Karlsruher Schriften zur Anthropomatik

Band 5

Thomas Usländer

Service-Oriented Design of Environmental Information Systems

KIT Scientific Publishing

Thomas Usländer

Service-Oriented Design of Environmental Information Systems

Karlsruher Schriften zur Anthropomatik
Band 5
Herausgeber: Prof. Dr.-Ing. Jürgen Beyerer

Lehrstuhl für Interaktive Echtzeitsysteme
Karlsruher Institut für Technologie

Fraunhofer-Institut für Optronik, Systemtechnik und
Bildauswertung IOSB Karlsruhe

Service-Oriented Design of Environmental Information Systems

by
Thomas Usländer

KIT Scientific Publishing

Dissertation, Karlsruher Institut für Technologie
Fakultät für Informatik,
Tag der mündlichen Prüfung: 20.01.2010

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

KIT – Universität des Landes Baden-Württemberg und nationales
Forschungszentrum in der Helmholtz-Gemeinschaft

KIT Scientific Publishing 2010
Print on Demand

ISSN: 1863-6489
ISBN: 978-3-86644-499-7

Service-Oriented Design of Environmental Information Systems

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)
vorgelegte

Dissertation

von

Thomas Usländer

aus Pforzheim

Tag der mündlichen Prüfung:	20.01.2010
Erster Gutachter:	Prof. Dr.-Ing. Jürgen Beyerer
Zweite Gutachter	Prof. Dr. rer. nat. Sebastian Abeck
	Prof. Dr. rer. nat. Lars Bernard

Zusammenfassung

Serviceorientierter Entwurf von Umweltinformationssystemen

Problemstellung

Die Behandlung übergreifender Aufgaben im Bereich des Umwelt-Monitorings und des Umweltrisikomanagements erfordert eine flexible Fusion von Informationen aus zumeist dezentralen sowie fachlich und administrativ getrennten Umweltinformationssystemen. Umweltinformationssysteme liefern eine jeweils fachlich, räumlich und zeitlich eingeschränkte Sicht auf den Zustand natürlicher Phänomene unserer Umwelt, z.B. die Qualität eines Wasserkörpers. Aus systemtheoretischer Sicht sind es eigenständige Module mit hoher Kohäsion. Eine fusionierte Sicht wird ermöglicht durch eine lose Kopplung der Module zu einem offenen „System aus Systemen“ basierend auf den Prinzipien der serviceorientierten Architektur (SOA) und den Festlegungen internationaler Organisationen für Geo-Dienste und Informationsmodelle. Im Falle der Umweltinformationssysteme sind dies die Standards des Technischen Komitees ISO/TC 211 „Geoinformation/Geomatik“ und die Empfehlungen des Open Geospatial Consortium (OGC).

Beim Entwurf von Umweltinformationssystemen stellt sich für den Systemarchitekten die Frage, wie im Zuge der Anforderungsanalyse und des Systementwurfs passende Fähigkeiten der Sub-Systeme gefunden und in die Systemarchitektur optimal eingebunden werden können. Relevante Fähigkeiten sind Dienst- und Informationsangebote sowohl auf Typ-Ebene (z.B. Spezifikation eines generischen Dienstes zur Interpolation von Messwerten) als auch auf Instanzen-Ebene (z.B. installiertes Dienst zur Beschaffung von Pegelmesswerten in einem Flussabschnitt).

Der Beitrag der vorliegenden Arbeit zur Lösung dieses Problems ist die Spezifikation einer für diese Klasse von Informationssystemen zugeschnittenen Entwurfsmethode.

Bisherige Arbeiten

Obwohl sich die Prinzipien und die Technologie der SOA in der Industrie etabliert haben, bleiben dazu passende Entwurfsmethoden und deren Werkzeugunterstützung ein Forschungsthema. Die heutigen Methoden beruhen zumeist auf Ausprägungen einer modellgetriebenen Software-Architektur (MDA) für serviceorientierte Systeme. Ausgehend von einer plattformunabhängigen Modellierung der Systemanforderungen wird durch schrittweise, möglichst automatisierte Spezialisierung die SOA abgeleitet.

Initiativen konkurrierender Organisationen arbeiten an abstrakten Meta-Modellen für Dienste, ohne dass sich bislang ein Standard herausgebildet hat. Die bestehenden Entwurfsmethodiken unterscheiden sich daher im Wesentlichen durch die Form der Dienstspezifikationen auf verschiedenen Abstraktionsebenen, die Werkzeugunterstützung sowie den Abdeckungsgrad des Lebenszyklus eines Systems. Während die SOMA-Methode beispielsweise den kompletten Lebenszyklus umfasst und in eine proprietäre IBM-Werkzeugumgebung integriert, konzentrieren sich die Entwurfsmethoden des SeCSE-Projekts auf die Term-Disambiguierung bei der Anforderungsana-

lyse und die Abfrageunterstützung bei der Suche nach vorhandenen Diensten. Komplementäre Ansätze beginnen den Entwurfsprozess mit einer formalen Beschreibung von Geschäftsprozessen und leiten daraus schrittweise und werkzeuggestützt die Spezifikation von Dienstketten und Schnittstellen von informationstechnischen Diensten (insbesondere Web Services) ab. Aktuelle Vergleiche bescheinigen den bestehenden Entwurfsmethoden noch zahlreiche Mängel in der Durchgängigkeit der Modellabstraktionen und der Einbeziehung der Anwendersicht. Einen Ansatz zur Lösung dieser Probleme versprechen Semantische Dienstumgebungen. Diese definieren die Anwenderanforderungen als Ziele und bilden sie über Ontologien auf Dienstbeschreibungen ab. Allerdings bringen diese Dienstumgebungen eine hohe Komplexität mit sich und haben sich bis heute weder in der Standardisierung noch in der Praxis durchgesetzt.

Keine der bekannten Entwurfsmethoden unterstützt bislang explizit die Besonderheiten offener, serviceorientierter Umweltinformationssysteme auf der Grundlage der ISO/OGC Standards oder nutzt ihre Möglichkeiten für den Systementwurf aus. Dies betrifft insbesondere die gezielte Recherche nach bestehenden Typen und Instanzen von Standard-Diensten und Informationsangeboten über Geo-Katalogsysteme sowie die Einbettung in entsprechende Architektur-Referenzmodelle.

Methodischer Ansatz der Dissertation

Der neue Ansatz leitet sich von der Beobachtung ab, dass in offenen Geo-Informationssystemen die Distanz zwischen Anforderungs- und Fähigkeitsmodell (z.B. das Metadatenmodell der Dienste) zu groß ist, um eine effektive Werkzeugunterstützung und eine nachvollziehbare Dokumentation des Entwurfsprozesses zu erreichen. Ein Kernstück der Entwurfsmethode ist deshalb die Konzeption einer einheitlichen Sprachebene, genannt „Semantisches Ressourcennetz", für beide Modelle.

Die Entwurfsmethode setzt voraus, dass die Anforderungen als Anwendungsfälle mit ihren funktionalen, informationellen und nicht-funktionalen Eigenschaften textuell oder semi-formal beschrieben sind. Andererseits müssen die Fähigkeiten der Diensteplattform formal spezifiziert und über ein Meta-Informationssystem (z.B. ein Katalogdienst oder eine Internet-Suchmaschine) abrufbar sein. Zu den Fähigkeitsbeschreibungen gehören die Syntax und Semantik der Schnittstellen und des Informationsangebotes eines Dienstes.

Im Gegensatz zu Diensten im betriebswirtschaftlichen Umfeld sind standardisierte Geo-Dienste sehr generisch ausgelegt, um den Grad ihrer Nutzbarkeit und Wiederverwendbarkeit zu erhöhen und ihre Anzahl gering zu halten. Dies hat allerdings den Nachteil, dass die Dienstoperationen (z.B. *GetFeature* oder *DescribeSensor*) sehr allgemein sind und sich deren eigentliche Bedeutung, und damit deren Nutzbarkeit, erst aus der semantischen Analyse der Informationselemente ergibt, die mit ihnen als Ein- oder Ausgabeparameter verbunden sind. Dadurch entsteht eine große semantische Lücke zu den Anforderungen des Anwenders, die zumeist mit den Begrifflichkeiten eines Fachmodells (z.B. aus der Hydrologie) ausgedrückt sind.

Das Semantische Ressourcennetz hat das Ziel, die Komplexität und Distanz der Abbildung von Anforderungen zu Fähigkeiten zu verringern. Es ist aus dem Architekturstil „Representational State Transfer“ wie folgt abgeleitet:

- Eine Ressource ist ein Informationsobjekt mit einem eindeutigen Bezeichner, auf das nur eine eingeschränkte Menge von Operationen mir klar definierter Semantik angewandt werden kann (u.a. Erzeugen und Löschen der Ressource, Lesen einer Ressource in einer Repräsentationsform und Schreiben der Ressourcenattribute).
- Eine Ressource hat mehrere Attribute, die den Zustand der Ressource beschreiben.
- Ressourcen werden durch relationale Attribute zu einem Netz verknüpft und durch Modellreferenzen zu Konzepten einer Ontologie semantisch annotiert.
- Eine Ressource kann in mehreren Repräsentationsformen dargestellt werden (z.B. als Diagramm, Tabelle, Karte oder als Dokument).

Der Entwurfsprozess beginnt mit der Umformulierung der Aktionssequenzen der Anwendungsfälle in ein Netz „angeforderter Ressourcen“ (*rephrasing*). Unabhängig von der jeweiligen Entwurfsaufgabe können die Dienstfähigkeiten als Netz „angebotener Ressourcen“ zusammen mit dem Dienst selbst oder abgekoppelt in einem Dienstkatalog beschrieben werden (*publishing*). Ein Entwurfsschritt entspricht dadurch einer Suche in dem Netz der „angebotenen Ressourcen“ gemäß den Kriterien der „angeforderten Ressource“ (*discovery*) sowie der Bewertung und Auswahl von Ressourcenkandidaten (*matching*). Da der Vergleich der Operationen trivial ist, reduziert sich die Such-Komplexität auf den syntaktischen und semantischen Vergleich der Ressourcen-Attribute. Durch semantische Annotation kann zudem Domänenwissen herangezogen werden, um ähnliche Ressourcen-Kandidaten zu finden, z.B. eine Suche nach Windmesswerten in „Baden“ findet Meteorologie-Messwerte in „Süddeutschland“.

Eine erfolgreiche Suche und Auswahl wird als Entwurfsentscheidung in Form einer Abbildung zwischen Ressourcen (Benutzt-Relation) dokumentiert, so dass das Ressourcennetzwerk quasi den Entwicklungsstand nach jedem Entwurfsschritt widerspiegelt. Dadurch kann der Systemarchitekt zusammen mit dem Anwender die Abbildung der Anforderungen auf die Fähigkeiten der Diensteplattform jederzeit nachvollziehen (*feedback generation*). Die Entwurfsmethodik verfolgt einen iterativen Ansatz, der es ermöglicht, sowohl die Systemarchitektur als auch das Anforderungsmodell schrittweise weiterzuentwickeln (Ko-Entwurf). Jeder Iterationsschritt baut auf dem vorhergehenden auf, verfeinert oder erweitert ihn und ergänzt dabei das Ressourcennetz um neue Knoten.

Die Entwurfsmethodik ist eingebettet in einen übergeordneten Entwurfsprozess und ein Referenzmodell für eine serviceorientierte Architektur. Dieses Referenzmodell ist das Ergebnis einer Interpretation des ISO Referenzmodells für verteilte Informationsverarbeitung (RM-ODP) für eine serviceorientierte Architektur mit Geo-Bezug.

Entscheidend ist hierbei der Ansatz, die Architektur eines verteilten Systems unter verschiedenen Blickwinkeln (*viewpoints*) zu betrachten und die Entwurfsartefakte entsprechend zu dokumentieren. Die Entwurfsmethodik konzentriert sich hierbei auf den abstrakten Entwurf der Dienst- und Informationsmodelle. Dies bedeutet, dass die Entwurfsartefakte im Sinne einer modellgetriebenen Entwicklung zunächst unabhängig von den technologischen Aspekten einer Diensteplattform beschrieben werden.

Die Arbeit spezifiziert die Entwurfsmethodik auf abstrakter Ebene in der Unified Modeling Language (UML). Sie skizziert zudem eine Implementierungsarchitektur auf der Grundlage eines Ontologie-Editors, eines Geo-Katalogdienstes mit ontologiegestützter Term-Expansion und eines semantischen Annotationsdienstes.

Evaluation und Ausblick

Die Evaluation erfolgt auf der Basis eines eingeführten Kriterienkatalogs für serviceorientierte Entwurfsmethoden. Dieser ist angereichert um eigene Bewertungskriterien, die aus den Erfahrungen bei der Entwicklung von offenen Umweltinformationssystemen abgeleitet wurden. Der Modellierungsansatz wird validiert durch Anwendungsfälle aus dem Bereich des Umweltrisikomanagements und ihrer Abbildung auf angebotene Ressourcen standardisierter OGC Sensordienste. Zudem wird die Nutzung und die Weiterentwicklung des Referenzmodells in Projekten angrenzender Fachbereiche (z.B. Umwelt und Gesundheit) erläutert.

Die Arbeit schließt mit einem Ausblick auf mögliche sich anschließende Forschungsarbeiten. Dazu gehören insbesondere die Implementierung und die Validierung der Entwurfsmethode in einen größeren SOA-basierten Entwicklungsprojekt sowie die Untersuchung des integrierten, schrittweisen Entwurfs von Domänenontologien im Zuge der Anforderungsanalyse.

Acknowledgements

This thesis is the result of about five years work carried out in parallel to my tasks as research scientist at the Fraunhofer IOSB in Karlsruhe, Germany. Its edition would not have been possible without the support of many people in my entourage. I would like to cordially thank all those who have been directly or indirectly concerned with this work.

My primary thank is addressed to my supervisor Prof. Dr. Jürgen Beyerer, who encouraged and endorsed me in my scientific ambition and gave me valuable advice. Due to the multi-disciplinary nature of the subject of my thesis I was very grateful to win two distinguished experts to act as co-supervisors: Prof. Dr. Lars Bernard representing the geospatial community and Prof. Dr. Sebastian Abeck representing the software and service engineering community. Thank you very much for your scientific guidance and the inspiring discussions!

The initial idea and motivation to start the PhD work came from the European research projects ORCHESTRA and SANY. Many thanks go to Prof. Dr. Ralf Denzer, one of the initiators and major drivers of both projects, who enabled me to jointly lead a team of highly skilled and dedicated computer scientists with the common vision to design a generic system architecture for environmental risk management applications. Thank you, Ralf, and all the team members for your excellent contributions and reviews, and not to forget, for the organisation of the unforgettable social events associated to the project meetings! Special thanks also deserve the staff and the members of the Open Geospatial Consortium whose open-minded attitude permitted a constructive discussion of the project and research results.

I got the practical foundation and expertise from the research projects of the cooperation network KEWA organized by the Ministry of the Environment in Baden-Württemberg. Many thanks are due to Roland Mayer-Föll, the coordinator and tireless driving force behind this network of excellence, and all the KEWA partners for their continuous support.

However, without the motivating atmosphere and the team spirit within the department "Information Management" of Fraunhofer IOSB, I would not have dared to start this work. For many years this atmosphere has been influenced and cultivated by Gottfried Bonn. Gottfried, I thank you for your precious advice as recognized computer science expert and colleague throughout my professional career! Furthermore, I give thanks to all members of my research group for their independent and professional work. Please excuse if some of my daily or project duties took longer or were affected during the edition of this work.

The participation to two doctoral consortia associated to European conferences on Semantic Web technologies and Information Systems helped to better explain my research to experts of multiple domains. The hints I got from experienced supervisors and the inspiring community with other doctoral students were an indisputable help to clear hurdles. Thanks to all of you!

Last but not least I want to thank my family for their sympathy and patience with me in the recent years. I am very grateful to my mother who enabled me a high-class course of education despite of difficult circumstances. Cordial thanks deserve my wife Martina and our two, meanwhile adult, daugthers Anja and Lena for their continuous encouragement and moral support. They reminded me that "the art of relaxation is part of the art of working" (John Steinbeck) and also laid the foundations that I could translate both arts into practice.

Karlsruhe, March 2010 Thomas Usländer

Table of Contents

1 Introduction

1.1 Motivation

Imagine … you are an expert in a thematic domain such as hydrology. According to the state of the art in environmental science you have profound expertise about the cause and effect relationships between hydrological phenomena and the impact on nature, infrastructure, society and human health. You are working for an environmental agency and you know all about the environmental legislation, the monitoring obligations of environmental parameters and the threshold values that must not be exceeded.

Now you get the request to introduce a new water information system that allows one to predict future values of critical parameters for water quality (e.g. nitrate concentration). The prediction shall be represented as a regular grid that covers a whole large-scale water body that goes beyond your area of responsibility. You know that this request means to fuse observation values from existing water monitoring systems offered by your own but also by other environmental agencies in your region. The fused information has to be fed into selected environmental models. The model results shall be visualized in maps. The monitoring systems rely upon water samples taken by humans and analysed offline in laboratories, or upon online sensors with on-site chemical analysis capabilities. The models require additional information from other application domains such as meteorology, or from remote sensing products offered by space agencies, e.g. maps about land use. The system design and development shall be carried out by an external software company. Thus, your first task is to analyse the problem and write down a first set of (user) requirements in terms of which information is required, which functions need to be offered and which additional characteristics the system shall exhibit, e.g. who may access which information and which functions.

Imagine … you are a computer scientist working as a system designer and information system architect in a software company. Your primary expertise is in innovative middleware technology, especially Web services, supporting large-scale distributed geographic information systems (GIS), e.g. Environmental Information Systems (EIS). You know all about the latest technological evolutions in tools and standards. You are involved in international expert groups that formulate guidelines and rules about how to construct spatial data infrastructures in order to improve the interoperability between GIS components.

Now your company has won the tender to develop the water information system described above. You have been assigned to draft the system architecture following the requirements written down by the hydrologist. However, the tender contains additional demands and side conditions: The architecture shall reuse as much as possible standard services and capabilities of the existing systems in order to get an open and cost-effective solution. The solution shall be flexible such that future needs may be satisfied, e.g. it shall be possible to easily integrate new sensor systems. The architectural approach shall even be applicable to other application domains, e.g. air quality monitoring.

The following questions arise: What methodology is available that supports the design process for this type of system and explicitly considers the side conditions? What language and notations shall be used such that the requirements of the hydrologist are understood by the computer scientist? How can the hydrologist trace and evaluate that the capabilities of the system and its architecture drafted by the computer scientist fulfils the requirements?

The lack of satisfying answers motivates this thesis from an engineering point of view.

What are the deficiencies in the design of software architectures today that motivates the subject of this thesis from the scientific point of view?

Environmental Information Systems (EIS) play a key role in environmental monitoring and environmental risk management tasks. They provide information about the past, current and future status of environmental phenomena, e.g. the chemical or biological quality of a water body, and thus contribute to our understanding of the environmental situation. The performance of large-scale or multi-disciplinary environmental tasks requires a fusion of environmental information stemming from several decentralized EIS that usually cover a thematic domain (e.g. air or water) in a delimited region and are managed by dedicated public authorities.

From the technological point of view EIS are information systems that deal with geospatial information and services with a reference to a location on the Earth. They allow the user to store, query and process environmental information and visualize it in thematic maps, diagrams and reports. EIS are associated with heterogeneous sensors and/or environmental models that deliver measured or calculated observations about environmental phenomena.

From the point of view of system theory EIS are autonomous modules with a high level of inner cohesion. They provide a restricted view upon the environment which is limited by temporal, spatial and thematic boundaries. Information fusion is then enabled by a loose coupling of EIS whereby the actual configuration of the resulting system-of-systems is dependent upon the environmental question to be answered. These system requirements are best met by the principles of **Service-oriented Architectures (SOA)**:

- SOA principles enable the sharing of geospatial information and services and their composition to higher-level resources across organizational and administrative boundaries in a loosely-coupled but controlled manner. This is essential for EIS as environmental phenomena are not limited to boundaries drawn by humans.
- Effective and flexible interactions between EIS require an agreement within the developer community about the syntax and semantics of service interfaces and information models. Thus, as a crucial side-condition in the design of in-

frastructures for EIS, the standards of the technical committee "Geographic information/Geomatics" of the International Organization for Standardization (ISO) and the recommendations of the Open Geospatial Consortium (OGC) have to be considered.

There are several initiatives on national, European and world-wide scale that define geospatial SOAs as an underlying foundation for EIS. However, one of the challenges is the design of applications that exploit the potential and the capabilities of such geospatial service networks.

Service engineering emerges as an own research discipline, strongly inheriting from the principles of software engineering, but enhancing them towards the open-world assumption of the SOA approach. The "open-world" is characterized by the complexities of unforeseen clients and use cases in systems-of-systems environments, but it bears the potential of a significant return on invest by the controlled reuse of existing services, in the case of EIS, for instance, services of the emerging Sensor Web.

Numerous methodologies for service-oriented analysis and design have been described in the literature and partly embedded in software development tools. The deficiency today is that there is no design methodology that brings together the requirements and the expert knowledge of EIS users with the services and information offerings of existing EIS, and, in addition, explicitly obeys the guidelines and constraints of geospatial standards as side-conditions.

This thesis aims at filling this gap by proposing a **geospatial reference model** and an associated **design methodology** tailored to the EIS application domain.

1.2 Alignment between Business and IT Strategies

Although the design methodology proposed in this thesis is motivated and exemplified by the needs of distributed large-scale EIS it is not exclusively restricted to the domain of environmental informatics. On a broader scope, the subject of this thesis falls into the discipline of Information System (IS) research with a focus on the design of IS and its infrastructure. Henderson and Venkatraman (1993) argue that there are "essential alignments between business and information technology strategies and between organizational and information systems infrastructures". Hevner et al (2004) state that on the one hand, "information technologies (IT) are seen as enablers of business strategy and organizational infrastructure", and on the other hand "available and emerging IT capabilities are a significant factor in determining the strategies that guide an organization".

As a consequence, this alignment leads to a dualism in the design activities illustrated in Figure 1-1 based upon the original "strategic alignment model" of Henderson and Venkatraman (1993):

- Organizational design to create an effective organizational infrastructure that is derived from the business strategy, and
- Information system design to create an effective IS infrastructure that is derived from the Information Technology strategy.

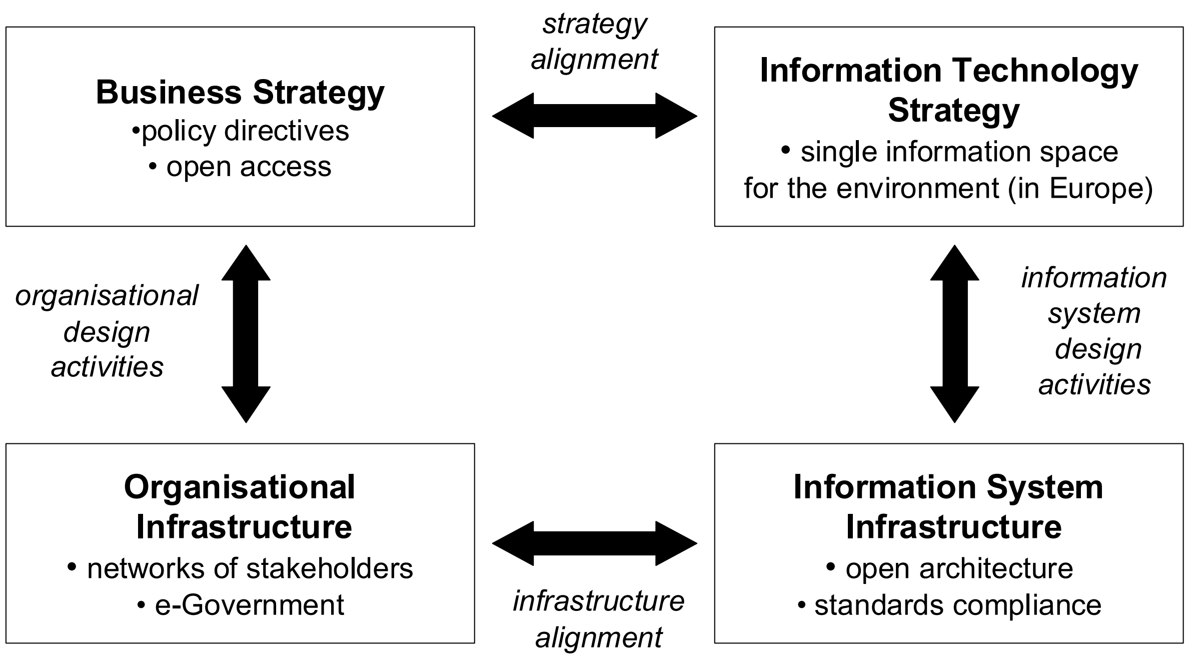

strategy alignment
Business Strategy
•policy directives
• open access
Information Technology Strategy
• single information space for the environment (in Europe)
organisational design activities
information system design activities
Organisational Infrastructure
• networks of stakeholders
• e-Government
infrastructure alignment
Information System Infrastructure
• open architecture
• standards compliance

Figure 1-1: Alignment between Business and IT Strategies, adapted to the Environmental Domain from Henderson and Venkatraman (1993)

For the purpose of this thesis, the strategies and infrastructures of Figure 1-1 are annotated by attributes that characterize the scope of the thematic domain which is in the main focus of this thesis: environmental risk management and environmental information systems (EIS), exemplified by the policies and strategies of the European Union (Usländer, 2009a).

- In Europe, the Business Strategy for environmental risk management is mainly driven by European and national environmental legislation resulting in policy directives such as the European Directive on Public Access to Environmental Information (EC 2003b) or the European Water Framework Directive (WFD) (EC 2000). These directives strengthen the need to exchange environmental information between the public and private stakeholders and to enable the access to environmental information by all parties (including the citizen) who may have a valid interest (Timmerman and Langaas (eds.), 2004).
- Translated to the IT domain, the Information Technology Strategy for environmental risk management is mainly determined by the ambition of the European Commission to create a “Single Information Space for the Environment in Europe (SISE)” (Coene and Gasser, 2007). The SISE is defined both as a vision and a need towards which all IT research activity in this domain has to be directed.

These strategies determine the design activities on both the organizational and the technical levels.

- The design of the Organizational Infrastructure (left side in Figure 1-1) follows the need to support European environmental legislation. Environmental agencies, environmental ministries, research institutes and private organizations are organized as nodes of organizational networks (e.g. networks of excellence for selected thematic domains) with defined levels of cooperation on managerial and technical level.
- The design of the Information System Infrastructure (right side in Figure 1-1) requires an open architecture. Ideally, it shall provide seamless access to information, services and applications across organizational, technical, cultural and political borders. "Open" here means that service specifications are published and made freely available to interested vendors and users with a view of widespread adoption. Furthermore, an open architecture makes use of existing standards where appropriate and possible, and contributes to the evolution of relevant new standards (Powell, 1991).

In the environmental domain we consider that the role of standards is an essential element in the alignment of business and IT strategies. The key value of standards in software engineering in general has been emphasized by the results of a Delphi study (Delphi, 2003) which "portray a shifting landscape where standards will provide the foundation for long term advances in the way software is built, bought and deployed". The importance of standards in creating geospatial information spaces that enable to overcome the "havoc of non-interoperability" has been described in a white paper of the Open Geospatial Consortium (OGC) (Reichardt, 2003) and stressed by a study of the NASA Geospatial Interoperability Office (Booz Allen Hamilton, 2005). The OGC Reference Model (Percivall (ed.), 2008) argues that "the enterprise return on investment in open interfaces is unquestionable today". It describes standards as "a set of rules that have been agreed to in some industry or public consensus forum such as the Internet Engineering Task Force (IETF), the World Wide Web Consortium (W3C), the International Organization for Standardization (ISO) or the OGC". Standards in the geospatial domain exist on both the abstract (i.e. platform-neutral) and the concrete (i.e. platform-specific) level in order to avoid being fixed to one technology such as for instance Web services.

1.3 Aspects of Service-Orientation

For this introductory section we adopt the SOA definition of Bieberstein et al (2006) that resulted from a survey of business executives: "A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components – services – that can be reused and combined to address changing business priorities." A more detailed discussion of SOA principles and definitions will follow in section 2.2.

Service-orientation has entered the practice in industrial software engineering already for several years. However, several architectural styles (i.e., set of architectural constraints) addressing how to realize the service paradigm on the concrete technological level are competing. On the one hand, there are Web services that follow the classical architectural style of remote invocation with arbitrary operational semantics (W3C, 2004a). On the other hand, there are the so-called RESTful Web services (Richardson and Ruby, 2007) that rely upon uniquely identifiable resources with a limited set of well-defined operations, following the Representational State Transfer (REST) architectural style of Fielding (2000). Basically, this competition on the technological level reflects the more fundamental discussion on the conceptual level to which degree the functional and the informational aspects determine the semantics of a service. This is especially relevant when drafting and assessing service designs.

Furthermore, the development and establishment of methodologies for the construction of an SOA is still considered as a key challenge for the success of SOA. Papazoglou et al (2007) state in their research roadmap for service-oriented computing that the "software industry now widely implements a thin SOAP/WSDL/UDDI[1] veneer atop existing applications or components that implement the Web services, but this is insufficient for commercial-strength enterprise applications." They claim that "SOA-based applications require a service-oriented engineering methodology that enables modelling the business environment, including key performance indicators of business goals and objectives; translates the model into service design; deploys the service system; and tests and manages the deployment".

This estimate of the importance of SOA engineering support is shared by Kontogiannis et al (2007). They have analysed research topics for service-oriented computing and categorized them into a business, engineering and operation domain. Among others, the engineering domain contains as two important research topics (1) "models to support strategic reuse of services", and (2) the "support of service-oriented development including model refactoring and incremental model synchronization". This thesis provides a reference model as a contribution to topic (1) and a design methodology as a proposal for (2).

Methodologies to support service-orientation in the design and analysis phase of information systems, in short, Service-oriented Analysis and Design (SOAD) methodologies, are still poorly developed (Offermann and Bub, 2009). The existing software engineering process models such as Object-oriented Analysis and Design (OOAD), the Rational Unified Process (Kruchten, 2000) and Component-based Development

[1] SOAP and WSDL are technologies of the Web services architecture of the World Wide Web Consortium (W3C, 2004). SOAP provides a standard, extensible, composable framework for packaging and exchanging XML (Extensible Mark-up Language) messages (W3C, 2007b). WSDL (Web Service Description Language) is the language to describe Web services (W3C, 2007a). UDDI (Universal Description, Discovery and Integration) is a specification of a Web services registry proposed by the Organization for the Advancement of Structured Information Standards (OASIS, 2004).

