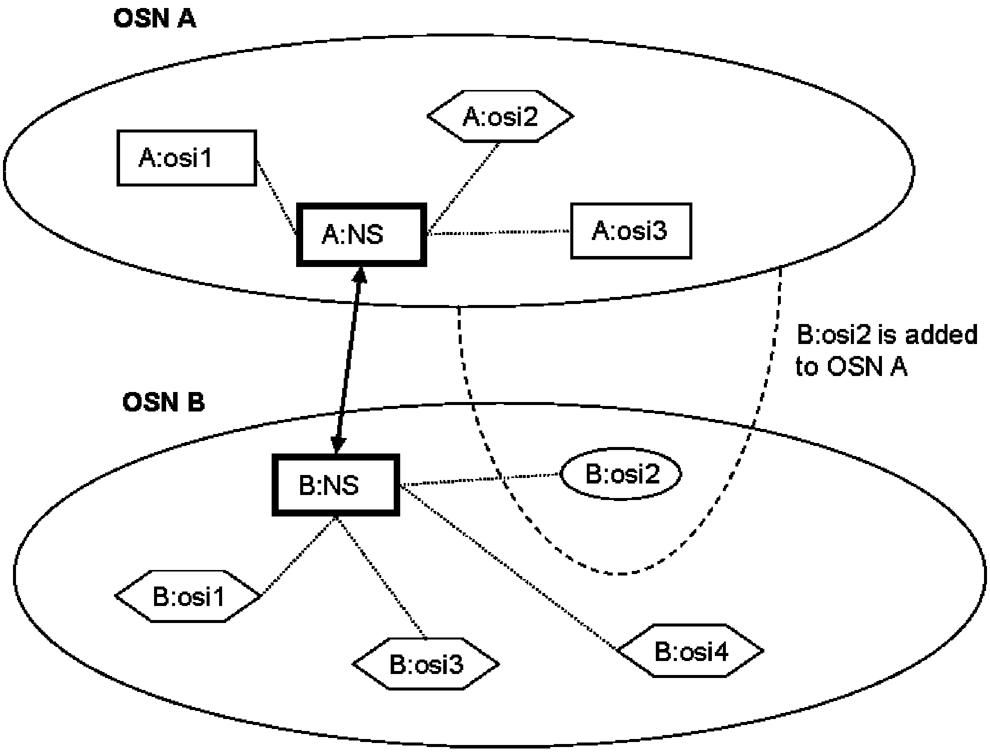


Figure 42: Linkage between Name Services

The figure shows two OSNs which are initially separated. Each OSN has its own Name Service instance indicated by A:NS and B:NS. Each OSI is registered at the Name Service instance of its OSN, indicated by the connecting lines. Now, B:osi2 from OSN B is in addition added to OSN A.

As the added OSI is registered at B:NS, a linkage is established between A:NS and B:NS. The linkage is used for name resolution in the following way. In order to resolve a name within OSN A, A:NS is used. If A:NS is not able to resolve the name among its registered OSIs, it uses the linkage and directs the request to B:NS. Thus, cascading name resolution is performed. This allows the resolution of the name of B:osi2 using A:NS.

Note that the linkage may be used in both directions for cascading name resolution. B:osi2 may use its original Name Service B:NS to resolve names within OSN A and OSN B.

A:NS and B:NS may use different naming policies.

To support linkage of Name Service instances, the Name Service has an additional interface called *NameServiceLinkage* that includes the following operations:

linkNameService(PSI)

This operation establishes a linkage between this Name Service instance and another one which is specified by its PSI within the current service platform. The linkage is used to allow for cascading name resolving. This means, if this Name Service instance has no information to map an OSI name to a PSI or vice versa, it can redirect the request to all linked Name Service instances.

unlinkNameService(PSI)

This operation removes a linkage between this Name Service instance and another one which is specified by its PSI within the current service platform.

10 Technology Viewpoint

According to the ORCHESTRA Reference Model as introduced in section 5.3, the Technology Viewpoint specifies the technological choices of the service platform and its operational issues. Thus, when considering the design process of the ORCHESTRA Service Network, it contains the specification of the service platform and its characteristics upon which the ORCHESTRA Services and ORCHESTRA Application Schemas are to be mapped.

The present RM-OA document, being a reference model for the design of ORCHESTRA Service Network, only contains the guidelines and requirements for the platform specification. It comprises the following parts:

- a specification of all properties that are required to be compliant with the SOA Reference Model of OASIS,
- a specification of how the UAA mechanisms are intrinsically supported by the platform,
- agreement on the usage of specific data formats (e.g. non-GML representation of coverages),
- a specification of a bijective mapping of the platform-specific schema language from and to UML (both for information models and for service types) according to the OMM,
- a specification of possible restrictions of the platform, e.g. to be considered in the service mapping process.

10.1 Specification of Platform Properties

Being a realisation of the OMM meta-class OMM_PlatformSpec (see section 9.2.2.2), a platform specification has to define the following set of properties, which are considered in the context of the OASIS Reference Model for Service Oriented Architecture 1.0 (SOA-RM, 2006). As an important example see the platform specification as part of (ORCH-ImplServ 2007) for the “ORCHESTRA Web Services Platform”.

1. Platform Name

   Name of the platform. In case of a standard platform, a reference shall be provided.

   Example: “ORCHESTRA Web Services Platform”

2. Interface Language

   Specification of the formal language used to define SOA-RM Service Interfaces. In case of a standard language, a reference shall be provided.

   Example: Web Services Description Language (WSDL)

3. Execution Context

   Specification of the SOA-RM Execution Context. The Execution context is an agreement between service providers and consumers. It contains information that can include preferred protocols, semantics, policies and other conditions and assumptions that describe how a service can and may be used. This includes, for example, the specification of the transport and the security layer, the format of the messages exchanged between service providers and consumers, etc. In case of a standard SOA-RM Execution Context, a reference shall be provided.

   Example: SOAP 1.2 HTTP binding for message transport, WS-Security in conjunction with SLL shall be used if encryption of messages is required, etc.

4. Schema Language

   Specification of the schema language used to define SOA-RM Information Models. The schema language defines the platform dependent encoding of a platform independent information model as specified by an ORCHESTRA Application Schema.

Example: XML-Schema and a GML Profile based on the GML Simple Feature Profile.

5. Schema Mapping

   Specification of how to map the abstract level (UML) to the schema language used for this particular platform.

   Example: XML-Schema/GML encoding rules for ORCHESTRA Application Schemas (ISO/DIS 19136 Annex E and F + additional rules for non-GML types)

6. Information Model Constraints

   Specification of the constraints on the SOA-RM Information Model, especially the constraints on the message format required to accomplish the SOA-RM Action model.

Furthermore, in the following sections some specific aspects are discussed that have to be considered on a platform level in order to increase the level of interoperability.

10.2 Selection of User Management, Authentication and Authorisation Mechanisms

The RM-OA concept for User Management, Authentication and Authorisation (UAA) and the respective abstract specifications are by intention specified at a high level of abstraction in order to be able to cope with established UAA mechanisms for dedicated platforms. Thus, a platform specification has to define how the ORCHESTRA UAA concept can be realised for a specific platform. This includes an agreement on the authentication and authorisation mechanisms permitted within an OSN, the transport and handling of session information among OSCs, the selection of a language for the expression of permissions and possibly the predefinition of common permissions and default subjects and principals. Some aspects of these definitions, especially the technical details, may not necessarily be part of the platform specifications but of the implementation specifications of the UAA services.

Example: The Authentication Service implements a simple username/password mechanism, and the Authorisation Service a role-based access control (RBAC) system. Additional authentication and authorisation mechanisms are not supported. Session information will be exchanged by means of a platform specific protocol, for example inside the header of a SOAP message.

Note: Corresponding rules to support this requirement will be added in the OMM-Service in a future version of the RM-OA.

10.3 Agreement on Data Formats

A platform specification may also contain an agreement on the usage of (de-facto or de-jure) standard data formats (e.g. MIME types) and specific, often proprietary data formats to be exchanged between OSCs.

Example: An agreement on well-known coverage representation formats (e.g. GeoTIFF, HDF) to represent coverage type information which is not encoded in GML.

10.4 Definition of a Reversible Platform Mapping for Information Models

Since an information model may also be modelled directly in a platform-specific schema language without the need to follow the OMM approach of defining an OAS and applying platform specific mapping rules, the conformance of such information models to the OMM has to be ensured.

It must be possible to generate the UML representation of a given information model, modelled in a platform specific schema language, to check compliance to the OMM. Therefore the definition of encoding rules for the mapping of an OAS to a platform specific transfer format must not be ambiguous and has to be specified as a reversible mapping as part of the platform specification.

A platform specification may also include an optional annex providing procedures and guidelines for how these mapping rules shall be applied.

Examples:

1) Usage of the reversible encoding rules from ISO/DIS 19136 Annex E and F for the platform "Web Services" to map (ORCHESTRA) Application Schemas to GML.

2) Provision of a table that maps basic UML data types (see section 8.7.2.2) to basic XML-Schema data types and vice versa (e.g. CharacterString ⇔ xsd:string).

3) Guidelines for the usage of UML to GML Application Schema tools.

10.5 Definition of Procedures for the Mapping of Service Interfaces

Procedures for the mapping of the platform-neutral service interfaces to a specific interface language may have to be defined. These procedures shall ensure that the mapping is in compliance with the rules of the ORCHESTRA Service Meta-Model (OMM-Service, see section 9.2). The procedures should be defined in an optional annex of the platform specification. The mapping itself shall be part of an implementation specification. If this can be accomplished, such a mapping should be bi-directional and described in a machine readable way.

Example: Description of how to transform XMI to WSDL using Enterprise Architect.

Note: In cases where ORCHESTRA Services are directly specified on a platform level, compliance with the OMM-Service must be assured nevertheless for interoperability reasons. A future RM-OA version will have a closer look to this problem.

10.6 Restrictions on certain Services

A platform specification may further reduce the complexity or restrict the scope of certain services, if this is required to meet the main characteristics of the selected platform.

Note that this complicates interoperability between different platforms. There should exist a bi-directional mapping between an abstract and an implementation specification and this mapping should be described in a machine readable way.

Example: A platform “OGC Web Services” may permit the mapping of some OA Services to OGC service interfaces by knowingly allowing a derivation from the abstract service interface specifications.

11 Engineering Viewpoint

According to the ORCHESTRA Reference Model as introduced in section 5.3, the Engineering Viewpoint specifies the mapping of the ORCHESTRA service specifications and information models to the chosen service platform and the specification of the characteristics of ORCHESTRA Service Networks.

Thus, when considering the design process of the ORCHESTRA Service Network, the mapping process itself belongs to the Engineering Viewpoint. It is documented in corresponding sections of the implementation specifications, see (ORCH-ImplServ 2007).

The present RM-OA document, being a reference model for the design of ORCHESTRA Service Network, restricts the description of the Engineering Viewpoint to the discussion on OSN Characteristics.

Note: The following sections in the RM-OA Engineering Viewpoint are preliminary ideas and need to be validated and formalised during the course of the ORCHESTRA project when further implementation experiences have been gained. Results of this validation will go into version 3 of the RM-OA.

11.1 OSN Characteristics

11.1.1 Policies

An ORCHESTRA Service Network (OSN) is defined as a set of networked hardware components and ORCHESTRA Service Instances (OSIs) that interact according to defined policies in order to serve the objectives of ORCHESTRA Applications (see section 5.3.3). Thus, the basic units within an OSN for the provision of functions are the OSIs, whereas their interaction principles are determined and characterized by policy definitions. Instead of pre-determining a specific policy for all possible OSNs, the following sections of the RM-OA only defines policy elements and rules for the definition and the existence of policies for different OSN characteristics. Using this approach, the policies of an OSN may be set-up according to the given individual business and organisational needs and models.

Note that this approach does not fix the model for policy enforcement, be it centralised or decentralised. Furthermore, it does not prescribe the time and the way that the policies are defined, be it (pre-) determined by a central authority or negotiated online between the participating parties. Thus, a wide spectrum may be covered, from a centrally-administered OSN with a high level of access control and a fixed and pre-defined list of OSIs up to an open and flexible OSN with dynamic registration and de-registration of OSIs and a distributed administration.

An OSN is characterized by following a harmonised approach for the following policies:

- resource naming
- resource discovery
- OSN operation
- UAA (User Management, Authorisation, Authentication)

11.1.2 Resource Naming Policy

The Resource Naming policy of an OSN deals with the question of how resources in OSN-like service instances and feature instances are identified. The uniqueness of resource names in an OSN and across OSNs is further discussed in the section 11.3.

The Resource Naming policy is defined by the following elements:

- name service: statement if a Name Service (see section 9.6.7) is used it is responsible for the provision of globally unique identifiers for OSIs and/or feature instances.
- naming policy for service instances: specification of which naming policy is used for the identification of OSIs. Currently, the following approach has been identified (see section

11.3.1):

  - platform as namespace: The global uniqueness of OSIs is enforced by using the service platform as namespace, i.e. the platform-specific identifier of an OSI is used.

- naming policy for feature instances: specification of which naming policy is used for the identification of feature instances. Currently, the following approach has been identified (see section 11.3.1):
  - OSI as namespace: Each OSI that acts in the role of a Feature Access Service shall be responsible for managing a namespace of related feature instance identifiers.

11.1.3 Resource Discovery Policy

The Resource Discovery policy of an OSN deals with the registration of resources in an OSN. Registration means the creation of an associated meta-information entry for a resource in a catalogue in order that a user who is part of the information community of that catalogue may discover the resource (see section 9.9.4).

The process of registration as well as the process of discovery is supported by operations specified in the Catalogue Service (see section 9.6.6). A resource may be registered in one or more catalogues.

The meta-information about resources is defined in OAS-MI according to the rules of the OMM Information Model. A resource may be the OSN itself, feature types and instances, service types and instances and UAA resources such as subjects.

The Resource Discovery Policy is defined by the following elements:

- discovery policy: statement about the discovery policy used in the OSN. Possible alternatives are:
  - centralised discovery: There is a distinguished Catalogue OSI (called OSN Catalogue) that serves as the “entry point“ to the OSN.

    Note: The presence of an OSN Catalogue does not exclude the existence of other instances of the Catalogue Service.
  - decentralised discovery: All instances of the Catalogue Service are equivalent.
  - no discovery: There is no Catalogue OSI. This means that the service interactions are not mediated through an instance of a Catalogue Service.

    Note: Whether a network of OSIs without a discovery capability based on a Catalogue OSI is a “valid” OSN is under discussion.
- definition of the OSN Catalogue
  - name of the OSN Catalogue
  - query language of the OSN Catalogue
  - ORCHESTRA Application Schema for Meta-information (OAS-MI) of the OSN Catalogue for the purpose of discovery
  - resource types that may be discovered through the OSN Catalogue
    - OSN
    - feature types
    - feature instances
    - service types
    - service instances
    - subjects
    - … others

  - ORCHESTRA Application Schema for Meta-information (OAS-MI) of the OSN Catalogue for the purpose of service invocation, i.e. the OAS-MI for the default service capabilities for all OSIs running in the OSN.

    Note: The default service capabilities usually correspond to the OAS-MI for service instance discovery (see above). However, this is not obligatory.

11.1.4 OSN Operating Policy

The OSN Management Policy is divided into three sub-policies which are described in the following sections:

- OSN management policy
- service management policy
- network management policy

11.1.4.1 OSN Management Policy

The OSN Management Policy deals with the requirements concerning the management and the operation of an OSN. It is defined by the following elements:

- general administrative information
  - name: globally unique name of an OSN.

    Note: An example for such a name is the name of the OSN Catalogue (see section 11.1.3).
  - description: human-readable textual information about the goals and purpose of the OSN.
  - OSN provider: Information about the institution or organisational unit operating the OSN
  - administrators: Names and addresses of those persons who are responsible for the operation of the OSN.
- Technical Information
  - platform: reference to the platform specification upon which the OSN is based

    Note: Currently, an OSN may only run on top of one specified platform.
  - name and platform-specific identifier (OSI) of the “OSN Catalogue” (if any, see section 11.1.3) as the entry point to the OSN
  - requirements for all OSIs interacting in the OSN:
    - minimal required set of formats (see the *acceptFormats* parameter of the *ServiceCapabilities* interface as specified in section 9.6.1) that every OSI has to support for the *getCapabilities*-operation
    - minimal required set of query languages
    - minimum required level of security to be provided by each OSN component (client or OSI). The security policy shall make statements (e.g. technologies or platform-specific mechanisms used) about the following topics:
      - encryption of communication

        Note: There are some limitations by law in some countries about the usage of encryption and some sort of communication technology (e.g. France).
      - measures against intrusion, alteration, eavesdropping, non-repudiation

- service registration: statement about whether a service can be registered at any time by any subject (open service registration) or whether the service registration is controlled (controlled service registration based on a resource discovery policy, see section 11.1.3).
- type of OSN (see section 11.2)
- list of mandatory services within the OSN, i.e. at least one OSI of this service type shall be operational in an OSN. This list may be derived from the type of OSN or listed explicitly.
- list of additional services that are allowed to be provided within this OSN. The alternatives are:
    - any service of any service type is allowed
    - no other service is allowed
    - a specified number of services of a specified list of service types are allowed

11.1.4.2 Service Management Policy

The Service Management Policy deals with the administrative requirements that OSIs of a specific service type have to fulfil when interacting within a specific OSN. It is defined by the following elements:

- service monitoring
    - list of service and network events to be monitored (e.g. just make calls to *getCapabilities*())
    - list of OSIs to be monitored (e.g. all OSIs that are registered in the OSN catalogue) and supported statistics about the usage of services in an OSN (see Service Monitoring Service described in section 9.6.11)
    - list of conditions under which management notifications have to be generated
- quality of service
    - availability of service (e.g. work hours, 24x7, redundant)
    - maximum response time for service operations

11.1.4.3 Network Management Policy

The Network Management Policy deals with the management of the communication resources of the specified platform. For this part of the OSN Operating Policy, the RM-OA refers to the corresponding management standards that are specific for the chosen platform and underlying communication protocols, e.g.

- for protocols based on the Internet protocol stack (IETF RFC 1122 Requirements for Internet Hosts -- Communication Layers), these are the IETF recommendations related to RFC 2570 Introduction to Version 3 of the Internet-standard Network Management Framework.
- for protocols based on ISO/OSI 7498-1 Open Systems Interconnection, these are the ISO standards related to ISO/OSI 7498-4 Management Framework.

11.1.5 User Management, Authorisation and Authentication Policy

The User Management, Authorisation and Authentication policy of an OSN is divided into the following sub-policies:

- user management policy
- authentication policy
- authorisation policy

There are many different concepts and technologies in the context of user management, authorisation and authentication. Often, these concepts and technologies cannot be applied independently from each other. Thus, it must be ensured that the policies are specified coherently.

11.1.5.1 User Management Policy

The User Management Policy deals with the way users are represented and made known (registered) in an OSN. It is defined by the following elements:

- subject information: minimum information to be provided when specifying a subject.
- dynamic registration of users: statement about whether dynamic registration is allowed or not. In case it is allowed the business process for dynamic registration shall be described for each of the following:
    - subjects (users, including ORCHESTRA Service Instances (OSIs)),
    - groups (group of subjects)

  A business process to register a new subject shall clarify responsibilities so that the liability for the registration of a new subject is explicitly expressed.
- pre-defined subjects and groups: statement about whether the OSN requires the existence of specific pre-defined subjects and groups.

11.1.5.2 Authentication Policy

The Authentication Policy deals with the generation of session information. It is defined by the following elements:

- set of allowed authentication mechanisms
    - default authentication mechanism
    - restrictions on the set of allowed authentication mechanisms
- representation of principals: specification of how principals are represented in an OSN (optional)

  Note: Even though the set of allowed authentication mechanisms determines the possible presentations of principals. It may be required for clarity to explicitly specify the representations of principals.
- single-sign-on or multiple authentication: statement whether single-sign-on and/or multiple authentication is used.
- treatment of session information: definition how session information is treated, either by a session key or by a session envelope
- session key validity: validity space for a “session key” returned by the Authentication Service after a successful authentication has to be assured

11.1.5.3 Authorisation Policy

The Authorisation Policy deals with the way the access to resources in an OSN is controlled. It is defined by the following elements:

- set of allowed authorisation paradigms (e.g. role based access control, trust management)
    - default authorisation paradigm for the whole OSN, i.e. for all OSIs of the OSN
    - authorisation paradigms that shall be applied for OSIs of a given service type or for individual OSIs
- default permissions for pre-defined subjects and groups
- policy enforcement: statement about whether the authorisation takes place on the service level and/or the data level.

11.2 OSN Types

In order to characterise OSNs and to provide constraints upon them for their classification into OSN types, the policies described above are structured into policy elements. Depending on the type of OSN that is to be designed specification of these policy elements is either mandatory or optional.

A preliminary list of OSN types is given in Table 38.

The main ideas are as follows:

- All OSNs shall use “platform as namespace” for the naming policy of OSIs and “OSI as namespace” for the naming policy of feature instances. These two policy elements are explained in section 11.3.
- For a “primitive OSN” there are no further constraints or rules, i.e. it may consist of an arbitrary network of OSIs as long as these OSIs have been designed according to the rules of the OMM.

  Note: As primitive OSNs do not necessarily support means for resource discovery, they do not, as a whole, comply with the architectural requirement of “self-describing components”, see section 6.3.7. However, for ORCHESTRA Applications with poor requirements on flexibility this may be a reasonable solution. Nevertheless, as ORCHESTRA aims at supporting environments that are “designed for change”, the question whether a “primitive OSN” should be supported as a “valid” OSN type or not is an ongoing discussion.
- A “mediated OSN” requires the usage of at least one catalogue OSI (called OSN catalogue) for the discovery of service and feature instances.
- A “managed OSN” is a “mediated OSN” that supports, in addition the policy element of “service monitoring” (see section 11.1.4.2).
- An “access-controlled OSN” shall support a harmonised approach for the UAA policy elements as described in section 11.1.5.
- A “secure OSN” is an “access-controlled OSN” that provides, in addition, policy elements for service monitoring (see section 11.1.4.2) and encryption of communication. As encryption of communication is currently not specified in the RM-OA, this OSN type is for later enhancement.
- One OSN may be of several types, e.g. there may be a mediated, managed and access-controlled OSN.

OSN Type	Resource Naming	Resource Discovery	OSN Operating	UAA
Primitive	Platform as namespace for OSIs, OSIs as namespace for feature instances			
Mediated	dito	OSN Catalogue		
Managed	dito	OSN Catalogue	Service Monitoring	
Access-controlled	dito			Harmonised UAA approach
Secure	dito		Service Monitoring Encryption of communication	Harmonised UAA approach

Table 38: Minimum Policy Requirements according to OSN Types

11.3 Naming Policy Examples

11.3.1 Platform as Namespace for OSIs

In the following a naming policy approach for OSIs is presented wherein the assignment of a name to an OSI is independent of the membership of an OSI in an OSN. In particular, a unique OSN name and an OSN-related namespace are not required for this approach.

According to the ORCHESTRA Architecture, an OSN is designed to be based on one or several service platforms. A service platform provides the basic communication and encoding mechanisms for the service interactions (the service infrastructure). By definition, an OSI is the result of a platform-specific deployment step making the OSI part of a certain platform domain. Thus, an OSI can be considered a service in the sense of the used service platform.

One of the characteristics of a service platform is that a service is identified by means of a platform-specific service identifier which is unique within the platform. The identifier is usually assigned when the service is deployed, i.e. entered into the platform. The service platform acts as a namespace for OSIs.

Service Platform Examples:

- Platform W3C Web Services: An OSI corresponds to a Web Service according to the W3C specifications. A Web Service is identified by a URI. A URI is a globally unique identifier for all Web Services.
- Platform Java RMI: An OSI corresponds to a Java Object which is remotely accessible and published in an RMI registry. The Java Object is identified by a URI (with an empty schema), i.e. a string of the form

  //<host>:<port>/<name>

  where <host> and <port> are used to locate the registry. <host> is a hostname (IP-Address or domain names according to DNS) and <port> is the host-specific port number. <name> is the published name of the Java Object which is unique within the registry.
- Platform CORBA: An OSI corresponds to a CORBA Server Object. In CORBA objects can be uniquely identified by an IOR (Interoperable Object Reference). Another way is to use the address of a Name Service and a name local to the Name Service in a similar way as for the Java RMI example.

In the current RM-OA version, it is assumed that a given OSN is based on just one pre-selected service platform. Thus, the service identifier of that platform can directly be used to name the OSIs. As the service identifier is unique within the platform and only one platform is used, the resulting OSI names are unique.

RM-OA version 4 will consider an enhancement of this naming policy for the case that an OSN spans several platforms.

11.3.2 Feature Access OSI as Namespace for Feature Instances

In the following a naming policy approach for feature instances is presented wherein the assignment of a feature instance identifier is combined with the identifier of the Feature Access OSI that provides access to the feature instance.

Thus, a feature access OSI manages a namespace of feature identifiers. The feature identifiers provided by such an OSI are initially not unique within the whole OSN, but only unique among all features of that OSI. In general there may be multiple Feature Access OSIs in an OSN. In order to obtain unique identifiers, the name of the corresponding feature access OSI is added in order to get a unique identifier within an OSN.

Figure 43 provides an example: Three feature access OSIs are backed by different source systems. OSI a and OSI b are part of one OSN, OSI b and OSI c are part of another OSN. All feature instances related to these OSIs have identifiers f1, f2, f3 which are unique for each OSI. By adding the OSI names, the resulting feature identifiers a:f1, b:f1, c:f1 etc. become unique within the OSNs. They are even globally unique because the OSI names are globally unique.

This naming policy for the identification of feature instances is summarised as follows:

Within an OSN, each OSI that acts in the role of a feature access service shall be responsible for managing a namespace of related feature instance identifiers. Each such OSI shall assign identifiers to feature instances which are accessed using that OSI. Such an identifier shall be combined of two elements:

- the OSI name
- an OSI-specific identifier which unambiguously identifies the feature instance among all other feature instances of that OSI.

Together these elements form a feature identifier which is unique within the OSN.

Note: The naming policy just described only ensures the uniqueness of feature instances in an OSN regardless of their real-world phenomenon that they are representing. The situation in which two feature instances provide a (possibly) different view upon the same real-world phenomenon (e.g. a road) is a question of semantic identity that is to be solved on the semantic level of the information model framework (see Figure 20), possibly based on inferencing about an ontology and/or a knowledge base of the respective thematic domain.

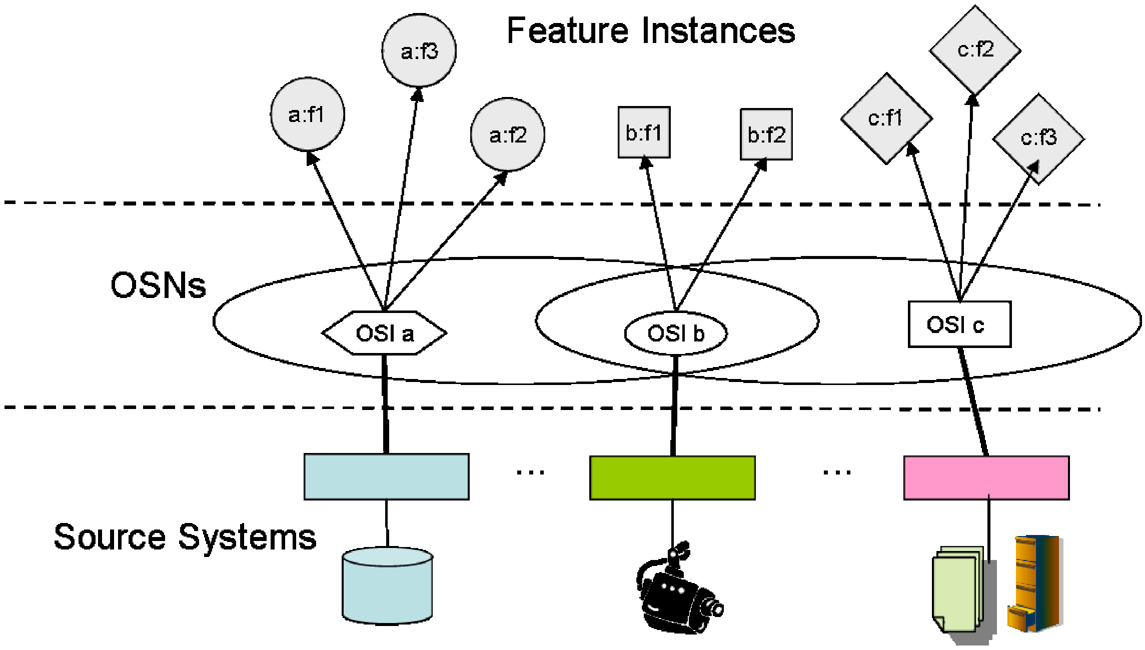


Figure 43: Constructing feature identifiers by using OSI-related namespaces

Constructing feature identifiers according to this rule has the following consequences:

- As each OSI name is globally unique as described in the previous section, the feature identifier is also globally unique.
- If the *createFeature* operation of the *FeatureAccessService* interface (see section 9.6.1) is used to create a new feature instance, the respective Feature Access OSI must assign a unique (i.e. not yet used) feature identifier to it.
- The feature identifier can act as a locator for the feature. The OSI used to access that feature can be obtained from the feature identifier. A client requesting attributes of the feature can therefore direct its request to that OSI. In the same way as a uniform resource locator is used in the Web to locate a resource, a feature identifier is used to locate a feature instance within one or multiple OSNs.

The way a feature access OSI assigns identifiers to its feature instances is not further specified. In order to simplify the mapping between feature identifiers and the underlying feature information, certain feature type-dependent key attribute values may be used when constructing an identifier. However, this is very much source system dependent.

A feature access OSI may also support version management of features, i.e. it may allow access to various former versions of a certain feature instance. The current version and former versions may exist at the same time. In principle the current version and each former version of a feature instance can be considered separate instances which are implicitly or explicitly associated with each other. All these instances can be distinguished by their identifiers. The way versioning is reflected in the identifiers is not specified here.

Note: The principle of constructing a global identifier by combining an OSI name with an identifier which is unique within the context of that OSI can be used for identifying purposes wherever a globally unique identifier is needed.

12 Conclusion

The present RM-OA Revision 2.0 represents the understanding of the ORCHESTRA consortium about an open, generic and standards-based service-oriented architecture for distributed environmental and risk management applications after the 2nd year of the project's runtime. Its focus is currently on syntactic interoperability whereby the upgrade towards the support of semantic interoperability has been prepared.

The following sections provide

- a summary of the major deviations of the RM-OA Design Decisions from ISO and OGC standards (section 12.1) and
- a summary of the items that are intended to be covered in future versions of the RM-OA.

12.1 Summary of Deviations from Standards

Note 1: Textual changes are underlined.

Note 2: Deviations on the level of service types and abstract interface specifications are not listed here as most of the OGC and ISO service specifications are not provided on abstract level.

12.1.1 RM-ODP Computational Viewpoint mapped to RM-OA Service Viewpoint

In order to highlight the fact that an ORCHESTRA deployment will have the nature of a loosely-coupled distributed system based on networked services rather than a distributed application based on computational objects, the "computational viewpoint" will be referred to as "service viewpoint" in ORCHESTRA.

Rationale: section 5.2.2.

12.1.2 The OpenGIS Service Architecture (ISO 19119:2005)

In the ORCHESTRA Reference Model the distributed computing platform is referred to as the service infrastructure. However, the distinction between IT and GI services of ISO 19119:2005 is not applied for the ORCHESTRA service taxonomy because the ORCHESTRA Architecture (and thus the ORCHESTRA services) shall contain an integrated information model that covers thematic, temporal and spatial aspects.

Rationale: section 5.4

12.1.3 ISO 19101 Service Taxonomy

Workflow/Task services are services for support of specific tasks or work-related activities conducted by humans or software components with a high degree of autonomy (agents). These services support use of resources and development of products involving a sequence of activities or steps that may be conducted by different persons.

Processing services are services that perform computations. These computation might range from the performance of mathematical equations up to large-scale computations involving substantial amounts of data.

Rationale: section 5.4.2

12.1.4 ISO 19119:2005 Requirements for Platform-Neutrality

As part of the engineering viewpoint, the ORCHESTRA platform-neutral models are mapped to a specific service infrastructure context. The resulting platform-specific service models may be defined in UML or in terms of the platform-specific language (e.g. WSDL). However, it is required that a description of their mapping to the corresponding platform-neutral models be maintained. This mapping shall show how the intentions of the platform-neutral specifications are met in the context of the service platform. In order to support interoperability, the reverse mapping back to the concepts in the platform-neutral model must be defined (instead of should be defined).

Rationale: section 5.4.1

12.1.5 ORCHESTRA as Simple Service Architecture according to ISO 19119:2005

- Known service type

All ORCHESTRA service instances are of specific service types and the client may access the service type description prior to calling the service. In the ORCHESTRA Reference Model, a "known service type" is a service type with an externally available description.

Rationale: section 5.4.3

Note: The RM-OA version 3 will contain a more refined assessment if the ORCHESTRA Architecture may be considered as a "Simple Service Architecture" in the sense of ISO 19119 taking into account the latest developments about UAA and service chaining in the ORCHESTRA project.

12.1.6 The ORCHESTRA Definition of a Feature

One basic concept of the RM-OA Information Viewpoint is the feature, where a feature is an abstraction of a real world phenomenon perceived in the context of an ORCHESTRA Application. The ORCHESTRA definition of features explicitly goes beyond geographic features. It includes tangible objects of the real world but also abstractions, concepts or software artifacts (e.g. documents, software components of IT systems) that may have a representation only in software systems. These features may, but need not, have spatial characteristics. The ORCHESTRA understanding of a "real world" explicitly includes these hypothetical worlds or worlds of human thoughts.

Rationale: section 8.2

12.1.7 The ORCHESTRA Meta-Model (OMM)

The OMM is derived from the basic ideas of the ISO 19109 GFM, but it is not a true profile of it. The OMM is an evolution of the ISO 19109 GFM, taking into account additional, ORCHESTRA-specific requirements. In particular:

- The OMM extends the GFM by aspects of services modelling (see the OMM Service Meta-model (OMM-Service) in section 9.2).
- The OMM does not mandate the usage of one particular meta-information model (e.g. ISO 19115) as prescribed by the GFM. Instead, it gives the OSN designer the freedom to specify the meta-information models as required for the various purposes. It only mandates that an application schema for meta-information (OAS-MI) be specified according to the rules of the OMM-Information (see section 8.8).

Rationale: section 8.7

12.2 Evolution of the RM-OA

It is envisaged to tackle the following issues in future versions of the RM-OA (this is a non-binding and non-exhaustive list):

- Extension of the OMM-Service by including aspects of Semantic Web Services, e.g. semantic description of services as part of their meta-information, usage of semantics in advanced versions of ORCHESTRA Service Types (concerned sections: 9.2)
- Investigation of rules for semantic interoperability (concerned sections: 8.8, 9.2)
- Usage and influence of ontologies for the RM-OA Information and Service Viewpoints (concerned sections: 8.7, 9.2)
- Alternative interaction modes in addition to synchronous and asynchronous interactions (if required) (concerned sections: 7.6, 9.2)
- Inclusion of changes during the course of the ISO/DIS 19136 standardization process that

influence the rules for the OAS design (concerned sections: 8.7)

- Relationship between service types and investigation about how uniqueness of service type names can be achieved (concerned sections: 9.2)
- Interoperability across platform-domains: scope of OSNs, communication between OSIs, generic interfaces to ORCHESTRA Services
- Enhancement of OSN Monitoring and Management Services
- Handling of multi-linguality, e.g. a combination of a Gazetteer Service with a Thesaurus Access Service in order to support multi-lingual gazetteers and fuzzy queries based on synonyms.
- Enhancement of the naming policies for the case that an OSN spans several platforms. In particular, the problem of naming of OSIs in dynamic OSN environments (i.e. the membership of OSIs in OSN changes during the lifetime of an OSI) must be investigated (concerned sections: 11.1.2, 11.2).
- Relationships between OSNs (hierarchical, overlapping).
- Refinement of the OSN characteristics and their grouping into OSN types according to the pilot implementation experiences (concerned section: 11.1)
- Support of further cases (e.g. service mediation) in the service mapping specification in addition to the service profile (concerned section: 9.2.9).
- Specification of a dedicated interface type for knowledge inferencing that may be used e.g. in advanced versions of the Knowledge Base Service (see section 9.7.10) and/or the Ontology Access Service (see section 9.7.7).
- Investigation of whether a distinction between feature types in the real and in the hypothetical world is useful, as the conventional understanding (e.g. within OGC) does not follow this approach.
- Extension in order to fully resolve the architectural principles of “self-describing components” (see section 6.3.7). E.g. a future RM-OA version might extend the OMM basic part in order to mandate that a feature instance contains (at least a reference) to the feature type specification, probably as part of its meta-information.
- Integration of (proprietary) UAA solutions that are already implemented in source systems and their environment into the UAA policy of an OSN.
- Need for OA Services dedicated to accounting.
- Decide about further candidates for OMM information types and investigate needs for their specification (see section 8.7.5).
- Process for maintaining and evolving the specification of the OA Services beyond the scope of the ORCHESTRA project.

FP6-511678

ORCHESTRA

Open Architecture and Spatial Data Infrastructure for Risk Management

Integrated Project

Priority 2.3.2.9 Improving Risk Management

Reference Model for the ORCHESTRA Architecture (RM-OA Version 2)

Annex A1

Development Dimensions

Date: 2007-01-31

Revision: 2.0

Start date of the ORCHESTRA project:	2004-09-01
Duration of the ORCHESTRA project:	3 years
Organisation name of lead contractor for this deliverable:	Fraunhofer IITB

Document Control Page

Title	Reference Model for the ORCHESTRA Architecture (RM-OA) D3.2.3: RM-OA Version 2 Annex A1 (Rev. 2.0) Development Dimensions
Creator	Thomas Usländer, Fraunhofer IITB (Ed.) e-mail: thomas.uslaender@iitb.fraunhofer.de
Subject	ORCHESTRA Architecture Design
Description	This document is the annex A1 of the Reference Model for the ORCHESTRA Architecture (RM-OA). It contains a description of the main RTD directions of ORCHESTRA, with short to long term goals, and to relate these goals to the state-of-the-art.
Publisher	ORCHESTRA consortium
Contributor	See RM-OA document
Date	2007-01-31
Type	Text
Format	application/msword
Identifier	ORCHESTRA Portal: SP3 / SP3 Quality Assurance / 09: D3.2.3 / 06: D3.2.3 RM-OA V2 (2.0) – published version
Source	Not applicable
Language	en-GB.
Relation	none
Coverage	Not applicable
Rights	© 2007 ORCHESTRA Consortium The ORCHESTRA project is an Integrated Project (FP6-511678) funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).
Deliverable number	D3.2.1
Audience	☒ public ☐ restricted ☐ internal

Major Revision History

Revision	Date	Sections Changed	Description
1.0	2004-12-23	all	Final draft of D3.2.1 submitted to the QA process
1.10	2005-10-14	all	editorial corrections, update following the ORCHESTRA Annual Technical Review
1.16	2006-10-26	all	Corrections made by the editor after the QA Review of the ORCHESTRA Technical Supervisor
1.17	2007-01-17	all	Change of RM-OA document structure
2.0	2007-01-31	all	Editorial changes for publication

Table of Contents

Figures

none

Tables

A1.1 Overview

The intention of this annex A1 is to describe the main RTD directions of ORCHESTRA, with short to long term goals, and to relate these goals to the state-of-the-art. For this purpose, an extensive discussion took place which ended in the definition of “development dimensions”.

Table 1 thus indicates the intended scope of the ORCHESTRA project with respect to these dimensions and describes in which direction the ORCHESTRA project will push forward the development of solutions. There is one row in the table for each of the dimensions. The columns indicate complexity steps with increasing complexity from left to right.

Semantic interoperability based on	Common understanding	Partial common understanding	No common understanding	
Interpretation based on	Fully structured information	Semi-structured information	Unstructured information	
Navigation / search paradigms	Isolated paradigms	Technically integrated (user integrates semantics)	Fully semantically integrated (system does semantic integration)	
Collaboration	Stand-alone	Intra-agency	Inter-agency	
Collaboration methods	Manually (no system support)	Standardised data exchange	Shared systems	
Workflow support (across network)	Built-in, change only by new version	Formally defined, change by configuration	Ad-hoc, spontaneous (user is doing it)	Intelligent guidance (system is assisting)
Thematic domain interaction	Intra domain	Inter domain		
Scale (# of semantically integrated information systems	Up to 10 / 100	Up to 100 / 1000	Up to 1000 / 10000	More than 1000 / 10000
Overall system adaptability	Through reprogramming	Through fixed mappings	Through dynamic interpreted mapping	Fully descriptive, self reconfiguring

Table 1: ORCHESTRA Development Dimensions

The colours indicate the relation between the ORCHESTRA goals and the state-of the-art[1]:

- “green” means: ORCHESTRA will use well known state-of-the-art solutions
- “blue” means: ORCHESTRA contributions to research
- “white” means: ORCHESTRA long term vision of research (not covered during the project)
- “blue / green” means: there exist (partly) state-of-the-art solutions which can not be used by ORCHESTRA as they stand, i.e. ORCHESTRA has to invest effort to extend them
- “blue / white” means: ORCHESTRA will provide some research contribution to the long term vision
- “green / blue / white” means a combination of all of the cases

The following sub-sections give a refined description of each dimension. Note that the term “user” in this section includes software agents.

A1.2 Semantic interoperability

Even in the case of an existing common understanding there are actually many ongoing initiatives but not yet fully satisfying solutions for semantic interoperability. Only in case of a common data model for the interface between systems can semantic interoperability be guaranteed. If this is not the case, either individual and proprietary solutions provide the interoperability or the heterogeneity is forwarded to the user which means that there is no semantic interoperability on the system level at all. ORCHESTRA will have to develop non-proprietary solutions based on existing initiatives and integrate solutions where partly available.

In the case of partial common understanding, interoperability solutions (i.e. services and tools) in ORCHESTRA will at least be able to

- help the users to identify missing common understanding by documentation of semantics of data and services (through meta-information and ontologies)
- enhance common understanding of users by offering powerful mapping tools to map semantic descriptions (e.g. mapping of ontologies)
- increase general common understanding by initiatives towards standardisation bodies

Partial common understanding is an understanding where some but not all concepts are shared among partners. There could be two users using two thesauri in different languages. There could exist already-defined equivalency relationships among concepts. But there are still concepts which are either not related across the thesauri or even for which no relationship exists.

There are solutions which, though they could be improved, will work when a common understanding is shared. Common understanding would be, for instance, a mutually agreeable communication protocol. Solutions which enable semantic interoperability when only partial common understanding is given do not presently exist.

A long-term research goal out of the scope of ORCHESTRA would be to enable semantic interoperability even if no common understanding is shared.

A1.3 Interpretation

The OSN will integrate many data sources with any kind of data structures ranging from well-structured formats to unstructured formats. ORCHESTRA needs to support the exchange of these different types, their presentation to end users and their access and processing by services.

The typical case of well-structured information is a RDBMS, where the structure itself follows semantic relations.

In the case of semi-structured information, additional meta-information is needed to structure the information sufficiently in order to make correct use of it. For some rather simple examples of this type of semi-structured information (e.g. indexed documents or CSV-files, where all values have known semantics) solutions already exist which ORCHESTRA will have to expand to cover as much as

[1] This classification will be part of a continuous process during the course of the ORCHESTRA project as a result of the technology assessment and the ongoing observation of relevant projects and technologies.

possible the problem of interpretation based on semi-structured information.

Examples of unstructured information include flat files without explicitly and/or implicitly attached meta-information like separators or file or data type definitions, e.g. an unformatted text file.

For the correct and integrated interpretation of this information, independent of the format, the structuring of this type of information by means of meta-information is necessary. This will be a research topic of ORCHESTRA, but it will certainly not be solved in even a nearly complete fashion.

A1.4 Navigation / search paradigms

It must be possible for users of the OSN to perform navigation/search in different “information worlds” with different paradigms for these operations (e.g. spatial and non spatial, documents). The users will need the possibility to switch between these information worlds at any moment maintaining as much as possible the semantic context at each switch.

Today, navigation and search for each information world are often isolated. This means that end users need to perform the transition manually, both in a syntactic and semantic sense.

It is also possible to integrate different navigation/search paradigms in a purely technical sense, but the user has to bridge the semantic connections (e.g. by transporting semantic meta-information manually). The challenge for ORCHESTRA is to integrate navigation and search across these information worlds on both the technical and the semantic level as much as possible.

A1.5 Collaboration

ORCHESTRA intends to provide means to improve collaboration between systems operated by different government agencies, where this is needed for a given purpose. In order to be open and more flexible ORCHESTRA will need to enhance existing solutions for such collaboration. Currently very little collaboration between agencies exists which really operates in a seamless way. Most applications work stand-alone. ORCHESTRA will improve existing solutions used inside agencies (intra-agency, meaning collaboration between different systems in the same agency) and across agencies (inter-agency).

A1.6 Collaboration methods

Existing systems often collaborate either only by human intervention or through exchange of standardised data using shared (technical) protocols.

For collaboration through sharing of systems an OSN will have to offer the possibility of sharing and mapping data to locally defined data models and formats. This simple kind of sharing of data can be done based on existing solutions, for example ETL and mapping tools. Collaboration on the semantic level will raise the level of interoperability from the syntactic (data models and formats, as mentioned before) to the semantic level, which means that a) equivalent concepts can be shared and b) related concepts can be mapped. The mapping of related concepts requires some processing, e.g. given two attributes representing temperatures. one measured in Celsius the other in Réaumur, it would be clear that both concepts are attributes representing a temperature and that both have a unit, but that they are not the same. The mapping would then require a transformation from one unit to the other.

For the sharing of services the situation is more complicated. There exist solutions in the case of very simple services (e.g. offered via the web). Concerning complex services, solutions also exist for sharing in the case of tightly coupled systems (e.g. the sharing of complex mathematical simulation).

ORCHESTRA will also have to develop solutions for the sharing of services supporting semantic interoperability in loosely coupled systems.

A1.7 Business process support (stand alone and across network)

Users working in a distributed environment will be confronted with situations where a spontaneous modification and/or creation of workflows is needed.

Business processes are currently often “hardwired”, so that changes in them can only be accommodated by a new version of the software.

There exist concepts and tools (especially in the “commercial” world) allowing the dynamic creation and invocation of fixed predefined workflows without “programming”, in other words configuration changes are sufficient to realise new business processes.

An OSN will have to be able to support collaboration of dynamically changing workflows even across the network. The problem becomes very complex, especially in the case of workflows, which are defined and initiated by end-users in an ad-hoc fashion. On the basis of existing approaches ORCHESTRA will contribute as much as possible to solutions for this problem.

Outside the scope of ORCHESTRA a long-term goal is the development of solutions for the support of dynamic intelligent adaptation of workflows by the system itself (e.g. in case of temporary unavailability of a service).

A1.8 Thematic Domain Interaction

In the Risk Management domain data and services coming from different thematic domains (e.g. risk management, environmental protection, meteorological forecasting) will have to interact to produce certain workflows. This type of interaction inside and across thematic domains already exists but will be improved by ORCHESTRA. As an application independent infrastructure, ORCHESTRA particularly intends to improve applications and workflows which span different thematic domains.

A1.9 Scale (# of semantically integrated information systems/users[2])

The number of integrated systems which will cooperate in the OSN is expected to become large. The added value and the number of users increase with the number of systems and the “lifetime” of an OSN. Though there is no precise number known it will probably be much larger than in typical federated state-of-the-art systems.

Existing integrated information systems that are to some degree integrated on the semantic level, integrate on the order of 10 systems and 100 users.

To reach the intended added value OSN’s will have to be able to integrate hundreds of heterogeneous information systems and handle thousands of users.

ORCHESTRA will also try to take into account larger scales as much as possible, but the integration of thousands of information systems and tens of thousands of users or more will be out of the scope of the project.

A1.10 Overall system adaptability

The anticipated period of operation of the OSN is longer than typical technological cycles in IT, partly due to the evolutionary character of the OSN. The probability that the system will have to adapt to changed or new requirements increases with its life span. Run-time adaptation (i.e. without reprogramming) will have to be as flexible as possible.

Existing solutions include those requiring reprogramming and those having fixed mappings, e.g. the vast majority of EAI tools rely on software platforms that require a human expert to implement or reprogram adapters and templates or to create/update fixed queries, mappings and transformations, each time a subsystem is added/changed.

One step further in the direction of a more adaptable and thus generic system is, for example, the use of (semantic) meta-information to construct mappings and transformations from local forms into a generic form. This means that such a mapping can become to some extent adjusted and interpreted dynamically. ORCHESTRA will develop solutions based on some existing approaches.

The idea of a fully descriptive, self reconfiguring and adaptable system, which “responds in real time to changing conditions by generating its own instruction sets on the fly when encountering unforeseen circumstances” (Pollock, Hodgson 2004) is subject to long-term research.

[2] X/Y means order of X systems with Y users

FP6-511678

ORCHESTRA

Open Architecture and Spatial Data Infrastructure for Risk Management

Integrated Project

Priority 2.3.2.9 Improving Risk Management

Reference Model for the ORCHESTRA Architecture (RM-OA Version 2)

Annex A2

Requirements for the ORCHESTRA Architecture and ORCHESTRA Service Networks

Date: 2007-01-31

Revision: 2.0

Start date of the ORCHESTRA project:	2004-09-01
Duration of the ORCHESTRA project:	3 years
Organisation name of lead contractor for this deliverable:	Fraunhofer IITB

Document Control Page

Title	Reference Model for the ORCHESTRA Architecture (RM-OA) D3.2.3: RM-OA Version 2 Annex A2 (Rev. 2.0) Requirements for the ORCHESTRA Architecture and ORCHESTRA Service Networks
Creator	Thomas Usländer, Fraunhofer IITB (Ed.) e-mail: thomas.uslaender@iitb.fraunhofer.de
Subject	ORCHESTRA Architecture Design
Description	This document is the annex A2 of the Reference Model for the ORCHESTRA Architecture (RM-OA). It contains a description of the system requirements for the specification of the ORCHESTRA Architecture and the design ORCHESTRA Service Networks.
Publisher	ORCHESTRA consortium
Contributor	See RM-OA
Date	2007-01-31
Type	Text
Format	application/msword
Identifier	ORCHESTRA Portal: SP3 / SP3 Quality Assurance / 09: D3.2.3 / 06: D3.2.3 RM-OA V2 (2.0) – published version
Source	Not applicable
Language	en-GB.
Relation	none
Coverage	Not applicable
Rights	© 2007 ORCHESTRA Consortium The ORCHESTRA project is an Integrated Project (FP6-511678) funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).
Deliverable number	D3.2.3
Audience	☒ public ☐ restricted ☐ internal

Major Revision History

Revision	Date	Sections Changed	Description
1.0	2004-12-23	all	Final draft of D3.2.1 submitted to the QA process
1.10	2005-10-14	all	editorial corrections, update following the ORCHESTRA Annual Technical Review
1.16	2006-10-26	all	Corrections made by the editor after the QA Review of the ORCHESTRA Technical Supervisor
1.17	2007-01-17	all	Change of RM-OA document structure
2.0	2007-01-31	all	Editorial changes for publication

Table of Contents

Figures

Tables

A2.1 Requirements for the OSN and the OA

In this section a line of argument is set up to define the requirements for the OSN and the ORCHESTRA Architecture.

