OGC PROJECT DOCUMENT 05-107

TITLE:	**Reference Model for the ORCHESTRA Architecture (RM-OA)**	
AUTHOR:	**Name:**	**Thomas Usländer (Ed.)**
	Address:	**Fraunhofer IITB**
		Fraunhoferstr. 1
		76131 Karlsruhe
		GERMANY
	Phone:	**+49-721-6091-480**
	FAX:	**+49-721-6091-413**
	Email:	**uslaender@iitb.fraunhofer.de**
DATE:	**2005-10-14**	
CATEGORY:		

1 Background

ORCHESTRA IST FP6-511678

The ORCHESTRA project is an Integrated Project funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).

2 References

ORCHESTRA Deliverable D3.2.2 http://www.eu-orchestra.org

Reference Model for the ORCHESTRA Architecture V1.10

3 Proposal

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 its information and service viewpoints.

See below.

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)

D3.2.2 Draft Architecture Design

Date: 2005-10-14

Version: 1.10

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) D3.2.2: Draft Architecture Design	
Creator	Thomas Usländer, Fraunhofer IITB (Ed.) e-mail: uslaender@iitb.fraunhofer.de	
Subject	ORCHESTRA Architecture Design	
Description	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 its information and service viewpoints.	
Publisher	ORCHESTRA consortium	
Contributor	Bernard, Lars	Joint Research Centre - IES
	Bügel, Ulrich	Fraunhofer IITB
	Cooper, Michael	ETH Zürich
	Denzer, Ralf	Environmental Informatics Group
	Dihé, Pascal	Environmental Informatics Group
	Ecker, Severin	ARC Seibersdorf Research
	Friis-Christensen, Anders	Joint Research Centre - IES
	Frysinger, Steve	Environmental Informatics Group
	Goodwin, John	Ordnance Survey
	Güttler, Reiner	Environmental Informatics Group
	Hofmann, Thomas	Environmental Informatics Group
	Holt, Ian	Ordnance Survey
	Humer, Heinrich	ARC Seibersdorf Research
	Kunz, Wolfgang	Environmental Informatics Group
	Kutschera, Peter	ARC Seibersdorf Research
	Lorenzo, José	Atos Origin Spain
	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	ETHZ
	Usländer, Thomas	Fraunhofer IITB

Date	2005-10-14
Type	Text
Format	application/msword
Identifier	http://www.eu-orchestra.org/documents.shtml OGC Project Document 05-107
Source	Not applicable
Language	en-GB.
Relation	none
Coverage	Not applicable
Rights	© 2005 ORCHESTRA Consortium The ORCHESTRA project is an Integrated Project funded under the FP6 (Sixth Framework Programme) of the European Commission in the research programme Information Society Technologies (IST).
Deliverable number	D3.2.2
Audience	[✓ public] ☐ restricted ☐ internal

Revision History

Version	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

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 integrated project ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) is one of the projects that recently started in this area (IST Integrated Project no. 511678). The overall goal of ORCHESTRA 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 software 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 is a self-contained document providing a specification framework for system architects, information modellers and system developers when designing service networks taking into account relevant standards from ISO, OGC and W3C.

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

- The RM-OA Engineering 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 dealt with by the ORCHESTRA Architecture, including their thematic, spatial, temporal characteristics as well as their meta-information. The basic informational unit is the concept of a feature as an abstraction of a real world phenomenon that may but need not have spatial characteristics. 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 ISO/IEC 10746-1 called Computational Viewpoint) specifies ORCHESTRA Architecture Services that support the syntactical and semantic interoperability between systems as well as the administration of ORCHESTRA service networks. The RM-OA distinguishes between OA Info-Structure services that are indispensable for the operation of an ORCHESTRA Service Network and OA Support Services that facilitate the operation of an ORCHESTRA Service Network. The current RM-OA version provides textual service descriptions according to a common service description framework. Furthermore, the RM-OA contains an initial description of so-called ORCHESTRA Thematic Support services that facilitate the development of thematic functionality such as the processing of statistical data.
- The ORCHESTRA Architecture is defined to be the combined specification of the RM-OA Information and Service Viewpoints on an abstract, i.e. platform-neutral level in UML. An RM-OA annex will contain the UML specification of the ORCHESTRA Architecture Services and default application-schemas for meta-information for an initial list of "purposes" (e.g. discovery).
- The RM-OA Engineering and Technology viewpoints yield the mapping of the application schemas and service specifications to service infrastructures (e.g. W3C Web Services). Here, the RM-OA just delivers the reference model. These viewpoint specifications will lead to dedicated ORCHESTRA Implementation Specifications in addition to the RM-OA.

2 ORCHESTRA-specific Summary

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 (2004)). The DoW is the technical part of the ORCHESTRA contract with the European Commission.

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

- To specify requirements which an ORCHESTRA Architecture (OA) for risk management needs to address.
- To design a draft OA, 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.
- In a further step, to refine the OA during a second iteration, leading to the second version of services.
- To establish an open communication with the WIN project, so that WIN and ORCHESTRA services can all run in the OA and are interoperable.

The work package is structure 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, co-ordination, 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 result of this task leads to the deliverable D3.2.1 “High Level Requirements Specification”. This deliverable corresponds to the Annex A.1 (see section 11) and Annex A.2 (see section 12) of the current version V1.10 of the RM-OA.

- 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 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). This draft architecture design will be discussed on a regular basis by the WIN project.

  The result of this task leads to the deliverable D3.2.2: “Draft Architecture Design”. This deliverable is identical with the current version V1.10 of the RM-OA.

- 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 and the partners from the WIN project, a refined architecture design will be elaborated until month 18.

  The result of this task leads to the deliverable D3.2.3: “Refined Architecture Design”. This deliverable will correspond to a future version of the RM-OA.

In a future version of the RM-OA, it is intended to include further results of the ORCHESTRA sub-project 3 into the RM-OA, possibly as annexes:

- The Specification of the Meta-Information Model (deliverable D3.3.2 of the work package 3.3 “Meta-Information Model”).
- The specification of the ORCHESTRA Services (deliverables D3.4.1, D3.4.2 and D3.4.3 of work

package 3.4 “Core service specification”).

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

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

  These user requirements constitute the major source of requirements for the RM-OA. Early results have already been included in the version V1.10 of the RM-OA. Refined results will be continuously incorporated into the RM-OA and mapped to system requirements according to the iterative process of the RM-OA design.

- The DoW as a basic 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.

The RM-OA is based on standards wherever possible. If standards are used but (slightly) changed this is highlighted by the ORCHESTRA logo near the text. Furthermore, all deviations from a standard are summarised in section 10.

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 its 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 section 12 “Annex A.2: Requirements for the OSN and the OA”.
- Section 7 “Design 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” describes the services that support the syntactical and semantic interoperability between 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 “Summary of Deviations from Standards” lists the major aspects where the RM-OA specification deviates from standards.

3.2 Intended Audience

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

3.3 References

3.3.1 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 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

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)

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

GMES (2004). Global Monitoring for Environment and Security (GMES): Final Report for the GMES Initial Period (2001-2003) http://www.gmes.info/action_plan/index.html

ORCH-D2.1.3 (2005). Report on Requirements for Components of End-User Applications. Deliverable D2.1.3 Integrated Project 511678 ORCHESTRA. Editor: BRGM. Revision [draft]. 3 December 2004

ORCH-D2.3.1 (2005). Domain and Task Ontologies. Deliverable D2.3.1 Integrated Project 511678 ORCHESTRA. Editor: Ordnance Survey. Revision 0.3. May 2005

ORCH-D2.4.1 (2005). Report identifying common service requirements. Deliverable D2.4.1 Integrated Project 511678 ORCHESTRA. Editor: DATAMAT. Revision [final]. 13 September 2005

ORCH-D3.3.1 (2005). High level Conceptual Meta-information Model. Draft deliverable D3.1.1 Integrated Project 511678 ORCHESTRA. Editor: ARCS. Revision 1.0. May 2005

ORCH-DoW (2004). Integrated Project 511678 ORCHESTRA: "Annex 1 – Description of Work". 6th Framework Programme Priority 2.3.2.9 Improving Risk Management. 20 July 2004

ORCH-ReqTrace (2005). Integrated Project 511678 ORCHESTRA: *Documentation of the Requirements for Traceability (to be provided)*

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 (2003a) Open Geospatial Consortium Doc. No. 03-022r3. Recommendation Paper "Observations and Measurements Version 0.9.2, 2003-02-04, http://portal.opengeospatial.org/files/?artifact_id=1324

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

Wytzisk, A. (2003): Interoperable Geoinformations- und Simulationsdienste auf Basis internationaler Standards. PhD Thesis University of Münster

Young, I.T., Gerbrands, J.J., van Vliet, L.J. Fundamentals of Image Processing. http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip.html

4 Glossary

The glossary contains a list of terms that are part of a coherent framework defined in the RM-OA. The relationships between the terms are defined in the concepts and relationships sections of the RM-OA, in particular in the descriptions of the information and service viewpoints.

4.1 Abbreviations

CSL	Conceptual Schema Language
DIS	Draft International Standard
DoW	ORCHESTRA Description of Work
EC	European Commission
ESA	European Space Agency
ESDI	European Spatial Data Infrastructure
GFM	General Feature Model
GMES	Global Monitoring for Environment and Security
HCI	Human-Computer Interaction
INSPIRE	Infrastructure for Spatial Information in Europe
IS	International Standard
ISO	International Standardization Organisation
IST	Information Society Technology
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	Open Advanced System for Improved Crisis Management
OGC	Open Geospatial Consortium
OIS	ORCHESTRA Implementation Specification
OMM	ORCHERSTRA Meta-model
ORCHESTRA	Open Architecture and Spatial Data Infrastructure for Risk Management
OSN	ORCHESTRA Service Network
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
W3C	World Wide Web Consortium
WIN	Wide Information Network for Risk Management

4.2 Terms and definitions

Access control

See Authentication and Authorisation.

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

Process of verifying that a potential partner in a conversation is capable of representing a person or organization.

Authorisation [Security Taxonomy and Glossary; http://www.garlic.com/~lynn/secure.htm]

Process of determining, by evaluating applicable access control information, whether a subject is allowed to have the specified types of access to a particular resource. Usually, authorisation is in the context of authentication. Once a subject is authenticated, it may be authorized 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

See 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/FDIS 19123, ISO/DIS 19131]

A feature that acts as a function to return values from its range for any direct position within its spatial, temporal or spatiotemporal domain

[The OpenGIS™ Abstract Specification Topic 6: The Coverage Type and its Subtypes Version 6 http://www.opengis.org/techno/abstract/00-106.pdf]

A feature that associates positions within a bounded space (its spatiotemporal domain) to feature attribute values (its range). GIS coverages (including the special case of Earth images) are multi-(often two-) dimensional metaphors for phenomena found on or near a portion of the Earth's surface. A coverage can consist of a set of features or feature collections. Earth images are seen as Grid Coverages that contain features whose geometries are of type "set of cells" or "set of pixels" (surfaces).

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

The 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 and information infrastructure.

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.

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 or worlds of man's thoughts. 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]

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

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].

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 [derived from (ORCH-D3.3.1 2005)]

Descriptive information about resources in the universe of discourse. The structure of the meta-information is given by a conceptual model (the meta-information model) that depends on a particular purpose.

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 these services play an indispensable role in the operation of an OSN. The requirements are defined in OSN conformance clauses.

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. No OSN conformance clauses are specified for OA Support Services.

Ontology (derived from ORCH-D2.3.1 2005)

Formal representation of the knowledge associated with a particular subject area (domain) or task. The ontology is specified by a shared vocabulary, that is, the type of objects and/or concepts that exist, their properties and relations. Its ultimate purpose is to enable machine understanding which in turn provides the potential for data and service interoperability.

Note: One major qualitative property that distinguishes an ontology from a conceptual schema is that an ontology specification is shared in a dedicated user community.

Open Architecture (Powell 1991)

An Open Architecture is an architecture which 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.

ORCHESTRA Architecture

The ORCHESTRA Architecture (OA) is an 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)

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

ORCHESTRA Application Schema (OAS)

Formal specification of the feature types, their properties and associations which are relevant for a specific information model in an ORCHESTRA Service Network.

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)

Collection 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 service infrastructure (e.g. W3C Web Services).

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 <<metaClasses> and associated rules for their instantiation in an ORCHESTRA Application Schema.

ORCHESTRA Reference Model

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

ORCHESTRA Service

Service offered by an ORCHESTRA Service Network. ORCHESTRA Services are functionally classified in ORCHESTRA Architecture Services (OA Services) and ORCHESTRA Thematic Services (OT Services).

ORCHESTRA Service Network

Composite set of networked hardware resources 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 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 Thematic Service (OT Service)

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.

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.

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. This information ubiquity is the beginning phase of a truly information-driven organization.

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; http://www.opengis.org/docs/02-112.pdf]

A computation performed by a software entity on one side of an interface in response to a request made by a software entity on the other side of the interface. A collection of operations, accessible through an interface, that allows a user to evoke a behaviour of value to the user.

Service Viewpoint

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

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.opengeospatial.org/resources/?page=glossary]

A comprehensive package of consensus and initiatives required to enable complete provision of data, access and privacy within the territory of the designated infrastructure.

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 and integrators that conceive and develop applications for an application domain.

Technology viewpoint

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

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

A user of the ORCHESTRA Architecture is a software vendor, a service provider, or other party engaged in offering data and software components that are compliant with ORCHESTRA specifications.

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.

5 Process of the ORCHESTRA Architectural Design

5.1 Overview

The ORCHESTRA Architecture is being designed in an iterative way recognising the fact that both the requirements of the system and end users and 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 and the design phase of the architecture is foreseen (see Figure 1).

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.

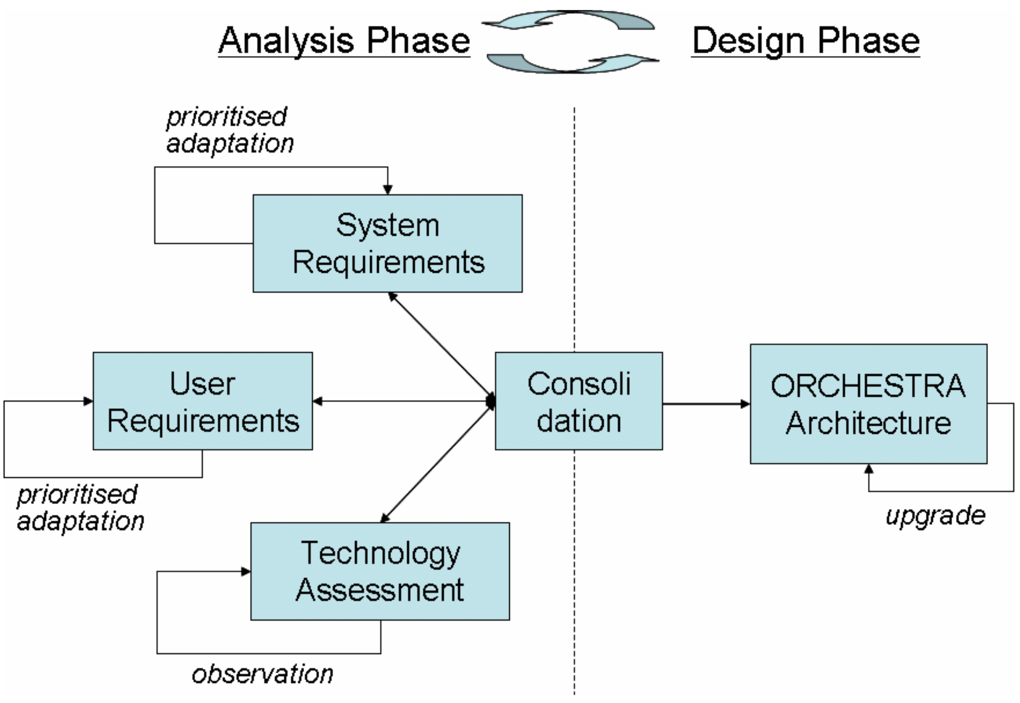


Figure 1: Dynamic ORCHESTRA Architectural Process

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 2004).

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 Annex A.2, section 11) 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.

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 the user requirements 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 requirements in a given environment ?

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 the input factors of the ORCHESTRA Architecture naturally leads to an iterative architectural design process. Various but controlled upgrades of the ORCHESTRA Architecture will be required to adapt the architecture to the changing needs.

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 like

- 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 1:

- The second column of Table 1 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 1 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 1 provides a concrete example of what will be defined in the respective viewpoint.

View-point Name	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 statistical 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 services that support the syntactical and semantic interoperability between source systems and the development of ORCHESTRA Applications.	UML specification of the statistical processing service.
Technology	Concerned with the choice of technology to support system distribution.	Focuses on the choice of technology.	Specifies the technological choices of the service infrastructure and the operational issues of the infrastructure.	Decision to map the generic specification to W3C Web Services and UDDI.
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 service and information infrastructure.	Mapping of the UML specification to WSDL.

Table 1: 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 ORHCESTRA 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 allow to adapt the architecture to changing or refined needs as specified in the enterprise viewpoint are not shown in Figure 2.

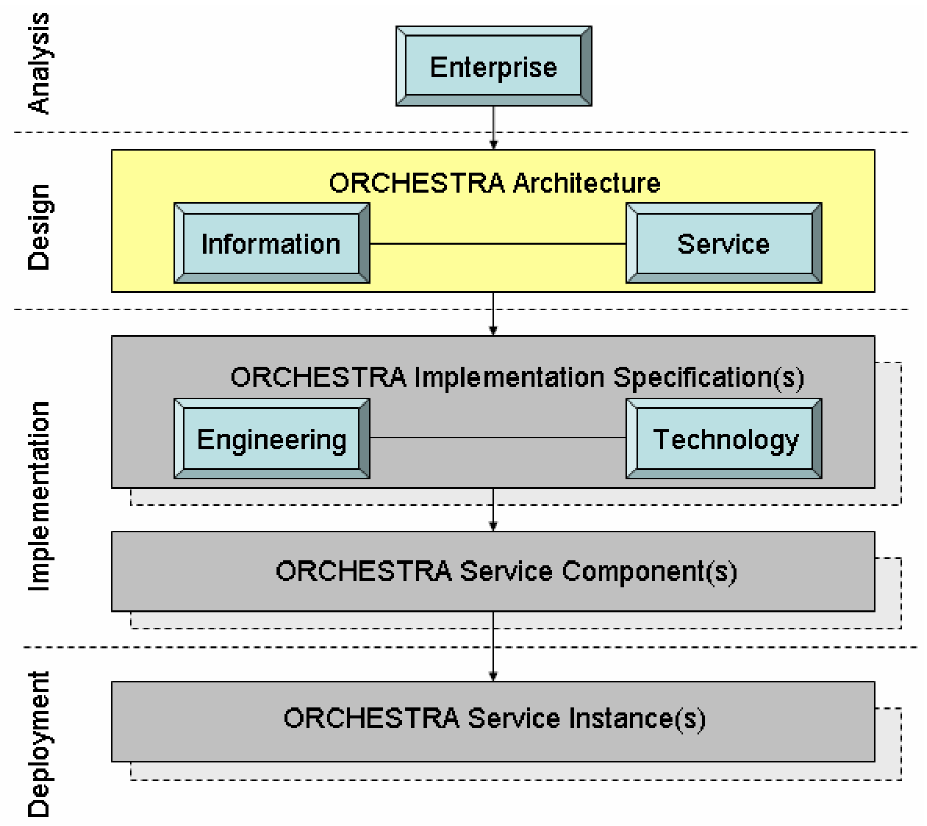


Figure 2: The ORCHESTRA Reference Model

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 UML). The ORCHESTRA Architecture is specified as part of the design phase and encompasses the harmonised specification of the Information and Service viewpoint resulting from requirements of the Enterprise viewpoint.

The ORCHESTRA Architecture does not cover the Engineering and Technology viewpoints.

5.3.2 The ORCHESTRA Implementation Specification

The aspects of the Engineering and Technology viewpoints are combined in a dedicated specification step that may be carried out multiple times. Each step represents one mapping of the OA (i.e. the Information and Service Viewpoint specification) to a specific service infrastructure (e.g. W3C Web Services) and leads to a platform-specific ORCHESTRA Implementation Specification (OIS).

Thus, an OIS contains platform-specific specifications of ORCHESTRA information models and ORCHESTRA services. This means concretely that OA information models expressed in UML have to be mapped to a schema language (e.g. XML or EXPRESS) that fits to the specific service infrastructure. Likewise, the UML specifications of the ORCHESTRA Services have to be mapped to a service description language (e.g. WSDL) that fits to this infrastructure, too. These mapping processes may be done manually of performed (semi-)automatically by a tool.

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

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 Engineering and Technology Viewpoints are not specified as part of the RM-OA document. They are considered in the documentation of an OIS.

The implementation phase encompasses the ORCHESTRA Implementation Specifications and their implementation in ORCHESTRA Service Components (OSC). An OSC is a component that provides an external interface of an ORCHESTRA Service according to an OIS.

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).

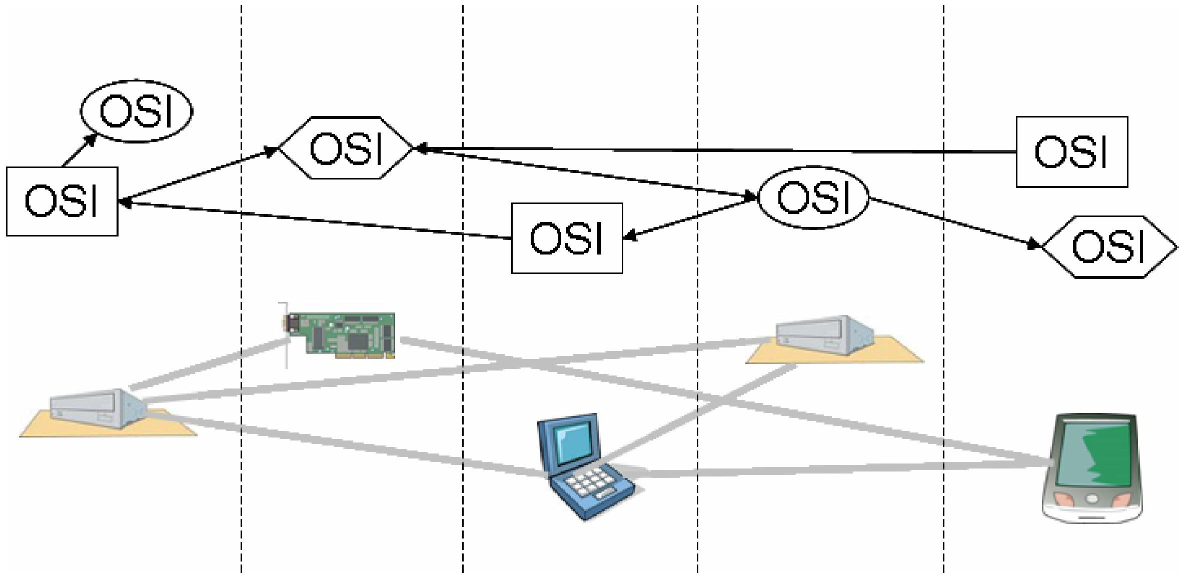


Figure 3: Deployment of ORCHESTRA Service Instance in an ORCHESTRA Service Network

The set of ORCHESTRA Service Instances connected through a communication network is called an ORCHESTRA Service Network (OSN). An OSN thus comprises the composite set of networked hardware resources 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 resources (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. The distributed software components that constitute an ORCHESTRA Application may contain OSIs or not.

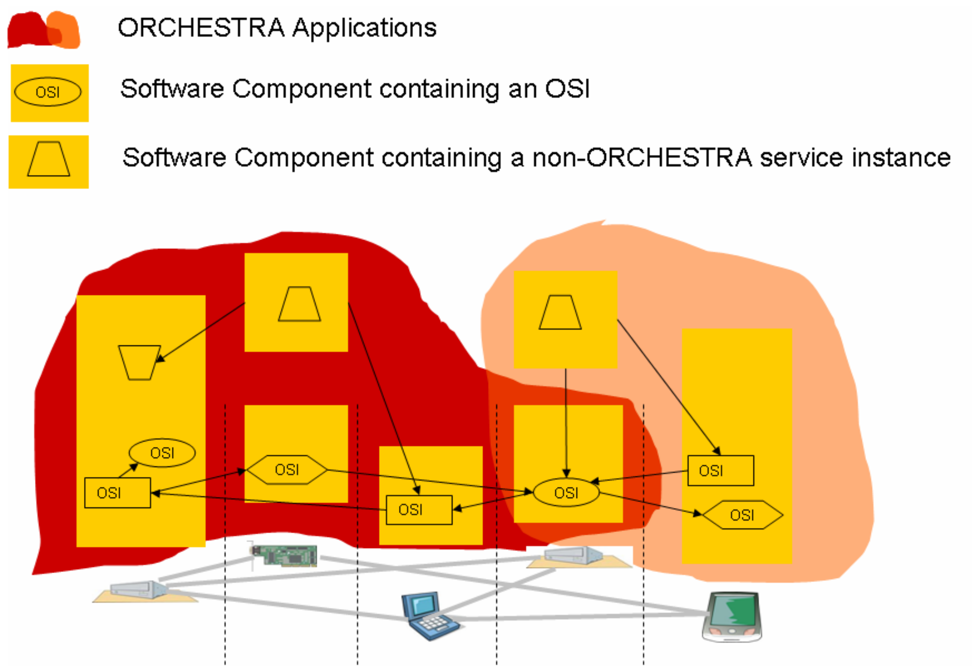

ORCHESTRA Applications
OSI
Software Component containing an OSI
Software Component containing a non-ORCHESTRA service instance
OSI
OSI
OSI
OSI
OSI
OSI
OSI

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

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.

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).

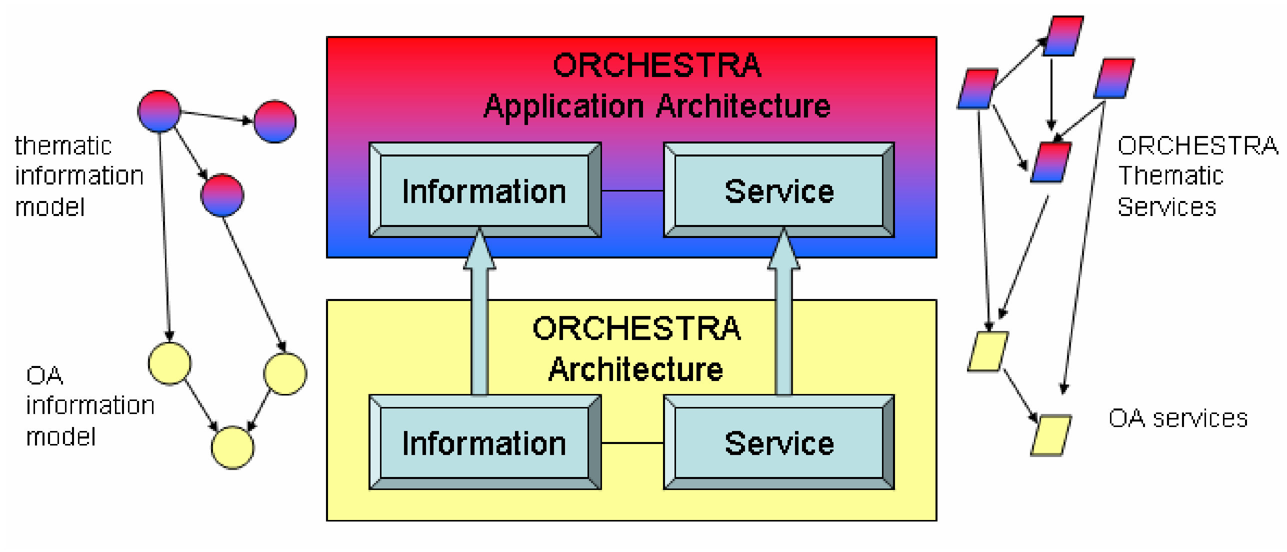


Figure 5: ORCHESTRA Application Architecture

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

Note: The process to identify on the conceptual level the most 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 a thematic information models and a thematic service and 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 (ISO 19119:2005)

The Topic 12 of the OpenGIS Abstract Specification: The OpenGIS Service Architecture 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).