(CBD) (Koskela et al, 2007), are tailored to the design of applications which, although potentially distributed, comprise tightly-coupled components with an object-oriented programming interface. However, service-orientation shares many of the same goals as object-orientation. Erl (2008a) states that "it seeks to establish a flexible design framework that allows for the agile accommodation of ever-changing business requirements", and "is very concerned with minimizing the impact of change upon software programs already deployed and in use". The basic distinction between the two paradigms is one of scope. OOAD and CBD primarily address the requirements and the design of a single application or collections of related applications, while SOAD has an enterprise-centric or even a cross-enterprise perspective. This means that the functionality embedded in one service should potentially be reusable by other applications in the same enterprise (via the Intranet of the enterprise) or in other enterprises (via an Extranet or the Internet) without knowing these interactions already at design time. Van den Heuvel et al (2009) call these aspects the **open-world assumption** that must be met by service networks as one major tenet and challenge for software service engineering. The "open-world" is characterized by "unforeseen clients, execution contexts and usage" of services operating in "highly complex, distributed, unpredictable, and heterogeneous execution environments".

Erl (2008a) identifies Enterprise Application Integration (EAI), Business Process Modelling (BPM) and Aspect-oriented Programming (AOP) along with the emergence of the Web Services technology as additional influences in the evolution from object-orientation to service orientation (Figure 1-2).

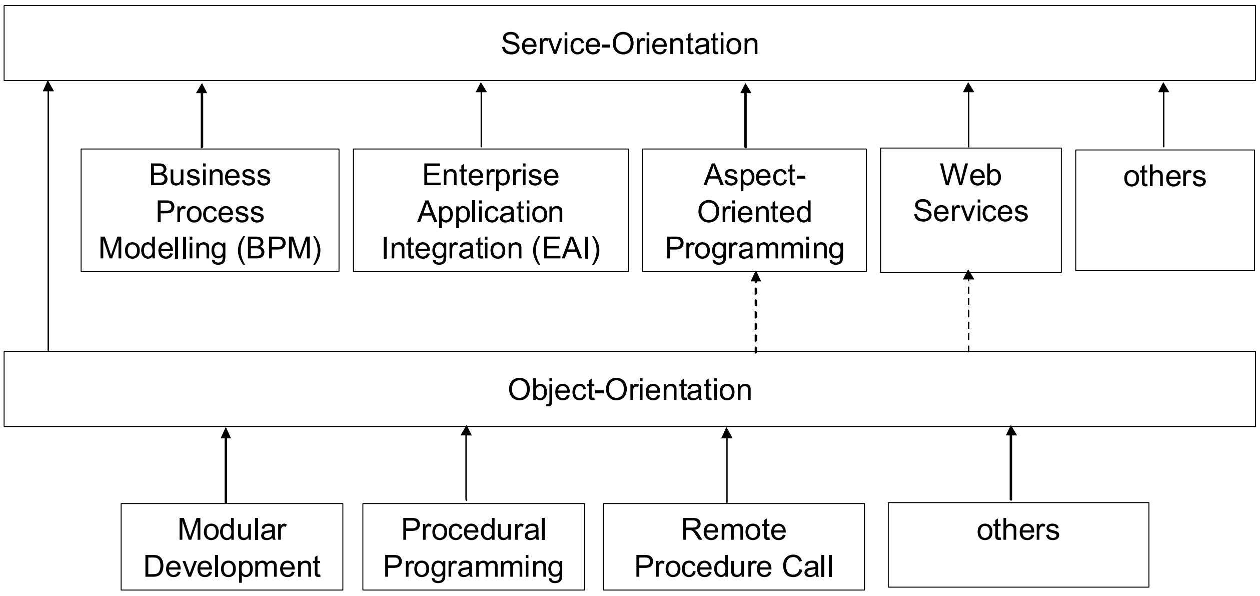


Figure 1-2: Evolution of Service-Orientation from Object-Orientation (Erl, 2008a)

Independent of but accelerated by the industry focus on SOA, "modelling" plays an increasing role in analysis and design. Bieberstein et al (2006) state that, "what used to be called analysis and design in previous technological eras is now often called "modelling". Such a modelling approach assumes that "there might be a continuum of definition, refinement, and transformation activities for analysing requirements, developing architectures and design, and generating software code for target execution platforms." The important question in software modelling is the level of abstraction from implementation and technological details, and the artefacts as the result of the modelling activity.

Apart from the different granularities of abstractions, Bieberstein et al (2006) also stress that the "number of artefacts increases dramatically as decomposition progresses top-down from enterprise models to business processes to services to components and objects", which is associated "with a great deal of undesirable redundancy at each of these layers." It is one of the challenges of an SOA design methodology and the supporting design tools to manage this redundancy and the mapping between the artefacts of the abstraction layers.

Furthermore, the demand for traceability (Hatley, Hruschka and Pirbhai, 2000) of the artefact mapping shall be satisfied. Applied to SOAD this means that the existence of a specified service shall be justified by related requirements (e.g. formulated in use cases), or vice-versa, it shall be checked if and how all requirements have been tackled in a system design.

1.4 Requirements for Service-oriented Design

1.4.1 Requirements, Design and Capabilities

Before starting the discussion about requirements for service-oriented design we clarify our understanding of the terms **requirement** and **design**. For this purpose we adopt the definitions of the IEEE Software Engineering Glossary (IEEE 610.12, 1990).

Definition (1.1):

A **requirement** is (1) a condition or capability needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) a documented representation of a condition or capability as in (1) or in (2).

Definition (1.2):

Design is (1) the process of defining the architecture, components, interfaces, and other characteristics of a system or component; (2) the result of the process in (1).

In order to fulfil the requirements of a user by the design of a system, there is a need to coordinate his/her actions with the system designer who is responsible for the design. A design process must explicitly support this coordination. The **user** represents the person who knows about the problem to be solved in the sense that a problem indicates the "differences between a goal state and the current state of a system" (Hevner et al, 2004). In this role, the user represents the view of the company or institution that has awarded the contract for the development of the system. The user is expert in the thematic domain of the problem but not necessarily expert in the technical details of the information system.

Thus, on the one side, the user transforms the problem into a set of **user requirement**s that represent expectations about the functionality and the characteristics of the resulting system, often provided in natural language associated with some diagrams. On the other side, there is the **system designer**[2] who is in charge of transforming the user requirements into a specification of the **system requirement**s. These should precisely describe the "external behaviour of the system and its operational constraints" (Sommerville, 2007). Therefore, there is a need for a more specialized notation such as structured natural language using standard forms or templates, or diagrams using graphical modelling languages.

In the software engineering literature, user and system requirements are often classified as functional requirements, that define "what the system should do", and non-functional requirements that define constraints on the system or on the process to design the system (Sommerville, 2007; Jacobson and Ng, 2005). In addition, Sommerville (2007) introduces domain requirements that "reflect characteristics and constraints" of a thematic domain as third class. For information systems these characteristics are primarily the terms, concepts and information elements that represent a thematic domain. Such informational requirements may be covered by a particular perspective of functional requirements that are implicitly embedded within functional requirements (Pohl, 2008). However, due to the importance of a common semantic understanding of the information elements across the system functions, we argue for handling them explicitly on the same level as the functional and the non-functional requirements.

As a consequence, we categorize the requirements into:

- **Functional requirements** that describe the functions and the processes that a system has to support,
- **Informational requirements** that describe the major terms, concepts of the application domain and information elements the system has to deal with, and
- **Non-functional requirements** (NFR) subdivided into:
    - **Qualitative requirements** that describe qualitative constraints upon the behaviour of the system, dealing, for instance, with dependability, performance and security aspects[3].
    - **Side conditions**[4] that describe constraints upon to the design process (Sommerville, 2007), dealing, for instance, with the request to use stan-

[2] We also use the term "system architect" as a synonym to system designer.

[3] In aspect-oriented software development methods these are also called cross-cutting concerns (Jacobson and Ng, 2005).

dards, to apply a given design methodology or to produce design artefacts according to a given template.

From the viewpoint of the system designer, the major advantage of an SOA is design efficiency and sustainability: already existing services shall be re-used if fitting and possible, new services shall be designed as generic as possible such they may also satisfy future, still unknown requirements. Thus, in the course of the design process, the system designer has to evaluate system requirements against existing services. This evaluation shall happen both on instance and type level:

- Re-use on **service instance** level is possible if a service component deployed in a service network fulfils the elements of the system requirements including their functional, informational and non-functional aspects. An example is a Web service that is operated by an environmental agency, is publicly accessible and delivers ozone measurement values in the requested region with a sufficient degree of availability and accuracy. If such characteristics, or at least a subset of them, are part of the user requirements, such a service instance may directly be integrated into the application instead of specifying and implementing an alternate solution.
- Re-use on **service type** level enables to re-use service specifications completely or in parts (e.g. in terms of service interfaces) although running instances of a chosen service type cannot be directly integrated, for instance, because of an insufficient access control policy of the service provider. However, re-use on type level paves the way for interoperable solutions and the usability of existing products or frameworks. This is reasonable especially if there is the side condition to apply standards as far as possible.

A pre-requisite to re-use service types and instances is the availability of corresponding descriptions. We use the term **capability** for these descriptions.

Definition (1.3): A **capability** is a description of service types or service instances including their functional, informational and qualitative aspects[5].

Capabilities are meta-information documents that consist of a mixture of textual descriptions and formal specifications (e.g. for the signature of the service). Capabilities of service types are stored in service inventories (Erl, 2008a) whereas capabilities of service instances may be stored "close" to the service instance itself, in service reposi-

[4] Pohl (2008) classifies this kind of requirements as a third class besides functional and qualitative requirements.

[5] This use of the term capability refines its definition of the SOA Modelling Language (SoaML) (OMG, 2008c). SoaML capabilities "represent an abstraction of the ability to affect change". Furthermore, this definition is compliant to Erl (2008b) who considers a "service as a container of capabilities associated with a common purpose based on a common functional context". As a refinement of Erl's definition, we explicitly include the informational and qualitative aspects into the "functional context".

tories[6] or in both places. For environmental information systems the geospatial service inventories of standardization organizations such as ISO and OGC play are major role. They comprise a significant and very important side condition for the service-oriented design.

1.4.2 Problem of Service-oriented Design

Taking requirements and capabilities as a conceptual foundation, the kernel challenge for a service-oriented design of an information system boils down to the question of how the requirements of the user can be assessed against the already existing capabilities of service platforms. Basically, functional, informational and qualitative requirements at one abstraction layer (A) have to be semantically matched to capabilities of another abstraction layer (B) taking side conditions into account (Figure 1-3).

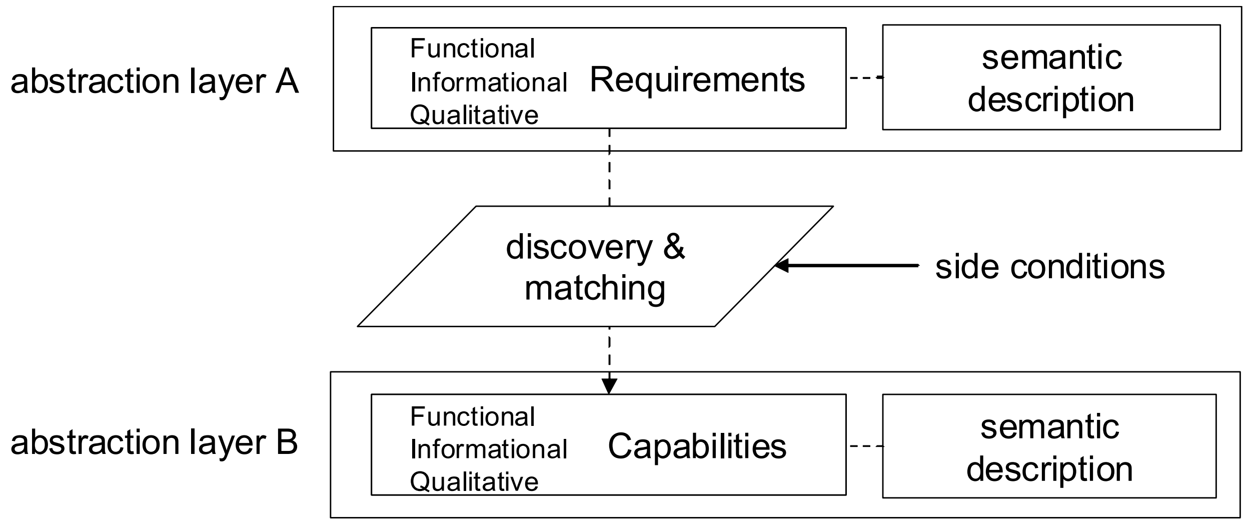


Figure 1-3: Mapping of Requirements to Capabilities

A pre-requisite to the matching is the discovery of possible capabilities which may fulfil the requirements. These are called **candidate capabilities**. The matching activity selects among the candidate capabilities those who fit best to the requirements, taking side conditions, such as the need to deliver standards-compliant services, explicitly into account. Discovery and matching need some associated semantic description of both requirements and capabilities in order to be effective. Such semantic descriptions give meanings to the terms used in the specifications. Their representation forms range from text in combination with a glossary in which all important terms are defined up to specifications in formal logics (section 3.3).

The combination of the tasks of discovery and matching serves as a generic mechanism to bridge the gap between heterogeneous descriptions and/or expectations and

[6] In the geospatial community such service repositories are also called "catalogues".

plays a key role in a service-oriented design[7]. This kernel problem repeatedly occurs when user requirements are broken down into multiple steps across several abstraction layers. In fact, capabilities turn into requirements for the next design step. Furthermore, there is a widely recognized need to package the development of requirement artefacts on a higher abstraction level and the development of architectural artefacts on a lower level (here: service type specifications and service instances) into one single design step, leading to a so-called co-development[8] of requirement and architectural artefacts (Pohl and Sikora, 2007). Such a co-development requires a stepwise refinement of the design artefacts.

The application of the MDA platform hierarchy (section 2.3.3) to the iterative requirements-to-capabilities mapping results in the following analysis, design and engineering steps (Figure 1-4):

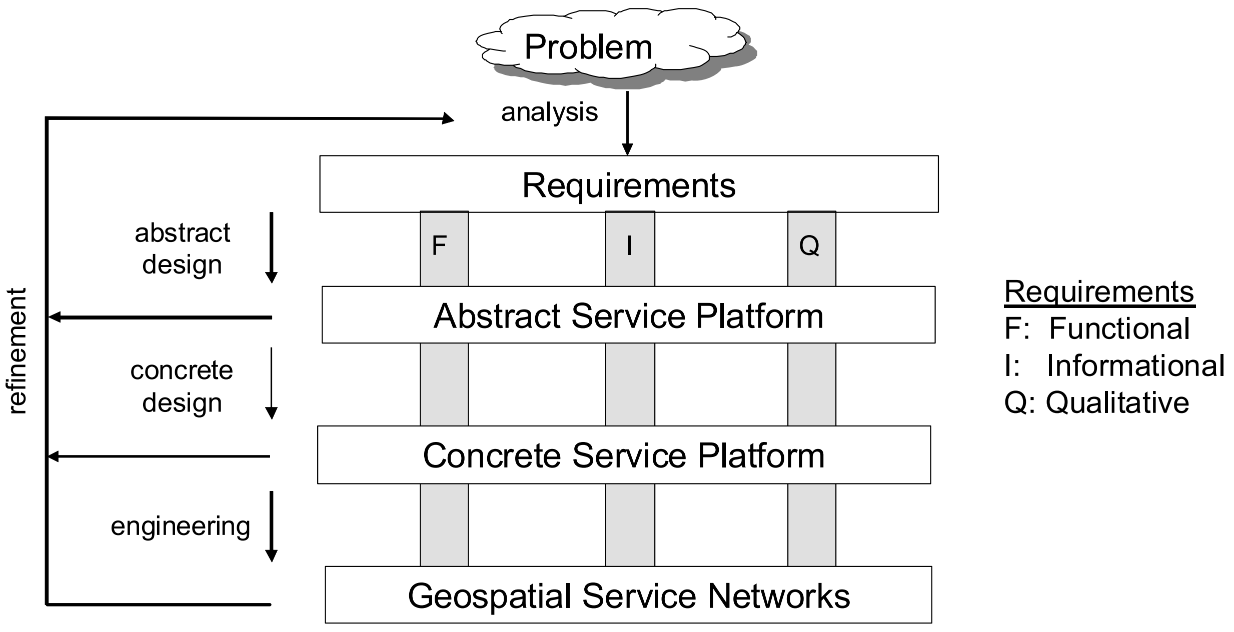


Figure 1-4: Analysis, Design and Engineering Steps

1. The **Analysis** step in which the user analyses the problem and expresses the outcome in the form of user requirements.

   Example: A use case that requests to *"get a diagram containing the average nitrate concentration of the groundwater bodies in the Upper Rhine Valley of the last 10 years"*.

2. The **Abstract Design** step in which the user requirements are transformed by the system designer into system requirements which then have to be matched

[7] The tasks of discovery and matching realise a "search process" that is inherent to problem solving processes according to Hevner et al (2004). The problem space in which an "effective solution" is searched for consists of the set of all possible capabilities that may satisfy the requirements.

[8] The principles of the co-design of requirements and architecture are further elaborated in section 2.3.3.

with the capabilities of an abstract service platform, i.e. a service platform that abstracts from the peculiarities of service platform technologies. Usually, this is not a direct mapping, but a tedious task that is carried out in multiple iterations and in close collaboration between the system designer and the user. The results of this step are capability specifications of an abstract service platform.

Example: Provide a service that enables to *"get observation values with a sampling time in the interval [2000-01-01, 2009-12-31] for the environmental parameter "nitrate" for all groundwater monitoring stations that are located in the Upper Rhine Valley"*.

3. The **Concrete Design** step in which the capabilities of the abstract service platform turn into requirements for the design of the concrete service platform and finally result in a specification of its capabilities. Again, depending on the characteristics, existing capabilities and constraints of the concrete service platform, this is not a direct or one-to-one mapping.

   Example: The *getObservation* operation request of the OGC implementation standard Sensor Observation Service (Na and Priest, 2007)[9]:

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetObservation version="1.0.0" service="SOS">
<sos:offering>Groundwater</sos:offering>
<sos:eventTime>
        <ogc:TM_During
                <ogc:PropertyName>urn:ogc:data:time:iso8601</ogc:PropertyName>
                <gml:TimePeriod xsi:type="gml:TimePeriodType"
                        <gml:beginPosition>
                                2000-01-01T00:00:00+0200
                        </gml:beginPosition>
                        <gml:endPosition>
                                2009-12-31T00:00:00+0200
                        </gml:endPosition>
                </gml:TimePeriod>
        </ogc:TM_During>
</sos:eventTime>
<sos:observedProperty>urn:ogc:def:phenomenon::nitrate</sos:observedProperty>
<sos:featureOfInterest>
<sos:ObjectID>urn:ogc:def:featureType:…:Upper_Rhine_Valley</sos:ObjectID>
</sos:featureOfInterest>
<sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
<sos:resultModel> gml:Observation</sos:resultModel>
</sos:GetObservation>
```

[9] Excerpt of the Web service request message encoded in the Extensible Markup Language XML. Name space declarations are omitted for the sake of simplicity.

4. The **Engineering** step in which the specified capabilities of the concrete service platform have to be implemented as service components and deployed in the context of a service network.

This thesis proposes a reference model for geospatial applications that underpins all these steps (section 5), and a design methodology (section 4) that relies upon the reference model but just focuses on the abstract design step. It is called the Design Methodology for Information Systems based upon Geospatial **Serv**ice-oriented Architectures and the Modelling of **Us**e Cases and Capabilities as Resources (**SERVUS**).

The following section presents a list of requirements for SERVUS.

1.4.3 Requirements for a Service-oriented Design Methodology

Bræk and Melby (2005) defined **human comprehension**, **analytical power** and **realism** as properties to be satisfied in order to reach the goals of model-driven development, in addition to the cross-cutting concern of platform independence. In the following a list of requirements for a model-driven service-oriented design methodology is presented that is derived from these properties.

1.4.3.1 Support of Human Comprehension

"Functionality should be represented in a way that enables human beings to fully understand it, to reason about it and to communicate precisely. To this end the concepts of the language must be well defined, match the problem domain and be easy to understand."

Human comprehension is translated into the following set of requirements:

R.1 **User Feedback**: The design methodology shall provide opportunities for the user to get feedback about those design artefacts that affect the externally visible behaviour and characteristics of the system. The user shall be able to easily judge whether his/her requirements are well understood or which improvements or adaptations must be applied. User feedback is widely considered as the key to successful development. Even more, it may be argued that users have the right to participate in developments that affect their lives (Avison and Fitzgerald, 2003).

R.2 **Usability**: The design methodology shall be easily understandable and usable by the users, who are, at least in the environmental domain, often scientists. Tan et al (2009) stress that "scientists have expertise in their specific domain (biology, physics, astronomy, etc) but understandably limited knowledge of IT technology and thus require easy-to-use tooling".

R.3 **User-near Analysis**: The design methodology shall provide methods and tools to better bridge the gap between the descriptive language of the user when expressing his/her requirements and the formal language used to specify the capabilities.

1.4.3.2 Support of Analytical Power

"It should be possible to reason about behaviours in order to compare systems, to validate interfaces and to verify properties. This requires a semantic foundation suitable for analysis."

Analytical power is translated into the following set of requirements:

R.4 **Architectural Framework**: The design methodology shall be embedded into an architectural framework for service-oriented architecture in order to be applicable to a class of systems and to allow the system designer to compare system properties.

R.5 **Property Coverage**: The design methodology shall provide means to handle functional, informational and qualitative properties of systems. This applies to both requirements and capabilities and explicitly includes the spatial-temporal properties as required for the application domain of environmental information systems (Visser, 2004).

R.6 **Standards Compliance**: The design methodology shall take existing SOA standards (e.g. interfaces, models) explicitly into account. In the application domain of Environmental Information Systems this requirement does explicitly encompass the set of geospatial standards provided by ISO/TC 211 "Geographic information/Geomatics" and the OGC.

R.7 **Semantic Enrichment**: The design methodology shall provide the means to semantically enrich the specification of requirements and capabilities based upon domain model such as agreed vocabularies of an application domain or ontologies. The technical report of the W3C Semantic Web working group (W3C-SWBPD, 2006) state that "domain models play a central role throughout the software development cycle, from requirements analysis to design, through implementation and beyond".

R.8 **Traceability**: The design methodology shall provide means for traceability in order to validate the coverage of the design. On the one hand, it shall become apparent how the requirements are mapped to capabilities, and on the other hand, it shall be possible to justify each capability by at least one corresponding requirement.

Note: The requirement of traceability is also stressed in the H/H/P design method for system design proposed by Hatley, Hruschka and Pirbhai (2000). H/H/P defines traceability as the possibility to derive all architecture and design components from given requirements, and to retrieve all requirements from which architecture and design components were derived.

R.9 **Service Factoring Guidance and Evaluation:** The design methodology shall provide guidance about how to factor the functional requirements into units of a reasonable size that are both re-usable and powerful. It shall provide means to evaluate the resulting service design.

1.4.3.3 Support of Realism

"The language should build on concepts that can be effectively and efficiently realized in the real world", because

(1) "it should be possible to derive efficient implementations automatically", and

(2) "functionality models may serve as valid documentation of the real system".

Realism is translated into the following set of requirements:

R.10 **Efficiency Gain**: The application of the design methodology shall improve the efficiency of the design process. One means to improve the efficiency is an integrated development environment that provides (at least partial) automation support for the individual design steps.

R.11 **Iteration and Documentation Support:** The design methodology shall support iteration steps in order to enable a spiral co-design and documentation of requirements and architectural artefacts (Pohl and Sikora, 2007). Berente and Lyytinen (2007) stress that the concept of iteration is fundamental to design activity and therefore inherent to each design methodology.

R.12 **Capability Discovery**: The design methodology shall provide the means to search for existing capabilities of a given service platform and take the results into account for the design. These capabilities may exist in a variety of forms and formats: as high-level textual specification in a document, as formal specification such as an application model specified in the Universal Modeling Language (UML), or as entries in a service catalogue or service registry.

The requirements R.1 – R.12 are used to assess the state of the art in the service-oriented analysis and design (section 3.4), to provide guidance for the conception of the SERVUS Design Methodology (section 4) and to finally evaluate it (section 9.2).

1.5 Structure of this Thesis

This thesis touches a large spectrum of application-oriented, methodological, technological and modelling aspects. The core of this thesis is the specification of the SERVUS analysis and design methodology for Environmental Information Systems (EIS) on the basis of open geospatial service platforms.

The structure of this thesis is illustrated in Figure 1-5.

Section 2 contains the foundations for this thesis.

- Firstly, it provides an outline of the application domain of environmental information systems and related application domains such as security. It presents research and project activities and initiatives that focus on the Information and Communication Technology (ICT) support for these application domains and associated technological requirements.
- Secondly, this section gives an overview of the principles of service-oriented computing. It starts with definitions of basic service-related terms and then fo-

cuses on aspects of service-oriented design and engineering. One particular point is the role of SOAs in systems that are composed of other systems leading to systems-of-systems.

- Thirdly, it presents the foundations in information systems engineering. Starting with the relationship of the work with the overarching design research framework of Hevner et al (2004) that is mainly applied in the Information Systems community, it then elaborates on methodologies for software development and model-driven architecture (MDA).

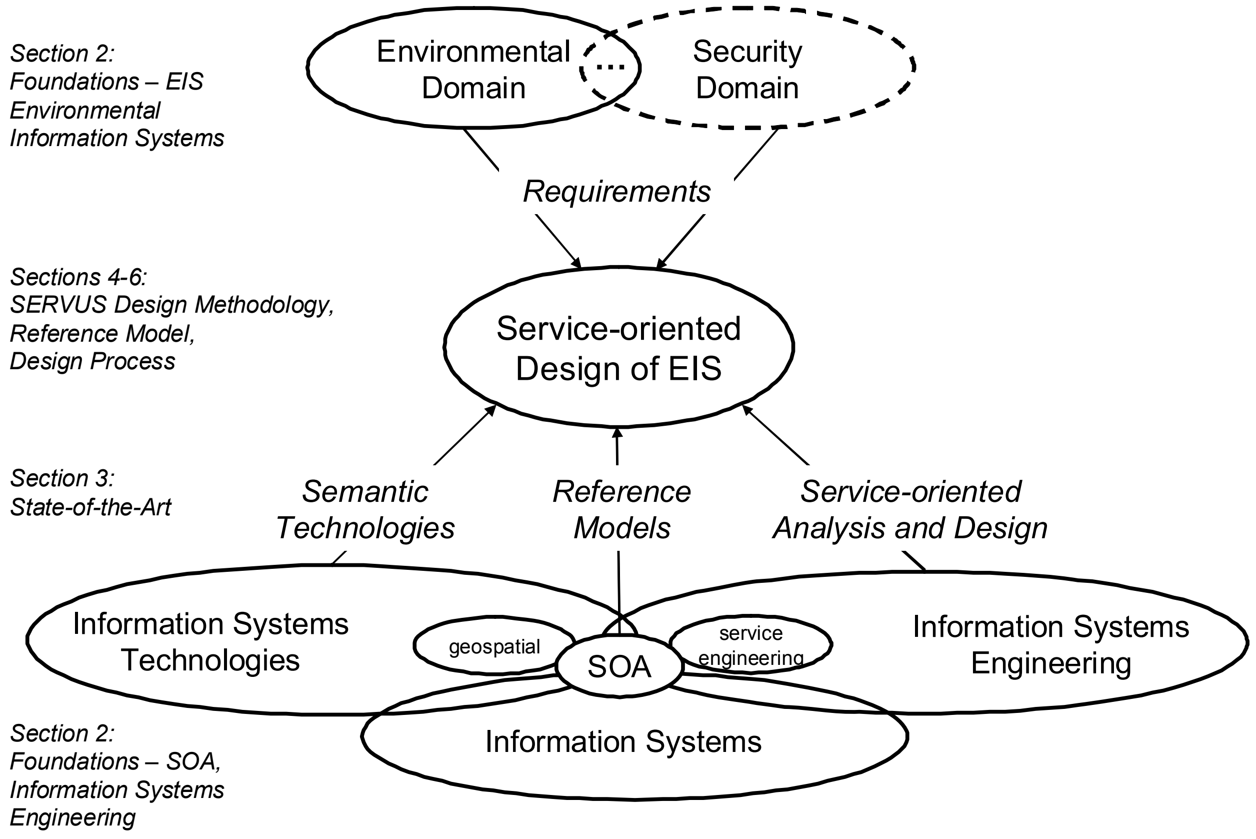

Section 2: Foundations – EIS Environmental Information Systems
Environmental Domain
...
Security Domain
Requirements
Sections 4-6: SERVUS Design Methodology, Reference Model, Design Process
Service-oriented Design of EIS
Section 3: State-of-the-Art
Semantic Technologies
Reference Models
Service-oriented Analysis and Design
Information Systems Technologies
geospatial
SOA
service engineering
Information Systems Engineering
Information Systems
Section 2: Foundations – SOA, Information Systems Engineering

Figure 1-5: Structure of Thesis and Domains Covered

Section 3 describes the **state of the art** of the design of information systems with a focus on service-orientation and EIS. Service design and engineering is an emerging research topic that has recently been taken up by the software engineering community as an "increasingly important approach to business application development" (Sommerville, 2007) and to the design of software architectures in general (Reussner and Hasselbring (eds.), 2009). It builds upon the foundations of information systems engineering, model-driven architecture and service-orientation, but also applies semantic technologies. Further facets such as reference models for geospatial SOAs have to be tackled when dealing with environmental information systems.

Thus, section 3 is structured into multiple state-of-the-art descriptions for three subtopics: reference models, semantic technologies and service-oriented analysis and design. Each sub-topic description finishes with an assessment that relates its relevance to the topic of this thesis and the list of requirements presented above in section 1.4.3.

The **conceptual core** of this thesis is contained in the sections 4 to 6:

- Section 4 describes the SERVUS Design Methodology in terms of its modelling approach and its design activities.
- Section 5 presents the Reference Model as the underlying architectural framework and conceptual model for SERVUS.
- Section 6 provides a specification of the SERVUS Design Process including activity diagrams for the individual design activities.

The **engineering part** of the thesis is contained in the section 7 and 8:

- Section 7 describes an example of an Implementation Architecture for SERVUS based upon components of geospatial service platforms that were developed in environmental risk management projects.
- Section 8 illustrates how SERVUS may be applied to a use case example of the marine domain.

Section 9 concludes with an **evaluation**, a summary of the major **results** and an **outlook**.

Section 1 contains the **references** used and applied in this thesis and some information about the author including a short curriculum vitae.

Section 1 contains abbreviations, a glossary of the major terms and a term **index** in order to enable a quick look-up and navigation through the document.

2 Foundations

2.1 Environmental Information Systems

The application domain of Environmental Information Systems (EIS) is inherently componentized due to the need for collaboration between the various public and private stakeholders involved and various environmental science disciplines (Schimak (ed.), 2005). This componentization gives the task of designing an EIS quite naturally the character of a system-of-systems engineering activity.

There is no commonly agreed definition of an EIS. In a very broad sense, Lam and Swayne (2001) defined an EIS to be the interface "between the advanced computer technologies and the requirements of environmental scientists and managers". As stated in the introduction EIS deal with objects that usually have a geospatial reference (geospatial information). Geospatial information is a ubiquitous element of almost all data as stated by Percivall (ed.) (2008): "Whether represented as a map or as an image, encoded as an address, zip code, or phone number, described in a text passage as a landmark or event, or any of the many other ways of representing Earth features and their properties; geography is pervasive". As illustrated in Figure 2-1 EIS are usually based on large databases that are indirectly (offline, e.g. by import/export mechanisms) or directly (online, e.g. by wireless sensor networks) coupled with in-situ or airborne environmental sensors of various types that deliver environmental information observed from environmental phenomena (e.g. water monitoring stations or cameras on aircrafts or satellites), or more concisely, observations that provide an "estimate of the value of some property of the feature of interest" (Cox (ed.), 2007).

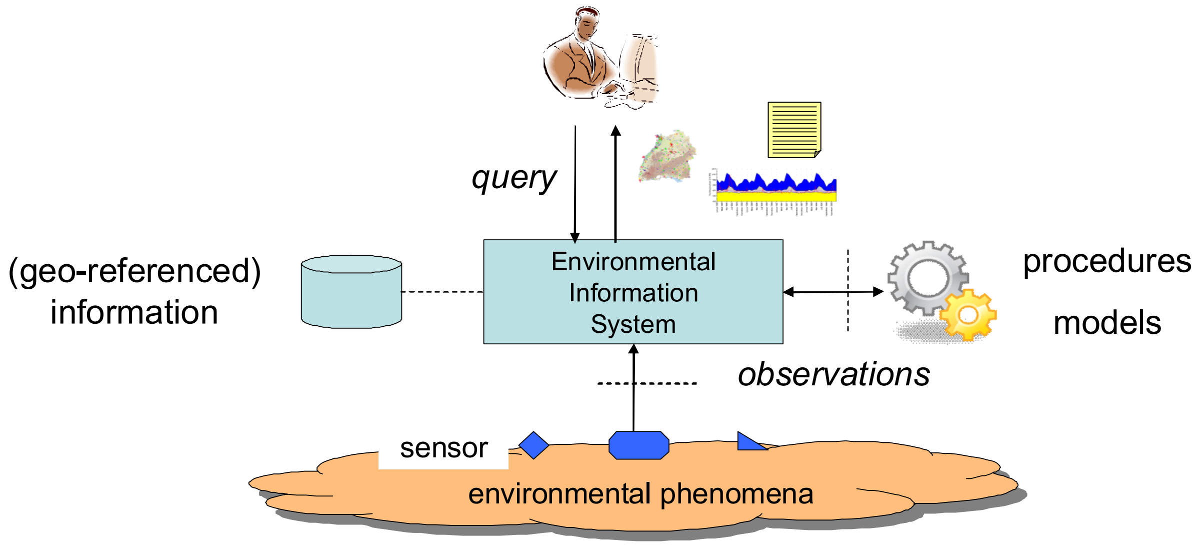


Figure 2-1: Environmental Information Systems

EIS allow the user to store, query and process environmental information and visualize it in thematic maps, diagrams and reports. More advanced functions cover advanced mapping functions of environmental data (Kanevski (ed.), 2008) such as geospatial predictions and simulations (e.g. the family of Kriging algorithms), spatial data analysis and mapping using machine learning algorithms (e.g. classification methods including neural networks or support vector machines) or geospatial visual analytics techniques by combining data mining and information visualization techniques (de Amicis et al (eds.), 2009). The additional integration of problem solving tools turns EIS into environmental information and decision support systems (Denzer, 2005).

2.1.1 Design Trends

In the last 10-15 years, the design of EIS has undergone fundamental changes following both the requirements of the users and the capabilities of the underlying ICT systems. Three major trends resulting from the demands of the stakeholders have determined EIS design in the last years (Usländer, 2008a):

- Domain Integration,
- Wider Distribution and
- Functional Enrichment.

2.1.1.1 Domain Integration

Domain Integration responded to the demand of enabling the correlation of EIS information and services across various thematic domains, mainly driven by the need to understand the complex inter-domain relationships in ecological systems. Traditionally, it has been essential for the success and acceptance of EIS that they are tailored to the thematic domain (e.g. air, water, soil), so that the end user easily recognizes in the user interface the notions and names of his/her respective thematic world and the information objects he/she has to deal with. Moreover, the EIS should adhere to the organizational working rules that the end users are used to. As end users prefer to focus on their thematic task instead of customizing generic tools, they expect systems that are tailored to their thematic knowledge and support them in the generation of thematic documents such as maps, diagrams or reports. An example of such a thematic user environment that enables personalization is the WaterFrame® system whose structure is illustrated in Figure 2-2 (Usländer et al, 2003). Thematic services offer functions such as data acquisition, data analysis and quality assurance, maintenance and selection of master and measurement data, generation of thematic documents, and decision support. These thematic services rely on a powerful set of generic services that support the management of thematic objects such as monitoring stations, wells, protection areas or measurement time-series. They may be configured according to personalized information views that are made persistent in so-called user-defined objects.

However, thematic information systems increasingly had to integrate aspects of other environmental domains in order to handle multi-domain correlation effects, e.g. to assess the impact of soil quality on groundwater quality. In order to enable a short learning curve, the integrated EIS should follow the same user interaction principles. From the ICT perspective, these demands initially (in the mid to late 1990s) resulted in the development of EIS programming frameworks based on mainstream but mostly platform-specific development environments of those days (e.g. Microsoft Active-X, Java/JavaBeans). Non-proprietary middleware technologies such as CORBA (Common Object Request Broker Architecture) specified by the OMG (Object Management Group) (Mowbray and Ruh, 1997) could overcome some of the platform-dependencies, but in the end have only been partially successful as standard distributed EIS middleware. One of the technical obstacles of CORBA was the tight coupling of client components to remotely accessible objects which led to problems in the robustness of the whole distributed application.

Nevertheless, CORBA laid the basis for thinking in terms of "distributed architectures" based on the notion of "interfaces" specified in a platform-neutral interface definition language. Since the middleware battle between Active-X/.NET and CORBA was not decided definitively in the ICT market, "Web services" emerged as the middleware of choice including for distributed Web-based EIS applications crossing thematic and organizational boundaries.

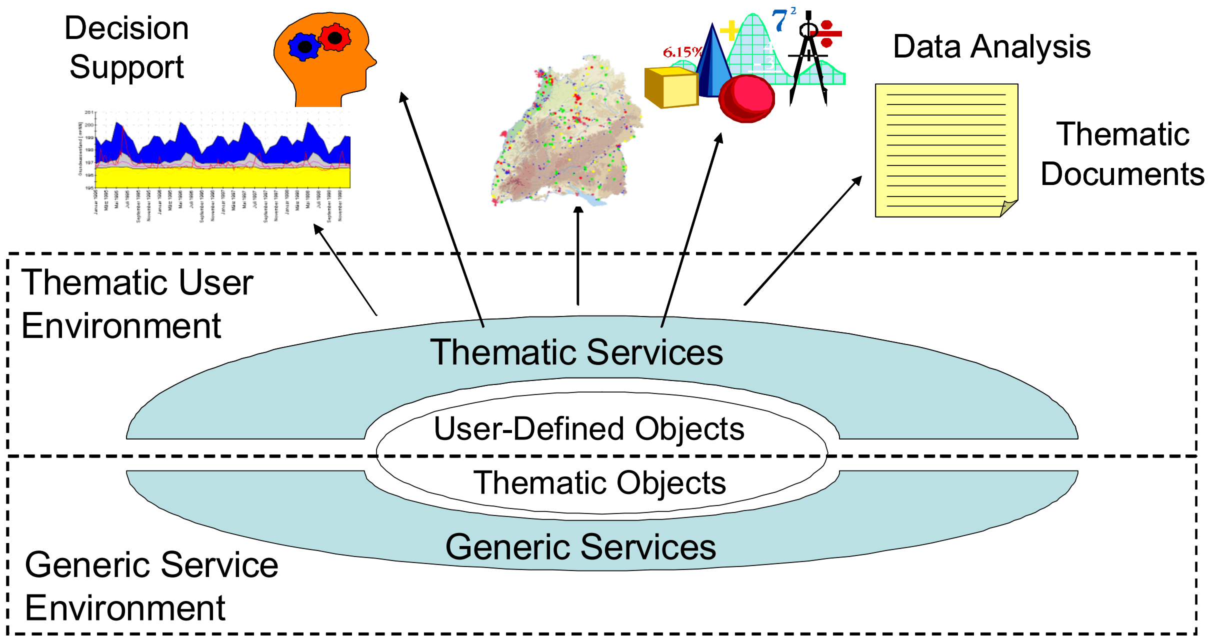


Figure 2-2: User and Service Environment of the WaterFrame® System

This middleware evolution was reflected in the design of integrated EIS covering multiple thematic domains in regional and national environmental authorities. One example is the framework conception of a comprehensive EIS operated in partnership by state and municipal administrations of the German federal state of Baden-Württemberg (Keitel, Mayer-Föll and Schultze, 2009). Driven by administrative reforms its first version appeared in 1998, already with the ambition to develop a set of reusable generic services that may be used for EIS in multiple thematic domains such as water, soil or waste management. The first implementations of this service-orientation were realized as Java components as part of rich-client thematic EIS applications. Nowadays, a smooth technical migration towards the support of Web services is taking place. Geospatial data of Baden-Württemberg is now offered to a multitude of users using multiple interfaces, tools and formats, including OGC-compliant

Web Map and Web Feature services. The latest research results in the context of the EIS Baden-Württemberg and beyond are available in Mayer-Föll et al (2009).

The European Commission follows this trend in their ambition to conceive a "Shared Environmental Information System (SEIS)" in Europe (EC, 2008). SEIS aims to be a decentralized but integrated Web-enabled information system based on a network of public information providers sharing environmental data and information. It will be built upon existing e-infrastructure, systems and services in member states and EU institutions. End-user institutions are expected to make investments to "render their existing systems interoperable and link them to an integrated system-of-systems" and integrate environmental knowledge in Europe (Hřebíček et al (eds.), 2009).

Its importance and urgency was stressed by a public resolution of the delegates of the conference "Towards eEnvironment" in Prague in March 2009 (eEnvironment, 2009).

2.1.1.2 Wider Distribution

The trend towards wider distribution responded to the demand to open up the EIS to a wider spectrum of users (from employees in environmental agencies to politicians in ministries and up to the citizen) as well as to design the functions of an EIS as units which may be invoked from other applications. This trend has been mainly driven by European and national directives such as the European Directive on Public Access to Environmental Information (EC 2003b) or the Water Framework Directive (WFD) (EC 2000) as further elaborated in section 2.1.3.1 below. At the technical level these directives resulted in the requirements to offer environmental information in a variety of formats and aggregation levels to a multitude of users.

The ICT answer to these demands was essentially influenced and pushed forward by the growing acceptance of the World Wide Web as a "computing platform" and not only as an information medium. Among others, the initial driving forces of "Web Services" were to provide an elegant platform-neutral solution in the middleware battle. However, the acceptance of SOA for the design of EIS resulted from the encouraging prospect of deploying an EIS as a set of re-usable components (services) with well-defined interfaces. These services should be callable from the Intranet/Internet, thus offering environmental information in an "open" manner to a growing and increasingly heterogeneous user community.

2.1.1.3 Functional Enrichment

The trend towards functional enrichment followed the demand to make more sophisticated functions, such as environmental simulations or geo-processing capabilities, directly available within an EIS (Usländer et al, 2003). This helps to lower the costs for purchase, development, user training and maintenance. Here, the SOA approach as described above helped greatly because these functions may also just be loosely-coupled with the EIS as a remote Web service instead of being provided in stand-alone systems, e.g. a Geographic Information System (GIS). Standardized geospatial

Web services (e.g. the Web Map Service) as specified by the OGC captured the generic parts of such remote functions.

The emergence of these ICT solutions created a more challenging vision which is an "ideal" ICT support that would make information available on demand for the end users and would allow the service providers to offer high-quality services at considerably lower cost in a plug-and-play manner. EIS applications of various types running in control or information centres, tailored to the analysis of environmental data or to its visualization in maps or diagrams, have to be coupled with data of various types such as geospatial thematic data, documents or information about environmental phenomena observed by monitoring stations, cameras or satellites – a vision formulated by Denzer (2005) of a functionally rich but generic platform as a need to effectively build environmental information and decision support systems.

2.1.2 EIS Business Processes

Process Models are used to represent the business processes of an organization. Basically, they represent "relationships between inputs and outputs, where the inputs are transferred into outputs using a series of work activities that add value to the inputs" (Johannesson, 2007). In general, business processes are motivated and driven by goals. The Business Motivation Model of the Object Management Group (OMG, 2008a) defines a **goal** as "a statement about a state or condition of the enterprise to be brought about or sustained through appropriate means. A goal amplifies a vision — that is, it indicates what must be satisfied on a continuing basis to effectively attain the vision". As business processes are directly related to the strategy of an enterprise, they are specific to the organization and the business rules of an enterprise. Furthermore, goals should be narrow, such that may be quantified by objectives. An **objective** is a "statement of an attainable, time-targeted, and measurable target that the enterprise seeks to meet in order to achieve its goals" (OMG 2008a).

Transferred to the domain of environmental security a goal of an environmental agency may be to assure good air quality in a city. Such a goal needs to be transferred into time-targeted and spatially targeted environmental objectives such as "reduce nitrogen oxide (NO_x) and particulate matter levels by 10% after two years in the city centre of Athens". First, this objective requires acquiring information about the status of the environment (e.g. the values of an air quality parameter such as NO_x or particulate matter) in a defined spatial and temporal resolution (e.g. every 2 hours for the city centre) associated with an uncertainty boundary. This information may be directly retrieved from environmental sensors deployed in the city, however, very often, due to budget limitations or malfunctions of sensors or the underlying communication infrastructure, some information processing (e.g. the execution of geostatistical interpolation techniques) is required to assure a reliable provision of this information to the decision maker. Depending on the observation results, the decision maker may decide to increase the resolution of the observation values (e.g. by the deployment of additional sensors or by increasing the sampling rate), or he may take a decision (e.g.

to reduce the emissions by traffic regulations). Note that the enforcement of the decision very often requires means that are outside of the EIS system boundary.

EIS have a limited variety of such business processes. This limitation has an impact on the design of EIS. Tan et al (2009) identified four stages of scientific workflows: discovery of relevant data and analytical services, service composition into workflows, workflow execution and result analysis. Based upon experiences in the development of EIS in Germany (Mayer-Föll et al, 2009), and especially water information systems (Usländer, 2005), we propose a more refined template for EIS business processes consisting of standard activities and some optional flows and sequences between the activities (Figure 2-3).

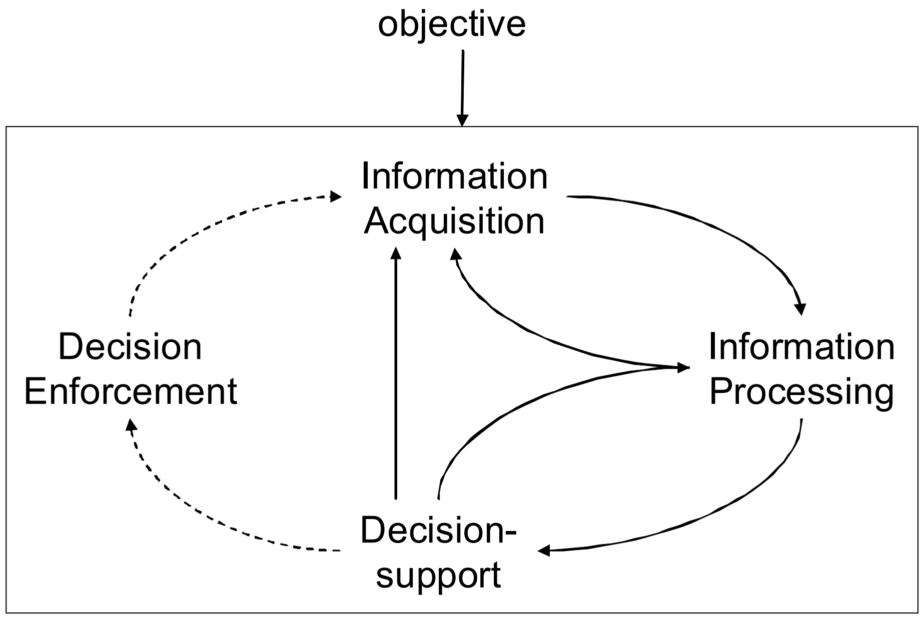


Figure 2-3: Typical EIS Business Process

The **Information Acquisition** activity is a core activity as it is responsible for the discovery and timely provision of environmental information and services as output. The input may originally come from sensors of various types, e.g. in-situ environmental monitoring stations or mobile airborne or space-borne cameras. It deals with the communication means to the sensors and the management of the underlying communication infrastructure. Note that this thesis adopts the abstract meaning of the term "sensor" as defined by the Sensor Web Enablement architecture of the Open Geospatial Consortium (Simonis (ed.), 2008). Sensors need not necessarily to be physical devices but may also comprise algorithms or environmental models. Furthermore, sensors are very often not directly coupled to the EIS. Instead, sensor observations may be collected in a separate system which may be an environmental monitoring network consisting of online measurement stations, or even a human carrying a chemical probe (e.g. a water probe from a river) to a laboratory. The measurement results are finally stored in a database or document which then acts as the source for sensor observations for the EIS.

In the acquisition of environmental information, the specifics of geospatial information and geospatial relationships have to be explicitly considered, e.g. when querying potential data sources for environmental information. Hart and Dolbear (2006) point out that 80% of all information have a direct or indirect geographic reference. The geographic reference may be a location on the Earth with precise geographic coordinates (e.g., latitude and longitude values). However, it may also be an address which often cannot be unambiguously resolved to a precise geographic location. In most cases this geographic reference is not dominant, but it acts more as support for the achievement of business objectives, especially in combination with others, possibly non-spatial attributes. For instance, an insurance company may only be interested in the geographic position of buildings because of the need to calculate the vulnerability with respect to flood or forest fire. Geographic relationships have their own semantics such as located-in, located-outside-of, overlaps-with or distant-from. They are often vague and therefore uncertain, e.g. "The hotel is located nearby the beach." or "The airport is located near London."

In nearly all cases, some **Information Processing** is necessary before the information may be used for decision support. This activity is based upon the output of the information acquisition activity and may range from a simple harmonization of measurement units up to the execution of complex geo-statistical or information fusion algorithms. Sometimes, it is necessary to go back to the information acquisition activity, e.g. if the level of uncertainty of the information or the number of missing values is too high. Furthermore, in order to assess and process geospatial information it often has to be transformed into the necessary form or format. As an example, refer to the location of an incident, e.g. "The accident occurred on the road behind the village." Depending on the application, the name of the place, its exact geographic coordinates as a point or polygon or its membership to a relevant region may be required. This information is usually kept in a dictionary of place names, often called a gazetteer, and made available by means of a gazetteer lookup function (Hill, 2006).

The output of the information processing activity is presented to the **Decision Support** activity which analyses and assesses the situation against the initial objective, and finally generates decisions as output. Various types of decisions are possible: either to go back to the information acquisition activity with adapted acquisition parameters, to start a further information processing activity or to enforce a decision. **Decision Enforcement** may take place through an actuator that is embedded into or connected to the system, or, very often, it may mean an action that is outside of the scope and system boundary of the EIS application.

The majority of the business processes in EIS follow this process pattern. For each activity, there are associated information objects. Some of them are specific to the activity, e.g. a sensor data retrieval object that is responsible for the acquisition of sensor observation from external monitoring stations, others are important for all activities. The former are called activity supporting objects, whereas the latter are typically abstractions from real world entities (e.g. an information object that stores information about a well) or human-generated entities (e.g. a water protection zone).

They are called **features** in geospatial literature and standards (Percivall (ed.), 2008). Features play a very important role in the design of EIS as they represent entities in the universe of discourse of the users and stakeholders.

2.1.3 Related Application Domains

2.1.3.1 Thematic and Business Domains

The environmental domain overlaps with other thematic and application domains. Figure 2-4 positions the types of IT systems (information systems and control systems) against selected thematic domains, here production control, economics, environmental security (discussed separately in section 2.1.3.2) and health.