For the purpose of this section the OSN is often simply referred to as the “system”.

The line of argument starts by describing the different types of *users* of the system and their *roles*. These *user roles* are connected with *fundamental challenges* which are considered relevant to the system. These fundamental challenges lead to *key system requirements* and finally to *architectural principles*. These steps are illustrated in Figure 1-1.

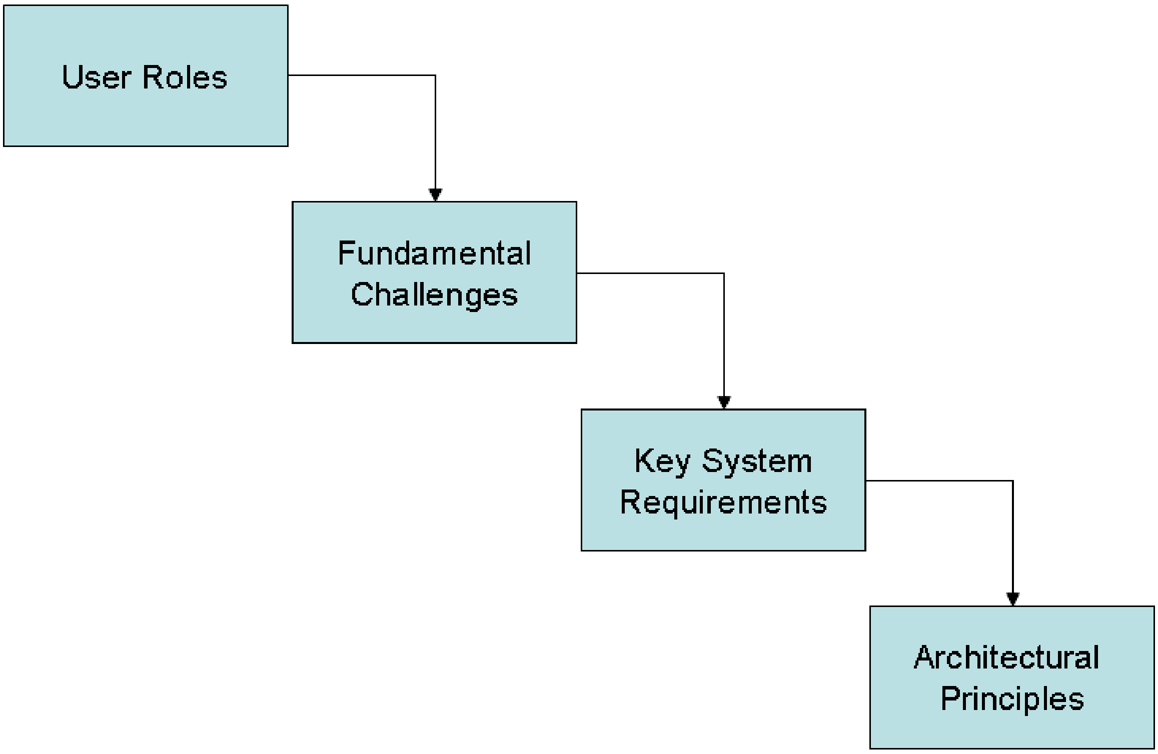


Figure 1-1: Line of argument to find the system requirements

Fundamental challenges are those major sets of challenges which the ORCHESTRA Architecture has to cope with. Architectural principles are derived from these challenges and form the set of major constraints for the architecture.

The user roles, fundamental challenges, key system requirements and architectural principles are identified in the following sub-sections and are summarised in

Table 1. The table indicates the relationships between the different elements. The matrix in the upper left part of the table connects user roles to fundamental challenges. These categories are then linked to key system requirements by the matrix in the upper right part of the table. Finally, the matrix in the lower right part of the table relates these key system requirements to architectural principles.

These relationships are described in the following sub-sections. The description of each element is complemented by a separate table indicating the dependencies of the element to the related elements of the previous and subsequent step in the line of argument.

The main purpose of the argumentation chain is to formulate a foundation for the architectural decisions which lead to improved interoperability between systems. Although considered as very important depending on the application field of an OSN (e.g. in the response phase of disaster management), aspects of security and dependability are not discussed in this section.

These aspects will be considered in a later version of the RM-OA.

Note 1: In the following sub-sections, all occurrences of terms appearing in

Table 1 are marked in italics in order to emphasise their dedicated meaning.

Note 2: All entries in the requirement tables point forward and backward in the argument chain. In case of the backward pointing, one has already read the definition of a term. In case of the forward pointing one will find the definition of a term in a later section of the document.

User Roles			Fundamental challenges	Key System Requirements			
Service Developer/System Administrator	Service provider	End user		Openness	Scalability	Usability	Accountability
✓			Scale and Scope	✓	✓		
	✓	✓	Integration/Collaboration	✓			
	✓	✓	Long Lifetime	✓	✓	✓	
	✓	✓	Quality		✓	✓	✓
✓	✓	✓	Transparency (Hidden Process Complexity)			✓	
	✓		Access Control		✓	✓	✓

Architectural Principles	Openness	Scalability	Usability	Accountability
Rigorous Definition and Use of Concepts and Standards	✓			
Loosely Coupled Components	✓	✓		
Technology Independence	✓			
Evolutionary Development - Design for Change	✓	✓		
Component Architecture Independence	✓			
Generic Infrastructure	✓		✓	
Self-describing Components			✓	✓

Table 1: ORCHESTRA System Requirements

A2.1.1User Roles

In the field of Human/Computer Interaction (HCI) system requirements are identified by using a user-centric approach. Three roles of users can be identified:

- System users such as
  - Service developer/system administrator (secondary user in HCI terms)
  - Service provider (tertiary user in HCI terms)
- End user (primary users in HCI terms)

A2.1.1.1 Service Developer/System Administrator

The *first user category* includes two types of users:

- service developers and
- system administrators.

The first type, a *service developer*, usually gets assignments from a *service provider*. These assignments usually have the following goals or combinations thereof:

- implementation of new services
- update and maintenance of existing services
- provision of new data/services
- publishing sources of data/services

An example for such a user would be a system integrator who connects a new data source (e.g. a database containing water-level measurements). This activity includes the production registration of a technical and semantic description of the data source in an ORCHESTRA comprehensive way.

Another example could be a service developer who implements a new service by chaining already existent services in order to

1. locate semantically fitting data sources, e.g. water level measurements satisfying the input needs of a specific simulation model, then
2. (if needed) transform that data (e.g. between different measures: millimetres to meters) and
3. feed it to that simulation model and launch execution,
4. provide the model's output data, and consequently
5. provide the data to the end user via a service which adequately represents the data.

The second type is a system or network administrator. This person is required to maintain network interaction between nodes involved in an OSN. To do this they must have access to information about the location of data and services running in an OSN.

User Role: Service developer/system administrator		
	Scale and Scope	The service developer/system administrator has a

→ Fundamental challenges	Scale and Scope	natural interest in two Fundamental challenges: Scale and Scope, and Transparency. Their interest in Scale and Scope results from the fact that the size of the problem will impact both service developers and system/network administrators as they attempt to construct and manage an OSN. Their interest in Transparency results from the sheer complexity of the processes required to support an OSN. Therefore, increased transparency facilitates better management of developed systems. For instance if a system administrator might want to shut down a system for maintenance, some monitoring service should be informed of the planned maintenance, so that appropriate messages could be generated. The administrator simply wants to shutdown the system for maintenance and not be bothered with detailed information on dependencies between services.
→ Fundamental challenges	Transparency (Hidden Process Complexity)	natural interest in two Fundamental challenges: Scale and Scope, and Transparency. Their interest in Scale and Scope results from the fact that the size of the problem will impact both service developers and system/network administrators as they attempt to construct and manage an OSN. Their interest in Transparency results from the sheer complexity of the processes required to support an OSN. Therefore, increased transparency facilitates better management of developed systems. For instance if a system administrator might want to shut down a system for [redacted] e messages could be generated. The administrator simply wants to shutdown the system for maintenance and not be bothered with detailed information on dependencies between services.

A2.1.1.2 Service provider

The service provider typically uses existing data sources and services to provide an integrated service.

For example, there may be different data sources and different models available for a concerned region endangered by flooding. The service provider wants them to interact to get a better result, e.g. he wants the output data of the French flooding model to be used as input data for the German flooding model, and he also wants German water level measurement data to be used.

	Integration/Collaboration	
	Long Lifetime	
	Quality	
	Transparency (Hidden Process Complexity)	
	Access Control	

A2.1.1.3 End user

End users are decision makers (in the risk management domain) who base their decisions upon information retrieved by use of an OSN. In most cases, but not exclusively, they interact with OT Services.

An example of an end user would be a decision maker assessing the risk for flooding in a given region, who is using an integrated service to get the needed information; additional examples would be civil protection authorities, land use planners, rescue teams, the general public and so forth.

User Role: end user		
→ Fundamental challenges:	Integration/Collaboration Long Lifetime Quality Transparency (Hidden Process Complexity)	End users need to focus on the application. Thus, they should not be concerned with problems associated with Integration/Collaboration of the data and services used in the application. They will expect a Long Lifetime of an OSN and will benefit as the OSN grows and becomes richer. They will expect high Quality to be assured that the decisions they reach using the system are well-founded. Integration/Collaboration are preconditions to provide the Transparency required by the end user so that they can do their work in what appears to be a seamless environment.

A2.1.2Fundamental challenges

This subsection describes fundamental challenges which are derived from the expectations and needs of system users as well as well-known experiences from former projects and common practice. The motivation for these categories is user driven and should show that the development addresses all of the identified user groups.

For each identified fundamental challenge a link to those key system requirements derived (partly) from this category is given along with a link to the corresponding user roles. The derived key system requirements are described in more detail in the subsequent subsection.

A2.1.2.1 Scale and Scope

The problems to be addressed by the ORCHESTRA Architecture are large in two important respects. On the one hand, they might involve a large number of heterogeneous elements (such as users, data and models). On the other hand, each such element may itself be large in size. The former is referred to here as *Scope*, while the second is called *Scale*. For example, an OSN might involve a large variety of disparate data sources (scope), each of which might have a large number of data points (scale).

Fundamental challenge: Scale and Scope		
← User Roles:	Service Developer/System Administrator	The service developer/system administrator has to be enabled to cope with the problem size.
→ Key System Requirements:	Openness Scalability	The Scale and Scope fundamental challenge naturally implies the requirements of Openness, and Scalability in order to achieve the overall goal. Large and comprehensive systems cannot persist without Openness and Scalability of the constituent elements in order to facilitate their growth. Openness means expandability, manufacturer neutrality and the obligation to a publicly accessible standardisation process.

The size of a system matters, especially when the vision is a huge system consisting of thousands of participating systems. The complexity of using and managing the system may or may not grow proportionally to the scale.

An example of such a problem would be the processing of a search request. As long as the information to search for does not exceed a specific size it would be sufficient to have one centralised server for it, but if it is larger than that size it becomes necessary to have a distributed processing facility. Another such example would be to consider a problem in which certain data are used. The addition of one new variable could significantly increase the complexity of the solution if it requires accessing data from a new and difficult source.

The number of the following types of elements may increase and may influence the complexity of an OSN considerably:

- Number of autonomous systems

  Information systems (IS) as data or service sources necessary for building an OSN are operated by autonomous stakeholders (e.g. institutions or departments). These information systems are in most cases solely under their local control and responsibility. The number of systems participating can change (grow) at any time and is expected and intended to become large, where large means thousands of autonomous systems. It is obvious that the number of systems integrated by the OSN will be larger than in typical federated state-of-the-art systems.

  Autonomous systems in this context means:

  - Each system is under control of one or more different bodies.
  - Singular systems can be switched off totally or partially without consideration of the impact this can have on the OSN.
  - In general these systems have local users using local applications that were implemented independently. They remain autonomous with respect to these applications and users.
  - The OSN cannot impose any restrictions or rules on the existing systems and local applications.
  - The existing systems and their local applications may be maintained and modified without considering any impact on the OSN.

- Number of concurrent users

  The number of users concurrently using the OSN is expected to be rather large, and is essentially unlimited.

  Factors influencing the number of users include:

  - the number of institutions (data or service providers) participating in an OSN,

  - their number of end users (authorities, public, ...)
  - the number of systems they operated (meteorological systems, earth observation systems, cadastre systems, ...) and connect to the OSN.

- Number of collaborating services

  For each task (or sequence of tasks) a different set of information systems may need to work together to provide a collaborated service. *Collaborated services* are built by chaining or orchestration of services.

  The number of collaborating services may become large for some typical use cases (e.g. flooding for major river basins crossing borders).

- Variety of information and functionality

  Information and services provided by participating systems may vary heavily according to information (syntax, semantics, amount) and functionality.

- Number and variation of terms used in different systems

  The absolute number of terms is large, and this problem is exacerbated by the number of terms for a specific concept.

- Number and size of data sources

  The number of data sources, as well as the volume of data to be handled, can both become large, e.g. in the case of

  - time series of measurement values,
  - spatial data (geo-referenced objects), or
  - imagery data.

  The exchange of such large amounts of data between services and data and the processing (e.g. by a simulation model) of large data sets can be very time consuming.

A2.1.2.2 Integration/Collaboration

Integration/Collaboration means the assimilation of information and methods from different disparate autonomous information systems into a single seamless system.

Fundamental challenge: Integration/Collaboration		
← User Roles:	Service provider	The service provider wants to integrate data and services to provide new (value added) services.
	End user	Decision makers want to work on a semantic level and do not want to be bothered by problems arising from different terminologies and languages that are used by different users of an OSN.
→ Key System Requirements:	Openness	Integration/Collaboration is extremely difficult, if not impossible, to achieve with closed component systems; therefore, Openness is a requirement key to achieving Integration/Collaboration.

Experiences of the past show us that integration is expensive, especially when accomplished by implementing an individual interface for each additional system to be integrated.

Concerning integration/collaboration the following aspects have to be considered:

- Semantic Interoperability

  Semantic interoperability is to be supported. In order to achieve semantic interoperability a number of problems have to be addressed, including:

  - Different conceptual models of the world

    Because different conceptual models of the world exist (e.g. in different organisations dealing with the same real world objects), it shall be possible to combine them and to merge information in terms of different conceptual viewpoints.

  - Different terminologies/languages

    Different terms can arise inside a language and in addition across different languages. The terminology problem is not only a problem of multi-linguality. The problem of multiple terms for semantically identical or similar things in different information resources is even harder to solve and not yet well understood.

- Fragmentation/Heterogeneity

  Heterogeneity refers to the mixture or combination of different information types and/or methods within a location. *Fragmentation* refers instead to the distribution of similar information types and/or methods over multiple locations. These two issues include:

  - There are many different types of heterogeneous data sources which are used in risk management, e.g. maps, databases or flat files. This heterogeneity exacerbates the integration of all of the existing data sources in risk management.

  - Information resources are fragmented and spread over many levels of administration. Boundaries between information sources include geographical, organisational and legal boundaries.

  - There are no general-purpose navigation, search and access methods helping end users to find and access data.

  - Currently, existing geographical information is fragmented, duplicated and difficult to identify, access and use.

  - Spatial and non-spatial information resides in two different "information" worlds and technologies which are not well integrated. There is no common systematic approach by which spatial and non-spatial information and computation services may collaborate.

  - Traditionally, geographical information has been a specialised activity organised by individual national states and professions.

  - Work paradigms are heterogeneous. Many tasks in risk management have different work paradigms. An example would be the search and navigation in maps, databases, catalogues and even within documents. Sometimes it is necessary to explore maps and documents or search databases and documents or browse all the combinations of all possible data sources.

  Fragmentation and heterogeneity has various aspects, e.g.:

  - Geographical Borders

    Integration of spatial data across geographical borders shall be supported. The problem is combined with the problem of organisational borders. It has technical as well as semantic aspects. One example would be when maps from different creators are to be matched at some border.

    Such semantic differences may exist due to legislation (e.g. different threshold values, different standard workflows for identical situations etc.).

  - Institutional/Organisational Borders

    Collaboration across different institutions and organisations shall be supported. In particular, different languages, legislations, terminologies and semantic concepts are some of the major problems which even arise between two similar organisations or within one organisation.

- Interfaces

Open interfaces, which allow one to search and navigate across system-borders, are not available in most cases. If interfaces exist, they are proprietary and thus heterogeneous.

- Application Domains

Applications of different domains need to be integrated and the collaboration of applications across different application domains shall be supported.

- Incompatibility

Applications within a domain or across domains may incorporate data of various dimensionality, specifically involving 1, 2, 2.5, or 3 spatial dimensions, and either considering the temporal dimension or not.

Applications within a domain or across domains may incorporate data using various units of measure or coordinate systems which must be harmonised when used together.

A2.1.2.3 Long Lifetime

An OSN is a system which needs to operate over a long period of time. The anticipated period of operation of an OSN is longer than typical technological cycles in IT partly due to the evolutionary character of the OA.

Fundamental challenge: Long Lifetime		
← User Roles:	Service provider	Service providers want to protect their investments.
	End user	A long lifetime is important to end-users because the usefulness of an OSN is expected to grow over time.
→ Key System Requirements:	Openness	For the system to have a Long Lifetime, its constituent elements must be Open to enhancements and modifications, in other words adaptable. They must also be Scalable, since the system will surely expand as time goes on. Finally, the system must be Usable if it is to last.
	Scalability	
	Usability	

The following aspects related to *Long Lifetime* are to be considered:

- Dynamic behaviour of components

  Connected components may be expected to undergo (as yet unforeseen) changes of their behaviour.

  Changes may occur in

  - format,
  - information quality,
  - content,
  - semantics and
  - workflows.

  The problem will consist of allowing on one hand these autonomous changes, but limiting on the other hand their effects on the OSN. In the ideal case no administrative action in the OSN should even be necessary. Otherwise the system developer should be provided with tools to cope with the problem of participating systems changing over time.

- Technical, financial and organisational aspects
  - technical

    Technology life cycles are very short, e.g. middleware technologies have changed every 3-5 years in the past. Therefore the OA has to be independent of even the most modern technologies, to ensure its future adaptability.
  - financial

    Very large systems are usually expensive to *set up*, *to maintain* and to *integrate,* which leads to the need for investment security. This also implies that it should be possible to implement billing services inside an OSN.
  - organisational

    Organisational structures (responsibilities and capabilities) will change during the lifetime of an OSN (e.g. elections, creation of new departments, new scientific institutions etc.).

A2.1.2.4 Quality

There is need for a service to support the distribution of quality information. Therefore the ORCHESTRA Architecture should provide a model which addresses confidence. A quality situation such as the one on the World Wide Web (in which information quality is not generally known) is not acceptable for an OSN. Levels of confidence need to be attached to data, services, providers, etc.

Fundamental challenge: Quality		
← User Roles:	Service provider	Service providers may want to have control of what happens with their services and data, and will want to know the quality of data and services originating from other providers.
	End user	Different aspects of trust and quality need to be expressed. That is because end users must be able to determine if available data and services satisfy their needs for trustworthiness and quality.
→ Key System Requirements:	Scalability Usability Accountability	A system must also be able to bear expansion in both size and scope without degradation of quality. Only systems which are highly Usable can be described as being of high Quality. Finally, in order to offer assurances to system providers and users, the system must provide Accountability with respect to access to and modification of data and/or services. Decent degrees of security, safety, robustness and accuracy are needed, so that a system (data/service) provider can offer assurances, for which the provider can be made accountable.

The following aspects of quality are of importance:

- Quality Measures

  Information quality is a vital issue for risk management. For different use cases the meaning of quality can differ. There can be use cases in which the best data are the most recent data available, while in other cases the best data can be those with the highest resolution. Thus, the way to express a measure for data quality is use case dependent.

Quality may concern requirements of

- time (age/currency of data, response time, etc.)
- accuracy/error/bias
- completeness of search results - some legislations require a 100% result-set of the search, which can only be guaranteed for specific search-spaces. In some cases it may be sufficient to provide a reduced result-set.

- Levels of confidence

  Confidence in information and/or models implies trust in

  - the originator of the data or service
  - the provider of the data or service
  - the transmission system
  - the service chaining/integration process(es)
  - the users' own selection of the particular data or service

A2.1.2.5 Transparency (Hidden Process Complexity)

In human/computer interaction, computer *transparency* is an aspect of user friendliness which prevents the user from worrying about technical details (like installing, updating, or downloading).

The high complexity of collaborating tasks is inherent to distributed systems. This complexity has to be hidden from the users in degrees depending on the user role. The *end user* wants fully transparent access when using the OSN to make decisions, whereas the *service provider* needs less transparency (e.g. failure logs), finally the *service developer/system administrator* needs the least transparency.

Fundamental challenge: Transparency		
← User Roles:	Service Developer/System Administrator	Service developer/system administrators are confronted with process complexity, which implies the need to hide this complexity from the service provider, the end user and where possible even from the service developer/system administrators, in other words, to provide transparency regarding access, location, persistence and transaction.
	Service provider	An OSN has to provide tools for system managers, in particular for data providers • to *easily and cheaply integrate* their system, including *legacy systems* into an OSN • to *easily monitor and manage* their participation in an OSN, so that it does not become a burden for them • to *easily scale* how their existing systems link into an OSN, in case their organisational structures change.
	End user	It is required that end users can seamlessly search, navigate and access information across different existing systems and seamlessly access and use services offered by other organisations.

→ Key System Requirements:	Usability	If the OSN is to be transparent, it must support a transparent user interaction, and therefore be highly Usable.

A2.1.2.6 Access Control

Organisations are reluctant to grant data access to other organisations, even within the same government. One technical reason for this is that there are no common strategies and technical solutions for handling access privileges across organisational borders within loosely coupled systems in a practical, transparent and reproducible way.

Fundamental challenge: Access Control		
← User Roles:	Service provider	Service provider wants to be able to control who has access to their data or services.
→ Key System Requirements:	Scalability	Access Control must be robust with changes in the number of users as the system is Scaled up. Such control must be highly Usable so that legitimate users of data and/or services are able to use the system appropriately and readily. And the system must be Accountable for accesses to data/services and able to report on these.
	Usability	
	Accountability	

Access control is related to Authorisation and authentication:

- Authorisation

  Authorisation refers to the granting of permission to users and/or other systems to access data and/or services through the OSN.

- Authentication

  Authentication refers to the determination that a user and/or systems presenting themselves for access to data and/or services are indeed authorized for such access.

A2.1.3Key System Requirements

This subsection describes key system requirements which have been derived from the fundamental challenges identified in the previous subsection. Links connect the fundamental challenges with key system requirements which derive from them. Additional links lead to architectural principles of these key system requirements which are described in the subsequent subsection.

A2.1.3.1 Openness

Within ORCHESTRA, the term “open” means that architectural specifications are vendor-neutral, publicly available and free of charge.

The ORCHESTRA Architecture has to be open to overcome fundamental problems such as *integration* of data, services and applications. If, for example, OA specifications were not publicly available, a wide spread usage of concepts, tools and services would be unlikely if not impossible.

Existing *de facto* standards created by industry, research or administrative consortia (e.g. OGC, W3C, OMG, IEEE), and *de jure* standards created by official bodies (e.g. ISO, CEN) will be a basis for ORCHESTRA activities.

Key System Requirement: Openness		
← Fundamental challenges:	Scale and Scope	Large and growing numbers of systems cannot be maintained if their components don't have *open* interfaces.
	Integration/Collaboration	The system has to be *open* to ease integration/collaboration across different: • Organisational Structures • Technologies • Data Source Types • Thematic Domains • Semantics
	Long Lifetime	The anticipated long lifetime of the system is enhanced by the system's *openness for change* of • Application Requirements • Information Flows
→ Architectural :	Rigorous Definition and Use of Concepts and Standards	Openness can best be achieved by the wise use of state-of-the art, yet widely accepted, Concepts and Standards. Loose Coupling of Components often facilitates Openness. To remain Open over time, the system must maintain independence from particular technologies. *Evolutionary Development* will be considerably more likely if the system is Open. The architecture must also remain independent of existing information systems in order to remain *open*. Finally, a *Generic Infrastructure* will greatly facilitate the Openness of the system.
	Loosely Coupled Components	
	Technology Independence	
	Evolutionary Development - Design for Change	
	Component Architecture Independence	
	Generic Infrastructure	

Openness is characterized by flexibility and extensibility as follows:

- Flexibility

  Flexibility is the ability of the OSN to change, and to adapt to changes in requirements, organisations, and technologies over time. The following types of changes may be relevant:

  - Change of application requirements

    An OSN must be able to adapt to changes in the requirements of the applications which use it.

  - Change of organisational structures

    Since the OSN is expected to operate over a large period of time, organisational structures (such as people, resources, aspirations, market trends, levels of competence, reward systems, and departmental mandates) may be expected to change.

    For example, this may include:

    - Splitting/combination of organisational structures
    - Attribution of new responsibilities

- Modification of hierarchical dependencies

An OSN should be able to accommodate these changes.

- Change of technologies

The OA has to be *open* to changes of underlying technologies. Examples of such technologies are implementation technologies (such as programming languages), integration technologies (such as middleware), and communication technologies (such as networks).

- Change of information flow

The representation of information flows is critical within an OSN. Information is exchanged inside an OSN between interoperating systems. An information flow is a series of information exchanges, and takes place to accomplish a collaborating task. It is expected that collaborating tasks change, hence information flows will also change, and it must be possible to adjust to these changes.

- Extensibility

Extensibility is the capacity of an OSN to be extended through the addition of new data sources, data source types, services and applications. It also refers to the potential for the OSN to address other thematic domains.

- Data Sources and Services

The OSN should facilitate the addition of new data sources of various types (e.g. databases, flat files, document stores, Web sites, etc.).

- Services

The OSN should facilitate the addition of services of various types (e.g. processing services, map services)

- Applications

The OSN should accommodate new applications which use the OSN to access data and services.

- Alternative Thematic Domains

The primary thematic domain of ORCHESTRA is environmental risk management. But the participating and offered systems and services are not necessarily bound to that domain. In particular, the OA Services are not bound to a specific thematic domain but claim to be generic or at least of generic use.

A2.1.3.2 Scalability

Scalability refers to how well the OSN will function when its size increases. The system has to be scalable in terms of:

- number of autonomous systems
- number of concurrent users
- number of collaborating services
- number and size of data sources

Key System Requirement: Scalability		
← Fundamental challenges:	Scale and Scope	Sustainability can be reached by a scalable system, where scalable means that it must be able to reliably accommodate future increases in size.
	Long Lifetime	The system has to be sustainable over a long period of time, during which the demands on the system can be expected to grow. Therefore, scalability is important for a long lifetime.
	Quality	The quality of the OSN should not degrade as the system grows.
	Access Control	The number of users which may be managed by the access control system should not hinder the growth of the system.
→ Architectural :	Loosely Coupled Components	To achieve a scalable system it is reasonable to build it with loosely coupled components.
	Evolutionary Development - Design for Change	The requirement of Scalability cannot be achieved with a One-Step-approach in the architecture and development of the system. Different aspects of the anticipated OSN can be tackled independently due to this evolutionary development: • number of autonomous systems • number of concurrent users • number of collaborating services • number and size of data sources

A2.1.3.3 Usability

Usability facilitates the users' access to the system. Because there are different user roles, the usability of the system is categorised according to the different users' expectations and needs.

Service developer/system administrator (e.g. in the role of a system integrator):

- *Easy to understand*

  The OA should be easy to understand and to learn for its users.

- *Easy to remember*

  Once a user has understood/learnt the system, they should be able to reuse it easily after a period of no use.

- *Easy to integrate*

  Little effort is needed to combine systems and services in the anticipated OSN into an overall system.

- *Easy to maintain*

  The system shall be manageable, no matter what technical obstacles/developments exist.

Service provider:

- *Easy to use*

  There should not be a high technology barrier to coupling existing systems into an OSN.

- *Easy to maintain*

  This matters for both service developers/system administrators and service providers. Because maintenance will be one job of service developer/system administrators and, if it’s not too hard, service providers will maintain their part of the system on their own. This leads to cheaper maintenance and thus enhances the overall system’s acceptability and long term sustainability.

End user:

- *Transparency*

  It is required, that the *end user* does not need to care about technical details to solve his problem, instead he should be able to work on the semantic description of the particular problem (cf. “Fundamental challenge/Transparency”).

  It should be easy for users of this role to switch between different “information worlds”, while maintaining the semantic context at each switch.

  The user has to be able to switch and switch back at any time between

  - visualizing (query, explore) records of non-spatial data
  - browsing documentation
  - performing spatial analysis
  - querying spatial and non-spatial objects

Key System Requirement: Usability		
← Fundamental challenges:	Long Lifetime	OSN will not live long if it is not usable.
	Quality	The OSN cannot be said to be of high quality unless it is highly usable.
	Transparency (Hidden Process Complexity)	Transparency of the OSN (especially to the RM application user) dictates a transparent user interaction design.
	Access Control	The degree to which a user's needs are met by the system will be constrained in part by their access to data and services which may be provided by the system.
→ Architectural :	Generic Infrastructure	A widely used Generic Infrastructure can improve the likelihood of achieving a Usable system because the interaction elements will have been more widely tested and become more standardised and familiar. Self-describing Components will dramatically improve the Usability of the system because they facilitate to be integrated.
	Self-describing Components	

A2.1.3.4 Accountability

The OSN should consist of elements and functions which can account for their characteristics and behaviours. In particular, data should be accompanied by meta-information, and methods (such as models) should account for their associated conceptual model (especially system definitions and assumptions). When discrete elements (data or functions) are integrated in any fashion, the characteristics of this integration (such as integrative assumptions) should also be self-explanatory.

Key System Requirement: Accountability		
← Fundamental challenges:	Quality	In order to give assurances of high quality, the system should be able to report access to and/or modification of data or services within the system.
	Access Control	The system should be able to report on access which was permitted by users and application systems to various data and services, and to ensure that only authorized access was permitted.
→ Architectural :	Self-describing Components	One can think of Self-describing Components as a form of Accountability in that the elements of a system are held accountable for themselves.

The following aspects of accountability are to be considered:

- Meta-information

  Data and services incorporated into the OSN should be fully described by meta-information.

- Model descriptions

  Models incorporated in the OSN should include complete descriptions, including

  - Boundaries of the system modelled
  - Model scope
  - Resolution
  - Assumptions and boundary conditions
  - Calibration and validation (including both the data sources and performance results)

- Quality Communication

  The users' trust in an OSN is based on the quality information actually provided by a data provider or a community for a given application domain.

  The OSN should communicate quality information to the tools and users needing them.

- Users/Applications

  Users and applications attempting to access the OSN should be accountable for their identities and their authority to exercise the access requested. In particular, they should be required by an OSN to provide suitable proof of identity for the purpose of authenticating access.

A2.1.4Architectural Principles

This subsection describes certain architectural principles for the OA which have been derived from the key system requirements.

During the OA specification process this list will be used to check if all crucial architectural properties have been taken into consideration and to assure that none of them has been forgotten. This check will be performed by a specific review process which will be undertaken after each final version of the RM-OA and final versions of all deliverables related to the specification and implementation of the RM-OA.

A2.1.4.1 Rigorous Definition and Use of Concepts and Standards

The *rigorous use of proven concepts and standards* is not only important for user acceptance. The use of open standards will decrease dependence on vendor-specific solutions and will also help ensure the openness of the OSN. Finally the consistent use of proven concepts will support the evolutionary development process of the OA.

Architectural Principle: Rigorous Definition and Use of Concepts and Standards		
← Key System Requirements:	Openness	Openness can best be achieved by the wise use of state-of-the art, yet widely accepted, Concepts and Standards.

A2.1.4.2 Loosely Coupled Components

It is essential that the components involved in an OSN are loosely coupled, where loose coupling implies the use of mediation to permit existing components to be interconnected without changes. This will permit the satisfaction of the primary goals:

- openness
- dynamic integration of different heterogeneous information systems, applications and networks with a minimum of effort
- scalability

Architectural Principle:	Loosely Coupled Components	
← Key System Requirements:	Openness	Loose Coupling of Components often facilitates Openness. To achieve a scalable system it is reasonable to build it with loosely-coupled components.
	Scalability	

A2.1.4.3 Technology Independence

As the OSN will be operated over a long period of time, the OA needs to be independent of technologies, their cycles and their changes. It must be possible to accommodate changes in technology (e.g. lifecycle of middleware technology) without changing the OA itself.

The influences of state-of-the art and emerging technologies and initiatives to the OA cannot be denied. But the overall architecture must be independent of specific implementation technologies (e.g. middleware, programming language, operating system). The OA design process shall not be influenced by or deal with technical limitations of specific implementation technologies.

Architectural Principle: Technology Independence		
← Key System Requirements:	Openness	To remain open over time, the system must maintain independence from particular technologies.

A2.1.4.4 Evolutionary Development - Design for Change

The OSN cannot be put into place all at once, and it cannot be developed, deployed and installed in the classical sense. It must be possible to develop and deploy the system in an evolutionary way.

The system must be able to cope with changes of

- user requirements
- system requirements
- organisational structures
- information flows
- data source types

The system must be designed to evolve.

Architectural Principle: Evolutionary Development		
← Key System Requirements:	Openness	Evolutionary Development will considerably facilitate development of an open system.
	Scalability	An "Evolutionary Development" approach will allow the system to be scaled up or down over the whole period of operation.

A2.1.4.5 Component Architecture Independence

Architectural independence describes the notion that existing information systems and information networks are independent of the OA in their architectural approach and vice versa.

This means that

- the OA does not impose any architectural patterns on existing information systems or information networks, for the purpose of them collaborating in an OSN,
- no existing information system or information network can impose architectural patterns on the OSN, and
- the OA and existing information systems and information networks are architecturally decoupled.

This will greatly improve the overall openness and acceptability of the OSN, since participating organisations are not obliged to change their internal workflows, systems, etc. in order to become part of the OSN.

Architectural Principle: Component Architecture Independence		
← Key System Requirements:	Openness	The architecture must remain independent of existing information systems and information networks in order to remain open and vice versa.

A2.1.4.6 Generic Infrastructure

The OA Services should not only be independent of organisational structures, information flows, etc. but also of the application domain.

Generic means that the OA Services should be designed in such a flexible and adaptable way that the OA Services can be used across different thematic domains and in different organisational contexts, and that the update of integrated components (e.g. applications, systems, ontologies) causes little or (ideally) no change visible to the users of the OA Services.

The richer the functionality of these OA Services is, the more the rest of the system (other services, applications, users) profits from building on a generic approach. A generic approach for the OA Services requires a generic approach in the description format of data sources and services.

Architectural Principle: Generic Infrastructure		
← Key System Requirements:	Openness	A Generic Infrastructure will greatly facilitate the Openness of the system. It can also improve the likelihood of achieving a Usable system because the interaction elements will have been more widely tested and become more standardised and familiar.
	Usability	

A2.1.4.7 Self-describing Components

The usage of self-describing components that provide context-sensitive formal and semantic descriptions of their interfaces can help to realise semantic interoperability. Components, such as data elements or models, should include descriptions of their critical characteristics and features, including sources, assumptions, etc. This information can be used to support tracing, monitoring, and logging facilities.

Architectural Principle: Self-describing Components		
← Key System Requirements:	Usability	Self-describing Components will dramatically improve the Usability of the system, most especially for the Service Developer. One can also think of self-describing Components as a form of Accountability in that the elements of a system are held accountable for themselves.
	Accountability	

FP6-511678

ORCHESTRA

Open Architecture and Spatial Data Infrastructure for Risk Management

Integrated Project

Priority 2.3.2.9 Improving Risk Management

Reference Model for the ORCHESTRA Architecture (RM-OA Version 2)

Annex A3

Conceptual Meta-Information Model

Date: 2007-01-31

Revision: 2.0

Start date of the ORCHESTRA project:	2004-09-01
Duration of the ORCHESTRA project:	3 years
Organisation name of lead contractor for this deliverable:	Austrian Research Centers GmbH - ARC

Document Control Page

Title	Reference Model for the ORCHESTRA Architecture (RM-OA) D3.2.3: RM-OA Version 2 Annex A3 (Rev. 2.0) Conceptual Meta-Information Model
Creator	Austrian Research Centers GmbH - ARC
Subject	Conceptual Meta-Information Model
Description	This document is the annex A3 of the Reference Model for the ORCHESTRA Architecture (RM-OA). This deliverable describes the conceptual meta-information model as particular requirements which will be used to clarify which type of meta-information will be present in the ORCHESTRA Architecture as well as expressing the ORCHESTRA understanding of the term meta-information.
Publisher	ORCHESTRA consortium
Contributor	Fraunhofer IITB, JRC, EIG, ATOS
Date	2007-01-31.
Type	Text
Format	application/msword
Identifier	ORCHESTRA Portal: SP3 / SP3 Quality Assurance / 09: D3.2.3 / 06: D3.2.3 RM-OA V2 (2.0) – published version
Source	Not applicable
Language	En-GB
Relation	none
Coverage	Not applicable
Rights	© 2007 ORCHESTRA Consortium The ORCHESTRA project is an Integrated Project (FP6-511678) funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).
Deliverable number	D3.3.1
Audience	☒ public ☐ restricted ☐ internal

,

Major Revision History

Revision	Date	Sections changed	Description
0.1	2005-04-18	all	First draft to develop a structure of the document and to assign tasks for contributions
0.6.	2005-05-01	all	Draft Version of D311 put on the portal
1.0	2005-06-07	all	Integration of updated references list and review of version 0.10 Updated second draft version put on the portal for review of SP 3 Leader
1.1	2005-06-16	all	Added section "Integration of Standards"; minor edits on wording
1.2	2006-01-18	all	New structure, as requested by the SP3 leader – for review by SP3 leader, EIG and IITB
1.21	2006-01-31	all	Restructuring of section 4.
1.23	2006-02-24	all	Section 6 rewritten
1.30	2006-02-28	all	Reviewed, Minor changes in Section 1, 2 and 3 Comments resolved, released for SP3 leader review
1.4	2006-04-06	all	Forwarding to SP3 leader for QA review
1-41	2006-07-03	all	Minor adoption according to review of SP3 leader in section 3.2.1 and 3.2.4 in correspondence to RM-OA V 1.10 section 8.5.1
1.41	2006-07-05	all	Forwarding to SP3 leader for final QA review
1.42	2007-01-16	all	Section 5.2.1 updated in order to use the same definition of keywords like "shall" as it is defined in the RM-OA
2.0	2007-01-31	all	Editorial changes for publication

Table of Contents

Figures

Table

A3.1 Management summary

A3.1.1 Purpose of this document

The purpose of this document is to:

1) Clarify the meaning of the "meta-information" and related concepts in the RM-OA.
2) Present a coherent set of requirements on meta-information imposed by ORCHESTRA high level requirements (see RM-O Annex A2) and the abstract service specifications.

These meta-information requirements are translated into a set of rules for building the meta-information models. This document provides rules prescribing how to specify ORCHESTRA Applications Schemas for meta-information (OAS-MI). Furthermore, it provides default OAS-MIs for selected purposes.

A3.1.2 Summary

One of the essential outcomes of the discussion about "What is data, meta-data or meta-information" during several discussion rounds is the conclusion that this question is not decidable at the data level itself. If we would try to do this again the confusion would remain as it has been since the beginning of the late 80's. The only way to express data or information as meta-information (see section A3.4.2) is in relationship to its purpose and context.

Thus the following can be stated: Meta-information is not necessarily needed for constructing a single data object. The need for meta-information arises from additional tasks (like catalogue organisation), when many different data objects need to have common attributes and descriptions (like the location of an object in a library).

Furthermore, meta-information can be defined only in the context of a special task/function. Only in this special context can a meta-information model of a set of objects of concern be defined. All data needed to fill up this meta-information model are per ORCHESTRA definition "meta-information". This culminates in the conclusion that for each purpose it will be necessary to define and specify a meta-information model.

Meta-information related requirements are listed in section A3.6. The requirements are hierarchically ordered by the referred source, in order to improve the requirements' traceability and ease the conflict resolution. Starting from the most important ones, the meta-information requirements are grouped into:

1. Requirements inherent to meta-information definition
2. Requirements inherited from the RM-OA
3. Meta-information related requirements on the ORCHESTRA architecture
4. Requirements originating from the architectural decisions in RM-OA
5. Requirements of the Meta-Information Model
6. Others

Subsequently, an initial list of the particular purposes is presented in section A3.7, followed by the discussion on meta-information for ORCHESTRA services.

In order to improve understandability, requirements and purposes are illustrated with examples, and notes explaining some of the consequences.

A3.2 Background and scope

The ORCHESTRA team determined that it is absolutely necessary to overcome the problems with the definitions of data, meta-data and meta-information. That's why the first section of the deliverable explains in a step by step fashion the term meta-information. This will help to form our understanding about meta-information, on which the following sections as well as the general ORCHESTRA understanding about meta-information can be based.

Having found that common ground we were able to start identifying classifications of meta-information according to its purpose. At the moment this is a list of the most common purposes needed within the ORCHESTRA Architectural Services (OA Services) like integration, discovery (search and navigation), interpretation, access/invocation of services as well as data. This list should not be seen as a closed (complete) list of purposes at the moment. We want to state that this list is a very good starting point for further work especially in relationship with requirements on services.

After having defined purposes, services must be found related to these purposes. But these services also need some kind of meta-information. It is necessary to point that out and identify the related requirements. That's why a section about required meta-information for services is introduced, specifying these services and requirements on a conceptual level.

A3.3 References

Australian Government, National Archive of Australia: Digital Record Keeping Guidelines; Glossary www.naa.gov.au/recordkeeping/er/guidelines/14-glossary.html

Barkmeyer E. J. et.al 2003 (Co-authros: A. B. Feeney, P. Denno, D. W. Flater, D. E. Libes, M. P. Steves, E. K. Wallace): “Concepts for Automating Systems Integration”; NIST; 2003

BPEL / BPE4WS: Business Process Execution Language (for Web Services) http://www.bpelsource.com/bpel_info/defined.html

BPML: Business Process Modelling Language http://www.bpmi.org/

CISCO Systems: Glossary www.cisco.com/en/US/products/sw/netmgtsw/ps829/products_user_guide_section09186a008007e93e.html

Colorado Digitization Programme: Digitization Glossary, www.cdpheritage.org/resource/introduction/rsrc_glossary.html

Denzer R. 1994, Anforderungen an Metainformationssysteme für den Umweltbereich, Article in R.Güttler, W. Geiger, 2. Workshop Integratio von Umweltdaten, Metropolis, 1994, pp. 77-88

Denzer R. and Güttler R. 1996, Requirements of Meta-Information Models for the Environmental Domain, Journal of Computing and Information, Vol. 2, No.1, pp. 1288-1295, 1996

Denzer R. and Güttler R. 1997, Meta-Information Concepts for Environmental Information Systems, GeoComputing, 1997

DOI: Digital Object Identifier System; The International DOI Foundation (IDF); Glossary of Terms www.doi.org/handbook_2000/glossary.html

Douglous D. Nebert 2004, The GSDI Cookbook, Version 2, 25. January, 2004, Global Spatial Data Infrastructure Association

EMELD: Electronic Metastructure for endangered Languages data and documentation; Glossary www.emeld.org/school/glossary.html

europe4DRM: Digital Rights Management; Glossary www.europe4drm.com/l_menue/glossary/glossary.htm

GRAINGER Engineering Library Information Center; Digital Library Research Projects, Glossary dli.grainger.uiuc.edu/glossary.htm

INTERA: Integrated European language data Repository Area www.mpi.nl/INTERA/glossary/glossary.html

IONET South African Internet Provider; Glossary
www.ionet.co.za/glossary.asp

JP1, Federal Standard Telecommunications: Terms and definitions extracted from Joint Pub 1-02 (DOD Joint Staff Publication No. 1-02), 1994, *Department of Defense Dictionary of Military and Associated Terms,* http://www.its.bldrdoc.gov/fs-1037/dir-010/_1401.htm

KWIC, Kids' Well-being Indicator Clearing House; Data Terms
www.nyskwic.org/u_data/data_terms.cfm

LDP: The Linux Documentation Project; Author Guide
ldp.rtin.bz/LDP/LDP-Author-Guide/html/glossary.html

NASA Glossary: Jet Propulsion Laboratory, Californian Institute of Technology
podaac.jpl.nasa.gov/glossary/

NASA Earth Observatory, Glossary
eobglossary.gsfc.nasa.gov/Library/glossary.php3

NoiseBetweenStations: Metadata Glossary
www.noisebetweenstations.com/personal/essays/metadata_glossary/metadata_glossary.html

OASIS: Organization for the Advancement of Structured Information Standards
http://www.oasis-open.org/home/index.php

O'NEIL Information Solutions: Glossary
www.oneil.com/cfm/glossary.cfm

OGC-Catalog-Services: „The OpenGIS™ Abstract Specification – Topic 13: Catalogue Services (Version 4)", 99-113.doc, 1999

OpenGIS Reference Manual (2003); Open Geospatial Consortium Inc.; date: 2003-09-16; Reference Number: OGC 3-040; Version 0.1.3; page 2

ORACLE FAQs: Glossary;
www.orafaq.com/glossary/faqglosm.htm

OWL-S: Web Ontology Language for Services
http://www.w3.org/Submission/2004/07/

OzEmail: Austalians Internet Service Provider; Glossary
members.ozemail.com.au/~ieinfo/ten-glossary.htm

PollHodg 2004: Jeffrey T. Pollock, Ralph Hodgson: Adaptive Information. ISBN 0-471-48854-2. Wiley 2004

SCHULTE: S. Schulte im Walde, I. Cramer and S. Schacht : Classification and Clustering for Computational Linguistics Introduction; Computational Linguistics Saarland University 2004

SAVI Interactive: Dynamic Community Information System, Central Indiana, www.savi.org/savii/documentation/glossary.aspx

SASL: Simple Authentication and Security Layer http://asg.web.cmu.edu/sasl/index.html

SEDRIS: The Source for Environmental Representation and Interchange; www.sedris.org/glossary.htm

Synergyanywhere: www.synergyanywhere.com/evaluate/document-glossary.htm

TerraLink International: www.terralink.co.nz/profile/glossary/

WebSite2Go: Glossary of Web Terms; FAQ's www.website2go.com/p78.html

W3C Introduction to RDF Metadata (1997); Author: Ora Lassila, ora.lassila@research.nokia.com, Nokia Research Center; http://www.w3.org/TR/NOTE-rdf-simple-intro-971113.html

WGUC Public Media: www.wgcu.org/watch/hdtv_glossaryofterms.html

WIKIPEDIA: Metadata (computing) en.wikipedia.org/wiki/Metadata_(computing)

WIKIPEDIA: Metadata (corporation) en.wikipedia.org/wiki/Metadata_(corporation)

Wolf-Fritz Riekert, Bericht der Arbeitsgruppe Metadaten, in R. Güttler and W. Geiger (editors), 3. Workshop, Schloß Dagstuhl, 1995

WordNet 2.0 Search Engine www.cogsci.princeton.edu/cgi-bin/webwn

WSCI: Web Service Choreography Interface http://www.w3.org/TR/wsci/

WSMO: Web Services Modelling Ontology http://www.wsmo.org/

XACML: OASIS eXtensible Access Control Markup Language;
Core XML schema for representing authorization and entitlement policies
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20

A3.4 Meta-information and related concepts

Our experience within ORCHESTRA and discussion undertaken within the architectural design phase in particular has shown that every topic involving "Meta-data", "Meta-information" and related concepts inevitably leads to confusing discussions about the definitions, the limits and the approach to modelling meta-information systems. In order to avoid further confusion, we decided to write down the

ORCHESTRA understanding of meta-information and related terms which will be used from now on in the ORCHESTRA project.

This section demonstrates that there is no widely accepted definition of the term “meta-information” yet, discusses the importance of the meta-information for the ORCHESTRA architecture, presents the ORCHESTRA understanding of meta-information and several meta-information related concepts, and finally discusses the meaning & consequences of meta-information within ORCHESTRA architecture.

A3.4.1 General understanding of the term meta-information

Taking the translation from the Greek language the term ***“Meta”*** expresses ***“about, on”***. This is very generic and abstract. For the purpose of writing the technical specifications a far more precise definition is needed.

A3.4.1.1 Existing definitions

A google (www.google.com) search using the query ”definition: metadata” returns the following definitions:

- Metadata provides information about the content, quality, condition, and other characteristics of data. (KWIC, Data Terms).
- Data about data, or information known about the image in order to provide access to the image. Usually includes information about the intellectual content of the image, digital representation data, and security or rights management information (Digitization Glossary, Colorado Digitization Program).
- (1) Information about a data set which is provided by the data supplier or the generating algorithm and which provides a description of the content, format, and utility of the data set. Metadata provide criteria which may be used to select data for a particular scientific investigation. (2) Information describing a data set, including data user guide, descriptions of the data set in directories, and inventories, and any additional information required to define the relationships among these. (ESADS, EPO, IWGDMGC, NASA Glossary).
- Informational data about the data, included in a signal's data stream (WGUC Public Media).
- Metadata is data about data. Used in the context of digital spatial data, metadata is the background information which describes the content, quality, condition, and other appropriate characteristics of the data (TerraLink).
- Meta-is a prefix that in most information technology usages means 'an underlying definition or description.' Thus, document metadata - as it relates to document-management - is a definition or description of the document it relates to. When using document-management-software this information is typically entered by an end user or a scanning operator. The Metadata Information can include physical location information (e.g., where the document is stored) and document identification information (e.g., date archived, creator, and contents) (Synergyanywhere).
- Information describing the content or utility of a data set. For example, the dates on which data were procured are metadata (NASA Earth Observatory, Glossary).
- A definition or description of data. In data processing, metadata is definitional data that provides information about, or documentation of, other data managed within an application or environment (NoiseBetweenStations / Metadata Glossary).
- Metadata is the term used to describe data about data. It describes who collects the data, what the data contains, where (and how) the data is stored, when (and how often) the data is collected, and why (SAVI Interactive).
- Data that is used to describe other data. Data definitions are sometimes referred to as metadata. Examples of metadata include schema, table, index, view and column definitions (ORACLE FAQs).