- The IT Services are general services in a distributed computing environment, like 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/TS 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 to 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 service taxonomy is considered as meta-information for the service. According to the ORCHESTRA handling of meta-information (see section 8.5.1), the adequacy of this service taxonomy has therefore to be reconsidered when defining purpose-oriented meta-information for services.

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

The ORCHESTRA Architecture is a service-oriented architecture. Furthermore, with respect to ISO 19119:2005, section 7.6 it shall comply with the characteristics of a “simple service architecture”. A simple service 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, operations of an interface should separate the control of the service from the access to the data resulting from a service. A client should have the option of receiving just the status of an operation and separately the data should be accessible 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.

- 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 (see OA V2.x, see section 6.2.3), this meta-information will stick to providing syntactical information like the operation signatures, the provider name and a textual service description. However, in a second step (see OA V3.x, see section 6.2.3) meta-information that includes semantic concepts for services will be provided.

- Adequate hardware

  The ORCHESTRA services are software implementations (OSCs) running 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 envisaged 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 cooperate with two other European integrated projects:

- OASIS: Open Advanced System for crisIS management (IST IP 4677)
- WIN: Wide Information Network for Risk Management (IST IP 511481)

These projects face the common 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, Mr. Guy Weets) 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. This series of meetings is on-going.

With respect to the relationship between ORCHESTRA, OASIS and WIN common architectural principles of an open info-structure have been discussed and are currently finalised in a white paper (see also section 6.2.2.3). As soon as a draft of this white paper is agreed between the authors, all ORCHESTRA partners will have to state whether they can fully support this white paper or not. It will be important that the General Assembly of ORCHESTRA agrees on this issue.

6.2.2.2 Requirements of the INSPIRE Relationship

Following three years of intensive collaboration with Member States experts and stakeholder consultation, the European Commission has adopted on the July 2004 a proposal for a Directive establishing an infrastructure for spatial information in the Community (COM 2004).

The adoption of the proposal marks an important step on the way forward to a European-wide legislative framework that helps in achieving a European Spatial Data Infrastructure (ESDI). This proposal does not only address policy related issues concerning the development of an ESDI but also dedicates three chapters to the technical requirements that have to be fulfilled by the member states to establish the ESDI. These three chapters are on metadata, interoperability of spatial data sets and services, and network services. Under these chapters the proposal list general requirements on these issues as well as it formulates the requirement to adopt appropriate implementing rules.

During the INSPIRE preparatory phase (2005-2006) (Dufourmont, Annoni, De Groof 2004) the ORCHESTRA project will provide input towards the drafting as well as the piloting of the INSPIRE implementing rules in the risk management domain. The first input can be expected on the topic of the INSPIRE Network Services.

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

- Upload Services (for meta-data and spatial data)
- Discovery services
- View services
- Download services
- Transformation services
- “Invoke spatial data services” services

Following the INSPIRE proposal, the network services will be established in a distributed environment by the Member States and they will be accessible via the European Geo-Portal. The definition of appropriate technical specifications requires that considered interface specifications are mature and proved by implementations and operational usage including performance consideration. The first tasks in this will have to provide a more detailed description as a basis for the common understanding about these network services. It will benefit from the definition of a software architecture describing the collaboration of the network services and the Geo-Portal addressing for instance:

- General architectural model (how to invoke network services, functionalities of Geo-portals, …)
- Multilingualism
- Conformance
- Performance
- Confidentiality

6.2.2.3 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 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 and 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 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.

Up to date, the relationship between ORCHESTRA and GMES is formally undefined. Potential contributions to GMES are 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 ORCHESTRA's work plan and budget. 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 shall consider from the beginning a three step approach:

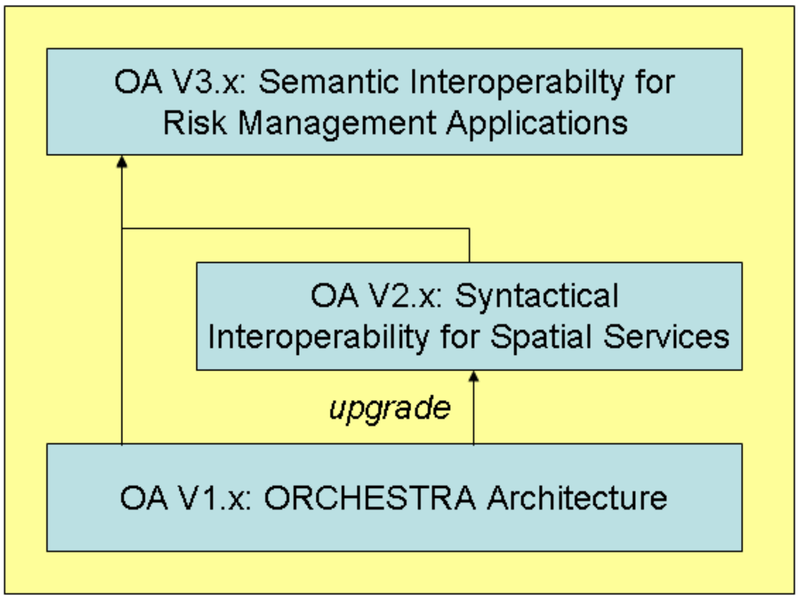

OA V3.x: Semantic Interoperabilty for Risk Management Applications
OA V2.x: Syntactical Interoperability for Spatial Services
upgrade
OA V1.x: ORCHESTRA Architecture

Figure 6: The evolution of the ORCHESTRA Architecture

- In OA Version 1.x, the ORCHESTRA Architecture shall be conceived. The ORCHESTRA Architecture shall provide a common view on how to harmonise the requirements for syntactical and semantic service and data interoperability including their thematic, temporal and spatial characteristics.
- In OA Version 2.x, the focus is on refining the OA V1 in terms of service specifications for syntactical interoperability in the spatial domain. This version shall link to the INSPIRE requirements for network services as given in section 6.2.2.2. OA Version 2.x
- In OA Version 3.x, the focus is on extending and refining OA V1 and OA V2 in terms of service specifications for semantic interoperability in the risk management domain.

Note: None of these OA versions include 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 form of corresponding OISs.

6.3 Architectural Requirements for the OSN Design

In the following, architectural requirements for OA and an OSN are specified. They have been derived in 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 OA,
3. deriving from that *key system requirements* and
4. finally *architectural principals*.

Here, just the architectural principals are briefly explained in terms of architectural requirements.

6.3.1 Rigorous Definition and Use of Concepts and Standards

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

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.

6.3.3 Technology Independence

The OA shall 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 OA 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.

6.3.4 Evolutionary Development - Design for Change

The OA and an OSN shall be designed to evolve, ie. it shall be possible to develop and deploy the system in an evolutionary way. The OA 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.

6.3.5 Component Architecture Independence

The OA shall be designed such that an OSN and source systems (ie. existing information systems and information networks) are architecturally decoupled. This means that the OA shall not impose any architectural patterns on source systems for the purpose of them collaborating in an OSN, and no source system shall impose architectural patterns 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 context, 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.

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.

7 Design of the ORCHESTRA Architecture

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 thus contributes to improve the syntactical and the semantic interoperability of these systems.

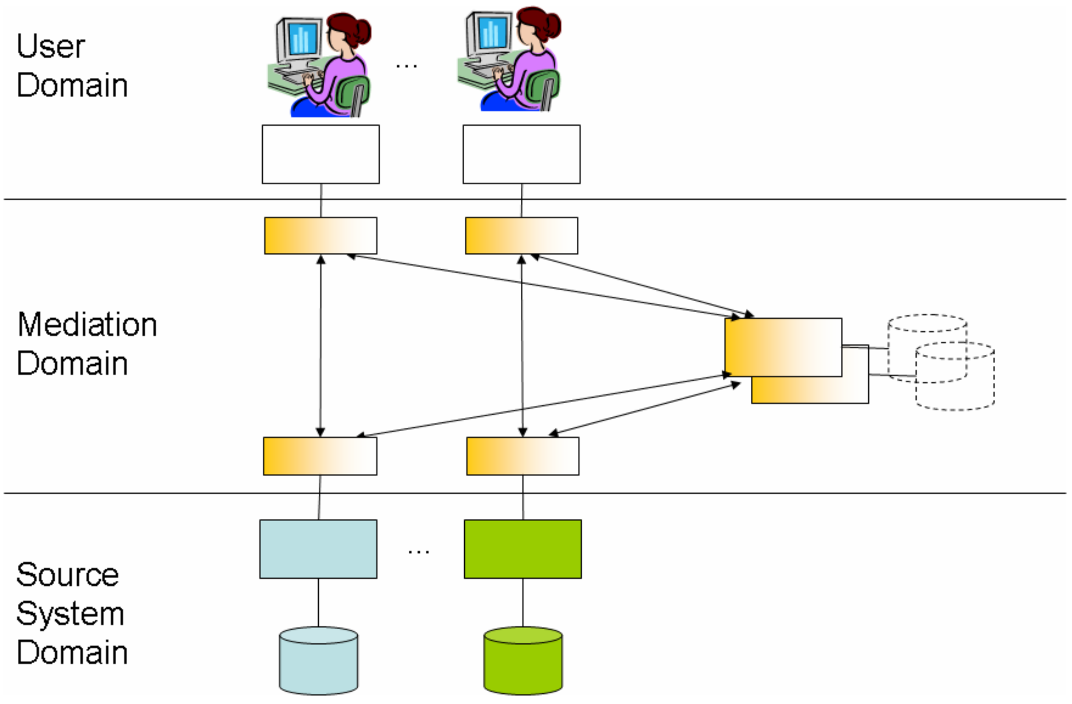


Figure 7: Functional Domains of the ORCHESTRA Service Network

The components of an ORCHESTRA Service Network are classified according to the following functional domains:

- User Domain: provides the interface to a user component (a human or a software component) and interacts with the ORCHESTRA services of the Mediation Domain.
- Mediation Domain: provides the main functional part of the OSN. It mediates the service calls from the user domain to the source system domain based on meta-information exchanged with the system components of the thematic domain (e.g. by means of a publishing or a retrieval pattern). Note that the implementation of the services in the mediation domain will be designed themselves as a distributed, possibly functionally redundant system.
- 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. By means of connector services, these source systems are connected to the mediation domain.

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).

7.2 Overview about Design Decisions

The ORCHESTRA Architecture is the combined specification of the ORCHESTRA Information and Service Viewpoint. 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 pure 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 in but further refined in the respective section. In this case, a forward reference is used.

7.2.1 Usage and Identification of Features

The basic informational unit as perceived by ORCHESTRA Applications is the concept of a feature. A feature is an abstraction of a real world phenomenon whereby the “real world” explicitly includes hypothetical worlds or worlds of man’s thought. The concept of a feature is further specified in section 8.2. In analogy to object-oriented paradigms, it has to be distinguished between feature types (see object classes) and feature instances (see objects).

In order to enable an unambiguous identification of feature instances within an OSN, feature instances are identifiable by a Unique Identifier (UID) that is unique with respect to at least one OSN. If a Feature Access Service (see section 9.4.2) is used to create a new feature instance it must assure to assign to it a unique (i.e. not yet used) UID.

Note: Rules to assure the uniqueness of UIDs within an OSN but also across OSNs as well as supporting mechanisms (e.g. OA infrastructure services) for their creation will have to be investigated.

7.2.2 Access control

The access to resources being feature instances or service instances is controlled by authentication and access control mechanisms. Access encompasses access from human users but also from software components. This is handled by three services: the User Management Service (see section 9.4.13), the Authentication Service (see section 9.4.15) and the Authorisation Service (see section 9.4.14).

Their combined usage is described in an OA pattern in section 9.7.1. The main concept is that of a “session key” that is returned by the Authentication Service after a successful authentication. This session key will then be used in all further interactions with the services in an OSN. However, a default session key allows the access to all resources that don’t require access control. This decision is up to the providers of the OSIs.

Note: The access control concept will be detailed in a later version of the RM-OA. Thus, the “session key” has not yet been included in the service descriptions in section 9.

7.2.3 Service Interaction Modes

ORCHESTRA Services will support a range 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: A mixture of these modes and other variants will be investigated.

Note 2: These modes are described here on the conceptual level. 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.

Note: The interaction between services is handled by the Service Chain Access Service (see section 9.5.9).

7.2.4 Registration of Resources

Registration means the creation of a respective meta-information entry of a resource in a catalogue such that a user that is part of the information community of the respective catalogue may discover the resource. The process of registration as well as the process of discovery is supported by operations of the Catalogue Service (see section 9.4.10). A resource may be registered in one or more catalogues.

Note: A generic description of the "registration process" is to be provided.

7.2.5 Generation of Meta-Information

In ORCHESTRA, several services provide for generation of meta-information. Figure 8 outlines known methods for that purpose and assigns the respective ORCHESTRA service to each method:

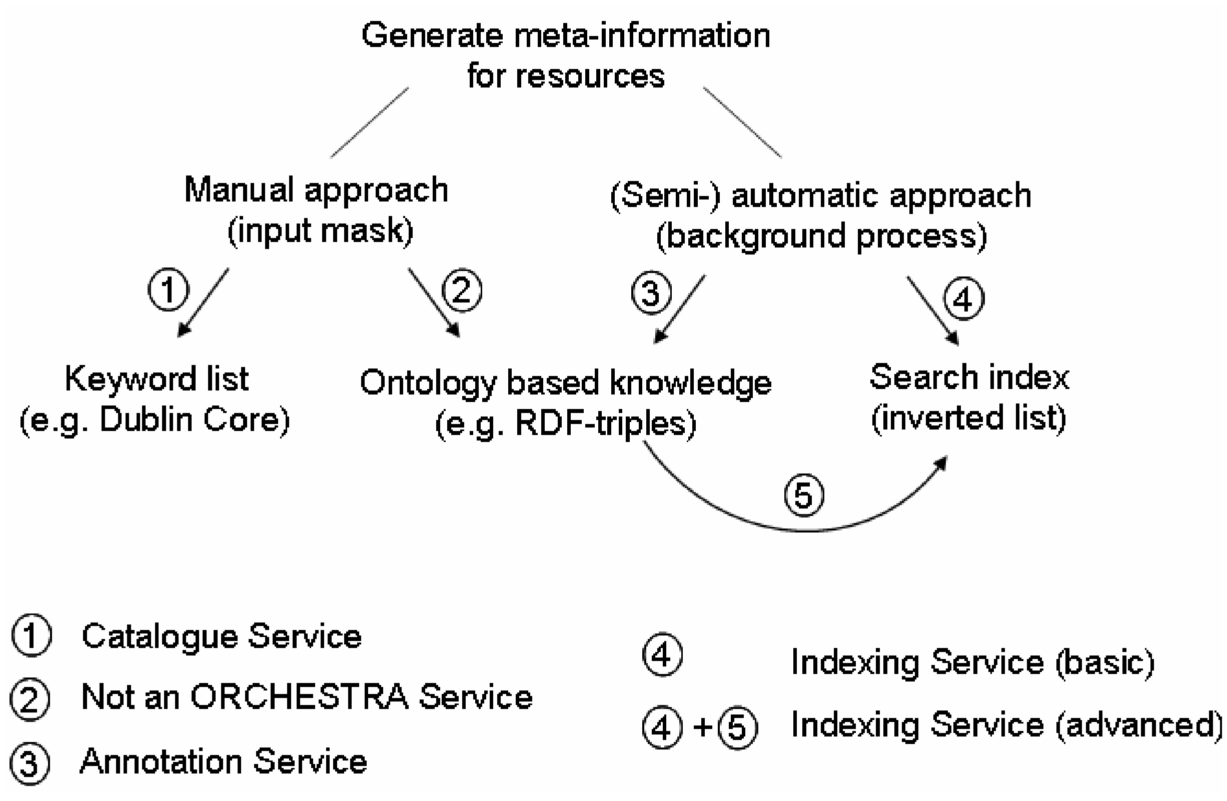

Generate meta-information for resources
Manual approach (input mask)
(Semi-) automatic approach (background process)
1
2
3
4
Keyword list (e.g. Dublin Core)
Ontology based knowledge (e.g. RDF-triples)
Search index (inverted list)
5
1 Catalogue Service
2 Not an ORCHESTRA Service
3 Annotation Service
4 Indexing Service (basic)
4 + 5 Indexing Service (advanced)

Figure 8: Services for generation of resource meta-information

Meta-information in ORCHESTRA is generated for various types of resources, being features or services, and according to a well-defined purpose (see section 8.5). The main distinction of methods for the generation of meta-information is the division into manual and automatic (respective semi automatic) approaches.

Manual generation of meta-information is usually done 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. One the other hand, meta-information may be schema information in order to support the mapping of information between several schemata. In ORCHESTRA, the Catalogue Service (see section 9.4.10) can be used for the access to meta-information for discovery purposes (see service type 1 in Figure 8).

A more advanced method for describing resources is to edit statements which can be added to a knowledge base, e.g. an RDF (Resource Description Framework) Triple Store. The triples describe the relationship of 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, manual generation of ontology-based knowledge (see service type 2 in Figure 8) is not foreseen as an ORCHESTRA Service, this is rather done by an automatic approach (see below).

Population of ontology-based knowledge bases in ORCHESTRA is done by means of an automatic approach implemented in the Annotation Service (see service type 3 in Figure 8 and the description in section 9.5.3), which automatically identifies 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 a dedicated navigation tool or by formulating exact queries in an ontology query language.

In many cases, users do not want to retrieve knowledge about resources, but search and retrieve 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, and references to the occurrence of the term (such an index is called an inverted list). In ORCHESTRA, the Document Indexing Service (see section 9.5.4) provides such a facility (see service type 4 in Figure 8) for all occurrences of documents (i.e. features of type OAS_DocumentDescriptor, see section 8.4.4.2) in an OSN.

A more advanced form of 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.

7.2.6 Interoperability between different Infrastructure Platforms

Note: The role and usage of gateways in order to enable the interoperability of OSIs running in different infrastructure platforms will be considered in a later version of the RM-OA.

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 to build 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 8.5).
- In a second step, the specification framework is enriched by considering the source system level (see section 8.5.4) and the semantic level (see section 8.7) aspects.

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 9.

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 just abstractions, concepts or software artefacts (e.g. documents, software components of IT systems) that may have a physical representation just 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.

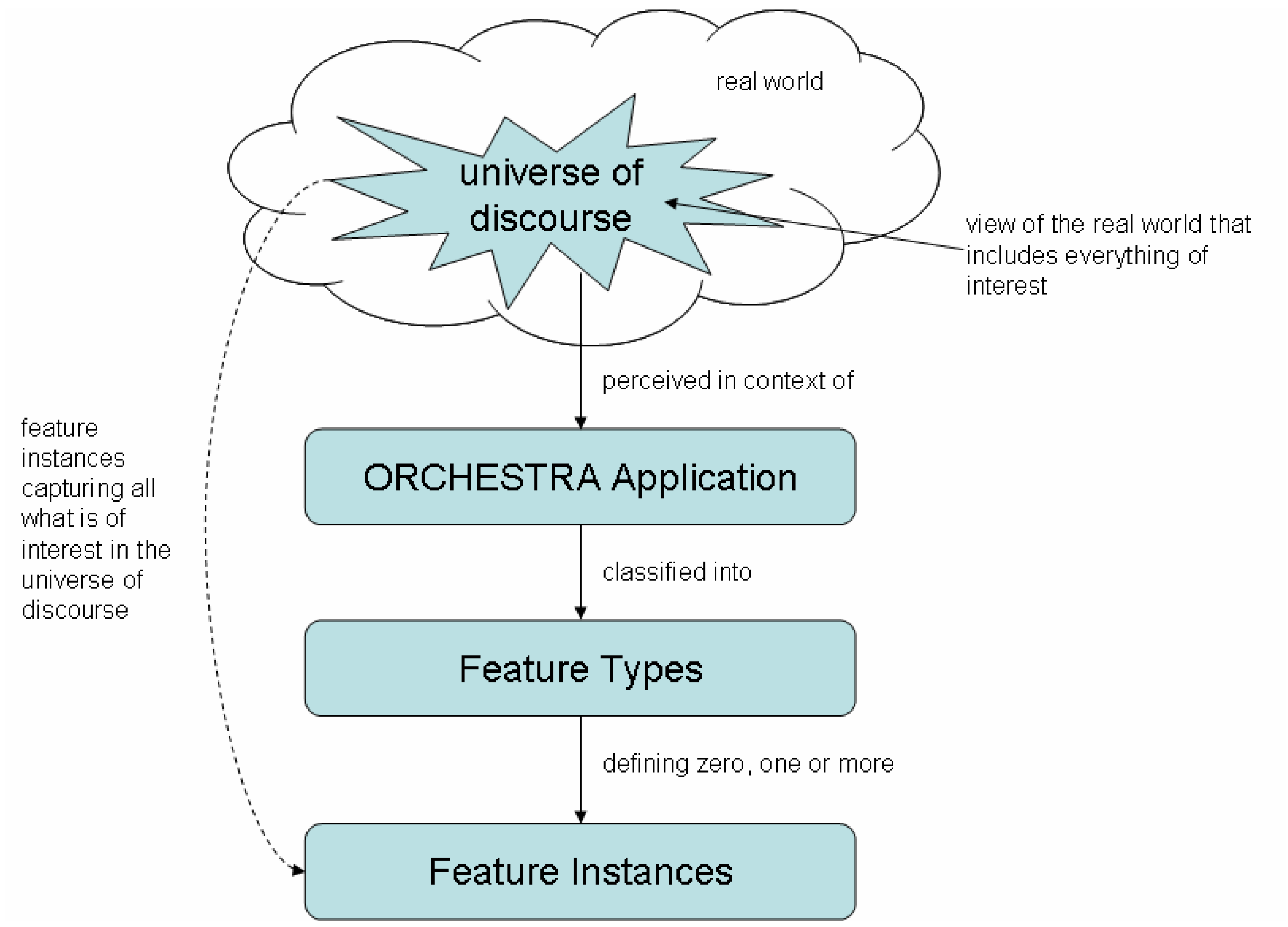

real world
universe of discourse
view of the real world that includes everything of interest
perceived in context of
ORCHESTRA Application
classified into
Feature Types
defining zero, one or more
Feature Instances
feature instances capturing all what is of interest in the universe of discourse

Figure 9: 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 8.4). 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, monitoring station but also sensor observation, measurement value, document, 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) 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 collection of feature instances following the information model formally specified in an OAS is called an ORCHESTRA Feature Set (OFS).

Note: An OFS may but need not be persistently stored in non-volatile memory. It may also be generated on-the-fly from a data set in a source system (see section 8.5.4) and just be kept in a transient store for processing.

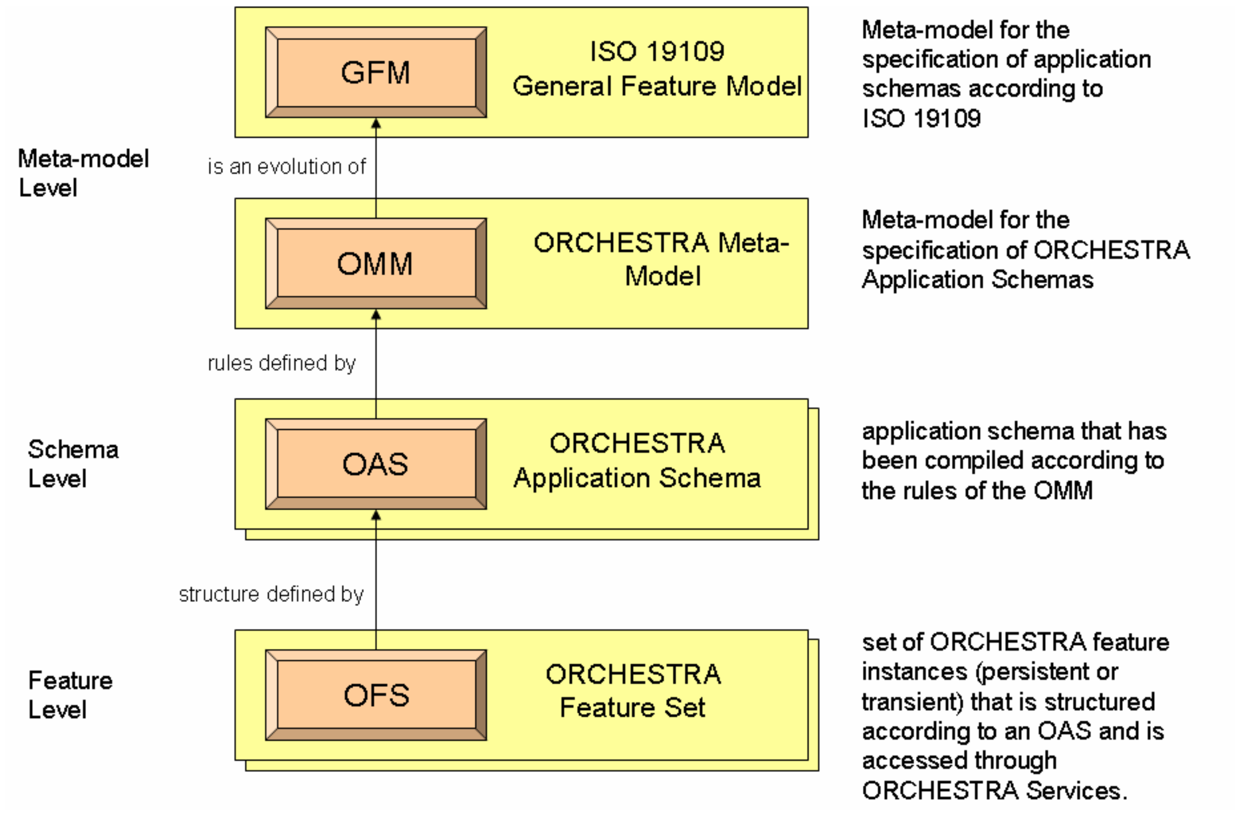

GFM
ISO 19109
General Feature Model
Meta-model for the specification of application schemas according to ISO 19109
Meta-model Level
is an evolution of
OMM
ORCHESTRA Meta-Model
Meta-model for the specification of ORCHESTRA Application Schemas
rules defined by
Schema Level
OAS
ORCHESTRA Application Schema
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 10: Framework for ORCHESTRA Information Models

8.4 The ORCHESTRA Meta-Model

8.4.1 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.
- a “GFM metadata element” has to be used as a GF_Metadata_AttributeType to carry “meta-data” 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 8.5 for additional details.