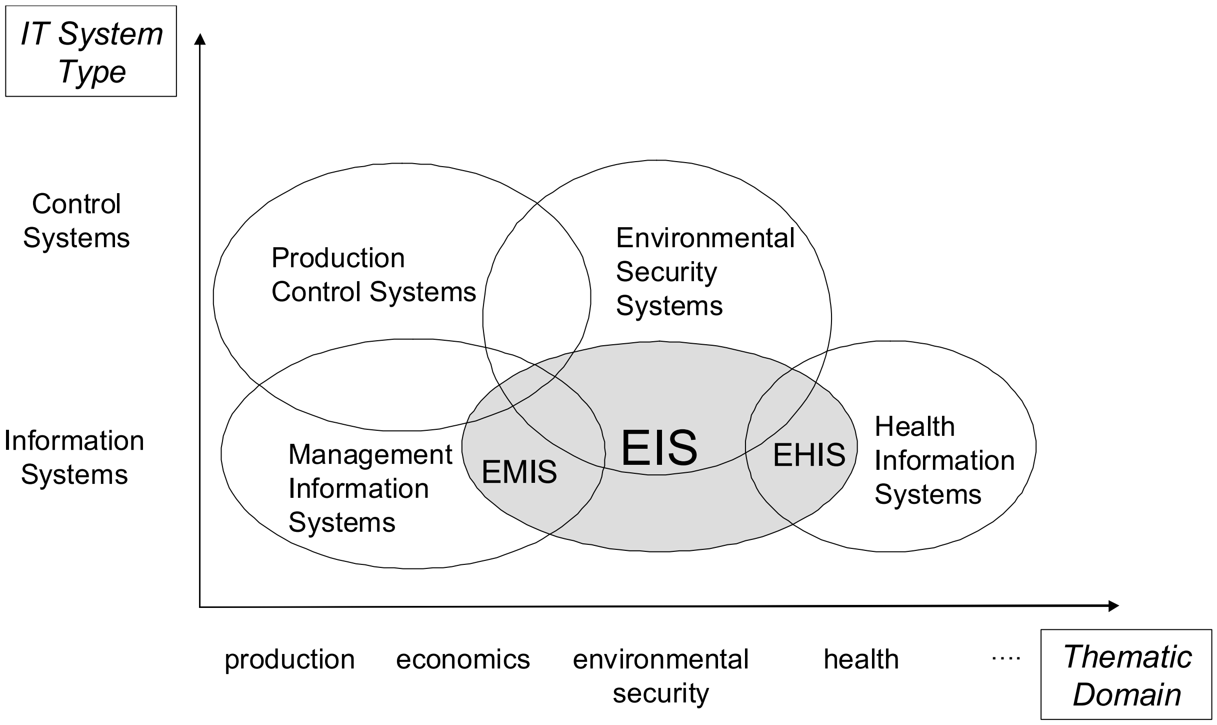


Figure 2-4: EIS and related Information and Control Systems in other Thematic Domains

The relationship of EIS to Management Information Systems in companies which are dedicated to economical, production control and recycling aspects is discussed by Hilty and Rautenstrauch (1997). They call IT system types that deal with the overlapping aspects Environmental Management Information Systems (EMIS).

A further selected example is the connectivity between environment and health information systems (EHIS) as an enabler to study the impact of the environmental situation to human health effects, one of the key goals for the European Union to achieve environmentally sustainable growth (EC, 2003a). This involves, among other things, "reducing the disease burden caused by environmental factors in the EU and preventing new health threats caused by the environment". In the final report about the Connectivity of EHIS (Skouloudis (ed.), 2009) two key scientific challenges were identified on the thematic level:

1) The measurement and/or modelling of individual exposure and its effects on health, and
2) The identification of the cumulative effects of exposure, especially to mixtures of chemicals and other stressors.

However, in order to enable research on this thematic level, the report concludes: "It is also essential to look continuously on the status of integration of heterogeneous architectures and promote the development of software modules which will enhance such integration and resolve the gaps in temporal and spatial dimension of existing data structures."

2.1.3.2 Environmental Security

In the military domain, environmental issues are discussed under the umbrella term "environmental security". The following five key elements need to be addressed in environmental security (Landholm (ed.), 1998):

- Public safety from environmental dangers caused by natural or human processes due to ignorance, accident, mismanagement, or design.
- Amelioration of natural resource scarcity.
- Maintenance of a healthy environment.
- Amelioration of environmental degradation.
- Prevention of social disorder and conflict (promotion of social stability).

From the technological perspective, one of the key issues is the integration of information that is related to environmental security, as stated by a dedicated NATO advanced research workshop on this topic (NATO, 2008): "In case of threats to environmental security, there is a need for temporal access to complete updated, reliable information, in a dedicated form, which is an essential prerequisite to effectively counter security threats".

On the one hand, environmental information from various sources, whether in-situ, airborne or space borne sensors or environmental data bases, is required to protect humans, natural resources or human-made artefacts from environmental hazards, either being subtle effects or sudden environmental events such as earthquakes, storms or forest fires (Douglas et al, 2008).

On the other hand, information resulting from the continuous monitoring of natural resources is a key factor in assessing the short-term or long-term impact of human activities upon the environment. Public safety from environmental dangers has to be considered "within and across national borders" (Landholm (ed.), 1998). Thus, environmental security applications must enable an efficient and flexible exchange of information as well as the remote call and eventually the reuse of their embedded functional components across system boundaries.

We consider the analysis and management of environmental security as a particular application of EIS (Usländer, 2009c). Environmental Security Systems typically contain EIS as kernel components for the gathering, processing and rendering of environmental information but also encompass actions upon the environment. These are either performed through actuators triggered by the EIS or by means of human activities outside of the EIS. In this case, Environmental Security Systems turn from information systems into control systems.

2.1.3.3 Water Resources Management

In Europe, the major driving force for the development and evolution of water information systems is the European Water Framework Directive (WFD) (EC, 2000). The Directive covers all water categories: rivers, lakes, groundwater, as well as coastal and transitional waters. Its primary goal is the improvement of water quality across Europe. This will be achieved by a combined approach of emission limits and quality standards. In the case of trans-boundary water bodies, co-operation between countries is needed (Timmerman and Langaas (eds.), 2004). Another important goal is the sustainable use of water resources throughout Europe. To ensure the achievement of these ambitious objectives and the consistent implementation of the directive in all Member States and across borders, implementation is carried out cyclically in a multi-step process: assessment of the qualitative status, initiation and adjustment of monitoring programmes and set-up of river basin management plans. The overarching requirement of the WFD is to achieve "good ecological and good chemical status" by 2015, unless there are reasons for derogation.

The WFD is not only a fundamental rethink of the EU water policy - its implementation is also a challenge for the supporting ICT and, especially, for a WFD-specific information management (Usländer, 2005). There is a huge need for harmonization and possibly standardization to achieve an efficient implementation of the WFD within Europe. Therefore, the European Commission has set up a WFD Common Implementation Strategy. In this context, a series of mostly thematic working groups and joint activities has been launched to support the development and testing of coherent WFD methodologies. From the ICT point of view, the working group Geographical Information System (GIS) has been the most relevant one.

However, the WFD implementation goes far beyond the implementation of just the geographic elements of the WFD. Regarding the ICT infrastructure, the main proposal of the GIS working group is a stepwise WFD implementation approach by means of three different European WFD "products", mainly defined in terms of the GIS aspects (Vogt (ed.), 2002):

1. Seamless and harmonized geometric data.
2. Centralized WFD database with a data exchange based on agreed formats to fulfil the WFD reporting obligations.
3. Federation of spatial WFD data servers based on OGC standards and aligned with other pan-European activities on spatial data integration, such as INSPIRE (EC, 2007).

For the implementation of the federated servers (Figure 2-5) there is a need for a generic ICT framework architecture that integrates multiple views within a single concept (Usländer, 2005):

1. an organizational view that considers the information flow across regional, national and organizational boundaries,
2. a process view that considers the life cycle of the information involved,
3. an informational view that integrates both geospatial data, tabular and textual data, thematic documents and meta-data, and
4. a functional view that considers what generic and specific functions as part of a standard service infrastructure are required on which level as well as their signatures and access methods across networks.

The realization of the informational and the functional view in the individual EU member states and organization units especially requires an examination of the information models and service components already specified, implemented and possibly deployed.

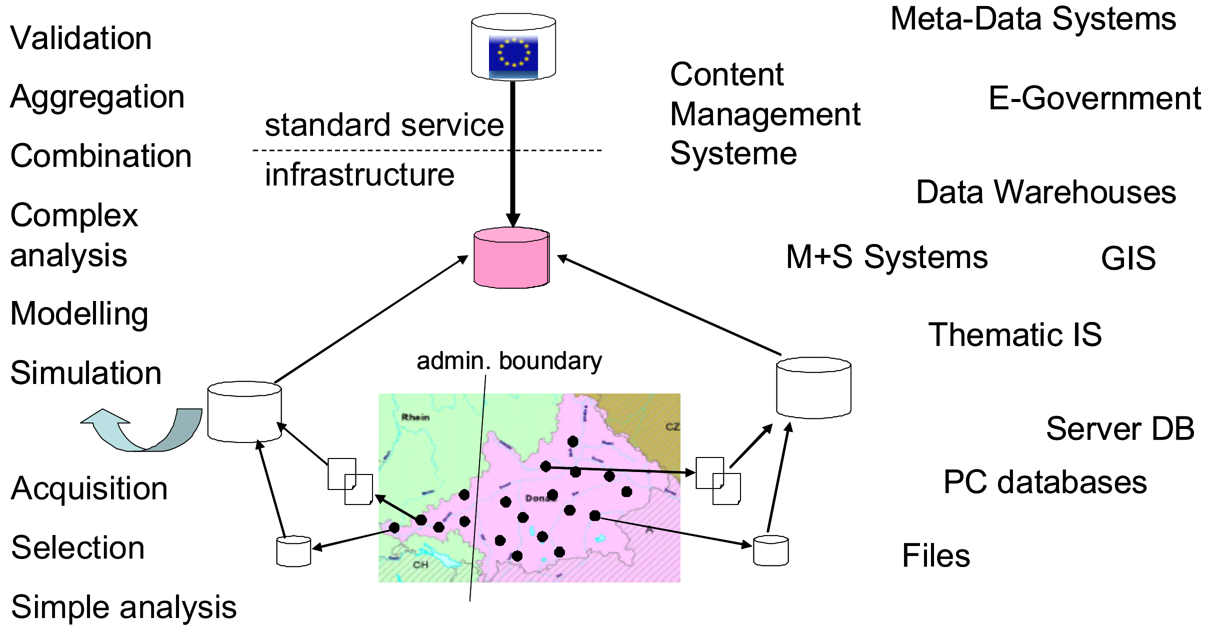


Figure 2-5: Reporting Service Infrastructure for the WFD Implementation

At the member states level, various WFD reporting schemes were set-up in order to fulfil the reporting obligations. In Germany, for example, the upload of state-level WFD data is performed through an Internet portal called WasserBLIcK (Busskamp, Fretter and Usländer, 2004) which assembles and validates the data before they are submitted as a federal data set standard electronic format to the Water Information System for Europe (WISE) (Biliouris and van Orshoven, 2009). WISE provides a centrally-based repository for the submission, validation and dissemination of data reported under the WFD, and then enables the Web-based query, view and analysis and mapping of the datasets.

Although not driven by legislative directives but more from a research and thematic point of view, there are similar efforts in other regions of the world. In the United States, the Hydrologic Information System group of the Consortium of Universities for the Advancement of Hydrologic Science, Inc., attempts to create Web services for exposing remote hydrologic time series databases (Goodall et al, 2008). Their motivation is that "much of the hydrologic data remain underutilized in the hydrologic sciences due in part to the time required to access, obtain, and integrate data from different sources". They argue that in the US National Water Information System (NWIS) Web services may offer a "means for sharing hydrologic data more openly by providing a standard protocol for machine-to-machine communication". These services provide a "middle-layer of abstraction between the NWIS database and hydrologic analysis systems, allowing such analysis systems to proxy the NWIS server for on-demand data access". However, the ICT approach currently underpinning the offering of the NWIS hydrologic data is not yet based upon the corresponding architecture, services and information models of the OGC Sensor Web Enablement (Simonis (ed.), 2008).

In Australia, recognizing that "water is becoming a scarce and vigorously contested resource" and that "efficient water resource management and informed decision making is currently hampered by a lack of accurate and timely water information" various efforts are bundled to design and implement a national Water Resources Operation Network Australia (WRON) (CSIRO, 2007). WRON aims at providing the technical framework and standards required to support water information management ranging from sensor networks up to information tools such as an Australian water forecasting system. An essential part of this ambition is the specification of an Australian Water Resources Information System (AWRIS) developed by the WRON Alliance (WRON, 2006).

The basic architecture of AWRIS is based upon SOA principles and structured around an "enabling framework" using distributed, standards-based interoperable Web services. This architecture will allow for a high degree of scalability so that, in the future, it can provide access to new data sources such as an access to 'live' sensor networks that measure water level or quality in real time. The WRON and AWRIS architects are working closely together with the OGC, e.g. in the newly formed OGC Hydrology domain working group (OGC-Press, 2009) that aims to develop and promote the "way in which water information is described and shared".

2.1.3.4 Environmental Risk and Emergency Management

The increasing intensity and frequency of natural disasters in the past few years have led to a heightened awareness of safety issues relating to environmental risk and emergency management both at the political and at the public level. Related research activities on the ICT support for environmental risk and emergency management were carried out (Fabbri and Weets, 2005). Following the terminology used by the Committee on Planning for Catastrophe of the US National Research Council (2007) the term **emergency** is understood here to mean a sudden, unpredictable event that poses a substantial threat to life or property. Furthermore, "**emergency management** is the organization and management of resources and responsibilities for dealing with all aspects of emergencies". There are various approaches to assigning emergency management activities to distinct phases in an emergency management cycle, e.g., Alexander (2002). The US National Research Council (2007) proposes a simplified model covering four phases (Figure 2-6):

- **Preparedness** activities are undertaken in the short term before disaster strikes in order to enhance the readiness of organizations and communities to respond effectively.
- **Response** activities are undertaken immediately following a disaster to provide emergency assistance to victims.
- **Recovery** activities are undertaken after a disaster to return the people and property in an affected community to at least their pre-disaster condition of well-being. Examples of short-term activities include the provision of temporary housing or the initial restoration of services and infrastructure repair. Examples of long-term activities involve rebuilding and reconstruction of the physical, economic and social infrastructure.
- **Mitigation** activities are undertaken in the long term after one disaster and before another disaster strikes. They aim at reducing the impacts of future disasters, e.g. by identifying and modifying hazards or by assessing and reducing vulnerabilities.

Each of these phases levies particular demands on emergency managers and responders. They follow one another in a continuous cycle, with a disaster event occurring between the preparedness and the response phases. The occurrence of a disaster event and its impact in space and time is communicated in an alarm event. If the disaster event may be forecasted, there may be a preceding warning event (Alexander, 2002).

The activities in each phase can be improved by the application of geospatial data and tools. For instance, early warning systems may issue a warning event before the occurrence of a disaster event by means of continuous monitoring of environmental hazards and conditions and the assessment of associated risks. This enables pre-impact activities to take place, such as the immediate multi-channel warning or even evacuation of coastal areas threatened by a tsunami-producing earthquake (DEWS, 2009).

When analysing requirements and concepts for effective ICT support for these phases, the US National Research Council (2007) identified a common need across all phases. Every aspect of emergency management requires geospatial knowledge such as the location of the disaster event itself or the magnitude of its impact on every part of the surrounding area, and also requires the capability to access and to process geospatial information, e.g. to produce damage maps or to visualize smoke plumes: If practical, these capabilities should extend from the local government up to county, state, and federal government. The response phase, however, additionally also requires speed, e.g. access to up-to-the-minute data, rapid generation of key products and rapid delivery of them to the personnel who need the products most. A broad overview of requirements, geospatial ICT architectures and solutions for disaster management is given by van Oosterom et al (eds.) (2005).

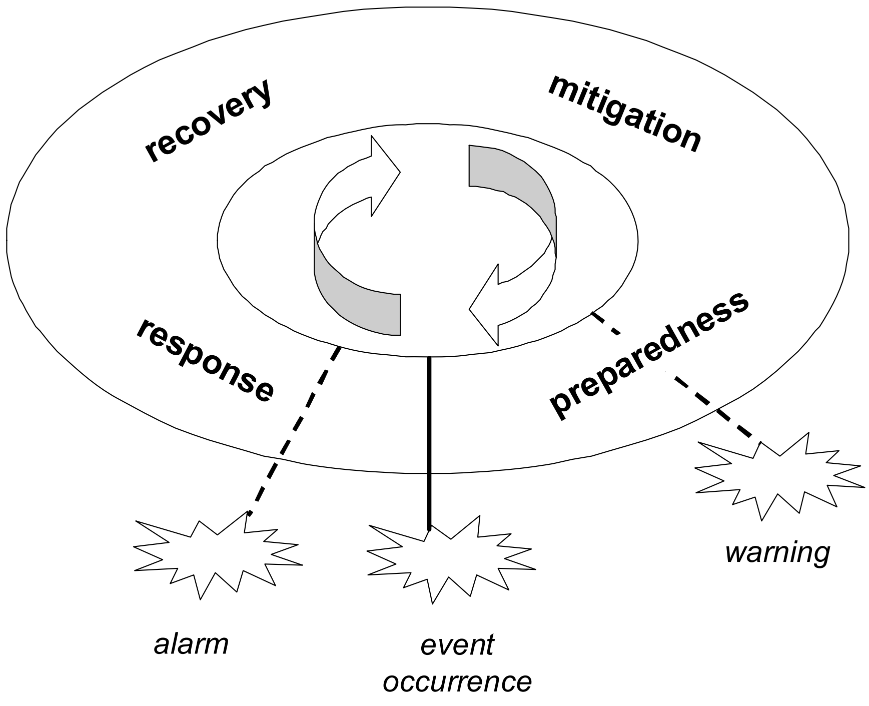


Figure 2-6: Phases of Emergency Management according to the National Research Council (2007), complemented by a Warning and an Alarm event (Alexander, 2002)

In Europe, ICT architectures for the response phase were developed, for instance, by the German security research project SoKNOS (Service-Oriented Architectures Supporting Networks of Public Security). SoKNOS aimed to develop concepts that are valuable in the support of governmental agencies, private companies, and other organizations that are active in the handling of disastrous events in the public security sector (Manske et al, 2009).

The focus of Environmental Risk Management is on the other phases in which the management of emergency risks is in the spotlight. According to the glossary of the Center for Disaster Management and Risk Reduction Technology (CEDIM) (Thieken (ed.), 2005), the term **risk** denotes "the probability and the amount of harmful consequences or expected losses resulting from interactions between natural or human induced hazards and vulnerable conditions". **Risk management** is understood as the set of preventative, integrated actions taken to deal with risk identification, analysis, and measures that are required during emergencies. It thus encompasses all the activities related to the identification and management of hazards, the analysis of vulnerabilities as well as the assessment and analysis of risks in a spatial-temporal domain. For instance, preparedness actions aim at shortening the time required for the subsequent response phase and potentially speed recovery as well.

An open ICT architecture for **environmental risk management** focuses on the preparedness, recovery, and mitigation phase. Within each of these phases similar methods and tools are used by the stakeholders. Some of them may also be relevant as background information and services in the response phase, but then their access must be assured by a dependable infrastructure. Some of the methods and tools are risk-neutral while others are specialized by risk domain (e.g. fire, flood, seismic, coastal zone and technological). Moreover, results from earlier phases are often re-used in later phases i.e. results from consequence/simulation models can be reused in a recovery phase or even during emergency response. However, the ability to share all relevant data, especially when considering emergencies which cross administrative or international borders, is often very limited because risk management tasks are mainly handled by public institutions on a variety of administrative levels, each with their own IT systems for the provision of data and services. For instance, a typical question that is often posed is "what are the risks that exist in my territory". The response is dependent on the phase of the emergency management cycle and on who posed the question.

The main problem today is that in any given activity in any given phase of the emergency management cycle, decision makers and stakeholders do not have easy access to the information that they need in order to fulfil their goals. Currently, there is no single integrated system architecture that can fulfil this request, and information produced in each phase is often incompatible.

The vision of an open ICT architecture for environmental risk management information systems has driven the work of the European Integrated Project ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) (Annoni et al, 2005). Environmental risk management is a broad application domain that copes with multiple types of risks such as forest fires, floods, earthquakes, geo-hazards or storms as well as multi-risk situations. An example of what needs to be investigated in multi-risk scenarios is the question of how the occurrences of forest fires in a mountain area influences the risk of flash floods after heavy rainfalls. The ORCHESTRA project was carried out between 2003 and February 2008 with the objective of contributing to a future "ideal" ICT infrastructure support for environmental risk and emergency management (Annoni et al, 2005). This required a generic approach. The final goal was to plug-and-play with environmental risk management resources and to provide the end users with cross-border services for risk and disaster management which they lack today. An essential element of such an ideal ICT support was an "open geospatial service platform" (Figure 2-7) which provides seamless access to resources (information, services and applications) across organizational, technical, cultural and political borders.

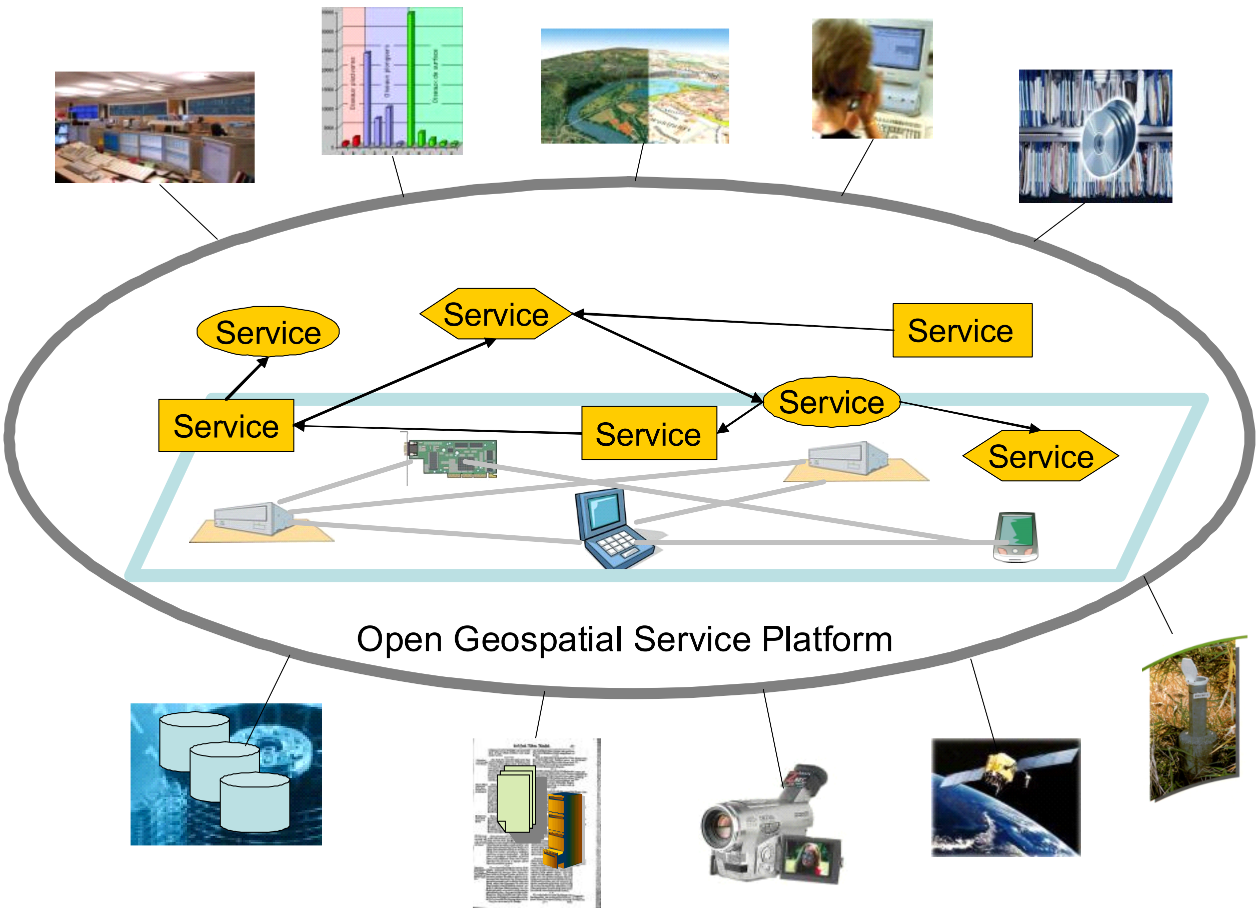


Figure 2-7: Open Geospatial Service Platform

2.1.4 Requirements for Design Support

There is a high demand for methodological support for the design of geospatial applications based upon geospatial SOA and related international standards. The reasons are as follows:

- Geospatial applications are highly characterized by their interdisciplinary nature. Thus, they require good cooperation between experts of different application domains (biology, geology, hydrology, traffic, ecology, etc.) in addition to cooperation with the computer scientists who are responsible for the SOA design (Tan et al, 2009).
- Geospatial applications very often require the integration and coordinated use of databases from different sources. Examples are emergency management (National Research Council, 2007; Manske et al, 2009; van Oosterom et al (eds.), 2005), environmental risk management (Klopfer and Kannellopoulos (eds.), 2008), biodiversity research (Giddy et al, 2009) or the connectivity of environment and health information systems (Skouloudis (ed.), 2009). In the thematic domain of biodiversity informatics "there is the dream of a unified infrastructure where data and analysis services all around the world are seamlessly accessible and integrated" (de la Torre et al (eds.), 2007).

- Two trends should be considered in parallel when designing the service infrastructure of EIS: On the one hand, geospatial applications have reached the mass market which is demonstrated by the recent success of navigation, cartographic and earth visualization systems such as GoogleEarth®, Microsoft VirtualEarth® or OpenStreetMap (Ramm and Topf, 2009). These systems tend to become components (often in terms of Web-based geospatial services) in mainstream ICT systems. On the other hand, existing geospatial SOA standards of ISO and OGC produce specifications on different abstraction levels (both platform-neutral and platform-specific) that play an increasingly important role as side conditions in the design of geospatial applications.
- There is a high potential for return on investment in the use of geospatial standards. A study carried out on behalf of NASA's Geospatial Interoperability Office (Booz Allen Hamilton, 2005) compared two e-Government projects, one utilising open geospatial standards to a high degree, and the other implementing few or none of these standards. The study provided important evidence that the adoption of standards can improve information sharing, foster improved decision-making, build business resilience and lower maintenance and operations costs over time. Klopfer and Kannellopoulos (eds.) (2008) reported that the project based upon open standards had a 119% return on investment over the program that did not implement standards. Though the initial costs, e.g. for system planning, development and implementation for the project utilizing a high degree of open standards were higher, the total costs dropped in the third year, reflecting lower costs for maintenance and operations.
- The geospatial market is driven by political decisions concerning the development of spatial data infrastructures (SDI) on regional, national and international levels. Geospatial e-Government applications have to make use of and be integrated into such SDIs which are usually based upon international standards (Bernard et al (eds.), 2005b). In Europe, this trend is pushed by the INSPIRE directive (EC, 2007) which must be implemented by all member states of the European Union. INSPIRE is a European directive establishing the legal framework for setting up and operating an **In**frastructure for **Sp**atial **I**nfo**r**mation in the **E**uropean Community. The Directive focuses on spatial data that are held by or on behalf of public authorities. INSPIRE targets environmental policies, however, other sectors such as agriculture, transport and energy may benefit, too, once this infrastructure is in place. The proposal of the INSPIRE directive lays down general rules for the various components of a framework for a European SDI based on existing standards and specifications if adequate and possible:
    - rules for metadata to support the discovery and evaluation of spatial data and services;
    - rules to achieve interoperability that allows integration of spatial data of the various themes addressed by INSPIRE;

- rules for interoperable network services for discovery, viewing, accessing and downloading spatial information;
- rules for data sharing; necessary coordinating structures; and
- rules for the development of a European geo-portal to provide a common entry to access all INSPIRE network services (Bernard et al, 2005a).

There is a trend towards an open market for geospatial services with a high potential for growth. Craglia et al. (2003) and Dufourmont (ed.) (2004) studied the impact and the market potential of the implementation of INSPIRE in national, regional and local authorities and organizations, including the European Commission. It turned that annual investments in the range of 93-138 M€ are expected over a 10 year period. However, the estimated annual benefits range from 770 M€ to 1.150 M€, i.e., there will be a cost-to-benefit ratio of 1:8.

2.2 Service-Oriented Computing and Architecture

2.2.1 Definitions

Before service-specific definitions are provided the basic understanding of the terms "system" and "architecture" has to be clarified.

<u>**Definition (2.1):**</u> A **system** is 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** (ISO/IEC 10746-2:1996).

This thesis applies this quite generic definition to information systems, especially EIS. An EIS is "of interest" to a user in the sense that it is the interface to retrieve, process and visualize environmental information. An EIS may comprise components which themselves are EIS. Such as structure leads to a system of EIS (section 2.2.4).

<u>**Definition (2.2):**</u> An **architecture** (of a system) is a set of rules to define the structure of a system and the interrelationships between its parts (ISO/IEC 10746-2:1996).

As the subject of this thesis is system design, we rely on a more refined architecture definition that takes design principles into account.

<u>**Definition (2.3):**</u> An **architecture** is the fundamental organization of a system embodied in its **components**, their relationship to each other and the environment, and the principles guiding its design and evolution (IEEE 1471-2000).

Service-orientation is a paradigm that utilizes "services" as fundamental elements of distributed application development (Papazoglou and Georgakopoulos, 2004), or, in other words, it is a "way of thinking in terms of services and service-based development and the outcomes of services" (OpenGroup, 2008). Accordingly, Service-oriented architecture (SOA) may be defined as follows.

__**Definition (2.4):**__ A **service-oriented architecture** (SOA) is an "information technology (IT) architectural approach that supports the creation of business processes from functional units defined as services" (Zhang et al, 2008).

For such an SOA definition the most predominant architectural element is the "service". As the English word "service" is overloaded in its meaning (Preist, 2004), it has to be interpreted in the context of the level of abstraction (e.g. business level or technical level) or the related architectural framework in which it is used. For the purpose of this thesis, we adopt the service-related definitions in the ISO series of standards on geographic information. Other definitions are discussed in the context of service-oriented reference models in section 3.2.

__**Definition (2.5):**__ A **service** is a "distinct part of the functionality that is provided by an entity through interfaces", whereby an "**interface** is a named set of operations that characterize the behaviour of an entity" (ISO 19119:2005).

However, a definition of an SOA that is just based on the notion of "services" and "interfaces" falls too short in the design of EIS for the following reasons:

- It is too much oriented at a functional view upon a system and neglects an informational view. Apart from performing business functions, services are used to mediate the access to information objects, e.g. time series of observed environmental phenomena. Very often the focus in EIS use cases is more on the informational aspects than on the functional aspects.
- Although quite concise on an abstract level, the meaning of the term "service" gets quite fuzzy when discussed on an implementation level as there are many ways to implement services, from message-oriented or object-oriented middleware up to Web services, i.e. computational facilities that are accessible over the Internet via an interface using Web service standards and protocols (Alonso et al, 2004; Preist, 2004). Thus the meaning of a "service" and the way its capabilities are described is very much dependent on the characteristics of the implementation environment, e.g. there are intense ongoing discussions in the IT community regarding in which cases one should use which type of "Web services" (Pautasso et al, 2008).

Thus, for the characterization of service platforms, we adopt the notion of an architectural style as defined by Fielding (2000). He argues that architectural styles are better suited to categorize architectures and to define common architectural characteristics. One reason is that it enables a hybrid style approach, i.e. it enables one to apply more than one architectural style within a single architecture.

<u>**Definition (2.6):**</u> An **architectural style** is a coordinated set of **architectural constraints** that restricts the roles/characteristics[10] of **architectural elements** and the allowed relationships among those elements within an architecture that conforms to that style (Fielding, 2000).

Thus, combining definitions 2.5 and 2.6, a service-oriented architectural style may be defined as follows:

<u>**Definition (2.7):**</u> A **service-oriented architectural style** is a coordinated set of architectural constraints that restricts the roles, characteristics and allowed relationships of **services** and service consumers.

A multi-style SOA that may embed hybrid architectural styles is defined as follows:

<u>**Definition (2.8):**</u> A **multi-style service-oriented architecture** is a service-oriented Architecture in which the service-oriented architectural style coexists with other architectural styles.

Based on architectural styles, we define a service platform as follows:

<u>**Definition (2.9):**</u> A **service platform** comprises the set of infrastructural means and rules that are applied in a multi-style service-oriented architecture.

For the service platforms that are in the focus of the present thesis we put a particular emphasis on the combination of a service-oriented architectural style with a resource-oriented architectural style, and the relationship of service and interfaces with resources and their representation. Resource-orientation will be introduced in section 4.2.3 as part of the presentation of the geospatial SOA design methodology. Other extensions may include the explicit support of events and notifications as further important architectural elements and their relationship to services and interfaces.

Abstraction levels of service platforms and related examples are further presented in section 2.2.3 about Service Engineering. Furthermore, section 2.3.4 elaborates on the term "platform" in the context of Model Driven Architecture (MDA).

Finally, we define an open geospatial service platform as follows:

<u>**Definition (2.10):**</u> An **open geospatial service platform** is a service platform that enables the access, management and processing of geospatial information. "Open" means that the specifications of the service interfaces are published, made freely available to interested vendors and users with a view of widespread adoption, and rely upon existing standards where appropriate and possible.

[10] In the original definition of (Fielding, 2000) the term "feature" is used. It has been replaced by "characteristics" in order to avoid misunderstandings with the concept of a "(geographic) feature" as defined in the OGC General Feature Model as an abstraction of a real-world entity (Percivall (ed.), 2003).

2.2.2 SOA Design Principles