- Information describing the characteristics of data. Data or information about data. Descriptive information about an organisation's data, data activities, systems, and holdings (SEDRIS).
- Data about data. For example, the title, subject, author, and size of a file constitute metadata about the file (O'NEIL Information Solutions).
- An HTTP tag which defines certain top-level information about the web page or web site. Usually contains keywords for search engines, a description of what the site contains in terms of subject matter and audience, can contain information about the author and tools used to create the page or site. Is one of the highest priority elements of a website when used in conjunction with search engines. Search engines typically weight the text found in the metadata tags higher than the text found in the actual contents of the pages (WebSite2Go).
- Metadata is simply data used to describe other data. It can be used to describe information such as file type, format, author, user rights etc. and is usually attached to files but invisible to the user (europe4DRM).
- Data about data (INTERA).
- Data about data. Includes information describing aspects of actual data items, such as name, format, content, and the control of or over data (GRAINGER).
- Metadata is terminology or jargon, used as a business language to communicate specific meaning. For example, accountants use a special jargon for accounting, while doctors and hospital staff use a different jargon to communicate medical meaning. This meaning must be understood for effective communication. Metadata tags are used by XML to surround data content and so identify data meaning (OzEmail).
- Structured information that describes and/or allows users to find, manage, control, understand or preserve other information over time (Australian Government, National Archive of Australia).
- Data about data. Metadata describes how and when and by whom a particular set of data was collected, and how the data is formatted. Metadata is essential for understanding information stored in data warehouses and has become increasingly important in XML-based Web applications (IONET).
- Could elevate the status of the web from machine-readable to something we might call machine-understandable. Metadata is "data about data" or specifically in our current context "data describing web resources." The distinction between "data" and "metadata" is not an absolute one; it is a distinction created primarily by a particular application ("one application's metadata is another application's data") (W3C, "Introduction to RDF Metadata" 1997).
- Data about data. Meta-data includes pertinent information about a collection of data, including information about the speaker, the collector and the format of the data. It is essential to accurate analysis of the data collected and increases portability (EMELD).
- Data about data; "a library catalogue is metadata because it describes publications" (WordNet).
- Metadata is data about data. An example is a library catalogue card, which contains data about the nature and location of a book: It is data about the data in the book referred to by the card (WIKIPEDIA).

A literature search yields following results:

- Denzer and Güttler published four types of meta-information (Denzer 1994, Denzer and Güttler 1996, 1997):
    - Semantic meta-information
      This type of meta-information describes the content of information. Users can identify relevant information on the basis of this semantic meta-information.

- Syntactic meta-information
  This type of meta-information refers to information needed to realise the technical access to a catalogue and (in the future) to data (e.g. data types or access methods).

- Structural meta-information
  This type of meta-information describes the structure of some information object. The structure might be, for example, a tree, list, set, table, etc. In other words it describes aggregates.
  Structural Meta-information has semantic and syntactic parts.

- Navigational meta-information
  The focus of Navigational meta-information is to describe other meta-information, so that it becomes structured and findable. This information is used to facilitate the navigation-process a user conducts in a user-interface.

- Wolf-Fritz Riekert and Ralf Kramer report in a Workshop about meta-data [Wolf-Fritz Riekert 1995] the following: “Meta-data are those data, that – depending on the respective question – allow one to navigate in the data and allow the correct interpretation of the data for this specific question.”

- The Open Geospatial Consortium (OGC) uses within their published reference model (OpenGis Reference Model, 2003) the definition that “Metadata is data about data”.

- Douglas D. Nebert, chair of the GSDI Association technical working group, describes in its GSDI cookbook (version 2, January 2004, p. 25) the categorisation of meta-data according to the purposes discovery, exploitation and exploration.

A3.4.1.2 Analysis of the search results

As we can read from above most of the definitions assert that “Metadata is data about data”. Typically two levels of information are mentioned: data and meta-data on the one hand but on the other hand the terms information and meta-information are often used in the same sense.

Some definitions found in the literature above go beyond these general statements, and clearly state that ***meta-information supports people and machines to make information more usable by describing it according to standards and by precise structuring and annotation (including the semantics) of information objects***.

All of the definitions above have in common that they are imprecise in defining boundaries between information and meta-information as well as between data and meta-data. For precisely this reason confusion often occurs in the common understanding of the term meta-data or meta-information.

A better and more precise approach was already addressed in the papers of Denzer and Güttler (Denzer 1994, Denzer and Güttler 1996, 1997) putting meta-information in the context of a special purpose (like navigation). So the most concrete definitions already have special use cases or purposes in their focus.

A3.4.2 ORCHESTRA understanding of meta-information

As shown in the last section, no commonly accepted understanding of meta-information and related terms has been established so far. In order to avoid the problems with different interpretations and existing weak boundaries, the term meta-information and several related terms will be defined, and this definition will consequently be used in all ORCHESTRA documents.

A3.4.2.1 Basic terminology related to meta-information

In order to clearly define the term meta-information, several simpler terms need to be defined first. Basic meta-information related terminology is listed in Table 1.

Terms	Meanings
data	Representation of facts, concepts, or instructions in a formalised manner suitable for communication, interpretation, or processing by humans or by automatic means (JP1 1994)
information	The meaning assigned to data by means of the known conventions used in their representation. (JP1 1994)
resources	Either functions (possibly provided through services) or data objects
universe of discourse:	View of the real world that includes everything of interest (see ISO 19101 and also section 8.2 of RM-OA V1.10)
objects of concern	All resources in the focus of a particular purpose and hence in the focus of the corresponding meta-information model. Objects of concern are a subset of all objects in the universe of discourse.
(particular) purpose	A particular use case or the goal of the usage of the objects of concern.

Table 1 – Description of terms used in the ORCHESTRA meta-information definition

A3.4.2.2 Meta-information conceptual model

In order to describe the common characteristics of the objects of concern it is necessary to develop a conceptual model which has to cover them all and has to be suitable as well as sufficient in order to form the base for the algorithms for the particular underlying purpose.

Using the basic terminology defined in A3.4.2.1, we can define the meta-information conceptual model as:

A meta-information conceptual model is a description of the meta-information needed to describe the objects of concern for a particular purpose.

Explanation: A conceptual model is a representation of the human understanding of meta-information needed to describe data constructs for a specific task and purpose in a specific given context, including the relationships that may exist among them. Conceptual models are meant for people to read and understand and not necessarily related to the way meta-information will be represented in an application. The “conceptual model” is made up of data-objects and the relations between them in order to realise a particular purpose (task, goal, …).

Note: Data types/building blocks are defined in the conceptual meta-information model.

A3.4.2.3 Meta-information model

Starting from the conceptual meta-information model, a concrete model can be defined. Such a concrete model is referred to as “meta-information model”.

Using the terminology from A3.4.2.1, the meta-information model can be defined as:

Note 1: OAS-MI Application schemas define the data types addressed in the conceptual meta-information model

Note 2: There is no such thing as "the meta-information model" valid for all purposes.

A3.4.2.4 Meta-information

All information needed to fill up the concrete meta-information model is "meta-information" for this particular purpose.

The ORCHESTRA understanding of the term meta-information is:

Meta-Information is descriptive information about resources in the universe of discourse. Its structure is given by a meta-information model depending on a particular purpose.

Note: The definition above stresses that meta-information is not needed to build a single data object. The need for meta-information arises from additional tasks or a particular purpose (like catalogue organisation), where many different services and data objects (objects of concern) must be handled by common methods and therefore have to have/get common attributes and descriptions (like a location or the classification of a book in a library).

A3.4.2.5 Metadata & co.

The terms data, metadata, meta-data, metainformation, information, meta-information, and meta-information are used in different places in the literature, and on the web.

While most authors clearly distinguish between "data" and "information", the terms meta-data and meta-information are often used interchangeably. This fact should not surprise us, as the term meta-data already implies that this particular data should be interpreted as meta-information.

In ORCHESTRA, the meaning of data is only given by the underlying information model, and certain pieces of data may have very different meanings depending on the information model. When referring to certain data in the context of a meta-information model, we are actually referring to the meaning given to this data within a model.

In order to avoid confusion, and to account for the fact that all data may have different meanings, the term meta-information shall be used in all the ORCHESTRA documents whenever a datum is seen in the context of a meta-information model. The related terms, including "metadata", "meta-data", and "metainformation" may not be used in ORCHESTRA documents.

A3.5 Formal meta-information model specifications

As is evident in section 5 "the meta information model" does not exist. In addition, a meta-information model can only be developed for a particular purpose and within constrains given by the chosen architecture and available services.

Consequently, this section develops a set of formal rules for building ORCHESTRA meta-information models rather than attempting to develop the meta-information model valid for all purposes. These rules shall be tested and improved by developing meta-information models for particular purposes.

Rules for building ORCHESTRA meta-information models for purpose-specific application schemas (so called OAS-MIs) are part of the RM-OA Annex B1.

With this in mind, the rest of this document concentrates on following tasks

1. Establishing a list of requirements and constraints on the ORCHESTRA architecture and ORCHESTRA meta-information model (section A3.6).
2. Establishing a list of purposes (use cases/functions) in the context of users and/or machines (section A3.7)
3. Elaborating the consequences of the requirements and purposes on ORCHESTRA services (section A3.8).

A3.6 Requirements relevant to ORCHESTRA meta-information models

A3.6.1 Introduction

A3.6.1.1 Purpose of this section

This section aims at listing the requirements related to meta-information and meta-information models, including the requirements on infrastructure and services, in order to assure the consistency of requirements related to meta-information.

Requirements in this section are either derived from the requirements listed in the RM-OA Annex A2 "Requirements for the OSN and the OA" ("RM-OA requirements" from now on), derived from the services specifications, or inherent in the definition of meta-information and related terms given in A3.4.2.

In order to improve traceability, explicit references to related requirements in RM-OA and service specifications are given for all the requirements on the ORCHESTRA meta-information model.

A3.6.1.2 Keywords used to indicate requirement levels

This document follows the ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards (Fifth edition 2004) w.r.t. the usage of the word "shall", "shall not", "should", "should not", "may" and "need not". The word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed.

A3.6.1.3 Other keywords used in this section

"MECHANISM" refers to a combination of data and services needed to achieve the requirement. This keyword is technology-independent and does not imply any preference for solving the problem.

A3.6.1.4 General considerations

Information and hence meta-information is available in many different formats with varying syntax and structure, such as RDF, formatted documents, or plain text. This implies that varying technologies have to be integratable and adoptable (cf. RM-OA section 6.3.3 Technology Independence). Moreover meta-information can originate from a multiplicity of information sources, such as databases, files or services (cf. RM-OA section 6.3.5 Component Architecture Independence). One part of a solution to overcome interoperability problems is the rigorous use of standards (cf. RM-OA section 6.3.1 Rigorous Definition and Use of Concepts and Standards).

Currently there are a lot of so called meta-data standards available, such as the ISO 19115 meta-data standard for geographic information and services, FGDC's (Federal Geographic Data Committee) "Content Standard for Digital Geospatial Metadata" (CSDGM) or the Dublin Core Metadata Initiative's (DCMI) Metadata Terms for documents.

Most of these standards are developed for a certain field of application and define a "fixed" set of meta-data elements. Even though the selection of one specific standard may be appropriate for a new information system this approach is not feasible in a heterogeneous environment, where many disparate systems are to be integrated. Not only is it most unlikely to find one standard that satisfies all the requirements of the participating systems, services and users, but it is also simply not possible to oblige them to drop their (from their point of view) good and well understood (quasi) standards or system and follow a new one. This is strongly connected with RM-OA's key system requirements "Openness" and "Usability" (cf. RM-OA Annex A2).

Moreover the attempt to develop one “universal all-purpose metadata standard” and the mapping of the respective local meta-data standard to this global one cannot be a reasonable solution for similar reasons.

As a conclusion the ORCHESTRA meta-information model shall not rely on any a priori known structure of meta-information. The ORCHESTRA meta-information model itself does not define any concrete meta-information set. Instead, it provides the foundations and means for the definition and the integration of any type of meta-information and hence every existing meta-information standard.

This course of action allows the participating systems to retain their established standards and gives communities the necessary liberty to define and agree on standards that satisfy their common requirements.

ORCHESTRA meta-information models shall support this process and shall not restrict it in any way. Moreover, the adoption of new meta-information standards shall only be a matter of configuration that must not affect the meta-information specifications and implementations.

Hence, ORCHESTRA meta-information models shall support the iterative design process of the ORCHESTRA architecture and meet the demand for a system designed to evolve.

Finally, ORCHESTRA meta-information models have to supply rules and guidelines for the development of meta-information schemas similar to the GFM approach. As a consequence of this, for each identified task within ORCHESTRA a dedicated meta-information model configuration has to be defined.

A3.6.2 Requirements inherent to (definition of) “meta-information”

All the requirements listed in this section are inherent to the definitions given in section A3.4.2, and independent of the architectural decisions taken in the RM-OA. In particular, requirements listed in this section do NOT assume any of the following:

- Distributed architecture
- Loosely coupled components
- Multitude of users and roles

A3.6.2.1 Any data MAY be interpreted as meta-information for a particular purpose

According to definitions in section A3.4.2.4, *meta-information exists only in the context of meta-information model for a particular purpose*. Any piece of data may be interpreted as information or as meta-information, depending on the objects of concern, and on the meta-information model for a particular purpose.

Note 1: The consequences of this rule for a general-purpose system (e.g. ORCHESTRA) are immense: a truly general purpose system SHALL provide a mechanism for interpreting any data as either information or meta-information depending on the purpose.

Note 2: The same data may even be interpreted in different ways depending on the particular purpose.

A3.6.2.2 Meta-Information SHOULD NOT be considered static

It is often assumed that meta-information is written down by humans or otherwise generated in advance and kept in the data store. This assumption is misleading, as some of the meta-information MAY be generated on the fly when needed and disposed of afterwards. One consequence of this rule is that meta-information MAY, and usually does change over time.

Note: Meta-information may in fact change even if none of the data are changed, e.g. when the algorithm for generating the meta-information changes!

Example: Air quality in an area is a note (“excellent”, “good”, etc.) that depends on the concentration of a certain pollutant, and on the “quality standards”. Changing the legal definition can easily change the air quality, even though the actual pollutant concentration does not change. For example, the EU council Directive 1999/30/EC of 22 April 1999 which set rather restrictive limits on particulate matter (e.g. PM10) has recently led to dramatic (perceived) decline of the air quality in the city of Vienna (Austria).

A3.6.2.3 Meta-Information CAN be classified according to a particular purpose

As already indicated in section A3.3 data become meta-information in the context of a meta-information model, and every model is defined with some “particular purpose” in mind. Since meta-information cannot be separated from the “particular purpose”, and “meta-information model for a particular purpose”, it makes sense to classify it according to its purposes. Obviously the classes/dimensions of meta-information are not orthogonal as a certain meta information item might be used for several purposes. The classification is particularly useful for special purpose systems with a limited number of “particular purposes”, where special data types can be assigned to all the “special purposes” of relevance for the system, because these data types inherently carry semantic information with them.

Classifying the meta-information according to purpose in general-purpose systems can never be complete, but it may still make sense to do it for some “well known” purposes.

Note: A “well known” purpose is by definition one that is known to relevant people (for instance us), here and now. As “relevant people”, “here” and “now” are subject to a change, so are the “well known” purposes. In fact, even the models of “well known” purposes may change with time. This has a profound influence on ORCHESTRA, as it implies that there SHALL be a simple way to implement new meta-information models and alter the existing ones!

A3.6.3 Requirements inherited from the RM-OA

“Requirements for the OSN and the OA” are defined in terms of “User Roles”, “Fundamental challenges”, “Key System Requirements” and “Architectural Principles”. The relationship between these terms is given in Table 2 – ORCHESTRA System Requirements of the RM-OA Annex A2, and reproduced in the following table for quick reference. Please refer to the original document for the full definition of the terms.

User Roles			Fundamental challenges	Key System Requirements			
Service Developer/System Administrator	Service provider	End user		Openness	Scalability	Usability	Accountability
✓			Scale and Scope	✓	✓		
		✓	Integration/Collaboration	✓			
		✓	Long Lifetime	✓	✓	✓	
		✓	Quality		✓	✓	✓
✓		✓	Transparency (Hidden Process Complexity)			✓	
			Access Control		✓	✓	✓

Architectural Principles	Openness	Scalability	Usability	Accountability
Rigorous Definition and Use of Concepts and Standards	✓			
Loosely Coupled Components	✓	✓		
Technology Independence	✓			
Evolutionary Development – Design for Change	✓	✓		
Component Architecture Independence	✓			
Generic Infrastructure	✓		✓	
Self-describing Components			✓	✓

Table 2 – ORCHESTRA System Requirements

A3.6.4 Meta-information related requirements on the ORCHESTRA architecture (OA)

The requirements listed in this section are direct consequences of the RM-OA requirements and the requirements inherent to the definition of the meta-information (A3.6.2). They should be seen as the interpretation and extension of the basic RM-OA.

Note: The requirements listed in this section are kept as general as possible, in order to allow easier integration in the future versions of the RM-OA. In particular, "data" is used instead of "meta-information" wherever possible.

A3.6.4.1 OA SHALL provide a mechanism for interpreting arbitrary data as meta-information (for a particular purpose).

Derived from: This requirement is derived from the combination of requirement A3.6.2.1 "Any data MAY be interpreted as meta-information for a particular purpose", RM-OA "Evolutionary Development – Design for Change", and RM-OA "Generic Infrastructure" principles

Explanation: The ORCHESTRA architecture shall be generic and designed for change. Consequently, the list of particular purposes is completely open, and there is no a priori way of saying which data may be interpreted as meta-information in the future. Therefore a mechanism for explicitly stating that certain data contain meta-information for a particular purpose is needed.

Note: As a side-effect of this requirement, all mechanisms defined for data are automatically valid for meta-information as well. In particular, mechanisms supporting data normalisation (requirement A3.6.4.1), combining the data from different sources (requirement A3.6.4.4), and discovering and collecting the data (requirement A3.6.4.6) are all valid for meta-information.

A3.6.4.2 OA SHALL provide a mechanism for easily introducing and altering meta-information models

Derived from: This requirement is derived from the combination of requirement A3.6.2.3 "Meta-Information CAN be classified according to a particular purpose", RM-OA "Evolutionary Development – Design for Change", and RM-OA "Generic Infrastructure" principles

Explanation: The ORCHESTRA architecture shall be generic and designed for change. Consequently, new meta-information models may be introduced and old ones altered at any time. Therefore a mechanism for easily introducing and altering meta-information models is needed.

A3.6.4.3 OA SHALL support data normalisation

Derived from: This requirement is related to "Scalability" and "Usability" requirements, and addresses some aspects of the "Long lifetime", "Quality" and "Transparency" challenges.

Explanation: One aspect of the ORCHESTRA architecture that is of great importance for meta-information, but which has not been discussed in RM-OA requirements is the "data normalisation" principle. This principle originates from database theory, and basically states that one SHOULD avoid duplicating data. Failing to do so inevitably lead to huge overhead in maintaining the consistency of the data, thus violating the "Scalability" and "Usability" requirements.

As a consequence, ORCHESTRA SHOULD provide a way to reference existing data ("foreign keys"), a way to calculate the derived data on the fly ("functions") and a way to transparently access the referenced and derived data across an OSN!

Note 1: Accessing referenced data over a network, as well as accessing data that are derived on the fly is obviously much slower than accessing data that are locally available on a single server. In order to improve performance, ORCHESTRA services MAY introduce the transparent caching and pre-fetching of data.

Note 2: In some cases, it will be desirable to mirror the data on several servers, for performance reasons, or as a means to improve the service availability. It is desirable to design ORCHESTRA services with this in mind, and for instance define interfaces for triggered data updates and data synchronisation.

A3.6.4.4 OA SHALL provide a mechanism for assuring the referential integrity and for handling the lack of referential integrity.

Derived from: This requirement is a direct consequence of the requirement A3.6.4.3 "OA SHALL support data normalisation**"**

Explanation: Because of data normalisation, OSN effectively behaves as a distributed database. Without a mechanism for assuring the referential integrity, moving the data to a new location may leave the OSN in an inconsistent state, with potentially disastrous consequences. The same is true for deleting the data, and may also be true for altering the data (depending on the nature of the meta-information and the nature of the change).

Note 1: The problem of referential integrity of altered data is a non-trivial one, because some of the references may remain valid although the data have changed. (e.g. a solid may melt on heating without changing the chemical composition; a thermometer always measures temperature although the measured value may change, and even converting from °C to K will not change this fact.).

Note 2: Depending on the data ownership, the references may be unidirectional or bidirectional. Both types of references are needed in OSN, but the referential integrity can only be assured for bi-directional references. The integrity of the unidirectional references could be improved with some kind of a public notification service (e.g. on a catalogue, similar to Google's "news alerts" notification service).

Note 3: In some cases, assuring the full referential integrity within OSN will be impossible because of the (lack of) rights for altering some of the affected data. In order to assure the highest possible level of referential integrity, a mechanism for notifying the data owners about necessary changes to their data SHOULD be provided.

A3.6.4.5 OA SHOULD provide a mechanism for combining the meta-Information from different sources

Derived from: This requirement is a direct consequence of the data normalisation principle (see A3.6.4.1)

Explanation: The distributed ORCHESTRA architecture will result in the distribution of data over different services in the OSN. As "anything can be meta-information on anything" this will result in distribution of the meta-information over different services.

In order to assure that all ORCHESTRA services and applications can use the distributed meta-information, a standard mechanism for assembling the meta-information from distributed sources should exist.

A3.6.4.6 OA SHALL provide a mechanism for discovering and collecting the distributed data.

Derived From: This requirement is strongly related to A3.6.4.5 "OA SHOULD provide a mechanism for combining the meta-Information from different sources", RM-OA's fundamental challenges "Scale and Scope" and "Integration/Collaboration", and the architectural consequence of "Loosely Coupled Components" (RM-OA, section 11.4.2).

Explanation: ORCHESTRA actors (users and services) have no a priori knowledge of the data (and thus meta-information) available in the OSN. Consequently, a mechanism for discovering the scattered data (and thus meta-information) is essential to the function of the OSN.

Note 1: This causes the need for services, which are able to collect meta-information at various locations in a network (e.g. crawlers, indexers). Such services would facilitate the implementation of services like the Catalogue Service.

Note 2: One important category of distributed meta-information that needs to be collected by some kind of a catalogue service and subsequently made available to all OSN actors is the meta-information describing the capabilities of the services available within OSN. A mechanism assuring that this special catalogue is automatically synchronised on introduction, change, and removal of services is essential for correct functioning of the OSN!

A3.6.4.7 OA SHALL provide a mechanism for introducing new information models in OSN

Derived from: This requirement is derived from requirement A3.6.2.2 **"**Meta-Information SHOULD NOT be considered static**"** and **"**Evolutionary Development – Design for Change" architectural principle.

Explanation: ORCHESTRA networks SHALL be able to adapt to changes in stakeholder requirements. In addition to adapting to new technologies, this means above all a possibility to accommodate new data models.

Note: Requirement A3.6.5.4 "Special meta-information related data types SHOULD be defined where appropriate" cannot be adequately satisfied without this mechanism.

Example: An undetermined and possibly unlimited set of particular purposes and associated meta-information models exists for any non-trivial set of data. At any given moment, only a small subset of the possible meta-information will be known to exist, and an even smaller subset will actually be provided, depending on the current interests of the actors. As both interests and meta-information models are subject to change, a mechanism for altering and adding meta-information models needs to be provided.

A3.6.4.8 OA shall provide a standard mechanism for using the arbitrary security mechanisms for authentication and authorization

Derived from: This requirement is a direct consequence of the "Evolutionary Development – Design for Change" and "Component Architecture Independence" principles.

Explanation: Security mechanisms are a short-lived and critical technology that has to be replaced as soon as it becomes obsolete. On the other hand, the ORCHESTRA architecture is built to last ("Designed for change"), and cannot be easily replaced.

In addition, no technology is 100% secure, and some actors may be satisfied with simpler and less secure mechanisms, while others may require more sophisticated mechanisms which might be more costly in terms of acquisition or use.

Last, but not the least important, the only way to assure the long-term security maintenance of the ORCHESTRA infrastructure beyond the end of the project is by incorporating some existing security framework that is widely used, and likely to be well maintained in the future (i.e. SASL).

A3.6.4.9 OA SHOULD provide a mechanism for aiding the consensus building process within OSN

Derived from: This requirement addresses the RM-OA fundamental challenges of assuring Integration/Collaboration within the OSN of arbitrary size ("Scale and Scope"), over long periods ("Long Lifetime") and transparently for the users ("Transparency"). It is closely related to requirement "A3.6.5.10 ORCHESTRA meta-information model SHALL be able to integrate arbitrary standards".

Explanation: The ORCHESTRA architecture imposes interoperability between all services within OSN on a syntactic level, but does not impose any limits on service functionality or data formats. Within OSN, users will define some kind of "standards" for data and services to assure full interoperability within a group. The ORCHESTRA architecture SHOULD provide mechanisms to aid the consensus building process and introduction of such "standards".

Note: In combination with the requirement A3.6.4.7, this requirement calls for a mechanism (service) aiding the consensus building process for introducing new data types. Such a mechanism would greatly improve the value of the ORCHESTRA architecture in the big networks (Water Framework Directive, GMES).

A3.6.4.10 OA SHALL provide a mechanism for assuring data and services Interoperability within OSN

Derived from: This requirement is closely related with the requirement “A3.6.4.9 OA SHOULD provide a mechanism for aiding the consensus building process within OSN”, and the RM-OA “Self describing components” RM-OA architectural principle.

Explanation: In a typical OSN, there will be several (possibly many) groups of users with their own “standards” for data and services. Even in a hypothetic “homogenous OSN”, data formats and services evolve over time, leading to the same type of compatibility issues as encountered in heterogeneous OSNs.

In both cases, ORCHESTRA SHALL assure that data and services originating in one user group can be easily combined with those of another user group, preferably in a completely transparent way for all the users and services.

Note 1: This requirement has three big consequences for ORCHESTRA. First, all the data and services SHALL be self-describing on both a syntactic and semantic level. Second, ORCHESTRA services SHOULD use this information to flexibly use completely new types of data and services as needed, rather than relying on a certain pre-defined set of data types and services. Finally, a fully developed OSN requires a set of services with advanced search, retrieval and transformation capabilities.

Note 2: As a consequence of requirement A3.6.4.7, services within OSN may use different authentication and authorization mechanisms. If this is so, a mechanism for assuring the interoperability of the authentication and authorization mechanisms SHALL be provided as well (keywords are: trust management, single sign on, identity federation).

Example: One user group may be using the SI units, while another uses the imperial units. OSN SHOULD provide a mechanism for using both data sources simultaneously (e.g. by a forecasting service), in a transparent way.

A3.6.4.11 A mechanism for adding explicit meta-information to legacy data SHALL be provided

Derived from: This requirement is derived from the key system requirements “Usability” and “Accountability” (RM-OA Annex A2, section A2.1.3.3 and A2.1.3.4) and derived architectural consequences “Generic Infrastructure” and “Self-describing Components” (RM-OA Annex A2, section A2.1.4.6 and A2.1.4.7).

Explanation: Features and/or services provided on any system in the thematic domain, which have not been designed according to the ORCHESTRA architecture – e.g. because they existed before ORCHESTRA - have a lack of available meta-information. However, such meta-information is needed by the services acting at the usage level. Therefore, meta-information has to be made available by providers in the thematic domain.

Note: Generation of meta-information in the thematic domain can be an extensive task for providers. Consequently, tools and services supporting automatic and/or semi-automatic generation of this meta-information are necessary to assure the acceptance of the OA by data and service providers.

Knowledge about features and/or processes is usually available in a form unsuitable for automatic processing in arbitrary formats (e.g. implicitly represented in documents, web pages, business processes databases etc.). In order to have access to the meta-information, it must be separated from its environment and transformed into an explicit representation, such that it can be collected and stored by ORCHESTRA services in order to apply certain processing on it (e.g. reasoning or other calculations).

On the generation level, meta-information is made accessible, explicit and transferable in an exchange format.

Data/text mining services, KDD services (Knowledge Discovery in Databases), (semantic) annotation services, converters, encoders might be examples for services supporting this requirement.

Example: In applications based on Semantic Web technology, web pages and web services are annotated by means of annotation tools. These tools are based on ontologies, i.e. they can annotate web pages as instances of concepts defined in the ontology. The language used for ontologies in the Semantic Web is OWL. Annotation tools usually generate RDF triples which describe the instances.

A3.6.5 Requirements on ORCHESTRA Meta-Information Models

This section establishes a list of requirements on ORCHESTRA meta-information models.

A3.6.5.1 A list of purposes of the ORCHESTRA architecture SHOULD be established

Derived from: This requirement is derived from the combination of requirement "A3.6.2.1 Meta-Information CAN be classified according to a particular purpose", RM-OA "Usability" requirement, RM-OA "Rigorous Definition and Use of Concepts and Standards", and RM-OA "Self-describing Components" principles.

Explanation: A list of "particular purposes" of the ORCHESTRA architecture is inherently open and application dependent. Nevertheless, some "particular purposes" are relevant to every ORCHESTRA network, and knowing them in advance is essential for defining the services, service methods and ORCHESTRA data types. In addition to this, knowing some of the purposes in advance allows re-use of the existing standard solutions suitable for a particular purpose (e.g. ISO 19119 for Geo-referenced data), and an extensive list of purposes is essential for defining the related data types and establishing the high quality set of rules for building the meta-information models in RM-OA Annex B1.

Note: An initial list of purposes will be presented in section A3.7 "Particular purposes".

Example: The most obvious example of such a purpose relevant to every ORCHESTRA network is data discovery.

A3.6.5.2 A machine-readable representation of meta-information SHALL be provided

Derived from: This requirement is derived from the architectural consequence of "Self-describing Components" (RM-OA, section A2.1.4.7).

Explanation: In order to process meta-information by machine, it needs to be represented in a machine-readable format. The representation format MAY depend on the methods of processing applied to the meta-information. For instance, simple access to certain meta-information fields can be realised, for example, through a feature access service (FAS), while the drawing of conclusions from certain facts requires rule-based knowledge representation familiar in expert systems and knowledge management systems. The choice of the representation format MAY be further influenced or constrained by existing standards on meta-information specifications.

Note: Design of the representation level SHOULD try to constrain the number of representation formats and repositories.

Example: Meta-information structured according to ISO 19115 could be described as resources as specified in the Resource Description Framework (RDF) of W3C, because this is a more generic representation format. Such an approach could be useful if both types of meta-information are to be processed.

A3.6.5.3 Standard meta-information data types SHOULD be used where possible

Derived from: This requirement is derived from the combination of requirement "A3.6.5.10 ORCHESTRA meta-information model SHALL be able to integrate arbitrary standards", RM-OA "Openness" requirement and RM-OA "Rigorous Definition and Use of Concepts and Standards" principle.

Explanation: In order to achieve the high-level requirement of "Openness", re-using existing meta-information data types (e.g. ISO standards) SHOULD be preferred to defining new data types.

Note: Most, if not all, of the standard data types will be agreed upon on the OSN level, or even at the level of a single community that is active within the OSN, rather than at the architectural level.

Example: ISO 19119 Standard SHOULD be used for all the geo-referenced meta-information. SI units SHOULD be used for all the measurement values.

A3.6.5.4 Special meta-information related data types SHOULD be defined where appropriate

Derived from: This requirement is derived from the combination of requirement A3.6.2.1 “Any data MAY be interpreted as meta-information for a particular purpose”, RM-OA “Usability” requirement, and RM-OA “Self-describing components” principle.

Explanation: Although any data may be interpreted as meta-information for a particular purpose, some data may actually be generated with a purpose of being the meta-information for a particular purpose. Tagging such data as meta-information, e.g. with a special naming convention or by means of special data types, improves the usability of the system by lowering the system entropy, associating the implicit semantics with the data and making it inherently self-describing.

Note: As OA is built for a change, this must be possible in a running OSN, not only during the design phase! The possibility to do so is assured trough requirement A3.6.4.7.

A3.6.5.5 A mechanism for providing meta-information at any aggregation level SHOULD be provided

Derived from: This requirement is derived from the combination of requirement A3.6.1.4 “General considerations” and A3.6.5.8 “ORCHESTRA meta information model SHALL support sophisticated knowledge organisation ”

Explanation: For a specific object of concern, the meta-information model has to define meta-information necessary to fulfil a specific purpose. This goes far beyond simply attaching a set of “meta-attributes” to an object, since both the object of concern and the meta-information are not always “object” and “attribute” in an object-oriented sense. Aggregation in this context means that any type of information may be composed of any type of information; each part of a whole may stay existent even if the whole is destroyed.

Objects of concern and meta-information

- can be of any type and any structure (see requirement A3.6.1.4)
- can stem from different sources (see requirement A3.6.5.8)
- can occur at any aggregation level (see Figure 1)

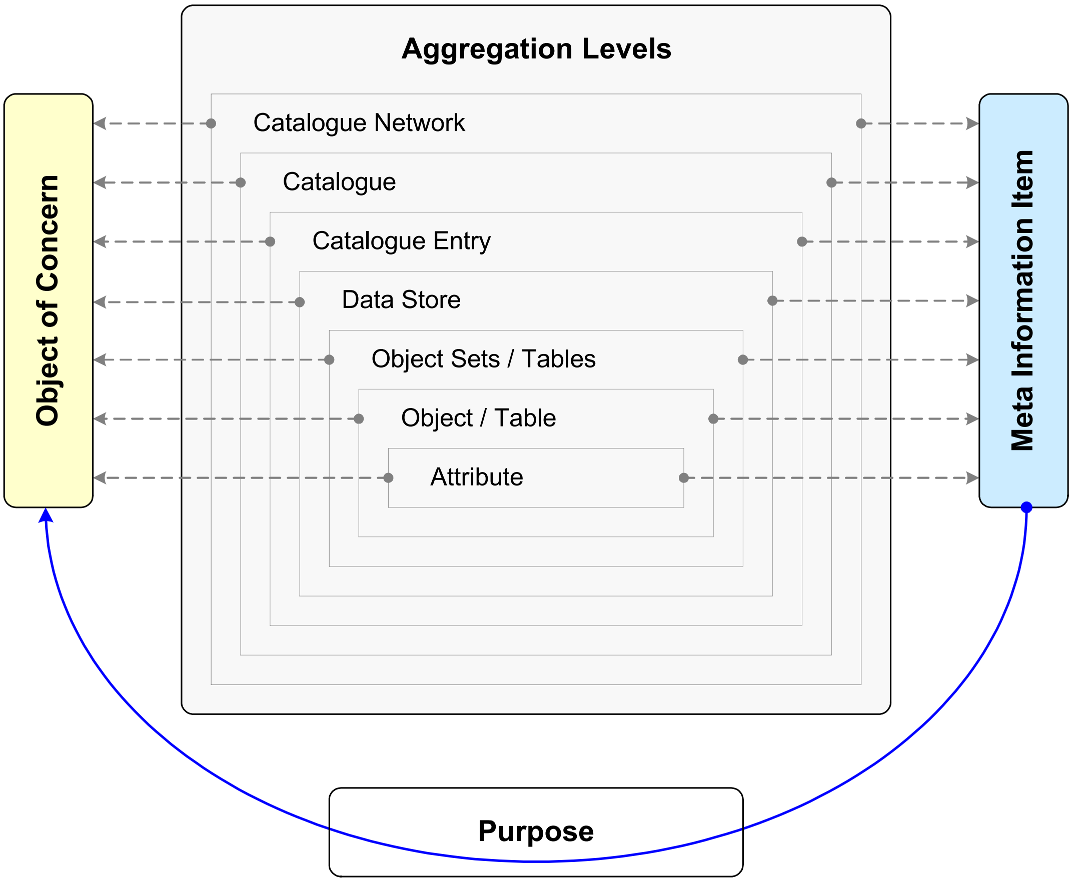


Figure 1: "Anything can be meta-information of anything" (simplistic diagram)

Consider sets of meta-information for different items on different aggregation levels, such as attributes, objects, object sets/-containers, data stores, catalogue entries, catalogues, etc. Each set of meta-information has items where each can be of any type or level, no matter where the meta-information belongs. For example it is possible that a set of meta-information for an attribute can be a catalogue in which any meta-information item can have attached meta-information itself, representing a network of meta-information.

A3.6.5.6 ORCHESTRA meta-information model SHALL support arbitrary meta-information relations ("Anything CAN be meta-information of anything")

Derived from: This requirement is derived from A3.6.2.1 "Any data MAY be interpreted as meta-information for a particular purpose", and related to RM-OA "Generic Infrastructure" and "Self-describing components" principles. This requirement is strongly connected with the requirement A3.6.5.5 "A mechanism for providing meta-information at any aggregation level".

Explanation: Arbitrary relations between meta-information and objects of concern shall be possible, e.g.:

- There may be meta-information for a particular purpose potentially at any aggregation level
- Elements of all aggregation levels can potentially be used as meta-information for a certain purpose
- The aggregation levels of the objects of concern and the corresponding meta-information are not directly related

Examples: In the following examples “object of concern” identifies the target where meta-information can be attached for the given purpose.

1. A whole data set as meta-information for an attribute. Purpose: “interpretation by humans”:

Object of Concern	An attribute “risk category number” of an object of type: “contaminated site”.
Meta-Information	A set of measurement results e.g. triples of the form: substance - measurement value – unit.

2. An object as meta-information describing a data source. Purpose: “discovery by navigation”

   See also the **Example** in A3.7.1.2 for more details.

Object of Concern	Any data source.
Meta-Information	Directory entry “Organisation B”.

3. A catalogue entry as meta-information for another catalogue entry. Purpose: “discovery”

Object of Concern	Catalogue for a node “organisation/institution”.
Meta-Information	Any link to a catalogue entry for another organisation having the same type of data or additional data or …

4. An attribute as meta-information for a data set. Purpose “quality-assessment” or “quality-ranked discovery”:

Object of Concern	Any data set.
Meta-Information	Any quality attribute, which may be only temporary (on demand) available (e.g. calculated by some algorithm solely for this purpose).

5. A data set as meta-information of a data source. Purpose: “integration”

Object of Concern	Any data source.
Meta-Information	Description of the underlying data model or a mapping of the data source’s data model to another one. This description can be any arbitrary set of structured data, e.g. an XML-document, a collection of SQL queries, a UML model, …

6. An object as meta-information for an attribute. Purpose: “interpretation”

Object of Concern	An attribute “protection category” of an object “water protection area”.
Meta-Information	A document describing the legislative context behind a protection category for a specific water protection area.

7. An object as meta-information of a catalogue. Purpose: “OSN operation”

Object of Concern	Any catalogue.
Meta-Information	Responsible administrator.

8. A catalogue as meta-information of a data source. Purpose: “OSN operation”

Object of Concern	any data source
Meta-Information	list of catalogues where the data source is referenced

9. A data-set as meta-information for a service. Purpose: “Discovery”

Object of Concern	any service
Meta-Information	A description of the service, e.g. its capabilities.

Figure 2 shows a scenario, in which a service or an administrator of a database wants to know where a specific database is registered, so the affected catalogue’s responsible administrator can be notified about changes of the database’s model.

To make this possible, a list of referrers could be attached to a data source as meta-information.

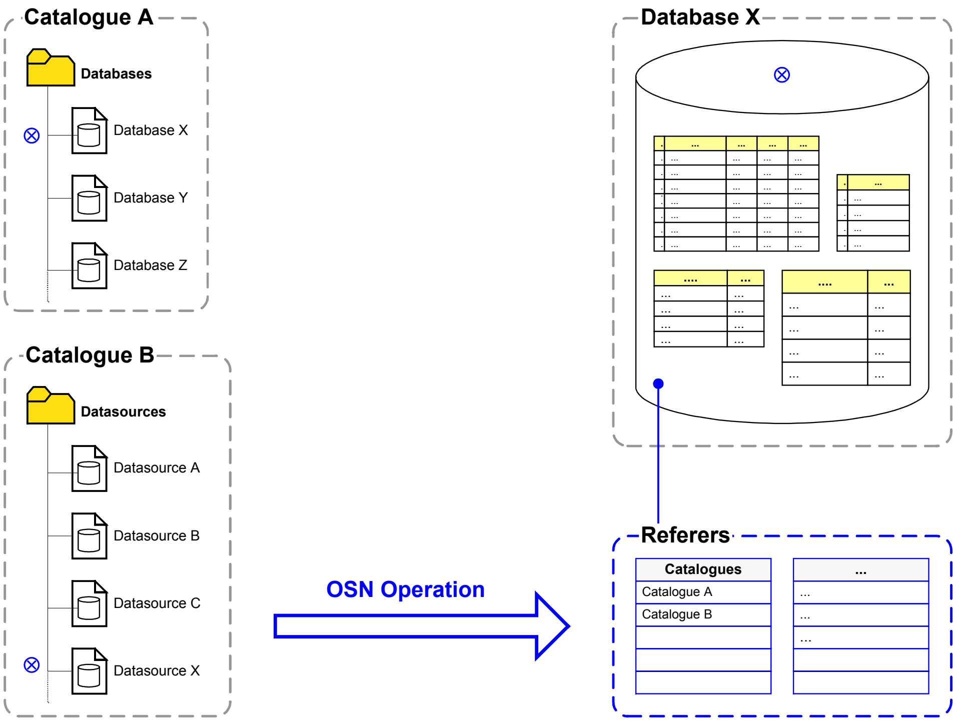


Figure 2: Meta-information is also required to operate the OSN

A3.6.5.7 ORCHESTRA meta-information model SHALL support arbitrary navigational structures

Derived from: This requirement is derived from the “Loosely Coupled Components” (RM-OA section 6.3.2) and “Component Architecture Independence” (RM-OA section 6.3.5) RM-OA architectural principles

Explanation: Since the ORCHESTRA-network will be highly distributed and heterogeneous the systems’ ability to provide proper navigation paradigms relies on supporting arbitrarily complex navigational structures as:

- List
- Tree
- Directed Acyclic Graph

- Network/Directed Graph
- Labelled Directed Graph/Semantic Network

In a directed graph labels of edges are capable of outlining semantics and hence determine semantic relations between objects. Since every piece of information can in principal be meta-information and there is no predefined structure in information and hence in meta-information, so ORCHESTRA's catalogues have to support all kinds of structures.

- Cascaded and interlinked Catalogues

Multiple catalogues participating in ORCHESTRA have to be able to cooperate, so that an overall view on the entire network can be gained. In other words the collaborating catalogues can represent a distributed catalogue. In addition to cascading catalogues this collaboration has to work on all levels e.g. links between arbitrary catalogue entries across various catalogue services have to be an integral part of the ORCHESTRA catalogue model.

A3.6.5.8 ORCHESTRA meta information model SHALL support sophisticated knowledge organisation structures

Derived from: This requirement is derived from the key system requirement "Usability" and the architectural consequence of "Self-describing Components" (RM-OA, sections A2.1.3.3 and A2.1.4.7).

Explanation: Meta-information stored in a knowledge-base needs to be organised such that the elements it describes (features, services) can easily be discovered. Concepts and utilities for knowledge organisation can be:

> catalogues, glossaries, taxonomies, classifications, thesauri, semantic networks, ontologies, frames, axiomatic systems, predicate logic.

Services implementing these concepts and utilities provide facilities for search and navigation in the meta-information base in order to retrieve elements described by the meta-information.

The discovery of features and services may involve semantic processing. If, for instance, features and services are described by means of semantic rules (and not modelled explicitly) then logical reasoners will be involved in the discovery process.

Knowledge represented in a repository in conjunction with methods for organisation builds a knowledge base (in simple cases, this may be a catalogue).

The result of a search for a service offering functionality as described by means of a semantic description need not be just a single service, but could be a number of services to be performed in a certain sequence. Invocation of the complete service then needs to be controlled by a workflow engine.

Note: This requirement indicates the need for knowledge organisation related services including catalogues, thesauri, ontology management services, logical reasoners, mediators and workflow engines.

Example: If the meta-information repository is an RDF triple store, structuring could be done by means of an ontology specified in OWL. Concepts specified in ontology provide for organisation of the instances. Applications on the usage level can navigate the semantic network provided by the ontology in order to explore the knowledge base.

A3.6.5.9 ORCHESTRA meta-information model shall be able to integrate meta-information of arbitrary origin

Derived from: This requirement is derived from "A3.6.1.4 General considerations" and "Technology Independence" and "Component Architecture Independence" of RM-OA architectural principles (RM-OA sections 6.3.3 and 6.3.5).

Explanation: As a considerable amount of meta-information can be derived from existing information systems, ORCHESTRA has to provide the facilities to support the integration and the creation of meta-information from arbitrary formats and information sources.

More precisely, the origin of meta-information shall not be predefined and the representation of meta-information shall not be restricted to certain formats or information models.

Note: This demand clearly leads to the requirement for services (OA services and/or tools) that are able to automatically access and extract meta-information from "arbitrary" data sources (relational database, object oriented database, file system, etc.) as well as services supporting the manual annotation of data.

A3.6.5.10 ORCHESTRA meta-information model SHALL be able to integrate arbitrary standards

Derived from: This requirement originates from the requirement that the overall OSN has to a) rigorously use concepts and standards (cf. RM-OA, section 6.3.1 "Rigorous Use of Concepts and Standards"), b) have a design for change (cf. RM-OA, section 6.3.4 "Evolutionary Development - Design for Change"). The ORCHESTRA meta-information model also has to comply with these requirements.

Explanation: The anticipated ORCHESTRA meta-information model shall be able to integrate arbitrary standards. During the operation of an OSN, communities of clients will evolve. Inside each community, specific standards (like ISO-19115 or CSDGM) will be used to achieve interoperability between these clients. Against this background, ORCHESTRA will have to support several levels of interoperability (see also ORCHESTRA's specification of the Schema Mapping Service).

Going beyond this is the integration of different standards so that it is possible to represent information given in one standard by means of another standard. This would e.g. enable an "ISO-19115"-community to also use information provided by a "CSDGM"-community. Obviously, it is not possible to create complete transformations/mappings for every combination of standards, but ORCHESTRA will provide as much interoperability across standards as possible. Wherever there is an intersection or a shared concept, a mapping from one standard to another standard for the according concept is possible.

A3.6.6 Other requirements related to ORCHESTRA meta-information models

This section contains several requirements that appear important but do not fit in any of the above categories. They are listed here in order to assure they aren't forgotten.

A3.6.6.1 ORCHESTRA infrastructure SHOULD be functional without the semantic services

Derived from: This requirement is the consequence of the technical immaturity of the semantic services, and SHOULD be re-considered in later revisions of this document.

Explanation: To the best of our knowledge, the capability of the currently available semantic (ontology-based) services is far from the expectations of the ORCHESTRA architecture. Consequently, early implementations of the ORCHESTRA infrastructure SHOULD NOT count on semantic services assuring the full interoperability within OSN.

Semantic services SHOULD be regarded as a way to improve the discovery, integration, interpretation and capabilities of the OSN. ORCHESTRA services and applications may use the semantic services if available, but should not require them in order to function correctly.

A3.6.6.2 ORCHESTRA services SHOULD provide a generic user interface for authors of meta-information

Derived from: This requirement relates to the fundamental challenges "Integration/Collaboration" and the anticipated "Long Lifetime" as identified in the RM-OA (RM-OA, section A2.1.2.2 and A2.1.2.3) and is derived from RM-OA's key system requirements "Openness", "Scalability" and "Usability" and the architectural consequences "Evolutionary Development – Design for Change" and "Generic Infrastructure" (RM-OA, section. A2.1.4.4 and A2.1.4.6).

Explanation: Meta-information may be captured by human users, who could be located anywhere. Therefore, a distributed authoring environment providing input forms for users is needed. The information to be provided must be transferable to an internal OSN representation format (see A3.6.4.10 and A3.6.5.2). As the structure of meta-information is expected to evolve during the lifetime of the entire OSN (e.g. because new meta-information with a new structure is considered), new input forms may need to be generated at any time. Thus, there is a need for automatic generation of input forms based on the specified structure of the meta-information.

A3.6.6.3 OSN should provide storage for meta-information

Derived from: Not directly mentioned in RM-OA but might relate indirectly to the fundamental challenges "Integration/Collaboration" and the anticipated "Long Lifetime" as identified in the RM-OA (RM-OA, section A2.1.2.2 and A2.1.2.3)

Explanation: In general meta-information (for services as well as for data) has to be stored as part of the OSN. Therefore storage has to be allocated within the OSN and it will be stored where it is used. For this purpose we distinguish between two cases:

> Case 1: Services like Feature Access Service or Catalogues (Meta-information Catalogue, Catalogue of Services) where meta-information is stored permanently.
>
> Case 2: Services that store meta-information only temporarily (meta-information is retrieved on the fly), for example, mediation services.

ORCHESTRA Services should provide sets of information as attributes which are used by others (services) as meta-information. To reach the completeness of meta-information it might be needed that meta-information is entered manually (see also A3.6.4.11 and A3.6.5.9). But the goal is to automate this process as much as possible.

Example: Possible usage of meta-information storage

As all ORCHESTRA services and clients should refer to ORCHESTRA standards (like predefined ontologies, feature types and well-known service descriptions) a common repository for these data (standards) should be made available.

Adaptors connect individual data/information sources with the OSN, meaning, in principle, that an adapter is the link from the local systems' (legacy systems) ontologies, feature types, services, access control lists, data(base) structure etc. into the global system (OSN). Therefore it has to have an interface at the back end to the local systems and also at the front end to the OSN.