This is why the OMM is an evolution of the ISO 19109 GFM taking into account additional, ORCHESTRA-specific requirements. The OMM is specified in two steps:

- the OMM selects the classes and properties of the GFM that are relevant for ORCHESTRA (see sections 8.4.2 and 8.4.3)
- the OMM adds additional meta-classes, namely for additional meta feature and attribute types (see sections 8.4.3 and 8.4.4). 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.

8.4.2 OMM Basic Part

The UML class diagrams in Figure 11 shows 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.

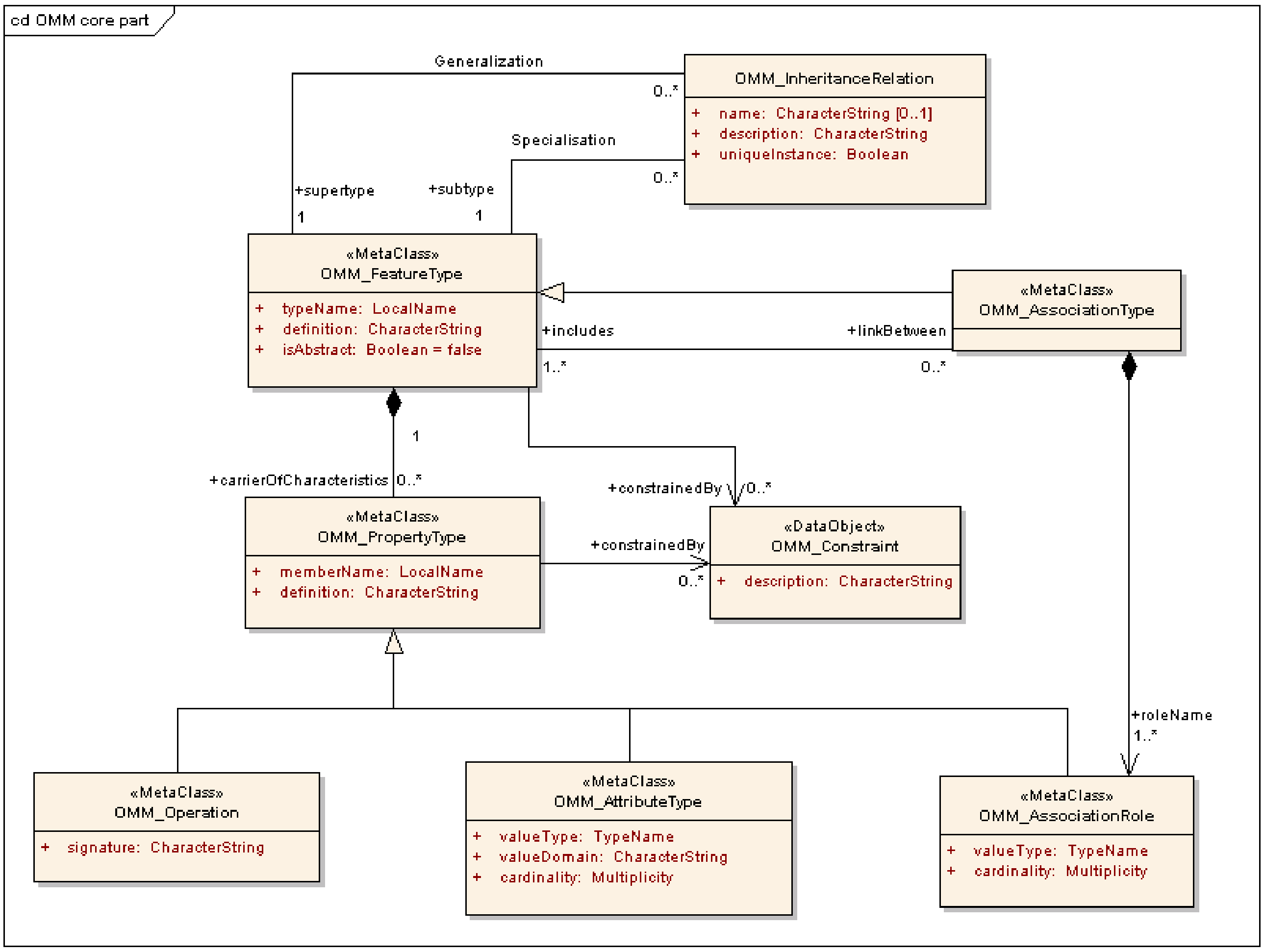


Figure 11: The basic part of the ORCHESTA Meta-model

8.4.3 OMM Attribute Types

The OA 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 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 8.5.1).

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 8.8.10

The resulting schema is illustrated in UML in Figure 12.

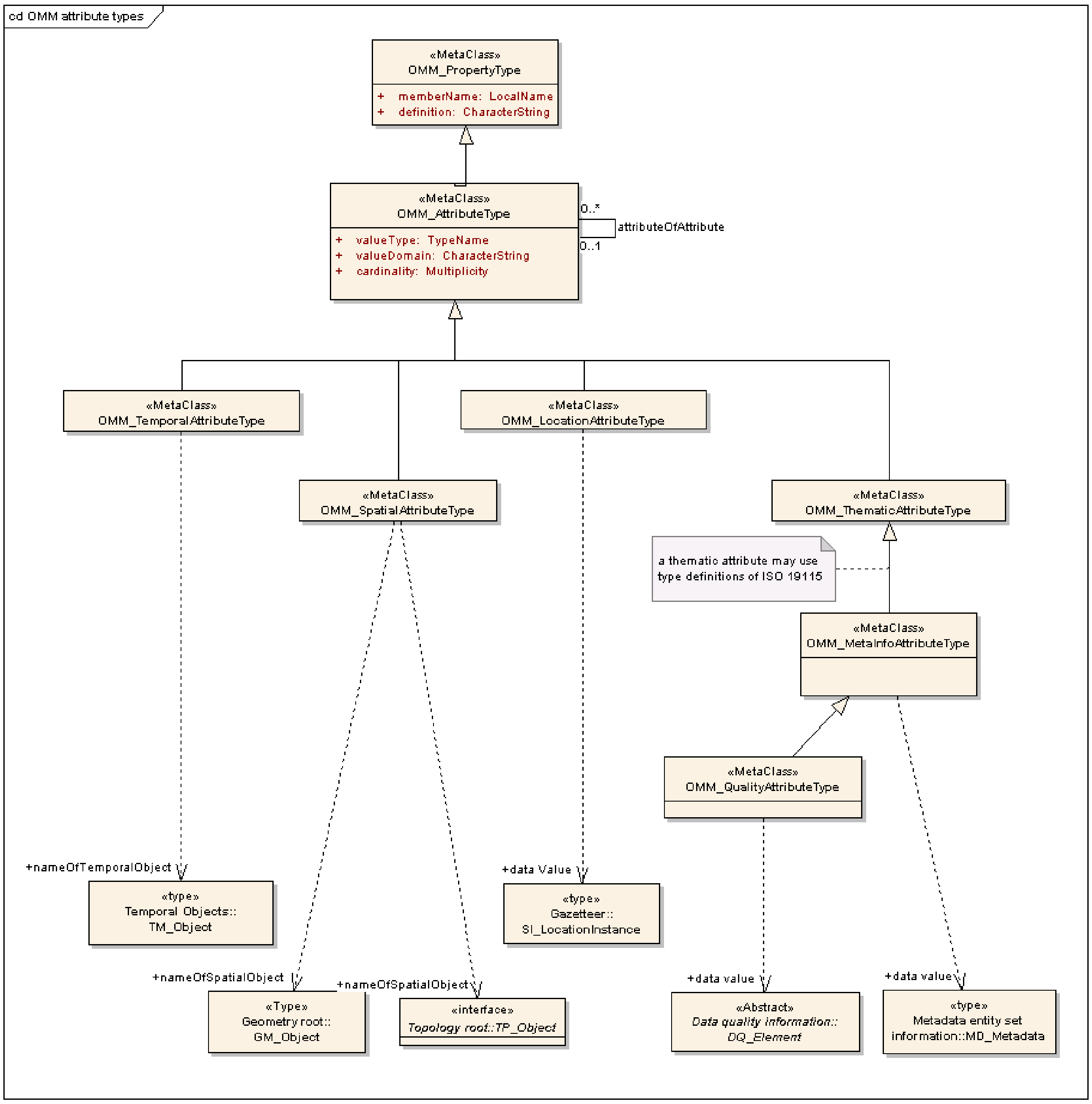

cd OMM attribute types
«MetaClass» OMM_PropertyType
+ memberName: LocalName
+ definition: CharacterString
«MetaClass» OMM_AttributeType
+ valueType: TypeName
+ valueDomain: CharacterString
+ cardinality: Multiplicity
0..*
attributeOfAttribute
0..1
«MetaClass» OMM_TemporalAttributeType
«MetaClass» OMM_LocationAttributeType
«MetaClass» OMM_SpatialAttributeType
«MetaClass» OMM_ThematicAttributeType
a thematic attribute may use type definitions of ISO 19115
«MetaClass» OMM_MetaInfoAttributeType
«MetaClass» OMM_QualityAttributeType
+nameOfTemporalObject
«type» Temporal Objects:: TM_Object
+data Value
«type» Gazetteer:: SI_LocationInstance
+nameOfSpatialObject
«Type» Geometry root:: GM_Object
+nameOfSpatialObject
«interface» Topology root::TP_Object
+data value
«Abstract» Data quality information:: DQ_Element
+data value
«type» Metadata entity set information::MD_Metadata

Figure 12: OMM Attribute types

8.4.4 OMM Extensions to Feature Types

8.4.4.1 Overview

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

- Document type (see section 8.4.4.2)

- Schema type (see section 8.4.4.3)
- Source system type (see section 8.4.4.4)
- Coverage type (see section 8.4.4.5)

Note 1: This list of information types is preliminary and corresponds to the current understanding of the user requirements as described in the ORCHESTRA SP2 deliverables.

Note 2: All schemas specified for these information types in this section are preliminary drafts and will be subject to change during the course of the ORCHESTRA specification and implementation process.

The following list of entities has been identified as further candidates for OMM information types. Their specification needs further clarification:

- equation/formulae
- model
- observation and measurement (see e.g. (OGC 2003a))
- dictionary and code list
- action (see (ORCH-D2.1.3), section 6))
- meeting/conference/telephone call, see (ORCH-D2.1.3), sections 4 and 6 or (ORCH-D2.4.1), section 4)
- software (see (ORCH-D2.4.1), section 4)

8.4.4.2 Document Type

Documents are resources that contain recorded information and can be treated as unit. As 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 to the definition of OAS_DocumentDescriptor as defined below.

Document types include

- Documents with page layout (e.g. PDF, MS-Word, MS-PowerPoint files)
- Web pages
- 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 13.

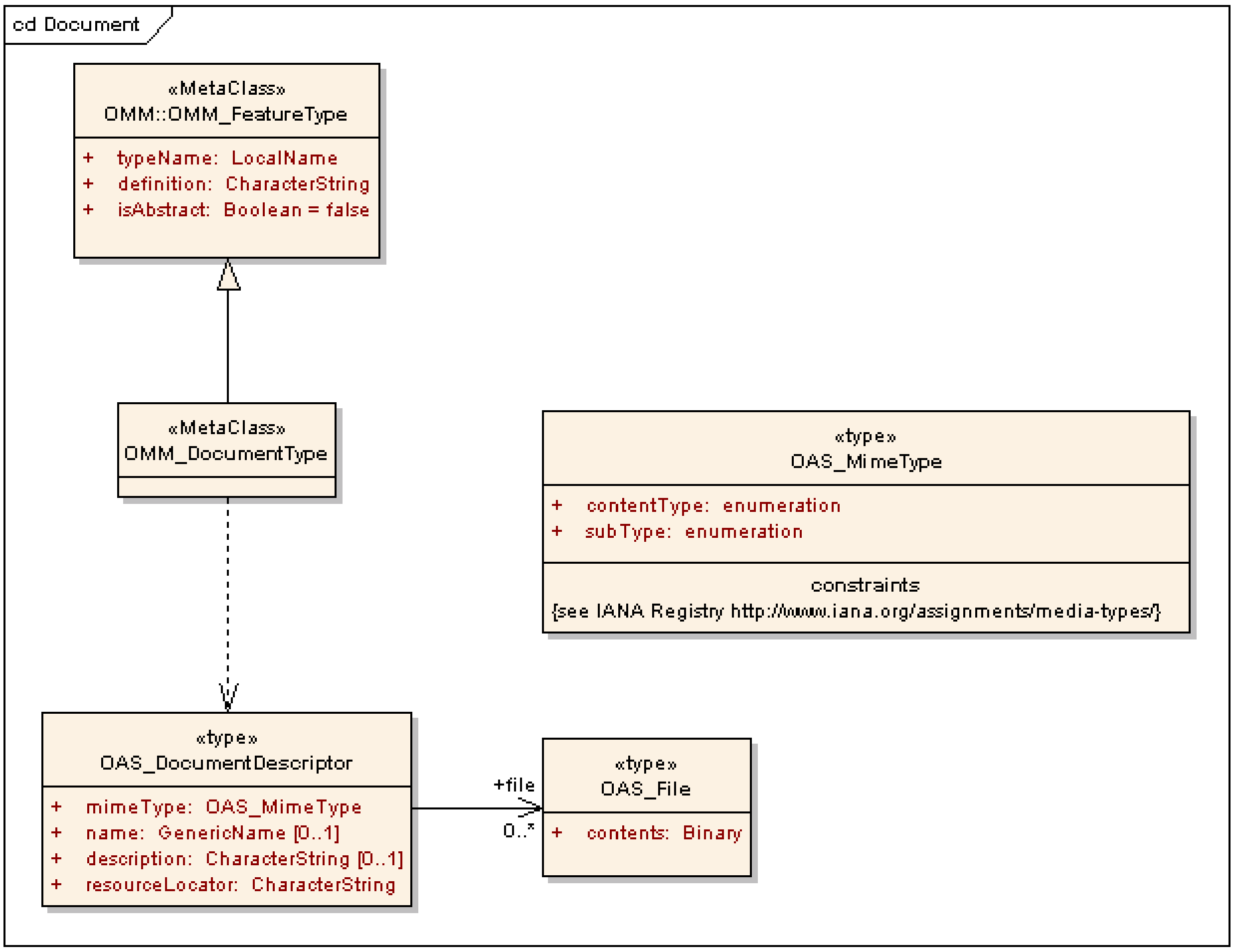

cd Document
«MetaClass»
OMM::OMM_FeatureType
+ typeName: LocalName
+ definition: CharacterString
+ isAbstract: Boolean = false
«MetaClass»
OMM_DocumentType
«type»
OAS_MimeType
+ contentType: enumeration
+ subType: enumeration
constraints
{see IANA Registry http://www.iana.org/assignments/media-types/}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

Figure 13: Schema of the OMM "document type"

8.4.4.3 Schema Type

A schema is a formal description of a model. Examples are the database schema of a relational data base or an application schema specified in UML or XML but also 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, native access method) and refers to one of more documents that contain the schema definition. The documents are represented by document descriptors that are themselves features, see section 8.4.4.2

The schema of the OMM "schema type" is specified in Figure 14.

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")

Note: This schema is a preliminary draft and will be subject to change during the course of the ORCHESTRA specification and implementation process.

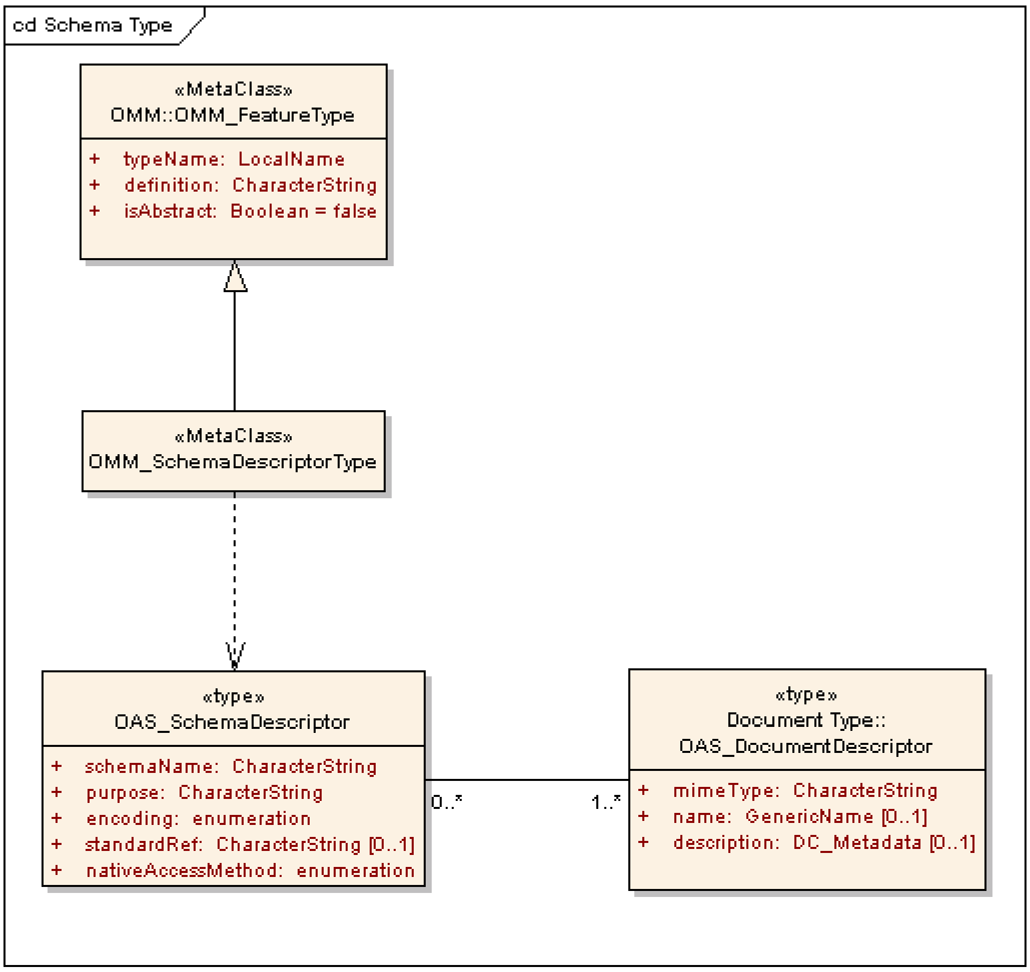

cd Schema Type
«MetaClass»
OMM::OMM_FeatureType
+ typeName: LocalName
+ definition: CharacterString
+ isAbstract: Boolean = false
«MetaClass»
OMM_SchemaDescriptorType
«type»
OAS_SchemaDescriptor
+ schemaName: CharacterString
+ purpose: CharacterString
+ encoding: enumeration
+ standardRef: CharacterString [0..1]
+ nativeAccessMethod: enumeration
0..*
1..*
«type»
Document Type::
OAS_DocumentDescriptor
+ mimeType: CharacterString
+ name: GenericName [0..1]
+ description: DC_Metadata [0..1]

Figure 14: Schema of the OMM "schema type"

8.4.4.4 Source System Type

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 very heterogeneous nature and contain information in a variety of types and formats.

Examples are:

- database containing structured data (e.g. numerical model data), i.e. information that is organized 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.

As ORCHESTRA feature type, a source system type is represented by a source system descriptor that possesses identifying information (such as name, provider name and source system type) and contains zero or more service descriptors as well as zero or more schema descriptors. The service descriptor corresponds to the meta-information that is specified for the purpose of service discovery (see section 8.5.1).

The schema descriptor identifies the structure of the data that is contained in the source system. If the schema is an OAS it may be directly accessed through the Feature Access Service (see section 9.4.2).

However, usually the data in the source systems is organized according to some different schema like in a relational database or in a file-based representation of tabular data like in MS-Excel. By knowing the schema type and the schema description, the Schema Mapping Service (see section 9.5.6) may be used in order to transform it to an OAS such that ORCHESTRA services may be directly used.

An instance of OMM_ThematicAttributeType may represent an attribute that carry source system information. Source system attributes shall have their value type according to the definition of OAS_SourceSystemDescriptor as defined below.

The source system schema used in ORCHESTRA is specified in Figure 15.

Note: This schema is a preliminary draft and will be subject to change during the course of the ORCHESTRA specification and implementation process.

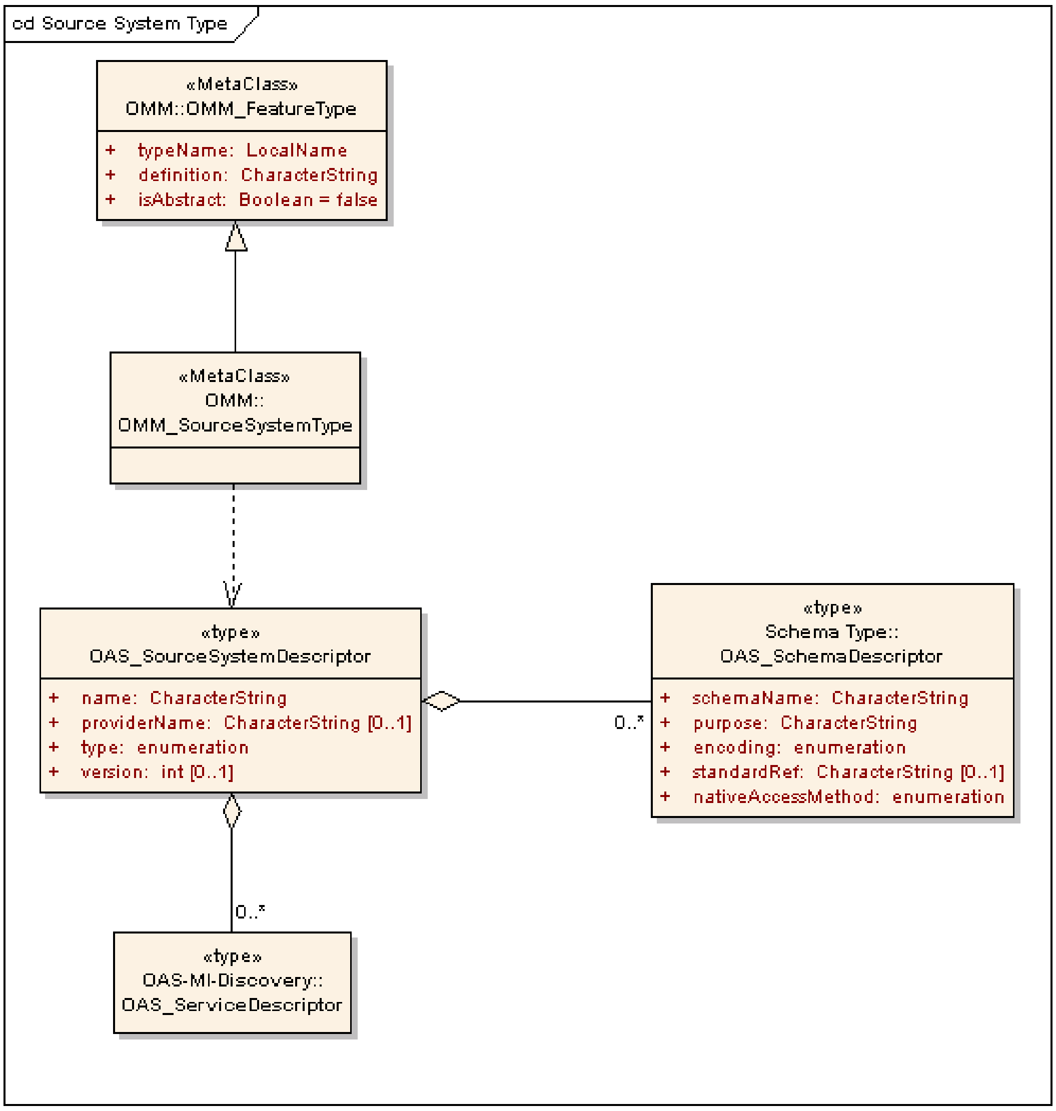


Figure 15: Schema of the OMM “source system type”

8.4.4.5 Coverage Type

A coverage is a feature that acts as a function to return values from its range for any direct position within its spatiotemporal domain. Examples include a raster image, polygon overlay, or digital elevation matrix. The coverage model is defined by ISO 19123.

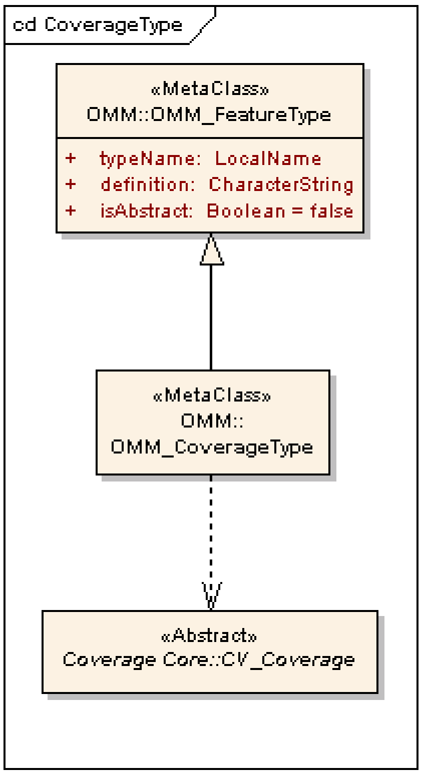


Figure 16: Schema of the OMM "coverage type"

8.5 Framework for ORCHESTRA Meta-Information Models

8.5.1 Overview

The following definition for meta-information that is derived from the principal ideas of (ORCH-D3.3.1 2005) 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 world that includes everything of interest (see ISO 19101 and also section 8.2)
- particular purpose: a particular use case or the goals 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: conceptual model for meta-information.

The above definition indicates that the need for meta-information does not arise in the context of managing single data objects or services, but 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 conceptual model (the meta-information model) 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 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.
3. Meta-information may also be implicitly derived from the existence or content of the resources without requiring that this information is 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 a single meta-information model which is valid for any purpose. Instead, ORCHESTRA defines for each purpose a dedicated meta-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 meta-information model, specify the conceptual schema for the meta-information model.

The RM-OA defines meta-information models for the following purposes that are further explained in the subsequent sub-sections:

- discovery (including search and navigation)
- access, storage and service invocation
- integration
- interpretation
- user profiling
- OSN Management

8.5.2 Description of Purposes

8.5.2.1 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 formulation of a search query that is submitted to the search engine. The search engine returns a number of objects that it has identified as relevant with respect to the query (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 meta-information needed 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.

8.5.2.2 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, access rights, login information etc. The storage and retrieval will be handled by a “data access service”, 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. used for discovery). 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).

8.5.2.3 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 means 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 process 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.

8.5.2.4 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.

8.5.2.5 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 can be provided for different users and tasks.

Example meta-information for a particular view on a catalogue might be user information (user group, service provider, service/data integrator, administrator, etc.) and a particular language. The related services are typically catalogue services.

8.5.2.6 Purpose “OSN 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.

8.5.3 Framework Specification

The framework for ORCHESTRA Meta-Information Models 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 for a persistent store is a catalogue for discovery or navigational purposes, an example for 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 the Meta-Information Bases are generated according to some purpose, there may be different application schemas for different purposes. ORCHESTRA does not specify one application schema for meta-information models for all tasks. Instead, the ORCHESTRA Meta-Information Models consist of the set of OAS-MIs that are defined according to the purposes identified above.

Note: The OAS-MIs according to these purposes are being specified in the ORCHESTRA deliverable D3.3.2. They will be attached as an annex in a further version of the RM-OA.