A service-oriented architectural style (definition 2.7) restricts the roles, features and relationships of services and interfaces (definition 2.5) by means of architectural constraints. On an abstract level, this corresponds to the SOA design principles defined by Erl (2008a) in his encyclopedia of service design. He defines a design principle as a "highly recommended guideline for shaping solution logic in a certain way and with certain goals in mind". In his encyclopedia he identified the following eight SOA design principles:

- **Service Contract: "**Services within the same service inventory are in compliance with the same contract design standards."

  The fundamental role of this design principle is "to ensure the consistent expression of service capabilities and the overall purpose of the service as defined by the parent service context". It advocates the use of formal, standardized service contracts whereby a service contract comprises a technical interface or one or more service description documents, and the parent service context is determined by the scope of a given service inventory.

- **Service Loose Coupling**: "Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment."

  This principle "emphasizes the reduction ("loosening") of coupling between parts of a service-oriented solution" whereby coupling is understood as the level of dependency between the parts. Loose coupling is "advocated between a service contract and its consumers and between a service contract and its underlying implementation".

- **Service Abstraction**: "Service contracts only contain essential information and information about services is limited to what is published in service contracts."

  The fundamental purpose of this principle is "to avoid the proliferation of unnecessary information about a service, meta or otherwise". Services should hide their internal logic and the technology used from the external environment except those parts that are published in service contracts.

- **Service Reusability**: "Services contain and express agnostic logic and can be positioned as reusable enterprise resources."

  This design principle "strives to get the most possible value out of each piece of software by advocating repeated reuse". It represents "fundamental design characteristics key to achieving many strategic goals associated with SOA". It affects every other principle to various extents. For instance, it demands that a service contract has to be kept as generic as possible such that it is "flexible enough to support multiple consumers with reasonably different interaction requirements". Furthermore, it influences the level of abstraction of service meta-information. On the one hand, there is an "inclination to make contracts as self-descriptive as possible" to maximize their reuse potential, but on the other hand a service contract needs to be "concise so that it does not describe or constrain itself in a way that would inhibit its future reuse" even for service consumers that will be unknown at the time the service is deployed.

- **Service Autonomy**: "Services exercise a high level of control over their underlying runtime execution environment."

  This design principle represents the independence with which a service implementation can carry out its logic and manage the resources it may need at runtime. It supports the reusability and the composability of services.

- **Service Statelessness**: "Services minimize resource consumption by deferring the management of state information when necessary."

  This design principle encourages the SOA designer to "incorporate state management deferral extensions within service designs so as to keep services in a stateless condition wherever appropriate". Erl (2008a) defines state data as "information primarily associated with a current activity" and state management as the "processing of this information". State management deferral is accomplished by temporarily delegating the responsibility for state management to another part of the architecture. The primary objective of service statelessness is to maximize service scalability.

- **Service Discoverability**: "Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted."

  This design principle will help the SOA designer to determine whether the "automation requirements that need to be fulfilled already exist within a service inventory", i.e. to decide whether to use an existing service or to build a new one. Erl (2008a) distinguishes between design-time and runtime discovery. Design-time discovery refers to the "manual process of discovery by humans", whereby runtime discovery is the "automated process" of programs and services to issue dynamic discovery queries to programmatic interfaces of service registries.

- **Service Composability**: "Services are effective composition participants, regardless of the size and complexity of the composition."

  This design principle contributes to "ensure that services are able to participate in multiple compositions to solve multiple larger problems", even when "immediate composition requirements do not exist". It is closely related to Service Reusability because "composition can be seen as a form of reuse". It is basically the application of the basic principle of composing a (software) system into (software) components to a service-oriented environment.

2.2.3 Service Engineering

When engineering a service-oriented distributed system, the SOA design principles presented above play a major role and must be considered. Even more, they should be exploited in order to leverage the potential of the SOA. Although different authors put different emphasis upon the individual principles, the three principles **service autonomy, reusability** and **composability** seem to be the most relevant ones from a business and design perspective. For instance, Zhang et al (2008) state that what IBM has learned from hundreds of projects is "that the construct of a service can be used to further realize proper separation of concerns, resulting in IT architectures that are easier to maintain and costs that are lower." High, Krishnan and Sanchez (2008) argue that the major asset but also the major challenge for an SOA design is the question of how to define their architectural elements (services and interfaces) such that their reusability is maximized. When designing an SOA, the principle of a service contract comprises two facets, a **document agreement** and a **communication agreement** (Erl, 2008a). We interpret the document agreement for a service contract to be the documented interfaces of a service, whereas the communication agreement refers to the usage of a common well-defined service platform (see the definition 2.9 in section 2.2.1). Service platforms may be abstract, i.e. technology neutral, or concrete, i.e. tied to a well-specified underlying communication infrastructure. Service platforms encompass a service model (a meta-model for services) that specifies the guidelines and constraints for the interactions between a service provider and a service consumer.

- An abstract service platform is usually specified by means of the Unified Modelling Language (UML) of the Object Management Group (OMG) (Rumbaugh et al, 1998) following a technology-neutral service model, e.g. the Reference Model for Service-oriented Architecture as specified by OASIS (2006), see section 3.2.4.
- Concrete service platforms provide execution contexts in the sense of OASIS (2006). Examples are:
    1. Web services according to the Web Services Architecture of the World Wide Web Consortium (W3C, 2004a), using the Web Service Description Language (WSDL) (W3C, 2007a) as interface description language, including a profile of the OGC Geographic Markup Language (GML) for the representation of geographic information (Percivall (ed.), 2008) and supporting event notifications according to the OASIS Web Services Notification standard (Graham et al (eds.), 2006); but also
    2. A distributed object-oriented infrastructure according to OMG's Common Object Request Broker Architecture (CORBA) specifications (Mowbray and Ruh, 1997) that enables the remote invocation and event architectural styles (OMG, 2003).

  Furthermore, we assume that in a concrete service platform service discoverability is enabled by collecting information about services (service meta-information) in service catalogues (OGC, 1999) resp. registries.

From the viewpoint of the system designer, the major advantage of an SOA is design efficiency and sustainability: already existing capabilities as well as capabilities incrementally added by the designer may be re-used and offered to fulfil future, still unknown requirements. Thus, the kernel challenge for the design of an SOA is the question of how the requirements of the user can be assessed against the already existing capabilities of service platforms. On one hand, capabilities may be deployed service instances with published interfaces and descriptions in catalogues but on the other hand might only be specifications of service types or information models. The latter is particularly true for geospatial information systems where "open" capabilities are required that comply with ISO or OGC standard specifications. For geospatial applications, this is a very important side condition for the SOA design.

The Internet as well as its related technologies and products provide the foundation for the technical realization of an SOA with semantic and geospatial extensions:

- The World Wide Web enables the direct access to information stores.
- Standard Web service technologies, recommended by the W3C enable the invocation of computational facilities (Web services) over the Internet via an interface (Alonso et al, 2004), independent of implementation aspects such as operating system platforms and the programming language used (Fensel et al, 2006). However, looking at these enabling technologies in detail, there are competing architectures which rely upon different architectural styles (Richardson and Ruby, 2007).
- The combination of semantic technologies with Web services enable the semantic description of Web resources (being data and service elements) and the development of languages, design and run-time frameworks for so-called Semantic Web Services (Fensel et al, 2006).
- Information models of the ISO 191xx series of geomatics standards provide a reference model (ISO 19101:2004) and a framework for geospatial information (ISO 19109:2005) and geospatial services (ISO 19119:2005).
- Standardized OGC services provide the call of geospatial services, e.g. for the access to geospatial data sets, the execution of geostatistical calculations and the generation of interactive maps from multiple geospatial servers (Percivall (ed.), 2008).
- Recent extensions of the OGC service environment tackle the access to, the tasking and the management of sensors over the Internet enabling a so-called Sensor Web (Simonis (ed.), 2008).
- Catalogue services facilitate the publication, search and discovery of geospatial resources (Nebert, Whiteside and Vretanos (eds.), 2007).

Technologies to implement the SOA infrastructures are largely available today. Current research focuses on the standardization of qualitative SOA functions, e.g. how to model and provide security in SOAs (Klarl and Preitsameter, 2006), or how to enrich geospatial catalogue services with semantic technologies (Stock, 2009; Hilbring and

Usländer, 2006). The SOA design principle of loose service coupling on the basis of an agreed service platform fosters its use in geospatial information systems. It enables sharing of geospatial resources, i.e. data and services with an explicit or implicit geo-spatial reference, in order to compose them to higher-level entities and use them in geospatial applications possibly distributed across organizational and administrative boundaries. This is essential for EIS as natural phenomena are not limited to boundaries drawn by humans.

However, as in many other businesses, the transition to service-oriented solutions is still ongoing. Although the maturity of currently available SOA technologies is acknowledged, their applicability to public services is still considered to be low (Wahlster and Raffler, 2008). One of the reasons is that SOA requires significant changes in business process design as well as in the modelling and solution development. More than any other objective, SOA is intended to create a stronger alignment between information technology (IT) and the businesses it supports (High, Krishnan and Sanchez, 2008). It is a discipline that spans the entire spectrum from business architecture to IT implementation. Business components are at the business end of the spectrum. They describe a business as a collection of coarsely grained and usually informally specified **business services**, whereas at the other end, **IT services** are fine-grained and specified precisely (Bercovici, Fournier and Wecker, 2008). The challenge is to bridge these two levels, i.e., to map business services to the IT services, and to provide corresponding design tools and software environments (Karakostas and Zorgios, 2008). Model-driven design approach are the cornerstone of currently existing, but mostly proprietary SOA design and modelling solutions such as IBM's Service-Oriented Modelling and Architecture SOMA (Arsanjani et al, 2008).

2.2.4 SOA in System-of-Systems Engineering

Users in the scientific domain use services owned by other organizations (Tan et al, 2009) which may be environmental agencies, typically acting as data providers, or specialized companies playing the role of independent "business components" that sell advanced business services, e.g. geo-referenced thematic maps derived from raw satellite images. Thus, large-scale EIS result in a system-of-systems architecture that spans multiple organizational, national and technological barriers. There is no agreed definition of the term system-of-systems (SoS), instead SoS are usually discussed by their characteristics that distinguish SoS from other large and complex but monolithic systems. The most distinctive characteristics of SoS are the independence of their component systems in terms of operation, management and evolution. This independence and the distribution of an SoS over a large geographic extent result in an "even greater emphasis on interface design that in traditional system architecting and engineering" (Maier, 1998). Béjar et al (2008) propose systems of systems as a conceptual framework to support the creation, evolution and study of spatial data infrastructures. The SoS approach is also discussed in the domain of homeland security covering the following five SoS building blocks: sensor layer, remote command post, central command post, cooperative systems and intelligence (Franceschetti and Grossi, 2008). They claim that the complexity of engineering such SoS solutions requires a complement to system engineering called SoS Engineering as the necessary tool to conceive, develop, deploy, and evolve homeland security SoS, and propose SOA to be the architectural approach to follow to integrate the building blocks.

The OGC Sensor Web Enablement (SWE) architecture (Simonis (ed.), 2008) and related research activities, for instance in the European project Sensors Anywhere (SANY) (Havlik et al, 2007), aim at integrating the sensor layer into an SOA landscape. It specifies an architecture, services and information models for the access to and management of sensors and sensor-related information such as observations and measurements. The upcoming Sensor Web standards are commonly the technical foundation for the integration of environmental services and information in SoS environments.

A recent example of an SoS approach in the environmental domain that is heavily based upon SOA and the OGC SWE architecture is the world-wide initiative to create GEOSS – a Global Earth Observation System of Systems (GEO, 2008). GEOSS is an intergovernmental programme that aims to integrate earth observation systems into a global system that can be applied to various areas of environmental science and management. It is focussing on so-called societal benefit areas, e.g. the reduction and prevention of disasters, climate change or energy and water management. GEOSS is composed of a variety of EIS including those for data collection, processing, discovery and dissemination (Figure 2-8). The coupling of the individual EIS shall be carried out in an SOA environment (GEOSS AIP, 2008).

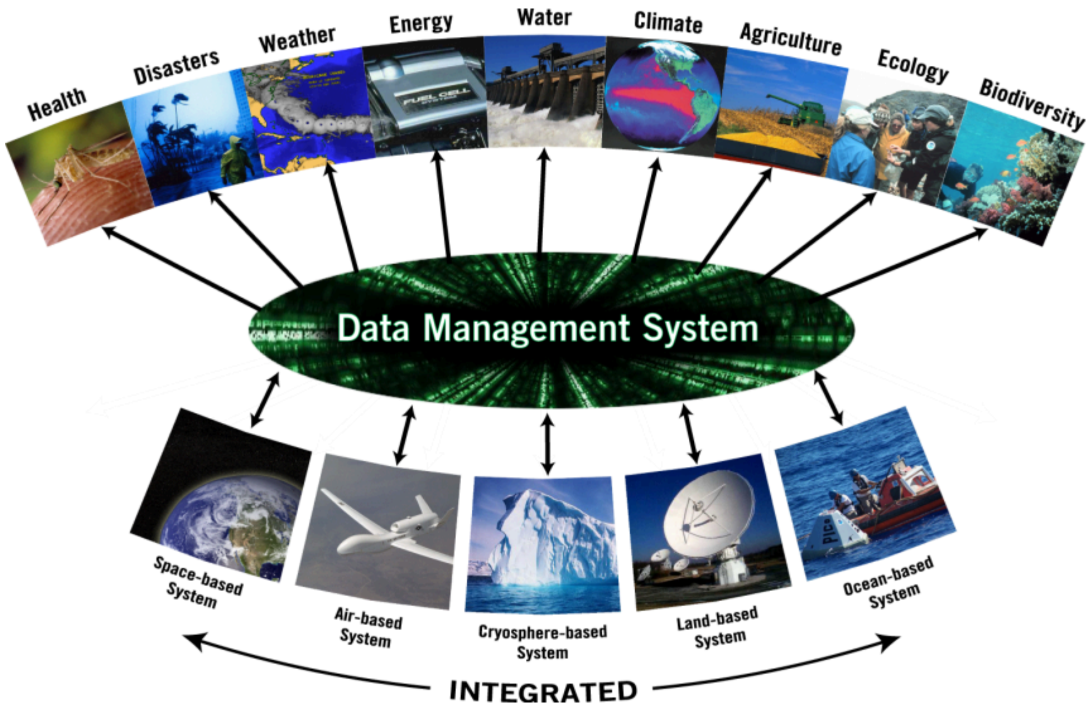


Figure 2-8: Global Earth Observation System of Systems (Source: GEOSS AIP, 2008)

2.3 Information Systems Engineering

2.3.1 Conceptual Framework

The research community of Information Systems (IS) deals with the design and engineering of information systems in a broader context, including socio-economical effects of IS applications. It is the purpose of the present section to relate service-oriented design methodologies to research in IS.

Hevner et al (2004) presents a conceptual framework for understanding and executing IS research combining the behavioural-science paradigm and the design science paradigm. As in the software engineering glossary of IEEE 610.12 (1990) the term "design" hereby denotes both a process (set of activities) and a product (artefact). The design process is considered as an iterative application of the expert activities **build** and **evaluate** that produces an innovative product (i.e., the design artefact). The evaluation of the artefact provides feedback information and a better understanding of the problem in order to improve both the quality of the product and the design process. Figure 2-9 shows a restricted version of this framework which focuses on the **building** and **evaluation** of artefacts as these are the high-level activities that characterize the design-science paradigm[11]. The behavioural-science paradigm which focuses on the development and justification of theories has been omitted as it is out of scope of this thesis.

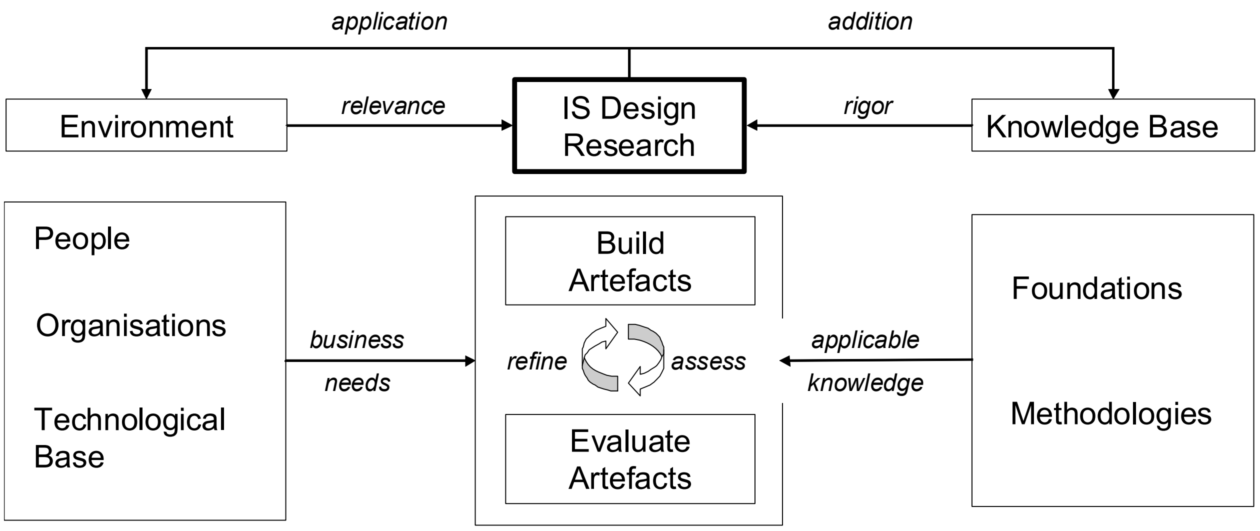


Figure 2-9: Framework for the Design of Information Systems, derived from Hevner et al (2004)

On the left side, there is the "environment" of the IS Design Research. The environment defines the problem space and thus the "relevance" for the research in IS design. The environment defines the business needs that determine the design of an IS. It is composed of

[11] The behavioural-science paradigm which focuses on the development and justification of theories has been omitted as it is out of scope of this thesis.

- **people** with their roles, capabilities and characteristics,
- **organizations** with their strategies, structure and culture and processes, and
- the **technological base** encompassing development capabilities, infrastructure and communication technologies as well as applications.

In the application domain of environmental management, people are mainly the EIS users in environmental agencies. The technological base comprises, among others, environmental sensors, service-oriented middleware and visualisation techniques for geospatial and temporal information. Usually, the technological base only comprises a subset of the technological capabilities potentially existing on the market. This is due to side constraints or pre-selections of the "environment" expressed by people and organizations. Examples are results of market studies, experiences with former and existing systems or even personal preferences of the customer (people), or technological roadmaps and the need for integration in an existing landscape of an organization. As a result, system designers are not totally free in their technological choice in the attempt to satisfy customers' business needs.

On the right side in Figure 2-9, there is the "knowledge base" of the IS Design Research. The knowledge base provides the "raw materials from and through which IS research is accomplished" (Hevner et al, 2004). It gives the "rigor" to the IS Design Research and is composed of

- **foundations** with theories, frameworks, instruments and the design artefacts,
- and **methodologies** which provide guidelines about how to use the foundations to a design problem.

Usually, just a subset of the existing knowledge base, here called the "applicable knowledge", may be applied in the design process. The reasons may be cost or availability constraints or the fact that an existing design methodology is not compatible with the technological base of the "environment". An example is the selection of a programming environment imposed by the customer in the technological base for which there is no affordable or effective development tool for a round-robin development, which is, for instance, a pre-requisite for the successful application of model-driven architecture (MDA) as a design methodology (Asadi and Ramsin, 2008).

This thesis aims at contributing to the "knowledge base" of this research framework. In the "foundations", there is a lack of an adequate reference model for large-scale EIS based upon geospatial standards. This gap is closed by the reference model presented in section 5. Furthermore, there is a lack of an associated design methodology. The SERVUS Design Methodology specified in section 4 is a contribution to this part of the knowledge base.

2.3.2 Software Development Methodologies

Avison and Fitzgerald (2003) argue that since the late 1990s we have been living in a "post-methodology era", largely because, after a long discussion among researchers and software engineers in organizations, no consensus has been reached which methodology to adopt and to use for which design task. Their argumentation is based on the understanding that a methodology is a "recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system", underpinned by the "set of beliefs and assumptions" that explains why it functions as it does. They consider the following seven broad themes (or approaches) as not mutually exclusive classifiers of classical methodologies:

- **Structured**: application of structured programming techniques
- **Data-oriented**: data as key element in a system's development
- **Prototyping**: building approximations of a system prior to its physical implementation
- **Object-oriented**: application of the principles of object-orientation such as inheritance and reuse
- **Participative**: involvement of users and other stakeholders
- **Strategic**: emphasis on the support and enablement of the overall business objectives
- **Systems**: holistic view far beyond a system's single-application boundaries.

These approaches characterize the "methodology era", however, methodologies specified on this basis are often said to be "one-dimensional", e.g. not addressing a particular organization's underlying issues or problems or not flexible enough to allow changes to requirements during development. The demand for a strict adherence to a methodology has led to the situation that some organizations "rejected the use of system development methodologies altogether, returning to less-formal, more off-the-cut, perhaps more flexible approaches".

The lack of acceptance of methodologies in practice is often due to the rigor that some of the methodologies request. Avison and Fitzgerald (2003) state that the "problem is not the concept of a methodology but their inadequacy". Offermann and Bub (2009) carried out an empirical comparison of methods for information systems development according to SOA. They concluded that the "ranking of all methods is relatively close to neutral. It is important to investigate whether there is a general problem in the way methods are being described." The Object Management Group (OMG) has recently taken up this problem by the specification of a meta-model for software and systems process engineering called SPEM (OMG, 2008d) which documents the continued interest in methodologies by the software industry. SPEM accepts the methodology diversity and the need for flexibility as it accommodates "a large range of development methods and processes of different styles, cultural backgrounds, levels of formalism, lifecycle models, and communities". This is achieved by a modular modelling structure in terms of packages specified in UML (Rumbaugh et al, 1998) that enables implementers to choose the generic behaviour modelling approach that best fits their needs.

Referring to the seven broad themes in the methodology era described above, we recognize that new themes such as service-orientation, MDA and semantics have emerged that may replace or at least complement the traditional ones. However, especially in SOA environments, we also observe the growing importance of the themes "participative", "strategic" and "systems".

2.3.3 Co-Development of Requirements and Architecture

It is widely recognized that the design of complex information systems cannot be performed in a single design step, neither when starting top-down from a list of requirements after a careful analysis of the problem space, nor bottom-up by drafting a solution space built upon the capabilities of an existing technological basis. Nuseibeh (2001) states that "Achieving a separation of requirements and design steps is often difficult because their artificial ordering compels developers to focus on either aspect at any given time. In reality, candidate architectures can constrain designers from meeting particular requirements, and the choice of requirements can influence the architecture that designers select or develop". As an alternative, there is a need for a co-development of system requirements and functional architecture in multiple iterations (Pohl and Sikora, 2007). However, there are only few methodologies to support such a co-development.

As an adaptation of the spiral life-cycle model Nuseibeh (2001) proposes a so-called Twin Peaks model to emphasize the equal status giving to requirements and architectures. The Twin Peaks model develops concurrently and progressively from coarse-grained to detailed requirements and architectural specifications. It addresses the following three management concerns

- **I'll Know It When I See It (IKIWISI)**: Requirements often emerge only after users have had an opportunity to view and provide feedback on models or prototypes.
- **Commercial off-the-shelf software**: Increasingly, software development is actually a process of identifying and selecting desirable requirements from existing commercially available software packages. With Twin Peaks, developers can identify requirements and match architectures with commercially available products, rapidly and incrementally.
- **Rapid change**: As the Twin Peaks model focuses on finer-grain development, it is receptive to changes as they occur. The analysis and identification of a software system's core requirements are a pre-requisite to developing stable software architectures regardless of changing requirements.

According to Berente and Lyytinen (2007) the concept of iteration is fundamental to design activity and therefore inherent to each design methodology. However, they argue that the artefacts whose design is performed in an iterative process must be carefully identified when describing a design methodology. With respect to the Twin Peaks model this means that the level of details achieved in each iteration step depends on the types of design artefacts resulting from each iteration step. Already early iteration steps may lead to very detailed implementation architectures or even a realization and deployment of a system in order to get first feedback from practical use as soon as possible. The subsequent iteration steps then enlarge the coverage of the user requirements.

Having this in mind, Pohl and Sikora (2007) distinguish in their s**C**enario and g**O**al based **S**yste**M** development meth**OD** (COSMOD-RE) for software-intensive systems between artefacts on four abstraction layers: system (L_1), functional components (L_2), hardware/software (L_3) and software deployment (L_4). Important requirements artefacts on system level are **goals** that refine the overall system vision into a hierarchically structured documentation of high-level system properties, and **scenarios** that define interactions between external actors and the system. The basic principle of COSMOD-RE is to develop requirements artefacts on level L_i and architecture artefacts on level L_{i+1} in one co-development process[12]. For instance, the system-level co-development process develops system requirements (L_1), e.g. goals and scenarios, and the related logical system architecture (L_2) which defines a decomposition of the overall system into a set of functional components with well-defined interfaces.

2.3.4 Model-driven Architecture

Conceptual modelling is one of the cornerstones in information systems engineering. It captures the general knowledge of the system in conceptual schemas (Krogstie et al (eds.), 2007). A systematic approach for a model-driven design of information systems is the Model-driven Architecture (MDA) originated by the Object Management Group (OMG). MDA separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform (Asadi and Ramsin, 2008). An MDA approach shall thus improve the portability, interoperability and reusability of software. The OMG has defined a number of standards to support these goals and to provide a modelling infrastructure of the MDA (OMG, 2003), among which the following general modelling standards are highly relevant for the present work:

- Unified Modeling Language (UML) (Rumbaugh et al, 1998) providing a standard modeling language for visualising, specifying, and documenting software systems. Models used with MDA can be expressed using the UML language.

[12] In Pohl and Sikora (2007) called co-design process.

- Meta Object Facility (MOF) (OMG, 2002) providing a model repository that can be used to specify and manipulate models, thus encouraging consistency in manipulating models in all phases of the use of MDA.

MDA achieves platform-neutrality by modelling the functionality of a system as long as possible independently of the peculiarities of an underlying platform. Hereby, the MDA Guide (OMG, 2003) defines a platform as “a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented.”

This definition is quite high-level in order to encompass a variety of platform types from hardware, operating systems and virtual machines up to software frameworks. As a refinement of the MDA platform model Atkinson and Kühne (2005) propose four basic facets to characterize a service platform:

- Language: describes the basic concepts with which applications designed to use the platform can be constructed.
- Predefined types: augment the core language capabilities with additional services.
- Predefined instances: contains pre-instantiated objects ready to be used out of the box.
- Patterns: consists of additional concepts and rules that are needed to use the capabilities in a meaningful fashion.

A hierarchy of models is proposed by MDA which quite nicely corresponds to our view of how requirements are mapped in multiple steps to service platform capabilities as introduced in section 1.4. The correspondence is illustrated in Figure 2-10.

The model hierarchy comprises:

- The **Computation-Independent Model (CIM)**, specifying the system requirements, corresponds to the “model of the problem domain”.
- The **Platform-Independent Model (PIM)**, describing the system design independent of the implementation platform, corresponds to the “abstract service platform” which results from the model of the problem domain by an abstract design step.
- The **Platform-Specific Model (PSM)**, describing system design in the form of a platform-dependent model, corresponds to the “concrete service platform”. The idea of the PSM is that the PIM may be (semi-)automatically transformed into a PSM with the help of a Platform Model. This transformation corresponds to the concrete design step that maps the elements of an abstract service platform to those of a concrete service platform.

- The **Implementation-Specific Model (ISM)**, describing the realization of the PSM in terms of software components and its arrangement in a system architecture, corresponds to a (geospatial) "service network" resulting from the elements of a concrete service platform in an engineering step.

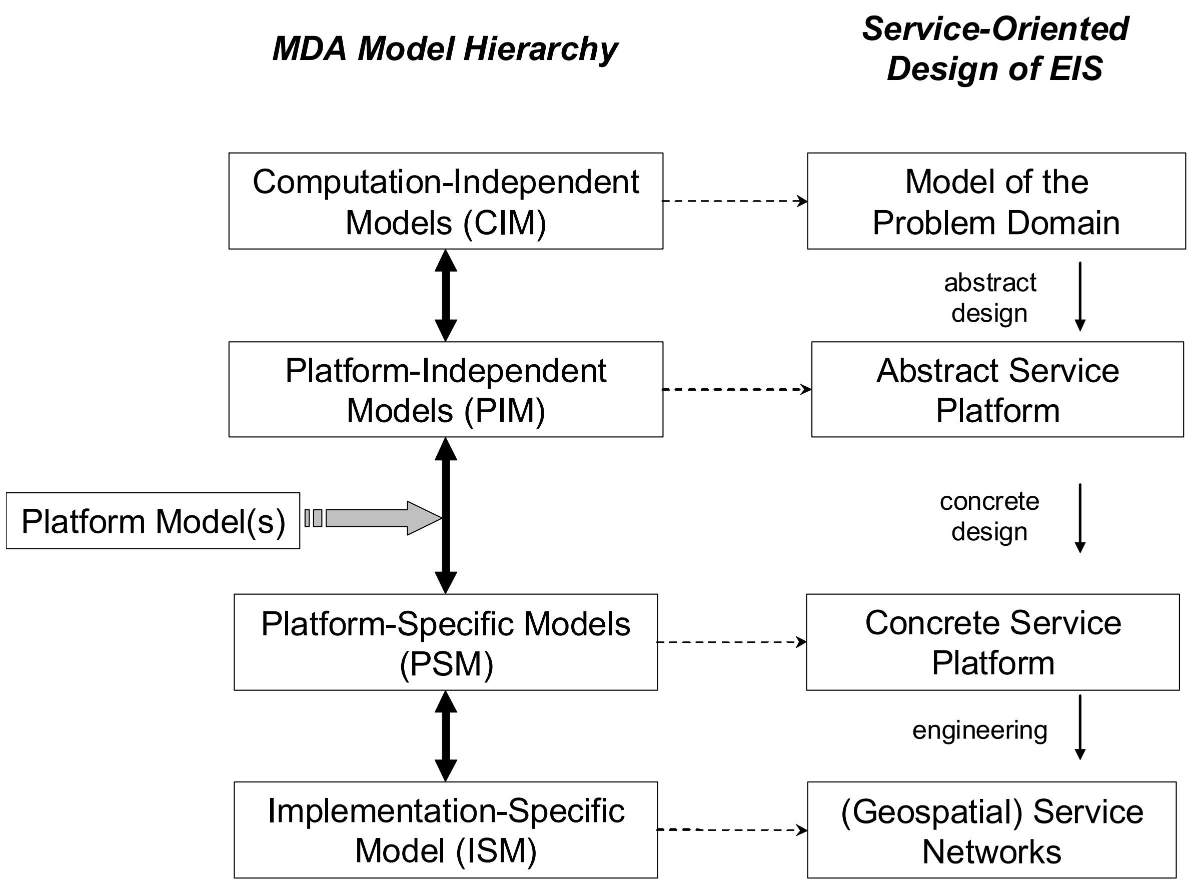

MDA Model Hierarchy
Service-Oriented Design of EIS
Computation-Independent Models (CIM)
Model of the Problem Domain
abstract design