To perform transformations the adapter has to have mapping information (configuration parameters) for all objects to be presented to the OSN. The adapter needs also physical space/storage in which to run and store its data. That can be a node in the OSN which is provided and maintained by an ORCHESTRA partner or any other kind of service or data provider.

In order to make it simple for data or service providers to plug into the OSN network or to program their ORCHESTRA services there should be an ORCHESTRA Connector Framework (a kind of class library usable, for example, for the front end development of an adapter or as front end development of an service).

Mediation services (needed, for example, to harmonise data coming from different data/information sources) will probably only need temporary storage.

Requirements:

- An ORCHESTRA Connector Framework (OCF), implementing the interface of an ORCHESTRA service, is needed to help programmers to create new ORCHESTRA services.
- To connect existing legacy systems a Generic ORCHESTRA Service Adapter (GOSA) should be provided. The goal is to be able to convert the legacy system to an ORCHESTRA service only by configuring the GOSA.
- Some existing legacy systems might be too complex for the GOSA. In such cases programming of an individual adapter using the OCF will be necessary.
- ORCHESTRA Standards Repository (OSR). It will store information about all predefined ORCHESTRA ontologies, all predefined feature types, all definitions of well-known services.

A3.7 Particular purposes

As stated in requirement A3.6.5.1 A list of purposes of the ORCHESTRA architecture SHOULD be established". These purposes are a starting point for developing the ORCHESTRA services, service methods, and ORCHESTRA data-types.

Explanation: The list of "particular purposes" of the ORCHESTRA architecture is inherently open and application-dependent. Nevertheless, some "particular purposes" are relevant to every ORCHESTRA network, and knowing them in advance is essential for defining the services, service methods and ORCHESTRA data types. Basic ORCHESTRA data types as well as rules for building meta-information models will become part of D3.3.2.

This section identifies some of these purposes needing meta-information. For each purpose, examples illustrate types of meta-information needed for this purpose.

Note: The list of purposes in this section is by no means complete and should be seen as a starting point. In particular the list is heavily depending on user requirements, which addresses vital purposes within the ORCHESTRA architecture.

A3.7.1 Discovery

Discovery helps users to iteratively narrow the set of objects of concern until only relevant results remain. In the broad field of discovery we distinguish between "navigation" and "search". These two purposes are discussed in sections A3.7.1.1 and A3.7.1.2.

Relevance: Discovery's purpose is ***relevant to most users and many OA (ORCHESTRA Architectural) and OT (ORCHESTRA Thematic) services in every OSN***. For the great majority of users, data & services discovery is usually a first step to solving the problem at hand.

Affected services: From the user's point of view, discovery is usually mediated by a Catalogue Service (standard OA data & services search/navigation service), Gazetteer Service (geo-search & navigation by geographic name), or Inferencing Service (Ontology based search & navigation).

These services in turn rely on the search & navigation capabilities of the various Access Services to gather the data in the first place, and may use an Annotation Service (semantic meta-information generation), Document Indexing Service (automatic generation of document search indexes), Format Conversion Service (converts between data formats), Schema Mapping Service (converts between schemas), or Thesaurus Access Service (synonym and antonym repository for data vocabulary terminology) for building the search indices and navigation trees.

A3.7.1.1 Search

In our context ***search is the process of discovering information by definition and execution of a query***.

A query describes a set of properties of relevant objects. The properties represent the search criteria defined by the query's context. For example, a query in the context of a geospatial search would most likely consist of properties that are some kind of spatial reference (bounding box, etc.)

Searches can be conducted by:

- Defining the query (e.g. by entering a keyword or phrase or by describing properties of valid results; every query could be combined with boolean operators, regular expressions, ...)
- Initiating a search process (e.g. one of: full text search, geo-spatial search, temporal search, semantic search)

Searches return a list of zero or more results to be reviewed by the user or by the calling process. Depending on the service, the results may be paged or hierarchically organised. Starting with these results, users may either pick up some of the results, refine the search query and repeat the search, or continue the discovery process with the navigation (A3.7.1.2)

Examples:

Two common examples for discovery by search are:

- textual search, in which a user might be interested in documents that contain specific keywords or whole phrases, and
- spatial search, in which a user wants to find geographic features or thematic maps covering a certain area.

Exemplary objects of concern, meta-information and services for “search” purpose are shown in Table 3.

textual search:

	Name	Description
Object of Concern	Document	Textual document of interest
Meta Information	Key word list	For each object a list of key words can be used for the search. The keyword list is extracted from the object of interest or added manually.
	Full text index	Index representing a model of the whole document.
Services	Document Search Service	Full text search over indexed textual documents or keyword lists.
	Document Access Service	Allows a client to download a document from a document store.

spatial search:

Object of Concern	Map	Spatial representation of our area of interest
Meta Information	Bounding areas	A bounding area can be defined and used for search.
	Gazetteer mapping	Search by name of geographic objects (street, address, district, etc.)
Services	Gazetteer Service	A gazetteer service allows to relate a geographic name (e.g. city, lake, region but also street) to a geographic location (i.e. a point, line, polygon or sets of these; might be also post codes=polygons) and vice versa.

Table 3 – Exemplary objects of concern, meta-information and services for the purpose search (textual, spatial)

A3.7.1.2 Navigation

Traditionally ***navigation is an iterative and interactive process as the user normally browses within navigational (catalogue) structures***.

Navigation can be conducted by usage of two kinds of catalogue:

- Static catalogue

  The user has to follow the catalogue's predefined data models; at best the user is supported with hints on the underlying structuring-principles via additional meta-information.

- Dynamic catalogue

  The user is not interested in the actual underlying model or how the data is categorised by its provider. When using a dynamic catalogue, the user retrieves an on-the-fly classification of the information according to his individual point of view.

A dynamic catalogue is obviously a perfect tool for navigating in a semantic network, because it is the nature of a semantic network to be highly dynamic.

Example: A user wants to find data sources by navigating across organisations or agencies. After having found the appropriate data, he wants to know to which organisation the data source belongs. To meet this user requirement, a catalogue of data sources can be built, in which each data source contains its corresponding organisation as meta-information.

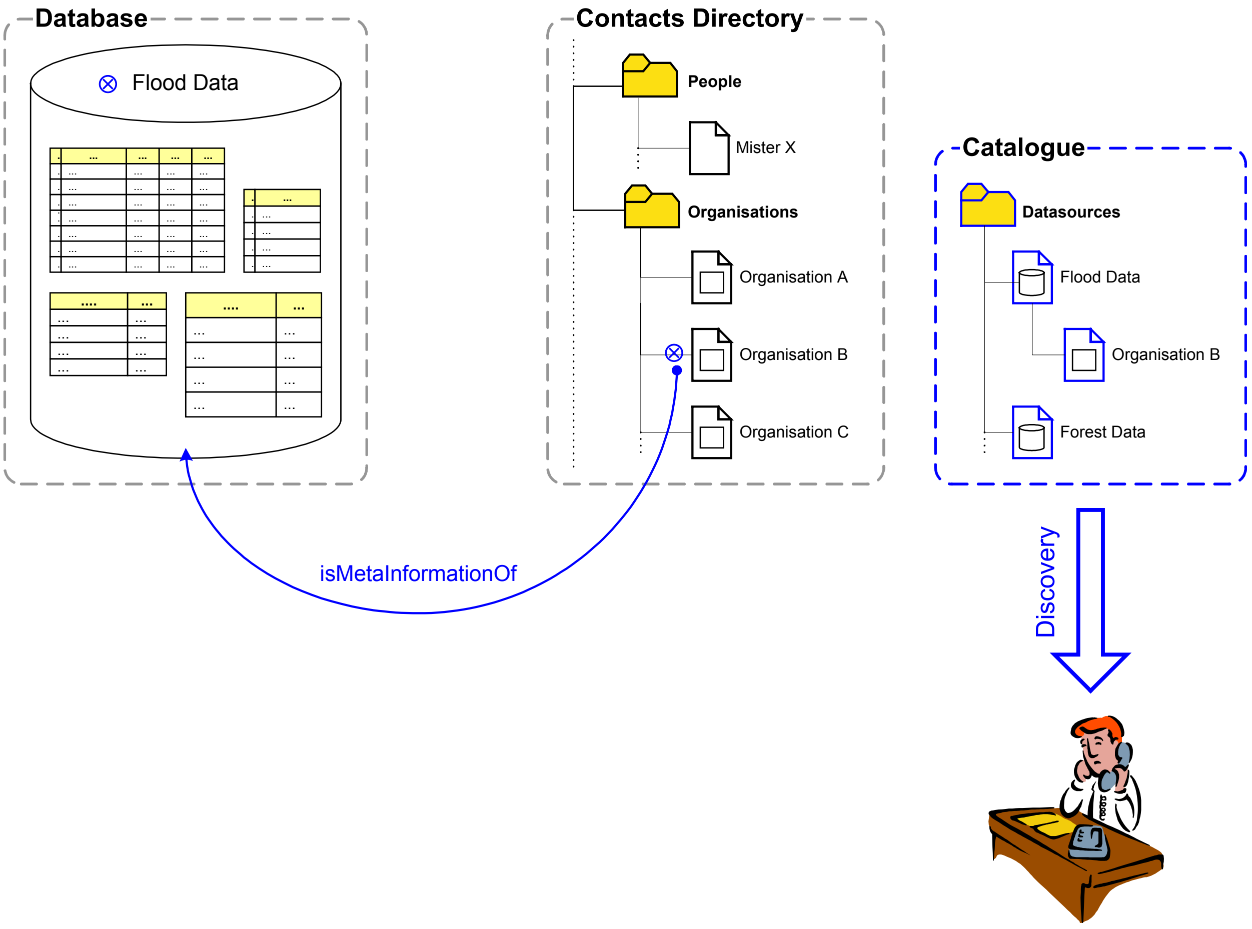


Figure 3: Catalogue for the discovery of data sources and their responsible organisations

Discovery		Na	De
Object of Concern		Da	Da
Navigation	Meta Information	Catalogue entries	References to data sources, organisation, …
		Catalogue structure	Representation of the relationships between data sources, organisations, …
	Services	Catalogue Service	Catalogue services support the ability to publish and search collections of descriptive information (meta-information) for data, services, and related information objects.

Table 4 – Exemplary objects of concern, meta-information and services for the purpose discovery - navigation

A3.7.2 Data access and service invocation

Different types of meta-information are required to facilitate the access to data sources and services. In ORCHESTRA we define two categories of access: “data access” and “service invocation”. Service invocation refers to any communication between the service user (client) and the service provider (server). Data access is a special type of service invocation with the main purpose of retrieving the data.

Relevance: Service invocation is the most basic mechanism of the OSN and thus relevant to all of the users and all of the ORCHESTRA services. Data access is the most important form of the service invocation for service providers and end users, and relevant to most services including all the “access” services and all the services that process data.

Affected Services: Service invocation affects all services. Data access primarily affects access and catalogue types of services such as the Catalogue Service, Feature Access Service, Document Access Service, Map Access Service, Sensor Access Service, Formula Access Service and Coverage Access Service. Other services, such as the Document Indexing Service, Format conversion Service and Geospatial Processing Service are indirectly affected.

A3.7.2.1 Data access

Access to data is a special form of the service invocation used for storing and retrieving data (e.g. storing a document on an OSI implementing a Document Access Service). The importance of the data storage and retrieval for the OSN is illustrated by the fact that the RM-OA defines different main types of access services (see above), and that most of the other OA and OT services interact with these access services in some way.

Example: A simple example for data access is the download of a file from an FTP server. The required meta-information includes the URI of the file and the login information.

A3.7.2.2 Service invocation

Service invocation is a communication between a service user and a service provider. The user application invokes services offered by one or more provider applications by sending request messages and processing response messages.

Example: A simple example could be the invocation of services’ operations, where the user decides which operations are the appropriate ones for his purposes on the basis of the capabilities-information of the services.

Access / Storage / Invocation		Na	De
Object of Concern		Fil	Fi
	Meta ion	Location	Location of the data source, including a protocol needed to access the data source.
		Login	If authentication is required.
	Services	Feature Access Service	A feature service allows interoperable access and transactions on features available in an OSN, it supports queries to select certain features based on their type, certain attribute values, and/or their spatial and temporal extent.
Object of Concern		Service	Invoked service
	Meta ion	Location	Location of the service, including the protocol.
		Login	If authentication is required.
		Interface	Description of the service interface.
	Services	Service Chaining Service	Definition / processing of service chains.

Table 5 – Exemplary objects of concern, meta-information and services for the purpose Data access/ Service invocation

A3.7.3 Integration (Collaboration)

One major purpose of ORCHESTRA is the integration of data sources and services into the OSN. This purpose is further divided into “data integration”, i.e. assuring that the data offered by one service are usable for another service or for the end user, and the “service integration”, i.e. assuring that services can be chained.

Relevance: Integration of data sources and services is of major interest for data & services providers, end users and most software developers[1]:

- Data integration is the major problem for the end users today. Solving the problem of data integration on the OSN level would greatly simplify writing of the services, client applications and would turn the world into much better place for data providers and end users.
- Services integration is the next step towards achieving more with less effort: rather than attempting to write huge and very expensive programs that attempt to do everything, smaller programs that excel at one task are combined in order to solve complex problems[2].

A3.7.3.1 Data integration

Data integration works through composition of

- data discovery
- data mapping
- data retrieval (see A3.7.2 Data access and service invocation)

An interface to retrieve data (data access service) is based upon a common service description. The main focus here lies on data structures retrieved, information on the whereabouts of data, and the relations between data items. A proper description has to include the syntax, structure, and semantics of the data in order to provide an insight into the underlying conceptual model.

Relevant standards for data and schema integration can be found in the field of model integration and schema integration (Barkmeyer E. J. et.al 2003).

A3.7.3.2 Service integration

Service integration works through description of service interfaces as well as service functionality. At first glance the interface description is mainly a technical one dealing with network-technology, syntax and structures. The result of this is a protocol used to interact with a specific service. (see A3.7.2 Data access and service invocation) This can be done

a) in a human readable way and/or

b) in a machine readable way

The functional description’s main objective is to clarify whether a service fits a specific task or not (for further details on “interpretation” refer to section A3.7.4).

Relevant standards in this context are the ISO-19119 “Geographic information – Services”, OWL-S and WSMO (refer to A3.8.2).

Examples:

[1] The only software developers that are likely to be unhappy are the established players with huge monolithic applications that may see their profits and market share diminished as users realize that they can do their tasks better and at a fraction of the cost by combining the new tools..

[2] The idea of combining simple tools that excel at one task has been around for over 30 years (e.g. UNIX tools!), came out of fashion in the nineties, and received a renaissance as people realized the real price and limitations of depending on the huge monolithic programs.

A flood forecast model and a database containing meteorological data have to be integrated. It should be possible to use the database as input for the simulation model and the model's output as input for any other integrated service.

Integration		**Name**	**Description**
Object of Concern		Flood forecast model	A simulation model.
Service integration	**Meta Information**	Model description	Human-readable description of the model's functionality. Can help to evaluate the applicability of the model to certain data, regions, etc.
		I/O format description	Computer-readable description of the model's input and output format.
		Invocation parameters	see A3.7.2 Data access and service invocation
	Services	Simulation Service	Run a simulation model.
Object of Concern		Meteorological data	This data source may serve as input for a simulation model.
Data integration	**Meta Information**	Data description	Description of data structures, e.g. data model, model language, …
		Access Parameters	see A3.7.2 Data access and service invocation
	Services	Conversion Service	Convert data into the proprietary input format of a simulation model.

Table 6 – Exemplary objects of concern, meta-information and services for the purpose integration

Example 2: Actors in an OSN may use different terms for the same type of objects or similar terms for different types of objects. In order to assure interoperability, every actor can, for instance, associate the data type with his data, and publish a conversion schema for transforming from his data into some kind of the "standard" data.

This can involve various types of meta-information, such as data models that contain syntactical and structural information about data, mappings that contain transformation rules for data models or semantic models (e.g. expressed by an ontology) to support the derivation of transformation rules.

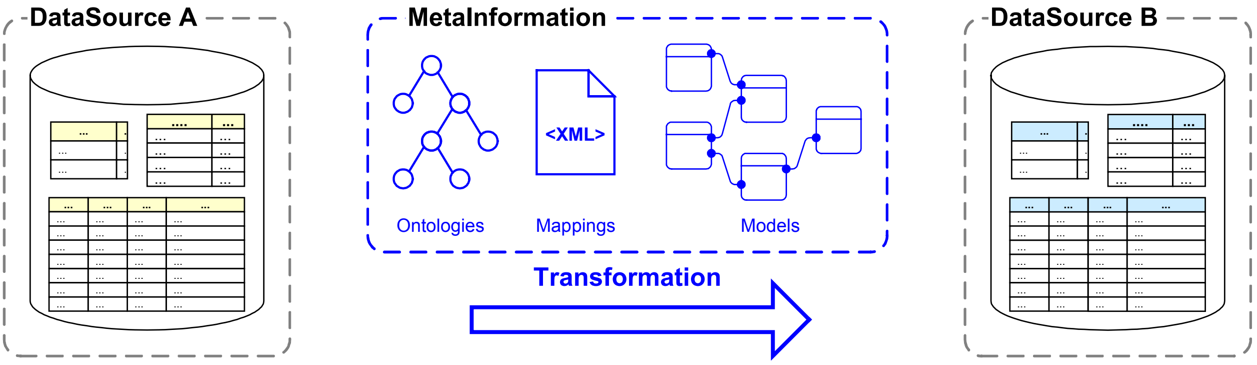


Figure 4: Meta-information for data transformation

A3.7.4 Interpretation

Interpretation describes the process of explaining and understanding a certain issue. The main goal of meta-information for interpretation is to enhance the understanding of information by users/processes to enable them to make better use of it.

Meta-Information for interpretation can be classified in two main categories:

- **Implicit:**
  Only the user/process possesses the semantic information necessary to understand the related information. So the meta-information is only an input/hint/fact for interpretation.

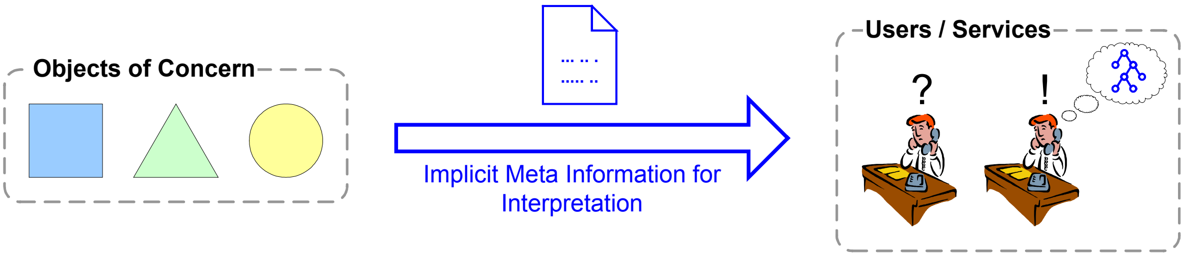


Figure 5: Implicit meta-information for interpretation

- **Explicit**
  The set of meta-information contains semantic information making it self explanatory up to the level of full interoperability.

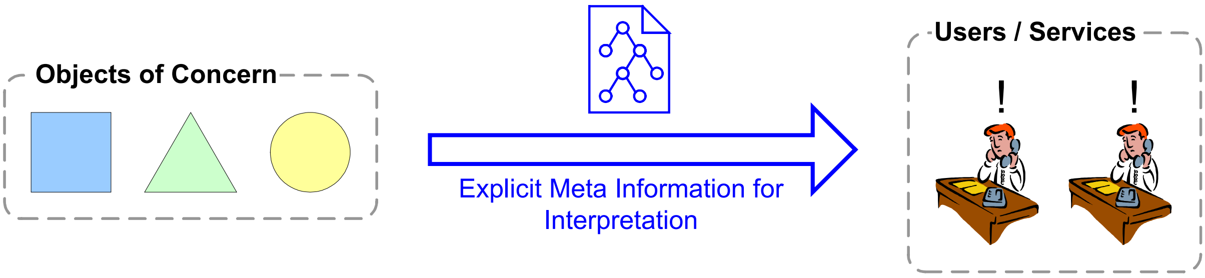
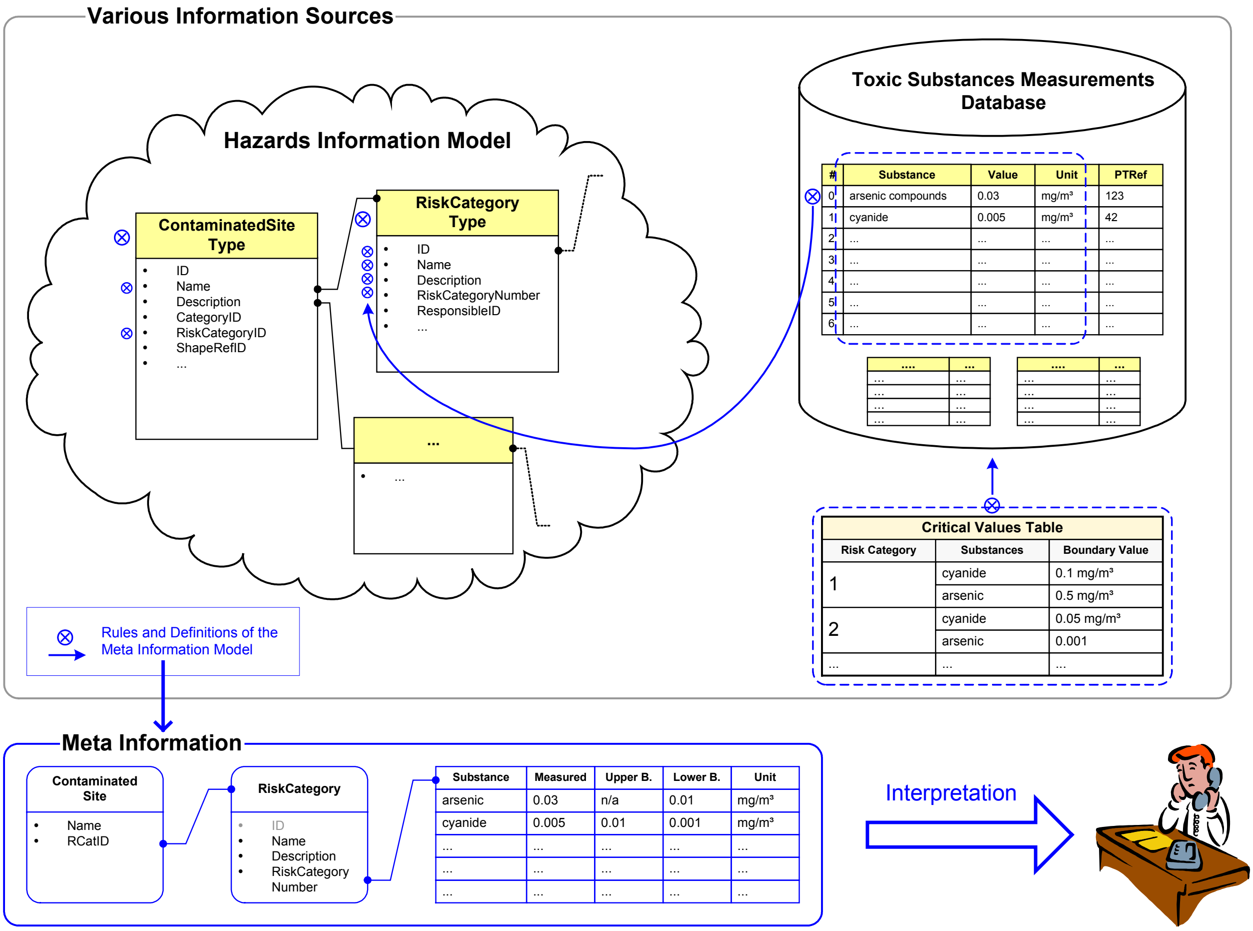


Figure 6: Explicit meta-information for interpretation

Relevance: The importance of providing meta-information for interpretation rises very quickly with the number of actors, number of data types, and time. In a typical OSN, all of these numbers will be large, leading to a high probability of encountering “unknown” data and services for all the users. In addition, a common understanding between actors is a fundamental to guaranteeing semantic interoperability. This means that each actor shall be able to check if he has the same understanding as the collaborating actor in different situations and at different levels of detail and abstraction.

The more complex case concerns the understanding of concepts behind terms. Even in the case of a common understanding there may be differences in the level of detail or in the data models. But there may also be very different understandings of the concept itself which have to be discovered to avoid the incorrect use of data.

Example:
Figure 7 shows a real world example of a user responsible for the evaluation of building applications. Although he only needs some information about contaminated sites and their classification according to risk categories, he has no access to the database that contains all measurements of toxic substances. But in some cases he might have to explain the origin of the category number. Therefore he needs the specific measurement values along with the corresponding critical values that caused this classification.



#	Substance	Value	Unit	PTRef
0	arsenic compounds	0.03	mg/m³	123
1	cyanide	0.005	mg/m³	42
2	...	...	...	...
3	...	...	...	...
4	...	...	...	...
5	...	...	...	...
6	...	...	...	...

....	...
...	...
...	...
...	...
...	...

....	...
...	...
...	...
...	...
...	...

Critical Values Table

Risk Category	Substances	Boundary Value
1	cyanide	0.1 mg/m³
	arsenic	0.5 mg/m³
2	cyanide	0.05 mg/m³
	arsenic	0.001
...	...	...

Substance	Measured	Upper B.	Lower B.	Unit
arsenic	0.03	n/a	0.01	mg/m³
cyanide	0.005	0.01	0.001	mg/m³
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

Figure 7: Implicit meta-information for the interpretation of risk classifications

In the example above an OAS-MI has to be built containing an aggregated and homogenous view of the three different information sources. This set of implicit meta-information, tailored to the specific needs of the user, allows him to correctly interpret the given facts.

Interpretation		Name	Description
Object of Concern		Building applications	For evaluation of building applications.
evaluation	Meta Information	Contaminated sites	Information about contaminated sites and their classification according to risk category numbers.
	Services	Catalogue Service	Catalogue of contaminated sites.
Object of Concern		Risk category number	Risk according to the degree of contamination.
explanation	Meta Information	Critical Values	risk category number.
		Measurement Values	Specific measurement values that caused the classification.
	Services	Meta-Information Extraction Service	Service to generate and extract this meta-information.

Table 7 – Exemplary objects of concern, meta-information and services for the purpose interpretation

A3.7.5 User profiling

It is necessary to provide views of data and services and interaction procedures to support different types of users on a per user-/task-basis. This means there is a need to provide the capability for users to efficiently operate in a heterogeneous environment through information views tailored to their responsibility and authority. The user characteristics include issues like language, domain specific knowledge, computer interaction experience etc. Task characteristics include granularity of data, task specific navigation paths, task specific structuring of data etc.

To be able to associate the appropriate views and interaction procedures with different user groups and tasks, we need meta-information describing the relationships.

Currently the ORCHESTRA user types include

- service provider
- service / data integrator or administrator
- end-user (decision maker)

Example: The same task (e.g. Discovery/Navigation) has individual characteristics for different types of users. A catalogue entry's visibility and representation might depend on the user. The individual languages and views for users should be configurable.

User Profiling	Name	Description

Object of Concern		Catalogue	Should be internationalised and should provide different views for different users.
User Profiling	**Meta Information**	User Information	Information like user group (service provider, service / data integrator or administrator or end-user), nationality, etc.
		Language	English, German, French, etc., depending on the user's nationality
	Services	Catalogue Service	The catalogue builds the foundation of search and discovery facilities that we need in order to make full and efficient use of available information resources.

Table 8 – Exemplary objects of concern, meta-information and services for the purpose human computer interaction

A3.7.6 Authentication, authorization and accounting

Authentication, authorization, and accounting (AAA) is a term for a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage, and providing the information necessary to bill for services and/or information. These combined processes are considered important for effective network management and security.

AAA requires a special set of meta-information that is directly related to business rules, and is of little to no use for anything else.

Relevance: AAA is of primary interest for OSN administrators, and for the data & service providers with confidential data and commercially interesting data & services. Basically, AAA has no value in itself, but acts as an enabler for enforcing the business rules in the OSN. Certain forms of AAA will be required even in the case in which all the providers are willing to provide unrestricted access to their data & services, for instance in order to limit the number of actors with administrative access to some service.

Affected services: AAA affects all operations of all services in all OSNs. Read/write/modify operations are generally allowed only to "trusted" users, while read-only access to some services may be completely exempted from the AAA.

Note: This topic is still under (OA service) discussion. For example, no decision has been made about how granting of rights (authorization) will work. The requirement A3.6.4.4 "OA SHALL provide a mechanism for assuring the referential integrity and for handling the lack of referential integrity." shows that this question cannot be answered on a platform-independent level. In fact, typical decentralised OSN is likely to have several authentication authorities with distinct authorization profiles, and possibly even use completely different AAA paradigms. Consequently, additional mechanisms for trust management, identity federation and single sign on the OSN level will have to be implemented to assure all the data and services within OSN can be used by all the users (subject to business rules limitations).

A3.7.6.1 Authentication

Authentication is a method for identifying the actors (users and resources) in an OSN. Authentication systems provide answers to the following questions:

- Who is the actor?
- Is the actor really who he/she represents themself to be?

Actual mechanisms used for the authentication can be as simple (and insecure) as a plain-text password challenging system or as complicated as the Kerberos system.

In all cases, however, authentication systems depend on some unique bit of information known (or available) only to the individual being authenticated and the authentication system -- a shared secret. Such information may be a classical password, some physical property of the individual (fingerprint, retinal vascularisation pattern, etc.), or some derived data (as in the case of so-called smartcard systems).

ORCHESTRA architecture specifications cannot impose any limitations on the number and type of authentication systems used within OSN. Unless such limitations are imposed on the implementation level, every service provider in a typical OSN will be free to use its own authentication system.

Note 1: In order to allow transparent access to all the services, OA has to foresee some kind of a Single Sign On mechanism capable of authenticating against different services on behalf of the user.

Note 2: Depending on the business rules, authentication may be required for all the resources, not only for the users.

Example: Imagine a situation in which falsifying the data would bring significant monetary or political advantage to one of the stakeholders, for instance an automatic taxing system based on actual emissions of CO_2. The design of such a system would have to assure that no data can be falsified, e.g. by introducing a faked service into OS, modifying the data on the server by unauthorized users, or modifying the data in transport from the originating server to the end user.

A3.7.6.2 Authorization

Authorization protects resources by restricting usage of those resources to those actors that have been granted authority to use them. The authorization process is used to decide if actor X is allowed to make use of resource Y. In order to identify those actors the authorization process makes use of the authentication process.

Apart from a static authorization list the Authorization might be also based on certain dynamic restrictions like certain time or date constraints, maximum number of concurrent resource access or location based restrictions (e.g.: no rights granted to remote accessing actors).

Note: At this stage, it is unclear whether a single authorization can be imposed on the OA, or even on the OSN implementation specifications level. If this turns out to be impossible, special attention will have to be given to the problem of assigning the authorization levels to “roaming” users, i.e. mapping the authorization levels of one provider to authorization level of another provider.

A3.7.6.3 Accounting

Accounting refers to the information gathered on an actor’s usage of resources. This can, for example, include time periods or size of the resources. Accounting information can further be used to support billing, fair-use, planning and many other purposes. In that sense accounting information can be used by the authorization process in order to provide a basis for the the granting of usage rights.

Example: An actor tries to retrieve data about forest fires. The actor needs to provide the personal secret (known by the authentication service) to the authentication service to verify the identity of the actor. In the next step the authorization service takes over and grants the rights to the identified actor. After obtaining the rights to read the forest fire data, the actor can access the FAS and query for the data.

AAA		Name	Description
Object of Concern		Forest fire data set	Data set on forest fires the actor is interested in.
Data retrieval	Meta Information	Personal secret	Some piece of information that uniquely identifies the actor and is only known by the actor and the authentication service (e.g.: biometric features).
		Actor/right mapping list	List (or dynamically created information) given by the data provider that contains the actors and their rights.
	Services	Authentication Service	Uses a shared secret to verify the identity of the actor.
		Authorization service	Grants read rights to the previously identified actor.
		Feature Access Service	Stores the number of accesses and the size of data retrieved by the actor. Can be used by the data provider to bill the actor.

Table 9 – Exemplary objects of concern, meta-information and services for the purpose AAA

A3.7.7 Quality control / management

Quality control/management is needed when certain criteria need to be fulfilled by data and/or services. Quality usually has different aspects depending on whether services or data are considered. Another concept that is closely related to quality control is trust relationship. While trust relationship is a different concept and can be used for many purposes it might play an important role when one needs to decide if information regarding the quality of a service or data is trustworthy. So trust relationships might be needed, but are not mandatory for quality control.

Relevance: Quality control is important to every actor in every OSN, and especially relevant whenever data and services have to meet certain legal requirements. In the worst case situation, working with the data that have no quality information may be just as bad as working with randomly generated data.

Affected services: all non-trivial services in every OSN.

Note: Quality Management is highly dependent on the AAA.

A3.7.7.1 Service quality

Service quality in this sense means the infrastructural properties. Examples for these are response times or availability. Another aspect that can be considered to be an attribute of service quality is the fee one has to pay to use the service. Quality regarding the output of services, whether it's back to the actor invoking the service, passed on to another service or stored in an internal data repository is considered to be data quality of the service. This type of quality is important especially in the context of service chaining when accumulation of errors becomes an issue.

A3.7.7.2 Data quality

Data quality becomes an issue when working with those data. Quality refers many different things and only an open list can be given to characterise this term in the context of data:

- absolute and relative errors of measurement data
- computational errors of data processing services
- numerical issues
- minimum and maximum degree of detail in the values of a data set on a specific service
- sensitivity to error accumulation
- refresh period of the data (if it’s not only a repository for old data)
- …

Obviously the list of criteria for data quality can become quite long but this degree of detail is not always needed in order to classify the quality of data.

A3.7.7.3 Trust relationships

Trust management and relationships are tightly coupled with quality assurance. Trust comes into play whenever authenticated and authorized but unknown parties join a network. When providing their data and services to the network they can and have to apply meta-information regarding the quality of what they are exposing. But how can an actor be sure if this meta-information really represents the quality of the actual data and services? The actor’s only choice is to either trust or distrust the actor that attached the quality meta-information. Besides deciding whether to trust an actor or not, degrees of trust can also exist. Many different information items can be considered important for trust relationships so an excerpt of some is provided here.

- Information about the actor: e.g.: name
- Certificates the actor has been granted
- The organisation that the actor represents
- …

In order to trust an actor, that actor must be identified first, so trust relationship relies on the authentication process. Trust relationship is not mandatory but highly recommended to ensure the quality of a network. A network that does not foresee trust management can be seen as a network where every actor is fully trusted by default.

A3.7.8 Transactions, Synchronisation and Locking

The ORCHESTRA distributed architecture defines a set of services that are built with interoperability in mind. In order to use the OA to its full extent, different services need to be transparently combined into new “virtual services”. Using such service combinations to the full extent requires mechanisms and meta-information that support building transaction-secure composed operations on the OSN level. These mechanisms can be further separated into Transactions, Synchronisation, and Locking.

Relevance: Transactions are needed when certain tasks that involve resources need to be carried out and it is important to ensure that the resources are not altered during this process. Transactions usually are not needed for read-only services because no other actor can alter any resources that are available at that service.

Affected services: no specific list of affected services can be given. The decision about whether to support a transaction or not is up to the service implementer.

Note: Normalisation of data within OSN (requirement A3.6.4.1!) inevitably leads to inter-connected data that has to be kept in sync. Updating distributed data without transactions is dangerous in two ways:

- First, distributed data will inevitably become out of sync during the update procedure.

Accessing the data while they are still out of sync, can lead to unpredictable outcomes.

- Second, the update procedure may break during execution, leaving the data in an unsynchronised state. Consequently, application programmers have to invest a considerable amount of work in checking the data consistency and assuring that the update is eventually completed.

Neither of these problems occurs if all the changes are encapsulated within a single transaction.

A3.7.8.1 Transaction

A transaction is a logical group of operations that succeeds or fails as a group. This means that either all tasks within a transaction are carried out or none are. That way a transaction appears to be atomic. It can’t be interrupted by any other transactions or operations, nor will it leave the system in an inconsistent state. If one task within a transaction fails, all changes (if any) are revoked, leaving the system in the exact same state as it was before the transaction started.

Additionally the changes made within a transaction will only be visible after the full set of tasks was successfully carried out. No intermediate state will be propagated to the users of the system.

A3.7.8.2 Locking

A lock is a mechanism to (temporarily) restrict the access rights to a resource for certain actors. Locking is used to guarantee the atomicity of transactions. Resources that are used within a transaction are locked leaving only rights of the transaction unchanged. Other actors might have their rights reduced to read only or revoked completely.

To assure correct behaviour a lock must be acquired in an atomic way. Otherwise, two actors could simultaneously acquire the lock on the same resource, rendering the locking concept useless.

Care must be taken when using a locking concept in order to avoid deadlocks. A deadlock is a situation in which multiple actors hold locks on multiple resources and each actor requires a resource locked by another actor.

Example:

An organisation provides a write-accessible FAS as a data store for another organisation. Multiple people work on the maintenance of the data and update them if needed. If new data are available they must be inserted, but it also must be ensured that they’re inserted only once. A FAS that supports transactions can assert this.

The update works as follows:

1. an actor starts a new transaction
2. reads the last available dataset
3. checks the timestamp: if it’s already the stamp of the new dataset go to 5:
4. write the new dataset into the FAS
5. end the transaction

A3.7.9 OSN Configuration and operation

The OSN itself has to be monitored and administered, especially if there are dedicated administrated catalogues (network nodes) for special purposes (having high security, trust and quality requirements). In this case it might make sense to monitor (parts of) the OSN using special ORCHESTRA services to control the network or the operation of services plugged into the OSN by data or service providers.

In order to be able to fulfill this task, all of the services within the OSN have to provide at least some meta-information, e.g. a self description[3] and statistical parameters. A self description of a service may include the following: ontology, feature types as well as the service's description for invocation.

Relevance: OSN configuration and operation is of primary interest to OSN administrators, and normally invisible for other user types. The importance of these two categories originates from the fact that no other service works properly in a badly configured and maintained network.

Requirements:

Each service available in the OSN shall have a self description

Each service shall provide statistical information (load, availability, etc.)

Affected services: Service Monitoring Service.

[3] Note: from the point of view of the service these are some of its attributes

A3.8 Description of meta-Information for services

This section describes the required meta-information for services when considered as objects of concern (following the definition of meta-information in section A3.3).

A3.8.1 Introduction

In section A3.7, a number of “particular purposes”, for which meta-information is to be defined, was given as a starting point. The tables in section A3.7 illustrate these purposes by giving examples of meta-information for selected objects of concern, and they point out which services are concerned with the handling of the meta-information.

In this section, services themselves are considered as objects of concern. More concretely, this section points out how rules for definition of meta-information for services can be developed. According to the definition in section A3.3, the application of the rules then leads, for each purpose, to a conceptual meta-information model with services being the objects of concern. In the sense of this definition, no meta-information that is specific to a particular service is considered here, but only meta-information that applies to all services. The rules (and the resulting model) could be reused for other objects of concern; however, services require very specific modelling techniques for parts of the meta-information that is only meaningful in the context of services. This observation results from the nature of services as active elements, and from the fact that languages for the description of behaviour in terms of IOPE (input, output, pre-conditions and effects) will be part of the resulting conceptual model.

In addition to the requirement for specific modelling techniques, the main motivation for selecting services here as an “object of concern” example results from the high importance of services in a Service Oriented Architecture (SOA), as reflected in the service viewpoint of the ORCHESTRA architecture.

A number of requirements for defining meta-information have been listed in section A3.6, which have been taken into account for this elaboration for services. The most important ones have been used as guidelines. The major steps have been the following:

- look at requirements resulting from the service architecture applied in ORCHESTRA,
- look at meta-information standards for services, what they can provide,
- refine and elaborate the defined purposes in the context of services (e.g. tasks to be performed).

Some of the purpose-specific tasks can directly be performed by a human user by utilising available meta-information. However, in many cases automation of these tasks is required, i.e. they are performed automatically by an application or a software agent.

The refinement and elaboration of the purposes has led to requirements for services as “objects of concern”. These requirements are listed in each of the subsequent “purpose” sections. Moreover, these subsections provide guidelines for the development of rules for the definition of meta-information for (all) services. These guidelines should be considered when developing the concrete rules for the construction of ORCHESTRA application schemas (OAS-MI) for each purpose (note that this is subject of the RM-OA Annex B1).

The meta-information in a concrete OAS-MI can refer to the concepts defined in a domain ontology in order to explicate semantics of the meta-information. In some cases, the meta-information can be generated (e.g. extracted from various sources) in accordance with the conceptual structure of a domain ontology, which would consider the ontology to be meta-information itself.

A3.8.1 Architectural impacts

The ORCHESTRA architecture is defined as a Service Oriented Architecture (SOA). This raises some basic requirements for service meta-information. In SOAs, services are managed according to certain roles:

- service providers develop services which they want to offer others for use,
- service requestors search for services with a given functionality and include them in their applications,
- service brokers publish services so that requestors can find and use them.

From these roles, it can directly be derived that meta-information is needed

- at any site covering one of these roles,
- for the purpose of publishing, searching for and invoking services.

In order to cover these roles with high quality, meta-information for services should not only comprise technical descriptions on a syntactical level, but semantic descriptions as well.

A3.8.2 Impact of standards

While meta-information for services on a syntactical level is available in standards like W3C Web Services or CORBA, semantic meta-information is a relatively new area of research. Relevant approaches for meta-information can be found in the following standards:

- ISO 19119 (provides a UML based structure of certain meta-information which can be of value for ORCHESTRA)
- OWL-S (provides an ontology for semantic mark-up of Web Services, i.e. it focuses on automation based on semantic descriptions)
- WSMO (a complete conceptual model for Semantic Web Services based on ontologies)

The requirements for service meta-information formulated in this document are decoupled from these standards, i.e. they are formulated independent of standards on a conceptual level. However, these standards are used as guidelines for the following descriptions, certain requirements which have been identified could directly be derived. Concrete specifications and implementations based on this document can be kept compliant to these standards.

A3.8.3 Purpose: Discovery

A3.8.3.1 Elaboration of the purpose

User driven discovery:

Current standards specifying meta-information for services focus on the description of syntactical frames for building service registries. Human users can find services by manually looking up these registries ("user driven discovery").

Example: The W3C specifications for Web Services, namely UDDI, WSDL and SOAP, define standards for service discovery, description and messaging protocols respectively. By means of WSDL a technical interface description of a service can be described and published such that a requestor is able to locate the service and get its description. On the basis of the description, program code for service invocation can then be generated. As the WSDL descriptions in an UDDI/WSDL service do not comprise any semantic descriptions, discovery of services offering a desired functionality has to be done by a human user.

Automated discovery (Semantic Services)

In order to discover services needed for achieving a defined goal, applications (or agents) need semantic descriptions of these services. By means of semantic descriptions, the capabilities of a service can be declaratively expressed in a formal language. Ontology languages can be used for this purpose. The application performs a semantic match between the description of services being sought and services being offered.

A3.8.3.2 Requirements derived from elaboration

Meta-information is required for the purpose of user driven and automated discovery of services.

A3.8.3.3 Guidelines for development of rules

For user driven discovery, ISO 19119 is the most advanced standard from which to start. Ideas from UDDI/WSDL, which are not reflected in ISO 19119, should be incorporated and formulated in a platform independent way. The rules should define a selection of ORCHESTRA specific profiles from these sources.

For automated discovery, OWL-S “profiles” and WSMO’s “capabilities” specified in the WSML ontology language provide generic schemas for modelling the meta-information.

A3.8.4 Purpose: Invocation

A3.8.4.1 Elaboration of the purpose

Basic Service Invocation:

Meta-information for service invocation comprises a syntactical description of the implemented operations of a service. The meta-information covers all aspects needed for service invocation such that it can be used directly for that purpose, including the URLs where operation requests have to be directed.

Automated invocation:

Once an application (or agent) has discovered a service based on its semantic description (section A3.8.3), it must be able to automatically invoke the service without any further manual intervention. For that purpose, it generates an invocation message in the requested format. The format itself is determined by the message exchange protocol used in the respective service infrastructure (e.g. SOAP in case of W3C Web Services). Information used for the purpose of constructing messages in the requested format is called “grounding” information.

A service grounding can be thought of as a mapping from an abstract to a concrete specification of those service descriptions elements that are required for interacting with the service (e.g. message format, serialisation, transport and addressing).

A3.8.4.2 Requirements derived from elaboration

Meta-information is required for the purposes of basic and automated invocation of services.

Meta-information for automated invocation comprises information for service grounding, i.e. the details of how to access services. Following the architectural approach of ORCHESTRA, it must be possible to ground services on any SOA technology, in a concrete specification and/or implementation grounding is to be based on a concrete technology.

A3.8.4.3 Guidelines for development of rules

Meta-information for basic invocation should take into account ISO 19119 (Open GIS Service Architecture) and UDDI.

Meta-information for grounding can be based on OWL-S or WSMO grounding mechanisms, which provide language constructs that map the constructs of the process model (IOPE) onto the detailed specifications of an implementation platform (e.g. WSDL, XML). The rules to be defined will not specify such concrete groundings, but rather restrictions on the mechanisms to be used.

A3.8.5 Purpose: Integration (Collaboration)

In the context of services, integration and collaboration can be achieved by various scenarios. In the following subsections, service composition, service interoperability and service mediation are elaborated.

A3.8.5.1 Service composition

8.5.1.1 Elaboration of the Purpose

Services can be composed in chains for sequential execution such that they build a new service. Composition approaches can have an effect on the complete life cycle of the service to be composed, e.g. on the design process, the discovery of participating services at run time, and finally on execution and monitoring of the new service.

A composition is based on a choreography, which defines the rules to communicate with each service participating in the composition in order to consume its functionality. A choreography defines the set of allowed sequences of messages between the participating services, i.e. it is based on knowledge about dynamic constraints of the service operations. A choreography can be made explicit and public as a declarative choreography specified in an appropriate language, or it may be implemented as internal knowledge of an agent that executes the composed service.

While a choreography describes allowed compositions of a set of services from a neutral point of view, an orchestration describes a concrete composition from the viewpoint of one of the participating services. An orchestration, for instance, can describe how the new service makes use of the participating services in order to achieve its capability. An orchestration can (but need not) be conformant to a choreography.

Compositions of services can be distinguished by the time at which the composition is determined.

Proactive composition:

A proactive composition is determined in the design phase of the overall application, i.e. a workflow description that determines the execution of the service chain is established in the design process. Once the composition has been determined it remains fixed, i.e. the new service is composed of the identical set of services during the whole run time.

Reactive composition:

A reactive composition is built dynamically at the time the new service is requested.

Choreographies in most cases are used to provide for a proactive composition of services (at system build time) in order to achieve service interoperation on a regular basis (long running interactions driven by an explicit process model).

8.5.1.2 Requirements derived from Elaboration

Meta-information is required for the purpose of reactive composition of services. Reactive composition has the advantage that services for building a chain can be selected dynamically on the basis of actual values of quality parameters such as price, performance and/or availability.

Meta-information is also required for the purpose of proactive composition of services which communicate on a regular basis. If, for instance, a service collecting environmental data on a European level is based on respective services acting on a national or regional level, the chain of services can be expected to remain fixed at run time.

8.5.1.3 Guidelines for Development of Rules

Meta-information for reactive and proactive composition can be based on OWL-S “process” descriptions or WSMO “capabilities” and “interfaces”.

A3.8.5.2 Service interoperability

8.5.2.1 Elaboration of the purpose

Automation of application integration across organisational boundaries requires interoperation between services. However, internal details of the organisation process should not be made publicly visible. Nevertheless, the external message exchange must be made public. The private process model should drive the behaviour, but it should be grounded on a public model visible to other services.

Internal business processes are usually modelled in an appropriate choreography language like BPELWS or BPML/WSCI, which can describe the decision mechanisms for the execution of a service, but these languages lack semantics to expose the public interface.

8.5.2.2 Requirements derived from elaboration

Meta-information is required in order to publish the external behaviour of services such that no information about internal business processes is exposed.

Note: To specify and implement generic service interoperability is a very complex task which cannot be completely covered within the ORCHESTRA project.

8.5.2.3 Guidelines for development of rules

Meta-information for the external behaviour of services based on explicit choreographies can be described by WSMO “capabilities” and “interfaces”.

A3.8.5.3 Service mediation and mapping

8.5.3.1 Elaboration of the purpose

If an operation request of a service has to be mapped to an operation request of another service (e.g. in a discovery, invocation or orchestration scenario), a mapping description for the services and its associated meta-information must be provided.

If, for instance, meta-information of services is described by means of (different) service ontologies, the mapping can be described as an “ontology to ontology mediator” as defined in WSMO.

8.5.3.2 Requirements derived from elaboration

Meta-information is required for the purpose of service mediation.

8.5.3.3 Guidelines for development of rules

Meta-information for service mediation can be described by means of WSMO mapping descriptions. Select mediator types (e.g. ontology to ontology mediators) from the WSMO specification which are useful for ORCHESTRA are yet to be investigated.

A3.8.6 Purpose: Interpretation

A3.8.6.1 Elaboration of the purpose

Purpose interpretation focuses on meta-information that helps the user to understand the meaning of data and services.

Meta-information on services can express what a service does. This is usually expressed informally, i.e. by means of text. In a more advanced scenario, the text may be annotated with links to a domain ontology; this would enable the user to interpret the semantics of a service through the domain ontology.

Another scenario could be to utilise semantic descriptions of a service given by a service ontology (profile, service model etc.) for generation of text that helps with interpretation of the service’s functionality. The semantic descriptions are designed such that they can be utilised by any application.

A3.8.6.2 Requirements derived from elaboration

Meta-information that helps the user to understand what a process does is required.

A3.8.6.3 Guidelines for development of rules

Most of the standards provide various fields that can be filled with information that help with interpretation. For instance, OWL-S profiles provides a field “textDescription”, which offers a facility to provide a brief description of the service. It summarises what the service offers, it describes what the service requires to work, and it indicates any additional information that the compiler of the profile wants to share with the receivers. The field “ServiceCategory” refers to an entry in some ontology or taxonomy of services.