Depending on the purpose, an OAS-MI may be related to an OAS through some relationships among 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 which already includes particular extensions (see section 8.4.3) and dedicated rules (see section 8.8) for the definition of OAS-MI.

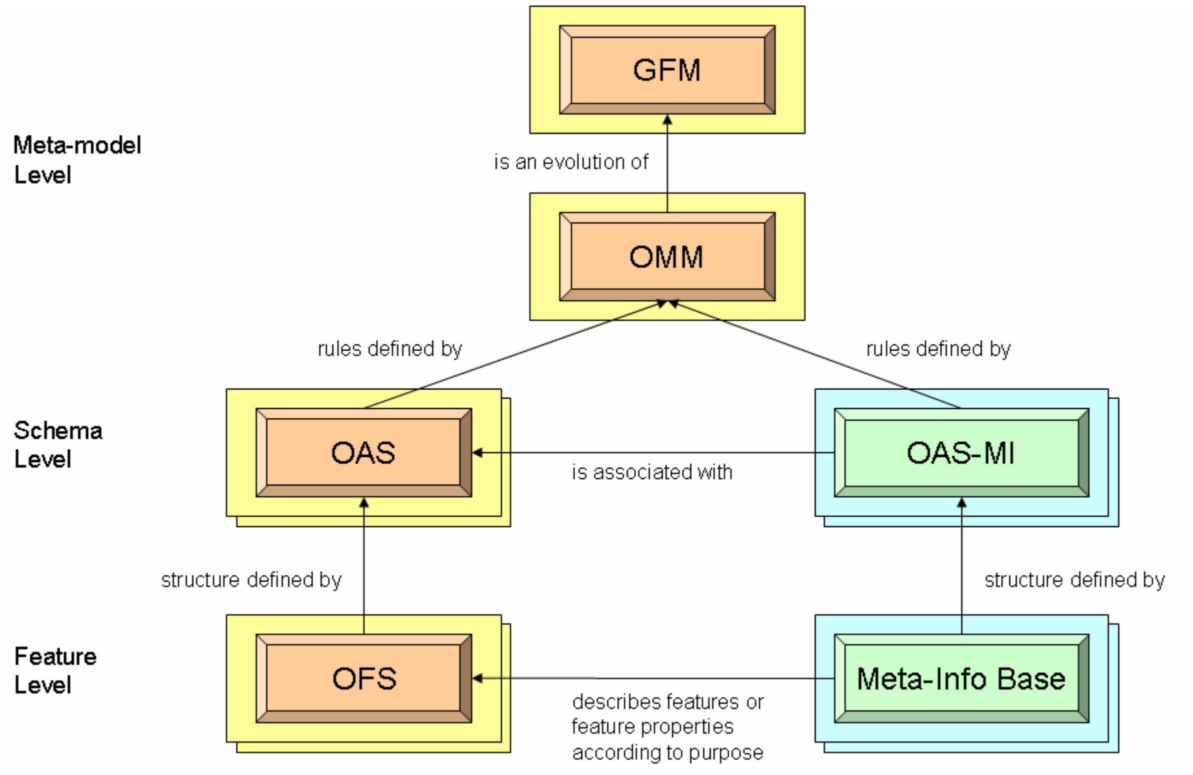


Figure 17: Framework for the ORCHESTRA Meta-Information Model

8.5.4 OMM Extensions for Meta-information Association Types

In order to allow an instance of OMM_FeatureType to serve as meta-information for another instance of an OMM_FeatureType another subclass OMM_MetadataAssociationType is added to OMM_AssociationType (see Figure 18). This means that in an OAS, classes marked as feature types can be associated with each other using instances of the OMM_MetadataAssociationType.

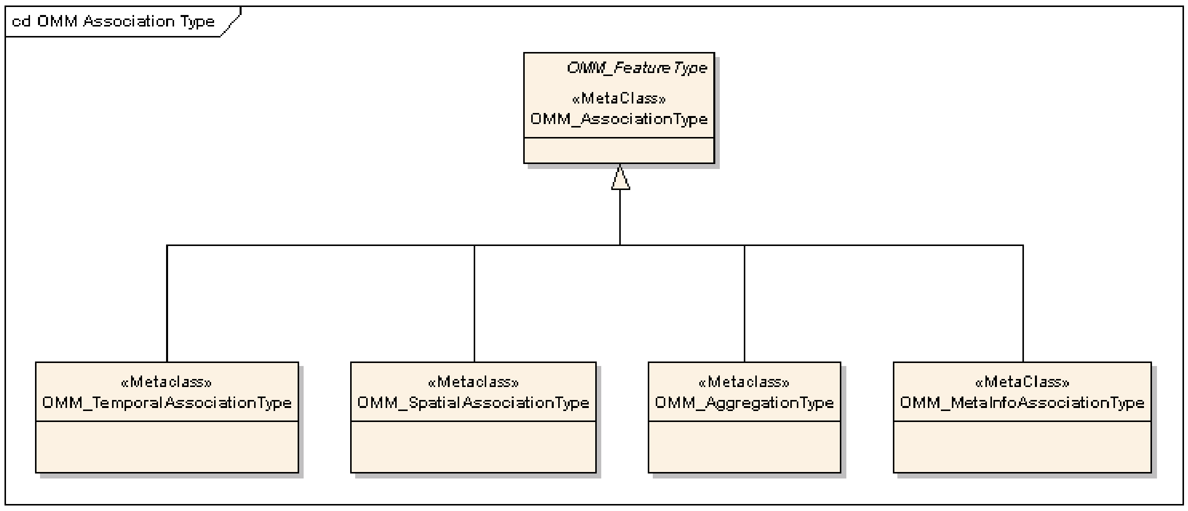


Figure 18: Subclasses of OMM_AssociationType

Note 1: The list of subclasses is not complete in Figure 18.

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.

8.6 Inclusion of the Source System Level

8.6.1 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. For this purpose, ORCHESTRA offers means and services for syntactical 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.

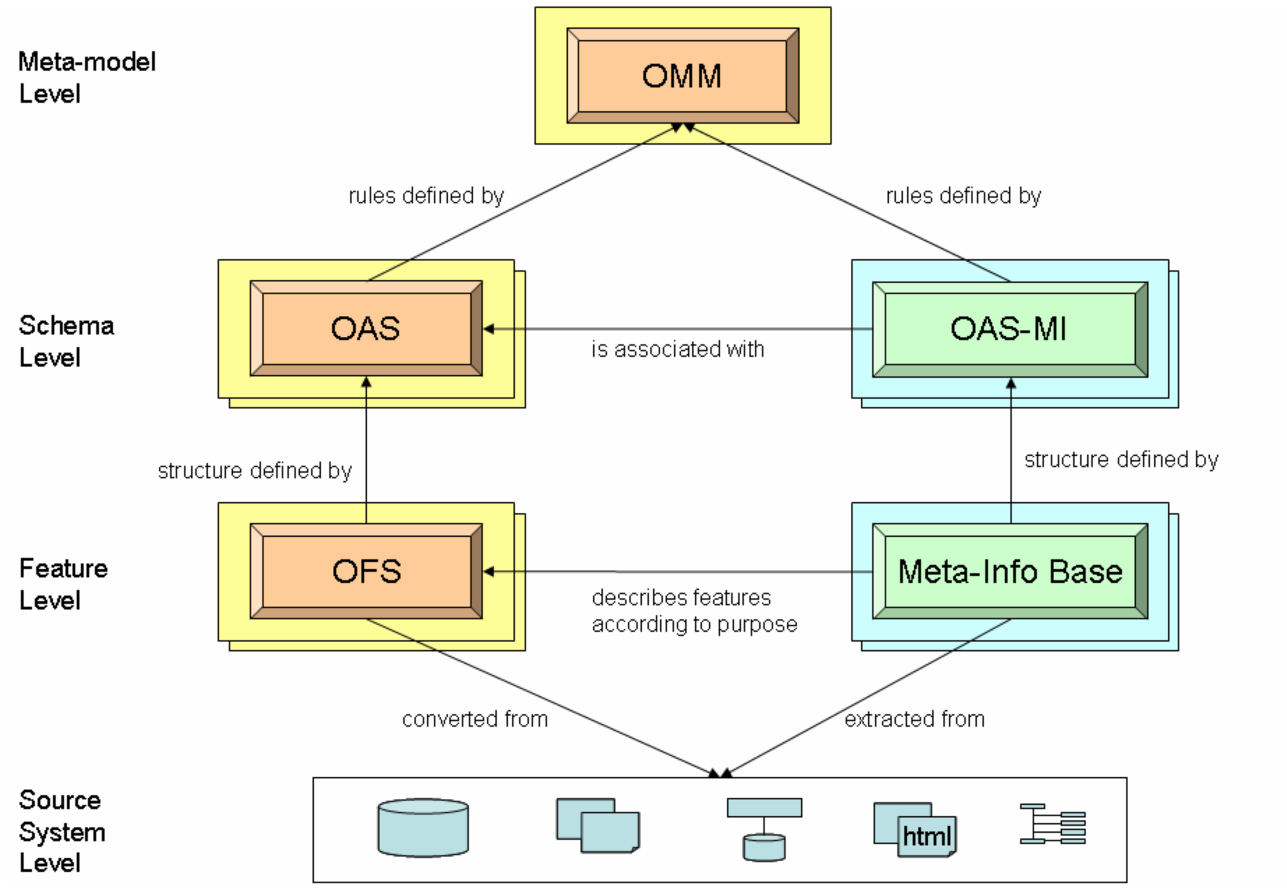


Figure 19: Inclusion of the Source System Level into the ORCHESTRA Information Model Framework

Source systems are of 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 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 19 illustrates the consequences for the information model framework when explicitly taking the source system level into account.

The majority of source systems does neither 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 that ORCHESTRA services may 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.

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 meta-information extraction (annotation) process initiated by an OSN component. In any case, the extraction of meta-information is guided by the respective OAS-MI specifically designed for this particular purpose.

Note: The conversion process of source system information into an OFS and the extraction process of 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).

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

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

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 20

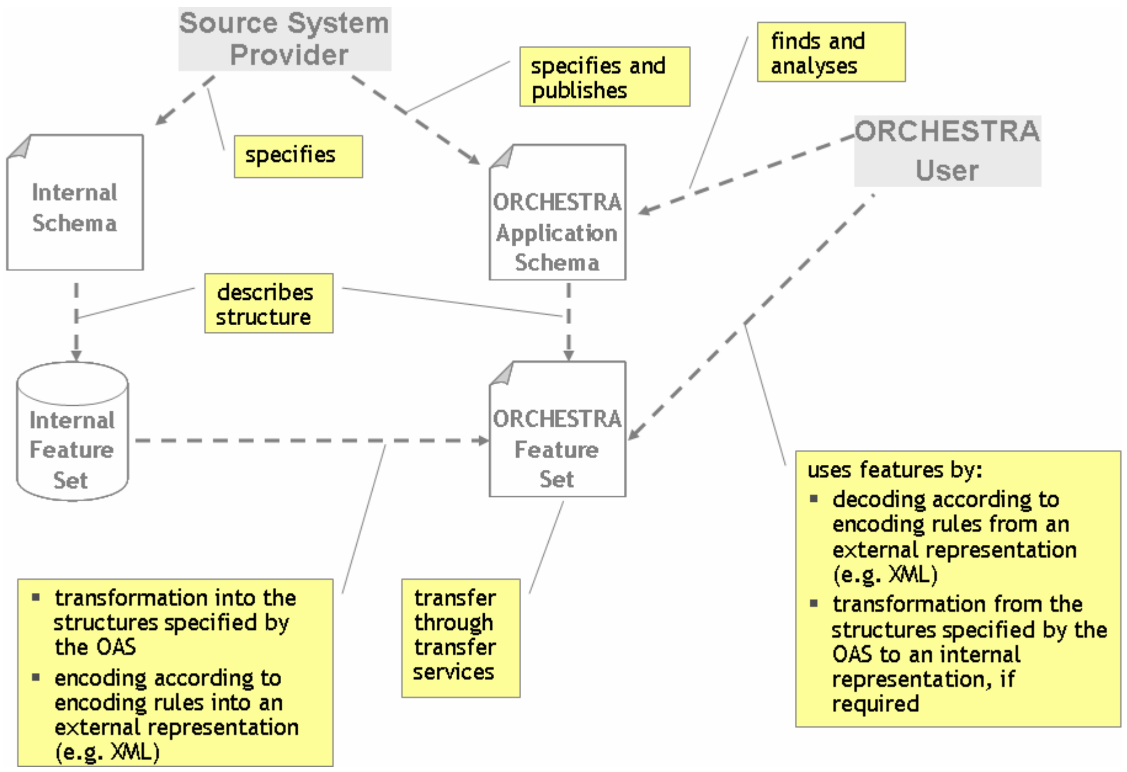

Source System Provider
specifies
specifies and publishes
finds and analyses
ORCHESTRA User
Internal Schema
ORCHESTRA Application Schema
describes structure
Internal Feature Set
ORCHESTRA Feature Set
uses features by:
▪ decoding according to encoding rules from an external representation (e.g. XML)
▪ 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 20: Ad-hoc use of published feature sets and application schemas

8.7 Inclusion of the Semantic Level

8.7.1 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 may be thought of as a formal representation of the knowledge associated with a particular subject area (domain) or task (ORCH-D2.3.1 2005). Their ultimate purpose is to enable machine understanding, which in turn provides the potential for data and service interoperability.

8.7.1.1 Ontology Classes

Ontologies may be broadly classified as listed in Table 2 (ORCH-D2.3.1 2005). 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 organisation 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 2: Ontology Classes (ORCH-D2.3.1 2005)

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 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 is meta-information for the source systems. Thus, they will be considered as part of the 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 is meta-information for the services of source systems. Thus, they will be considered as part of the 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 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-D2.3.1 2005).

Note 2: The current version of the RM-OA has its focus on the support of syntactical 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 on how ontologies influence the RM-OA Information and Service Viewpoints.

8.7.1.2 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-D2.3.1 2005). 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 (ORCH-D2.3.1 2005):
  - 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.

8.7.1.3 High-level Ontologies

A high-level ontology could be expected to hold 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-D2.3.1 2005). For example in the thematic context of risk management, a “flood risk” domain ontology may include concepts like “flood risk map”, “risk of flood”, “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 information model framework.

8.7.2 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 21.

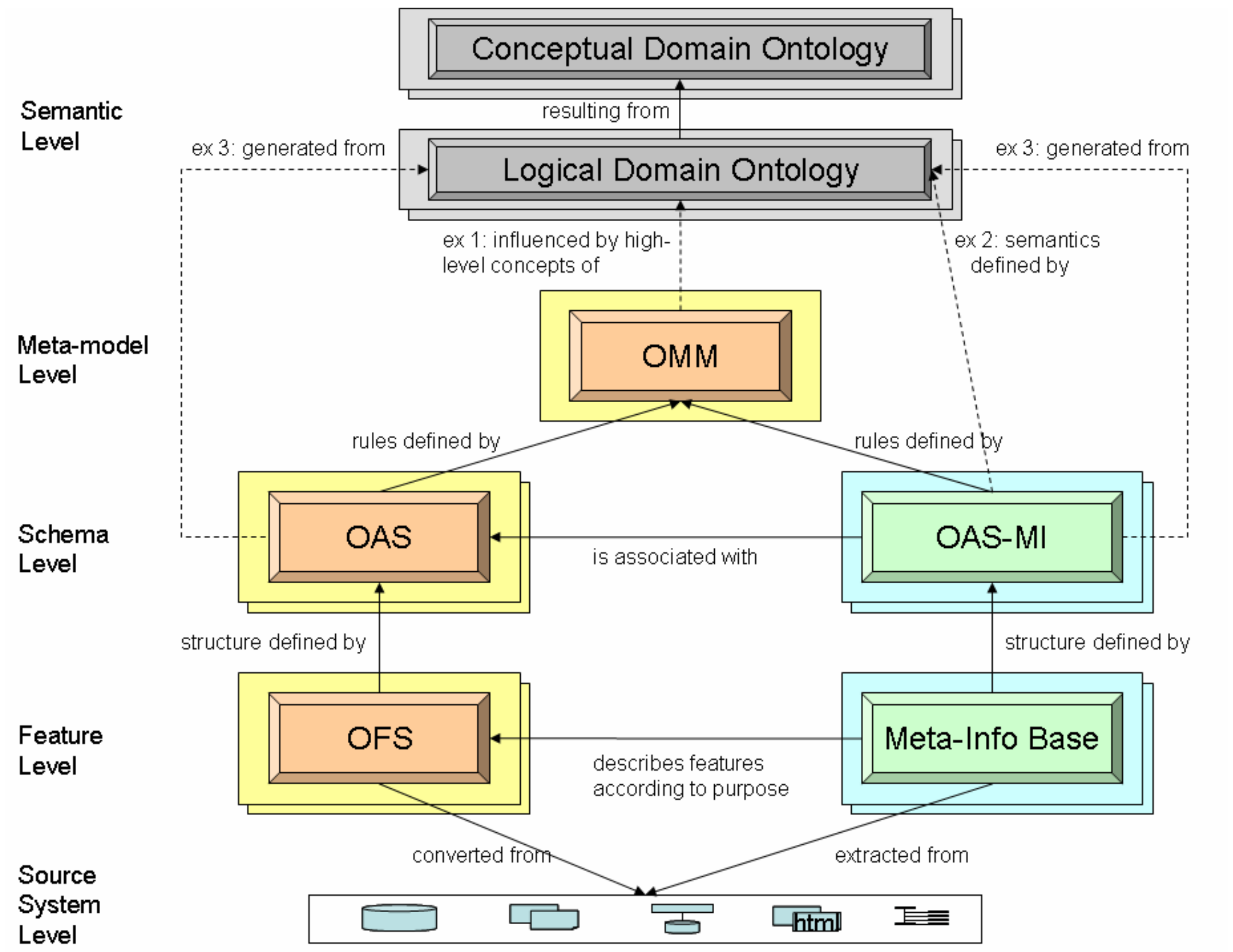


Figure 21: Inclusion of the Semantic Level into the Information Model Framework

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 is specified as a generic architecture, the information viewpoint is neither tied to a specific domain ontology on the conceptual nor on the logical level.

Note: The RM-OA describes an IT architecture based on machine-readable and machine-interpretable information. Thus, the handling of the conceptual model and the transformation process to the logical ontology itself is currently out of scope of the RM-OA.

Examples of relationships to the other levels of the specification framework are illustrated in Figure 21:

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 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.

8.8 Rules for ORCHESTRA Application Schemas

8.8.1 General Approach

The modelling process for OAS corresponds to the description in ISO 19109, section 8.1.

OAS shall be modelled in UML following the rules of ISO/TS 19103, ISO 19109 and ISO/CD 19136.

Note 1: This allows automatic derivation of GML Application Schemas 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 Implementation Platforms are possible from the conceptual UML model.

Note 2: The relationship to the rules for application schemas as specified in ISO 19109, section 8, (conformance, changes and/or extensions) are explicitly indicated in respective notes.

Note 3: Changes that are to be expected during the course of the ISO/CD 19136 standardization process will be incorporated in future versions of the RM-OA.

8.8.2 Main Rules

<u>Rules:</u>

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

   Note: Rule conforming to ISO 19109, section 8.2.2, rule 2).

3) An OAS shall be documented using class diagrams.

4) An OAS shall use UML 2.0 as its conceptual schema language.

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

5) Every class and association 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 the application development, e.g., to decide whether a class is a feature type or not.

6) All package names shall be unique.

7) Dependencies between packages must be modelled explicitly.

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

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

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

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

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

13) Generalization relationships are allowed only between feature types and between data types. These generalizations shall have no stereotype or the stereotype <<disjoint>>. All generalization relationships with other stereotypes will be ignored. The discriminator property of the UML generalization shall be blank.

14) A class may only have more than one supertype if there are no conflicts, e.g., naming conflicts between the inherited attribute and operation.

8.8.3 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: Rule conforming 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.

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

   Note: Rule extending ISO 19109, section 8.2.3, rule 1).

8.8.4 Rules for the Documentation of an OAS

Rules:

1) An OAS shall be documented.

   Note: Rule conforming 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 OAS to the version number of the RM-OA document.

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

   Note: Rule conforming 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: Rule conforming 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: Rule conforming to ISO 19109, section 8.2.4, rule 4).

8.8.5 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 standardized schemas. This organization 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: rule derived from ISO 19109, section 8.2.5, rule 1).

8.8.6 Rules for the Specification of an OAS in UML

Rules:

1) OMM_FeatureType: An instance of OMM_FeatureType shall be implemented as a CLASS stereotyped with <<FeatureType>> except for Rule 2 Case 1 (see OMM_AssociationType below)

   Note: rule extending ISO 19109, section 8.3.1, rule 1).

2) OMM_AssociationType: An instance of OMM_AssociationType shall be implemented as one of the following cases:

   - Case 1: An instance of OMM_AssociationType that is not associated with any instances of OMM_PropertyType. In this case, it has the role of linkBetween in association to instances of OMM_FeatureType being implemented as CLASSes. It shall be implemented as an ASSOCIATION with a stereotype <<Spatial>> in case of a spatial association and <<Temporal>> in case of a temporal association between these CLASSes.

   - Case 2: An instance of OMM_AssociationType that is associated with one or more instances of OMM_PropertyType. It shall be implemented as an ASSOCIATION CLASS stereotype with a valid sterotype for classes; the associated instances of OMM_PropertyType shall be implemented as ATTRIBUTES of the ASSOCIATION

   Note: Rule conforming to ISO 19109, section 8.3.1, rule 2).

3) 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: Rule conforming to ISO 19109, section 8.3.1, rule 3).

4) 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: Rule conforming to ISO 19109, section 8.3.1, rule 4).

5) 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 sterotype 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: Rule extending ISO 19109, section 8.3.1, rule 5).

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

6) 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: Rule conforming to ISO 19109, section 8.3.1, rule 6).

7) 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).

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

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

9) 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: Rule extending ISO 19109, section 8.3.1, rule 9).

8.8.7 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: Rule conforming to ISO 19109, section 8.4.2, rule 1).

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

8.8.8 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: Rule conforming 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: Rule conforming to ISO 19109, section 8.4.3, rule 2).

   Note 2: The specifications of OMM extension types (see section 8.4.4) are handled like standard schemas. The rules to be considered for a possible reduction are specified in section 8.9.

8.8.9 Rules for Adding Information to an OAS

Rule:

1) If it is necessary to extend 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.

8.8.10 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: Rule conforms to the RM-OA approach to handle meta-information (see section 8.5.1). The decision if an attribute is to be considered as meta-information cannot be taken at design time.

8.8.11 Rules for Meta-Information Attributes

Note: These rules will be specified when the specification of the OAS-MIs will be available (ORCHESTRA deliverable D3.3.2).

8.8.12 Rules for Temporal Attributes

Rule:

1) If a common representation of time across systems is required then it is recommended that any description of temporal aspects is 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.

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 specified otherwise in the RM-OA.

   Note: This recommendation is still to be validated in the course of the ORCHESTRA specification and implementation process.

8.8.13 Rules for Spatial Attributes

Rule:

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 an profile of 19107 widely adopted (see the OGC simple feature specification). If there is no need to use other data types than those specified in ISO 19125-1, then ISO 19125-1 shall be used.

   Note: Rule extending 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

Rule:

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

Note 1: Rule conforming to ISO 19109, section 8.9, rule 1).

Note 2: This rule is still to be validated in the course of the ORCHESTRA specification and implementation process.

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.

8.9 Rules for Information Types extending the OMM

8.9.1 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).

8.9.2 Rules for Coverages

Coverages are considered in the OMM as instances of ORCHESTRA feature types, see section 8.4.4.2. 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 standardized schemas (see RM-OA, section 8.8.5, 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 8.8.2) 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 1: It is to be validated if all of these coverage types are required for most of the applications of the RM-OA or if it may be restricted.

Note 2: The term coverage strongly implies the presentation of 2D spatial phenomena, where especially in the context of geo-scientific modeling also 3D and 4D phenomena have to be considered. For these applications the notion of *field* - as known from physics – would be more appropriate. Here it has to be validated whether fields can be nevertheless modeled using the given coverage type, whether the coverage type needs adoption or even renaming, or whether an additional type field is needed to cope with the respective requirements.

8.9.3 Rules for Documents

Documents are considered in the OMM as instances of ORCHESTRA feature types. Their schema is defined in section 8.4.4.2.

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 OAS_DocumentDescriptor as defined in section 8.4.4.2 and Figure 13 or a subtype as the data type for its value.

8.9.4 Rules for Source Systems

Source systems are considered in the OMM as instances of ORCHESTRA feature types. Their schema is defined in section 8.4.4.4.

Rules:

1) A source system 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 OAS_SoucreSystemDescriptor as defined in section 8.4.4.4 and Figure 15 or a subtype as the data type for its value.

8.10 A Simple Example

An extremely simplified model of an earthquake feature type is illustrated in Figure 22. In terms of the OMM, the feature type "earthquake" has the following properties:

- a spatial property type with the name "location", the value type is a spatial point (see ISO 19107);
- a temporal property type with the name "occurredAt", the value type is a temporal instant (see ISO 19108);
- 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 8.4.4.2).

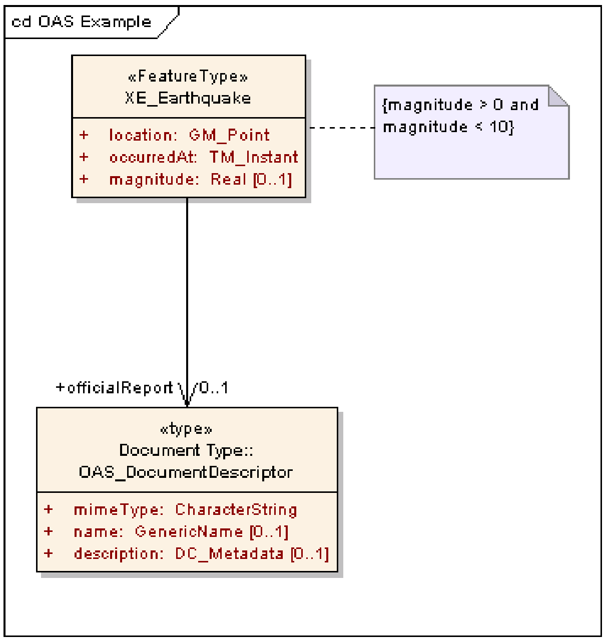


Figure 22: Earthquake example

9 Service Viewpoint

9.1 Overview

The Service Viewpoint of the RM-OA specifies the ORCHESTRA Services that support the syntactical and semantic interoperability between source systems and between services and the development of ORCHESTRA Applications. This includes the management of an OSN as one particular application, too.

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

ORCHESTRA Services are services offered by an ORCHESTRA Service Network whereas a service is a collection of operations, accessible through an interface, that allows a requestor of the service to evoke a behaviour of value to the requestor. The RM-OA specifies the ORCHESTRA Services and their interfaces in two different ways:

- A coarse service description is given for each service in human-readable textual format by using a service description framework, see section 9.2.
- A refined service specification will be given for each service in a dedicated annex to the RM-OA by using UML as the conceptual schema language. These annexes will be provided in an incremental manner according to the priorities of the ORCHESTRA project.

ORCHESTRA Services are classified according to different perspectives in section 9.2

9.2 Classification of ORCHESTRA Services

9.2.1 Functional Classification

ORCHESTRA Services are functionally classified in service categories. 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. The mapping to the OA and OT Services will be documented for future requirements traceability analysis in (ORCH-ReqTrace).