Platform-Independent Models (PIM)
Abstract Service Platform
Platform Model(s)
concrete design
Platform-Specific Models (PSM)
Concrete Service Platform
engineering
Implementation-Specific Model (ISM)
(Geospatial) Service Networks

Figure 2-10: MDA Model Hierarchy mapped to the Service-Oriented Design of EIS

Asadi and Ramsin (2008) state that "MDA is not a methodology, but rather an approach to software development". Their argumentation is based on the definition that a software development methodology is more comprehensive, i.e. methodologies may be defined on the basis of an MDA approach.

They should at least consist of two main parts[13]:

- A set of modelling conventions comprising a **modelling language** (syntax and semantics), and
- a **process** which provides guidelines as to the order of the activities and specifies the artefacts developed using the modelling language.

[13] Following the methodology definition of Avison and Fitzgerald (2003) as introduced in section 2.3.1, a methodology should in addition encompass tools, documentation, management, and training used to develop a system.

Asadi and Ramsin (2008) provide an analytical survey of six MDA-based methodologies (MODA-TEL, MASTER, C^3, ODAC, DREAM, DRIP-Catalyst). They analysed the methodologies in terms of tool and MDA support as well as general evaluation criteria such as lifecycle coverage, reusability support and application scope. For our purposes the following results are important:

- All methodologies incorporate activities for creating the PIM and the PSM, however, the creation of the CIM is mostly neglected.
- Tool support or even just guidelines for tool support are insufficient or incomplete.
- Most of the methodologies provide techniques for creating and applying reusable artefacts, however, it is not comprehensive enough as techniques for recording the syntactic and semantic features of the reusable artefacts are neglected.
- Most of the methodologies use conventional object-oriented analysis and design techniques to produce PIMs.

They conclude with the statement that the MDA-based methodologies studied in their analytical survey are "not mature enough, especially as pertaining to providing support for standard software engineering activities." Nevertheless, there is ongoing research to extend MDA-based approaches towards ontology-based software engineering approaches (Hesse, 2008). The idea is to exploit the expressiveness of ontology languages for the specification of computation-independent models (CIM). However, there are doubts about the use of ontologies as the adequate starting point for CIMs. The doubts are related to the problem of "closure": since ontologies, at least domain ontologies, are typically developed for given thematic domain without having a particular application in mind, there may be difficulties regarding the completeness of an ontology for an information system to be designed (Bubenko, 2007).

In the modelling framework of the SERVUS Design Methodology presented later on in section 4, these problems are handled by applying techniques for semantic annotation (see section 3.3.3) instead of directly using ontologies as the modelling language for the information system designer.

3 State of the Art

3.1 Overview

Research about the design of EIS based upon open geospatial service platforms and the principles of service-oriented architectures is not a dedicated research discipline of its own. Thus, the description of the state of the art in this research domain is manifold. The selection of the concepts and technologies presented in this section was motivated and guided by their potential to support the requirements for a design methodology listed in section 1.4.3. The following three research domains are investigated and assessed against the list of design requirements:

- **Reference models** in the context of Service-oriented Architecture (SOA) and their evolution towards geospatial SOA and semantic SOA (section 3.2). The state of the art is evaluated against their support of analytical power (R.4, R.5, R.6, R.7 and R.8) and realism (R.10, R.11).
- **Semantic Technologies** (section 3.3), in particular semantic description of information and services, geospatial semantics and the application of semantic technologies to software engineering. The state of the art is evaluated against their support of analytical power (R.7) and realism (R.11 and R.12).
- **Service-oriented Analysis and Design** (section 3.4) including application of model-driven architecture to SOA and service matching. The state of the art is evaluated against all requirements of human comprehension (R.1 – R.3), analytical power (R.4 -R.9) and realism (R.10 - R.12).

3.2 Evolution of Reference Models

3.2.1 Introduction

For many years, the European Commission has supported research activities to analyse user and system requirements, to derive architectural principles, and to specify and implement generic components of ICT architectures for large-scale environmental information systems (Coene and Gasser, 2007). In parallel, on a world-wide level, various approaches of standardization organizations such as ISO, OGC, OMG, OASIS and The Open Group were launched to stimulate a market based on agreed architectures with the aim of fostering interoperable solutions. Each standardization activity has started by the definition of terms, high-level concepts and their relationships, resulting in a series of reference models (see Figure 3-1), partly competing and partly complementary.

OASIS (2006) defines a reference model (RM) as an "abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment." These reference models set the conceptual foundation of distributed systems. Originally inspired by the architectural style of interacting objects for distributed processing (see the ISO RM-ODP presented in section 3.2.2 and illustrated in the lower left corner of see Figure 3-1), they are nowadays discussed in the context of SOAs.

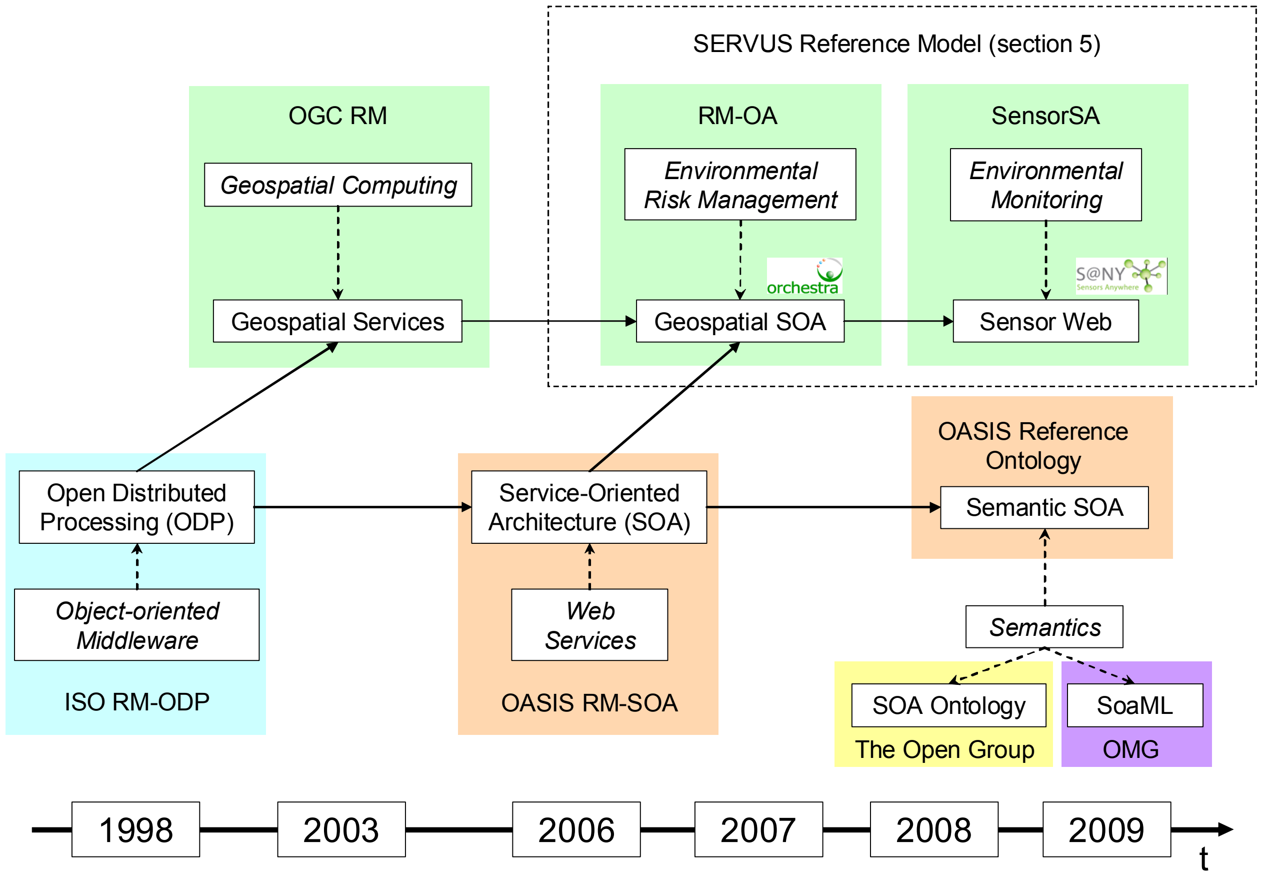


Figure 3-1: Evolution of Reference Models (RM)

Figure 3-1 illustrates the two major evolution lines of reference models. The upper line shows the interpretation of the RM-ODP for geospatial distributed processing. It has started with the OGC Reference Model (section 3.2.3) followed by the Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer, 2009a) whose core was developed by this thesis. The latest extensions include sensors and sensor service networks resulting in the Sensor Service Architecture (SensorSA) (Usländer (ed.), 2009b) based upon OGC Sensor Web Enablement architecture (Simonis (ed.), 2008). The RM-OA and the SensorSA are described in section 5 as part of the SERVUS Reference Model. They both were influenced by the specification of SOA reference models that are drafted by OASIS. Their evolution is shown in the lower line in Figure 3-1. Their latest movement towards semantically-enabled SOAs and SOA ontologies is described in section 3.2.5 and in section 3.2.7 in the context of the assessment of reference models.

3.2.2 ISO Reference Model for Open Distributed Processing

Inspired by "distributed processing systems based on interacting objects", ISO defined the Reference Model for Open Distributed Processing (RM-ODP) (ISO/IEC 10746-1:1998). The RM-ODP is an international standard for architecting open, distributed processing systems that

- constitutes a way of thinking about architectural issues in terms of fundamental patterns or organizing principles,
- provides a set of guiding concepts and terminology for building distributed systems in an incremental manner, and
- defines a framework for distributed system specification covering all aspects: "enterprise" context, functionality, distribution, infrastructure and technology.

The RM-ODP standards were widely adopted. They constituted the conceptual basis for the ISO 191xx series of geomatics standards. Their original ideas were partly applied in the specification of the OMG Object Management Architecture (OMG, 1997) as a framework of object-oriented distributed systems. 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.

There is no single way of specifying ODP systems, instead an architecture of such a complexity is typically specified from several viewpoints. The use of viewpoints is derived from the principle of abstraction as the heart of architectural specifications, i.e. the process of suppressing selected detail to establish a simplified model. The RM-ODP defines a viewpoint as a "form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system."

The viewpoints enable the **separation of concerns** in an ODP system specification whereby each viewpoint uses dedicated language constructs to express the viewpoint-specific concerns and decisions.

The RM-ODP distinguishes between five standard viewpoints:

- The **Enterprise Viewpoint**: A viewpoint on the system and its environment that focuses on the purpose, scope and policies for the system, i.e. the roles played by an ODP system in its organizational environment.
- 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 decomposition of the system into units of functionality, e.g. 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 including quality of service constraints.
- The **Technology Viewpoint**: A viewpoint on the system and its environment that focuses on the choice of technology in that system. Furthermore, it specifies the hardware and software components from which the system is built.

Notes:

1) The RM-ODP does not impose or propose rules about how to assign viewpoint specifications to the individual activities of a design process.
2) The classical H/H/P method of Hatley, Hruschka and Pirbhai (2000) for software and system design distinguishes between requirements, architecture and design, and related models. Compared to the RM-ODP viewpoints, "requirements" would be captured in the enterprise viewpoint, the "architecture" would be described in the information and computational viewpoint, and the engineering and technology viewpoint are belonging to the "design", having in mind that the boundary between architecture and design is fluent in the H/H/P method.

   When being applied, these viewpoint definitions have to be interpreted according to the needs and purposes of the ODP system to be designed, e.g. see below its interpretation for the OGC Reference Model in section 3.2.3.
3) Although the Object Management group (OMG) adopted the ISO RM-ODP principle and definition of the term "viewpoint", its specification of a Model-Driven Architecture (MDA) (OMG, 2003) uses a different classification system of viewpoints. See the MDA description in 2.3.4.

The RM-ODP standard encourages the use of formal description techniques, but does not propose itself any specific language or notation for the viewpoint specifications. Furthermore, in contrast to the "perspectives" of Zachman's framework of information systems architecture (Zachman, 1987), the representation form of the RM-ODP viewpoints is not differentiated according to the "participant" in the construction of the viewpoint.

There is, however, a follow-on ISO standard (ISO/IEC 19793:2008) on the use of the Unified Modelling Language (UML) (Rumbaugh et al, 1998) for the expression of ODP system specification. For each viewpoint, it identifies the major concepts and extends UML by the specification of dedicated meta-classes (OMG, 2007). An example is the UML expression of a (business) policy together with the objects and the behaviour (modelled as interactions between objects) constrained by the policy.

3.2.3 OGC Reference Model

The RM-ODP approach was used in the design of the original OGC Reference Model of 2003 (Percivall (ed.), 2003) which laid the basis for the development of a series of OGC standards for geospatial services and information models. The OGC Reference Model interprets the RM-ODP viewpoints as described in Table 3-1. However, it does neither provide any guidance about how to use and order the viewpoints in a design process nor is the interpretation oriented at the principles of service-oriented computing.

RM-ODP Viewpoint	Definition according to ISO/IEC 10746	Definition according to the OGC Reference Model Version 0.1.3
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.
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.
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.
Technology	Concerned with the choice of technology to support system distribution.	Focuses on the choice of technology.
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.

Table 3-1: Interpretation of the RM-ODP Viewpoints for the OGC Reference Model (Percivall (ed.), 2003)

As language for the specification of the information viewpoint it adopted the specification of the General Feature Model (GFM) defined in the ISO rules for application schema (ISO 19109:2005). The GFM is a meta-model for information models using UML extension mechanisms (OMG, 2007). It defines a feature as an abstraction of a real world phenomenon. A feature is the basic unit for information modelling. Individual feature instances are grouped into feature types where all instances of a certain type are described by common properties such as thematic, temporal or spatial attributes or associations with other feature types.

For the specification of the computational viewpoint the OGC Reference Model identified types and names of geospatial services and categorized them according to ISO 19119:2005. An overview and short introduction to all OGC standards that are derived from the OGC Reference Model (Version 0.1.3) of 2003 is covered by its latest version of 2008 (Percivall (ed.), 2008).

3.2.4 OASIS Reference Model for Service Oriented Architecture

The OASIS organization was the first in specifying a reference model tailored to a service-oriented architecture (OASIS, 2006). It is an abstract framework for understanding significant entities and relationships between them within a service oriented environment. Being based on unifying concepts of SOA, it fosters the development of consistent standards or specifications supporting that environment, and may be used by architects developing specific SOA or in training and explaining SOA. The OASIS SOA Reference Model (SOA-RM) defines seven principal concepts. Their relations are described and illustrated in OASIS (2006). Here, just a short overview about the concepts is given:

- **Service:** A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the **service description**.
- **Service description**: The service description represents the information needed in order to use a service. Its elements depend on the context and the needs of the parties using the associated entity. The purpose of description is to facilitate **interaction** and **visibility**, particularly when the participants are in different ownership domains. Service descriptions make it possible for potential participants to construct systems that use services (e.g. client applications) and even offer compatible services. Best practice suggests that the service description should be represented using a standard format facilitating the use of processing tools (such as discovery engines).
- **Visibility**: Visibility is the relationship between **service** consumers and providers that is satisfied when they are able to interact with each other. Preconditions to visibility are awareness (i.e., a state whereby one party has knowledge of the existence of the other party), willingness (i.e., the intent act to initiate and to participate in a service interaction) and reachability (i.e., the relationship between service participants where they are able to interact).
- **Interaction**: Interaction deals with the question about how to interact with the service in order to achieve the required objectives. This involves performing actions against the service. In many cases, this is accomplished by sending and receiving messages. Key requirements for successful interactions revolve around the **service description** which references an information model and a behaviour model. The information model characterizes the information that may be exchanged in terms of structure and semantics.

The behaviour model comprises the knowledge of the actions invoked against the service and their **real world effect** (action model) and the temporal dependencies between actions on the service (process model).

- **Real world effect**: A real world effect captures the consequence of invoking a **service**. It may include information returned in response to a request for that information, a change to the shared state of defined entities, or some combination these two.
- **Contract and Policy:** A policy represents some constraint or condition on the use, deployment or description of an owned entity as defined by any participant. A policy is defined by a policy assertion, the policy owner and policy enforcement. Policies potentially apply to many aspects of SOA: security, privacy, manageability, quality of service and so on. A policy is associated with the point of view of individual participants. As a complementing concept the SOA-RM defines a contract as an agreement between two or more participants.
- **Execution context**: The execution context of a service **interaction** is the set of infrastructure elements, process entities, **policy** assertions and **contracts** that are identified as part of an instantiated service **interaction**, and thus forms a path between those with needs and those with capabilities.

Furthermore, the SOA-RM outlines expectations to system designs that claim conformance. For instance, conformant system specification shall identify the entities that can be identified as services and means how to describe services.

3.2.5 OASIS Reference Ontology for Semantic Service Oriented Architectures

OASIS has worked on semantic extensions of the SOA-RM. These are well summarized in Fensel et al (2008). A major result is the Reference Ontology for Semantic Service Oriented Architectures (SSOA-RO) (OASIS, 2008) which is an "abstract framework for understanding significant entities and relationships between them within a Semantically-enabled Service-Oriented environment." On the one hand, this reference ontology is considered to be a formal specification of the basic concepts of the SOA-RM. On the other hand, it enhances the SOA Reference Model by "key concepts of semantics that are relevant for Semantically-enabling Service Oriented Architectures", which finally means, that the semantics-based approach has slightly changed the original SOA-RM conceptual model.

The semantics-based approach relies on two fundamental principles:

1. All service descriptions shall be made in an ontology-based formalism.
2. All ontology-based descriptions shall be capable of being connected via mediation.

Thus, the SOA-RM concept of visibility**,** especially with its claim to know about the existence of the other party (awareness) plays a vital role. By extending ontologies to describe services in an SOA, a machine can reason about the functions they provide, the mechanism to invoke them and the input and output data. Each service that currently has a syntactic description (i.e., a WSDL document) will also have a semantic description in some formalism. This requirement may be fulfilled by semantic annotations of service capability specifications, e.g. based upon SAWSDL (see section 3.3.3.1), or by full-fledged Semantic Web services frameworks such as WSMO or OWL-S, see section 3.3.2. There are other aspects of extension in the semantics-driven approach. As shown in Figure 3-2, visibility is now seen as a more fundamental concept and has been replaced by the grouping concept of **mediation**. In a similar way, the SSOA-RO groups the description of functionality into a concept **capability**, and the SOA-RM concept of interaction, into a concept **interface**.

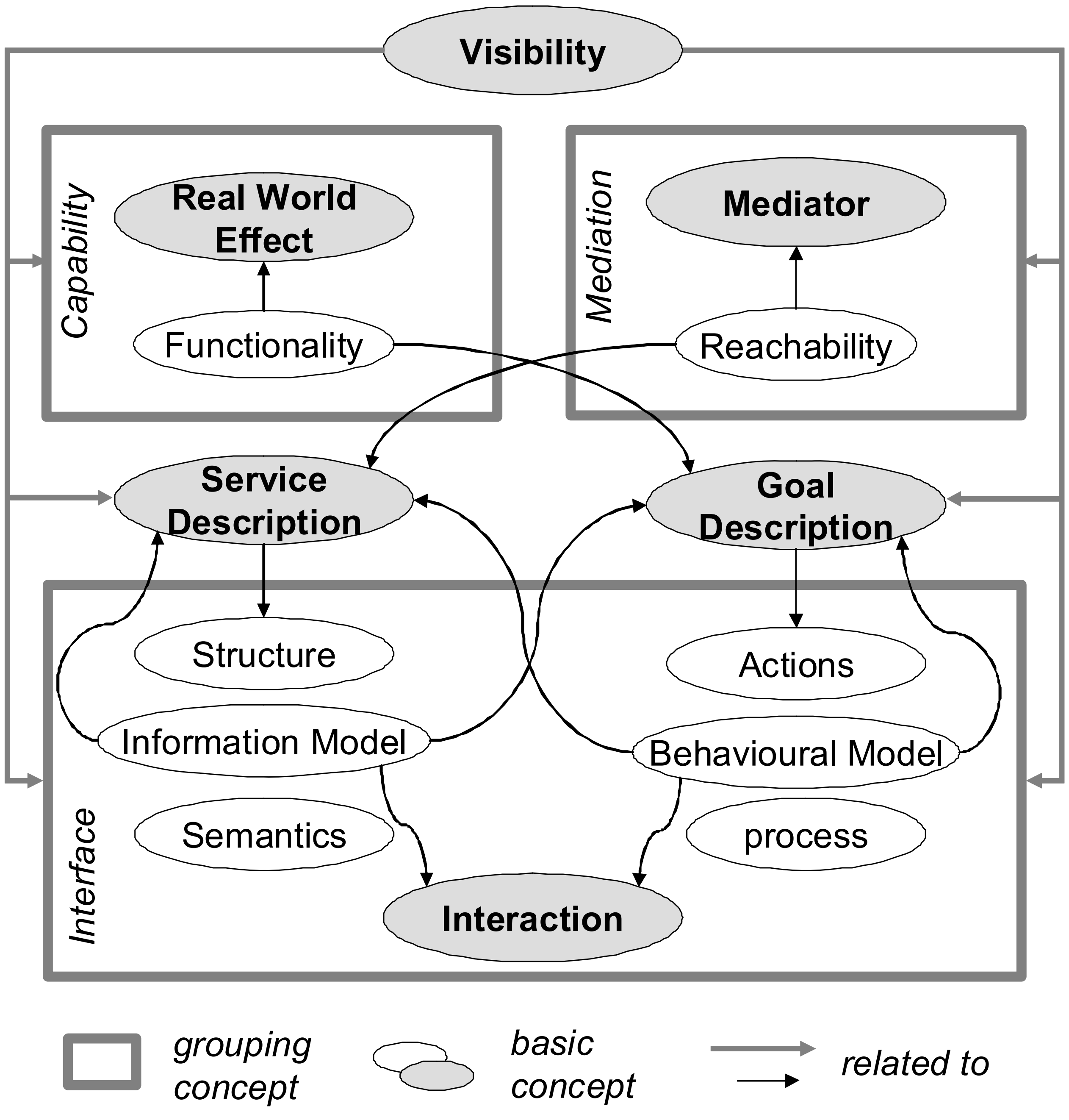


Figure 3-2: Extension of SOA-RM in the OASIS SOA Reference Ontology (OASIS, 2008)

In order to enable automation in the discovery process, the SSOA-RO introduces the concept of a goal. A goal is described by a **goal description** that formally defines the needs of a requester independently of the peculiarities of the infrastructure and the technologies of the service provider.Now, the basic approach of the SSOA-RO is that a goal description is formed from the same elements as a service description: namely a **capability description** and a set of **interfaces**. Thus, as the structure of both is the same and can be directly compared, matching between a goal description and a service description is heavily facilitated. The capability description is described in terms of conditions on the state of the world before and after the execution of the service. The conditions fall into two groups:

1. Conditions related to the state of the information space, i.e. the values of input and output parameters, defined in pre-conditions (state before the service execution) and post-conditions (state after the service execution).
2. Conditions related to the state of the real world, defined in assumptions and (real-world) effects, e.g. the closing of a valve in a chemical process.

The interface, which is defined as "the means for interacting with a service" according to the SOA-RM, is split into two parts:

1. an information model, usually called the signature of a service, here assumed to be based upon an ontological description, and
2. a behavioural model from both the service requester perspective (choreography) and the service's communication with other services (orchestration).

The SSOA-RO aims at automating many of the tasks that previously required human intervention in building and maintaining an SOA-based application. Task examples are service discovery based upon semantic service descriptions, service selection based upon advertized quality of service and data mediation to overcome syntactical and semantic heterogeneity. For instance, so-called SG-Mediators (i.e. service-to-goal mediators) shall connect service descriptions to goal descriptions. This is the basic task of service discovery.

This automation may only happen if there are ontology-based descriptions of all of these SOA-RM concepts, especially the service and goal descriptions. There are several semantically enhanced service platforms that support this approach (e.g. based upon WSMO or OWL-S as introduced in section 3.3.2).

As an ongoing activity, OASIS is about to define a Reference Architecture for Semantic Execution Environments (OASIS, 2009) abstracting from the technological details of service platforms. It specifies interfaces to so-called "broker services" that are collectively termed a Semantic Execution Environment for Semantic Web Services. The idea is that the combination of these broker services provides the platform that enables automation and interoperability of tasks to be performed by service consumers, for example service discovery, ranking, selection, composition and execution. Currently, just an initial draft is available.

3.2.6 SOA Ontology of The Open Group

The Open Group, a vendor-neutral and technology-neutral consortium, has also started to work on a Service-Oriented Architecture Ontology (OpenGroup, 2008). It is formally defined in description logics using the OWL-DL ontology language (see section 3.3). Its primary objective is to enable communications between business and technical people about SOA, promote a mutual understanding about SOA and to facilitate SOA adoption. Thus, this ontology tries to "define concepts, terminology and semantics of SOA in both business and technical terms" which enables its use by architects as meta-data for architectural artefacts, and by architecture methodologists as a component of SOA meta-models. An excerpt of the major concepts is illustrated in Figure 3-3 and explained below:

- A **service** is defined as a kind of a broader concept which is an **activity.**
- An **activity** is a system of actions that are performed by **actors** in response to events. Furthermore, an activity is classified into a governance activity, architecture development activity, design activity and implementation activity.

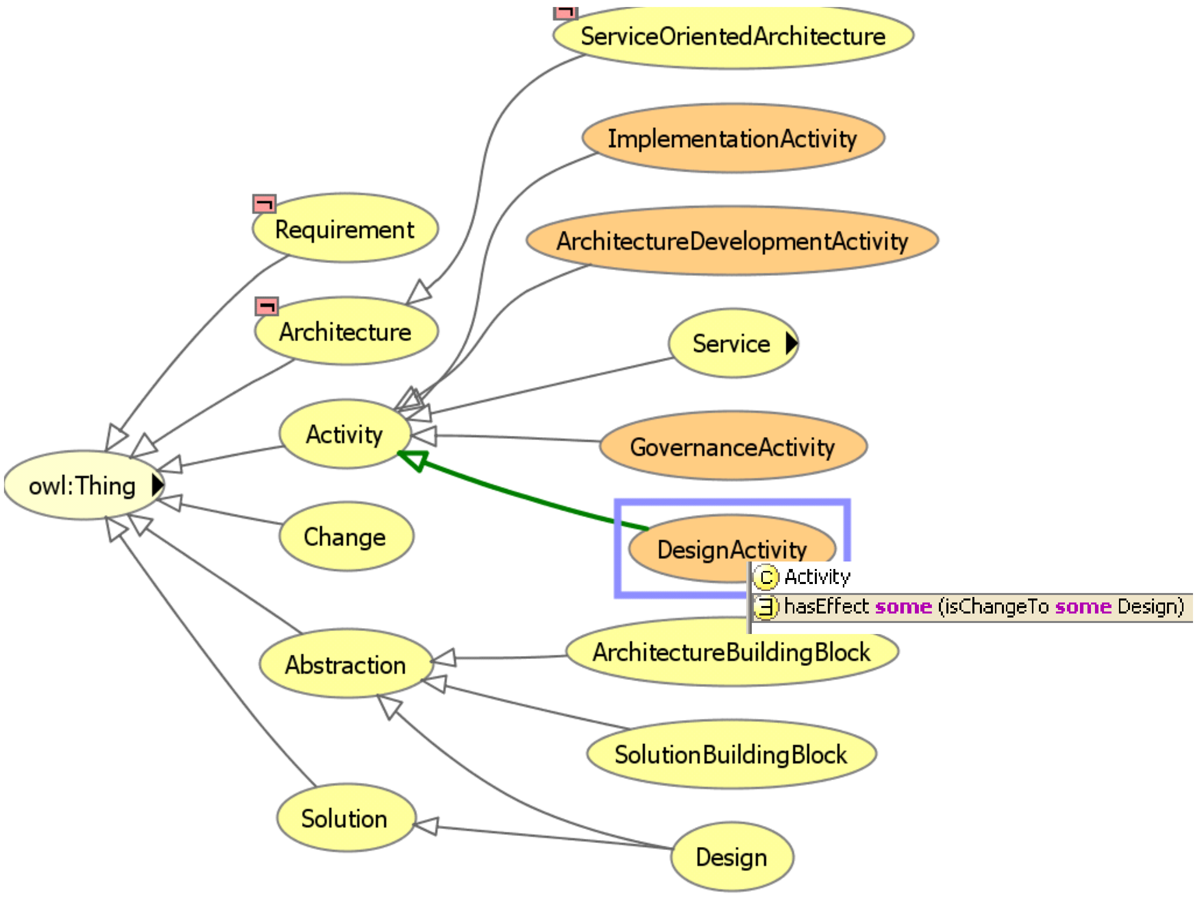


Figure 3-3: Excerpt of the OWL-based SOA Ontology of The Open Group (2008)[14]

[14] Visualisation generated with the OWLviz plug-in to Protégé 3.4.1.

- On the one hand, a **design** is seen as a process of creating an abstract **solution**. Note that such a solution is called "a design", too, whereby a **design activity** creates or modifies a design ("*hasEffect some (isChangeTo some Design)*"). This is consistent to the framework of design research of Hevner et al (2004) as presented in section 2.3 where the term "design" denotes both a process (set of activities) and a product (artefact).
- On the other hand, a design is an abstraction (i.e. an idea of a class of things, e.g. an architecture building block or a solution building block) that meets a **requirement**.
- A **requirement** is a desire that an actor has for something to have some particular characteristic(s). A requirement is satisfied by a **solution**.
- A system has an **architecture** that itself is composed of architecture building blocks. An architecture is instantiated by a **solution**.
- A **Service-oriented Architecture** is a sub-class of **architecture**.

The SOA ontology is still in an unapproved draft version 2.0 and is subject to comments of the SOA community which will probably result in changes to the ontology. For instance, comparing the conceptualization of an architecture with the textual definition of the term "architecture" of ISO/IEC 10746-2:1996 and IEEE 1471-2000 (see definitions 2.2 and 2.3 in section 2.2.1), it seems that it is just restricted to the structural aspects of an architecture and does so far neglect design principles, especially for SOA. Furthermore, although in the associated text document an SOA is defined in terms of architectural styles, this approach is not yet reflected in the formalized ontology.

3.2.7 Assessment: Reference Models

There are various approaches to reference models, mostly initiated and driven by standardization organizations, in order to structure the way of thinking about the key aspects of distributed systems. The ultimate objective is to speak a common language when talking about elements of distributed systems, that is, to use terms with a well-defined meaning in a given community and have the same understanding about their relationships to other terms. Reference models for distributed systems are generic in the sense that they do not prescribe how a distributed system architecture should look like, but they just provide a way of designing and documenting a distributed system architecture. Typically several people and stakeholders with a different expertise, technical background and objectives, look at distributed systems. Thus, reference models provide different abstraction mechanisms and layers in order to enable a separation of concerns.

The ISO RM-ODP is the most generic, architectural style-independent and technology-independent approach how to look at distributed systems. Its kernel idea is the structuring of distributed system specifications in terms of viewpoints, however, it does not impose an order in which the viewpoint specifications have to be drafted.

Thus, according to the definition of a methodology of Avison and Fitzgerald (2003) (section 2.3.2), the RM-ODP itself does not propose a design methodology. However, its structuring approach may be defined in a design methodology as a rule. Thus, the RM-ODP may be applied by a design methodology to satisfy the design requirements R.4 (architectural framework) and R.11 (iteration and documentation support). This is the case in the SERVUS Design Methodology (section 4).

Furthermore, the RM-ODP proposes that there are "dedicated language constructs of RM-ODP viewpoints" and recommends the use of formal description techniques for the specification of the architecture. Although theoretically reasonable due to the complexity of the problem space, these dedicated languages complicate an integrated design methodology across several viewpoints due to unavoidable language gaps and the need for mappings between the languages. Thus, without any further definition, the RM-ODP does not satisfy the design requirements of human comprehension (R.1– R.3).

Having this problem in mind, ISO/IEC 19793:2008 "extends the definition of how ODP systems are specified by defining the use of the Unified Modelling Language for the expression of ODP system specification". However, the UML profiles defined by ISO/IEC 19793:2008 are very complex, and, due to their platform-independent approach, they define terms that are not used in the practice of today's distributed system design. For instance, for the engineering viewpoint, it defines concepts such as capsule, nucleus and cluster that no SOA designer would understand without further explanation. Furthermore, looking at the applicability of ISO/IEC 19793:2008 to the geospatial domain, there are similar concerns. Currently, there is no link between the General Feature Model (GFM) (ISO 19109:2005) and the concepts defined in ISO/IEC 19793:2008 for the specification of the RM-ODP information viewpoint.

As a conclusion we state that the ISO RM-ODP is very useful in structuring the design artefacts of distributed system architectures (R.4), however, its application for the design and engineering of geospatial distributed information systems such as EIS needs further consideration that goes beyond the work carried out in the OGC Reference Model (Percivall (ed.), 2003). This is especially true when these architectures should be based upon a service-oriented paradigm because of two reasons:

1. The OGC Reference Model of 2003 does only contain a meta-model for information (i.e. the General Feature Model of ISO 19109:2005), but no corresponding meta-model for services with conceptual links between both. As the OGC strategy is to provide service specifications on an abstract level (to be specified in UML) and possibly for a multitude of concrete service platforms (e.g. Web services of different kinds), this is a severe limitation and breaches design requirement R.5 (property coverage).
2. Other standardization organizations are proposing SOA reference models that are not using the conceptual model of the ISO RM-ODP. The most advanced example is the SOA-RM of OASIS as described above in section 3.2.4 above. However, the development within OASIS is still quite dynamic. Semantic ex-

tensions of the SOA-RM required a quite fundamental adaption of the SOA-RM conceptual model which may lead to changes in the SOA-RM itself in the future. Thus, the fulfilment of R.7 (semantic enrichment) is problematic as long as there is no consolidated reference model of OASIS including semantic extensions.

Furthermore, OASIS is not the only standardization organization trying to define common architectural frameworks for SOA. The Object Management Group (OMG) has issued in 2006 a request for proposal for a "UML Profile and Metamodel for Services" (OMG, 2006). There is a submission of a joint industry and academic consortium to this request which aims at specifying an **SoaML**, i.e., an **SOA Modelling Language** based upon built-in extension mechanisms of UML such as stereotypes (OMG, 2007). For instance, SoaML defines stereotypes for basic SOA concepts such as participants, service interfaces or service contracts which are partly comparable or synonymous but not identical to the SOA-RM concepts, respectively. At least, in the annex B of the SoaML specification (OMG, 2007) SoaML concepts are mapped to those of the OASIS SOA-RM (OASIS, 2008) and the SOA Ontology of The Open Group (OpenGroup, 2008). The SoaML is declared to be conformant to the OASIS SOA-RM as it provides responses to all SOA-RM conformance guidelines (see section 3.2.4).