A3.8.7 Purpose: User Profiling

A3.8.7.1 Elaboration of the purpose

In order to provide tailored views to users, services – as any other objects of concern – must exhibit a profile describing these views. The profile might contain meta-information that is only meaningful in the context of services, such as service costs, workload generated by a service, or execution domain (e.g. geographical boundaries). The service profiles can also specify restrictions on the provision of operations and/or parameters to users.

A3.8.7.2 Requirements derived from elaboration

Meta-information is required for the purpose of profiling. The concrete requirements for user profiling will become obvious in the context of building a concrete OSN.

A3.8.7.3 Guidelines for development of rules

ISO 19119 does not provide a generic approach to service profiles. Meta-information of services is structured in predefined fields. Some of these could be used in a concrete profile.

OWL-S and WSMO provide language constructs to describe profiles of services. In OWL-S for instance, a ServiceCategory describes categories of services on the bases of some classification (an ontology, a taxonomy etc.) that may be outside OWL-S and possibly outside OWL. This is a generic approach by which any profiling task can be specified.

A3.8.8 Purpose: Authentication, Authorisation and Accounting (“AAA”)

A3.8.8.1 Elaboration of the purpose

Note: *The ORCHESTRA model for AAA is not yet complete; the following elaboration is based on the assumption that the model is flexible enough such that the properties of a particular service w.r.t. to AAA can be described in a proper policy notation language.*

A seamless integration of services (see section A3.8.5) does not only require knowledge about the service interface, but also additional information such as requirements, capabilities and preferences of a service. This concept is generally thought of as a “service policy”. The policy can express the service’s needs w.r.t. AAA issues. Examples of such expressions could be

- Is service security supported or required?
- Which kind of authentication is required in order to get the grant to access service operations?

In order to evaluate policies automatically at run-time, a proper run-time environment for services is needed.

In order to enable service clients to get the policy of a service, the policy is to be kept as meta-information in a proper store (e.g. the service catalogue). The client and the service may then establish a trust relationship between each other (trust management). Therefore the client needs to acquire a security token for the service which can be requested from a third authority that is trusted.

An important feature of the policy language (and the overall AAA model) is that fine-grain access control to services can be applied. This makes it possible to explicitly include or exclude specific service operations from user access. For instance, it is important that a Map Access Service might grant access to certain map layers to users. Fine-grain access controls provide a greater level of security, allowing individual organisations to control how authenticated users can use resources.

A3.8.8.2 Requirements derived from elaboration

The requirements for meta-information are not exactly clear at the moment, as the model for AAA is under development. Since provision of controlled and highly secure access to geospatial services can of be high importance in OSNs, a flexible model based on individual descriptions of services' needs is required.

In addition to the service policy, a service should generate meta-information on service usage (accounting) and provide the necessary information for service billing.

A3.8.8.3 Guidelines for development of rules

The specification for Web Services defines a grammar for expressing the capabilities, requirements and general characteristics of entities in an XML Web Services based system in the "Web Services Policy Framework" (WS-Policy).

Policies can be expressed in a platform-independent way by means of ontologies. The interoperability and mapping facilities of ontologies can be used to mediate between different policies in heterogeneous domains (e.g. as shown in the ARTEMIS project in the medical sector). In a concrete application, the ontological policies have to be grounded onto policies expressed in the platform-dependent language of the applied AAA-Framework, e.g. the Web Service Policy Framework.

A3.8.9 Purpose: Quality Control/Management

A3.8.9.1 Elaboration of the purpose

In an SOA implementation, it is usual that different providers offer equivalent or similar services. In such cases, a user needs criteria to select the most suitable service on the basis of his requirements for Quality of Service (QoS). Very often the execution of a service itself is preceded by a negotiation and agreement process (Service Level Agreements or SLA). Any non-functional property such as price, payment method, security, trust, and most notably QoS can be the basis of such a negotiation.

Quality can be described relative to a standard, an industrial benchmark or a ranking schema. The meta-information describing the quality of service could then refer to these schemas, asserting that conformance testing has been done or a certification was obtained documenting QoS attributes of the service.

A3.8.9.2 Requirements derived from elaboration

Meta-information is required for the purpose of quality control and quality management. The concrete requirements for quality management will become obvious in the context of building a concrete OSN.

A3.8.9.3 Guidelines for development of rules

In OWL-S, any non-functional property can be included by using the ServiceParameter from ServiceProfile. These non-functional properties are described in the ServiceProfile part and explicitly formalised using OWL.

WSMO recommends a set of non-functional properties for each particular element of a web service description, e.g. accuracy, coverage, financial, network-related QoS, performance, reliability, robustness, security, transactional, trust and others.

Various languages for describing SLA have been developed from vendors (IBM, HP) and universities. Integration of SLA aspects into Web Services (UDDI) is currently being investigated in certain projects.

A3.8.10 Purpose: Transactions, Synchronisation and Locking

A3.8.10.1 Elaboration of the purpose

Transactions, synchronisation and locking (here abbreviated to "TSL") do not stand for a purpose itself, but can be seen as a sub-purpose of other purposes. This can be illustrated by following examples:

- In an orchestration/service chaining scenario, a requirement may be that only services providing a defined transaction level may be incorporated into a chain.
- In section A3.8.9, meta-information about TSL is considered as a Quality of Service (QoS) property.

While these examples make clear that descriptions of these mechanisms are needed as meta-information for various purposes, the coordination effort for achieving TSL is usually hidden or made transparent to the user:

- In the specifications for Web Services, the layers WS-Transaction and WS-Coordination enable the use of standard protocols for transactions, workflows and other applications requiring some kind of (transparent) coordination. In the current specification, properties or configuration parameters of these protocols are not made visible in UDDI or WSDL.
- In RM-ODP, which forms the basis for the RM-OA, transaction is defined as a distribution transparency in the Engineering Viewpoint.

In applications with dedicated dependability requirements, the TSL mechanisms may require very specific meta-information such as synchronisation points, checkpoint information for service backup and recovery at defined synchronisation points, etc.

A3.8.10.2 Requirements derived from elaboration

A description of properties of transaction, synchronisation and locking mechanisms available for services should be provided as meta-information (non-functional properties). It is needed for various purposes (e.g. discovery, integration, quality management or monitoring).

A3.8.10.3 Guidelines for development of rules

See section A3.8.9.3 guidelines about non-functional properties in OWL-S and WSMO covering this subject.

A3.8.11 OSN management

A3.8.11.1 Elaboration of the purpose

OSN management requires non-functional requirements on services such as

OSN configuration management:

Meta-information for configuration management comprises descriptions of the topology of services of the entire OSN, e.g. which services are available at which sites. Such topologies can, for instance, be described by means of ontologies. On the user level, services for change management (e.g. add/remove services) and exploration of the OSN topology should be provided.

OSN monitoring:

Meta-information for OSN monitoring comprises

- Information on the actual load
- Execution traces of services, which are important especially to document and trace execution of services which have been composed reactively (see section A3.8.5).

A3.8.11.2 Requirements derived from elaboration

Meta-information is required for the purpose of configuration management and system monitoring.

The need for further non-functional requirements in the context of OSN management (e.g. financial, security issues) is to be investigated when building a concrete OSN.

A3.8.11.3 Guidelines for development of rules

Requirements for non-functional properties can be collected from ISO, OWL-S, WSMO and OASIS, e.g.

- SAML Markup Language for Security (OASIS standard)
- XACML extendable access control markup language (OASIS)
- OWL-S Markup Language

The rules to be developed therefore depend on requirements of the entire OSN.

A3.8.12Purpose-free meta-information

As outlined in the previous sections, meta-information for services is dedicated to a specific purpose. In addition, certain meta-information can be identified which is required for any of the listed purposes (e.g. the service name). Therefore it makes sense to define this intersection set of meta-information and store the respective values for each service by default in its self-description.

Note: By means of this mechanism, a catalogue of services could be filled automatically with a minimum set of meta-information.

FP6-511678

ORCHESTRA

Open Architecture and Spatial Data Infrastructure for Risk Management

Integrated Project

Priority 2.3.2.9 Improving Risk Management

Reference Model for the ORCHESTRA Architecture (RM-OA Version 2)

Annex B1

Rules for ORCHESTRA Application Schemas for Meta-Information

Date: 2007-01-31

Revision: 2.0

Start date of the ORCHESTRA project:	2004-09-01
Duration of the ORCHESTRA project:	3 years
Organisation name of lead contractor for this deliverable:	Austrian Research Centers GmbH - ARC

Document Control Page

Title	Reference Model for the ORCHESTRA Architecture (RM-OA) D3.2.3: RM-OA Version 2 Annex B1 (Rev. 2.0) Conceptual Meta-Information Model Rules for ORCHESTRA Application Schemas for Meta-Information (OAS-MI)
Creator	Austrian Research Centers GmbH - ARC
Subject	Specification of Rules for OAS-MIs and Examples for OAS-MIs
Description	This document represents the Annex B1 of the Reference Model for the ORCHESTRA Architecture (RM-OA). It describes rules and examples of ORCHESTRA Application Schemas for Meta-Information (OAS-MIs).
Publisher	ORCHESTRA consortium
Contributor	EIG, Fraunhofer IITB
Date	2007-01-31.
Type	Text
Format	application/msword
Identifier	ORCHESTRA Portal: SP3 / SP3 Quality Assurance / 09: D3.2.3 / 06: D3.2.3 RM-OA V2 (2.0) – published version
Source	Not applicable
Language	en-GB
Relation	none
Coverage	Not applicable
Rights	© 2007 ORCHESTRA Consortium The ORCHESTRA project is an Integrated Project (FP6-511678) funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).
Deliverable number	D3.3.2
Audience	☒ public ☐ restricted ☐ internal

Major Revision History

Revision	Date	Sections changed	Description
0.1	2006-03-21	all	Initial (rules-section)
0.4.	2006-06-29	all	Rules for OAS-MIs
0.7	2006-07-03	all	Updated Rules (common, for OAS-Mis, for Interoperability)
0.8	2006-07-04	all	Update common GetCapabilities (Common Part)
1.0	2006-07-13	all	Sent to SP3 leader for review
1.0	Nov. 2006		Deliverable by 2nd annual technical review accepted
1.01	2007-01-16	all	ATR2 comments (of Yves Coene) integrated. Respectively: Insert of section 1.4 Keyword used to indicate requirement levels. Insert of section 2.1 Intended audience
2.0	2007-01-31	all	Editorial changes for publication

Table of Contents

Tables

Figures

B1.1 Management Summary

This document includes the following topics:

- definitions of basic data types
- rules for building ORCHESTRA Application Schemas for meta-information (OAS-MI)
- methodological approach for identification of meta-information
- a recommended example OAS-MI including service capabilities as a recommended presentation of meta-information shown on the use case of pilot scenario

This document will be updated and harmonized in two further cycles incorporating feedback we get from implementation examples (out of the pilots).

B1.1.1 Important note

In the following, all shown examples have to be understood as recommended examples. We use the term recommended examples because the universal meta-information system does not exist and consequently there can be no universal OAS-MI (see also RM-OA Annex A3).

B1.1.2 Keywords used to indicate requirement levels

This document follows the ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards (Fifth edition 2004) w.r.t. the usage of the word “shall”, “shall not”, “should”, “should not”, “may” and “need not”**.** The word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed.

B1.1.3 Abbreviations

AAA	Authentication, Authorisation and Accounting
ACC	Access and Invocation
DIS	Discovery
DRM	Digital Rights Management
COL	Integration and Collaboration
DoW	Description of Work
D331	Deliverable 3.3.1
HCI	Human-Computer Interaction
INT	Interpretation
ISO	International Standardisation Organisation
IOPE	Input, output, post conditions and effects
MI	Meta-information… (used in pre/postfixes)
OA	ORCHESTRA Architecture… (as a prefix)
OT	ORCHESTRA Thematic… (as a prefix)
OASIS	Organization for the Advancement of Structured Information Standards
OAS	ORCHESTRA Application Schema
OAS-MI	ORCHESTRA Application Schema for Meta-Information
OMM	ORCHESTRA Meta Model
OSM:	OSN Management
OSN	ORCHESTRA Service Network
OWL-S	Web Ontology Language for Services
QC	Quality Control
RDF	Resource Description Framework
RM-OA	Reference Model for the ORCHESTRA Architecture
SDS	Simulation Data Service
SMS	Simulation Management Service
TSL	Transaction, Synch and Locking and
UP	User Profiling
W3C	World Wide Web Consortium
WSMO	Web Services Modelling Ontology
WSML	Web Service Modelling Language
WSDL	Webs Service Description Language
XML	eXtensible Markup Language

B1.2 Background and scope

The ORCHESTRA architectural group identified that it is essential to the project that a set of basic data types has to be specified in order to achieve harmonisation and interoperability. The task was appointed to the meta-information working group. That is why you will read about basic data types in this deliverable too. The whole section including the related basic data type rules is intended to be transferred into the second version of the RM-OA.

This document in particular concentrates on rules needed to set up and implement ORCHESTRA Application Schemas for Meta-Information. In principle the rules are split in common rules and purpose related rules. The common rules shall be seen as overall rules relevant for all OAS-MIs especially for meta-information in an OSN. The purpose related rules shall be seen as high level rules relevant for several purposes. Further, a separate section addresses specific rules for OAS-MIs for services.

A methodological approach for identification of meta-information is shown in section B1.6. Selected use cases (with the focus that they are relevant to end-users of an OSN) are shown there and shall be understood as a recommended (examples) approach for analysing and modelling of meta-information in relationship to several selected purposes already defined in the RM-OA Annex A3.

Examples of ORCHESTRA Application Schemas for Meta-Information (OAS-MI) are described in section B1.8. In these examples, the rules defined in section B1.5 are applied in addition to the rules defined for the OMM. When creating examples, the context of an application shall be assumed and described by means of an OAS. The OAS reflects the “normal functionality” of an application, while the OAS-MI describes meta-information needed for a certain purpose. A separate sub-section (OAS-MI for Service Capabilities) defines an OAS-MI which may be used to structure the capabilities of any ORCHESTRA Service.

Finally, this document presents an OAS-MI shown/applied on a real world example based on one of the pilot requirements of ORCHESTRA.

B1.2.1 Intended audience

This document, especially the sections about rules, is used for design as well as extension to the RM-OA. Thus the audience is preliminary expected to be service providers. They shall understand these rules in order to set-up and use ORCHESTRA services to their full extent.

B1.3 References

B1.3.1.1 Normative references

ISO/IEC TR 14252:1996. Information technology - Guide to the POSIX Open System Environment

ISO/IEC 10746-1:1998 (E). Information technology - Open Distributed Processing - Reference model

ISO/IEC 10746-2:1996 (E). Information technology - Open Distributed Processing – Foundations

ISO 19101:2004(E). Geographic information - Reference model

ISO/PRF TS 19103. Geographic information - Conceptual schema language

ISO 19107:2004(E). Geographic information - Spatial schema

ISO 19108:2004(E) Geographic information - Temporal schema

ISO/FDIS 19109:2003. Text for FDIS 19109 Geographic information – Rules for application schema, as sent to the ISO Central Secretariat for issuing as Final Draft International Standard

ISO 19111:2003(E). Geographic information - Spatial referencing by coordinates

ISO 19119:2005. Geographic information - Services (see also “The OpenGIS Abstract Specification - Topic 12: OpenGIS Service Architecture” under http://www.opengis.org/docs/02-112.pdf)

ISO 19118:2005 (E), Geographic information – Encoding

ISO 19119:2005(E). Geographic information – Services.

ISO 19123:2005(E). Geographic information -- Schema for coverage geometry and functions

ISO 19125-1:2004(E). Geographic information -- Simple feature access -- Part 1: Common architecture

ISO/CD TS 19136. Text for final ISO/CD 19136 Geographic information - Geography Markup Language, 2005-05-30, http://www.isotc211.org/protdoc/211n1834/

ISO/CD TS 19139 . Geographic Information - Metadata - XML schema implementation

ISO/TC 211 19115:2004(E). Geographic Information – Metadata

ISO 8601:2000 Date/Time Representations (The committee in charge of ISO 8601 is ISO TC 154)

ISO 4217: 2001 Codes for the representation of currencies and funds

ISO 9241-x: series of standards for Ergonomics of human-system interaction

ISO 8859-x: series of standards for Information technology -- 8-bit single-byte coded graphic character sets

B1.3.1.2 Documents and Books

Agirre-2000: Eneko Agirre, Olatz Ansa, Eduard Hovy and David Martinez: Enriching very large Ontologies using the WWW; 14th European Conference on Artificial Intelligence ECAI'00, Berlin, 2000

Alfonseca: Enrique Alfonseca and Pilar Rodriguez: "Modelling Users' Interests and Needs for an Adaptive Online Information System", 2003

CSS: http://www.w3.org/Style/CSS/

RM-OA Annex A3: ORCHESTRA Deliverable D3.3.1 – Conceptual Meta-Informatio Model; Revision 1.42, 2007

Keller, Uwe et. al: D5.1v0.1 WSMO Web Service Discovery. WSML Working Draft 12 11 2004.

http://www.wsmo.org/2004/d5/d5.1/v0.1/

Knuth: Donald E. Knuth – The Art of Computer Programming, Addison-Wesley, Reading, Massachusetts, 1973, Vol3.

Kopecký, Jacek et. al: D24.2v0.1. WSMO Grounding. WSMO Working Draft 16 September 2005. http://www.wsmo.org/TR/d24/d24.2/v0.1/

OASIS: Organization for the Advancement of Structured Information Standards http://www.oasis-open.org/home/index.php

OGC-Catalogue-Services: „The OpenGIS™ Abstract Specification – Topic 13: Catalogue Services (Version 4)", 99-113.doc, 1999

OGC 05-008: OGC Web Service Specification, Version 1.0; 2005-05-10; Ed: Arliss Whiteside

OpenGIS Reference Manual (2003); Open Geospatial Consortium Inc.; date: 2003-09-16; Reference Number: OGC 3-040; Version 0.1.3; page 2

OWL-QL : Fikes, R.; Hayes, P.; Horrocks, I.: OWL-QL – A Language for Deductive Query Answering on the Semantic Web. Knowledge Systems Laboratory, Stanford University, Stanford, CA, 2003

OWL-S: Web Ontology Language for Services http://www.w3.org/Submission/2004/07/

Sebastiani: Fabrizio Sebastiani - Machine Learning in Automated Text Categorization, ACM Computing Surveys, Vol. 34, No. 1, March 2002

W3C Introduction to RDF Metadata (1997); Author: Ora Lassila, ora.lassila@research.nokia.com, Nokia Research Center; http://www.w3.org/TR/NOTE-rdf-simple-intro-971113.html

Van Rijsbergen: C.J. van Rijsbergen – Information Retrieval, 1979

W3C-CSS-Media: http://www.w3.org/TR/2006/WD-CSS21-20060411/media.html

Wikipedia "Performance measures":
http://en.wikipedia.org/wiki/Information_retrieval#Performance_measures

WSMO: Web Services Modelling Ontology
http://www.wsmo.org/

B1.4 Basic Data Types

B1.4.1 Introduction

The following section defines the most fundamental data types available in the ORCHESTRA framework. In order to achieve interoperability a common basis shall be made available and well defined. ORCHESTRA Basic Data Types (and OA Types) are part of such a basis.

All data types used and defined in ORCHESTRA shall be built directly and/or indirectly (e.g. OA-Types) using Basic Data Types. This enables ORCHESTRA users having only one definition for a single type instead of a multitude of definitions (e.g. every service developer and/or every application designer defining its own types for equal purposes).

Additionally ORCHESTRA basic data types relate and refer to definitions in already accepted standards (like ISO 191xx series) and therefore are well known among IT specialists, even if they aren't (yet) involved in ORCHESTRA itself.

B1.4.2 Basic Data Types

Basic Data Types have a standardized definition outside of ORCHESTRA documents (e.g. ISO 191xx series). The names of these types will not be prefixed and refer to standard types. The related standard document can be found in the *Origin* column of Table 1.

Type Names	Origin	Brief Description
Any	ISO19103	The root of all classes. Often not an actual class in the implementation, it essentially is used where the target class of a member name is not known.
Binary	ISO19118 section A.5.2.1.14	Finite-sequence of arbitrary binary data.
Real	ISO19103 section 6.5.2.5	A signed real (floating point) number consisting of a mantissa and an exponent. (not necessarily the exact value as the common implementation of a Real type uses base 2)
Decimal	ISO19103 section 6.5.2.4	A number type that represents an exact value as a finite representation of a decimal number. (Unlike real, it can represent 1/10 without error)
Integer	ISO19103 section 6.5.2.3	A signed integer number. Exact with no fractional part.

Table 1: Basic Data Types

Type Names	Origin	Brief Description
CharacterString	ISO19103 section 6.5.2.7	Type representing a simple string. The whole string has a single specific encoding. This encoding is retrievable from the string.
CountryCode	As will be defined in ISO 19139	List of country identifiers.
LanguageCode	As will be defined in ISO 19139	List of language identifiers.
MD_CharacterSetCode	As defined in ISO 19115	List of character encodings.
PT_Locale	As will be defined in ISO 19139	Type combining language, country and encoding.
LocalisedCharacterString	As will be defined in ISO 19139	A CharacterString with the addition of a field specifying the language of the string.
Enumeration	ISO19103 section 6.5.4.2	Defined and closed list of valid mnemonic identifiers.
CodeList	ISO19103 section 6.5.4.3	An open Enumeration.
Boolean	ISO19103 section 6.5.2.11	A value specifying TRUE or FALSE
Date	ISO19103 section 6.5.2.8	Type representing a date.
Time	ISO19103 section 6.5.2.9	Type representing a point in time.
DateTime	ISO19103 section 6.5.2.10	Type combining date and time.
Set	ISO19103 section 6.5.3.2	Unordered finite collection of non duplicate objects.
Bag	ISO19103 section 6.5.3.3	Unordered finite collection of possibly duplicate objects.
Sequence	ISO19103 section 6.5.3.4	Ordered 'bag-like' structure.
Dictionary	ISO19103 section 6.5.3.5	Container for key-value pairs where the key and value types are not predefined.

Table 2: Basic Data Types (cont.)

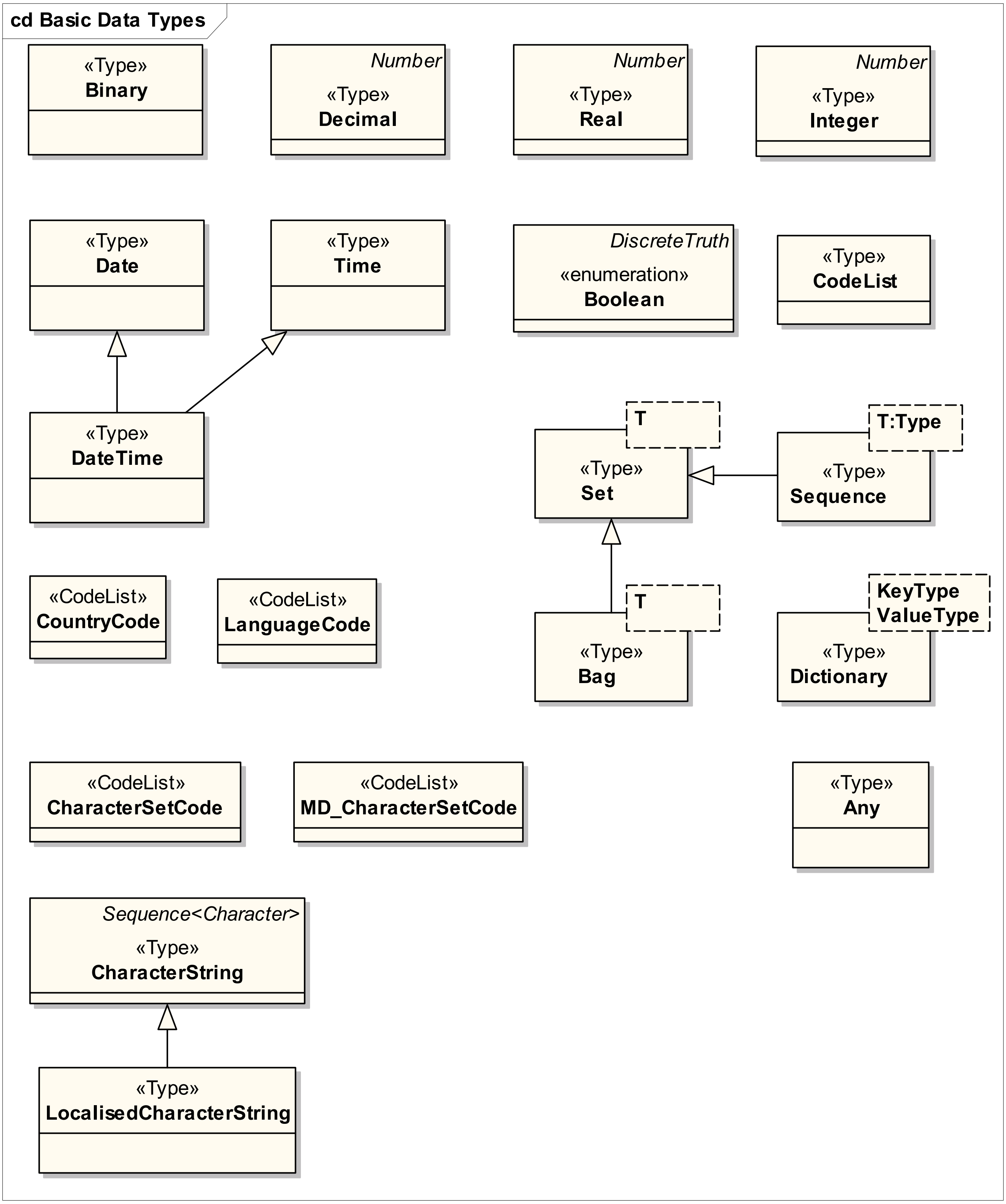


Figure 1: Basic Data Types

B1.4.3 OA_Types

OA_Types are predefined types in the OMM which do not have a standardized definition outside of ORCHESTRA documents. They are composed of ORCHESTRA Basic Data Types and other already defined OA_Types. OA_Types might still be rather simple.

B1.4.4 User-defined types

User-defined types are not predefined within the OMM. They usually refer to types defined for a specific application (e.g.: in an OAS) and may only consist of well known types. These well known types are, Basic Data Types, OA_Types and already specified User-defined types.

B1.4.5 Rules for type definitions

The following rules shall be used when defining new data types:

- Basic Data Types and OA_Types shall be used where applicable.
- Types defined in OA Services shall be prefixed by OA_ (e.g. OA_GetCapabilitiesRequest).
- Types defined in OT Services shall be prefixed by OT_.
- Prefixes for the specification of user-defined types (e.g. in an OAS) are not enjoined on the OAS designer, however, OA_ and OT_ are excluded.
- Types that are specified in UML shall be stereotyped with «Type»

Note: Section B1.3 might go completely into RM-OA and be deleted from D332 in one of the next versions!

B1.5 Rules for building OAS-MIs

B1.5.1 Introduction

All ORCHESTRA specifications, including those on meta-information are built to leave the biggest possible freedom to OSN implementers, including the freedom to implement their own services. As a consequence, all OAS-MIs presented in this document and elsewhere in ORCHESTRA specifications should be seen as examples to illustrate how OAS-MIs can be built. Future OSN developers are free to combine these example OAS-MIs in any way they see fit, or even completely ignore them and develop their own OAS-MIs with no relation to any of the examples developed by the ORCHESTRA consortium.

In order to assure basic interoperability of data and services in an OSN, and especially to promote the interoperability between OSN developed separately, *all OAS-MIs*, including those presented in this document and those that may be developed independently *SHALL follow a set of basic rules presented in this section.*

In addition to the rule description, each rule mentioned in this section is accompanied with a rationale, and with a set of meta-information tags indicating the area of relevance of this particular rule. These meta-information tags are grouped in „Purpose“,and „Object“ columns.

The “Purpose” column indicates the purpose this rule applies to. ORCHESTRA distinguishes between following purposes (see RM-OA Annex A3): Discovery (DIS), “Access and Invocation” (ACC), Integration and Collaboration (COL), Interpretation (INT), User Profiling (UP), “Authentication, Authorisation and Accounting” (AAA), Quality Control (QC), Transaction, Synchronisation and Locking (TSL) and OSN Management (OSM)

The “Object” column shows whether the rule applies to Features (F), Services(S), OSN-wide (N) or a combination of them.

Note: This is a first draft of abstract rules for building OAS-MI. Rules will be improved and expanded with the help of service developer feedback.

Note: These rules aim to achieve a high level of homogenisation within OSN, and promote the interoperability between OSNs through use of standards and semantic equivalence[1] of data and services.

[1] Our understanding of semantic equivalence is based on definition available on Wikipedia (“two data elements from different vocabularies contain data that has similar meaning” http://en.wikipedia.org/wiki/Semantic_equivalence). Semantic equivalence of data and services assures that two OSNs can be merged by mean of gateways.

B1.5.2 Common rules

This section contains general rules for specification of MI, which are applicable to any object of concern. Due to the general-purpose nature of all rules presented in this section, the “Purpose” column should be understood in the context of the rationale. For example, rule 4.2.6 is valid for all purposes, but its rationale is related to quality assurance.

Rule No.	Rule description	Rationale	Purpose	Object
4.2.1	All OAS-MIs SHALL be built according to rules of OMM	Consequence of RM-OA "Rigorous Definition and Use of Concepts and Standards" architectural principle.	all	all
4.2.2	All meta-information SHALL be provided at least in a form suitable for interpretation by humans. Syntactic meta-information SHALL also be provided in a form suitable for interpretation by machines.	Consequence of RM-OA "Self-describing Components" architectural principle.	all	all
4.2.3	Providing semantic meta-information in a form suitable for interpretation by machines (e.g. by means of an ontology) is highly encouraged.	Consequence of RM-OA "Self-describing Components" architectural principle.	all	all
4.2.4	All objects of concern, independent of the aggregation level SHALL be accompanied with adequate meta-information. The required meta-information is defined by an OAS-MI. **Example:** meta-information describing user access policy SHALL be provided on OSN level.	Consequence of RM-OA "Self-describing Components" architectural principle and D3.3.1 requirement 5.5.5.	all	all

Table 3: Common OAS-MI rules

Rule No.	Rule description	Rationale	Purpose	Object
4.2.5	Meta-information models SHALL be specified in a way that is independent from the technology of the underlying implementation platform (e.g. using UML). **Example:** If services in an OSN are described by means of service ontologies, MI for invocation of services in service ontologies shall be specified such that it can be grounded on any SOA implementation platform (e.g. WSDL, XML).	Consequence of RM-OA "Technology-independence" architectural principle.	ACC, COL, INT	all
4.2.6	Whenever appropriate, a reference to publicly available meta-information SHOULD be used rather than providing a local copy. **Example:** Licensing conditions should be published once within OSN, and each piece of data published under a certain license should contain a reference to this license, rather than full license text.	Consequence of RM-OA "Evolutionary Development – Design for Change" architectural principle, and D3.3.1 requirement 5.4.3.	QC	all

Table 4: Common OAS-MI rules (cont.)

B1.5.3 High level purpose related rules

Rules presented in this section apply to one or several purposes. Further purpose related rules can be found in section B1.7.

Rule No.	Rule description	Rationale	Purpose	Object
4.3.1	All data and services SHOULD be accompanied with meta-information usable for data/service discovery (cf. B1.6.1 Discovery). **Example:** summary, keywords, types etc.	Consequence of RM-OA "Loosely Coupled Components" and "Self-describing Components" architectural principles.	DIS	F, S
4.3.2	All features in an OSN SHALL be accompanied with meta-information sufficient to *access* this information without any prior knowledge specific to this OSN. **Example:** All data has to be accompanied with meta-information about schema language used to encode the information (e.g. "OWL", "XML", "RDF",etc.)	Consequence of RM-OA "Self-describing Components" architectural principle.	ACC	F
4.3.3	All objects of concern in an OSN SHALL be accompanied with meta-information sufficient to *interpret* this information without any prior knowledge specific to this OSN. **Example:** environmental measurements (numbers) shall be associated to units, measurement type, accuracy, geographic and temporal data.	Consequence of RM-OA "Self-describing Components" architectural principle.	ACC, COL, INT	F, S, N

Table 5: High level purpose related rules

Rule No.	Rule description	Rationale	Purpose	Object
4.3.4	Meta-information for discovery of data SHALL have an explicitly modelled relation to the feature (data) described by this meta-information. The direction of this relation is from the meta-information to the feature	Consequence of RM-OA "Self-describing Components" architectural principle.	DIS	F
4.3.5	Meta-information for discovery of data SHALL contain a relation to meta-information for data access. This relation SHOULD be either in form of an URI for direct access or at least as a description how to gain access. **Example:** in the worst case, at least a contact information should be provided.	Consequence of RM-OA "Self-describing Components" architectural principle.	DIS, ACC	F
4.3.6	All information SHALL be accompanied with meta-information relevant for “Authentication, Authorisation and Accounting” (AAA), quality control (QC), and digital rights management (DRM). **Example:** person(s)/organisation(s) responsible for the data/service operation and integrity; data/service owner(s); information about data licensing and terms of use; service availability; a guarantee for future availability of the data/service ("unknown", "10 years", ...); information concerning the data quality such as measurement procedure; "raw data" versus "verified according to procedure X"; error estimates.	Consequence of RM-OA "Loosely Coupled Components" and "Self-describing Components" architectural principles.	UP, AAA, QC	F, S, N

Table 6: High level purpose related rules (cont.)

Rule No.	Rule description	Rationale	Purpose	Object
4.3.7	Rule 4.3.6 remains valid for derived data and services. **Example:** service chaining, data fusion	Consequence of RM-OA "Self-describing Components" architectural principle.	QC	S
4.3.8	Meta-information for User Right Management SHALL be provided at all aggregation levels. **Note:** This does not necessarily mean that a mechanism for manipulating this meta-information at all aggregation levels has to be implemented by all services in all OSNs.	Consequence of RM-OA "Generic infrastructure" architectural principle.	ACC, AAA	F

Table 7: High level purpose related rules (cont.)

B1.5.4 Rules for building OAS-MI for services

Rules presented in this section apply to OAS-MIs for services.

Rule No.	Rule description	Rationale	Purpose	Object
4.4.1	OAS-MIs for services SHALL be built in a way that allows merging of several service OAS-MIs into one in a straightforward way (e.g. by simply adding new sections to OAS-MI for service.)	Consequence of RM-OA "Loosely Coupled Components" and "Evolutionary Development – Design for Change principles". Allows greater flexibility in service implementation.	COL	S
4.4.2	OAS-MI for services SHALL be built in a way that allows accessing specific parts of the service capabilities. **Example:** the schema for common capabilities may be structured in sections. GetCapabilities(STRUCTURE) could then return just a list of available sections, and GetCapabilities(SECTION_ID) would return a specific section.	Service capabilities may be of arbitrary size. Therefore, a mechanism which assures that parts of reasonable size and relevant to the problem at hand can be accessed independently. (Scalability and Usability system requirements; "Loosely Coupled Components" and "Evolutionary Development – Design for Change principles")	DIS, ACC, COL	S

Table 8: Service related rules

Rule No.	Rule description	Rationale	Purpose	Object
4.4.3	OAS-MI for services SHALL allow to distinguish between "common" and "service specific" service capabilities. "Common" service capabilities SHALL be implemented by all services in an OSN. **Note:** Common capabilities for services may not be the same in all OSN instances! **Note:** Currently under discussion: As this may cause problems with interoperability! **Example:** GetCapabilities(COMMON) returns only the common part of service capabilities.	OAS-MI homogenisation does not stop at homogenised representation. An effort shall be made to identify a set of capabilities common to all services ("Loosely Coupled Components" system requirement).	COL	S
4.4.4	OAS-MI for invocation of services SHALL contain all information necessary to invoke a service in a form suitable for interpretation by machines. **Example**: parameter and response types; access points; message format	Consequence of RM-OA "Self-describing Components" architectural principle.	ACC, COL	S
4.4.5	All service related meta-information, with exception of service monitoring information (load, uptime, etc.) SHALL be contained in GetCapabilities() operation. **Note:** In some cases, "GetCapabilities()" will return a reference to relevant information sources rather than information itself.	This rule assures that complete service meta-information except the one for service monitoring is available from one source. ("Loosely coupled Components" and "Self-describing Components" principle.)	DIS, ACC, COL	S

Table 9: Service related rules (cont.)

Rule No.	Rule description	Rationale	Purpose	Object
4.4.6	OAS-MI for services SHALL include a service classification according to RM-OA, a service summary, and a reference to a full service documentation. **Note:** this needs to be discussed further on! **Example:** Simulation Management Service (SMS) shall be distinguishable as such. In addition, each service instance shall offer a summary and a reference to the full documentation of the SMS, and of each incorporated simulation model.	Consequence of RM-OA "Self-describing Components" architectural principle.	ACC	S
4.4.7	In addition to a complete syntactic and semantic description of service capabilities in a human readable form (rule 4.2.2), an OAS-MI for a service SHALL at least provide a syntactical description of all service capabilities in a form suitable for the interpretation by machines. Preferably, it should contain both, machine readable syntactical and semantic information, e.g. by means of a service ontology.	Syntactic description is needed to invoke a service. Machine interpretable semantic information is a prerequisite for developing intelligent ORCHESTRA clients and automatic service chaining ("Self-describing Components" architectural principle).	DIS, ACC, COL	S

Table 10: Service related rules (cont.)

B1.5.5 Rules related to interoperability of data and services

Rules presented in this section intend to assure interoperability of data and services within an OSN and between OSNs. Interoperability at OSN level can (and should) be achieved through homogenisation. e.g. through the use of standards. Due to “Technology Independence” and “Evolutionary Development - Design for Change” ORCHESTRA architectural principles, inert-OSN interoperability will have to be achieved by means of gateways and conversion services.

Rule No.	Rule description	Rationale	Purpose	Object
4.5.1	Platform, data formats, interfaces, protocols, messages, AAA, and all other aspects of an OSN SHOULD be as homogenised as possible.. **Example:** OSN developers are highly encouraged to use the same ontology language for all services within "their" OSN.	Homogenisation is a simplest, least expensive, and most robust way of assuring interoperability within OSN. (RM-OA "Usability" system requirement)	ACC, COL, INT	all
4.5.2	Existing open standards for meta-information SHOULD be used whenever possible. **Example:** ISO 19115 Standard SHOULD be used for all the geo-referenced meta-information. **Example:** SI units SHOULD be used for all the measurement values.	The use of standards is the most efficient way of achieving homogenisation and thus interoperability between OSN instances ("Rigorous Definition and Use of Concepts and Standards" architectural principle; D3.3.1. requirement nr. 5.5.3)	ACC, COL, INT	F, S, N

Table 11: Interoperability related rules

Rule No.	Rule description	Rationale	Purpose	Object
4.5.3	If a standard applicable to a problem at hand does exist but can not be used, the meta-information concepts used instead SHOULD be (at least) semantically equivalent to those defined in the standard. **Example:** If an OSN supports keyword based discovery, the way keywords are encoded should be semantically equivalent to ISO 19115 attribute "MD_Keywords". **Example:** If an OAS-MI contains monetary information, e.g. for the purpose of billing, this information should be provided in a way semantically equivalent to one used in ISO 4217. **Example:** Information about institutions or persons for the purpose of establishing contact with the person in charge of an object or service (e.g. owner, author, editor service administrator...) or similar should be semantically equivalent to ISO 19115 attributes CI_ResponsibleParty and/or CI_Contact (and subsidiary classes).	This approach is in line with RM-OA "Rigorous Definition and Use of Concepts and Standards" and "Loosely Coupled Components", architectural principle.		

Table 12: Interoperability related rules (cont.)

Rule No.	Rule description	Rationale	Purpose	Object
4.5.4	If a standard applicable to the problem at hand does exist but can not be used, rules for mapping between meta-information used within an OSN instance and the standard SHOULD be part of the schema documentation. **Example:** If units other than SI units are used, their relation to SI units shall be documented. **Example:** If an OSN instance uses a non-ISO 19115 schema for geo-referenced information, the relation to ISO 19115 schema shall be documented.	This approach is a consequence of the rule 4.3.3 and in line with RM-OA "Rigorous Definition and Use of Concepts and Standards" and "Loosely Coupled Components", architectural principle.	ACC, COL, INT	F, S, N
4.5.5	If no final consensus can be found concerning homogenization within an OSN, a service capable of translating between competing data formats, encodings, query languages, etc. SHOULD exist within the OSN. **Example:** a service capable of translating data between different cartographic projections will be needed in case geo-referenced data encoded in more than one cartographic projection is available within an OSN. **Example:** a service capable of translating the service messages from French to German and vice versa will be needed to assure interoperability in a trans-boundary French-German OSN.	This approach is in line with following ORCHESTRA architectural principles: "Loosely Coupled Components", "Evolutionary Development – Design for Change", and "Generic Infrastructure".	ACC, COL, INT	F, S

Table 13: Interoperability related rules (cont.)

Rule No.	Rule description	Rationale	Purpose	Object
4.5.6	If no final consensus can be found concerning homogenization within an OSN, competing mechanisms and elements SHALL either be semantically equivalent, or semantic subsets of the OSN default data formats, encodings, query languages, etc. **Example:** In case some of the strings used within an OSN are represented in incompatible encodings (e.g. ISO-8859-x national encodings), the default OSN encoding shall be capable of representing characters of all other encodings used within the OSN, such as UTF-8, UTF-16.	Correct translation can only be guaranteed between semantically equivalent systems. Failure to follow the rules 4.5.2 or 4.5.3 or 4.5.6 inevitably leads to deficiencies in interoperability within an OSN. ("Usability" system requirement!)	ACC, COL, INT	F, S

Table 14: Interoperability related rules (cont.)

B1.6 Methodological Approach for Identification of Meta-information

All use cases shown thereafter shall be understood as a recommended (example) approach for analysing and modelling of meta-information in relationship to several selected purposes already defined in the RM-OA Annex A3 "Conceptual Meta-Information Model". Furthermore the use-cases are selected with the focus that they are relevant to end-users of an OSN. Their purpose is to clarify possible uses and information necessary to provide the respective functionality.

A concrete use-case would be an information-consumer who wants to assess soil conditions in a specific region. The data which this information-consumer is interested in shall contain information on polluted areas, and on the nature of the pollution. The user has to find relevant information and to access it. Every time information is presented to the user it has to be rendered.

This "pollution-information" use case is divided into different sub use cases:

- Discovery
  - Navigation
  - Search
- User-Profiling
  - User-management
  - Authentication and authorisation
  - Human interaction
- Access

B1.6.1 Discovery

The use case described above has different aspects. Here we discuss how information is found. This could either happen by navigation or by search.

B1.6.1.1 Navigation (Categorization)

To enable a systematic navigation it would be necessary to provide information on polluted areas categorised in respect of spatial-properties. Consequently only such pieces of information containing information on pollution and containing spatial information are relevant. These pieces of information could be categorised according to some predefined set of spatial areas. This information on spatial areas could be provided as a polygon or a bounding box, e.g. conforming to ISO 19136. In addition to these spatial areas for "spatial" categorisation information to decide whether a specific piece of information is "pollution information" is necessary. This again can be considered as categorisation from a different point of view. Therefore the category "information on pollution" needs to be described, and potentially bidirectional references need to be established.

The following abstraction is derived from this concrete use-case:

Pieces of information are categorised. Categorisation takes place according to invariants (invariant properties among pieces of information contained in a category). There are different approaches towards categorisation/classification, a concrete classification-mechanism might combine some of them.

- Predefined categories

  Either a category refers to the information contained in it or the pieces of information refer to the categories they belong to. In the first case this could be done explicitly or more generic by describing this category's invariant properties or by a query matching all contained pieces of information. The latter case is trivial since each piece of information explicitly points to the categories it belongs to.

- User-specified categories

  Categorisation according to user-defined descriptions of categories. An example for such descriptions could be categorised sets of documents. In case of a machine learning approach these sets could be used as training, test and validation sets (Sebastiani, 2003).

- Automated categorisation

  In automated categorisation different kinds of classifiers are used. Among these classifiers there are probabilistic, decision-tree, decision-rule, linear and example-based classifiers, also neural networks can be used for classification. For more details one might refer to (van Rijsbergen, 1979) and (Sebastiani,2003). Often predefined sets of documents for training, evaluation and testing purposes are needed; In the field of "information retrieval" evaluation measures are known as precision, recall, and fall-out.

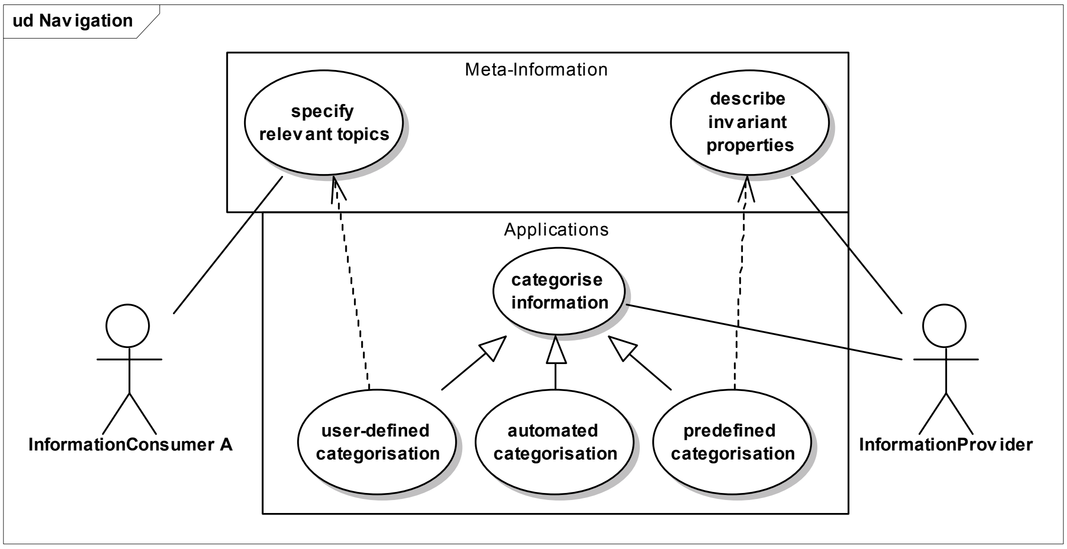
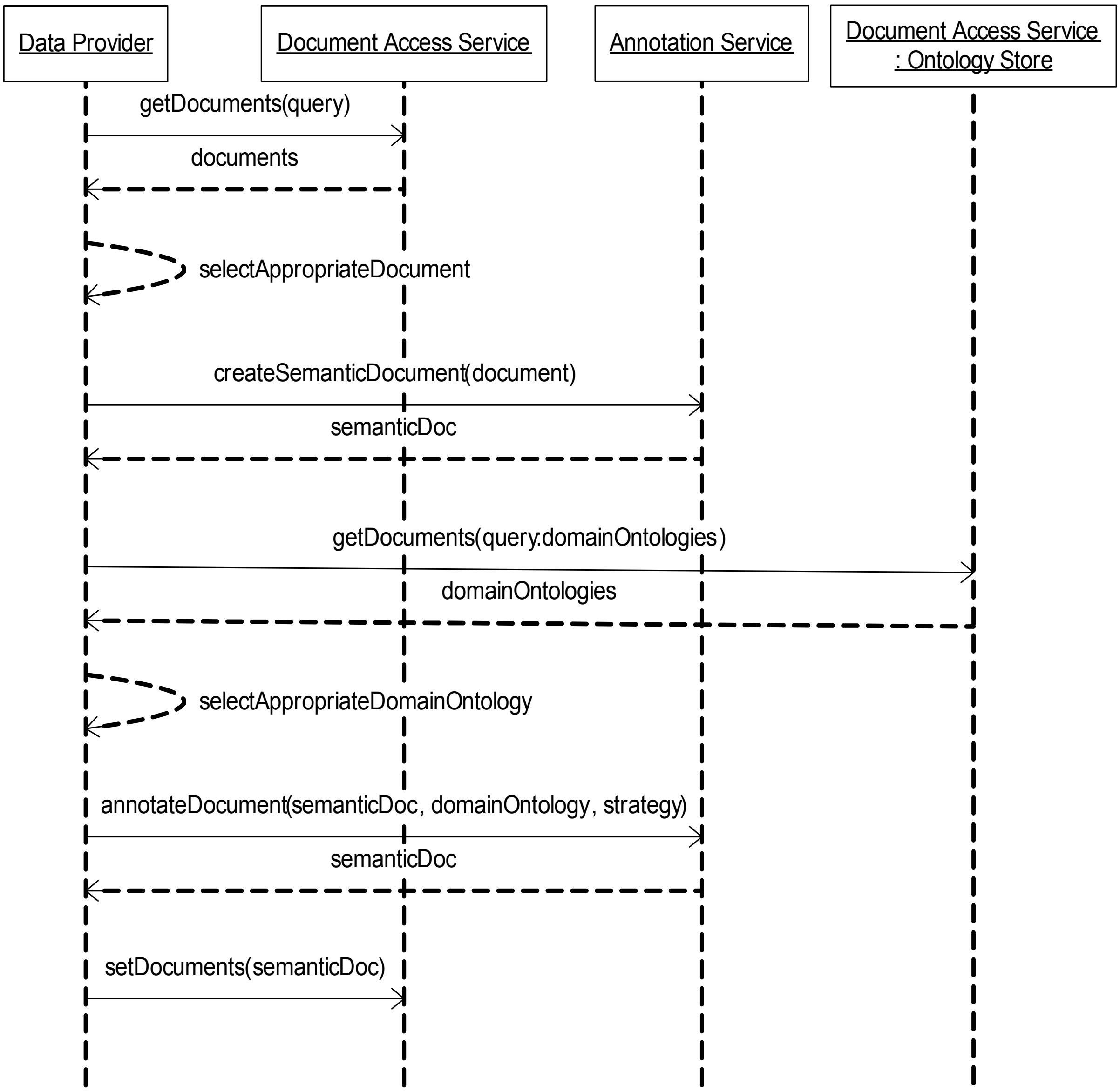


Figure 2: - Use-cases: categorisation

In Figure 3 is shown how a data-provider generates semantically annotated documents. The process is either automatic or semi-automatic. Referring to the "pollution-information use-case" these annotations can be used to decide whether a document contains information on pollution and spatial information.