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.

9.2.1.1 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. 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 9.4.10). 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 9.4.5). The degree and kind of indispensability of these services with respect to one OSN are defined in OSN Conformance Clauses, see section 9.8.

- 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. However, no OSN conformance clauses are specified for OA Support Services as these services are not indispensable for the operation of an OSN.

Both 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.

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 23.

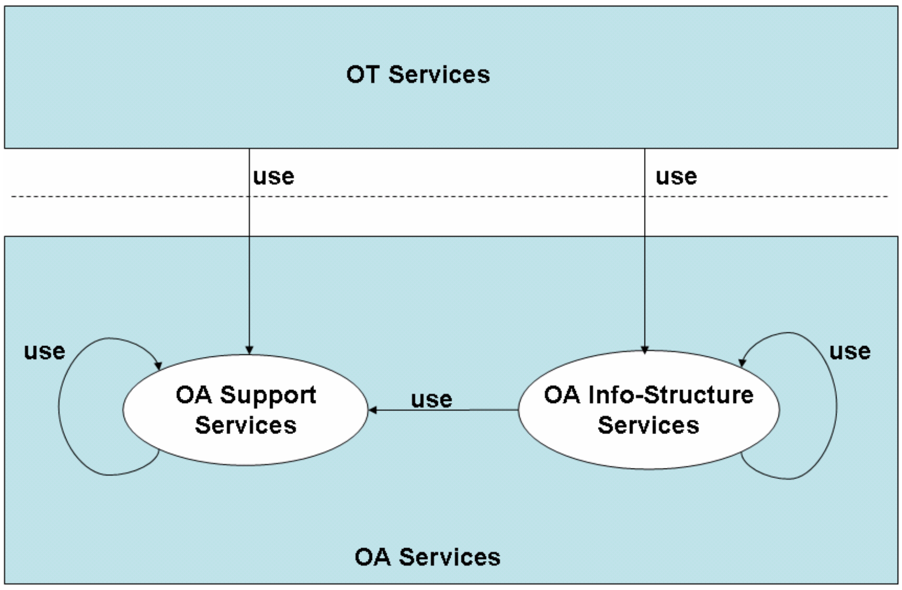


Figure 23: Service Categories as seen by the OA Services

The OA Services do not address any specific thematic application domain, nor do they impose any structure on the OT Services. Table 3 shows the current list of OA Services.

Note 1: The categorisation of an OA Service to be either a mandatory OA Info-Structure service or an optional OA Support service is a preliminary proposal that needs to be validated during the course of the ORCHESTRA project. The current idea is that the mandatory part of an ORCHESTRA service infrastructure must support discovery and access to resources residing in source systems, whereby access means read and/or write access, and, in addition, a possibility to monitor the running services. The rationale for this selection is a compromise between, on the ine 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.

Note 2: The column “ISO 19119 Service Taxonomy” provides just a hint of the position of the OA Service in the ISO 19119 Service Taxonomy.

Note 3: The need for services dedicated to the management of OSIs (i.e. control of the execution of an OSI) has to be investigated.

Service Name	Service Category	ISO 19119 Service Taxonomy	Section
Abstract Services			
RM-OA Service	OA Info-Structure	none	9.4.1
Information Management Services			
Feature Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.2
Map Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.3
Diagram Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.4
Document Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.5
Source System Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.6
Formula Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.7
Coverage Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.8
Sensor Access Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.9
Catalogue Service	OA Info-Structure	Geographic Model/Information Management Services	9.4.10
Gazetteer Service	OA Support	Geographic Model/Information Management Services	9.5.2
Annotation Service	OA Support	Geographic Model/Information Management Services	9.5.3
Document Indexing Service	OA Support	Geographic Model/Information Management Services	9.5.4
Schema Mapping Service	OA Support	Geographic Model/Information Management Services	9.5.6
Format Conversion Service	OA Support	Geographic Model/Information Management Services	9.5.5
Query Mediation Service	OA Support	Geographic Model/Information Management Services	9.4.11
Ontology Access Service	OA Support	Geographic Model/Information Management Services	9.4.12
Thesaurus Access Service	OA Support	Geographic Model/Information Management Services	9.5.7
Inferencing Service	OA Support	Geographic Model/Information Management Services	9.5.8
Processing Services			
Coordinate Operation Service	OA Support	Geographic Processing Services	9.5.1
Workflow/ Task Management Services			
Service Chain Access Service	OA Support	Workflow/Task Management Services	9.5.9
OSN Management Services			
User Management Service	OA Info-Structure	Geographic System Management Services	9.4.13
Authorisation Service	OA Info-Structure	Geographic System Management Services	9.4.14
Authentication Service	OA Info-Structure	Geographic System Management Services	9.4.15
Service Monitoring Service	OA Info-Structure	Geographic System Management Services	9.4.16

Table 3: List of OA Services

9.2.1.2 OT Services

OT Services provide application domain-specific functionality. However, within and also between different application domains high-level functions may be identified that have a generic nature. 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.

As an informative example of further sub-categories of OT Services, although outside the scope of the RM-OA, the application domain of environmental risk management is taken. 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 24.

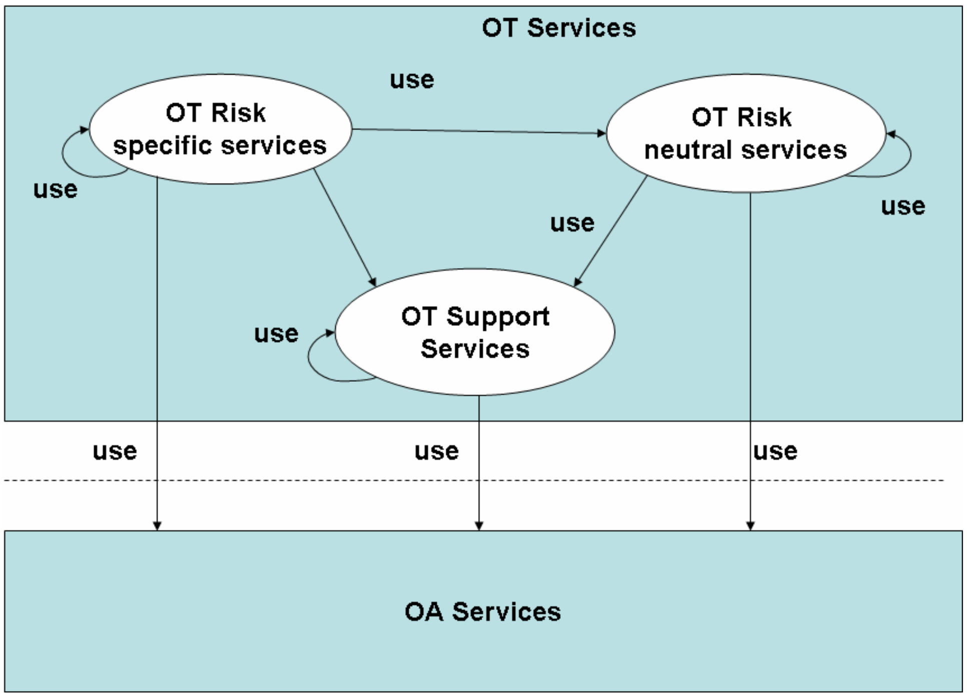


Figure 24: Example of OT Service sub-categories for the application domain of Environmental Risk Management

As an example, Table 4 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.

Service Name	Service Category	ISO 19119 Service Taxonomy	Section
Processing Services			
Statistical Calculation Service	OT Support	Geographic Processing Services	9.6.1
Geospatial Calculation Service	OT Support	Geographic Processing Services	9.6.3
Image Processsing Services	OT Support	Geographic Processing Services	9.6.4
Computer Algebra Service	OT Support	Geographic Processing Services	9.6.2
Simulation Management Services	OT Support	Geographic Processing Services	9.6.5
Workflow/ Task Management Services			
Sensor Planning Service	OT Support	Workflow/Task Management Services	9.6.6
Project Management Support Service	OT Support	Workflow/Task Management Services	9.6.7
Communication Service	OT Support	Workflow/Task Management Services	9.6.8
Calendar Service	OT Support	Workflow/Task Management Services	9.6.9
Reporting Service	OT Support	Workflow/Task Management Services	9.6.10

Table 4: List of OT Support Services

9.2.1.3 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 considered to be provided by so-called Human Interaction Components.

Component Name	User Interface to OA/OT Service	ISO 19119 Service Taxonomy
Map Viewer	Map Access Service	Geographic Human Interaction Services – Geographic Viewer
Diagram Viewer	Diagram Access Service	Geographic Human Interaction Services – Geographic Viewer
Catalogue Viewer	Catalogue Service	Geographic Human Interaction Services – Catalogue Viewer
Geographic Feature Editor	Feature Access Service	Geographic Human Interaction Services – Geographic Feature Editor
Service Editor	Service Chain Access Service	Geographic Human Interaction Services – Service Editor
Chain Definition Editor	Service Chain Access Service	Geographic Human Interaction Services – Chain Definition Editor

Table 5: List of 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.

However, as an informative addition, Table 5 provides a list of Human Interaction Components that may be required when building ORCHESTRA applications by usage of the above listed ORCHESTRA services. The column “ISO 19119 Service Taxonomy” provides a hint of the position of these components in the ISO 19119 Service Taxonomy.

9.3 Service Description Framework

A coarse description of the ORCHESTRA Services is provided in textual format according to the following template. The detailed specification of the services will be provided in respective annexes to the RM-OA document. These detailed specifications will also contain formal specification of the information objects that are referred to in the service operations (e.g. parameter types).

Name	Name of the Service Convention: All individual words in the service name are capitalized.	
Standard Specifications	Reference to an abstract or a concrete service specification according to a standardisation organisation (e.g. ISO, CEN, OGC,…) or to important reference material that will be taken into account when specifying the service or the contained service operations. In case of no adequate reference the field is set to “no corresponding standard known”	
Extension of	Reference to the name of the ORCHESTRA service(s) in the service interface model from which operations are directly inherited. Default entry is “RM-OA Service”. Note: This entry provides a first hint to a service hierarchy based on functional inheritance. The service hierarchy will be included in the RM-OA after the service specifications have been provided.	
Description	Human understandable description of the service	
Service Operations		
servOper1	Description	Human understandable description of the service operation 1. Convention: 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.
	Input	Parameters provided by the requestor
	Output	Return parameters
…		
servOperN	Description	Human understandable description of the service operation n
	Input	Parameters provided by the requestor
	Output	Return parameters
Example usage		
Comments		

Table 6: Service Description Framework

9.4 OA Info-Structure Service Descsriptions

9.4.1 RM-OA Service

Name	RM-OA Service	
Standard Specifications	Relevant concepts to be taken from: - WS-I Basic Profile (http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html) - OGC Web Services Common Specification (OGC 05-008) - UDDI Version 3.0.2 Specification (http://uddi.org/pubs/uddi_v3.htm) - Web Notification Service	
Extension of	none	
Description	The RM-OA Service provides an abstract interface for all ORCHESTRA Services. This means that its operations are inherited by all ORCHESTRA Services. They may be extended or adapted according to the context of a specific ORCHESTRA Service. The RM-OA Service furthermore provides descriptions of service functions for which a common architectural approach is required for all ORCHESTRA Services.	
Service Operations		
getCapabilities	Description	Informs the requestor of the capabilities of an ORCHESTRA Service Instance (OSI).
	Input	
	Output	A document containing service meta-information.
Example usage	The RM-OA 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 list of operations will be enhanced during the course of the specification of the OA and OT Services. A candidate for the enhancement is a common support for synchronous and asynchronous interaction modes of a service. The contents of the service meta-information will be defined as part of the specification of the OAS-MI for services in a future annex of the RM-OA.	

9.4.2 Feature Access Service

Name	Feature Access Service
Standard Specifications	ISO 19109 for feature modelling (abstract) SQL and OQL standards for feature query (abstract and implementation) OGC Simple Feature for CORBA; SQL; OLE/COM (implementation) OGC Web Feature Service (implementation spec) OGC/ISO GML for feature encoding
Extension of	RM-OA Service

Description	The Feature Access Service allows interoperable read and write access on feature instances available in an OSN. Features provided by a Feature Access Service are instances of a certain feature type (see the definition of ORCHESTRA features in section 8.2). Moreover, the Feature Access Service offers information about - the feature types it is capable to provide or support, - the supported encoding(s) to transfer requested or submitted feature data, and - the query mechanism it supports for filtered feature access. It allows queries to select certain features based on their type, certain attribute values, and/or their spatial and temporal extent. The selection statement is encoded using a query language that supports all this functionalities (e.g. SQL including spatiotemporal statements). Within an OSN at least one common feature query language has to be defined, which can be used by the ORCHESTRA services. The query language shall support spatiotemporal queries but also queries based on an object-structured model, in particular database structure according to an OAS. SQL3 can be seen as one example for such a query language. Furthermore, the query language shall also support the expression of simple arithmetic operations (e.g. count, maximum, minimum), e.g. to know about the number of feature instances that conform to a certain filter criteria without having to retrieve the feature instances themselves. The Feature Access Service supports the update of existing feature instances and the creation of new feature instances (also referred to as a transactional feature service). However, it does not allow defining new feature types (i.e. feature types that are not already supported by the respective feature service). For the updating of feature instances an appropriate synchronisation mechanism (e.g. locking) is required to prevent access to inconsistent data sets. Feature instances are identifiable by a Unique Identifier (UID) that is unique with respect to at least one OSN. If a Feature Access Service is used to create a new feature instance it will also create an appropriate UID for this feature instance. The Feature Access Service may be implemented based on cascading services. An instance of a Feature Access Service would integrate various feature access service instances and support an integrated – and by use of appropriate schema mappings (see section 9.5.6) – access to the features of the underlying feature access services.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Feature Access Service instance.
	Input	none
	Output	list of supported feature types for the access, the service options (e.g. transaction support, for which feature types), the supported filter mechanisms.
describe-FeatureType	Description	Generates a description of feature types serviced by a Feature Access Service instance. The descriptions define how a Feature Access Service instance expects feature instances to be encoded on input and how feature instances will be generated on output.
	Input	type name, version, service
	Output	document that may be used to validate feature instances generated by the Feature Access Service in the form of feature collections on output or feature instances specified as input for transaction
getFeature	Description	Allows retrieval of features.

	Input	version, service, query statement, maximum number of features to be returned (optional)
	Output	feature collection (conditional), reference to a feature collection (conditional), information about feature collections, e.g. number of feature instances that conform to the query statement (conditional)
setFeature	Description	Supports the update and deletion of existing feature instances as well as the creation of new feature instances carried out in individual transactions. However it does not allow defining new feature types (see section 8.2) (i.e. feature types that are not already supported by the respective feature service instance). For the updating of feature instances an appropriate synchronisation mechanism (e.g. locking) is required to prevent access to inconsistent data sets. The *setFeature* operation is used to describe data transformation operations that are to be applied to feature instances. A Feature Access Service may process a *setFeature* operation directly.
	Input	feature collection that defines a <Transaction> list. The list may contain one or more <Insert>, <Update>, or <Delete> elements that describe which and which way feature instances are to be created, modified or destroyed. features instances identified by UIDs.
	Output	completion status, new feature instances including UIDs (conditional)
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 *getFeature* request for the specified area and data 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.	
Comments	Aspects of access control are handled by the Feature Access Service by using the Authorisation Service (see section 9.4.14) and the Authentication Service (see section 9.4.15). An informative usage pattern is given in section 9.7.1). For transactional operations, at the end of a transaction, the Feature Access Service shall apply transaction semantics appropriate to the particular system used to persistently store features. For example, if the data store is a SQL based relational database, then a commit will be executed at the end of the transaction (or a rollback should the transaction fail). Any locks maintained by the Feature Access Service for the duration of the transaction shall be released after the end of the transaction.	

Table 7: Description of the Feature Access Service

9.4.3 Map Access Service

Name	Map Access Service
Standard Specifications	ISO – ISO/DIS 19128 - Web Map Service – ISO/CD 19136, ISO 19107, ISO 19108, ISO/PRF 19118, ISO/FDIS 19123 - Geography Markup Language – ISO/FDIS 19117 - Portrayal – ISO 19119:2005 - Geographic Information - Service Architecture

	OGC – Web Map Service (WMS) – Styled Layer Descriptor (SLD) – Web Feature Service (WFS) – Web Coverage Service (WCS) – Filter Encoding (Filter) – OpenGIS Abstract Specification (AS). Topic 12: OpenGIS Service Architecture	
Extension of	RM-OA Service	
Description	The Map Access Service symbolizes geographic "raw data" provided in vector, grid or raster formats, according to cartographic visualisation requirements. The output of this service is a map in either raster or vector format. It allows the integration of a cartographic interface (optional), which admits the definition of an individual feature symbology, and the integration of data from other OA services like: - OA Info-Structure Feature Access Service - OA Info-Structure Coverage Access Service The service mainly provides read access to the requestor. Only the optional "addStyles" operation allows the requestor to upload user-defined styles and layers which will be stored on the server.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Map Access Service instance.
	Output	A document containing a list of supported operations and mapping parameters (e.g. bounding box, layer information including supported feature types)
getMap	Description	Returns a map in the specified format which is specified in the format parameter by the client application.
	Input	version, request, layers, overlay properties, layer precedence, styles, coordinate reference system, bounding box, width, height, format, transparent (optional), background colour (optional), time (optional), elevation (optional)
	Output	map of the spatially referenced information layer requested, in the desired style, and having the specified coordinate reference system, bounding box, size, format and transparency output format can be either raster (e.g. png, jpeg) or vector (e.g. SVG, pdf)
getFeatureInfo	Description	Asks for information about particular features shown on a map.
	Input	version, request, query layers, information format, feature count (optional), coordinates of the image coordinate system, e.g. the result of a mouse-click event (I, J)
	Output	response according to the requested information format The nature of the response is at the discretion of the service provider, but it shall pertain to the feature(s) nearest to (I,J)
describeLayer	Description	Returns a description containing feature/coverage-type information for a named layer.

	Input	version, request, layers
	Output	Document containing a list of layer definitions
getLegend-Graphic	Description	Acquires legend symbols
	Input	version, request, layer, style (optional), feature type (optional), rule (optional), scale (optional), style information (optional), width (optional), height (optional), format
	Output	legend graphic. The output format can be either raster (e.g. png, jpeg) or vector (e.g. SVG, pdf)
getStyles	Description	Retrieves user-defined styles
	Input	version, request, layers
	Output	Document containing a list of style definitions
addStyles	Description	Stores user-defined styles and user-defined layers
	Input	version, request, mode, styleInformation
	Output	
Example usage	A requestor accessing this service wants to create a map that shows a hydrological network on top of a soils layer. The data for the hydrological network resides on an OA Feature Access Service, and the soils layer on an OA Coverage Access Service. The requestor now invokes a *getMap* operation by passing a cartographic description, which defines the location of the data and the cartographic symbolization of each layer. The response of the service will be a map provided in the requested format.	
Comments	This service does not (!) provide a human interface like the Map Viewer as one of the human interaction components, whereas a Map Viewer could act as a requestor to this service. This Map Access Service can be seen as an intermediate service in a service chain in between of a service providing raw data (e.g. Feature Access Service) on the one side, and a Map Viewer of the human interaction components on the other side.	

Table 8: Description of the Map Access Service

9.4.4 Diagram Access Service

Name	Diagram Access service
Standard Specifications	no corresponding standard known
Extension of	RM-OA Service
Description	The Diagram Access Service transforms numerical data into a visual representation, i.e. diagrams either in raster or vector format. A diagram may contain the common visualisation of several datasets with different legends in form of several overlaid curves. The purpose of this service is multi-fold. It could be used for visualizing the result of a statistical analysis. It could also create diagrams to be used in a map, but the integration (placement and symbol definition) of the diagrams will then be handled by the Map Access Service. Due to the similarity to the Map Access Service (graphic rendering, but without any geo-reference) the service operations are quite similar to the ones of the Map Access Service. In analogy to the cartographic interface of the Map Access Service there are diagram style operations to define the symbolization and the legend information.

	The selection of the data sources that will be used for the diagram rendering is performed by using the Feature Access Service (see section 9.4.2) based on the service pattern that will be defined in section 9.5 . The Diagram Access Service expects the data to be rendered in tabular format.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Diagram Access Service instance.
	Input	
	Output	list of available diagram types and visualization parameters
getDiagram	Description	Creates a diagram by passing the diagram parameters
	Input	version, request, width, height, format, datasets in tabular format, location of the diagram style interface
	Output	visual representation of data, can be either in raster or vector format.
getDataValues	Description	Retrieves the numerical values of the data points of selected curves in a given time period.
	Input	Identification of a curve in a diagram, time period
	Output	Tabular data sets including context information (e.g. units) associated to the selected curves
Example usage	It is interesting to combine the Map and the Diagram Access Service in order to allow diagrams to be visualised in the context of a map. There is a simple solution to allow this integration on the client side. The requestor first requests a map, then the user is able to click on a feature that contains some attributes that can be visualized in a diagram. The requestor calls the *getFeatureInfo* operation of the Map Access Service and retrieves the attribute values of the feature. The requestor passes those values to the Diagram Access Service (using the diagram style interface) and retrieves the resulting diagram. An extended version of the Diagram Access Service will support the integration to happen on the server side.	
Comments		

Table 9: Description of the Diagram Access Service

9.4.5 Document Access Service

Name	Document Access Service
Standard Specifications	OASIS Open Document Format for Office Applications (OpenDocument) v1.0 http://www.oasis-open.org/specs/index.php#opendocumentv1.0
Extension of	RM-OA Service

Description	The Document Access Service supports the access to a document of any type (textual documents, images, …). A document can be stored locally in the Document Access Service's document store, or it can be retrieved on the fly from a source system (e.g. from an HTTP server). Whether a document should be stored locally or remotely depends on the requirements of the requestor (up-to-date-ness vs. reliability). Removal and Update (versioning) is only supported, if a document is stored locally. An implementation of the Document Access Service must support at least one access protocol (file, ftp, http, ...).	
Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of a Document Access Service.
	Input	none
	Output	list of supported access protocols and a list of supported MIME-types
getDocument	Description	Returns and optionally converts the content of a document. Whether a document is retrieved from the local document store or directly from a source system's data source is denoted by document reference in OAS_DocumentDescriptor. If the document reference points to an external data source, *getDocument* should delegate the call to *downloadDocument*. If the MIME type of the output format is specified, this operation delivers the document already in the specified format, e.g. by using the Format Conversion Service (see section 9.5.5). An OAS_DocumentDescriptor instance must have been created previously by the Annotation Service.
	Input	OAS_DocumentDescriptor, MIME type of desired output format (optional), version number of document (optional)
	Output	document content in native format (conditional)
setDocument	Description	Registers a new document, stores the document data locally in the document store. Storage of documents in source system data sources is not supported.
	Input	instance of OAS_DocumentDescriptor, document content in native format
	Output	none
updateDocument	Description	Updates an existing document in the document store.
	Input	OAS_DocumentDescriptor, document content in native format, comment (string) (optional)
	Output	Version number of document (Integer)
removeDocument	Description	Removes an existing document from the document store. Deleting of documents from source system data sources is not supported.
	Input	instance of OAS_DocumentDescriptor
	Output	not applicable

download Document	Description	Retrieves a piece of data from an arbitrary source system's data source (file system, web, stream, data base, ...).
	Input	OA_ResourceLocator that points to the location of the document
	Output	instance of OAS_File that has been populated with the document data (conditional)
Example usage	An end-user wants retrieve a document in his preferred format. A system user wants to integrate documents that are stored in diverse data sources at different locations. In order to assure reliability, he decides to store a local copy of the documents in the Document Access Service's document store.	
Comments	The registration of a document is done through the Catalogue Service (see section 9.4.10).This requires adequate meta-information (see the respective OAS-MI for the purpose discovery) possibly as an extension of 'OAS_DocumentDescriptor'.	

Table 10: Description of the Document Access Service

9.4.6 Source System Access Service

Name	Source System Access Service	
Standard Specifications	no corresponding standard known	
Extension of	Feature Access Service	
Description	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 very heterogeneous nature and contain information in a variety of types and formats. A source system in this sense means that it provides some functionalities (provision of data and operations in terms of services). Consequently, accessing a source system means to use its provided operations. However, usually this access is not offered in a way that may be directly used by ORCHESTRA services. Instead, as ORCHESTRA feature type, a source system is represented by a source system descriptor (OAS_SourceSystemDescriptor, see section 8.4.4.4). The Source System Access Service provides transparent and ORCHESTRA conformant access to the features of a source system and invocation of its services. Thus, the features of a source system can be accessed throughout the ORCHESTRA network without specific knowledge of the source system itself, nor are any other services (e.g. the Feature Access Service) required in the retrieval of the data, once an OSI of the Source System Access Service has been found (possibly through a catalogue lookup). In this sense the Source System Access Service can be seen as a wrapper around an already existing source system. Consequently, every source system, being an existing source system or a source systems whose native interface is the interface Source System Access Service, can be accessed in a common way.	
Service Operations		
getCapabilities (from RM_OA Service)	Description	Provides information about the capabilities the Source System Access Service provides. In case of an already existing source system, these are the capabilities of the source system, the implementer wants to make publicly accessible.
	Input	

	Output	Description of the capabilities of this Source System Access Service. This includes: - List of supported conceptual schema languages (CSL), which the source system's schema can be provided in (e.g.: OWL, UML, XMLSchema, RDF). - List of supported access languages, which are used to formulate a query for feature retrieval (used in *getSourceSystemData* and *setSourceSystemData*. e.g.: XPath, XQuery) - Possible types of bounding boxes that can be used to narrow the feature request in *getSourceSystemData* - Information about the meta-information schema that describes the service types of this source system (e.g. the OAS-MI for service discovery). - Type of the source system data schemas (e.g. OAS, relational data base, object-oriented data base, spreadsheet)
getSchema	Description	Returns the schema of the selected source system specified by its name. The CSL in which the schema must be provided has to be specified. Within the schema feature types are used and referenced as primitives.
	Input	OAS_SourceSystemDescriptor, schema name, conceptual schema language (CSL)
	Output	Selected schema of the selected source system in the requested CSL.
getOntology	Description	Returns the ontology of the source system.
	Input	OAS_SourceSystemDescriptor
	Output	Source system ontology.
getFeatureTypes	Description	Retrieves the list of feature types, including their description, which are accessible at the selected source system.
	Input	OAS_SourceSystemDescriptor, feature type name(optional) If no feature type name is given the operation is carried out as if all available feature types have been provided.
	Output	one or more feature type descriptions (conditional)
getServiceTypes	Description	This operation gets the list of service types which are accessible at the selected source system.
	Input	OAS_SourceSystemDescriptor, name of the service type
	Output	Returns the service types accessible at the source system and their access method (e.g. ORCHESTRA service, URL of a SOAP operation, reference Java RMI method)
callService	Description	Invokes the specified service the selected source system provides.
	Input	OAS_SourceSystemDescriptor, name of the service to be invoked, parameters for the specified service (optional)
	Output	Return value(s) of the service that has been invoked (conditional)
getSource SystemData	Description	Retrieves features from the selected source system.
	Input	OAS_SourceSystemDescriptor, name of the schema to navigate in,

		query, access language (optional)
	Output	one or more OA_Feature(s) (conditional)
setSource SystemData	*Description*	Updates, adds or removes the selected feature in the selected source system. Is always carried out in the context of a transaction. Either the whole feature is written (inserted, updated or deleted), or it is left unchanged. The operation must guarantee to commit the changes on a success or rollback the changes on a failure.
	Input	OAS_SourceSystemDescriptor, name of the schema to navigate in, query, access language used for the query, feature containing the values to be set for the specified feature, operation that should be carried out on the feature
	Output	success or failure indication
Example usage	The list of schema languages can be used to select one specific schema type and retrieve the schema of the source system with *getSchema*. Then a feature can be retrieved formulating a request in a provided access language and invoking the *getSourceSystemData* operation.	
Comments	If the schema type of the source system is not OAS, the Source System Access Service implicitly transforms the data to ORCHESTRA feature instances, e.g. using the Schema Mapping Service (see section 9.5.6) if respective mapping rules have been specified.	

Table 11: Description of the Source System Access Service

9.4.7 Formula Access Service

Name	Formula Access Service	
Standard Specifications	MathML 2.0 specification (http://www.w3.org/Math/)	
Extension of	RM-OA Service	
Description	The Formula Access Service allows the user to access to feature instances of type “formula” in an OSN. A formula will be specified in the RM-OA Information Viewpoint as a generic information type in section 8.4.4 an will e.g. include information like independent and dependent variables, their units, its source (reference for the formula, e.g. the article where the formula is from etc.) and any additional comments. In addition to the formula function, the formula type specification will also include context information (e.g. semantics, usage conditions) such that the retrieved formula may be used together with other services, in particular Processing Services of the service category OT Support (see section 9.2.1.2). Also the user should be able to find which dependent and independent variables the formula applies to. The processing, execution and calculation using the retrieved formula is handled in other services.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Formula Access Service instance.
	Input	none
	Output	list of formula types, names and formula formats supported The format of a formula is the standard (e.g. MathML) used to spec-

		ify a formula so that it can be used within other services and the correct independent and dependent variables are used when evaluating the formula.
getFormula	Description	Accesses a given formula in the format required by other services (e.g. the Computer Algebra Service, see section 9.6.2) by specifying its name and further information that will be specified in the "formula" feature type in section 8.4.4.
	Input	name
	Output	function, independent variables, dependent variables, source, comments
createFormula	Description	Create a given formula in the format used by other services. A formula is defined by specifying its name, its function, information about its independent and dependent variables, its source and any additional comments.
	Input	name, function, independent variables, dependent variables, source, comments
	Output	
setFormula	Description	Sets the attributes of a given formula specified by its name.
	Input	name, function, independent variables, dependent variables, source, comments
	Output	name, function, independent variables, dependent variables, source, comments
Example usage	Client retrieves formula and passes it to the Computer Algebra Service (see section 9.6.2) in order to execute it against some data. Client creates new formula to store within an OSN.	
Comments	A method for defining functional forms of any equations needs to be implemented within these processes. Functional form is the form of the equation, e.g. z=a sin x + b cos y is a functional form with dependent variable z and independent variables a, b, x and y. It is not known if there is a standard representation of functional forms but it is thought that different software uses different formats. There exists a XML language MathML that can be used to express mathematical constructs. A method for expressing the independent and dependent variables of a function needs to be implemented.	

Table 12: Description of the Formula Access Service

9.4.8 Coverage Access Service

Name	Coverage Access Service
Standard Specifications	ISO 19109 for feature modelling (abstract) ISO/TR 19121 Imagery and gridded data (abstract) OGC Abstract Spec , Topic 6 The coverage type (abstract) OGC Web Coverage Service (implementation spec)
Extension of	RM-OA Service Feature Access Service
Description	The Coverage Access Service allows interoperable access and transactions on n-

	dimensional coverages stored in a database. A coverage (see also note 2 in section 8.9.2) is a digital geospatial information representing space-varying phenomena. It is specified as an extension of the OMM_FeatureType in section 8.4.4.5. It supports queries to select certain parts of a coverage based on their type, certain attribute values, and/or their spatial and temporal extent. The selection statement is encoded using a query language that supports the corresponding filter mechanisms.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Coverage Access Service instance.
	Input	none
	Output	list of supported coverage types (i.e. specialised feature types) for the access, the service options (e.g. transaction support, for which types), the supported query language/filter mechanisms.
describe-CoverageType	Description	Generates a description of coverage types serviced by a Coverage Access Service instance. The descriptions define how a Coverage Access Service instance expects coverage instances to be encoded on input and how they will be generated on output.
	Input	type name, version, service
	Output	list of documents The document(s) presented by the *describeCoverageType* request describe the schema of coverage instances provided by the Coverage Access Service
getCoverage	Description	Allows retrieval of coverages.
	Input	version, service, query statement
	Output	coverage or reference to a coverage (conditional),
setCoverage	Description	Supports the update of existing coverage instances and the creation of new coverage instances carried out in terms of individual transactions. However it does not allow defining new coverage types. See description of *setFeature* in Feature Access Service.
	Input	A coverage that defines the values of the coverage to be created or updated. In terms of an update the coverage might define only a spatial and/or temporal subset covered by the considered coverage instance. As coverage instances are feature instances, they are identified by UIDs (see section 7.2.1).
	Output	completion status, new coverage instance including UID (conditional)
Example usage	A client makes a request for a digital elevation model of a specified area and a coverage is returned. This for instance then can be used for spatial analysis purposes, e.g. calculation of catchments.	
Comments		

Table 13: Description of the Coverage Access Service

9.4.9 Sensor Access Service

Name	Sensor Access Service	
Standard Specifications	Draft services and sensor schemas of the Sensor Web Enablement working group of OGC Web Notification Service	
Extension of	Feature Access Service	
Description	The Sensor Access Service allows interoperable access to data that is continuously collected by a sensor. It generally behaves as a feature access service, whereas it does obviously not allow writing features. The Sensor Access Service allows to query the sensor measurement data that is modelled following the ORCHESTRA feature model. In addition to a Feature Access Service a Sensor Access Service allows to use a notification mechanism. A service user can define a notification that should be sent when certain conditions are met, for instance if data for a certain date is accessible, a measured value of a sensor exceeds a certain threshold, etc. The ways a notification is passed can be specified by the client.	
Service Operations		
getCapabilities (from RM_OA Service)	Description	Provides information about the capabilities the Sensor Access Service provides
	Input	
	Output	description of the capabilities of this Sensor Access Service.
getFeature (from Feature Access Service)	Description	Retrieves the OAS_SensorDescriptor of the Sensor Access Service.
	Input	version, service, query statement, maximum number of features to be returned (optional)
	Output	Sensor descriptor
getSchema	Description	Retrieves the description of the requested (network of) sensor(s) in the specified schema language. (e.g.: SensorML, OGC 04-019r2)
	Input	Sensor descriptor, schema language type
	Output	The schema of the specified sensor..
addNotification	Description	Adds a notification for a specified event (an event happens when a new sensor value is available that fulfils the notification condition) at the specified sensor. The notification remains active until it is removed.
	Input	sensor which the notification has be to attached to, condition, arming mode, fire counter, name, contact, notification text (optional)
	Output	Notification identifier
alterNotification	Description	Changes the provided notification to new provided values (e.g. contact information)
	Input	Notification identifier, condition (optional), arming mode (op-

		tional), fire counter (optional), name (optional), contact (optional), notification text (optional)
	Output	
remove Notification	Description	Removes the specified notification from the Sensor Access Service so it no longer fires.
	Input	Notification identifier
	Output	
getFeatureTypes (from Feature Access Service)	Description	Retrieves the list of feature types, including their descriptions, which are accessible at the Sensor Access Service.
	Input	specific feature type (optional) If no parameter is given the operation is carried out as if all available feature types have been provided.
	Output	one or more feature type descriptions (conditional)
getSensorData	Description	Retrieves features from a specific sensor.
	Input	Sensor descriptor, start time, end time, timeout
	Output	zero or more OAS_Feature(s) (conditional)
configure Sensor	Description	This operation is used to configure the sensor. Configuration might also include position and orientation settings (e.g.: if the sensor is mounted on a movable device).
	Input	
	Output	
Example usage	A client accesses water level measurements from a river in order to monitor flood risks.	
Comments	The functional boundary to the Sensor Planning Service (see section 9.6.6) still has to be defined. Simulation results could also be understood as sensor data (virtual sensors) and then accessed through the Sensor Access Service (Wytzisk 2003).	

Table 14: Description of the Sensor Access Service

9.4.10 Catalogue Service

Name	Catalogue Service	
Standard Specifications	OpenGIS™ Abstract Specification Volume 13: Catalog Services OpenGIS™ Catalogue Service Specification 2.0 (Doc n° OGC 04-021r2)	
Extension of	RM-OA Service	
Description	The Catalogue Service supports the ability to publish, query and retrieve descriptive information (meta-information) for resources (i.e. data and services) and instances of feature types that are referred to by extensions of the OMM_FeatureType in section 8.4.4, such as documents, schemas, source systems, coverages, dictionaries, equations, models,… The Catalogue Service is not tied to a particular schema of a meta-information standard (e.g. ISO 19115), instead it supports all application schemas for meta-information (OAS-MI) that are designed according to the rules of the OMM (see section 8.5.3). Pre-defined OAS-MIs for identified "purposes" and resource types (data and services) are specified in an annex to the RM-OA. 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).	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Catalogue Service instance
	Input	Optional identifier(s) of requested parts of the complete service meta-information document
	Output	Document containing the supported schema language, the specific protocol binding, and on other details of that protocol binding. Other document contents depend on the types of data defined by the specific application profile, and on other details of that profile.
query	Description	Allows clients to ask an OSI of a Catalogue Service to execute a query (one executable command) that searches the registered meta-information and produces a result set containing (zero or more) references to all the registered resources that satisfy the query. The server may maintain the result set for subsequent retrieval requests.
	Input	Specifications of the query expression and of the meta-information elements to be returned
	Output	Number of items in result set, and/or selected meta-information for some or all of the result set. The client can specify the maximum number of records for which meta-information is returned. When meta-information return is requested, the service implementation shall first sort the result set as specified by the client. Most of the meta-information returned depends on the meta-information requested and on the types of data defined by the specific Application Profile.

present	Description	Allows clients to retrieve selected meta-information for some or all of the resources referenced in a specific previous result set or a list of resource identifiers. This operation can be used repetitively to retrieve more of the result set, each time retrieving meta-information for a maximum number of the resources listed, starting at a specified position.
	Input	Specifications of sorting and of meta-information to be returned, optionally including maximum number of records for which meta-information is to be returned
	Output	Metadata document containing selected meta-information for some or all of the specific result set, after it is sorted as specified by the client. Most of the meta-information returned depends on the meta-information requested and on the types of data defined by the specific Application Profile
describe RecordType	Description	Allows clients to retrieve type definition(s) used by meta-information of one or more registered resource types
	Input	List of identifications of requested record types and of desired formats (optional)
	Output	Type definition document containing definition(s) of type(s) used by the meta-information of one or more registered resource types. This type definition shall include the structure (schema definition), queryables, element sets, and formats of the meta-information used for one or more registered resource types. The contents of the result of this operation depend on the types of meta-information that can currently be used by registered resources.
getDomain	Description	Allows clients to retrieve the domain (allowed values) of a meta-information property or request parameter at the time the request is invoked. The returned information may be static domain information, but may also be dynamic in that the allowed values are determined at runtime. The operation does a *best attempt* at returning information about a meta-information property or request parameter.
	Input	Names of one or more requested meta-information properties or request parameters associated with one or several resource types.
	Output	Descriptions of domains of one or more requested meta-information properties or request parameters
transaction	Description	Allows clients (here: a resource provider) to request a specified set of “insert ”, “update”, and “delete” actions on the content managed by a Catalogue Service instance. The uniqueness of the meta-information ID must be managed by the Catalogue database implementing the catalogue information model
	Input	Specification of set of “insert”, “update”, and “delete” actions (at least one action), plus an optional identifier.
	Output	A summary of the transaction results that identifies newly created entries when applicable. Most contents of the result depend on the types of data defined by the specific protocol binding and Application Profile.
harvestResource	Description	Allows a user to request that a catalogue service attempts to retrieve a resource from a specified location (e.g a URL), and to optionally create one or more entries in the catalogue for that resource. A harvest attempt may occur periodically if an interval is specified.

	Input	A request message containing the source of the resource (e.g. an URL) to be harvested
	Output	An acknowledgement that a harvestRequest has been received and validated (if a responseHandler is specified) or a summary of the harvest results that identifies newly harvested records (if a responseHandler is not specified). Most contents of the result depend on the types of data defined by the specific protocol binding and Application Profile.
navigation Edges	Description	Returns the set of possible navigation path available from the current focus
	Input	The position in the navigation graph to navigate from.
	Output	Array of edges that are starting from the given reference node.
navigate	Description	Returns the node that follows the given reference node following the given edge
	Input	The position in the navigation graph to navigate from and the edge to follow for navigation
	Output	The new node after the navigation is completed.
Example usage	In the context of a risk assessment scenario for earthquakes in a given region, a user uses the Catalogue Service to retrieve the information which maps and documents are available to this topic in the given region.	
Comments	The Catalogue Service supports the concept of cascaded catalogues. This means that a instance of the Catalogue Service may expand a query to other instances of Catalogue Service that have been previously registered and assembles the query results into one query result to the caller. If the cascaded catalogues are structured according to different OAS-MI, the root Catalogue Service Instance may use the Schema Mapping Service (see 9.5.6) in order to transform the query results into one harmonised query result. This process is, however, transparent to the requestor of the query operation. For an implementation specification in a Web environment, two profiles could be used: one profile using ebRIM, another one using the ISO 19115/19119 standards. If the ISO 19115/9119 standard is used, a profile can be defined with or without extensions (the way to define extension is foreseen by the standard). By combination of the Catalogue Service with the Query Mediation Service (see section 9.4.11) a so-called semantic catalogue component may be built (see OA usage pattern described in section 9.7.3).	

Table 15: Description of the Catalogue Service

9.4.11 Query Mediation Service

Name	Query Mediation Service
Standard Specifications	OpenGIS Catalogue Services Specification (project document: OGC 04-021r2 – version 2.0 - 2004-05-11) W3C-Ontology Web Language (OWL) OWL-QL (http://ksl.stanford.edu/projects/owl-ql/) RFC 2651 Common Indexing Protocol (CIP) http://www.networksorcery.com/enp/rfc/rfc2651.txt)

Extension of	RM-OA Service	
Description	The Query Mediation Service supports other services (especially the Catalogue Service) in the processing of queries against heterogeneous source systems (e.g. catalogues accessible through other Catalogue Service Instances in case of cascaded catalogues). 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 very heterogeneous nature (e.g. catalogues, ontologies, XML repositories, relational data bases, word processing files,…) that require drastically different query mechanisms (e.g. CQL, X-Query, SQL, text matching, ORCHESTRA Source System Access Service Interface). Note that this heterogeneity of interfaces goes far beyond the common ODBC/JDBC mediator models applied in the context of relational data bases. The efficiency of the query mediation procedure is significantly improved if meta-information about the source systems is available in a catalogue (see Catalogue Service, section 9.4.10). The more characteristics of a source system (e.g. schema information, available feature types) are registered in a catalogue, the better-tuned sub-queries may be generated. Furthermore, query mediation might exploit semantic information if specified (semantic mediation). This requires that the meta-information within a catalogue refers to an ontology in order to support the generation of sub-queries by concept expansion and concept resolution. Note 1: The result set might comprise meta-information on data or services, but also features or feature properties themselves. Note 2: The current specification just refers to select queries. Update and create queries will be considered in future versions.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Query Mediation Service instance.
	Input	identifier(s) of requested parts of the complete service meta-information document (optional)
	Output	query language supported as input (syntactical or semantic query language), the type of sub-query mechanisms supported, type of semantic query and assembly mediation technique supported (see the Inferencing Service, section 9.5.8).
createSub Queries	Description	Creates sub-queries to individual source systems out of a given “select”-query. The sub-queries are built according to the type of source systems against which the sub-queries will be performed. Note: In case of semantic mediation, an ontology query language like RQL should be used to specify the input query from which the sub-queries are derived. Such a query language has the ability to infer knowledge which is not explicitly described in the meta-information.

		Examples are 1) concept expansion: extract all sub-concepts of the queried term from the global ontology, or 2) concept resolution: extract a set of unique values for each of the queried terms
	Input	input query, language of input query, list of source system identifiers (instances of the respective feature type) and related source system types, reference to ontologies to be used, inference technique to be applied (e.g. concept expansion or concept resolution)
	Output	list of sub-queries for each specified source system
perform SubQuery	Description	Performs an individual sub-query for a given source system. The sub-query must fit to the specified source system type. Note that this operation just performs one sub-query. The order of the performance of several sub-queries that have resulted from the createSubQueries operation is a decision of the requestor.
	Input	input sub-query, source system identifier, maximum number of records for which data is returned (optional), sorting criteria (optional)
	Output	Result set containing selected references to resources, and/or selected data for some or all of the result set. Note: When requested, the service implementation shall first sort the result set as specified by the client.
assemble Results	Description	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. For example, in case of a multi-language environment, the Thesaurus Access Service may be called. In case of geospatial information mediation, query results may contain fragments of geospatial information with possibly different reference systems, style definitions or different (raster or vector) formats. In case of temporal information mediation, query results may contain fragments of time references with different time zones or formats. In case of semantic mediation, specific inference techniques may be applied such as concept merging or aggregation.
	Input	input result sets, assembly service to be applied, reference to an ontology specification to be used (optional), type of type of inference technique to be applied (optional)
	Output	result set providing a common logical view (according to the assembly service provided) of the result sets provided as input.
Example usage	An OSC called “Semantic Catalogue” may be built by combining the Catalogue Service and the Query Mediation Service (see section).	
Comments	The operation *assembleResults* may also be specified as a distinct Content Mediation Service. 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.	

Table 16: Description of the Query Mediation Service

9.4.12 Ontology Access Service

Name	Ontology Access Service	
Standard Specifications	W3C-Ontology Web Language (OWL) OWL-QL (http://ksl.stanford.edu/projects/owl-ql/) RDF Query Language (RQL) http://139.91.183.30:9090/RDF/RQL/	
Extension of	RM-OA Service	
Description	The Ontology Access Service supports the read access to the specification of a logical ontology (see section 8.7.1.2) and to export or import a complete specification of a logical ontology into an ontology store. Furthermore, it contains operations to manage an ontology store by providing list and delete operations. Note: It is assumed that the maintenance of Ontologies is performed by dedicated tools (e.g. Protégé). Future version might also include update or create operations as part of this service.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of an Ontology Access Service instance.
	Input	
	Output	list of formal languages in which the specification of an ontology may be provided (e.g. OWL)
deleteOntology Spec	Description	Deletes the specification of an ontology stored in the ontology store.
	Input	name of an ontology specification in the ontology store.
	Output	
importOntology Spec	Description	Imports a specification of an ontology from a given file and store it into the ontology store. Note: For simplicity, partial import is not considered.
	Input	URL of an ontology specification.
	Output	
exportOntology Spec	Description	Allows to retrieve the specification of an ontology stored in the ontology store and write it to a file
	Input	URL of an ontology specification, language in which the ontology is to be provided
	Output	
listOntologies	Description	lists all ontologies including their meta-information
	Input	
	Output	name of all ontology specifications in the ontology store, including their creator and creation date.
Example usage		
Comments	Further operations for the navigation within an ontology will be added in future.	

Table 17: Description of the Ontology Access and Mapping Service

9.4.13 User Management Service

Name	User Management Service	
Standard Specifications	Lightweight Directory Access Protocol (LDAP) OASIS eXtensible Access Control Markup Language (XACML) (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)	
Extension of	RM-OA Service	
Description	The User Management Service allows authorised users (e.g. system administrators) to add new users and user groups to an OSN and to remove them as well. User groups are currently defined explicitly by the system administrator by assigning users to a group. A future version might define other policies for user groups, e.g. implicitly defined user groups according to some user attributes. This functionality reflects the way how user access to resources (feature instances or service instances) is granted in an OSN: - Access control is performed by one or more instances of the Authorisation (see section 9.4.14) and Authentication Service (see section 9.4.15). The Authorisation service can assign access rights to users and user groups. - Once being created, users can grant other users access to their resources by means of the Authorisation Service.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a User Management Service instance.
	Input	
	Output	attributes required when creating a new user.
listUsers (Groups)	Description	List all users (user groups) currently registered.
	Input	
	Output	List of user (group) identifications and associated attribute/value pairs.
createUser	Description	Creates a new OSN user
	Input	user identification, initial user attribute/value pairs (e.g. user name, address, email, phone, fax, preferred language)
	Output	success or failure indication
setUser Attributes	Description	Sets the attributes of an existing user
	Input	user identification, user attribute/value pairs (e.g. user name, address, email, phone, fax, preferred language)
	Output	success or failure indication
getUser Attributes	Description	Gets the attributes of an existing user
	Input	user identification

	Output	user attribute/value pairs (e.g. user name, address, email, phone, fax, preferred language, list of user groups the user belongs to)
deleteUser	Description	Deletes an OSN user
	Input	user identification
	Output	success or failure indication
addUser	Description	Adds a user to a group of users
	Input	user identification, user group identification
	Output	success or failure indication
removeUser	Description	Deletes a user from a group of users
	Input	user identification, user group identification
	Output	success or failure indication
createGroup	Description	Creates a new OSN user group and assigns initial OSN users to this group.
	Input	user group identification, user identifications
	Output	success or failure indication
deleteGroup	Description	Deletes an OSN user group
	Input	user group identification
	Output	success or failure indication
addUser ToGroup	Description	Adds an OSN user to a group of users
	Input	user identification, user group identification
	Output	success or failure indication
removeUser FromGroup	Description	Removes an OSN user from a group of users
	Input	user identification, user group identification
	Output	success or failure indication
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		

Table 18: Description of the User Management Service

9.4.14 Authorisation Service

Name	Authorisation Service	
Standard Specifications	OASIS Security Services (SAML) TC	
Extension of	RM-OA Service	
Description	The Authorisation Service determines what level of access an authenticated user (see Authentication Service, section 9.4.15) should have to resources, i.e. it stores information about user access levels. It provides answers to questions such as: - Is user X authorised to access to the feature instance F? - Is user X authorised to execute service S? - Is user X authorised to perform operation O on instances of feature type FF? - Is user X authorised to perform operation O provided by service S? Authorisation can be granted to single users and to groups of users (see the User Management Service as described in 9.4.13). As features may contain other features (feature collections are considered to be features, too), it is possible to grant access rights on a broader scope. The access rights granted to feature collections are recursively granted to all members of a feature collection. This implies that the user invoking the *setAccessRights* operation for a collection has the right to invoke this operation for each feature of the collection. The same applies for composed services. Setting of an execution right granted for a composed service requires the right of setting the execution right for each of the elementary services building the composition.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of an Authorisation Service instance.
	Input	
	Output	levels of access which can be handled by the service.
setAccessRights	Description	Sets the access rights for users to resources (features or services).
	Input	list of resource identifiers, list of user identifications and/or user groups, access rights to be set
	Output	success or failure indication
getAccessRights	Description	Gets the access rights for a user to resources (features or services).
	Input	resource identifier, user or group identification
	Output	access rights granted to the user/group
Example usage	Rights like read, write, access, execute services, compose services or feature collections, modify rights etc. are granted for users/user groups 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	Links to Digital Rights Management will be investigated in the future.	

Table 19: Description of the Authorisation Service

9.4.15 Authentication Service

Name	Authentication Service	
Standard Specifications	OASIS Security Services (SAML) TC IETF RFC2903 (AAA) Kerberos Network Authentication Protocol (http://web.mit.edu/kerberos/)	
Extension of	RM-OA Service	
Description	The Authentication Service provides secure identification of a user of an OSN. It decides whether the user really is who he/she pretends to be. The decision is based on some unique information known (or available) only to the user being authenticated and the instance of the Authentication Service -- a shared secret ("identity credential"). Shared Secrets are pre-shared keys that have been allocated to the communicating parties prior to the communication process starting. By default, this information is just a simple password, however, authentication by more sophisticated mechanisms (e.g. biometric identification, derived data from smart cards, public/private keys) is not excluded. In order to verify the identity of a user, the Authentication Service challenges the user to provide his shared secret. If it can verify that the shared secret has been presented correctly, the user is considered authenticated. The Authentication Service returns a session key, which is supplied by the user when he wants to access a resource (a feature or service). Each session key has a validity label that indicates the validity of the session key over time, e.g. a session key may be invalidated within a given time period of user inactivity.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of an Authentication Service instance.
	Input	
	Output	type of credentials accepted for authentication, validity label types for session keys
logon	Description	Asks for authentication within an OSN and starts a session.
	Input	user identification, identity credential, requested validity label
	Output	session key by which the user can identify himself when applying for access to resources within the session opened by this operation request, assigned validity label for the session key
logoff	Description	Terminates the session between a user and an OSN.
	Input	Session key
	Output	Success or failure indication.
verifySession Key	Description	Verifies if the session key is still valid.
	Input	session key
	Output	user identification

Example usage	A human user has been registered by means of the user management service within an OSN. The administrator has entered a user name and an initial password for the new OSN user and sends the password to the user by means of an email. When contacting an OSN via an ORCHESTRA Application, at first an authentication request mask is shown to the user. He enters his user name and password and the Authentication Service authenticates him by sending back a session key. The session key is automatically submitted by the ORCHESTRA Application with every new request, it expires within half an hour and a new authentication is necessary in order to continue browsing the OSN.
Comments	It is up to the designer and provider of an OSN if user authentication is necessary and if yes, by using which authentication mechanism.

Table 20: Description of the Authentication Service

9.4.16 Service Monitoring Service

Name	Service Monitoring Service	
Standard Specifications	OASIS Framework for Web Services Implementation (FWSI)	
Extension of	RM-OA Service	
Description	The Service Monitoring Service provides an overview about ORCHESTRA Service Instances (OSIs) currently registered within an OSN. For each of these OSIs the following information is provided: 1. Actual status of the OSI (e.g. running, stopped, offline) 2. Statistical information (e.g. average availability, response times) Statistical information can be aggregated to get an overview about the OSN as a whole. Note that all OSIs must have to be registered in one of the catalogues within an OSN before they can be monitored.	
Service Operations		
getCapabilities (from RM-OA Service)	Description	Informs the requestor about the capabilities of a Service Monitoring Service instance.
	Input	
	Output	monitored service types, types of statistical information available
getActualStatus	Description	Gets actual status of all OSIs currently registered in the catalogues of an OSN.
	Input	catalogue to use constrains for monitored services (optional, default: all services in the catalogue)
	Output	list of service and corresponding status
getStatistics	Description	Get statistics of all OSIs for the last monitored period (from the time of the last *startMonitoring* request up to now or up to the corresponding *stopMonitoring* request)
	Input	constraints for monitored services (optional, default: all services monitored)

	Output	list of services and corresponding statistics
startMonitoring	Description	Start of a Monitoring period with a defined list of services
	Input	catalogue to use, constraints for a list of services to be monitored (optional, default: all services in the catalogue)
	Output	identifier of a monitoring process
stopMonitoring	Description	Stops a monitoring period
	Input	identifier of a monitoring process
	Output	
Example usage	A service provider wants to have statistical numbers for a certain collection of OSIs about the number of accesses in order to judge the economical value of the OSIs.	
Comments	The monitoring of services heavily depends of the amount of information these services offer to the Service Monitoring Service. At least, the information provided by the *getCapabilities* operation inherited from the top RM-OA Service may be used. It will be decided later if a dedicated monitoring interface as part of the top RM-OA Service interface will be defined in addition (e.g. providing additional statistical usage information).	