Data Provider
Document Access Service
Annotation Service
Document Access Service : Ontology Store
getDocuments(query)
documents
selectAppropriateDocument
createSemanticDocument(document)
semanticDoc
getDocuments(query.domainOntologies)
domainOntologies
selectAppropriateDomainOntology
annotateDocument(semanticDoc, domainOntology, strategy)
semanticDoc
setDocuments(semanticDoc)

Figure 3: Sequence: annotation by data-provider

Note: The getDocuments-operation of the Document Access Services returns a list of “documents”, while the called operations of the Annotation Service expect only one single document as input. Therefore the data provider has to select one document.

The data-provider might replace the original document with the semantic document or decide to create a new document. The data-provider might even use another document access service to store the semantic document, e.g. in order to create or contribute to a dedicated knowledge base.

Meta-information identified in the context of discovery includes:

- Evaluation measures to judge automated categorisations:

$$precision = \frac{\left|\{relevant\_document\} \mathrm{I} \ \{retrieved\_document\}\right|}{\left|\{retrieved\_document\}\right|}$$

$$recall = \frac{\left|\{relevant\_documents\} \mathrm{I} \ \{retrieved\_documents\}\right|}{\left|\{relevant\_documents\}\right|}$$

$$fall-out = \frac{\left|\{irrrelevant\_documents\} \mathrm{I} \ \{retrieved\_documents\}\right|}{\left|\{irrelevant\_documents\}\right|}$$

Figure 4: Formulas for evaluation measures (precision, recall and fall-out; fromWikipedia "Performance measures")

- Semantic annotations, to raise the level of integration from syntax to semantics, this allows machines to work with explicit semantic information.

B1.6.1.2 Search

Following the "pollution-information use-case", also a search could satisfy our end-users information-needs. For an introduction towards search one may refer to (van Rijsbergen, 1979) for an information retrieval view and to (Knuth, 1973) for a mathematical or algorithmic view.

A searching user would define a query consisting of the type of information he is interested in (here: pollution-information) and stating which spatial areas should be considered. In addition, the user could define criteria to sort and to filter the search-results. In case of "sort" the user-specified criteria could be relevance, author's reputation, age or any other sortable attribute. In addition, the computation of such a query would require that information on the type of the information is available (is it "pollution information") and that it has spatial information so that the relevance for the defined spatial area(s) can be determined. Like in the case of "navigation" (see above) it could be required that the spatial information follows a specific standard, for example ISO 19136. This use-case is also related with user-profiling (for details please refer there).

Note: The set of documents considered as being valid hits may be further constraint by the access rights the information-consumer has (refer to section B1.6.3.3 User Management).

The following abstraction is derived from this use-case:

Sorting criteria could be the date (ascending), the author, the reputation of the authors, relevance in respect of the requested topic, … In the latter case so called *"topic signatures"* could be of interest to model a topic as meta-information (refer to Alfonseca, 2003).

Sorting criteria:

- Date

  So, that the most recent documents can be found

- Relevance

  The relevance of a search-result in respect of a topic specified by a *topic-signature* can be used for sorting order.

  Note: according to (Agirre, 2000) a topic-signature is a "family of related terms $\{t,<(w_1, s_1)\ldots(w_i, s_i)\ldots>\}$, where t is the topic (i.e. the target concept) and each w_i is a word associated with the topic, with strength s_i."

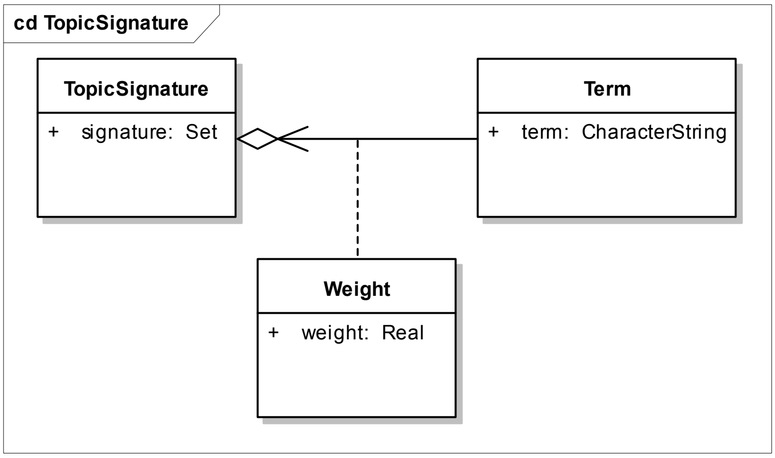


Figure 5: Class-diagram: topic-signature

- Reputation of author

  In order to get an idea of an author's reputation, information on that author's papers would be helpful. This information could be statements of readers like "helpful", or "not helpful". A rating system or a system comparing two papers would be ideas to get a key to this.

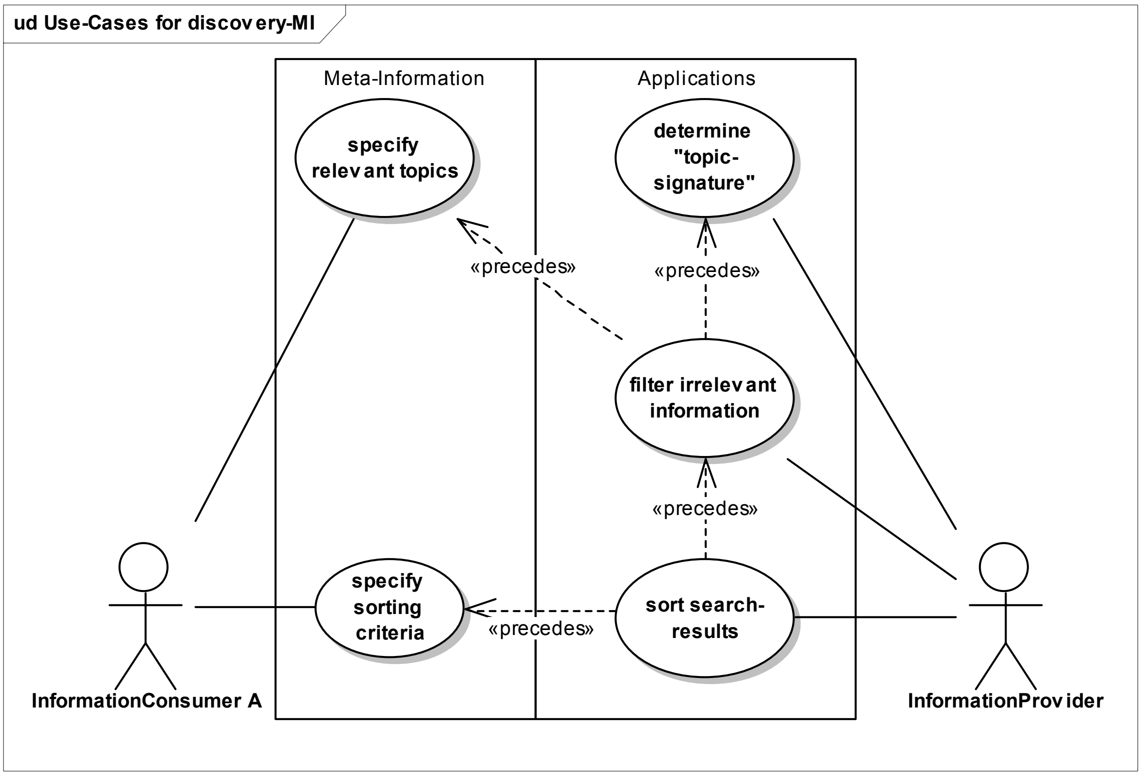


Figure 6: Use-cases for discovery-MI

Referring to (van Rijsbergen, 1979) search can be an interactive process, where the user gets hints in order to refine the query.

Meta-information that can be used for this would be:

- term-frequency in database

  The term-frequency could show that a term is quite often and hence lead the user to the decision to use a more specific term.

- number of fitting documents

  If this number is high, the user would define a more specific query.

- alternative and related terms

  These terms would give the user a hint for alternative query-formulations. the Thesaurus Access Service could provide this information.

- small cites out of the fitting documents

  A user could decide whether a documents fits without accessing the complete document.

- terms used to index the cites

  This can help the user to get an idea of how the underlying indexing-querying mechanism works.

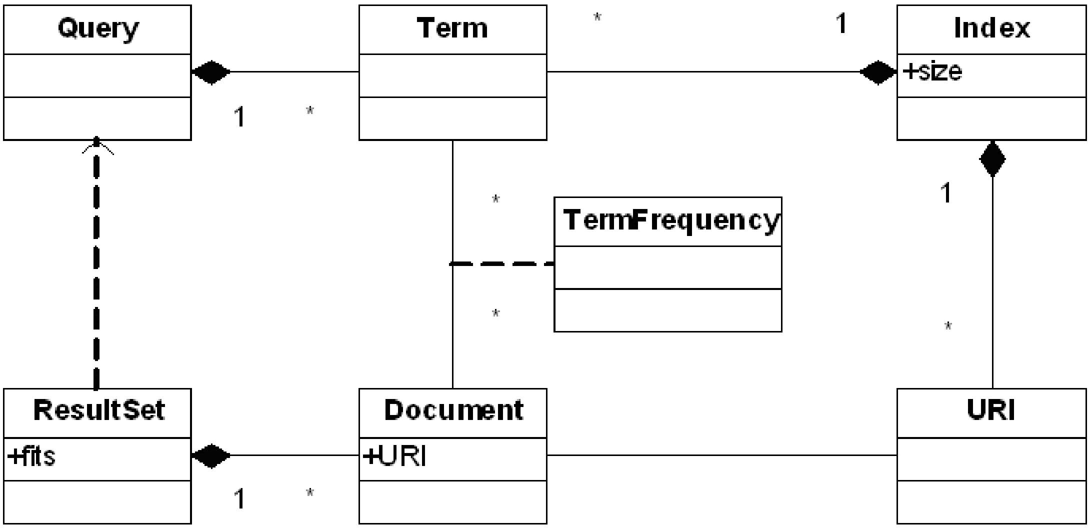


Figure 7: Meta-information to enable interactive search

B1.6.2 User Profiling

Following our particular use-case "pollution-information", user-profiling could provide meta-information to enhance the user's experience. It could enable the service to present the detected information in a user specific way just think about layout and level of detail, also the knowledge on preferred subjects of that particular user could provide hints whether information has to be considered a hit. Information on systems this user is capable to access would further constrain the set of information to be considered as being appropriate hits. This directly leads to the need to access discovered information and one might refer to the section on "Access" to further follow our use-case.

B1.6.2.1 Human Interaction Components (HIC) provide user-defined interaction components

The purpose of HCI-services is to provide interaction components for human users. An example for such a component would be a view on a specific feature. Such a view would contain textual parts and possibly multimedia content. A view shall reflect and meet the respective user's functional requirements and should accommodate that user's preferences. Functional requirements could be described by means of service meta-information (e.g. B1.8.1 OAS-MI for Service Capabilities), additional preferences a user has, need to be described, consequently, schemas for preferences like "layout" need to be defined.

Use cases:

- Preferred languages

  List of languages the user prefers to use (hear, read, speak); the level of skill should be stated, in order to provide a sorted list to pick the language from.

- Level of detail

  A user might want to get a brief overview of a specific field of interest, so the level of detail information is rendered should be reduced. An adequate level of detail might be to reduce relevant scientific papers to headlines and summaries. The possibly automated *summarisation* of texts could be one approach to cut down information to the required level of detail, another approach would be to filter irrelevant information (*"topic filtering"*), for details one might refer to (Alfonseca, 2003).

- User-specific pieces of information (privacy/nondisclosure)

  Only a subset of the overall available pieces of information is provided depending on the specific user. Reasons for this are to meet certain privacy requirements, or to protect secret information. This is strongly connected with the "level of detail" mentioned before.

- Rendering according to device type

  One important aspect of HI is rendering of output according to capabilities of a programme/device. For example, the layout and level of details may be different when information is presented on PDA-device, compared to output rendered on high resolution computer screen, or the output rendered by text-to-speech device. Descriptive information may be found in (W3C-CSS-Media, 2006).

- Layout

  The layout of a document might be defined by the information-consumer. A use-case in this context would be a user being a speed-reader, who knows to perform best when the text is provided according to a specific layout. A sample set of relevant properties would be colours, font-types, font-sizes, column-width, and so on.

  - Colours

    Colours are relevant for displayed information. The colours of text and background could be described.

  - Font-types

    A user might want to define the font-types of specific types of text. This could be by defining font-families (serif, sans-serif) or by defining specific font-types (Times New Roman, Arial, …).

  - Font-sizes

    A user might want to define the font-sizes of headlines, abstracts, normal text, and so on.

  - Column-width

    The column-width influences the reading-speed, therefore a user might want to specify a column to contain six to ten words per line or to contain not more than thirty characters.

The list of layout criteria is longer than this, an appropriate reference to further properties could be the definition of "cascading style sheets" (CSS, see http://www.w3.org/Style/CSS/).

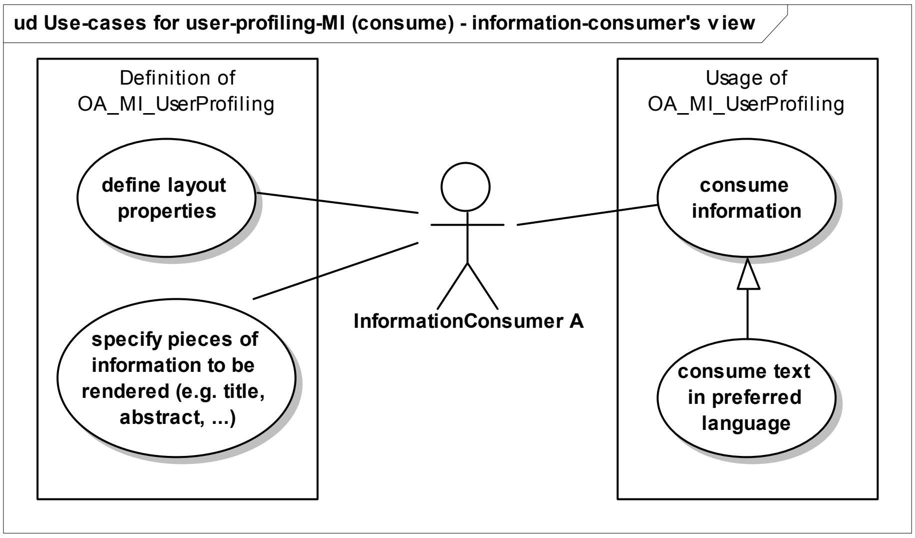

ud Use-cases for user-profiling-MI (consume) - information-consumer's view
Definition of OA_MI_UserProfiling
define layout properties
specify pieces of information to be rendered (e.g. title, abstract, ...)
InformationConsumer A
Usage of OA_MI_UserProfiling
consume information
consume text in preferred language

Figure 8: Use-cases for user-profiling-MI (consume) – information-consumer's view

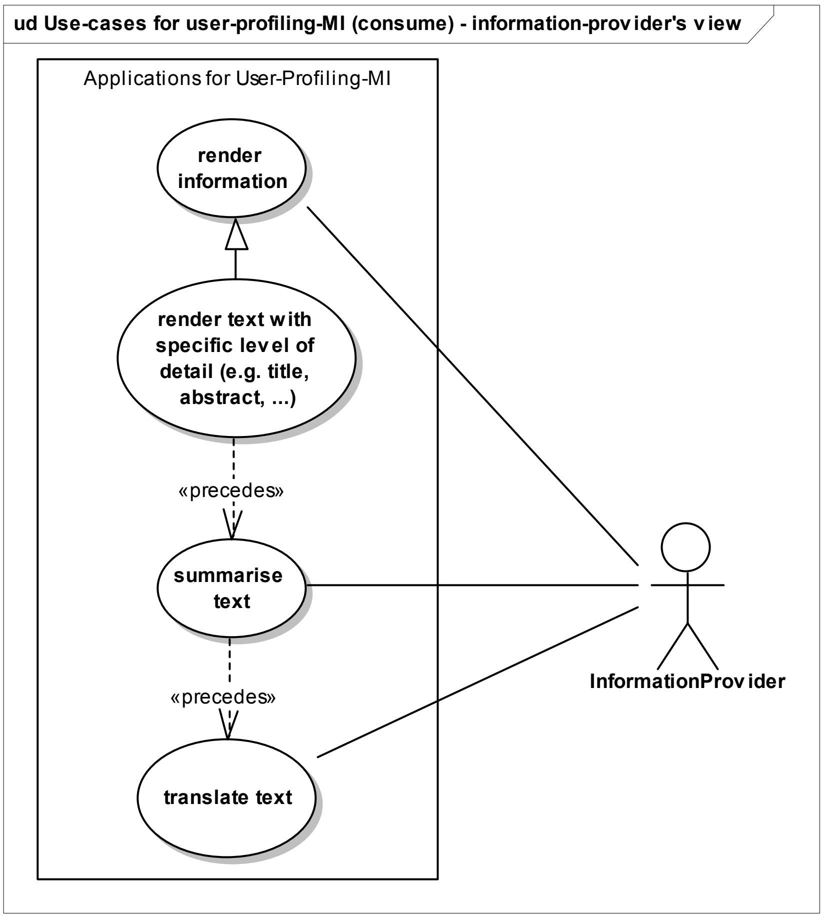

ud Use-cases for user-profiling-MI (consume) - information-provider's view
Applications for User-Profiling-MI
render information
render text with specific level of detail (e.g. title, abstract, ...)
«precedes»
summarise text
«precedes»
translate text
InformationProvider

Figure 9: Use-cases for user-profiling-MI (consume) - information-provider's view

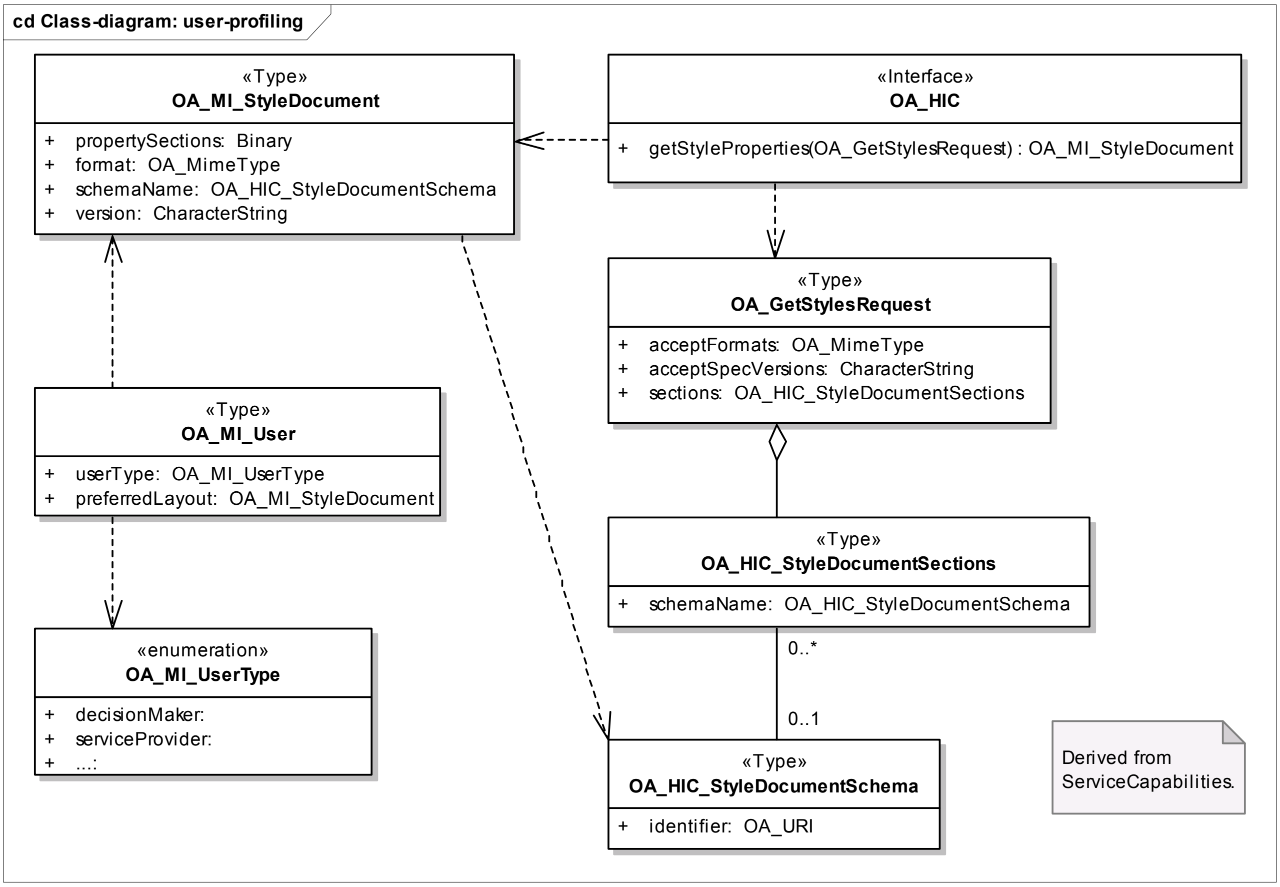

cd Class-diagram: user-profiling
«Type»
OA_MI_StyleDocument
+ propertySections: Binary
+ format: OA_MimeType
+ schemaName: OA_HIC_StyleDocumentSchema
+ version: CharacterString
«Interface»
OA_HIC
+ getStyleProperties(OA_GetStylesRequest) : OA_MI_StyleDocument
«Type»
OA_GetStylesRequest
+ acceptFormats: OA_MimeType
+ acceptSpecVersions: CharacterString
+ sections: OA_HIC_StyleDocumentSections
«Type»
OA_MI_User
+ userType: OA_MI_UserType
+ preferredLayout: OA_MI_StyleDocument
«Type»
OA_HIC_StyleDocumentSections
+ schemaName: OA_HIC_StyleDocumentSchema
0..*
0..1
«enumeration»
OA_MI_UserType
+ decisionMaker:
+ serviceProvider:
+ ...:
«Type»
OA_HIC_StyleDocumentSchema
+ identifier: OA_URI
Derived from ServiceCapabilities.

Figure 10: Class-diagram - style-document

- Trust

A fundamental challenge for OSNs is to provide quality information (cf. RM-OA section 11.2.4 “Quality”).

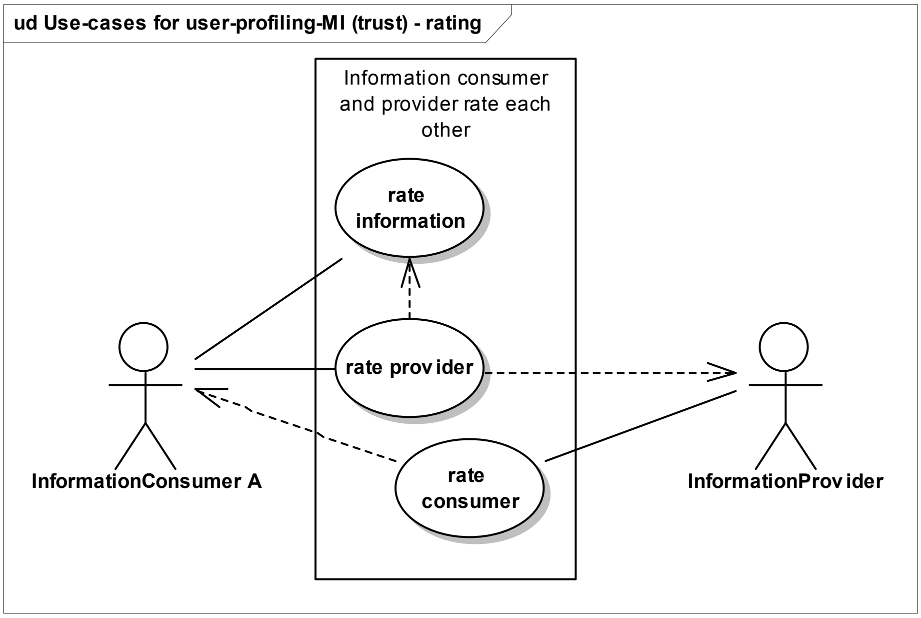

ud Use-cases for user-profiling-MI (trust) - rating
Information consumer and provider rate each other
rate information
rate provider
rate consumer
InformationConsumer A
InformationProvider

Figure 11: Use-cases for user-profiling-MI (trust) – mutual rating

Ratings are prerequisites to decide whether some specific user (information consumer/provider) is trustable. The rating of information is an indirect rating of the original information-provider. Examples for the application of reputation systems would be eBay or Slashdot.

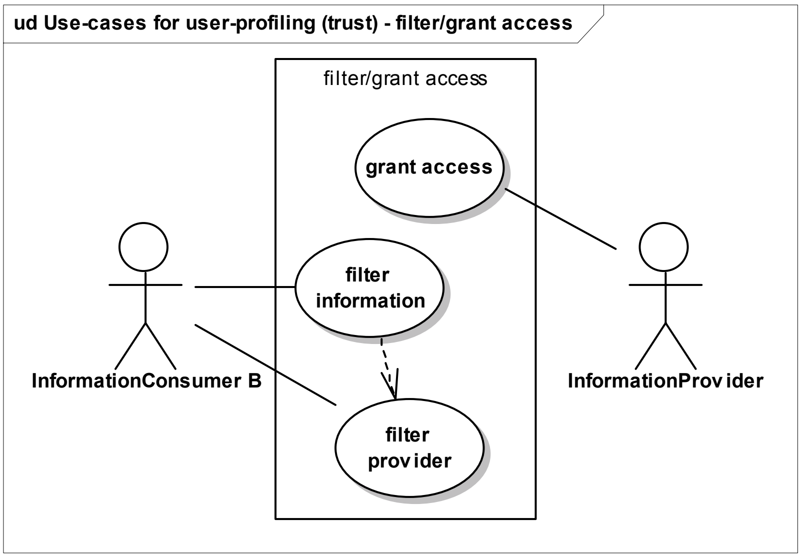


Figure 12: Decide to access/grant access on basis of ratings

Only trusted users will further on be considered as interaction partners, this holds for both (provider and consumer). If a provider is filtered, also its information is filtered.

- workbench

A workbench should provide the user with the means to fulfil the daily-work. Hence it is required, that the respective functional requirements are met. Therefore it is necessary, that the

- tasks the user regularly performs are described, and the
- pieces of information the user regularly needs are right at hand

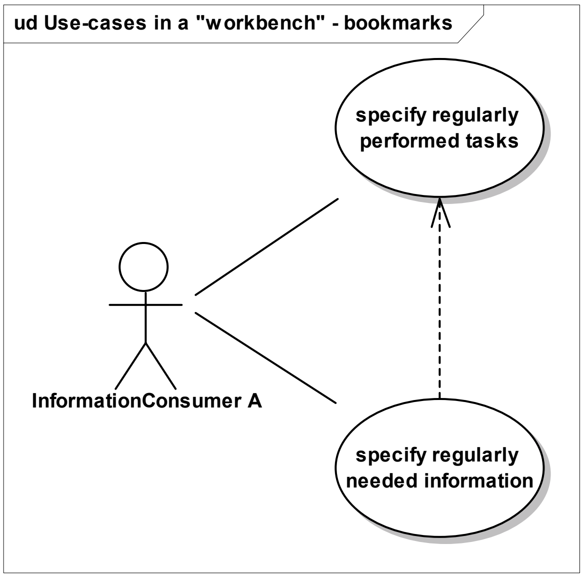

ud Use-cases in a "workbench" - bookmarks
specify regularly performed tasks
InformationConsumer A
specify regularly needed information

Figure 13: Use-cases in a "workbench" – bookmarks

The needed pieces of information are related with the tasks the user performs. A history of performed tasks containing relations to the used information enables to provide “shortcuts” to the user.

In addition to a history a user might use bookmarks.

A workbench may provide a bookmark-mechanism to allow the user to specify regularly performed tasks and regularly needed information. Also a workbench may log the users activity in order to provide a *history*.

Note: A software-engineer or frontend-designer might find relevant information in EN ISO 9241 “Usability” (the parts 10 to 17 address software), among other information the assessment of user-tasks and the involved information is described there.

B1.6.3 Access

In the context of our “pollution-information” use case the information-consumer has already discovered information either by navigation or search, decided which information should be accessed, and defined how to render the information. Access is the last step the user has to take before the information is rendered and finally presented. The aspects of the technical access are described in section B1.8.1, here some further aspects are covered:

- Authentication and Authorisation
    - Authentication mechanisms
    - Access right
- Payment
    - Modalities
    - Prices
    - Payment-Services

B1.6.3.1 Authentication and Authorisation

Authentication mechanisms

If an information-provider has certain constraints on the used authentication-mechanism, then he shall provide information on this. Examples would be that an information-provider only grants access to users of a specific type or instance of an authentication-service – instances could be identified by an URI.

Access rights

An information provider should set access rights for the data. Access rights might be set globally, for groups of users, or individually.

B1.6.3.2 Payment

Modalities

The modalities of payment describe when, how and where to pay the bill. Examples for payment-modalities would be payment in advance, in parts, on account, and so on.

Prices

The information-consumer has to be informed about the prize of the requested service in order to decide whether to actually retrieve the information. It is foreseen that this is done on the service-level of the specific service.

Note: under discussion!

Payment-services

There might be payment-services, an information-provider might collaborate with a set of such payment-services and hence require a consumer to use one of these services.

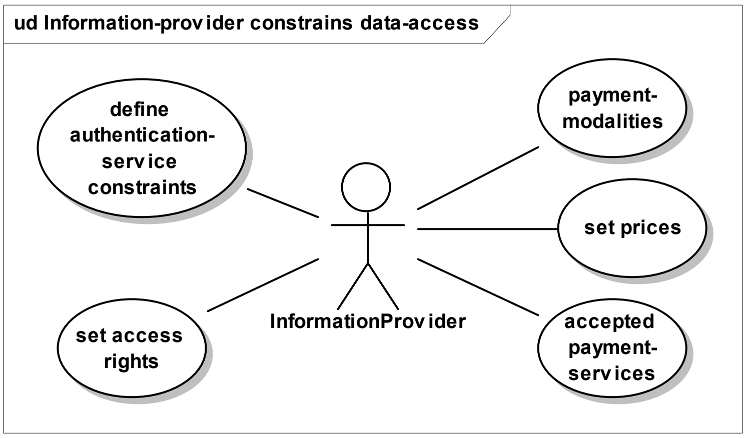

ud Information-provider constrains data-access
define authentication-service constraints
payment-modalities
set prices
set access rights
InformationProvider
accepted payment-services

Figure 14: Information-provider constrains data-access

B1.6.3.3 User-Management

As written in section B1.6.3.1 Authentication and Authorisation an information-provider might demand from the information-consumer to use a specific Authentication Service. Such an Authentication Service collaborates with a User Management Service.

The information provided by a User Management Service can be considered as meta-information for particular purposes:

In case of the purpose to *access* a resource it is obvious that the consumer shall have the appropriate rights. Referring to the User Management Service a subject representing this consumer shall exist. Principals are associated to a subject. There are two ways a subject gets the right for some action,

a) directly from a principal owning that right,

b) from a principal being a member of group carrying that right.

While these pieces of information are pure information in respect of access, they might turn into meta-information, in case of the purpose *discovery* the only documents the consumer is allowed to read may be considered as being valid hits.

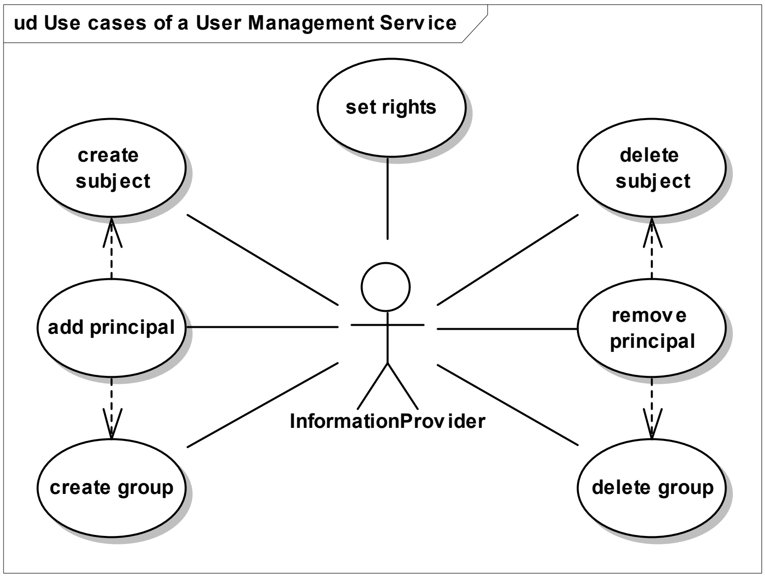


Figure 15: Origins of meta-information in the User Management Service

B1.7 Specification of Meta-Information Models

B1.7.1 Purpose "Discovery"

The purpose "discovery" encompasses methods to find relevant objects within a set of objects, namely search and navigation.

The procedure of searching starts with the formulation of a search query, which is submitted to the search engine. The search engine returns a number of objects it has identified as relevant with respect to the query (these are the search-results). Then, the initiator of the query can select objects from the results and/or refine the query.

Examples of meta-information supporting the search procedure are keyword lists, full text index, bounding areas or gazetteer mapping. Examples of services are the Document Access Service and the Gazetteer Service.

Navigation is the process of finding relevant information via browsing within navigational structures. These are provided either by a static or a dynamic catalogue. Example meta-information supporting navigation are catalogue entries or catalogue structures; an example of a service is the "Catalogue Service".

Discovery of services requires specific meta-information. The type of the needed meta-information depends on the quality of the discovery process: discovery might be user driven and just based on syntactical attributes, or it might be automated and based on semantic descriptions.

B1.7.2 Purpose related rules

Note: This section extends the rules of section B1.5.3. It will be improved and expanded in the next version and further identified general rules will be moved into section B1.5.3.

B1.7.2.1 Discovery of data

Search and navigation on data has different aspects, given by the nature of the data. The list of aspects is not closed, but time, space and thematic-aspects of discovery of data will be elaborated. Existing standards will be used if appropriate.

General rules for application-schemas of meta-information for discovery are listed in the following:

1. there shall be an **explicit relation from the meta-information to the described feature**, a relation between information and meta-information is obviously necessary. The direction of the relation is a consequence of a general ORCHESTRA requirement that is not to burden data-providers with unnecessary implementation-hurdles.
    a. a meta-information attribute about an attribute shall use the association "attributeOfAttribute" as defined in the RM-OA to establish a relation
    b. features being meta-information about features shall use an association of the OMM-MetaInfoAssociationType (see RM-OA, chpt. "7.5.4 OMM Extensions for Meta-Information Association Types).
2. meta-information shall contain **access-information**, this is information describing how to access the data. It is required because only knowing that data does exist and not to be able to retrieve access-information renders the data worthless. This access-information could either be technical information in case the data is functionally integrated with respect to access, or it could be a human readable description on how to gain access, e.g. contact information to a responsible person.
3. the meta-information should **use existing attribute types** where possible (refer to RM-OA, chpt. 7.4.3 OMM Attribute Types)

The aspects mentioned above (temporal, spatial, thematic) influence the requirements in the context of search and navigation, especially those requirements derived from components involved. These components are in case of search an index, and in case of navigation a catalogue-structure. Both need to be specialised in order to be able to discover relevant information under different aspects. The rules derived are:

- There may exist different types of search, the index used obviously has to support the underlying search paradigm. Different types of search are e.g. temporal, spatial and thematic search.
- The same applies for navigation and the underlying catalogue-structure.

In every case access-information is necessary. In case of "time" the index shall also contain information on the time-aspects of the data. This information shall follow a known syntax. To further raise the level of integration from syntactically, and functionally to semantically integrated, information on the meaning of the stated "temporal aspect" is necessary. This starts with the question whether the temporal aspect represents a **point in time** or a **duration**, a birthday, a publishing year, the date of collection, and so on.

B1.7.2.2 Example schemas

The Schemas (OAS-MIs) are required to be conformant to the above mentioned rules. The syntax and semantics of basic data types are described in section "B1.3".

Syntax and Semantics

The RM-OA defines different attribute types, among them types in the context of "time", these are derived from base classes out of the ISO 19100 series.

Temporal

Relevant types for temporal information can be found in ISO 19115 and ISO 19108. It is recommended to use these existing types where appropriate.

Spatial

ISO/FDIS 19107 "Geographic information – Spatial schema" and 19109 "Geographic information – Rules for application schema".

B1.8 Example OAS-MI

In this section, examples of ORCHESTRA Application Schemas for Meta-Information (OAS-MI) are described. In the following examples, the rules defined in section B1.5 are applied in addition to the rules defined for the OMM. When creating examples, the context of an application shall be assumed, which itself is to be described by means of an OAS. The OAS reflects the "normal functionality" of the application, while the OAS-MI describe meta-information needed for a certain purpose. Dependant on the level of detail chosen in the examples in the following sections, the OAS-MI may be associated to respective OAS.

B1.8.1 OAS-MI for Service Capabilities

This sub-section defines OAS-MI which may be used to structure the capabilities of any ORCHESTRA Service. The OAS-MI do not depend on assumptions about the service type or the application context of the service. They define the structure of service capabilities in the level of detail that can be achieved independent from these assumptions. Consequently this section does not contain the description of an OAS associated to the OAS-MI. The OAS-MI described in this section vary dependent on the quality needed for the descriptions, e.g. whether they are on a purely syntactical level or they also cover semantic descriptions.

B1.8.1.1 Introduction

Service capabilities comprise a set of meta-information of a service which can be delivered to a service user as a self-description of the service. They can be specified by means of an OAS-MI, which can be published in a catalogue and used by clients to discover a service. Furthermore, it should contain all necessary information enabling a client to invoke operations provided by a service.

In order to retrieve service capabilities, a common interface (Service Capabilities Interface) is defined in the context of the RM-OA Service to be supported by each ORCHESTRA Service. This interface defines a *getCapabilities* operation which is designed such that it is backward compatible with the concepts of the getCapabilities operation as defined in the OGC Web Services Common Specification (OGC 05-008). This means that the operation can be used to retrieve the service capabilities according to the schema defined in that OGC standard. According to the note in section B1.1.1 , the usage of that schema is just one possibility how the service capabilities may look like. The schema to be used to describe the service capabilities is not predefined by the getCapabilities operation specification. The operation allows the capabilities to be delivered according to any appropriate service meta-information schema supported by the service. The schema to be used can be selected by the client in the getCapabilities request. In principle, the list of possible schemas is kept open and may include standard service meta-information schemas from ISO (ISO 19115/19119), OGC (OGC 05-008) and the Semantic Web community (e.g. OWL-S and WSMO).

However, for interoperability purposes, it is recommanded to use one schema (so called default-schema) within an OSN in order to express service capabilities. This schema shall be defined according to the rules for ORCHESTRA application schemas for meta-information. The schema defined here is intended to act as a default OAS-MI for Service Capabilities.

Service capabilities are mainly used for the purposes of service discovery and service invocation. Therefore, in a first step, the schema defined here concentrates on meta-information based on these main purposes. Focus is on meta-information based on a syntactical description. Section B1.8.1.6 shows how the schema may be extended in order to include a semantic service description to be used for semantic service discovery and invocation.

Within the schema defined here schema sections are identified. Each section has a name and comprises a well-defined part of the schema. The idea behind the introduction of schema sections is related to the getCapabilities operation. In the getCapabilities request the client can explicitly restrict the returned set of capabilities to certain sections by listing the section names. Schema sections which are not referred in the request can be omitted in the getCapabilities response in order to minimize the size of transmitted data. The schema sections are definied within the following sub-sections.

B1.8.1.2 Service Capabilities – Overall Structure

Service capabilities can either be common capabilities or service-specific capabilities.

- Common capabilities are capabilities which are relevant for each service and therefore have a common structure. The common capabilities are further refined in the following sections.
- Specific capabilities are only relevant for certain service types. The structure of these capabilities is therefore service-specific and can not be refined here. This has to be done in the context of each service specification resulting in service-specific schema extensions.

The common part may contain for example detailed information about all the operations which a service provides as each ORCHESTRA Service is supposed to provide a certain set of operations. An example for a specific capability is a list of supported query languages as this may only apply to some service types.

The following figure shows the top-level type OA_MI_Service_Capabilities as an aggregation of OA_MI_Service_CommonCapabilities and OA_MI_Service_SpecificCapabilities. While the common part is further refined in the next section, the specific part has to be specialized by service-specific schema extensions as needed (find an example in section B1.8.2.3.3).

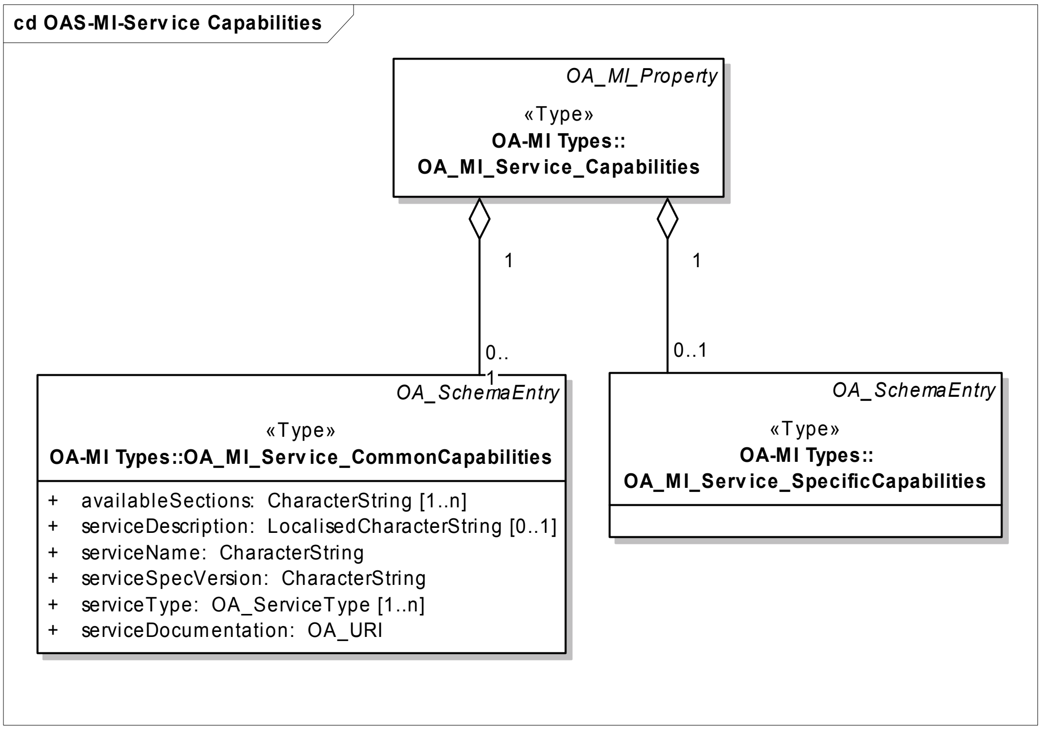

cd OAS-MI-Service Capabilities
OA_MI_Property
«Type»
OA-MI Types::
OA_MI_Service_Capabilities
1
1
0..
1
0..1
OA_SchemaEntry
«Type»
OA-MI Types::OA_MI_Service_CommonCapabilities
+ availableSections: CharacterString [1..n]
+ serviceDescription: LocalisedCharacterString [0..1]
+ serviceName: CharacterString
+ serviceSpecVersion: CharacterString
+ serviceType: OA_ServiceType [1..n]
+ serviceDocumentation: OA_URI
OA_SchemaEntry
«Type»
OA-MI Types::
OA_MI_Service_SpecificCapabilities

Figure 16: Overall structure of Service Capabilities

As a consequence, on the top-level the schema is divided into the following general sections:

Section Name	Section Contents
common	OA_MI_Service_CommonCapabilities
specific	OA_MI_Service_SpecificCapabilities

Table 15: General Sections of OA_MI_Service_Capabilities

A client invoking the getCapabilities operation can thus retrieve e.g. only the common capabilities by referring to the section name "common" in the request. The result is then an instance of OA_MI_Service_Capabilities where the specific capabilities part is absent.

The section "common" has further subsections which are defined in the subsequent section. In the same way, the section "specific" may also contain subsections if necessary which then have to be defined in the context of the respective service specific schema extensions.

B1.8.1.3 Common Service Capabilities – Overall Structure

The overall structure of the common service capabilities is shown in the following figure.

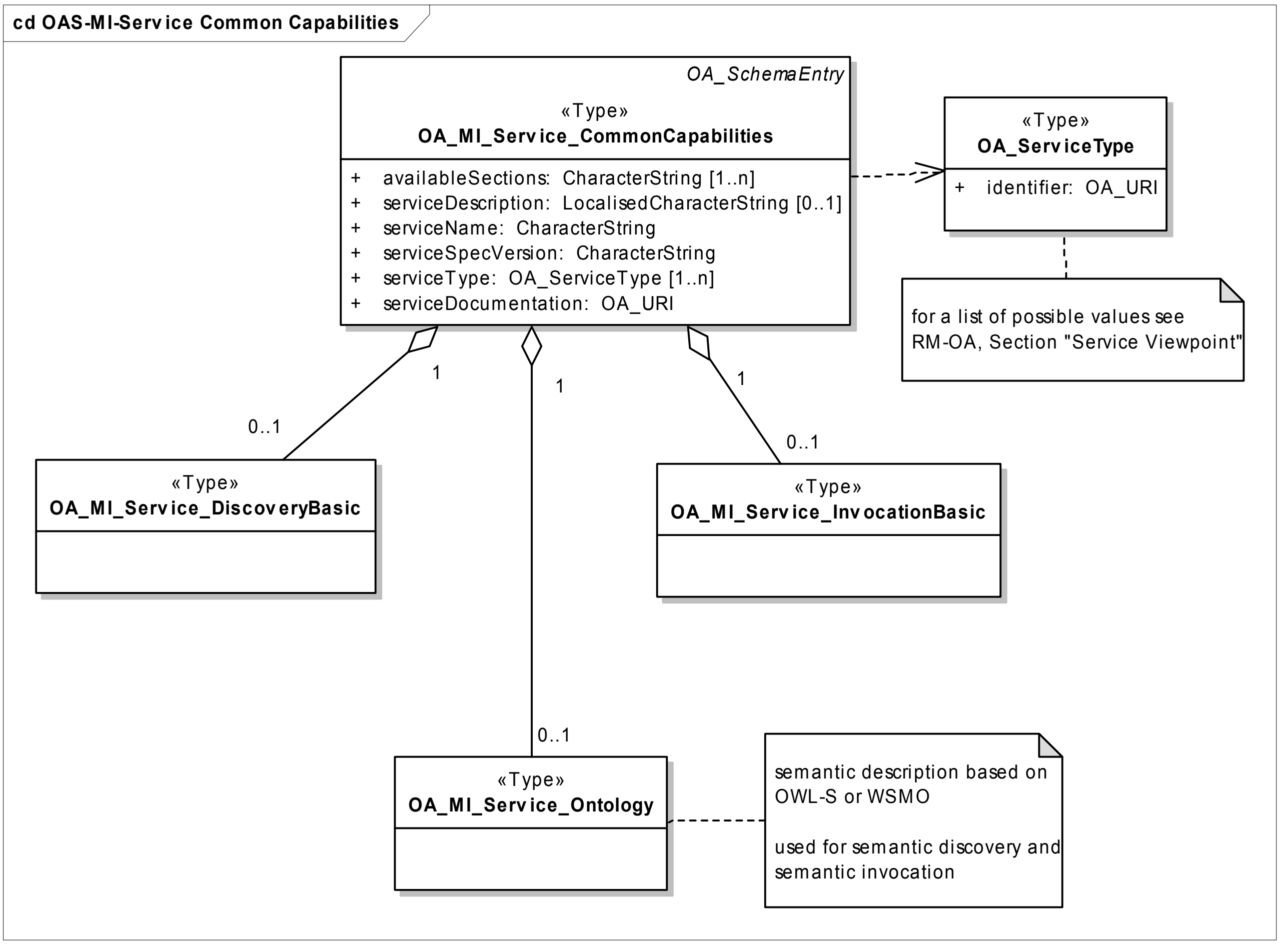

cd OAS-MI-Service Common Capabilities
OA_SchemaEntry
«Type»
OA_MI_Service_CommonCapabilities
+ availableSections: CharacterString [1..n]
+ serviceDescription: LocalisedCharacterString [0..1]
+ serviceName: CharacterString
+ serviceSpecVersion: CharacterString
+ serviceType: OA_ServiceType [1..n]
+ serviceDocumentation: OA_URI
«Type»
OA_ServiceType
+ identifier: OA_URI
for a list of possible values see RM-OA, Section "Service Viewpoint"
1
0..1
1
1
0..1
«Type»
OA_MI_Service_DiscoveryBasic
«Type»
OA_MI_Service_InvocationBasic
0..1
«Type»
OA_MI_Service_Ontology
semantic description based on OWL-S or WSMO
used for semantic discovery and semantic invocation

Figure 17: Overall structure of Common Service Capabilities

Common capabilities comprise some fields independent of a certain purpose and aggregates further schema parts which are purpose specific.

The purpose independent fields are:

Name	Definition	Data Type	Multiplicity and Use
serviceName	Service instance name	CharacterString, not empty	One (mandatory)
serviceDescription	Brief narrative description of this service instance, normally available for display to a human	LocalisedCharacterString	Zero or one (optional)
serviceType	A service type identifier from a list of service types. The RM-OA contains a list of service types as part of the Service Viewpoint.	OA_ServiceType	One or more (mandatory)
serviceSpecVersion	Version of the service implementation specification (OIS) to which the service instance (OSI) conforms.	CharacterString	One (mandatory)
availableSections	List of the names of all schema sections and subsections for which information can be provided by the service instance.	CharacterString, not empty	One or more (mandatory)
serviceDocumentation	Link to the full documentation of the service	OA_URI	One (mandatory)

Table 16: Purpose independent fields of the common section

The attribute availableSections acts as a kind of "table of contents" for the capabilities of a service instance. It contains a list of the names of all schema sections for which information is available and can be provided by the service instance. This is a "flat list" containing section names as well as any subsection names. The list refers both to all available sections of the common capabilities part (which are subsections of the "common" section) and to all available sections of the service-specific capabilities part (which are subsections of the "specific" section).