Table 21: Description of the Service Monitoring Service

9.5 OA Support Service Descriptions

9.5.1 Coordinate Operation Service

Name	Coordinate Operation Service	
Standard Specifications	OGC Abstract Specification Topic 2 - Spatial referencing by coordinates ISO 19111 - Spatial referencing by coordinates OpenGIS Coordinate Transformation Service Implementation Specification GeoAPI (http://docs.codehaus.org/display/GEO/Home)	
Extension of	RM-OA Service	
Description	The Coordinate Operation Service changes coordinates in a geometry based in OGC simple feature 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 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 to depict 3-dimensional data (the shape of the earth) in 2 dimensions (a piece of paper/a screen). Exampled are UTM and Lat/Lon. 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.	
Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of a Coordinate Operation Service instance.
	Input	None
	Output	List of all conversions / transformations that is supported by the service
convert-Coordinates	Description	Convert coordinates that does not include any change of Datum
	Input	coordinates, projection, datum, region (e.g UTM zone), (height system for 3d, e.g., gravity related height or ellipsoidal height), accuracy
	Output	coordinates, accuracy
transform-Coordinates	Description	Transform coordinates that usually includes a change of Datum
	Input	coordinates, projection, datum, region (e.g UTM zone), (height system for 3d, e.g., gravity related height or ellipsoidal height)
	Output	coordinates, accuracy

Example usage	Coordinate conversion: A user wants to transform 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	This service combines the coordinate conversion and transformation services. The combination is useful and natural because conceptually it is the same thing: you give coordinates as parameters and are returned converted/transformed coordinates. As the separation into two distinct operations is the approach that is currently followed in the standards (e.g. ISO 19111, see below) and corresponding application programming interfaces like GeoAPI 2.0, it has been kept for this service. ISO 19111: A coordinate transformation differs from a coordinate conversion in that the coordinate transformation parameter values are derived empirically: therefore there may be several different estimations (or realizations). Once the parameter values are obtained, both coordinate conversion and coordinate transformation use similar mathematical processes.

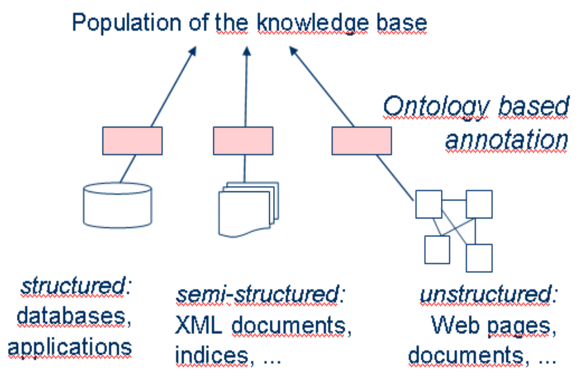
Table 22: Description of the Coordinate Operation Service

9.5.2 Gazetteer Service

Name	Gazetteer Service	
Standard Specifications	ISO 19112 Geographic information — Spatial referencing by geographic identifiers (abstract) Gazetteer Service Profile of the OGC WFS Implementation Specification	
Extension of	Feature Access Service	
Description	The 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). A client can provide a geographic name and is returned a location or a client can provide a location and is returned a geographic name. The Gazetteer Service is a feature access service and can also be transactional, allowing the extension and updates of the underlying gazetteer, e.g., by making request of geographic name updates or geometry updates. If this is the case, the gazetteer service also needs to include authorization.	
Service Operations	None	The Gazetteer Service inherits all operations from the Feature Access Service (see section 9.4.2) and no additional operations are required at the conceptual level.
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	From an abstract viewpoint the Gazetteer Service might be seen as (a specialisation of) a feature service. Thus the need of a gazetteer service – at least on the abstract level – might be discussed. This specialisation could then also be used to allow for geo-coding of non-geo-coded features (having only an implicit geo-reference, e.g. ZIP-code, City-Name etc.). Thus one would e.g. link two instances of a feature access service (the source feature service and the specialised gazetteer service) to provide these geo-coding functionalities.	

Table 23: Description of the Gazetteer Service

9.5.3 Annotation Service

Name	Annotation Service	
Standard Specifications	W3C-Ontology Web Language (OWL) http://www.w3.org/TR/owl-features/ W3C-Ontology Web Language – Query Language (OWL-QL) http://ksl.stanford.edu/projects/owl-ql/ Annotea Project http://www.w3.org/2001/Annotea	
Extension of	RM-OA Service	
Description	As outlined in section 7.2.5, ORCHESTRA provides several services for generation of meta-information, the annotation service is one of these. The Annotation Service generates semantic meta-information on resources (information elements or services) of source systems by means of automatic or semi-automatic annotation. Based on the concepts defined in an ontology (e.g. an OWL ontology), the service identifies instances of these concepts and their relationships between each other and describes these instances in the generated meta-information (e.g. RDF triples). Annotated resources can be queried by means of a query language defined for the input ontology, e.g. OWL-QL. The instances and their relationships usually are not defined in the resources themselves. The resource is assumed to be established without any relationship to the ontology. The annotation service discovers instances by using methods from various fields, e.g. Statistical Natural Language Processing (SLNP), data mining, image processing, Knowledge Discovery in Databases (KDD). It generates meta-information which is not explicitly given in the resource, but can be extracted by means of resource analysis. The extracted meta-information is used to populate an ontology based knowledge base.  Population of the knowledge base Ontology based annotation structured: databases, applications semi-structured: XML documents, indices, ... unstructured: Web pages, documents, ...	
Service Operations		
getCapabilities	Description	Provides information about the capabilities of the annotation service, e.g. the type of resource for which the annotation service is capable of extracting meta-information.
	Input	
	Output	Description of the capabilities of the annotation service
annotateResource	Description	Generate semantic meta-information describing the resources.

	Input	List of resource identifiers and related resource type (as outlined in section 8.4). A formally specified ontology.
	Output	List of terms identified in the document which can be described as instances of a concept defined in the input ontology, and relationships between identified instances (triples). The instances found by the Annotation Service are added to the ontology and can be queried by means of the ontology query language.
Example usage	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. Web services are usually annotated by means of OWL-S (OWL for services, see http://www.daml.org/services/owl-s) or WSMO (Web Service Modelling Language, see http://www.wsmo.org/).	
Comments		

Table 24: Description of the Annotation Service

9.5.4 Document Indexing Service

Name	Document Indexing Service
Standard Specifications	no corresponding standard known
Extension of	RM-OA Service
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 8.5.2.1) "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. he cannot formulate an exact query for what he is searching for, but just give a vague description by means of some search terms, - Then, the user refines his search based on results got in previous searches. - Finally, the user retrieves the complete document where the search term is contained (not only 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 just a 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 a 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.

Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of an Document Indexing Service instance.
	Input	None
	Output	list of supported MIME-types, for which this service can automatically extract meta-information
startGenerate Index	Description	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 is 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
	Input	list of OAS_DocumentDescriptors for each document: a reference list containing those terms identified as instances of concepts of an ontology (optional). Note: These references may have been generated by the Annotation Service before. update cycle (optional)
	Output	generation identifier, search index
stopGenerate Index	Description	Stops the generation of the document search index.
	Input	generation identifier
	Output	success or failure indication
Example Usage	A user needs an efficient search mechanism for all documents that may be accessed within the entire OSN. Therefore, he needs 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.	
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 9.5.3). 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 25: Description of the Document Indexing Service

9.5.5 Format Conversion Service

Name	Format Conversion Service
Standard Specifications	MIME Media Types (http://www.iana.org/assignments/media-types/) XSL Transformations (XSLT) (http://www.w3.org/TR/xslt)
Extension of	RM-OA Service

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 adapters or a service chain. Data categories of different granularity are supported by this service (e.g. a string, a text file, an image file). The data categories supported are listed as return parameters of the *getCapabilities* operation.	
Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of a Format Conversion Service instance.
	Input	source format (optional), target format (optional)
	Output	List of supported data categories, list of the supported pairs of source and target formats If the parameters are given the output will state if there is a conversion for the specified pair of source- and target-format available or possible.
convert	Description	Performs the conversion (if a conversion that is capable to transform from data with format f_1 to the corresponding data in format f_2 is available).
	Input	Data, source format , target format
	Output	Data with target format
Example usage	An implementation of the Document Access Service (see section 9.4.5) may use the Format Conversion Service in order to deliver a document in the specified format. Example: If a client component wants to view an image with a viewer that doesn't support the vector format it is needed to convert the image file into a non-vector format like jpg.	
Comments		

Table 26: Description of the Format Conversion Service

9.5.6 Schema Mapping Service

Name	Schema Mapping Service
Standard Specifications	Draft Technical Specification 19103, Geographic information – Conceptual schema language The current mappings are expressed following one of the corresponding languages such as • SQL, SQL:1999 • SQL/XML, SQL:2003 • XQuery, http://www.w3.org/TR/xquery/ • XSLT, http://www.w3.org/TR/xslt
Extension of	RM-OA Service
Description	The Schema Mapping Service transforms data from one schema to another one assuming that this mapping has been formally described before. A schema has a unique name and is formally described using a conceptual schema languages (CSL) (e.g. entity-relationship-diagram, UML, RDF, OWL, XML Schema, SQL Schema, …). A schema is

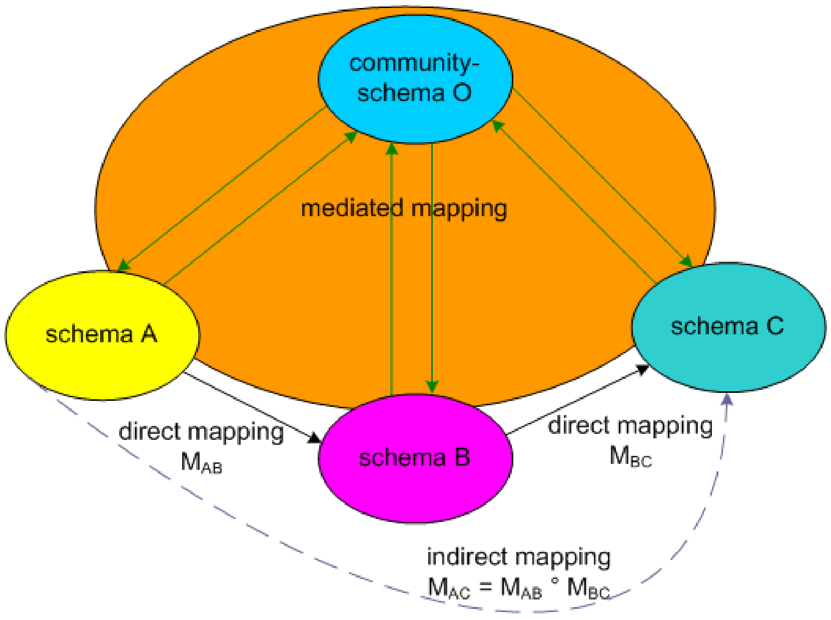
identified by a OAS_SchemaDescriptor as defined in section 8.4.4.3.

A schema mapping rule is defined by

- a unique name,
- the CSL used to describe the source and the target schemas,
- the CSL used to describe the mapping,
- the specification of the mapping.

There are two kinds of mappings: a) mediated (upper part of the figure) and b) not mediated (lower part of the figure). “Not mediated” means, that there is a direct mapping between schemas, “mediated” means that there is an indirection, so that each participating schema is mapped to a community-schema (common schema). Thus these “schemas” form a community.

After connecting a schema to a community-schema, mediated mapping immediately allows to map it to each other schema being in the same community. Direct mapping has the advantage, that it can be adapted to the specific local requirements.



Some of the mappings can be described (e.g. a XSLT can describe the mapping between two XML schemas), in other words the mapping can rather be configured, but usually adapters will be required, so programming will be inevitable.

Note: Not every pair of schemas can be mapped in a sensible way. Furthermore, information loss may occur if a mapping of a particular schema concept in schema A is not possible.

Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of a Schema Mapping Service instance.
	Input	
	Output	list of supported schema mapping rules and also the supported mapping rule language (e.g. SQL, XQuery, XSLT or program code).
transformData	Description	Performs the transformation of data from one schema to another one according to predefined schema mapping rules.
	Input	input data source (e.g. jdbc database url), reference to a schema

		mapping rule
	Output	data that satisfies the given output schema
setMapping Rule	Description	Set a new mapping rule instead of an old mapping rule.
	Input	The new mapping rule to replace and the to be replaced mapping rule
	Output	
getMapping Rule	Description	Returns a mapping rule from given source schema to given target schema.
	Input	references to OAS_SchemaDescriptor for the source schema and target schema
	Output	mapping rule from source schema to target schema
getIndirect MappingRule	Description	Returns a mapping rule from a source mapping rule to a target mapping rule.
	Input	references of source mapping rule and target mapping rule
	Output	Mapping rule from source mapping rule to target mapping rule
generate Community Schema	Description	Generate from all the given or existing schemas and mapping rules into a integrated community schema as a suggestion for community schema.
	Input	Optional mapping rules and references of local schemas
	Output	Community schema
Example usage	Within the integration of a new data source it is necessary to define a schema conversion between the database schema of the new data source and an OAS.	
Comments	Within a transfer of data from a database schema A to an ORCHESTRA Application Schema (OAS) B there is a need for programmatic statements. Consider A has RDBMS table attributes like name, first name, street, town. The OAS has the elements name, first name and address. In this case a mapping between the different representations of an address is not only a simple 1:1 mapping between A's attributes and B's elements. More than that there are additional transformations needed. In order to get the B's address a concatenation of A's attributes street and town has to be performed. In general the mapping describing the conversion has to allow programmatic statements (e.g. Java). This is necessary because of the transformations (string-, arithmetical operations, ...) required within some conversions. A predefined set of conversions that can be used declaratively will not be sufficient.	

Table 27: Description of the Schema Mapping Service

9.5.7 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. This standard extends a monolingual thesaurus as defined in ISO-2788, so that multiple languages are also covered. W3C Quick Guide to Publishing a Thesaurus on the Semantic Web

	(http://www.w3.org/TR/2005/WD-swbp-thesaurus-pubguide-20050517)	
Extension of	RM-OA Service	
Description	The Thesaurus Access Service supports the 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. the question if the meaning of two terms is similar, broader or narrower or not. 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 for shall be provided.	
Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of a Thesaurus Access Service instance.
	Input	language (optional)
	Output	list of the supported languages or states if the optional language is supported (conditional).
getScope	Description	Get 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).
	Input	term – the term for which the scope is to be returned.
	Output	human readable description of the scope corresponding to the given term.
getPreferred Term	Description	Get the preferred term when a choice between synonyms or quasi-synonyms exists.
	Input	term – the term for which the preferred term is asked
	Output	preferred term
getSynonyms	Description	Get the synonyms of a given term in a given language.
	Input	a term, desired language for synonyms
	Output	list of synonyms for the given term.
getAntonyms	Description	Get the antonyms of a given term in a given language.
	Input	a term, desired language for antonyms
	Output	list of antonyms for the given term.
getTopTerm	Description	Get 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)
	Input	a term

	Output	broadest class the given term belongs to.
getBroaderTerm	Description	Get a concept having a wider meaning than the given term has.
	Input	A term
	Output	concept with a wider meaning than the given term has.
getNarrower Terms	Description	Get a concept with a more specific meaning than the given term has.
	Input	a term
	Output	concept with a more specific (narrower) meaning than the given term has.
getRelatedTerm	Description	Get an associated term, but that term is not a synonym, a quasi-synonym, a broader term or a narrower term.
	Input	a term
	Output	term that is associated with the given term, but that is neither a synonym, nor a quasi-synonym or a broader/narrower term.
setScope	Description	Set a note attached to a term to indicate its meaning within an indexing language
	Input	term – the term for which the scope is to be set, description – note to indicate the meaning of the term
	Output	None
setPreferred Term	Description	Set the preferred term for another term
	Input	preferredTerm – the preferredTerm for a term term – this term gets an association to a preferredTerm
	Output	None
setSynonyms	Description	Set a synonym for a term in a given language.
	Input	term, synonym, language
	Output	None
setAntonyms	Description	set an antonym for a given term in a given language.
	Input	term, antonym, language
	Output	None
setTopTerm	Description	Set the broadest class to which a term belongs
	Input	term, topTerm
	Output	None
setBroaderTerm	Description	Set a broaderTerm for a term.
	Input	term, broaderTerm
	Output	None
setNarrower Terms	Description	Set a narrowerTerm for a term.
	Input	term, narrowerTerm
	Output	None
setRelatedTerm	Description	Set 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.
	Input	term, relatedTerm
	Output	None
Example usage	An end-user can use the Thesaurus Access Service to determine synonym terms, which can consequently be used to broaden a search.	
Comments	SN - Scope note; a note attached to a term to indicate its meaning within an indexing language USE - The term that follows the symbol is the preferred term when a choice between synonyms or quasi-synonyms exists UF - Use for; the term that follows the symbol is a non-preferred synonym or quasi-synonym TT - Top term; the term that follows the symbol is the name of the broadest class to which the specific concept belongs; sometimes used in the alphabetical section of a thesaurus BT - Broader term; the term that follows the symbol represents a concept having a wider meaning NT - Narrower term; the term that follows the symbol refers to a concept with a more specific meaning RT - Related term; the term that follows the symbol is associated, but is not a synonym, a quasi-synonym, a broader term or a narrower term	

Table 28: Description of the Thesaurus Access Service

9.5.8 Inferencing Service

Name	Inference Service
Standard Specifications	W3C – Resource Description Framework (RDF, W3C - Resource Description Framework (RDF) Schema Specification 1.0 Web Ontology Language (OWL) XQuery (http://www.w3.org/TR/xquery/)
Extension of	RM-OA Service
Description	The Inference Service supports the derivation of implicit information from ontology knowledge models and the check of the logical consistency of ontologies. This service will accept ontology knowledge models and queries as input, and provide responses to those queries as outputs. Typically a knowledge model will consist of two parts: - a TBox: the ontology – a description of the terminology, and - an ABox: assertions about individuals (typically stored in a relational database or RDF triple stores). An inference service can be used to combine different TBoxes, and TBoxes and ABoxes. This allows different knowledge models to be aligned or merged and also allows complex querying to be made onto the underlying knowledge model. Query languages for ontologies have not been standardised – some services use XQuery (W3C standard XML query language), others use yet to be standardised languages such as RDQL, OWLQL etc. Thus this service indicates to the requestor which query language it supports.

	Note that "Class Expression" in the below can either refer to the URI of a named class or a complex class expression built from named classes. whereas the class names may be taken from ontologies retrieved by the Ontology Access Service (see section 9.4.12). For example one could ask for subclasses of a name classes like "River" or one could ask for subclasses for complex class constructs such as "River AND contains (someValues-From) PolutedWater".	
Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of an Inferencing Service instance.
	Input	
	Output	query language supported as input, type if inferencing operations supported.
isConsistent	Description	Run on the whole ontology to see whether it is logically consistent
	Input	name or URI of ontology
	Output	true or false
isSubClassOf	Description	Determines whether a class "C1" is a subclass of another class "C2"
	Input	Two class expression
	Output	true or false
subClasses	Description	Return subclasses of a class. It will return "direct" subclasses – those one level down in the hierarchies and directly under the target class, "all" returns all subclasses of a class, "proper" contains all subclasses of a class that are not equivalent classes to the target class.
	Input	class expression, plus qualifier "direct", "all" or "proper"
	Output	list of subclasses
subProperties	Description	Return sub-properties of a property. It will return "direct" sub-properties – those one level down in the hierarchies and directly under the target class, "all" returns all sub-properties of a property, "proper" contains all sub-properties of a property that are not equivalent properties to the target property.
	Input	named property
	Output	list of sub-properties
isDisjointClass	Description	Determines whether a class "C1" is disjoint from class "C2"
	Input	class expression
	Output	true or false
equivalent-Classes	Description	Return equivalent classes to a given class expression
	Input	class expression
	Output	list of equivalent classes
equivalentProp-erties	Description	Return equivalent properties to a given property expression
	Input	named property
	Output	list of equivalent properties
isSuperClassOf	Description	Determines whether a class "C1" is a superclass of another class "C2"

	Input	two class expression
	Output	true or false
superClasses	Description	Return superclasses of a class. It will return “direct” superclasses – those one level down in the hierarchies and directly under the target class, “all” returns all superclasses of a class, “proper” contains all superclasses of a class that are not equivalent classes to the target class.
	Input	class expression, plus qualifier “direct”, “all” or “proper”
	Output	list of superclasses
superProperties	Description	Return superproperties of a property. It will return “direct” superproperties – those one level down in the hierarchies and directly under the target property, “all” returns all superproperties of a property, “proper” contains all superproperties of a property that are not equivalent properties to the target property.
	Input	named property, plus qualifier “direct”, “all” or “proper”
	Output	list of superproperties in XML
satisfiableClass	Description	Determines whether a class is satisfiable. An unsatisfiable class is one that can contain no members. For example the intersection of male and female is unsatisfiable – there is no individual that is both male and female
	Input	class Expression
	Output	true or false
containingClass	Description	Returns most specific classes that contain a given individual
	Input	URI for OWL instance
	Output	list of containing classes and the classes equivalent as an XML document
isDifferentIndi-vidual	Description	Determines where two individual names represent different instances
	Input	URI for two OWL instances
	Output	true or false
isSameIndi-vidual	Description	Determines where two individual names represent the same instance
	Input	URI for two OWL instances
	Output	true or false
isTransitive *isSymmetric* *isFunctional* *isInverse Function*	Description	Determines whether a given name property is transitive, symmetric, functional or inverse functional
	Input	name property
	Output	true or false
isInverseOf	Description	Determines whether two name properties are the inverse of each other
	Input	two name properties
	Output	true or false
describeClass	Description	Lists all explicit axioms for a named class (not class expression)

	Input	named class
	Output	list of axioms used to describe target class. (similar output to that of "classProperty" below)
intersecting-Classes	Description	Lisst all classes that are not disjoint from a given class expression
	Input	class expression
	Output	list of intersecting classes
subClassPath *superClassPath*	Description	Returns sub/super class tree below/above a given class expression
	Input	class expression
	Output	hierachy tree
ontologyProper-ties	Description	Returns meta-information for ontology such as names of all classes, properties and individuals along with metrics given total numbers of class, properties and individuals.
	Input	Ontology name/URI
	Output	Information mentioned n "description"
entails	Description	"Ontology 1 entails Ontology 2" means that each axiom in Ontology 2 is implies by Ontology 1
	Input	URIs of two ontology
	Output	true or false
classProperty	Description	Returns list of name classes that are related to a class expression via a named property. One can either specifiy a named property or fine results for all properties.
	Input	class expression and optionally name property
	Output	list of properties relating the target class expression to a class filler – answer provided, e.g. <ClassProperty> <class> River </class> <property><name> flowsInto </name></property> <filler><name> Sea </name></filler> </ClassProperty>
Example usage	Query clients can be written to load ontologies into the Inferencing Service. These ontologies (and any underlying data structures) can then be queried. - Find information about "weather events that cause flooding that are not either storms or heavy rain fall" - Details of flood events in Cornwall between 1990 and 2005 - Records of water level measures of rivers in the UK - Find water levels in a certain geographic area	
Comments	There are various off the shelf solutions available at various levels of maturity Some of theme include further reasoning operations that will be considered in future, see e.g. RACER (**R**enamed **A**Box and **C**oncept **E**xpression **R**easoner, http://www.racer-systems.com/products/racerpro/index.phtml).	

Table 29: Description of the Inferencing Service

9.5.9 Service Chain Access Service

Name	Service Chain Access Service	
Standard Specifications	BPEL Business Process Execution Language BPML (Business Process Modelling Language from bpmi.org (Business Process Management Initiative) as a strict superset of BPEL4WS Business Process Execution Language for WebServices WSCI Web Service Choreography Interface (W3C initiative) WSDL (Web Services Description Language) WSMO (Web Service Modelling Ontology, http://www.wsmo.org) WSML (Web Service Modelling Ontology Language, http://www.wsmo.org)	
Extension of	RM-OA Service	
Description	The Service Chain Access Service gets a specification of a service choreography and creates thereof an aggregate service (with an underlying service chain). The installer (requestor of the *createServiceChain* operation) for the aggregate service can register the service in a catalogue to enable other OSN users to use this service. The installer might be a person or an OSI. The resulting aggregate service is itself an ORCHESTRA Service (i.e. inherits from the RM-OA Service) with all generic properties (like e.g. a *getCapabilities* operation). It is identified by an OAS_ServiceDescriptor. The execution of the service chain thus corresponds to the call of an ORCHESTRA Service Instance in an OSN. This approach reflects the opaque chaining pattern of the ISO 19119 standard in a catalogue.	
Service Operations		
getCapabilities	Description	Informs the requestor about the capabilities of a Service Chain Access Service instance.
	Input	
	Output	list of choreography languages
createServiceChain	Description	Creates an aggregate service as an ORCHESTRA Service composed of several ORCHESTRA Services defined within the choreography.
	Input	name and rights of the aggregate service. list of services identified by OAS_ServiceDescriptors to be aggregated choreography of service interoperation of the ORCHESTRA Services
	Output	identifier of the installed aggregate service (which might then be registered in a catalogue)
deleteServiceChain	Description	Deletes the aggregate service
	Input	name or identifier of the aggregate service
	Output	
Example usage	A flood forecast service might be orchestrated in the way that several access services might deliver input into a simulation model service.	

Comments	The aggregate service may provide certain operations for its own management and administration. The list of these operations depends on the implementation of the Service Chain Access Service and can be queried using the *getCapabilities* operation of the aggregate service.

Table 30: Description of the Service Chain Access Service

9.6 OT Support Services

Note: The description of the OT Support Services do not (yet) comprise detailed 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 9.2.1).

9.6.1 Statistical Calculation Service

Name	Statistical Calculation Service
Standard Specifications	MathML 2.0 specification (http://www.w3.org/Math/) important references are existing statistical analysis libraries such as R (http://www.r-project.org/) or Matlab (http://www.mathworks.com).
Extension of	RM-OA Service
Description	The Statistical Calculation Service provides a set of functions that allow the requestor to perform statistically processes that are commonly employed in a risk management context but also in other fields. Example operations are: - *sort*: sort elements in x into descending or ascending order (depending on value of direction) and return the new indices of the sorted elements in index - *bin*: count the number of elements in x within different intervals (limits of intervals defined by y) and return the count and also the indices of the elements in each interval - *regression*: apply weighted least-squares regression analysis to a set of data, where x_1, …, x_n are independent variables, y is the dependent variable, function is the functional form and weight are the weights to apply to the data. It returns the coefficients, $a_1,\dots,a_m$, of the function. This can handle computing the weighted mean of some data by defining function as simply a constant - *mode*: compute the mode of a set of data - *ftest*: apply the F-test to a set of data - *pdf*: compute values of the probability density function (pdf) at given points, x, for a given distribution (normal, uniform, exponential, beta etc.) with some parameters - *probability*: compute the probability of an occurrence based on a given probability density function - *random*: generate random numbers, x, between limits x_{min} and x_{max} using values of the probability density operation (pdf) - *sum*: return the sum of a list of numbers. - *intersect*: calculate the point where two curves (defined by interpolation between points or by functions) intersect - *extrapolate*: extrapolate a given set of data, y_1, …, y_n, specified at points x_1, …, x_n, to estimate the value of a variable, y_{pred}, at a point, x_{pred}, not within the input range of data. Different extrapolation methods, method, can be employed (e.g. nearest, linear, spline) The Statistical Calculation Service indicates the following capabilities to the requestor: list of operations supported, including the parameters and their expected format

Example usage	Some examples for the above mentioned operations: *sort*: if x=4, 3, 1 then the output would be x_{sort}=1, 3, 4 and index=3, 2, 1 *bin*: if x=1.1, 3.2, 1.5, 3.45, 3.46 and y=1, 2, 3, 4 then output would be count=2, 0, 3 and index=1, 3 ; ; 2, 4, 5 *sum*: if x=1, 3, 2, 7.2, 10.9, -2 then sum=22.1
Comments	The list of operations requires a check against the requirements and priorities of the risk application(s) as soon as they are well analysed. A method for defining functional forms of any equations is required when specifying this service. The mathematical algorithms used by the service operations could be taken from public domain libraries, e.g. the RANDLIB (http://odin.mdacc.tmc.edu/anonftp/) Fortran library.

Table 31: Description of the Statistical Calculation Service

9.6.2 Computer Algebra service

Name	Computer Algebra Service
Relationship to Standard Specifications	MathML, OpenMath important references are existing mathematical analysis libraries such as R (http://www.r-project.org/), Matlab (http://www.mathworks.com) or other computer algebra tools (MuPAD, Maple)
Extension of	RM-OA Service
Description	The Computer Algebra Service is a front-end to mathematical analysis libraries and computer algebra tools that provide mathematical functions. These tools can among others be used in the context of thematic domains like environmental management or risk management. This service relies on a standardised format to represent mathematical formulas (e.g. MathML, see the Formula Access Service in section 9.4.7) and is able to transform this representation into the proprietary i/o-formats of the integrated libraries and tools. The Computer Algebra Service indicates the following capabilities to the requestor: list of operations supported, including the parameters and their expected format
Example usage	<format of formula access service> (e.g. MathML) evaluateFormula (<format of formula access service> (e.g. MathML), <String targetLibrary> (e.g. Maple))
Comments	

Table 32: Description of the Computer Algebra Service

9.6.3 Geospatial Calculation Service

Name	Geospatial Calculation Service
Standard Specifications	OGC Web Processing Service (discussion paper): provides access to calculations or models which operate on spatially referenced data. statistical analysis libraries such as R (http://www.r-project.org/) or Matlab (http://www.mathworks.com) or the geostatistical analyst as an extension of ESRI's ArcView/ArcGIS (http://www.esri.com)
Extension of	RM-OA Service

Description	A geospatial calculation service is a processing service that supports a number of spatial operations based on vectorial as well gridded structured spatial input and output data. Example operations will handle buffering, overlay, calculation of areas/perimeters or temporal extensions for certain attributes. More sophisticated operations comprise functions that can be used to process geospatial data to estimate values of the variable at geographical locations not in the original set of data points. Examples here are kriging operations that return the predicted value of *y*, y_{pred}, based on data specified at given points (not necessarily regularly distributed) in n-dimension space : $x_1, \ldots, x_n$ using kriging of a given *type* e.g. Simple Kriging, Ordinary Kriging, Indicator Kriging.
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. Groundwater pollution data (e.g. for nitrate) is available only at monitoring locations, ie. measurements are only available as a point object. By using a kriging algorithm an interpolation for points in a given surface is possible (e.g. the estimated concentration of nitrate in a given region).
Comments	The refinement of this service shall be carried out according to the priorities set by the ORCHESTRA users.

Table 33: Description of the Geospatial Calculation Service

9.6.4 Image Processing Services

Name	Image Processing Service
Standard Specifications	no corresponding standard known
Extension of	RM-OA Service
Description	The Image Processing Service provides a wrapper to the most important image processing operations. For simplicity, the service first restricts to two-dimensional (2D) image processing whereby most of the concepts and techniques can be extended easily to three or more dimensions. This service considers an image as a function of two real variables, for example, a(x,y) with a as the amplitude (e.g. brightness) of the image at the real coordinate position (x,y) (Young et al). An image may be considered to contain sub-images sometimes referred to as regions-of-interest, ROIs, or simply regions. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve colour rendition. The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantization process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantized. In other image forming procedures, such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real phase. Examples for Image Processing service operations range in the following categories: - operations based on the image histogram (e.g. contrast stretching),

	- on simple mathematics (binary arithmetic and ordinary arithmetic), - on convolution (in the spatial and the frequency domain) and - on mathematical morphology. The Image Processing Service indicates the following capabilities to the requestor: available Image Processing service operations, image formats supported
Example usage	Risk assessment processes often requires the combination of information contained in images, e.g. from satellites, high-altitude platforms or photos. The Image Processing Service helps in automatically adapting the images for a combined processing in a common software environment (e.g. a visualisation tool).
Comments	The refinement of this service shall be carried out according to the priorities set by the ORCHESTRA users (e.g. need for the processing of three-dimensional data).

Table 34: Description of the Image Processing Service

9.6.5 Simulation Management Services

Name	Simulation Management Service
Standard Speci-fications	IEEE 1516 High-level Architecture Predictive Model Markup Language (PMML) of the Data Mining Group (http://sourceforge.net/projects/pmml)
Extension of	RM-OA Service
Description	The Simulation Management Servcie provides a wrapper with generic interfaces to existing simulation services. It allows to launch a simulation run of a simulation model. The service requires a description of the required simulation input parameters, allows to hand this parameters to the existing simulation service that will then initiate the simulation run. Depending on the required calculation time and the chosen messaging principles (e.g. push or pull) the Simulation Management Service may either directly provide the simulation results as response, or the Simulation Management Service may use an appropriate specialisation of the feature access service and a notification service to provide (in an asynchronous manner) access to the simulation results.
Example usage	
Comments	Simulation results could also be understood as sensor data (virtual sensors) and then accessed through the Sensor Access Service as described in section 9.4.9. (Wytzisk 2003)

Table 35: Description of the Simulation Management Service

9.6.6 Sensor Planning Service

Name	Sensor Planning Service
Standard Speci-fications	OGC Sensor Planning Service NASA/JPL Sensor Webs Project (http://sensorwebs.jpl.nasa.gov/). Sensor Model Language (sensorML) (http://vast.uah.edu/SensorML/)
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 do 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 36: Description of the Sensor Planning Service

9.6.7 Project Management Support Service

Name	Project Management Support Service
Standard Specifications	ISO10006 / ISO 10007 Project Management PMBOK (Project Management Body of Knowledge of the Project Management Institute) (http://www.pmi.org/) Project Management XML Schema (PMXML) (http://xml.coverpages.org/projectManageSchema.html) dotProject (http://www.dotproject.net/index.php)
Extension of	RM-OA Service
Description	The Project Management Support Service supports the planning and performance of operations (projects) in a cooperative distributed environment. The purpose is to specify a project based on definitions according to the following three dimensions of project management: - the structure of a project into project elements, i.e. the division of a project into subprojects, 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 e.g. 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 and services 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 of the project results - 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 indicates the following capabilities to the requestor: 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 37: Description of the Project Management Support Service

9.6.8 Communication Service

Name	Communication Service
Standard Specifications	Collaboration Standards: - HTTP(S): HyperText Transfer Protocol (over Secure Socket Layer) is the set of rules for transferring files (text, graphic images, sound, video and other multimedia files) on the WWW - XML (Extensible Markup Language): flexible way to create common information formats and share both the format and the data on the WWW and intranets - SOAP (Simple Object Access Protocol): is a way for a program running in one kind of OS to communicate with a program in the same or another kind of an OS by using HTTP and XML as the mechanisms for information exchange - XMPP: The eXtensible Messaging and Presence Protocol (XMPP) is an IETF standard for server-to-server IM interoperability and presence awareness - SIMPLE: an add-on to the Session Initiation Protocol (SIP) that some industry insiders predict will be the basis for a new Instant Messaging and Presence Protocol (IMPP) - The International Telecommunications Union (ITU) T.120 standard: to create compatible products and services for real-time, multipoint data connections and conferencing - H.323 (and H.320): other ITU standard to promote compatibility in videoconference transmissions over IP networks (and over circuit-switched media) - SIP: an Internet Engineering Task Force (IETF) standard protocol for initiating an interactive user session that involves multimedia elements such as video, voice, chat, gaming, and virtual reality - OGC 03-029, OWS Messaging Framework (OMF), where it is defined a mes-

	saging framework to conduct communications between the OGC web services independently of any transport protocol and any messaging encoding
Extension of	RM-OA Service
Description	The objective of the Communication Service is to provide a harmonised access to direct user-to-user communication means based on multi-media technologies and data exchange between users. A 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 or logistics.
Comments	It is to be decided if at least parts 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 38: Description of the Communication Service

9.6.9 Calendar Service

Name	Calendar Service
Standard Specifications	ISO 8601. Data elements and interchange formats - Information interchange - Representation of dates and times. International standard for date and time representations ISO 19108. Geographic information - Temporal schema. Temporal schema defines standard concepts needed to describe the temporal charac-

	teristics of geographic information. It therefore specifies the Gregorian calendar and Universal Coordinated Time as a preferred basis for interchanging temporal information
Extension of	RM-OA Service
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 the calculation with 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 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 to enter a date, then specify a number of days to be added (to check a future date) or subtracted from (to check a past date) and get the new date. Or, it allows to specify a pair of dates in order to calculate the number of days between these.
Comments	

Table 39: Description of the Calendar Service

9.6.10 Reporting Service

Name	Reporting Service
Standard Specifications	OASIS Open Document Format for Office Applications (OpenDocument)
Extension of	RM-OA Service
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 every format for a report is thinkable for practical reasons only standardized 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 40: Description of the Reporting Service

9.7 OA Patterns

The combined usage of the OA Services and the ORCHESTRA Information Models is illustrated by means of OA patterns. Note that these OA patterns are informative and just provide examples. It does neither claim to be the only way of using and combining the OA Services nor does it claim to be complete.

The following OA patterns are currently described:

- Controlled user access to resources
- Integration of source system data into an OSN
- Semantic catalogue component

Note: Further OA patterns will be added in future versions of the RM-OA, e.g.

- Feature rendering in maps

9.7.1 Controlled User Access to Resources

The User Management Service, the Authorisation Service and the Authentication Service are intended to work together in the following way under the following initial assumptions:

- Some unique information known (or available) only to the user being authenticated and the authentication service - a shared secret (“identity credential”) - is assumed to exist.
- An OSN administration creates users or user groups (User Management Service: *createUser*,…).
- Resource access rights for the users/user groups are granted by other users (Authorisation Service: *setAccessRights*).

The user access to resources is then controlled in the following way:

1. The user authenticates himself in order to get a session key using the *logon* operation of the Authentication Service.
2. The user accesses a resource with a respective service (e.g. the Feature Access Service) using his session key.
3. The requested service verifies the authorisation using the *getAccessRights* operation of the Authorisation Service.
    - The Authorisation Service verifies the session key using the *verifySessionKey* operation of the Authentication Service.
    - The Authorisation service checks the existence of the user and his group relation using the *getUserAttributes* operation of the User Management Service.
4. The Authorisation Service returns the rights to the requested service as output parameter of the *getAccessRights* operation.

Note: This usage pattern is a preliminary draft and will be refined in a future version of the RM-OA with respect to efficiency and other more general approaches (e.g. digital rights management).

9.7.2 Integration of Source System Data into an OSN

The integration of source system data as feature instances into an OSN is performed through a combined usage of the Catalogue Service, the Annotation Service, the Source System Access Service, the Schema Mapping Service and the Feature Access Service.

The principal usage pattern may be as follows:

1. The Catalogue Service provides the means (the catalogue component) to store meta-information about a source system. The meta-information is structured according to the respective OAS-MI for the purpose of discovery. In the following, it is assumed that the

OAS_SourceSystemDescriptor is referenced by the OAS-MI classes.

2. A source system instance is either discovered actively by the catalogue component ("discovery", case a. below) or the source system instance itself registers in the catalogue ("publish", case b. below).
   a. The Annotation Service may be used in order to fill the meta-information entry for the discovered catalogue component.
   b. The meta-information is either already available in the form the OAS-MI requires it or dedicated OAS-MI object instances have to be filled by the provider (e.g. a human administrator) of the source system component.
3. Once the meta-information about the source system instance has been entered in the catalogue component, a user of the Catalogue Service may find it as a result of a catalogue search operation.
4. Through the respective entry in the catalogue component, the user gets a reference to an instance of OAS_SourceSystemDescriptor and retrieves the capabilities of the Source System Access Service that provides that specific OAS_SourceSystemDescriptor. These capabilities are examined by the user to get a list of supported schema languages, access languages and bounding box types.
5. The schema of the source system is requested from the Source System Access Service in the schema language requested by the caller.
6. Using this schema, one of the access languages supported by the Source System Access Service, the user can then retrieve the features of the source system using the *getSourceSystemData* operation of the Source System Access Service.

   Note 1: If the schema type is not OAS, the data contained in the source system instance is implicitly transformed to ORCHESTRA feature instances by the Source System Access Service, e.g. using the Schema Mapping Service (see section 9.5.6) if respective mapping rules have been specified.

   Note 2: If no mapping rules have been specified or no mapping is possible, a user application may also directly access to the source system data using the native access methods of the source system as specified in the OAS_SchemaDescriptor.

9.7.3 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 25. 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 one or more OGC Catalogue Services or any other catalogue service. This, however, should be transparent to the user of the Semantic Catalogue component. This variant is illustrated in Figure 25, too.

The OGC catalogues can be heterogeneous with respect to the structure of meta-information. 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.

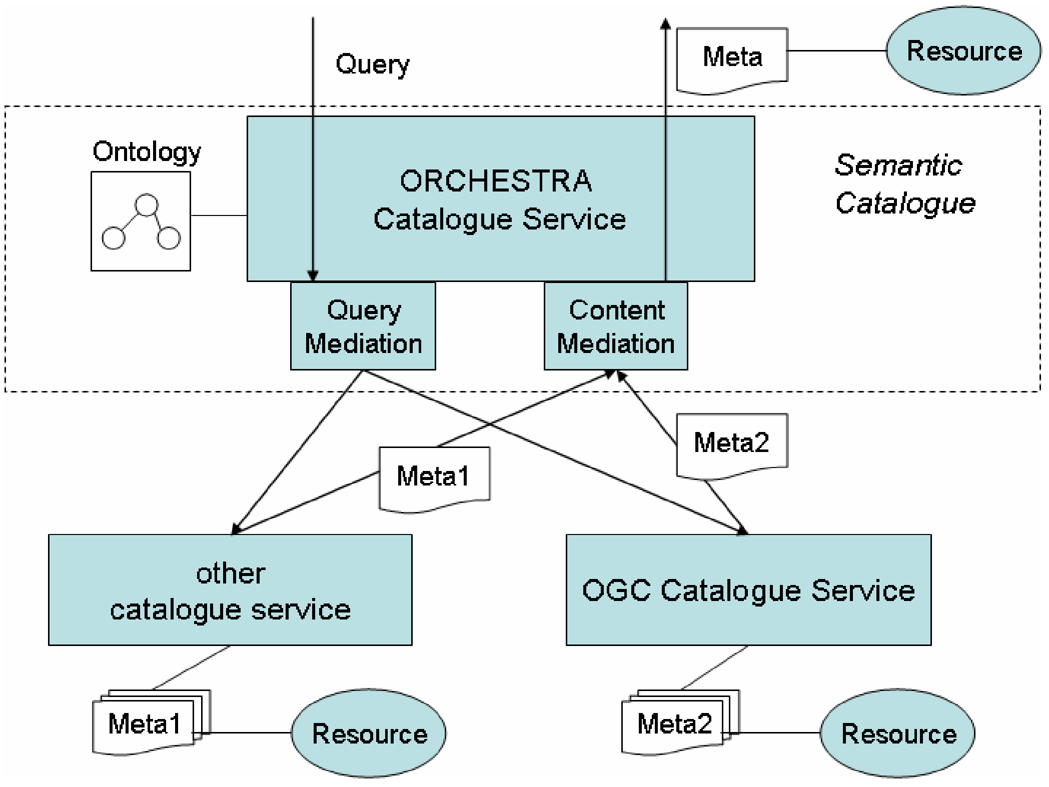


Figure 25: Example of a semantic catalogue

9.8 OSN Conformance Clauses

OA Info-Structure Services 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.

The indispensability with respect to one OSN is defined in terms of conformance clauses in Table 41.

Note: The list of conformance clauses is a preliminary draft and is subject to change during the course of the ORCHESTRA specification and implementation process.

OA Info-structure service	OSN Conformance Clauses
Feature Access Service	ORCHESTRA feature instances shall be accessed through operations of the Feature Access Service
Map Access Service	Maps shall be accessed through operations of the Map Access Service
Diagram Access Service	Diagrams that are described in form of an ORCHESTRA feature instance of type “diagram descriptor”, shall be accessed through operations of the Diagram Access Service. Note: The diagram descriptor still has to be defined.
Document Access Service	Documents that are represented in form of an ORCHESTRA feature instance of type “OAS_DocumentDescriptor”, shall be accessed through operations of the Document Access Service
Source System Access Service	Source Systems that are represented in form of an ORCHESTRA feature instance of type “OAS_SourceSystemDescriptor”, shall be accessed through operations of the Source System Access Service.
Formula Access Service	Documents that are represented in form of an ORCHESTRA feature instance of type “formula descriptor”, shall be accessed through operations of the Formula Access Service. Note: The formula descriptor still has to be defined.
Coverage Access Service	Documents that are represented in form of an ORCHESTRA feature instance of type “CV_Coverage”, shall be accessed through operations of the Coverage Access Service
Sensor Access Service	Sensors that are represented in form of an ORCHESTRA feature instance of type “sensor descriptor”, shall be accessed through operations of the Sensor Access Service. Note: The sensor descriptor still has to be defined.
Catalogue Service	at least one OSI shall be available in an OSN
User Management Service	All users that call operations of ORCHESTRA services within the domain of an OSN shall be managed through operations of the User Management Service. Pre-defined users may exist (e.g. any).
Authorisation Service	All access rights for users in an OSN shall be assigned through operations of the Authorisation service. Pre-defined access rights may only exist for pre-defined users.
Authentication Service	Authentication of users in an OSN shall be performed exclusively through operations of the Authentication Service.
Service Monitoring Service	at least one OSI shall be available in an OSN

Table 41: OSN Conformance clauses for OA Info-structure services

10 Summary of Deviations from Standards

In the following all deviations form standard specification as applied in the RM-OA are summarised. Textual changes are underlined.

10.1 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.

10.2 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

10.3 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

10.4 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 to maintain a description of their mapping to 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 (instead of should be defined).

Rationale: section 5.4.1

10.5 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

10.6 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 just abstractions, concepts or software artefacts (e.g. documents, software components of IT systems) that may have a physical representation just 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.

Rationale: section 8.2

10.7 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.

Rationale: section 8.4

11 Annex A.1: Development Dimensions

The intention of this chapter 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 42 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.

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.

11.1.1 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, semantic interoperability can 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)
- support the users to enhance common understanding 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 no relationship between such concepts exists.

There are improvable solutions which work when common understanding is shared. Common understanding would be for instance an agreed 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.

[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.

11.1.2 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) there exist already solutions which ORCHESTRA will have to expand to cover as much as 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 their 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 an even nearly complete fashion.

11.1.3 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 on the semantic level as much as possible.

11.1.4 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).

11.1.5 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 syntactical (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, there it would be clear, that both concepts are attributes representing a temperature and that both have a measure, but not the same. The mapping then would require to transform from one measure 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 there exist solutions for the sharing in the case of tightly coupled systems (e.g. the sharing of complex mathematical simulation.

ORCHESTRA will have to develop solutions for the sharing of services supporting semantic interoperability also in the case of loosely coupled systems.

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 / users)	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 42: ORCHESTRA Development Dimensions

11.1.6 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 of 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 first 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).

11.1.7 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.

11.1.8 Scale (# of semantically integrated information systems/users[2])

The number of integrated systems which will cooperate in the OSN is intended 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.

11.1.9 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 is increasing with its life span. Run-time adaption (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. On the basis of some existing approaches ORCHESTRA will develop solutions.

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

12 Annex A.2: Requirements for the OSN and the OA

In this chapter 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 from 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 consequences*. These steps are illustrated in Figure 26.

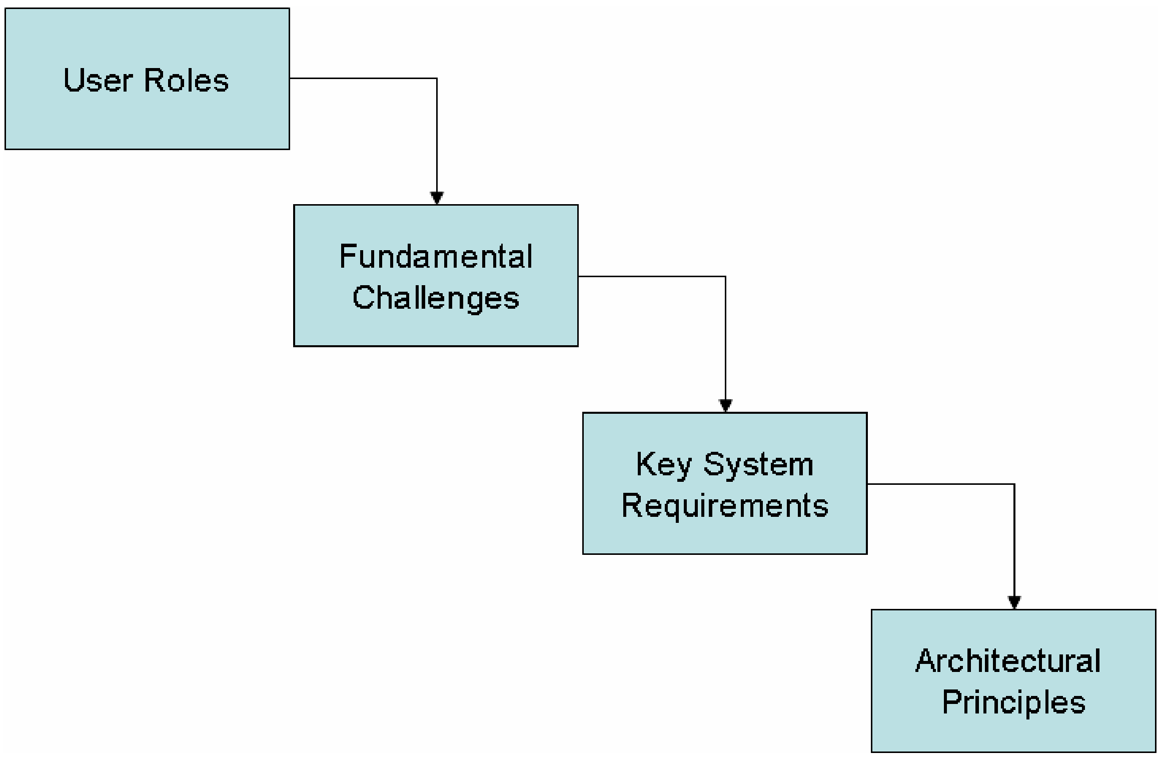


Figure 26: Line of argument to find the system requirements

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 43. 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:

a. In the following sub-sections, all occurrences of terms appearing in Table 43 are marked in italics in order to emphasise their dedicated meaning.

b. Please do note that all entries in the requirement tables do point forward and backward in the argument chain. In case of the backward pointing, you have already read the definition of a term. In case of the forward pointing you will find the definition of a term in a later chapter of the document.

User Roles			Fundamental challenges	Key System Requirements			
Service Devel-oper/System Administrator	Service pro-vider	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 Independ-ence	✓			
Generic Infrastructure	✓		✓	
Self-describing Components			✓	✓

Table 43: ORCHESTRA System Requirements

12.1 User Roles

In the field of Human Computer Interaction (HCI) system requirements are identified by using a user-centric approach. Three categories of system users can be identified:

- Service developer/system administrator (primary user in HCI terms)
- Service provider (secondary user in HCI terms)
- End user (tertiary users in HCI terms)

12.1.1 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		
→ Fundamental challenges	Scale and Scope Transparency (Hidden Process Complexity)	The service developer/system administrator has a 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 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.

12.1.2 Service provider

The service provider typically uses already 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.

User Role: Service provider		
→ Fundamental challenges	Integration/Collaboration Long Lifetime Quality Transparency (Hidden Process Complexity) Access Control	The service provider has a vested interest in five fundamental challenges. To provide new services to developers of end-user applications, they will require Integration and Collaboration in order to better integrate data and services into their offerings. Having provided such new services, they will want an OSN to have a Long Lifetime so that their investment will be secure. In order to assure the quality of the applications they provide to others, they will require Quality of the data and services which they integrate for this purpose. They will expect Transparency in the data and services they integrate so that they can cost-effectively employ them in their own offerings. And they will need Access Control in order to protect access to the data or services which they themselves provide. Access control is not trivial to provide in the anticipated distributed environment, as solutions to provide single-sign-on along with trust across different security domains are both a technological and a social challenge.

12.1.3 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.

12.2 Fundamental challenges

This subchapter 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 subchapter.

12.2.1 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 for such a problem would be the processing of a search request. As long as the information to search 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 a certain set of 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 can not 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.

12.2.2 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 done 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 reside in two different "information" worlds and technologies which are not well integrated. There is no common systematic approach on how spatial and non-spatial information and computation services 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. Especially 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.

12.2.3 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 in allowing on one side these autonomous changes but limiting on the other side 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.).

12.2.4 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 of 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 where 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

12.2.5 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.

12.2.6 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 on how to handle 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 Usability Accountability	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.

Access control is related to authorization and authentication:

- Authorization

  Authorization 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.

12.3 Key System Requirements

This subchapter describes key system requirements which have been derived form the fundamental challenges identified in the previous subchapter. 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 subchapter.

12.3.1 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 e.g. 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.

12.3.2 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

12.3.3 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 couple 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 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 systems 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 can not 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 Self-describing Components	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 standardized and familiar. Self-describing Components will dramatically improve the Usability of the system because they facilitate to be integrated.

12.3.4 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 which need 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.

12.4 Architectural Principles

This subchapter 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 action which will be applied after each final version of the RM-OA and final versions of all deliverables related to the specification and implementation of the RM-OA.

12.4.1 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 Consequence: 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.

12.4.2 Loosely Coupled Components

It is essential that the components involved in 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 Consequence:	Loosely Coupled Components	
← Key System Requirements:	Openness	Loose Coupling of Components often facilitates Openness.
	Scalability	To achieve a scalable system it is reasonable to build it with loosely-coupled components.

12.4.3 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 Consequence: Technology Independence		
← Key System Requirements:	Openness	To remain open over time, the system must maintain independence from particular technologies.

12.4.4 Evolutionary Development - Design for Change

The OSN cannot be put into place at one time, 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 Consequence: Evolutionary Development		
← Key System Requirements:	Openness	Evolutionary Development will considerably facilitate an open system.
	Scalability	An "Evolutionary Development" approach will allow the system to be scaled up or down over the whole period of operation.

12.4.5 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 Consequence: 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.

12.4.6 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 changes 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 Consequence: 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 standardized and familiar.
	Usability	

12.4.7 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 provide means to provide trace-, monitoring-, logging-facilities.

Architectural Consequence: 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	