In addition to the described fields, there are associated parts which are purpose specific (according to RM-OA Annex A3:Section A3.6 "Particular purposes") and represented by the following related types:

- OA_MI_Service_DiscoveryBasic
- OA_MI_Service_InvocationBasic
- OA_MI_Service_Ontology

These types and the related purposes are described in the subsequent sections.

As a consequence, at the common capabilities' level the following sections (which are subsections of the "common" section) are identified:

Section Name	Section Contents
common.root	OA_MI_Service_CommonCapabilities (without associated purpose specific parts)
common.discovery	OA_MI_Service_DiscoveryBasic
common.invocation	OA_MI_Service_InvocationBasic
common.semantics	OA_MI_Service_Ontology

Table 17: Subsections of the common section

The section "common.root" can be used as a starting point when retrieving the service capabilities of a service instance using the getCapabilities operation. When a client requests only this section in a getCapabilities request, then all other sections are omitted. Thus only a small amount of data needs to be transferred to the client, containing only the attributes of type OA_MI_Service_CommonCapabilities including the list of available sections. A client can then explicitly request sections of that list in one or more additional getCapabilities requests.

When the schema is extended to support additional purposes, corresponding sections should be defined in addition.

Note: Structuring the schema into purpose-specific parts as done in the following is a conceptual approach. However, a strict distinction between the purposes can not always be done. For example, meta-information which is primarily used for service invocation (like e.g. supported operations) may sometimes also be useful for service discovery (e.g. searching for a service which supports a certain operation). At least, it should not be forbidden to use meta-information originally designed for one purpose also for other purposes.

B1.8.1.4 Service Discovery

The part of the service meta-information schema related to the purpose discovery based on a syntactical description is outlined in the following figure. The corresponding root type is called OA_MI_Service_DiscoveryBasic to express that the service is described on a basic level in contrast to an advanced semantic level.

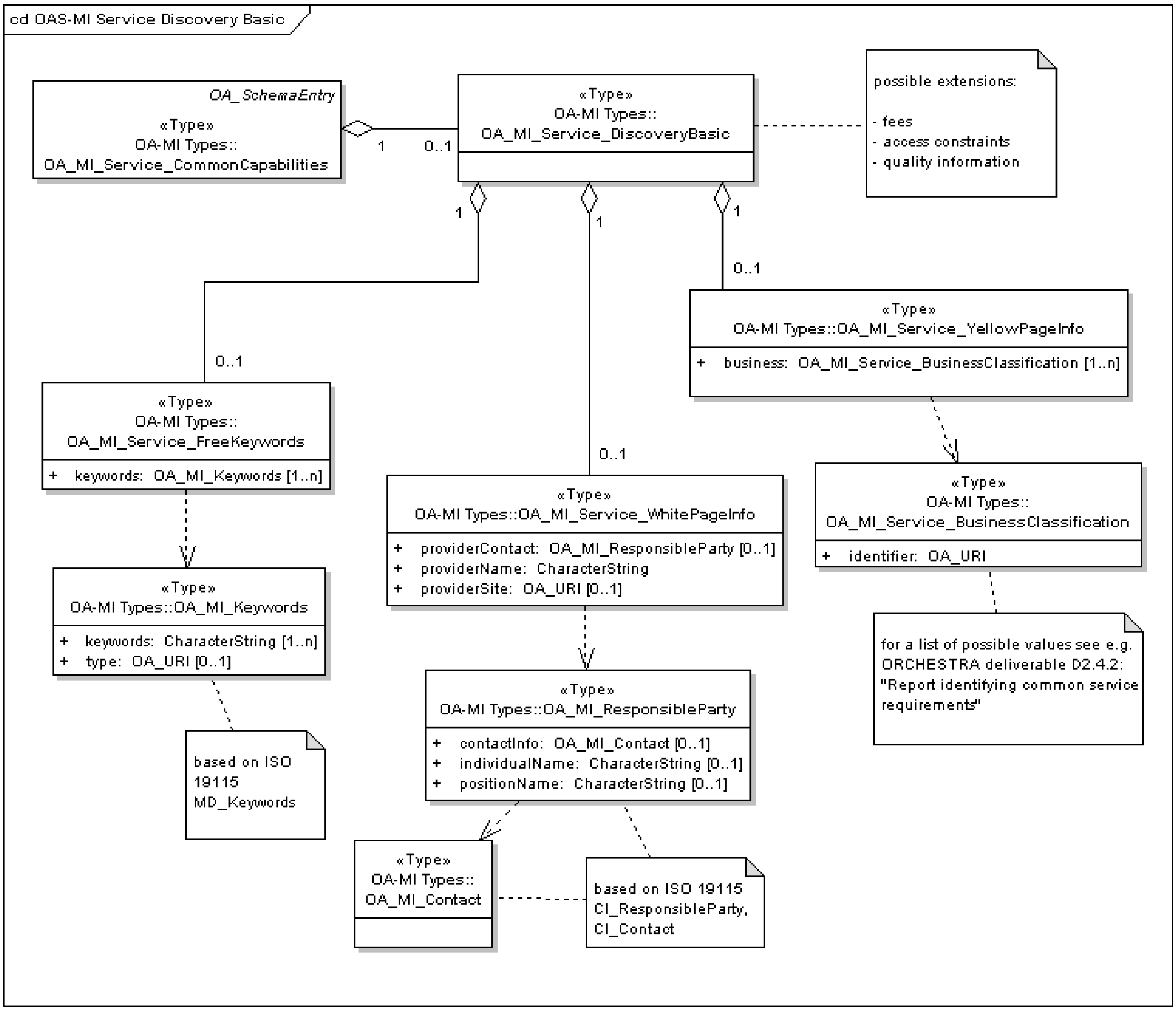

cd OAS-MI Service Discovery Basic
OA_SchemaEntry
«Type»
OA-MI Types::
OA_MI_Service_CommonCapabilities
1
0..1
«Type»
OA-MI Types::
OA_MI_Service_DiscoveryBasic
possible extensions:
- fees
- access constraints
- quality information
1
1
1
0..1
«Type»
OA-MI Types::OA_MI_Service_YellowPageInfo
+ business: OA_MI_Service_BusinessClassification [1..n]
0..1
«Type»
OA-MI Types::
OA_MI_Service_FreeKeywords
+ keywords: OA_MI_Keywords [1..n]
0..1
«Type»
OA-MI Types::OA_MI_Service_WhitePageInfo
+ providerContact: OA_MI_ResponsibleParty [0..1]
+ providerName: CharacterString
+ providerSite: OA_URI [0..1]
«Type»
OA-MI Types::
OA_MI_Service_BusinessClassification
+ identifier: OA_URI
«Type»
OA-MI Types::OA_MI_Keywords
+ keywords: CharacterString [1..n]
+ type: OA_URI [0..1]
for a list of possible values see e.g. ORCHESTRA deliverable D2.4.2: "Report identifying common service requirements"
«Type»
OA-MI Types::OA_MI_ResponsibleParty
+ contactInfo: OA_MI_Contact [0..1]
+ individualName: CharacterString [0..1]
+ positionName: CharacterString [0..1]
based on ISO 19115 MD_Keywords
«Type»
OA-MI Types::
OA_MI_Contact
based on ISO 19115 CI_ResponsibleParty, CI_Contact

Figure 18: Schema of Service Discovery

Figure 18 shows that the type OA_MI_Service_DiscoveryBasic is an aggregation of structures for specifying free keywords, white page information (information about the service provider) and yellow page information (information about the related business). Each of these blocks contains a number of attributes which are described in the following table. The value in the “multiplicity and use” column is only of interest if the respective block is present.

Name	Definition	Data Type	Multiplicity and Use
keywords	Unordered list of one or more commonly used or formalised words or phrases used to describe this service	OA_MI_Keywords based on MD_Keywords class in ISO 19115	One or more (mandatory)
providerName	Unique identifier for service provider organization	CharacterString, not empty	One (mandatory)
providerSite	Reference to the most relevant web site of the service provider	OA_URI	Zero or one (optional)
providerContact	Information for contacting the service provider	OA_MI_ResponsibleParty based on CI_ResponsibleParty and subsidiary classes in ISO 19115	Zero or one (optional)
business	In ORCHESTRA deliverable D2.4.2 ("Report identifying common service requirements") services are classified according to business classes such as "Meteorological Services", "GEO Information Services" etc.	OA_MI_ServiceBusiness-Classification	One or more (mandatory)

Table 18: Attributes for Service_Discovery

The type OA_MI_Keywords is used as a container for a set of keywords to characterize the service. It is based on the corresponding type MD_Keywords defined in ISO 19115. In addition to the set of keywords it contains an optional type attribute which can be used to identify the source of the keywords by indicating its URI.

The type OA_MI_ResponsibleParty is based on the corresponding type CI_ResponsibleParty defined in ISO 19115. It contains the following attributes:

- individualName: name of the responsible person (optional)
- positionName: role or position of the responsible person (optional)
- contactInfo: contact information (optional). The type OA_MI_Contact is based on the corresponding type CI_Contact defined by ISO 19115.

B1.8.1.4.1 Possible Extensions

OGC Web Services Common Specification (OGC 05-008) defines in its Service Identification section additional service capabilities which may also be relevant for service discovery.

- Fees: Fees and terms for retrieving data from or otherwise using this server, including the monetary units as specified in ISO 4217.
- Access Constraints: Access constraints that should be observed to assure the protection of privacy or intellectual property, and any other restrictions on using this service.

Such extensions could become relevant for a future extension of this schema. The same applies to information concerning the quality of service.

B1.8.1.5 Service Invocation

Meta-information for service invocation comprises a syntactical description of the implemented operations of a service. The corresponding root type is called OA_MI_Service_InvocationBasic to express that the description is on a basic level in contrast to an advanced semantic level. The type is an aggregation of operation descriptions as outlined in Figure 19.

Note: Currently, only operation-specific information is defined here. When necessary, the schema can be extended to contain also meta-information for service invocation which is not operation-specific. The type OA_MI_Service_InvocationBasic can be used for such extensions.

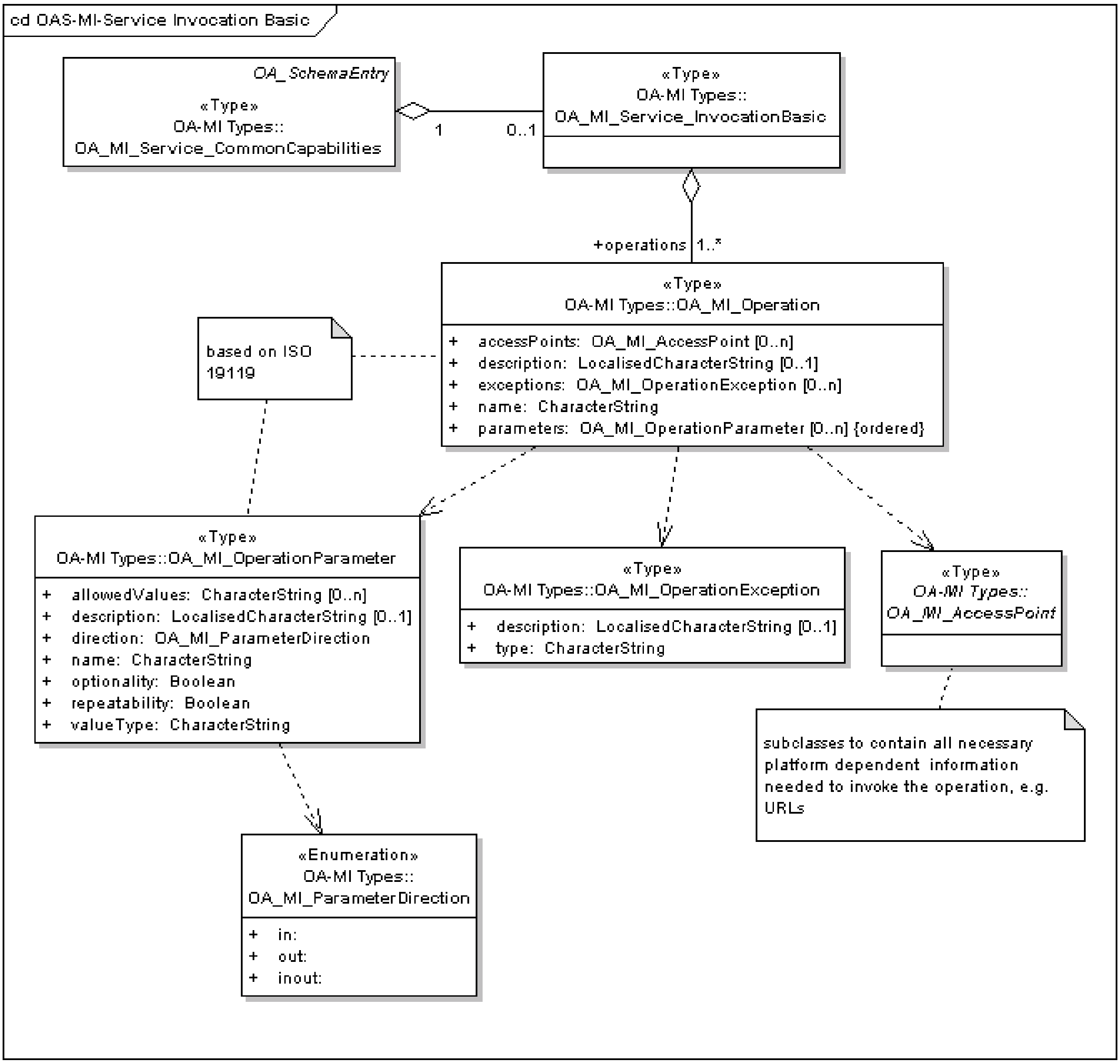

cd OAS-MI-Service Invocation Basic
OA_SchemaEntry
«Type»
OA-MI Types::
OA_MI_Service_CommonCapabilities
1
0..1
«Type»
OA-MI Types::
OA_MI_Service_InvocationBasic
+operations
1..*
«Type»
OA-MI Types::OA_MI_Operation
+ accessPoints: OA_MI_AccessPoint [0..n]
+ description: LocalisedCharacterString [0..1]
+ exceptions: OA_MI_OperationException [0..n]
+ name: CharacterString
+ parameters: OA_MI_OperationParameter [0..n] {ordered}
based on ISO 19119
«Type»
OA-MI Types::OA_MI_OperationParameter
+ allowedValues: CharacterString [0..n]
+ description: LocalisedCharacterString [0..1]
+ direction: OA_MI_ParameterDirection
+ name: CharacterString
+ optionality: Boolean
+ repeatability: Boolean
+ valueType: CharacterString
«Type»
OA-MI Types::OA_MI_OperationException
+ description: LocalisedCharacterString [0..1]
+ type: CharacterString
«Type»
OA-MI Types::
OA_MI_AccessPoint
subclasses to contain all necessary platform dependent information needed to invoke the operation, e.g. URLs
«Enumeration»
OA-MI Types::
OA_MI_ParameterDirection
+ in:
+ out:
+ inout:

Figure 19: Schema of Service Invocation

The way of describing operations and its parameters is based on ISO 19119 (Open GIS Service Architecture), where basic meta-information needed for service invocation is described. Each operation supported by the service is represented by an instance of type OA_MI_Operation. Its attributes are described in Table 19.

Name	Definition	Data Type	Multiplicity and Use
name	Operation name	CharacterString, not empty	One (mandatory)
description	Brief narrative description of the operation, normally available for display to a human	LocalisedCharacterString	Zero or one (optional)
parameters	Ordered list of operation parameter descriptions	OA_MI_Operation-Parameter	Zero or more (optional) One for each operation parameter
exceptions	Description of each exception which can be thrown by the operation	OA_MI_Operation-Exception	Zero or more (optional) One for each possible operation exception
accessPoints	Platform dependent information needed to invoke the operation (e.g. URLs in case of web services)	OA_MI_AccessPoint (abstract type, to be specialised by platform-dependent subtypes)	Zero or more (optional) One for each implemented access point

Table 19: Attributes for OA_MI_Operation

Each operation parameter is represented by an instance of type OA_MI_OperationParameter. The order of these instances corresponds to the order of parameters as expected by the operation. The attributes of type OA_MI_OperationParameter are described in Table 20.

Name	Definition	Data Type	Multiplicity and Use
name	Parameter name	CharacterString, not empty	One (mandatory)
description	Brief narrative description of the parameter, normally available for display to a human	LocalisedCharacterString	Zero or one (optional)
optionality	True, if the parameter is optional; false, if the parameter is mandatory	Boolean	One (mandatory)
repeatability	True, if more than one value of the parameter may be provided	Boolean	One (mandatory)
direction	Indicates whether the parameter is an input parameter, an output parameter or a combined input/output parameter	OA_MI_Parameter-Direction	One (mandatory)
valueType	Identifier of the type of the parameter	CharacterString, not empty	One (mandatory)
allowedValues	List of valid values for the parameter, if applicable	CharacterString, not empty	Zero or more (optional)

Table 20: Attributes for OA_MI_OperationParameter

Each exception which can be thrown by the operation is represented by an instance of type OA_MI_OperationException. Its attributes are described in Table 21.

Name	Definition	Data Type	Multiplicity and Use
type	Identifier of the type of the exception	CharacterString, not empty	One (mandatory)
description	Brief narrative description of the exception including its cause	LocalisedCharacterString	Zero or one (optional)

Table 21: Attributes for OA_MI_OperationException

B1.8.1.5.1 Example of an Operation-specific Extension

An instance of type OA_MI_Operation can be regarded to contain the basic information necessary to invoke a certain operation. However, in some cases this basic information is not sufficient to use the operation in an appropriate way. Therefore, an operation description may extend the type OA_MI_Operation to provide additional information necessary to invoke an operation. As an example, one such extension is described in the following.

Consider the operation *invoke* of the Synchronous Interaction Interface defined by the RM-OA Service. According to the specification the invoke operation has a single input parameter “request” of type OA_OperationRequest and a single output parameter “response” of type OA_OperationResponse. By means of these two parameters, the invoke operation can be used to dynamically invoke some other operation *Op* and retrieve its result. To do so, the name of the operation *Op* and its input parameter values have to be combined into an instance of OA_OperationRequest which is then used in the invoke request. After execution of the invoke operation, the output parameter values of *Op* are available in the resulting OA_OperationResponse structure (including any occurred exception related to *Op*).

In order to use the invoke operation in the described way, information is needed about the operations which are supported by the invoke operation. In principle, it is necessary to know the names and parameters of the possible operations *Op*. Therefore, a straightforward approach is, to describe these operations *Op* in the same way as done for any other service operation. The resulting list is then attached as additional attribute to the description of the invoke operation.

Figure 20 illustrates that the type OA_MI_InvokeOperation which is introduced for this purpose extends the OA_MI_Operation by adding a list of supported operations where each element in the list is again of type OA_MI_Operation.

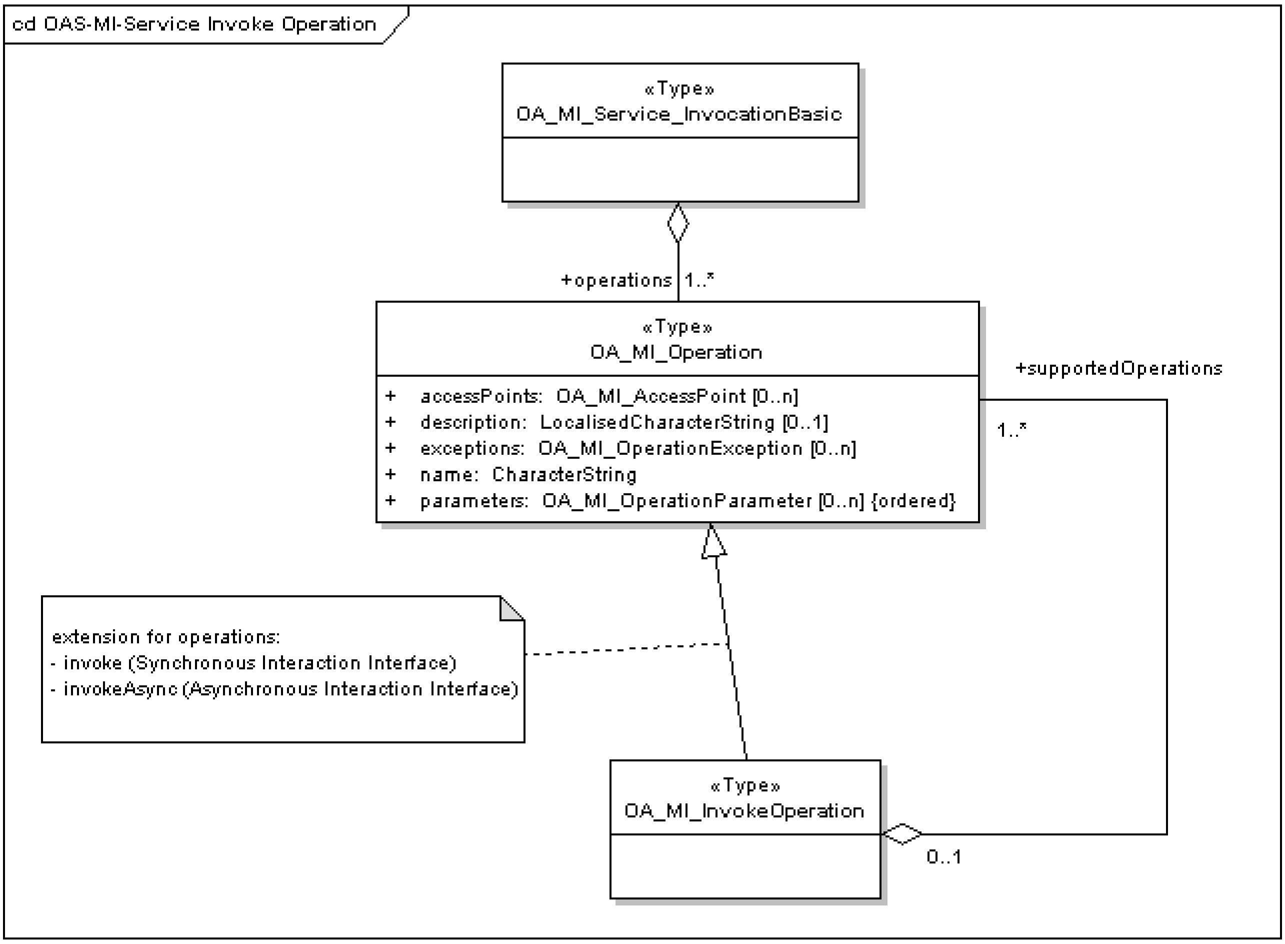

cd OAS-MI-Service Invoke Operation
«Type»
OA_MI_Service_InvocationBasic
+operations
1..*
«Type»
OA_MI_Operation
+ accessPoints: OA_MI_AccessPoint [0..n]
+ description: LocalisedCharacterString [0..1]
+ exceptions: OA_MI_OperationException [0..n]
+ name: CharacterString
+ parameters: OA_MI_OperationParameter [0..n] {ordered}
+supportedOperations
1..*
extension for operations:
- invoke (Synchronous Interaction Interface)
- invokeAsync (Asynchronous Interaction Interface)
«Type»
OA_MI_InvokeOperation
0..1

Figure 20: Operation-specific extension for invoke operations

In the same way the *invokeAsync* operation of the Asynchronous Interaction Interface of the RM-OA Service can be handled. The same type OA_MI_InvokeOperation can be used to describe that operation including the list of operations which can be invoked through the invokeAsync operation.

Example

As an example, a list of operation descriptions is shown in the following Figure 21 which is structured according to this schema. The syntax is very much simplified and should be self-explanatory. The operation names are written in bold font. For each operation only the parameters and its types are indicated in ()-brackets.

The getCapabilities entry is an example of a normal operation description structured according to type OA_MI_Operation. The invokeAsync entry is structured according to the extended type OA_MI_InvokeOperation. Therefore, this entry contains an additional part “supportedOperations” which lists two operations: exampleOp1 and exampleOp2. Each one is again structured according to OA_MI_Operation.

Note that exampleOp1 only appears inside of the invokeAsync entry which means that exampleOp1 is an operation which can only be invoked by means of the invokeAsync operation. By contrast, exampleOp2 appears both inside invokeAsync and on the top-level which indicates that this operation can be invoked by means of invokeAsync but also as a normal service operation.

```
operations {

    getCapabilities {
        (request: OA_GetCapabilitiesRequest,
         response: OA_CapabilitiesDocument
        )
    },

    invokeAsync {
        (request: OA_OperationRequest,
         callback: NotificationCallback,
         result: OA_InvokeID
        ),
        supportedOperations {
            exampleOp1 {(req: CharacterString,
                         result: Integer)},
            exampleOp2 {(xyz: CharacterString)}
        }
    },

    abort {(invokeID: OA_InvokeID)},
    exampleOp2 {(xyz: CharacterString)}
}
```

Figure 21: Example of operation descriptions

B1.8.1.6 Semantic Service Description

This section intends to show how a schema is extended in order to include a semantic service description used for semantic service discovery and invocation.

Currently, there are two major initiatives for semantic descriptions of services: the Web Ontology Language for Services (OWL-S) and the Web Service Modelling Ontology (WSMO). In ORCHESTRA, semantic service descriptions can either be implemented on an OWL-S or on a WSMO basis. In both cases the service is described by means of a service ontology: OWL-S defines an OWL based upper ontology for services, while WSMO Ontologies are expressed in WSML, Web Service Modelling Language.

Note: For different purposes (e.g. discovery, invocation) the same service ontology can be used, it then shall contain corresponding aspects as explained in the following subsections.

In Figure 22 the service ontology is represented by means of the type OA_MI_Service_Ontology which is an additional part of the common capabilities aggregation.

The concepts described in the service ontology may depend on certain domain knowledge; such knowledge is usually expressed in a domain ontology. For instance, discovery of road maps may be based on knowledge about what roads are, that streets and highways are roads, too, and so on. Therefore, the service ontology may use (or import) the concepts described in one or more domain ontologies. The service ontology refers to these domain ontologies.

B1.8.1.6.1 Service Discovery based on Semantic Descriptions

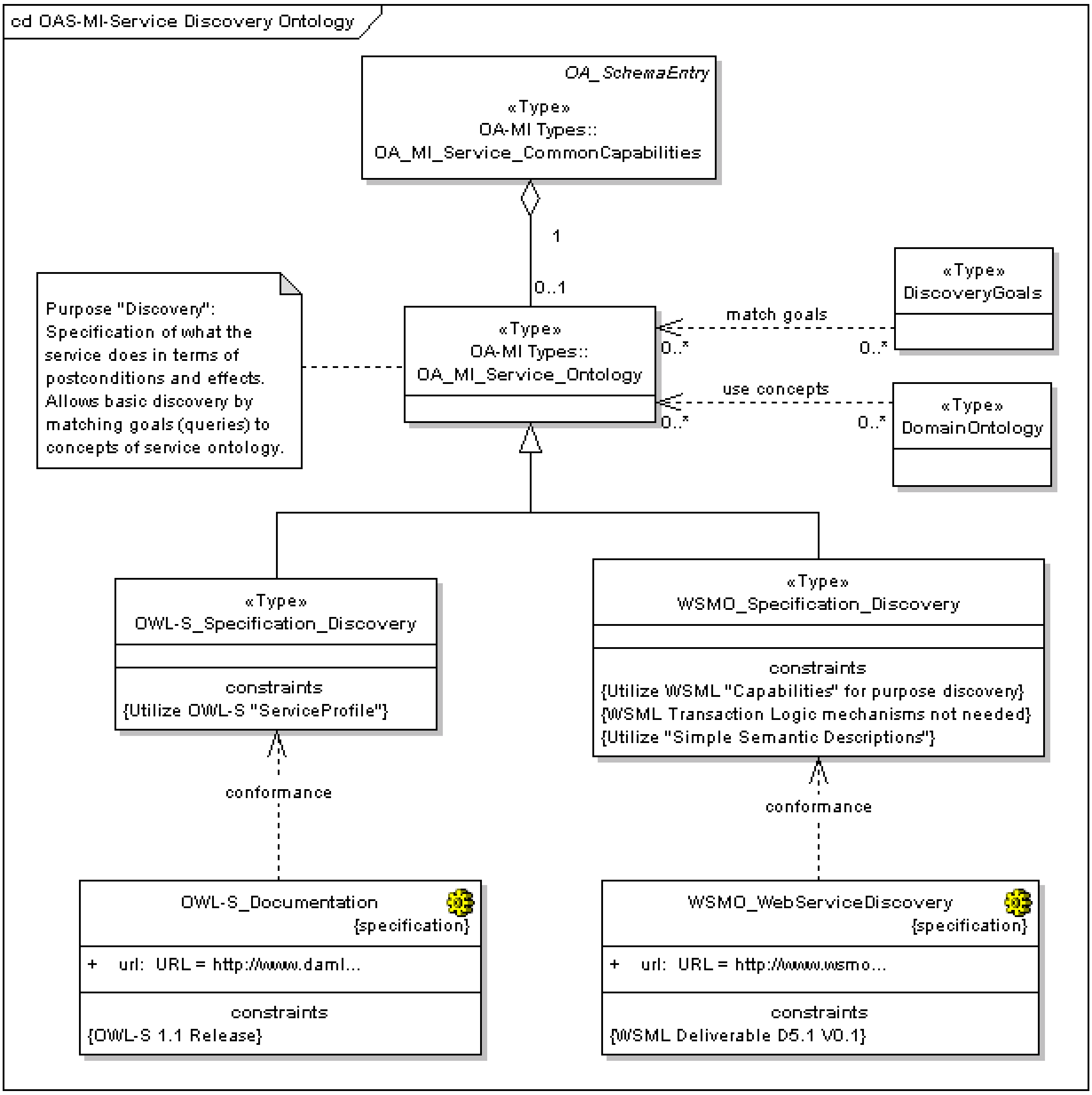

cd OAS-MI-Service Discovery Ontology
OA_SchemaEntry
«Type»
OA-MI Types::
OA_MI_Service_CommonCapabilities
1
0..1
Purpose "Discovery":
Specification of what the service does in terms of postconditions and effects.
Allows basic discovery by matching goals (queries) to concepts of service ontology.
«Type»
OA-MI Types::
OA_MI_Service_Ontology
match goals
0..*
0..*
«Type»
DiscoveryGoals
use concepts
0..*
0..*
«Type»
DomainOntology
«Type»
OWL-S_Specification_Discovery
constraints
{Utilize OWL-S "ServiceProfile"}
«Type»
WSMO_Specification_Discovery
constraints
{Utilize WSML "Capabilities" for purpose discovery}
{WSML Transaction Logic mechanisms not needed}
{Utilize "Simple Semantic Descriptions"}
conformance
conformance
OWL-S_Documentation
{specification}
+ url: URL = http://www.daml...
constraints
{OWL-S 1.1 Release}
WSMO_WebServiceDiscovery
{specification}
+ url: URL = http://www.wsmo...
constraints
{WSML Deliverable D5.1 V0.1}

Figure 22: Schema for Service Discovery based on Semantic Descriptions

As shown in the Figure 22, both OWL-S and WSMO can be used to describe a service for the purpose of discovery. OWL-S defines an OWL based upper ontology for services, while WSMO Ontologies are expressed in WSML, the Web Service Modelling Language. Discovery is done by matching goals (queries) to concepts of the service ontology. In WSMO, goals can be expressed by means of an own language construct, while in OWL-S queries can be formulated by means of OWL compatible query languages. With both approaches, concepts of domain ontologies can be used in order to specify the concepts of the service ontology.

For the purpose of discovery, OWL-S and WSMO offer specific language constructs:

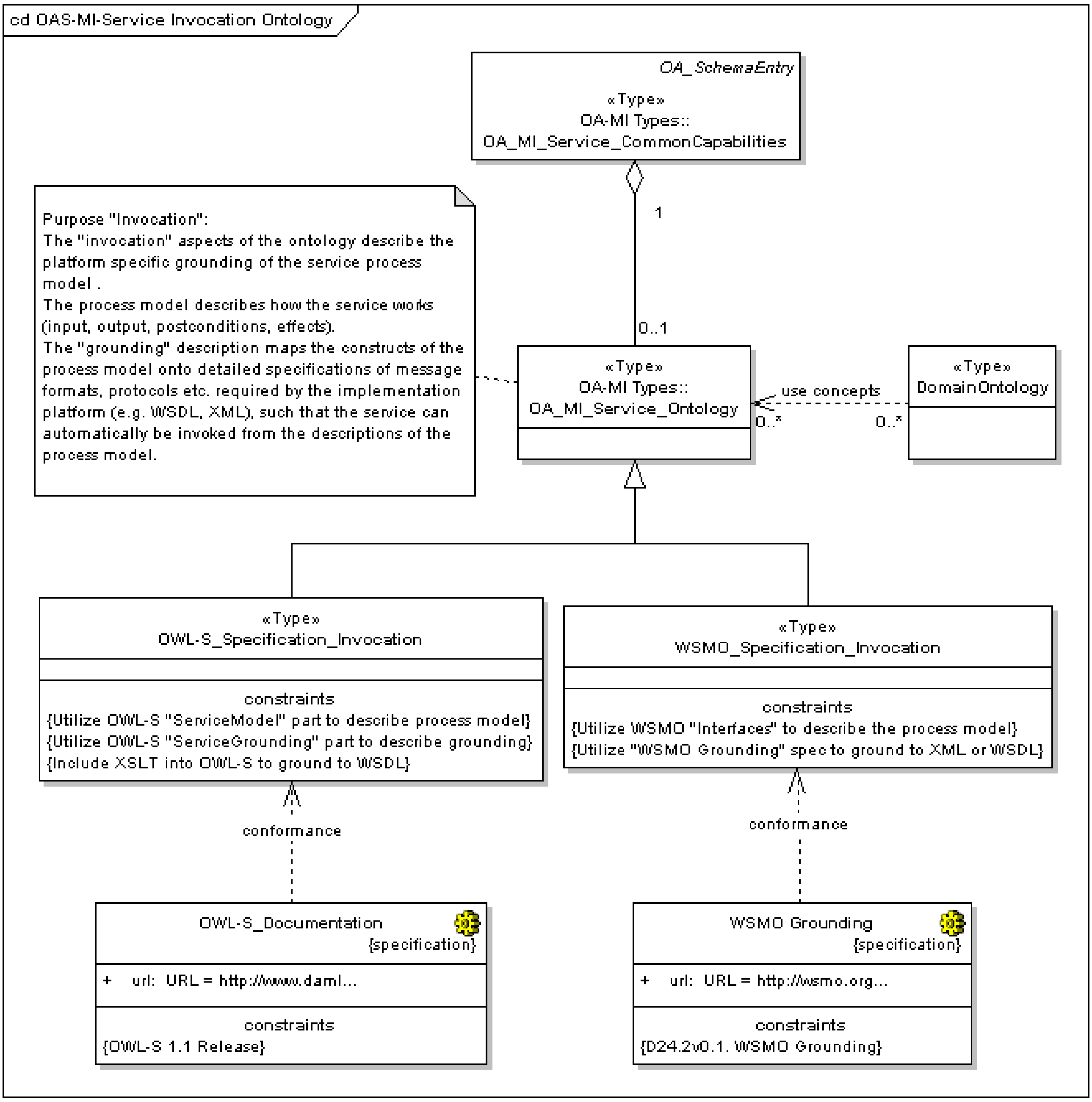
- In OWL-S, capabilities of services can be described by means of the “ServiceProfile” concept. In order to be compatible to OWL-S, an ORCHESTRA implementation should utilize profiles as specified in the OWL-S documentation.
- WSMO has specified an own deliverable for purpose discovery (Keller et.al, 2004). In WSMO, “capabilities” are the constructs for semantic description of services. However, WSMO

capabilities are a rich construct and only a subset of them is needed in order to specify a service ontology capable of matching goals to capabilities. WSMO utilizes a set based approach for the matching, i.e. goals and capabilities are expressed in terms of sets and matching is based on logical considerations on these sets. The purpose discovery only requires the so called “simple semantic descriptions”, which do not make use of transaction logic (an extension of First Order Logic which can be used to describe how service input relates to output by explicitly considering state transitions).

B1.8.1.6.2 Service Invocation based on Semantic Descriptions

In order to automatically invoke a discovered service without any further manual intervention, a mapping between the semantic service description and the concrete service invocation interfaces has to be done. Information used for the purpose of constructing messages in the requested format is called “grounding” information.

A service grounding can be thought of as a mapping from an abstract process model onto detailed specifications of those service description elements that are required for interacting with the service (e.g. message format, serialization, transport and addressing). The abstract process model is thus to be defined in the service ontology, as shown in Figure 23:



cd OAS-MI-Service Invocation Ontology
OA_SchemaEntry
«Type»
OA-MI Types::
OA_MI_Service_CommonCapabilities
1
0..1
«Type»
OA-MI Types::
OA_MI_Service_Ontology
use concepts
0..*
0..*
«Type»
DomainOntology
«Type»
OWL-S_Specification_Invocation
constraints
{Utilize OWL-S "ServiceModel" part to describe process model}
{Utilize OWL-S "ServiceGrounding" part to describe grounding}
{Include XSLT into OWL-S to ground to WSDL}
«Type»
WSMO_Specification_Invocation
constraints
{Utilize WSMO "Interfaces" to describe the process model}
{Utilize "WSMO Grounding" spec to ground to XML or WSDL}
conformance
conformance
OWL-S_Documentation
{specification}
+ url: URL = http://www.daml...
constraints
{OWL-S 1.1 Release}
WSMO Grounding
{specification}
+ url: URL = http://wsmo.org...
constraints
{D24.2v0.1. WSMO Grounding}

Purpose "Invocation":
The "invocation" aspects of the ontology describe the platform specific grounding of the service process model .
The process model describes how the service works (input, output, postconditions, effects).
The "grounding" description maps the constructs of the process model onto detailed specifications of message formats, protocols etc. required by the implementation platform (e.g. WSDL, XML), such that the service can automatically be invoked from the descriptions of the process model.

Figure 23: Schema for Service Invocation based on Semantic Descriptions

The process model describes how the service works in terms of input, output, post conditions and effects (IOPE). The grounding maps the constructs of the process model onto the detailed specifications of the implementation platform (e.g. WSDL, XML). This allows the service to be automatically invoked by a program or an agent from the semantic descriptions in the service ontology.

Both OWL-S and WSMO can be used to specify the process model and grounding:

- In OWL-S, the process model can be described by means of the “ServiceModel” concept. The grounding itself can be described by using the “ServiceGrounding” concept. OWL-S provides support for grounding to WSDL, the W3C language for describing web service interfaces. The groundings are described by means of XSL Transformations (XSLT), which can be included in the OWL-S service ontology. A detailed description of the grounding mechanism can be found in the OWL-S documentation.

- In WSMO's language WSML, the process model can be described by means of "interfaces". For WSMO, a deliverable dedicated to grounding (Kopecký et.al, 2005) describes the grounding of the interfaces to the specifications of the implementation platform. Currently, WSML specifications can be grounded to XML or to WSDL.

B1.8.1.7 Points still under discussion

This section lists some points which came up during discussion of the schema.

- Type OA_MI_Service_DiscoveryBasic, attribute "providerContact": Should it really be optional?
- Type OA_MI_OperationParameter, attribute "repeatability": Is the boolean type appropriate here? Is an indication of a minimal and maximal number required? Note that the boolean type was chosen according to ISO 19119.
- Structuring the schema into purpose-specific parts as done here is a conceptual approach. A strict distinction between the purposes can not always be done. There might be overlapping parts. Meta-information classified for a certain purpose may be utilized in the context of another purpose. In order to support new purposes, capabilities from existing parts may be used and/or new parts can be defined if necessary. The question is whether a more appropriate structure can be found.

B1.8.2 OAS-MI for ORCHESTRA Pilot “German Bight”

This section defines an OAS-MI that is used in the ORCHESTRA pilot „Assessment of risks generated by ship traffic activity in the German Bight – Wadden Sea Area” (in the sequel called “German Bight Pilot”). The pilot application comprises services for simulation management and access to simulation data. These services are described by an OAS-MI that comprises a selection of common capabilities as described in section B1.8.1, and specific capabilities dependent on the service type. An example of an OAS-MI of a concrete service described by an OAS is elaborated, and the association between the OAS-MI and the OAS is explicated in UML diagrams.

B1.8.2.1 Pilot Overview

The ORCHESTRA German Bight pilot application allows access to and construction of multi risk maps for end users and institutions, which operate information systems in the area of the German Bight. Map generation is based on the integrated use of information about risk factors, shipping routes, environmental databases, numerical models and cartographic data. Data entered via the human user interface or retrieved from source systems are evaluated in simulation models. In “what if” scenarios, e.g. simulated introduction of new shipping routes, the end user is able to simulate the spatial distribution of risks (e.g. antifoulant pollution, spills of harmful substances) by means of dynamic generation of risk maps showing the dependencies form ship traffic, weather situation and other impacts.

The source data (e.g. current, weather, bathymetrie, temperature, cartography, species distribution, toxicity levels, shipping routes, leaching rates of antifoulants) are made accessible by means of dedicated ORCHESTRA Service Instances (OSIs). This category of services, the Simulation Data Services (SDS), provide access to the source data by implementing the interface of the Source System Integration Service (SSIS), which allows to include external source systems into an ORCHESTRA Service Network (OSN).

In a simulation run the source data are conducted to a Simulation Management Service (SMS). An SMS is an OSI that carries out a workflow in order to retrieve data from various SDS in the right order. An SMS is able to execute various simulation models.

The focus of the pilot w.r.t. the overall goals of ORCHESTRA is to provide a mediated access to simulation services. The mediation takes place by means of a Catalogue Service that is compliant to OGC and the respective implementation specification of the ORCHESTRA Catalogue Service. Figure 24 displays the basic interworking of the pilot components:

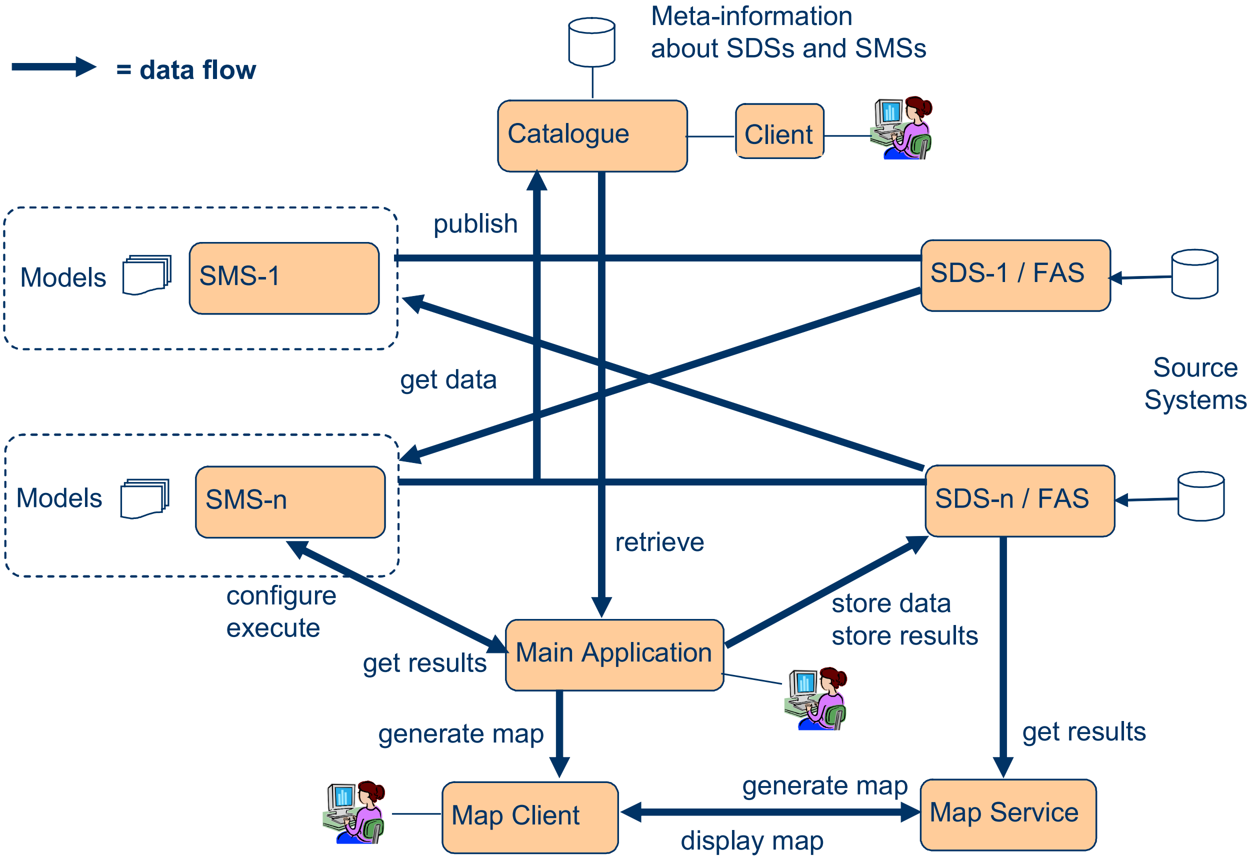


Figure 24: Generic Use Case: Basic Interworking

- In a step prior to a simulation run, the meta-information describing the SMSs and the SDSs available in this OSN has to be published in the Catalogue.
- The user of the main application can retrieve meta-information from the Catalogue. He/she looks at that meta-information in order to decide about which SMS to select for a simulation run, and which of the SDSs should deliver the data needed for the simulation.
- In a pre-processing phase, the main application prepares the data needed for the simulation such that it can be retrieved by the SMS during a simulation run, and the workflow of the SMS is set up.
- In the next phase, the simulation is executed. The SMS gets all data needed from the respective SDS, which has been configured with the data accommodated for the SMS.
- In a post-processing phase, the main application converts the results computed by the SMS into a format needed for further processing steps (visualisation, report generation). The data needed in these presentation steps (e.g. maps in GML format) are stored at an SDS and are thus accessible through an SDS/FAS interface.
- The Main Application then instructs the Map Client to generate a map from the simulation results. The Map Client instructs the Map Service for this. The Map Service will get a reference of an SDS where it can retrieve the results. The Map Service hands over the generated map to its client for presentation.

Meta-information entered into the Catalogue in order to enable the main application user to select the right SMS and to mediate the right SDS to the selected SMS comprises

1. Meta-information about each SMS
2. Meta-information about each SDS

In the following sections, the meta-information in the catalogue will be modelled based on the structure defined for service capabilities, as described in section B1.8.1.This structure foresees “common” capabilities, which describe the meta-information that has a common structure for all services, and "specific" capabilities, for which the structure is defined specifically for a certain service type. SMS and SDS could be seen as service groups, for which the specific capabilities may comprise a part that is commonly structured for the respective group.

Meta-information held in the Catalogue about a service is a subset of the capabilities held at the service itself.

B1.8.2.2 Meta-information about SMS

B1.8.2.2.1 SMS Common Capabilities

The following figure shows a selected profile of the common capabilities for discovery of SMS as described in the ORCHESTRA application schema. Those parts to be described in the catalogue are explained in attached notes (text in red colour within the diagrams).

Meta-information comprises a description by means of keywords which can be freely chosen, a textual description and a business classification. For SMS Services, the respective value is set to "Simulation Management Service".

According to rule 4.3.5 in section B1.5.3, the meta-information for discovery shall be linked with the information needed to access discovered objects. This is indicated in the figure by the type "OAS-MI-ServiceInvocationBasic", which describes the capabilities needed to invoke the service (as explained in section B1.8.1.5.

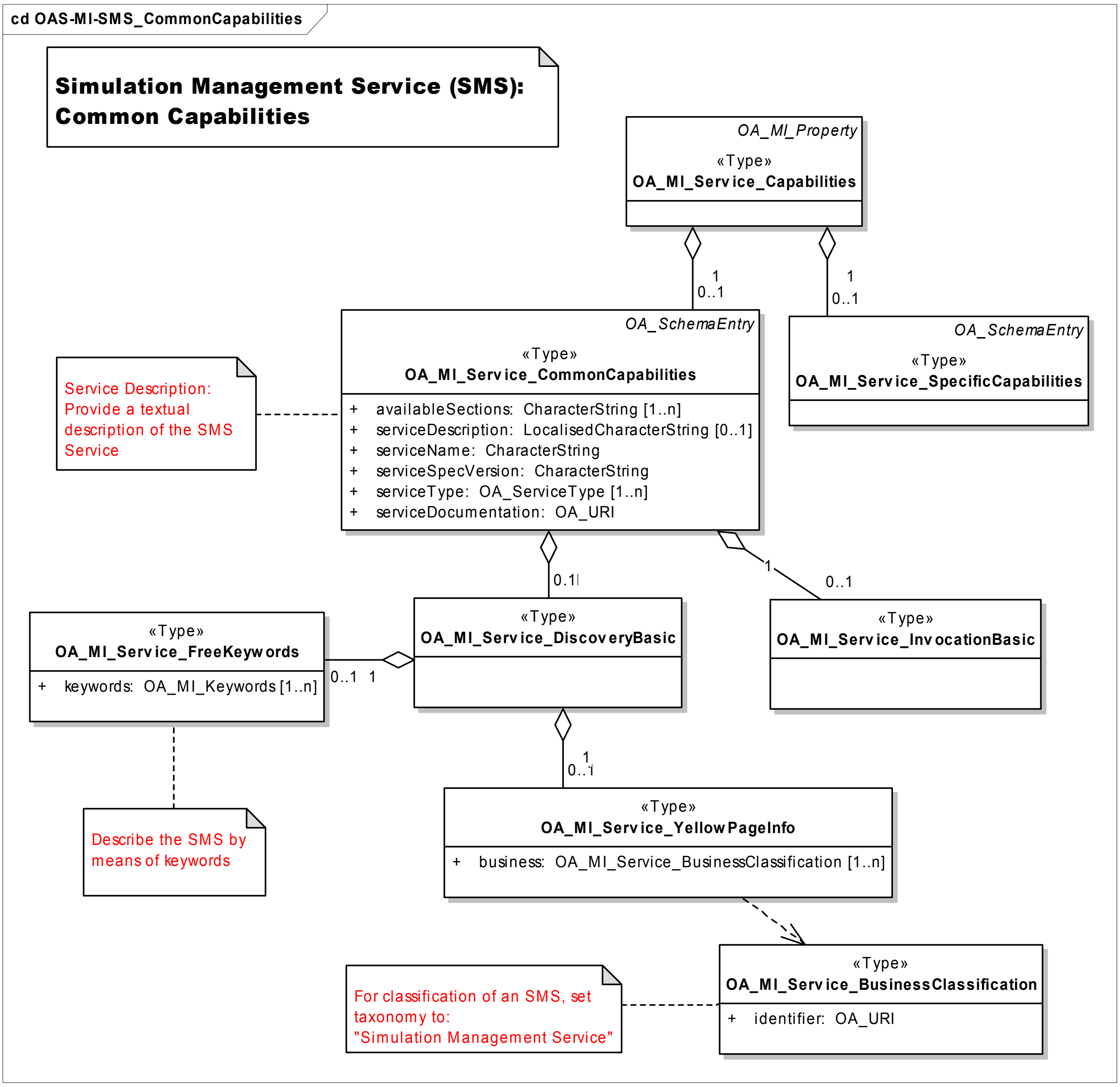

cd OAS-MI-SMS_CommonCapabilities
Simulation Management Service (SMS): Common Capabilities
OA_MI_Property
«Type»
OA_MI_Service_Capabilities
1
0..1
1
0..1
OA_SchemaEntry
«Type»
OA_MI_Service_CommonCapabilities
+ availableSections: CharacterString [1..n]
+ serviceDescription: LocalisedCharacterString [0..1]
+ serviceName: CharacterString
+ serviceSpecVersion: CharacterString
+ serviceType: OA_ServiceType [1..n]
+ serviceDocumentation: OA_URI
OA_SchemaEntry
«Type»
OA_MI_Service_SpecificCapabilities
Service Description: Provide a textual description of the SMS Service
0.1
1
0..1
«Type»
OA_MI_Service_FreeKeywords
+ keywords: OA_MI_Keywords [1..n]
0..1
1
«Type»
OA_MI_Service_DiscoveryBasic
«Type»
OA_MI_Service_InvocationBasic
1
0..1
Describe the SMS by means of keywords
«Type»
OA_MI_Service_YellowPageInfo
+ business: OA_MI_Service_BusinessClassification [1..n]
For classification of an SMS, set taxonomy to: "Simulation Management Service"
«Type»
OA_MI_Service_BusinessClassification
+ identifier: OA_URI

Figure 25: SMS Common Capabilities

B1.8.2.2.2 SMS Specific Capabilities

The specific capabilities of the SMS basically describe the available models, which can be executed by the SMS, and scenarios which have been defined for the SMS to run a specific simulation.

A description of a model comprises the model name, a textual description and a classification schema.

Moreover, the types of SDS needed for provision of input data are described here. They are described according to a service taxonomy for business classification (see section B1.8.2.3.1), and whether they are mandatory for the model or optional (i.e. they could enhance quality of the simulation but are not essentially needed).

A defined scenario is described by an identifier, which can be used to run the scenario again, and a textual description of the scenario. Moreover, results which have been computed in previous executions, can be stored here.

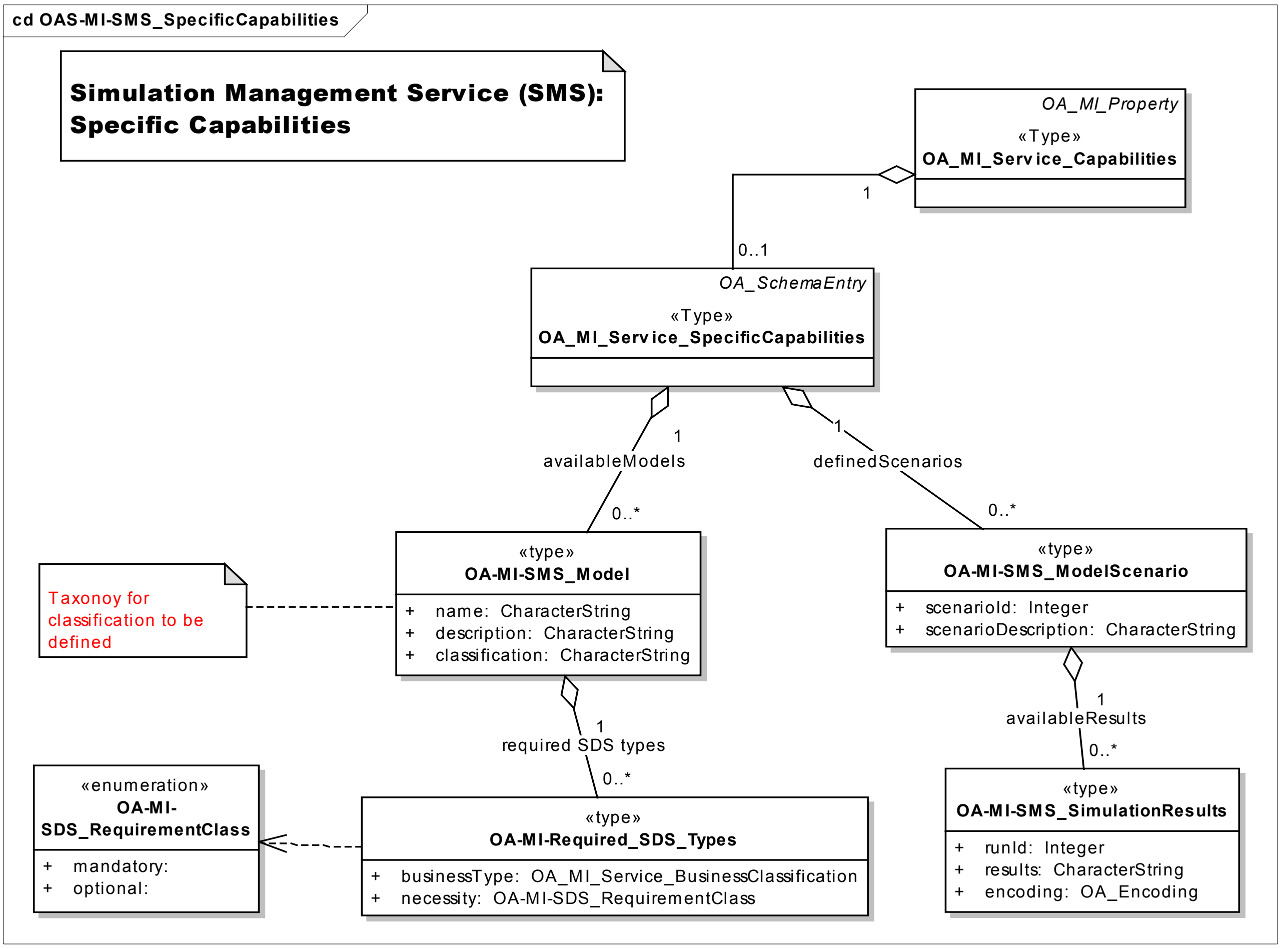

cd OAS-MI-SMS_SpecificCapabilities
Simulation Management Service (SMS): Specific Capabilities
OA_MI_Property
«Type»
OA_MI_Service_Capabilities
1
0..1
OA_SchemaEntry
«Type»
OA_MI_Service_SpecificCapabilities
1
availableModels
0..*
1
definedScenarios
0..*
Taxonoy for classification to be defined
«type»
OA-MI-SMS_Model
+ name: CharacterString
+ description: CharacterString
+ classification: CharacterString
«type»
OA-MI-SMS_ModelScenario
+ scenarioId: Integer
+ scenarioDescription: CharacterString
1
availableResults
0..*
1
required SDS types
0..*
«enumeration»
OA-MI-SDS_RequirementClass
+ mandatory:
+ optional:
«type»
OA-MI-Required_SDS_Types
+ businessType: OA_MI_Service_BusinessClassification
+ necessity: OA-MI-SDS_RequirementClass
«type»
OA-MI-SMS_SimulationResults
+ runId: Integer
+ results: CharacterString
+ encoding: OA_Encoding

Figure 26: SMS Specific Capabilities

B1.8.2.3 Meta-information about SDS

B1.8.2.3.1 SDS Common Capabilities

The following figure shows a selected profile of the common capabilities for SDS as described in the ORCHESTRA application schema.

Meta-information here, comprises a description by means of keywords which can be freely chosen, a textual description and a business classification according to the service taxonomy developed in ORCHESTRA SP2 (Workpackage 2.4.2).

Note: The service taxonomy is available, but it needs to be adapted, as it does not distinguish SMS and SDS.

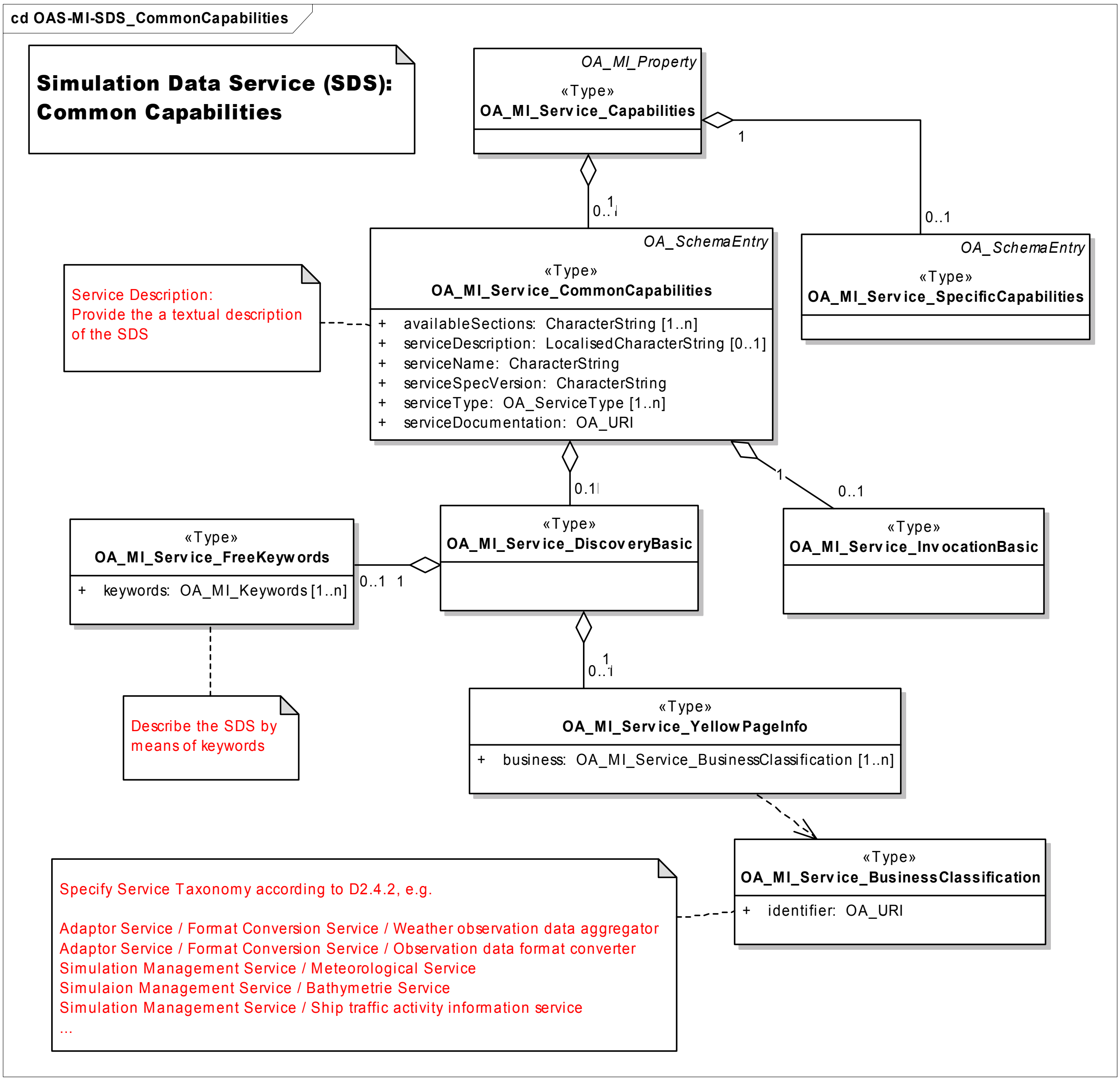

cd OAS-MI-SDS_CommonCapabilities
Simulation Data Service (SDS): Common Capabilities
OA_MI_Property
«Type»
OA_MI_Service_Capabilities
1
0..1
0..1
OA_SchemaEntry
«Type»
OA_MI_Service_CommonCapabilities
+ availableSections: CharacterString [1..n]
+ serviceDescription: LocalisedCharacterString [0..1]
+ serviceName: CharacterString
+ serviceSpecVersion: CharacterString
+ serviceType: OA_ServiceType [1..n]
+ serviceDocumentation: OA_URI
OA_SchemaEntry
«Type»
OA_MI_Service_SpecificCapabilities
Service Description:
Provide the a textual description of the SDS
0.1
1
0..1
«Type»
OA_MI_Service_FreeKeywords
+ keywords: OA_MI_Keywords [1..n]
0..1 1
«Type»
OA_MI_Service_DiscoveryBasic
«Type»
OA_MI_Service_InvocationBasic
Describe the SDS by means of keywords
1
0..1
«Type»
OA_MI_Service_YellowPageInfo
+ business: OA_MI_Service_BusinessClassification [1..n]
«Type»
OA_MI_Service_BusinessClassification
+ identifier: OA_URI
Specify Service Taxonomy according to D2.4.2, e.g.
Adaptor Service / Format Conversion Service / Weather observation data aggregator
Adaptor Service / Format Conversion Service / Observation data format converter
Simulation Management Service / Meteorological Service
Simulaion Management Service / Bathymetrie Service
Simulation Management Service / Ship traffic activity information service
...

Figure 27: SDS Common Capabilities

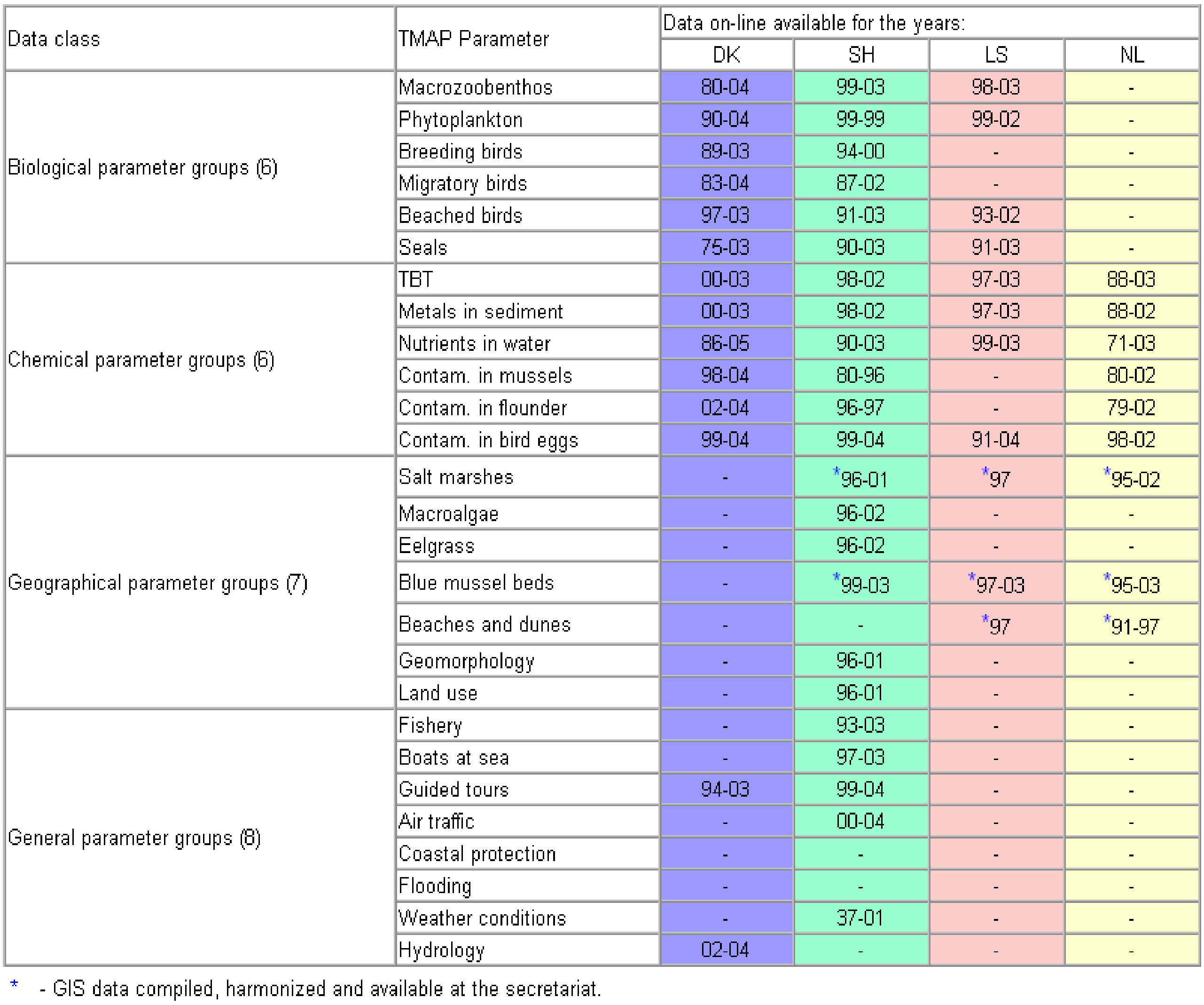
B1.8.2.3.2 An Example OAS for the TMAP Service

The specific capabilities of an SDS partially depend on its concrete service type. In order to illustrate how the meta-information should be specified for each service type, we select an example of a concrete service type:

We describe the meta-information provided by the database of the Trilateral Monitoring and Assessment Program (TMAP), as outlined under

http://www.waddensea-secretariat.org/TMAP/Data-Unit/Data.html

Here, we find the following figure:



Data class	TMAP Parameter	Data on-line available for the years:			
		DK	SH	LS	NL
Biological parameter groups (6)	Macrozoobenthos	80-04	99-03	98-03	-
	Phytoplankton	90-04	99-99	99-02	-
	Breeding birds	89-03	94-00	-	-
	Migratory birds	83-04	87-02	-	-
	Beached birds	97-03	91-03	93-02	-
	Seals	75-03	90-03	91-03	-
Chemical parameter groups (6)	TBT	00-03	98-02	97-03	88-03
	Metals in sediment	00-03	98-02	97-03	88-02
	Nutrients in water	86-05	90-03	99-03	71-03
	Contam. in mussels	98-04	80-96	-	80-02
	Contam. in flounder	02-04	96-97	-	79-02
	Contam. in bird eggs	99-04	99-04	91-04	98-02
Geographical parameter groups (7)	Salt marshes	-	*96-01	*97	*95-02
	Macroalgae	-	96-02	-	-
	Eelgrass	-	96-02	-	-
	Blue mussel beds	-	*99-03	*97-03	*95-03
	Beaches and dunes	-	-	*97	*91-97
	Geomorphology	-	96-01	-	-
	Land use	-	96-01	-	-
General parameter groups (8)	Fishery	-	93-03	-	-
	Boats at sea	-	97-03	-	-
	Guided tours	94-03	99-04	-	-
	Air traffic	-	00-04	-	-
	Coastal protection	-	-	-	-
	Flooding	-	-	-	-
	Weather conditions	-	37-01	-	-
	Hydrology	02-04	-	-	-

* - GIS data compiled, harmonized and available at the secretariat.

Figure 28: Meta-information of the TMAP Database

At first, we have to make assumptions based on interpretation of Figure 28 for our purposes. We have to distinguish between the "normal functionality" of the TMAP database, and the meta-information.

Considering the "normal functionality", we assume that the TMAP database provides access to parameters (see column "TMAP Parameter"). When we look at the specification of the ORCHESTRA Feature Access Service, see http://portal.opengeospatial.org/files/?artifact_id=12985 we see that this service is intended to provide an interface to databases.

This means, if we describe the information model of the TMAP database in an ORCHESTRA compliant form, we have to specify the features which can be accessed in an ORCHESTRA Application Schema (OAS). This is outlined in the following Figure 29:

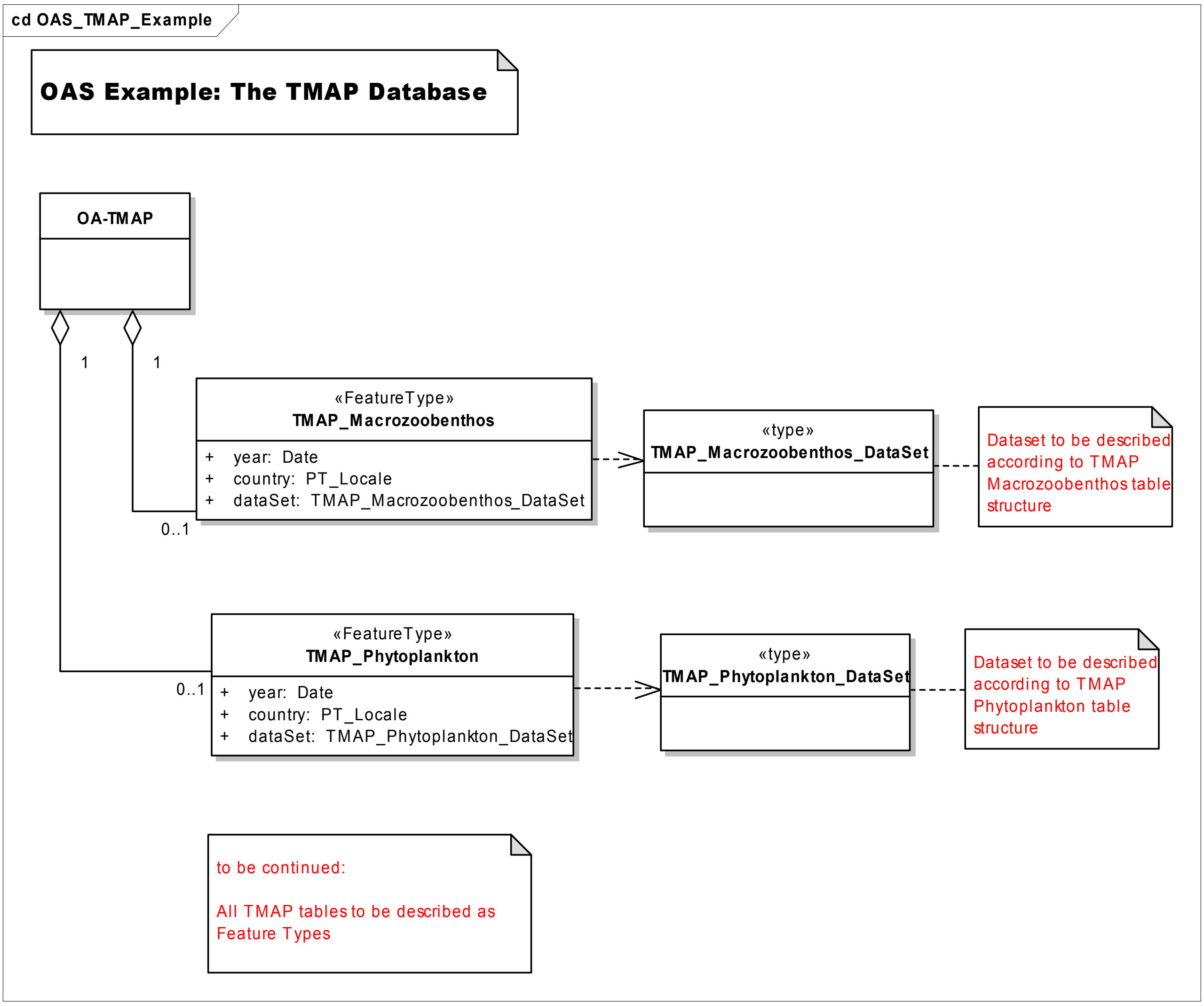

cd OAS_TMAP_Example
OAS Example: The TMAP Database
OA-TMAP
1
1
«FeatureType»
TMAP_Macrozoobenthos
+ year: Date
+ country: PT_Locale
+ dataSet: TMAP_Macrozoobenthos_DataSet
0..1
«type»
TMAP_Macrozoobenthos_DataSet
Dataset to be described according to TMAP Macrozoobenthos table structure
«FeatureType»
TMAP_Phytoplankton
0..1
+ year: Date
+ country: PT_Locale
+ dataSet: TMAP_Phytoplankton_DataSet
«type»
TMAP_Phytoplankton_DataSet
Dataset to be described according to TMAP Phytoplankton table structure
to be continued:
All TMAP tables to be described as Feature Types

Figure 29: Example Orchestra Application Schema (OAS) of the TMAP Database

In this OAS, we describe a feature for each of the TMAP parameters. An instance of the TMAP parameter feature type contains a dataset according to the internal structure of the TMAP database, which is not described on the TMAP web page and which is not needed to explain our description method. Moreover, we assume that a dataset has associated the year and the country, for which the data have been composed.

The FAS interface of the TMAP database allows retrieval of instances of these features. The implementation of the interface provides access to the source system by constructing database queries in the query language of the TMAP database.

The meta-information described in Figure 29 is discussed in the next section.

B1.8.2.3.3 SDS Specific Capabilities

The figure below illustrates the specific capabilities to be described for an SDS.

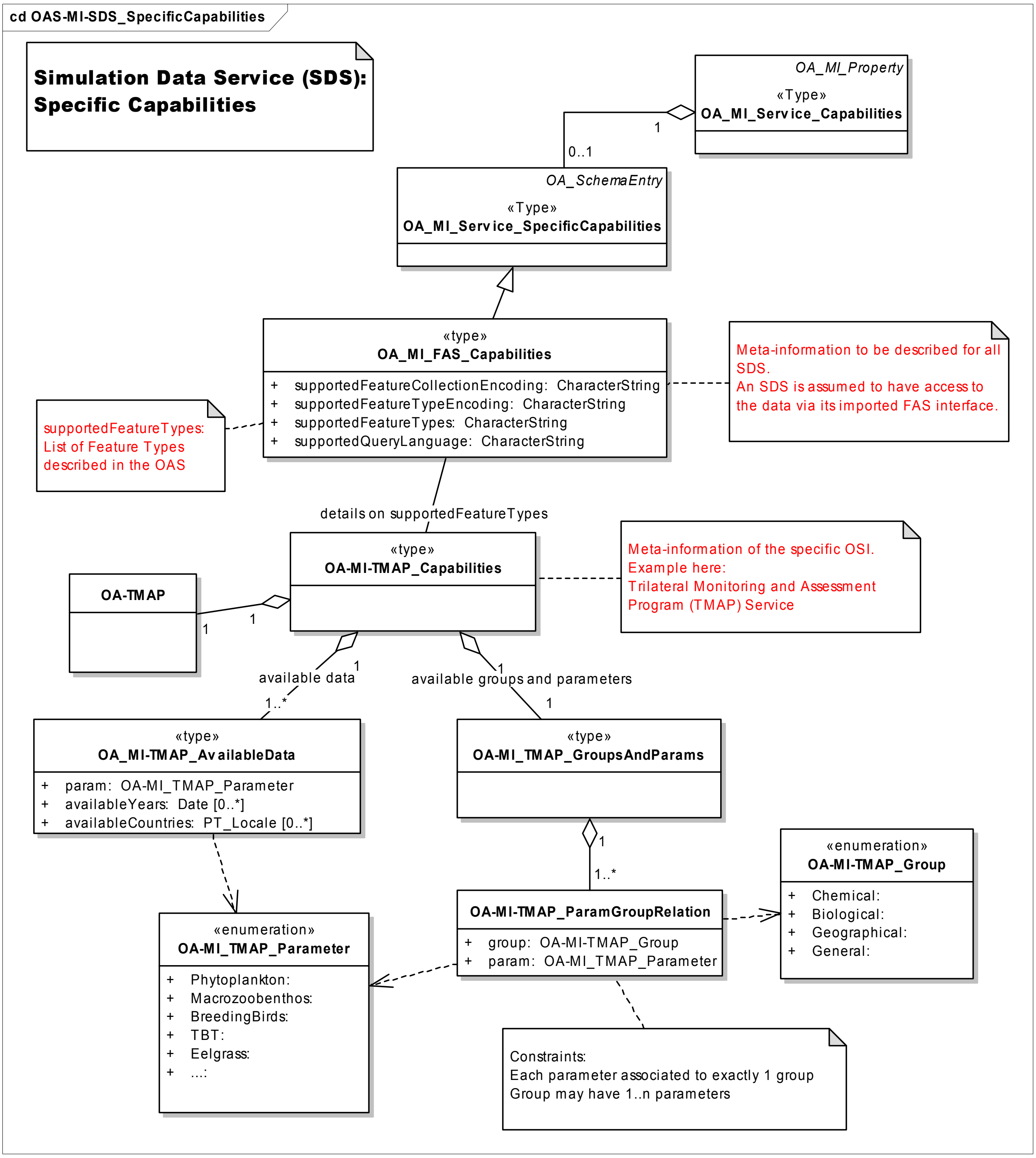

cd OAS-MI-SDS_SpecificCapabilities
Simulation Data Service (SDS): Specific Capabilities
OA_MI_Property
«Type»
OA_MI_Service_Capabilities
1
0..1
OA_SchemaEntry
«Type»
OA_MI_Service_SpecificCapabilities
«type»
OA_MI_FAS_Capabilities
+ supportedFeatureCollectionEncoding: CharacterString
+ supportedFeatureTypeEncoding: CharacterString
+ supportedFeatureTypes: CharacterString
+ supportedQueryLanguage: CharacterString
Meta-information to be described for all SDS.
An SDS is assumed to have access to the data via its imported FAS interface.
supportedFeatureTypes: List of Feature Types described in the OAS
details on supportedFeatureTypes
«type»
OA-MI-TMAP_Capabilities
Meta-information of the specific OSI.
Example here:
Trilateral Monitoring and Assessment Program (TMAP) Service
OA-TMAP
1
1
available data
1
1..*
available groups and parameters
1
1
«type»
OA_MI-TMAP_AvailableData
+ param: OA-MI_TMAP_Parameter
+ availableYears: Date [0..*]
+ availableCountries: PT_Locale [0..*]
«type»
OA-MI_TMAP_GroupsAndParams
1
1..*
«enumeration»
OA-MI-TMAP_Group
+ Chemical:
+ Biological:
+ Geographical:
+ General:
OA-MI-TMAP_ParamGroupRelation
+ group: OA-MI-TMAP_Group
+ param: OA-MI_TMAP_Parameter
«enumeration»
OA-MI_TMAP_Parameter
+ Phytoplankton:
+ Macrozoobenthos:
+ BreedingBirds:
+ TBT:
+ Eelgrass:
+ ...:
Constraints:
Each parameter associated to exactly 1 group
Group may have 1..n parameters

Figure 30: SDS Specific Capabilities (TMAP Example)

It is assumed that all SDS provide access through the interface of the Feature Access Service. All SDS therefore have meta-information as described in the specific capabilities of the FAS, especially the supported feature types.

Moreover, an instance of an SDS is described by the specific capabilities depending on the type of the service. Meta-information entered in the Catalogue is a part of the OAS-MI (ORCHESTRA Application Schema for Meta-Information) which can be obtained by calling the getCapabilities-operation of the service. The OAS-MI in the Catalogue may contain either the complete OAS (see section B1.8.2.3.2), a subset of the OAS or nothing from the OAS at all. This depends on the way applications intend to utilize meta-information (as illustrated in the following figure):

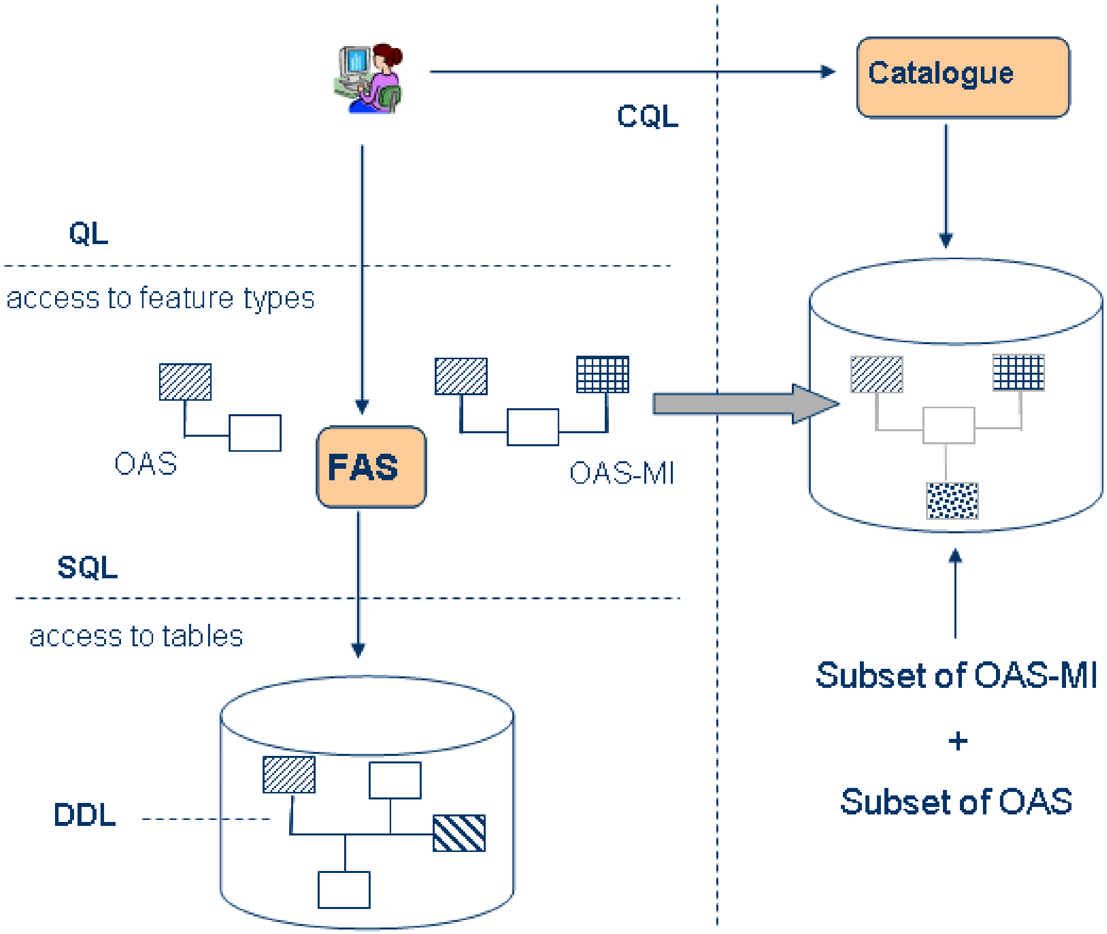


Figure 31: FAS Meta-Information in the Catalogue

The meta-information needed to mediate between the SDS and an SMS specific to the TMAP database is described by the type OA-MI-TMAP_Capabilities. It describes the availability of data specific to years and countries, and the structuring of the TMAP parameters into groups. In order to enable exploration the parameter types, the OA-MI Type is contained in the Catalogue as well; instances of the TMAP feature types are not needed to be contained in the Catalogue.

B1.8.2.4 Mapping to terraCatalog meta-information schema

In phase 1 of the BMT pilot, the implementation of the Catalogue is realized by using the terraCatalog of con terra Software, see http://www.conterra.de/de/software/sdi/terracatalog/index.shtm

The UML diagrams in the previous sections describe the ORCHESTRA conform structure of the meta-information of the services included in the Catalogue. This information shall be mapped to the ISO19115/19119 meta-information schema used by the terraCatalog. Since meta-information includes only meta-information about services, the UML diagrams need to be mapped to ISO19119. Meta-information about tightly coupled data sets with services can be described with ISO19115 and referenced as coupled dataset in ISO19119.

B1.8.2.5 Workflow of Meta-Information handling

This section describes the workflow of the meta-information handling in this pilot.

The following action needs to be done during preparation of the OSN:

Publish meta-information into the catalogue. This includes meta-information about simulation services (SMS) and meta-information about services (SDSs) providing data needed in SMSs.

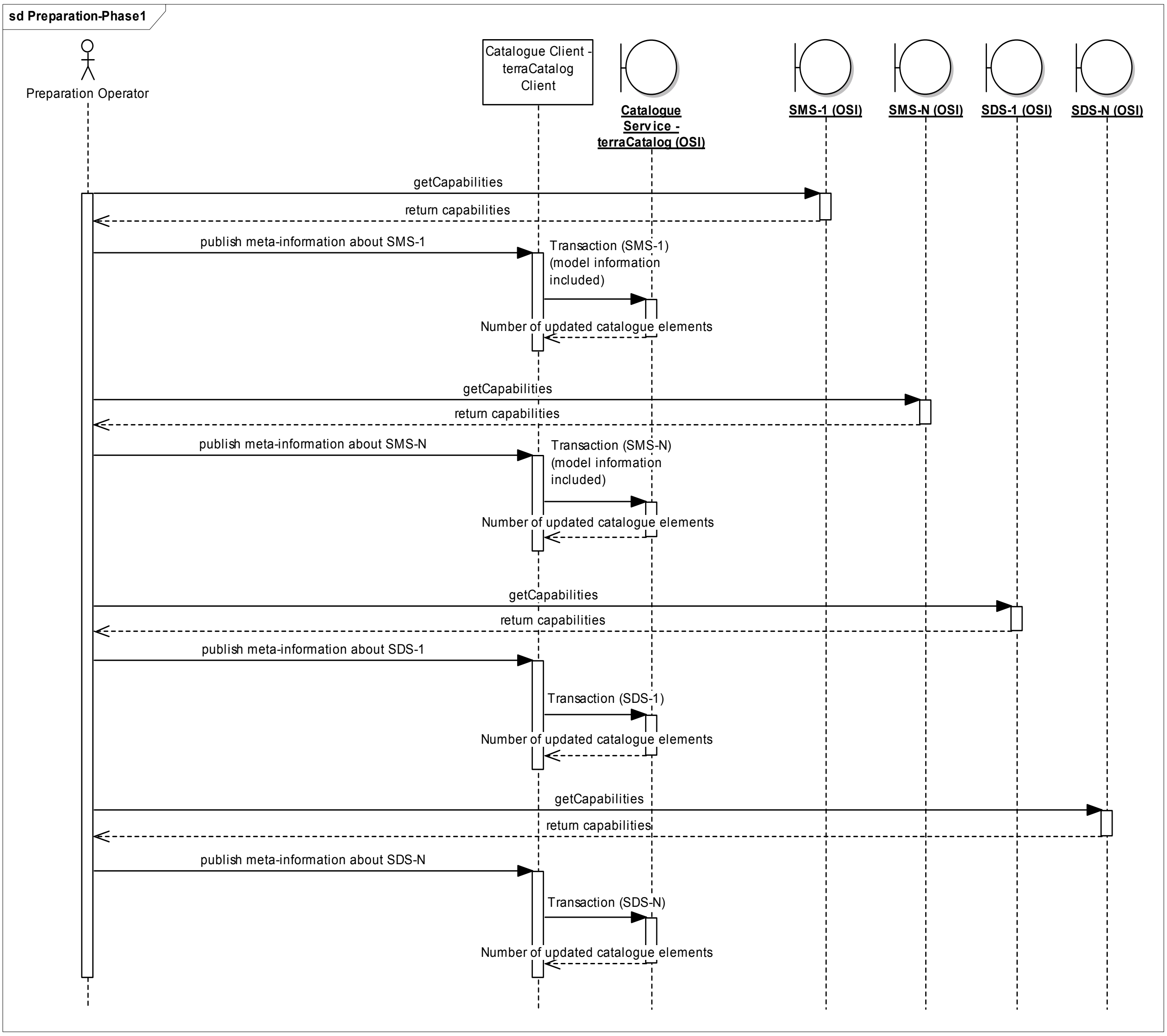

sd Preparation-Phase1
Preparation Operator
Catalogue Client - terraCatalog Client
Catalogue Service - terraCatalog (OSI)
SMS-1 (OSI)
SMS-N (OSI)
SDS-1 (OSI)
SDS-N (OSI)
getCapabilities
return capabilities
publish meta-information about SMS-1
Transaction (SMS-1) (model information included)
Number of updated catalogue elements
getCapabilities
return capabilities
publish meta-information about SMS-N
Transaction (SMS-N) (model information included)
Number of updated catalogue elements
getCapabilities
return capabilities
publish meta-information about SDS-1
Transaction (SDS-1)
Number of updated catalogue elements
getCapabilities
return capabilities
publish meta-information about SDS-N
Transaction (SDS-N)
Number of updated catalogue elements

Figure 32: Use case Preparation

The core actions for the creation of simulation scenarios are:

1. Use Case: DiscoverSMS

   Find in the catalogue simulation services (SMSs), which could be used for the simulation. The operator needs to decide, which available SMS is going to be used for the simulation.

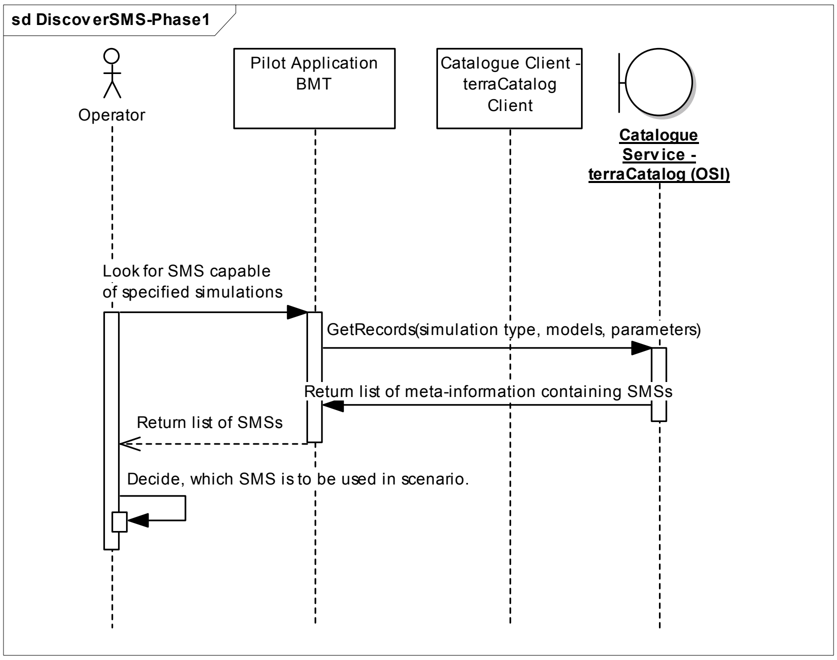

sd DiscoverSMS-Phase1
Operator
Pilot Application BMT
Catalogue Client - terraCatalog Client
Catalogue Service - terraCatalog (OSI)
Look for SMS capable of specified simulations
GetRecords(simulation type, models, parameters)
Return list of meta-information containing SMSs
Return list of SMSs
Decide, which SMS is to be used in scenario.

Figure 33: Use case DiscoverSMS

2. Use Case: ExploreSMS

   Explore the meta-information about the selected SMS and find in the catalogue information about used SDS types (e.g accessing services to shipping routes, vessel leaching rates, maritime data, cartographic information, etc.). The meta-information shall include information, which SDS types are mandatory and which are optional for the SMS.

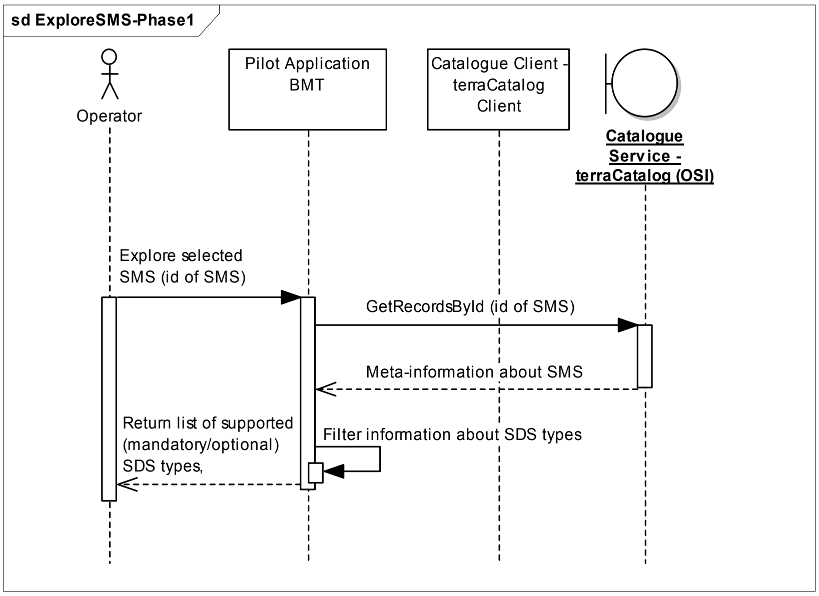

sd ExploreSMS-Phase1
Operator
Pilot Application BMT
Catalogue Client - terraCatalog Client
Catalogue Service - terraCatalog (OSI)
Explore selected SMS (id of SMS)
GetRecordsById (id of SMS)
Meta-information about SMS
Return list of supported (mandatory/optional) SDS types,
Filter information about SDS types

Figure 34: Use case ExploreSMS

3. Use Case: DiscoverSDSInstancesFromTypes

   Look for available SDS instances corresponding to needed SDS types in the catalogue.

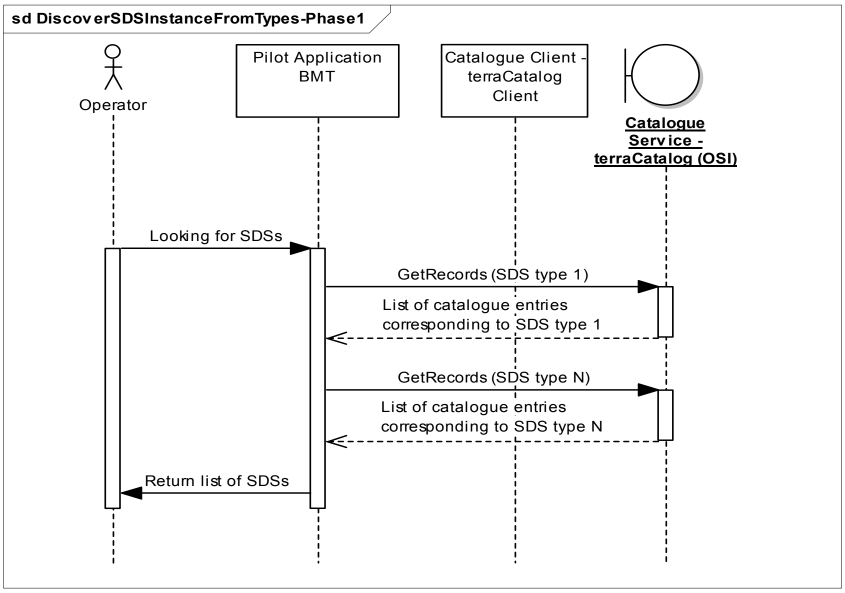

sd DiscoverSDSInstanceFromTypes-Phase1
Operator
Pilot Application BMT
Catalogue Client - terraCatalog Client
Catalogue Service - terraCatalog (OSI)
Looking for SDSs
GetRecords (SDS type 1)
List of catalogue entries corresponding to SDS type 1
GetRecords (SDS type N)
List of catalogue entries corresponding to SDS type N
Return list of SDSs

Figure 35: Use case DiscoverSDSInstanceFromTypes

4. Use Case: ExploreSDSs

   Explore the resulting SDS instances of step 3 and find in the catalogue the information about input data (parameters) needed for SDS instances. The operator needs to decide, which SDS instances are to be used for the scenario.

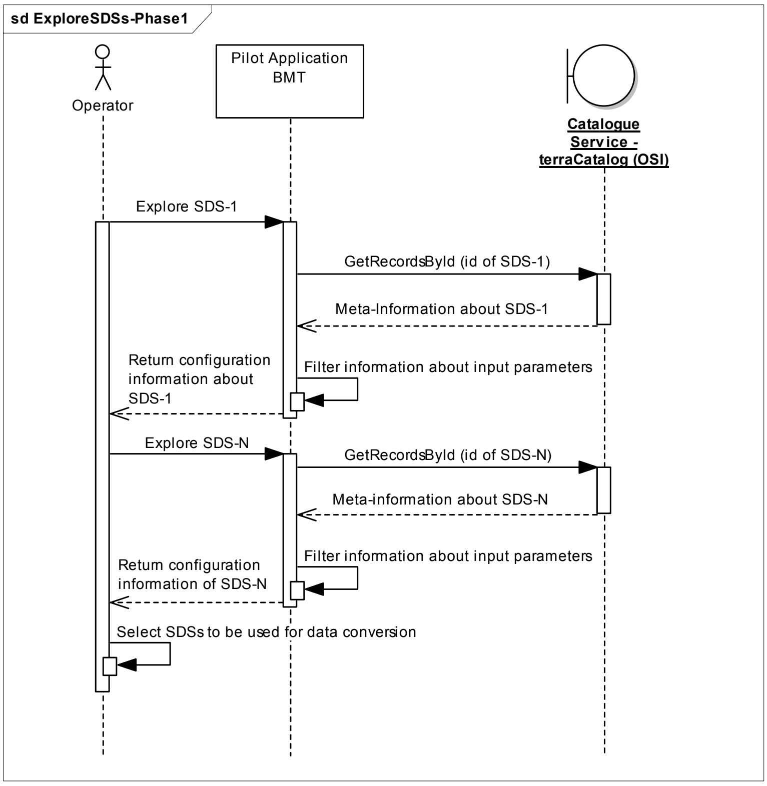

sd ExploreSDSs-Phase1
Operator
Pilot Application BMT
Catalogue Service - terraCatalog (OSI)
Explore SDS-1
GetRecordsById (id of SDS-1)
Meta-Information about SDS-1
Return configuration information about SDS-1
Filter information about input parameters
Explore SDS-N
GetRecordsById (id of SDS-N)
Meta-information about SDS-N
Return configuration information of SDS-N
Filter information about input parameters
Select SDSs to be used for data conversion

Figure 36: Use case ExploreSDSs

The handling of the meta-information is completed after this step. In subsequent steps, the simulation data are prepared for use through the SMS, the simulation run is configured, the simulation run is executed, the results are prepared for presentation, the resulting map and the simulation report is created.