Recently, The Open Group, OASIS and OMG have started a trilateral discussion about the relationships between their reference models, reference architectures, maturity models and modelling languages with the aim of finding agreements on core SOA and SOA governance concepts and helping the SOA community to "navigate the myriad of overlapping technical products produced by these organizations" (Kreger and Estefan (eds.), 2009). However, as the competition between these organizations is still ongoing, there is no sound and stable foundation for a service-oriented reference model that could be directly applied to the geospatial domain, e.g. as an extension to the OGC Reference Model, in order to fulfil R.6 (support of standards). This situation has been the starting point and motivation for the edition of the Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer (ed.), 2007) that is used as the architectural framework for the SERVUS Design Methodology.

Furthermore, the following lesson has been learnt in the ORCHESTRA and SANY projects when designing pilot applications according to the RM-ODP approach. The gap is too big between the (sometimes huge amount of) user requirements often only specified in an imprecise textual and informal way and the abstract service platform. This endangers both property coverage (R.5) and traceability (R.8):

- It is very difficult for the system designer to prove that all user requirements are properly covered (uncertainty about the requirements coverage of the design).
- Due to the different specification "languages" used for the requirements and the architecture, the user who has originally stated the requirements cannot be

sure that the architecture and the final system satisfies them (lack of backwards traceability).

It is the major objective of the SERVUS Design Methodology proposed by this thesis to address this issue. Furthermore, the following conceptual ideas of the other reference models were taken up for the SERVUS Design Methodology:

- The OASIS SOA Reference Ontology (OASIS, 2008) argues that a structural coherence in the description of goals (as requested capabilities and interfaces) and services (as offered capabilities and interfaces) is beneficial for the service-goal matching. This coincides with the basic idea of the SERVUS Design Methodology to express both requirements and service capabilities in terms of a common resource model (see section 4.2.4) in order to fulfil design requirement R.3 (user-near analysis).
- In the text document of the SOA ontology of The Open Group (OpenGroup, 2008) the definition of an SOA is founded upon the definition of an architectural style as follows:
  - An SOA is an architectural style that supports service orientation.
  - An architectural style is the combination of distinctive features in which architecture is performed and expressed.

In this thesis, the importance of defining an SOA in terms of a service-oriented architectural style is supported. Even more, we argue that for geospatial SOAs multiple architectural styles shall coexist in a hybrid manner (see the multi-style SOA definition 2.8 in section 2.2).

3.3 Semantic Technologies

One of the cornerstones of this thesis is the investigation how semantic technologies may be embedded into the SOA design process for the benefit of the system analyst as well as the system designer.

In general, semantic technologies provide essential means to work towards Tim Berners-Lee's vision of the next generation of the Web, the so-called Semantic Web (Berners-Lee, Hendler and Lassila, 2001). He proposed to enrich the human-readable data on the Web with machine-readable meta-information (annotations) in order to enable software applications to better process, integrate and interpret information offered through Web resources. A comprehensive introduction and summary to the semantic technologies in the context of service frameworks is given by Fensel et al (2008). We will focus in the following upon the aspects that are relevant for the service-oriented design following the design requirement R.7 (semantic enrichment). These aspects comprise the semantic description of information (section 3.3.1) and services (section 3.3.2), semantic annotation (section 3.3.3), the specifics of geospatial semantics (section 3.3.4) and the application of semantic technologies to software engineering (section 3.3.5).

3.3.1 Semantic Description of Information

Semantic description of information resources is realized by knowledge representation languages. They enable the formal specification of ontologies. **Ontologies** are conceptual models that define concepts and their relations, together with constraints on those objects and relations (Alexiev et al, 2005). However, there is the important additional intention that these conceptual models represent shared knowledge, i.e. they represent a common understanding (consensus) of the discourse of the universe between the parties involved.

The expressiveness of the knowledge representation language determines the classification of ontologies leading to the following ontology spectrum (McGuinness, 2003; Alexiev et al, 2005):

- **Controlled vocabularies**: restricted to a controlled list of terms. Example: DublinCore[15] used for specifying meta-data terms of documents such as "Author", "Title" or "Publisher".
- **Thesaurus**: additional provision of relations between terms, e.g. synonyms and hypernyms, for instance, WordNet® providing all English language words[16].
- **Taxonomies**: additional support of inheritance, i.e. generalization and specialization relations.
- **Frames:** support of class properties inherited by subclasses. Example: ontologies specified in the RDF Schema (RDFS) (Brickley and Guha (eds.), 2004). RDFS provides a vocabulary for defining classes, class hierarchies, properties, property hierarchies and property restrictions. This vocabulary is used for the specification of so-called light-weight ontologies. RDFS is an extension of the Resource Description Framework (RDF) (Klyne and Caroll (eds.), 2004) that enables metadata specifications in the form of „subject–predicate–object" graphs with reification possibilities (i.e. statements about statements are possible).
- Ontologies with **value restrictions** of properties (e.g. by a data type).
- Ontologies with **general logic constraints** (e.g. by logical or mathematical formula)
- Ontologies with **expressive language constraints** (e.g. disjoint classes, inverse properties or part-whole relationships).

Most ontology languages with logic constraints are based upon first-order predicate logic (FOL). FOL supports the universal quantifier (∀) and the existential quantifier (∃), Boolean operators, variables, predicates, and rules for putting them together in formulas (Sowa, 2007).

[15] http://dublincore.org

[16] http://www.cogsci.princeton.edu/~wn

However, one essential practical criterion for the selection of an ontology language is its decidability. Decidability refers to the question whether reasoning algorithms that are applied to ontologies terminate, i.e., whether new knowledge may be generated out of existing knowledge in reasonable time. Decidability is achieved by restricting the application of FOL language elements, for instance, to those that are supported by description logics (Baader et al, 2007). An example of such a restriction is the rigorous distinction between classes and instances that is enforced when using description logics.

The W3C recommends OWL (Web Ontology Language) as a family of ontology languages. (Bechhofer et al, 2004). In order to achieve decidability, it is mostly applied in its restricted form called OWL-DL which is the decidable subset of OWL reduced to description logic elements.

3.3.2 Semantic Description of Services

Kuhn (2004) claims that the semantic definition of services in terms of their interfaces (signatures) is essential in order to support semantic interoperability. Usually, syntactic service specifications (e.g. based upon WSDL) do not include semantics. Thus, two service descriptions can have similar descriptions but totally different meanings, and vice-versa, similar meanings with totally different descriptions (Fensel et al, 2007).

There are basically two ways of achieving semantic service descriptions:

1. The semantic annotation of existing (Web) service frameworks (section 3.3.3).
2. The definition of (new) service frameworks as an ontology and the development of corresponding ontology-based design and run-time environments.

In the last years, enormous research efforts have been put into the second approach (see below), however, up to now there is no unified standard for a Semantic Web Services framework. The following two proposals were competitively submitted to the W3C Semantic Web Services Interest Group[17] by the corresponding research communities:

- **OWL-S** (Semantic Markup for Web Services) described in (Martin et al, 2004a) and submitted to W3C as (Martin et al, 2004b).
- Web Service Modeling Ontology (**WSMO**) described in (Fensel et al, 2007) and submitted to W3C as (Lausen, Polleres and Roman (eds.), 2005).

OWL-S is a service ontology specified in OWL (see section 3.3.1 above). It basically specifies a "service" as the kernel ontological concept that is provided by "resources", presented by a "service profile" and described by a "service model". The main use of the service model concept is to enable invocation, enactment, composition, monitoring and recovery of services. Furthermore, a service supports a "service grounding"

[17] http://www.w3.org/2002/ws/swsig/

that describes the mapping to detailed (Web) service specifications provided, for instance, in WSDL.

WSMO follows a more comprehensive approach that also includes built-in goal-driven discovery and mediation support. It identifies four top-level elements: **Ontologies** provide the terminology used for the other elements, **Web services** represent the computational elements, **goals** describe aspects related to user desires with respect to the requested functionality, and **mediators** describe elements that handle interoperability problems, e.g. between services on protocol level, or between the users' goals and the Web services.

A more detailed comparison between OWL-S and WSMO is given in Fensel et al (2007). Although one of these two approaches may be the technology of choice for semantic (Web) services in the long run, this thesis just uses semantic annotation applied to services (see section 3.3.3) because of two reasons:

1. It enables to rely upon the capabilities of existing service platforms (see design requirement R.12).
2. It is currently the only W3C standard recommendation (see design requirement R.6).

3.3.3 Semantic Annotation

3.3.3.1 SAWSDL

OGC Web Service standards typically use XML schemas to describe the format of ingoing and outgoing messages. Semantic annotation of geospatial operations shall therefore be realized by XML mechanisms. The document "Semantic Annotations for WSDL and XML Schema" (SAWSDL) (Farrell and Lausen (eds.), 2007) is the recommendation of the W3C on how semantic descriptions of XML-based Web Services should be performed. SAWSDL defines a set of extension attributes for the XML Schema definition language. This mechanism may especially be applied to the Web Services Description Language (WSDL)[18] which is the XML language recommended by the W3C for the description of Web services (W3C, 2007a). It provides specification of essential Web service components such as operations, their grouping into interfaces, the structure of related input and output messages as well as their mapping (binding) to an underlying transport protocol.

SAWSDL allows the description of additional semantics of WSDL components. The approach is to annotate the WSDL document components by referring from individual component elements to concepts in semantic models, e.g. ontologies (Figure 3-4). SAWSDL does not specify a language for representing the semantic models themselves. Instead, the annotation mechanism is independent of the ontology expression language and does not enforce a particular ontology language such as OWL. Instead it

[18] Especially WSDL2.0 but basically also applicable to WSDL1.1

provides mechanisms by which concepts of the semantic models, typically defined outside the WSDL document, can be referenced from within WSDL and XML Schema components using annotations. Examples are references to categorization information of service taxonomies that can be used when publishing a service in a service registry (Karakostas and Zorgios, 2008).

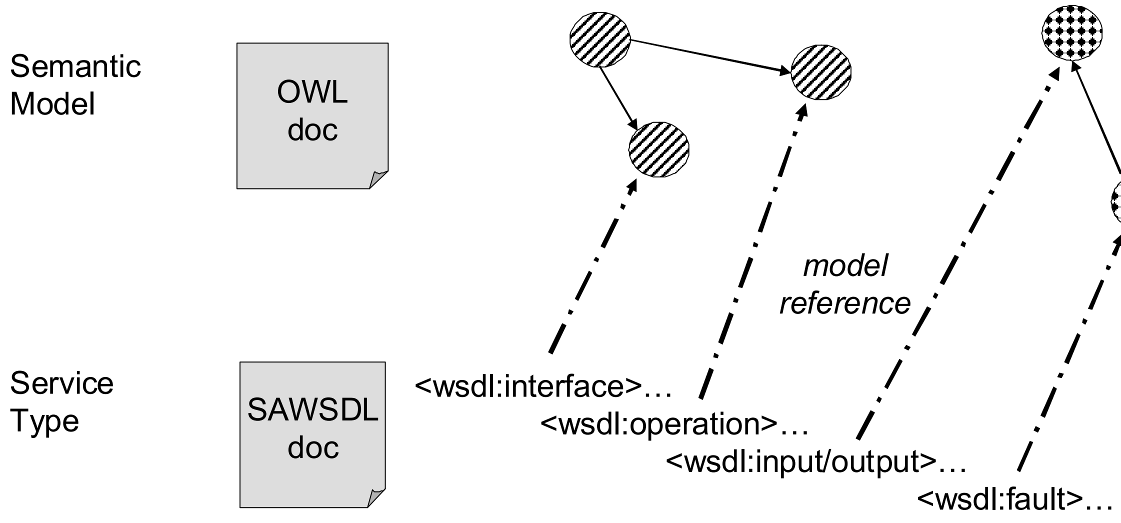


Figure 3-4: SAWSDL Model References

SAWSDL uses the following terminology:

- **Semantic Model**: A semantic model is a set of machine-interpretable representations used to model an area of knowledge or some part of the world, including software. Examples of such models are ontologies (e.g. but not necessarily specified in OWL) that embody some community agreement and logic-based representations. Depending upon the framework or language used for modelling, different terminologies exist for denoting the building blocks of semantic models.
- **Concept**: A concept is an element of a semantic model. This specification makes no assumptions about the nature of concepts, except that they must be identifiable by URIs.
- **Semantic Annotation**: A semantic annotation in a document is additional information that identifies or defines a concept in a semantic model in order to describe part of that document. In SAWSDL, semantic annotations are XML attributes added to a WSDL or associated XML Schema document, at the XML element they describe. Semantic annotations are of two kinds: explicit identifiers of concepts, or identifiers of mappings from WSDL to concepts or vice versa.

The basic approach of SAWSDL is to add an extension attribute, named *modelReference*, to specify the association between a WSDL component and a concept in some semantic model. Important WSDL components are interfaces, operations, types and faults. However, SAWSDL is defined such that the *modelReference* extension attribute may be applied as an XML fragment to any other XML schema.

The value of a *modelReference* extension attribute is a set of zero or more URIs that identifies concepts in a semantic model. Thus, multiple semantic annotations may be associated with a single WSDL element, also pointing to different semantic models.

Additionally, SAWSDL enables the annotation of schema components (simple or complex types, elements) with a so-called *liftingSchemaMapping* and/or *loweringSchemaMapping* attribute. These are used to associate a schema type or element with a mapping (via a mapping language, e.g. the XSLT) to an ontology or vice versa. This enables a client to transform XML instance data into semantic data of an ontology and/or generate concrete XML components (documents, sub-elements,…) from semantic data in an ontology.

The primary application field of SAWSDL is the dynamic discovery (i.e. discovery at run-time of the services), composition and invocation of Web services. Two application examples that are of concern for this thesis are service categorization and service matching. Referenced concepts, especially from interfaces, may be interpreted as categorization information that can be used to publish a Web service in a registry (e.g. an UDDI registry or an OGC Catalogue) and to group it there according to this category. Service matching based on SAWSDL is described in section 3.4.4.1.

3.3.3.2 Semantic Annotation of Geospatial Resources

Maué (ed.) (2009) discusses the potential of annotating geospatial resources, in particular OGC Web services, in detail. He distinguishes semantic annotation at three levels as illustrated in Figure 3-5:

1. Service Metadata Level: Each OGC Web service provides a "capabilities" document that tells the user how to access and invoke the service, as well as some resource metadata with information about the service provider, licensing, a title and description, or a keyword section.
2. Data Model Level: Each OGC Web service provides an additional (XML schema) document that represents the data types used for the parameters of each service operation. Both documents, the metadata and the schema, are describing the underlying data, and are therefore explicitly linked (highlighted by the orange arrow in Figure 3-5).
3. Level of the Actual Data Entities: OGC Web service implementations are relying on data entities, encoded in the format predefined in the data model of level 2.

Semantic annotation at these three levels is realized by references to knowledge models (the numbered arrows in Figure 3-5). One possible technology for the semantic annotation investigated is SAWSDL as described above in section 3.3.3.1. Figure 3-5 shows an OGC Web Feature Service (WFS) based on an underlying database that serves quarry features. At level 1 a keyword within the capabilities document can be directly linked to the corresponding concept in the domain ontology. At level 2 the data model, in this case the application schema of the WFS, is linked to concepts of an application ontology dedicated to this particular WFS. This means that this application ontology provides semantic definitions of the feature types (such as *exploitation*) and attribute types (such as *allowedproduction*) of the WFS application schema. At level 3 semantic annotations are applied directly to the feature instances in the database.

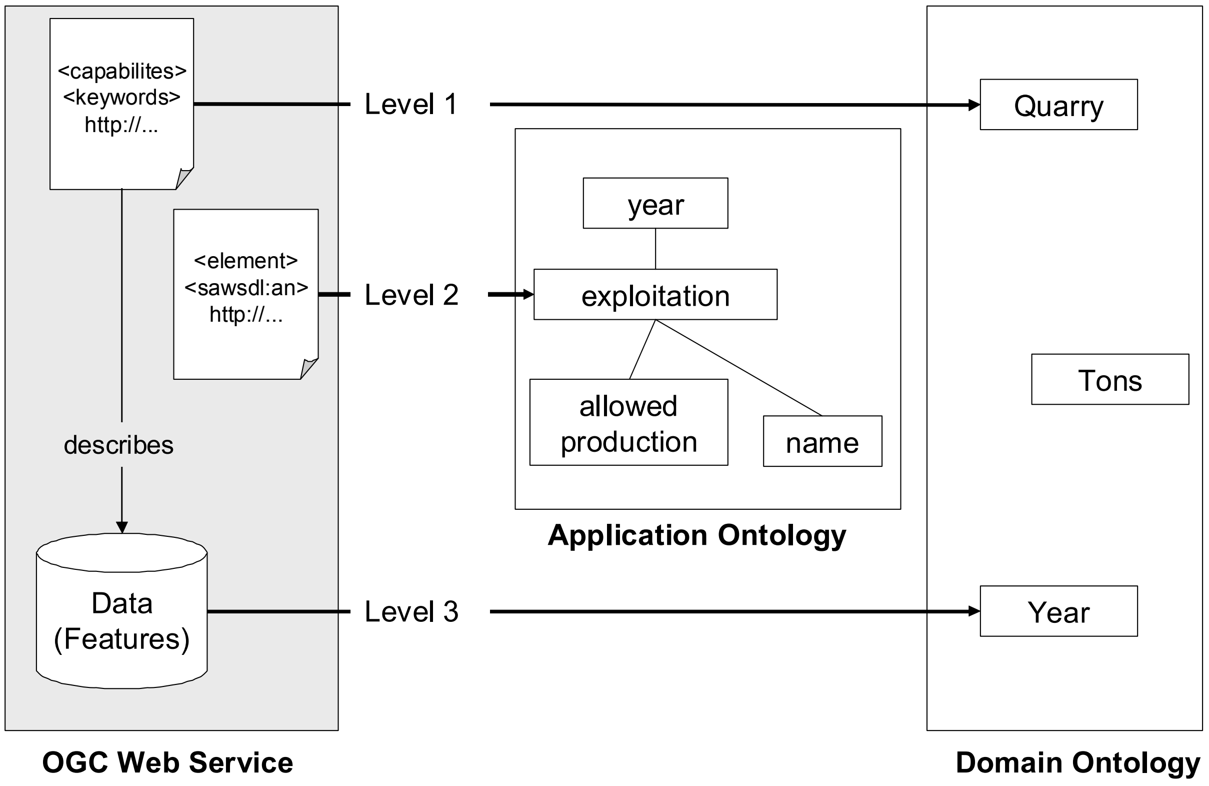


Figure 3-5: Semantic annotations of OGC Web Services at three different levels (Maué (ed.), 2009)

Each type of annotation has different implications on the discoverability of the Web service, and the possibility of the user to evaluate if the served data satisfies his needs. Furthermore, they differ in their potential of applying reasoning capabilities, i.e. the abilities to infer either new knowledge or to detect conflicts in existing knowledge.

Maué (ed.) (2009) concludes with a first evaluation of the benefits and limitations as well as the possible applications of semantic annotations at these three levels. For the purpose of geospatial SOA design the following results for "service discovery" are important:

- Service discovery may already benefit from level 1 annotation as there is a much better recall then no annotation at all (due to semantic-enabled query processing). Furthermore, no modification of the underlying data model of the service is necessary.
- Additional annotation on level 2 increases both recall and precision since the inner structure of the data model may be exploited. However, the effort is high as the semantic annotation of complex data models is considered to be quite tedious and requires additional documentation.

- Annotation on level 3 is considered a good solution for data without explicit data models. However, the effort may be enormous in case of high data volumes.
- As a drawback for both level 1 and 2 annotation, they state that semantically-enabled discovery of resources requires specialized interfaces that allow users to select the needed concepts.

In order to exploit these potentials, Maué (ed.) (2009) request a "harmonized best practices (…) as a good step forward to offer good semantic annotation capabilities throughout the OGC stack".

3.3.4 Geospatial Semantics

Semantics has become one of the most prominent research themes within the domain of geographic information science and is often referred to as Geospatial Semantics (Rodriguez et al, 2005) or the Semantic Geospatial Web (Egenhofer, 2002). Research work initially focused on (geo-)ontologies and semantic similarity, e.g. the question about how relations between geospatial objects (like "within" or "next to") can be semantically expressed with the peculiarity that such location properties may be vague and uncertain (Hart and Dolbear, 2006). Recent research work has focused on ontology-based spatial information retrieval (Lutz and Klien, 2006) as well as semantic annotation of geographic information (Klien, 2008) and geospatial services (Maué (ed.), 2009) in order to facilitate their discovery and use.

There are multiple experimental approaches dealing with the semantic discovery of geospatial resources, among which are the following:

1) Hilbring and Usländer (2006) propose to support the formulation of a query to a meta-information store (e.g. an OGC catalogue) by ontologies. This ontology-based resource discovery approach, which is implemented as a so-called Semantic Catalogue (SemCat), focuses on the client side and enables the use of standard OGC catalogue services and application profiles on the server side. The SemCat component is used in the implementation architecture of SERVUS (section 7).
2) Stock (2009) defines a semantic meta-information model as an alternate application profile of an OGC catalogue service. This approach extends the server side and enables the use of standard OGC catalogue client applications.
3) Janowicz (2006) proposes the use of Semantic Web Service frameworks (e.g. WSMO, see section 3.3.2) for the improvement of the discovery process.

The OGC has launched in 2006 a working group on geosemantics with the mission "to establish an interoperable and actionable semantic framework for both representing the geospatial knowledge domains of information communities and mediating between them" (Lieberman, 2007).

As of today, the most advanced submission to OGC is the discussion paper about semantic annotation in OGC standards (Maué (ed.), 2009). However, there are up to now no agreed OGC standards dedicated to geospatial semantics.

3.3.5 Semantic Technologies in Software Engineering

Dobson and Sawyer (2006) state that "there are clear overlaps between what an ontology engineer aims to achieve in the modelling of a domain and the modelling that a requirements engineer will perform during the requirements process." The reason is that "ontologies offer one possibility for representing, organising and reasoning over the complex sets of knowledge that requirement documents embody." There has been a long scientific history dating back to 1984 in applying knowledge representation techniques to the requirements analysis phase, however, these approaches did not find their way into the practice apart from the use of formal techniques (such as UML use cases) to structure and document user requirements. Now, with the growing availability of the Semantic Web technologies (section 3.3.1) there is a renewed interest in the application of semantic technologies to information systems engineering, or software engineering in general.

The applicability of semantic technologies for the engineering of software systems is discussed in the Semantic Web Best Practices and Deployment Working Group (SWBPD) of the W3C. The technical report (W3C-SWBPD 2006) discusses the usage of ontology languages (in particular OWL and RDFS) for the design of software architectures.

One particular aspect is the systematic handling of the non-functional requirements, especially the qualitative requirements. In a distinguished activity of the SeCSE project (section 3.4.2.2) Dobson et al (2007) scrutinized whether an ontological approach could be applied here. They argued that an ontology that is dedicated to non-functional requirements should be generic, i.e. independent of the application domain, and relatively small. From a taxonomical point of view, their approach resulted in two hierarchies representing attributes and metrics of quality of service (QoS), respectively. Both are joined by an OWL object property that indicates that a QoS attribute has a relation to QoS metric (Dobson, 2005).

3.3.6 Assessment: Semantic Technologies

The application of semantic technologies to support design processes, especially in the requirements engineering phase, is an emerging research topic (Dobson and Sawyer, 2006). It may contribute to fulfil the design requirement of user-near analysis (R.3) and to strengthen the analytical power by approaches for semantic enrichment (R.7) such as SAWSDL (Farrell and Lausen (eds.), 2007) and property coverage (R.5) such as quality of service ontologies (Dobson, 2005). Full-fledged frameworks for Semantic Web Services exist (section 3.3.2), however, beyond prototypical applications in research projects, they have not yet gained acceptance in the geospatial community which is a pre-requisite to satisfy design requirement R.6 (standards compliance). As of today, the only standardisation activity in the OGC is the proposal of how SAWSDL may be used in the context of OGC Web services (Maué (ed.), 2009).

A systematic migration path from geospatial standards as specified by the OGC to the inclusion of semantic technologies is not easily possible. On the architectural level the OGC Reference Model, both in its original version of 2003 (Percivall (ed.) 2003) and in its updated version of 2008 (Percivall (ed.) 2008), does not support any semantic technologies at the moment. As one of its cornerstones, especially in its original version of 2003, the OGC Reference Model defines a conceptual model that just provides guidance and rules for application schemas specified in UML (Rumbaugh et al, 1998). Even when these information models may be considered as "domain ontologies", their expressiveness is strongly limited by the capabilities of UML. Thus, established Semantic Web technologies (e.g., reasoning tools) may not be applied in a seamless manner. A further barrier to using the OGC Reference Model in the context of Semantic Web technologies is the rule to support ISO 19115:2003 and ISO 19119:2005 as meta-Information schemata for information and services, respectively. Thus, references to ontological concepts from meta-information elements or the semantic annotation of OGC Web services are not in line with the conceptual approach when applied rigorously.

On the level of the architectural framework, some of these deficits were covered and removed by the conceptual model of the Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer (ed.), 2007) as part of this thesis. The RM-OA aims at satisfying the design requirements of analytical power, especially R.4 (architectural framework), R.5 (property coverage) and R.6 (standards compliance). Furthermore, it enables R.7 (semantic enrichment) by providing a flexible meta-information model that may be specified according to the purpose of a dedicated application or application component. This approach paves the way towards the inclusion of semantic concepts.

3.4 Service-oriented Analysis and Design

3.4.1 Overview

Since today, numerous methodologies for Service-oriented Analysis and Design (SOAD) have been proposed in the literature. An analysis and review of existing methodologies is contained in Ricken and Petit (2008) and Kohlborn et al (2009). For this thesis, only those methodologies are presented in more detail whose characteristics contribute to the requirements for service-oriented design listed in section 1.4 . We classify them into two major categories:

1. MDA-to-SOA: Extension and/or adaptation of model-driven architecture (MDA) based methodologies (see section 2.3.4) to service-oriented environments.

2. BPM-to-SOA: Business-Process Oriented approaches that start with the modelling of business processes in dedicated enterprise-level BPM languages and maps them to the elements of the service layer.

Figure 3-6 illustrates the basic schemas of these two categories in the context of the abstraction layers of Bieberstein et al (2006):

1. The **Enterprise Layer** focuses on the way an enterprise operates in a particular industry. Artefacts are business models that comprise core and supporting competences of an enterprise. The role of business models in the Enterprise Layer is emphasized in various research works, e.g. Johannesson (2007) or Bercovici, Fournier and Wecker (2008).

   Note: The term "enterprise" also encompasses governmental organizations (e.g. environmental or civil security agencies) or non-governmental organizations (e.g. interest groups, rescue organizations).
2. The **Process Layer** models business processes that are incorporated into enterprise business models. Each process is unique in its handling of one major functional area of the business and may be decomposed into sub-processes as required.
3. The **Service Layer** models individual business functions as services. It provides a conceptual bridge between the business aspects tackled in the Enterprise and Process Layer (which are in the focus of the business analyst and result in the modelling of business services) and the technological and implementation aspects (which are in the focus of the IT specialist and result in the modelling of IT services). Note that business services may be realized as a composition or orchestration of IT services.
4. The **Component Layer** models software components that comprise implementation building blocks of one or more technical services. These technical services may either be a result of the top-down modelling from the higher layers, but may also be candidate services as a result from the bottom-up analysis of existing application systems.
5. The **Object Layer** models business objects, their attributes, and behaviours needed for each of the business functions required, as well as programming objects in order to implement services in an object-oriented programming environment.

Note that there is a gray area between the process and the service layer. Business functions may be modelled as processes and later implemented as services (Bieberstein et al, 2006), typically as **composite services**, i.e. services whose implementation is based upon the orchestration or choreography of other services following the SOA design principle of Service Composability (see section 2.2.2).

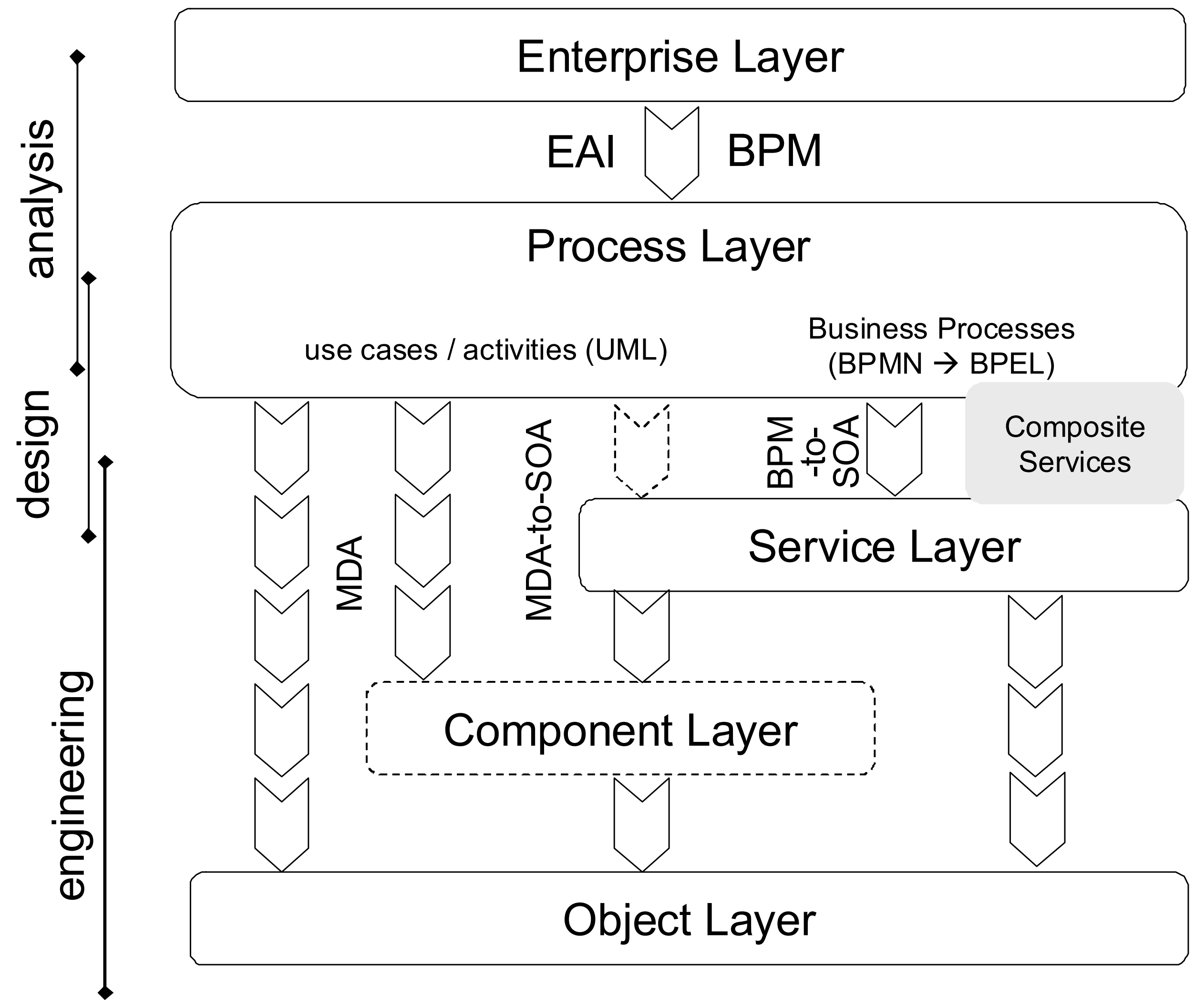


Figure 3-6: Basic Categorization of SOAD Methodologies

3.4.2 MDA-to-SOA Methodologies

MDA-to-SOA methodologies are typically based upon UML and conceived for object-oriented environments and their development methodologies such as object-oriented analysis and design (OOAD) and component-based developments (CBD) (Koskela et al, 2007).

As stated by Chang and Kim (2007), OOAD and CBD need to be enhanced with additional facilities such as service modelling, service interface design, and composition. However, their proposal is restricted to a keyword-based matching technique for Web services based on a UDDI registry and lacks a "semantic-based matching using the semantic description of unit service specification" which is important to overcome different semantic understandings of keywords describing requested and offered services.

Karakostas and Zorgios (2008) propose a model-driven approach for the engineering of service-oriented systems that uses the modelling language IDEF (Integrated Definition Methods) as the starting point for the modelling of the enterprise and process layer. However, as IDEF is domiciled in the manufacturing domain (Noran, 2003) and did not reach wide prominence in the geospatial domain compared to UML, this methodology is not presented in more detail.

IBM offers a methodology and a product suite for a service-oriented modelling and architecture (SOMA) (Arsanjani et al, 2008), whereas the European SeCSE project has developed service discovery concepts and tools for the analysis and design phases (Di Penta et al, 2009). Both approaches are based upon UML use cases and/or sequence diagrams and tailored to corresponding UML products. SOMA is described in more detail in section 3.4.2.1 whereas the SeCSE results that are of relevance for this thesis are presented in section 3.4.2.2.

3.4.2.1 IBM Methodology for Service-oriented Modelling and Architecture (SOMA)

IBM has developed a methodology called Service-oriented Modelling and Architecture (SOMA) to effectively analyse, design, implement, and deploy SOA projects (Arsanjani et al, 2008). SOMA covers the complete life-cycle of a software development for designing and building SOA-based solutions. According to IBM, it is derived from "hundreds of successful experiences and lessons learned from the difficulties and challenges encountered in early SOA design and implementation projects" in multiple industries. The SOMA methodology includes seven major phases of a service-oriented software development, however, they are to be applied in a so-called "fractal" manner, i.e., they "contain capabilities that can be leveraged as needed in different sequences", no rigid sequencing is imposed. The seven phases including their activities are:

1. **Business modelling and transformation**: define business architecture and models, discuss SOA maturity, and draft transformation roadmaps.
2. **Solution management:** initiate project management, select (possibly hybrid) solution templates and patterns, conduct method adoption workshops, and customize method.
3. **Identification**: identify (candidate) services, components, flows and information by goal-service modelling which includes the decomposition of domains, the analysis of existing assets and the re-factoring and rationalising of services.
4. **Specification:** specify services, analyse subsystems, specify components, flows and information, and re-factor and rationalize services
5. **Realization**: refine and detail components, establish realization decisions, perform technical feasibility exploration, and detail SOA solution stack layers.
6. **Implementation:** construct, generate and assemble services, execute unit, integration and system tests.
7. **Deployment**: deploy services, execute user-acceptance test, monitor and manage processes and performance.

There are several interesting aspects of SOMA that are relevant for this thesis:

1. SOMA thinks in terms of several releases or versions of an SOA solution, i.e. the solution is developed in an iterative manner with well-defined milestones.
2. SOMA starts the service identification from the definition of business goals (or simply goals) instead of use cases that are used in OOAD to "capture functional requirements". They argue that use cases are too static in the sense that they are often realized as hardcoded object interactions. Instead, a service-oriented solution should focus on "a set of business-aligned IT services that collectively enable the fulfilment of business goals" whereby the IT services "can be recombined in unanticipated service contexts."
3. In the identification phase, only candidate services are identified as not all business capabilities and functionalities will be exposed as IT services in the end or for a planned release.
4. Several service identification techniques may be combined. The goal-service modelling combines top-down and bottom-up approaches and "pulls them together into alignment". It requires that all services should be traceable back to a business goal.
5. Re-factoring and rationalising of services may happen several times, at least both in the identification and specification phases. The SOMA method explicitly foresees review and re-evaluation activities of the service composition, service grouping by functional areas and assessment of the service relevance, even together with the business stakeholders (rationalization).
6. Flows of messages and information models are tackled as design artefacts that are equivalent to the services. The elaboration of a conceptual data model into a logical data model is performed in the specification phase.
7. Non-functional requirements affecting the quality of service (e.g. cost of transactions, performance, availability, and security) are explicitly handled as part of the service-case (instead of use-case, see above) in the specification phase.

The SOMA methodology has been published by IBM up to a certain degree of details in Arsanjani et al (2008), however, an effective use in an SOA project requires a dedicated modelling and development environment in order to manage the complexity and comprehensibility of the SOMA method and to leverage its potential. IBM has addressed this need by offering SOMA-ME, the SOMA modelling environment (Zhang et al, 2008). SOMA-ME is embedded into the IBM Rational Software Architect and follows a model-driven approach based upon a UML profile that provides a meta-model for the representation of the SOA solution artefacts identified and described in the SOMA methods, e.g. candidate services or existing assets. SOMA-ME was implemented as a framework with plug-in capabilities such that other modules (e.g. for service discovery and composition) may be embedded.

Bercovici, Fournier and Wecker (2008) describe a model-driven business architecture design tool that produces meaningful artefacts related to SOMA. It comprises a tabular and graphical editor based upon XML Schema Definitions. In their conclusion, they claim to increase the design efficiency by having shortened the "total lead time of the complete solution provided", however, on the backside, there seem to an insufficient degree of service reusability as they propose "smart searches connected to the model for the management of the artefacts to enrich the level of reusability and repeatability achieved by the tool."

3.4.2.2 Service Engineering – The European Integrated Project SeCSE

The main mission of the European Integrated Project SeCSE (Service Centric System Engineering) is to create new methods, tools and techniques for requirements analysts, system integrators and service providers that support the cost-effective development and use of dependable services and service-centric applications in the European automotive and telecommunication sectors (Di Penta et al, 2009). The focus is on service engineering defined as the "activity of specifying, designing, implementing and maintaining services offered by a service provider" (Sawyer, 2005).

One specific SeCSE research topic is the improvement of the requirements analysis activity by both the removal of ambiguities and the completion of specified requirements. SeCSE proposes to exploit the described characteristics of discovered service to make the requirements specifications more complete (Zachos et al, 2007). First, use cases specified in natural structured English are reduced to their atomic terms by natural language pre-processing algorithms. These atomic terms are the basis for the generation of queries for service discovery. However, before issued to the SeCSE service registry, two steps are applied through the support of the WordNet®, a large lexical database of English[19]:

1. Query expansion: Terms are added in the query that have the same or similar meaning to existing query terms, to make the query more complete.
2. Term disambiguation: The meaning, or sense of each term in the query is added in order to make the query unambiguous.

The algorithm to match the query with the description of the services in the registry is based on a vector-space model. The discovered services are then presented to the users with the aim to evaluate their applicability to the problem to be solved. The idea is to complete the list of originally specified requirements, either through novel requirements motivated by the discovered services or to refine existing requirements. The approach has been tested in different workshops with experts of the automotive and telecommunication sectors with "encouraging results" as claimed by Zachos et al (2007).

Query expansion and term disambiguation are also important concepts in the SERVUS Design Methodology as described in section 4. Query expansion is applied to geospatial queries to a geospatial OGC-compliant catalogue in the implementation

[19] http://www.cogsci.princeton.edu/~wn

architecture of the SERVUS Design Methodology by the integration of the Semantic Catalogue (Hilbring and Usländer, 2008). Term disambiguation is achieved by the rephrasing of user requirements into a semantically-annotated requested resource.

3.4.3 BPM-to-SOA Methodologies

The mapping of formally specified business processes to an SOA-based infrastructure including an automation tool support is investigated in various approaches, both in research, e.g. Emig et al (2007), Ricken (2007) or Tan et al (2009) for the support of scientific workflows, and in industry-driven standardization organizations, e.g. OMG's Business Process Definition MetaModel (OMG, 2008b) or the OASIS Web Services Business Process Execution Language BPEL (OASIS, 2007).

Emig, Weisser and Abeck (2006) argue that "business process oriented programming (…) is the next step in the evolution of software engineering", following and complementing structured, modular, object-oriented and component-oriented programming. Business process oriented programming focuses on the analysis phase "where business processes of the future user of the software system have to be analysed". It provides executable process descriptions (in section 2.1.1 called "business services") which are based upon service-oriented elements, typically Web services (in section 2.1.1 called "IT services"). This methodology, called "programming in the large" by Leymann (2003), starts with the identification of business processes and their specification in the Business Process Model Notation (BPMN), which is graphical notation for business processes and has been standardized by the Business process Management Initiative. At the transition to the design phase, BPMN-specified business processes may then be automatically mapped to the Business Process Execution Language (BPEL) as standardized by OASIS. BPEL is an XML-based language and allows the designer to specify the orchestration of IT services in service chains. Various "BPEL engines" exist that support the parsing of BPEL code and thus may serve as run-time environments for BPEL-specified service chains.

Emig, Weisser and Abeck (2006) describe how such a "programming in the large" may be practically applied in the software analysis and design phase. They argue that this approach has significant benefits over the traditional approach based upon UML where use cases (as part of business processes) may be specified in UML use case diagrams and refined in UML activity diagrams. However, the transition to the design phase, i.e. the mapping to software and service components has to be done manually as there is not yet a standardized mapping of UML business process or service orchestration profiles to business process execution languages such as BPEL. An emerging alternative would be to use BPMN and integrate it into an UML design environment by the usage of OMG's Business Process Definition Metamodel (BPDM) (OMG 2008b). BPDM provides the "capability to represent and model business processes independent of notation or methodology" and thus enables the exchange of user models between different modelling tools. It is based upon the OMG MOF standard and supports both orchestration and choreography as fundamental process views.

Emig et al (2007) proposes a UML service meta-model that supports an "explicit design of atomic and composite services along with the dependencies between them". Their final objective is to integrate the specification of business processes (e.g. using BPDM) into this service model in order to enable a uniform model-driven development of an SOA infrastructure. Furthermore, the service meta-model comprises the possibility to specify deployment information as part of a service modelling, thus preparing the identification and reuse of runtime services as part of the design activity.

3.4.4 Service Matching

The problem of finding matches between requested and offered services has to be looked at when thinking about a design methodology in service-oriented environments. Trastour et al (2001) and Lutz (2007) define service discovery essentially as "finding a match between descriptions of service capabilities (i.e. of what the service provides) and user requirements (i.e. what is needed to solve a given problem)". This discovery and matching problem is an inherent problem of Information Retrieval (Baeza-Yates and Ribeiro-Neto, 1999). Nowadays, it is intensively re-discussed and researched in the context of the Semantic Web, and especially Semantic Web Services with a focus on the improvement of the effectiveness of the matching process rather than on the efficiency of the whole retrieval process (ISWC, 2007).

Note: In the literature the terms "matching" and "matchmaking" are used dependent on the technological context in which they are used. Euzenat and Shvaiko (2007) define matching in the context of "ontology matching" as "finding correspondences between semantically related entities in different ontologies". The term "matchmaking" is often used if the acting roles or the conceptual approach used in the discovery and matching process need to be emphasized. For instance, in the literature on agent systems, matchmaking is defined as "mediating among requesters and providers of services for some mutually beneficial cooperation" (Sycara et al, 2002). For the following discussion of the state of the art of the existing conceptual approaches we use the more general term "service matching".

3.4.4.1 Service Matching based upon Semantic Service Descriptions

Apart from the application of description logic reasoning, it has to be investigated which technology to use for the semantic description of the service capabilities. As the SERVUS Design Methodology is not related to a Semantic Web Services framework such as OWL-S or WSMO, service matching based on SAWSDL (see section 3.3.3.1) is considered in more detail. It enables to map a request for a service to advertised services in a registry. Instead of performing service matching on a syntactical level based on the comparison of WSDL elements, SAWSDL enables to perform it on semantic level based on reasoning techniques.

The basic idea is that the user expresses the request of a service type in a semantically-annotated WSDL document. Service matching may then be based on reasoning in the semantic models that are both referenced by the requested and the advertised service types. Often, the annotations of the requested service type refer to a different semantic model as the annotations of the advertised service types. In this case there is a need for a conceptual mapping between the semantic models in order to make reasoning applicable (Figure 3-7).

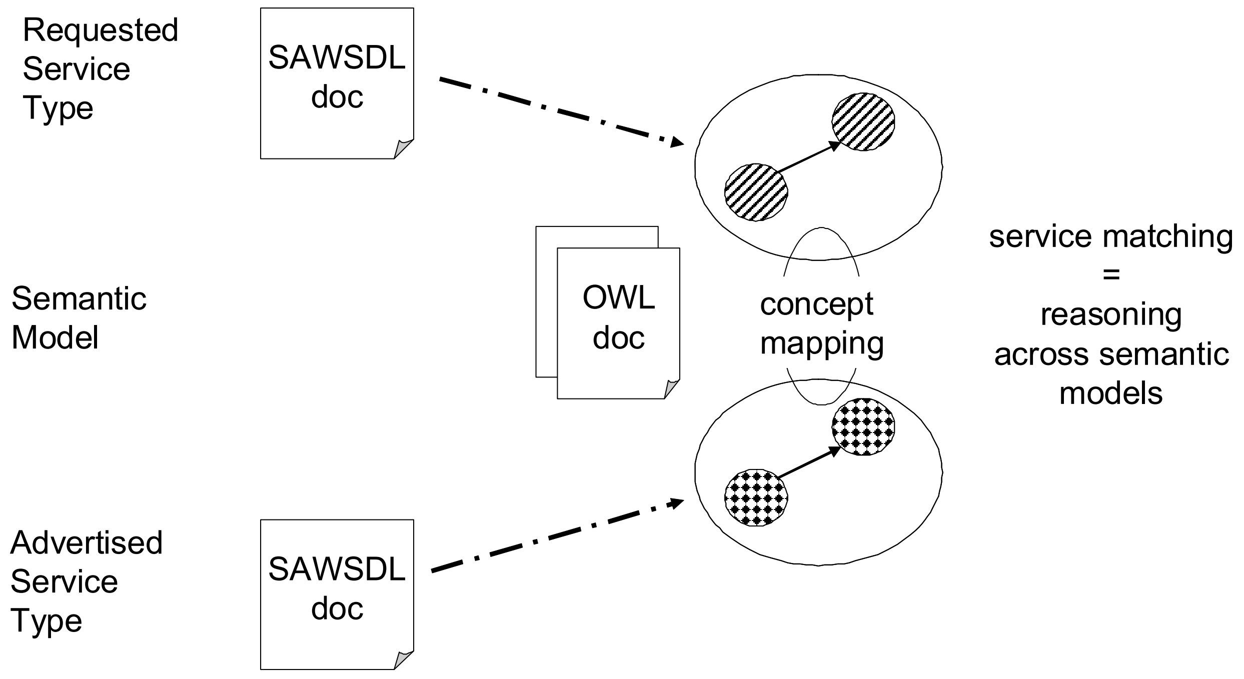


Figure 3-7: SAWSDL Service Matching with Mappings between Semantic Models

3.4.4.2 Service Matching based upon State-oriented Service Description Ontologies

Krutz et al (2007) describes a semantic service description model that has been inspired by the frameworks of Semantic Web services such as OWL-S (see section 3.3.2) and WSDL-S (a former version of SAWSDL, see section 3.3.3.1) but claims to reduce the effort for the creation of the semantic descriptions. The semantic description covers the four service description elements Input, Output, Precondition and Effect (IOPE). Input and output parameters of a service are semantically annotated by referring to concepts of a domain ontology. However, preconditions and effects are defined by referring to the "states" of the objects that are managed by the service and that appear as input and output parameters. For each object, these states have to be defined in advance in a state ontology. Krutz et al (2007) focus upon the thematic domain of education and training. An example in this domain is a document publishing service that manages so-called "Living Documents". These documents may have the major states "published" and "transmitted", whereby the latter state has two substates "mailed" and "faxed". Service matching is carried out as a reasoning activity in the domain and state ontology.

3.4.4.3 Service matching for the Composition and Discovery of Geo-processing Services

Lutz (2007) describes a method for ontology-based descriptions for the semantic discovery and composition of geo-processing services. It relies upon the following assumptions:

- A service only provides one operation.
- It is possible to rely on an agreed-upon set of operations that is specified as a part of a shared vocabulary in a particular domain (possibly based on existing standards such as ISO 19100 series).
- Semantic advertisements and queries are formulated based this shared vocabulary (domain ontology).

It is based on two following basic principles:

1. It requests that the geospatial operations of the requester and of the supplier are specified in reference to a (pre-existing) geospatial domain ontology that includes also operations, i.e. operations from domain ontologies are used as templates for formulating descriptions of requirements and capabilities. Operations are represented by so-called semantic signatures which contains concepts of description logics (instead of data types) to represent input and output parameters, and a specification of pre- and post conditions in First Order Logic (FOL).
2. The matching between the requested operation and the offered operations of the service providers is performed in two steps: In a first step, a pre-selection is carried out by means of subsumption reasoning based on the concept of "function subtypes" (Simons, 2002) which just considers the compatibility of the signatures, whereas in a second step behavioural compatibility of the pre-selected operations is taken into account specified as pre- and post conditions.

The method of Lutz (2007) aims at supporting the automatic composition of geospatial services rather than helping a human system architect in the design of an SOA. Thus, preciseness of the matches is very essential in his approach in contrast to service-oriented analysis and design (SOAD) where approximate matches are sufficient (Toch et al, 2008). An exact match of a requirement with a capability is usually not given in an SOA design phase. Additional reasoning functions for "negotiable matches" have to be investigated.

Furthermore, service discovery in running service networks aims at finding candidate services to satisfy user queries. It is tolerated that the functionality of the candidate services is more restricted than the user request. When translated into the theory of function subtypes this means that the task is to find candidate services that are subtypes of the user queries. However, for SOAD, the task is usually of inverse nature: The capability should be more generic than the requirement such that it may be reused for other requirements, too.

Lutz (2007) has chosen a lightweight semantic representation for operations (semantic signature) instead of other much more complex approaches such as WSMO (Fensel et al, 2006) or OWL-S (Martin et al, 2004a). However, it is still too complicated to be used in the practice of an SOA design where the requirements to be matched are often still ambiguous and incomplete. This property has been considered in the work of Zachos et al (2007) who proposes to exploit the described characteristics of discovered service to make the requirements specifications more complete and consistent.

Overall, the following question remains unresolved: How can operations be compared or even be matched whose signature does not fit, i.e. if the functional requirements are structured in a way that cannot be directly compared with the capabilities of the system? Furthermore, for the design phase, the focus of Lutz (2007) is more on relating possibly ambiguous and incomplete, mostly textually specified functionalities to existing capabilities specified in multiple forms.

3.4.4.4 Services Matching based on Qualitative Service Characteristics

Zhang et al (2007) proposes a methodology called Cascaded Services Exploration (CSE). CSE provides an architectural framework and an enabling technology for a business services analyser that supports analysing, clustering and adapting heterogeneous services for dynamic application integration. The CSE methodology is composed of three steps:

- services categorization,
- services clustering and
- services exposure.

It is assumed that every service may be associated with a service category that is extractable from the service metadata, may be used as structural criteria in service registries and therefore reduces the search space if used in queries to the registry. The core of the CSE approach is the service clustering that further refines each category. The clustering algorithm is based on a feature space whereby each feature provides a quantified value of qualitative service characteristics such as the reliability, accessibility, throughput, latency, cost or security of a service. The value for each feature has to be given by the service provider or is derived by observing the behaviour of the service. Each cluster is represented by a “functional nucleus” that is a function that delivers a single numerical value by combining the individual feature values. The distance between a cluster and a particular service request is defined as the distance between the functional nucleus of a cluster and the requested service point that represents the combination of features of the required service in the feature space. The cluster holding the smallest distance contains the potentially exposed service to satisfy the customer’s request. The final service selection is carried out by a human with “expert knowledge”.

3.4.5 Assessment: Service-oriented Analysis and Design

The methodologies described in more detail in the previous section do only cover a portion of the very dynamic and complex research domain of SOAD. The selection has been made with respect to the subject of this thesis and the design requirements presented in section 1.4. Further SOAD research work is, for instance, performed for the following aspects:

- Security, in particular access control in service platforms, is one of the most important non-functional requirements to be considered already in the analysis and design phase. Emig et al (2008) investigates a model-driven approach to define access control policies which are independent of an identity management architecture. Related to this work, Dikanski et al (2009) present an approach how to integrate existing security products in SOAs, especially how to map security policies defined at service level to product-specific policies.
- A collaborative approach to model-driven software development for SOA-based systems is described in Karle and Oberweis (2008).
- The systematic development of semantic Web applications is one of the research topics in the discussion of Semantic Web Services frameworks. For instance, Brambilla et al (2007) propose a model-driven methodology to design and develop semantic Web service applications and their components. They show that business processes and Web engineering models have sufficient expressive power to support the semi-automatic extraction of semantic descriptions (see the WSMO elements described in section 3.3.2), such that the complexity of dealing with semantics may be partially hidden. Although, at the current stage of development, by means of "conventional design", they also build software that can run on conventional Web technology, this design methodology is mainly targeted towards the WSMO environment,

Although there are numerous SOAD methodologies, the service-oriented design of EIS poses specific needs that are not yet sufficiently addressed. None of the existing SOAD methodologies is tailored to the geospatial domain (R.6). This means that the SOAD methodologies do not contain knowledge about models, guidelines and constraints of geospatial ISO and OGC standards, e.g., interfaces of geospatial service types. However, compliance with these standards is a pre-requisite for interoperability between geospatial services (Lake and Farley, 2007) and thus a side condition to be followed in the service-oriented design of EIS.

One of the major design artefacts in such a design process is the documentation of the resulting architecture (R.11), either on conceptual level covering the abstract design platform or on implementation level covering the concrete service platform. In an ISO/OGC compliant environment, there are guidelines or at least recommendations to structure the architecture specifications according to an interpretation of the ISO RM-ODP viewpoints as described and exemplified in section 3.2.3. A service-oriented design methodology for EIS should take these reference models into account (R.4).

Most SOAD methodologies do not consider in their knowledge base the specifics of spatial-temporal representation of information (R.5) that is inherent to geospatial applications (Visser, 2004).

A typical use case shall illustrate this deficiency. During the information acquisition phase of the EIS Business Process (Figure 2-3) there is often the need to retrieve sensor observations for a given environmental property ("nitrate in groundwater") for a given region ("Upper Rhine Valley") in a given time frame ("in the last ten years"). If the capabilities of service instances in a geospatial service network indicate that nitrate concentration values for groundwater bodies are available but their location is provided in geographic coordinates, and their time range is represented as YYYY.MM.DD.hh.ss, then a design methodology must be capable of semantically relating these spatial and temporal statements, e.g. by means of a gazetteer service (Hill, 2006).

Furthermore, the existing SOAD methodologies are insufficient for the service-oriented design of EIS concerning the following aspects:

- They typically start from the assumption that there is a variety of "business processes" to be analysed, specified and mapped to chain of service operations. However, in the design of EIS, we claim that this variety is quite limited. See the typical EIS Business Process as illustrated in Figure 2-3. As a result, the existing SOAD methodologies are focussing mostly at the functional aspects and neglect the consideration of information objects (R.5).
- The mapping of functional requirements to service capabilities must take into account the way how the meta-information of geospatial services (their capabilities) is described in OGC-compliant services (R.5, R.6).
- According to Ricken and Petit (2008) the management of the computation-independent model (CIM) is neglected. However, this MDA level is important for the iteration support (R.11) and the feedback to the user in a design process (R.1, R.2 and R.3).
- The tools supporting an SOAD methodology are mostly relying upon proprietary product suites (e.g. IBM products supporting the SOMA methodology). This hinders the application of the methodologies for open geospatial service platforms (R.6) and its acceptance in a network of private and public stakeholders (e.g. environmental agencies) that is typical for the design of EIS.
- The CSE methodology provides a design methodology that includes the qualitative characteristics of a service in terms of a multi-dimensional feature space into the service selection. However, it remains unclear from the description in Zhang et al (2007) how the interface of a service as one of the major service characteristics is handled: If it must be mapped to fine-grained service categories in the first step, or how it may be "quantified" in the feature space as part of the clustering algorithm. Furthermore, there are no clear objective guidelines or rules how to quantify qualitative (i.e. non-functional) service character-

istics and requirements (R.5). As a result, the CSE methodology remains to be an interesting approach to include qualitative service characteristics into the overall task of service discovery and matching. However, as such, it does only provide a partial solution of an overall SOA design methodology.

4 The SERVUS Design Methodology

The SERVUS Design Methodology which denotes a Design Methodology for Information Systems based upon Geospatial **Serv**ice-oriented Architectures and the Modelling of **Us**e Cases and Capabilities as Resources (**SERVUS**) comprises the conceptual contribution of this thesis to the research domain of service-oriented analysis and design for Environmental Information Systems (EIS). This section provides an overview about its modelling approach and the design activities. The following sections specify the foundations that underpin the SERVUS Design Methodology and provide formal design activity diagrams before implementation aspects (section 7) and applications examples (section 8) will be presented.

4.1 Overview

The concepts of geospatial SOAs are implemented as geospatial service networks taking into account the constraints and rules of service platforms. A (geospatial) service network hereby comprises the set of networked hardware components and (geospatial) service instances that interact in order to serve the objectives of applications. In this context, the problem of EIS design boils down to the following question:

- What parts of the required EIS application is already provided by the capabilities of existing geospatial service networks in terms of service types ("which services are specified ?") and service instances ("which services are operational and usable ?") ?

Figure 4-1 relates the design problem to the EIS business environment, the use cases, as well as the requirements and capabilities of a geospatial service network.

There is an **EIS Business Environment** which subsumes all the high-level business and organizational needs of the stakeholders concerned with the design, engineering, deployment, maintenance and the use of the EIS. The analysis of the EIS Business Environment itself is outside the scope of this thesis and the design methodology. It is just assumed that user needs have their origin in the EIS Business Environment.

They are condensed and represented in the form of "**use cases**" that describe the desired behaviour of a system from the external perspective of a user and/or the stakeholders. The concept of use cases and use-case-driven development was first introduced by Jacobson (1987) for object-oriented developments. The term use case is used here in a generic sense and not tied to a particular graphical notation such as UML use case diagrams.

Capabilities of specified and deployed geospatial service networks (i.e. the services) are considered as basic building blocks of the targeted implementation of the EIS application. EIS applications are designed such that their generic parts are conceived as compatible increments to existing platform capabilities. The EIS design problem now corresponds to the following question:

- How shall the EIS application be designed such that, on the one hand, it supports the use cases, and, on the other hand, makes a maximum use of the those capabilities of geospatial service networks that the user is allowed and enabled to access ?

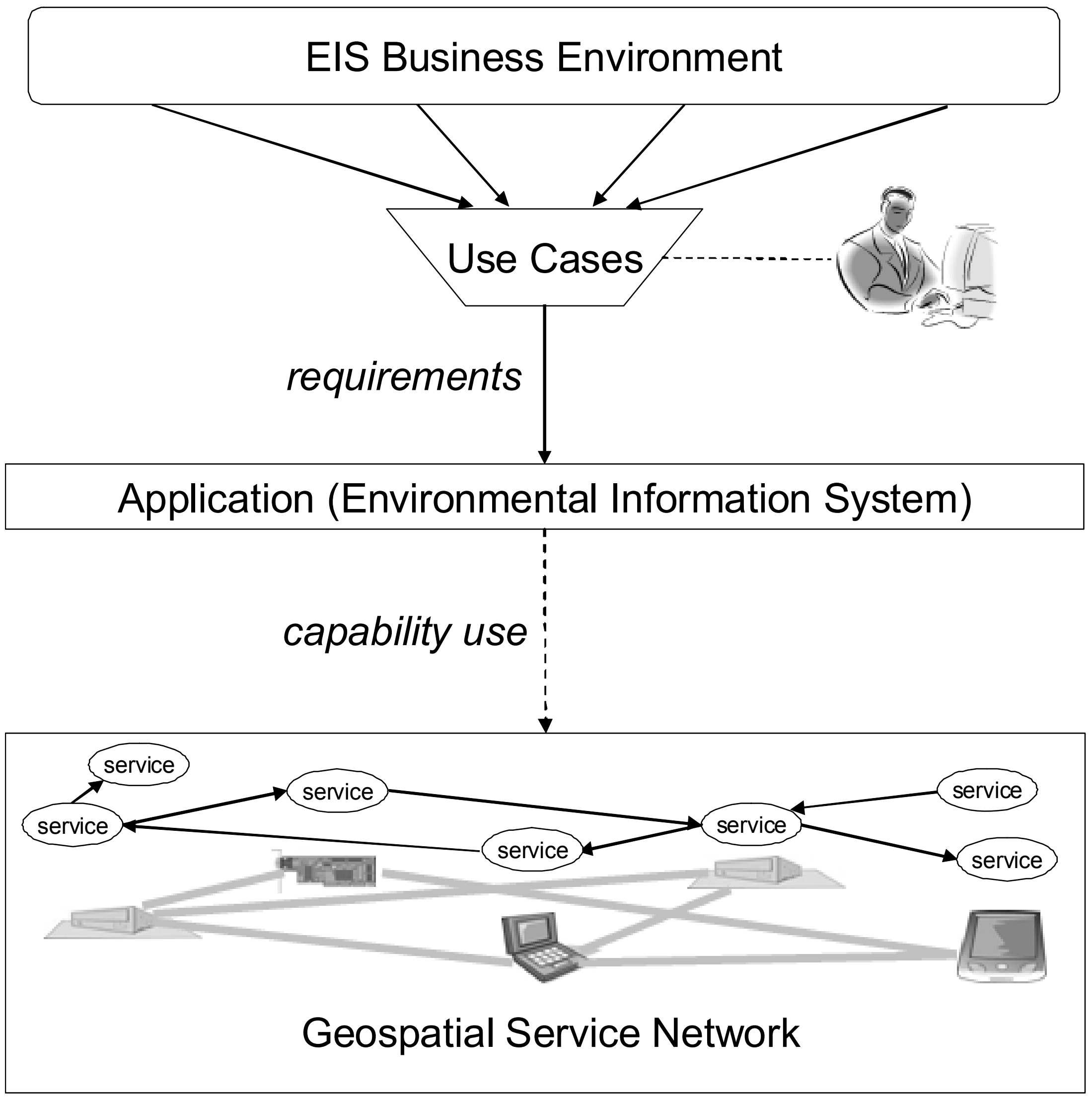


Figure 4-1: EIS Design Problem on top of Geospatial Service Networks

The basic intention of the SERVUS Design Methodology is to answer this question. SERVUS understands the design of an EIS as an iterative discovery and matching activity: available capabilities are discovered and matched against the requirements of the user formulated as use cases.

According to Asadi and Ramsin (2008), a design methodology requires a **modelling language, guidelines** for the order of **design activities** embedded into a **design process**, and resulting **design artefacts** specified in the modelling language.

Table 4-1 summarizes how these elements are covered by the SERVUS Design Methodology.

Design Methodology Element	SERVUS Design Methodology	References
Modelling Language	Reference Model SERVUS Resource Model	Section 5 Section 5.3.2
Design Process	Model of the Design Process	Section 6, Figure 6-2
Design Activities ➢ Sub-activities	Identify Problem ➢ Problem Modelling ➢ Domain Modelling Analyse Capabilities ➢ Publishing Analyse Requirements Abstract Design ➢ Rephrasing ➢ Discovery ➢ Matching ➢ Feedback Generation Concrete Design ➢ Grounding	 Section 4.3.1 Section 4.3.2 Section 4.3.3 Section 4.3.4 Section 4.3.6 Section 4.3.5

Table 4-1: Elements of the SERVUS Design Methodology

4.2 Modelling Approach

The modelling aspects of SERVUS are covered by a modelling framework that embeds knowledge about the thematic domain, the basic elements, capability types and the structure of the underlying geospatial service networks. It belongs to the "foundation" in the sense of the design knowledge base of Hevner et al (2004) as introduced in the IS research framework in section 2.3 and illustrated in Figure 2-9.

Following an MDA approach (see section 3.4.1), the challenge of an SOAD methodology is to enable a consistent and traceable model mapping and model management across the **Enterprise**, **Process** and **Service** layers of abstraction as defined by Bieberstein et al (2006). Figure 4-2 shows the modelling hierarchy of SERVUS in relationship to these layers.

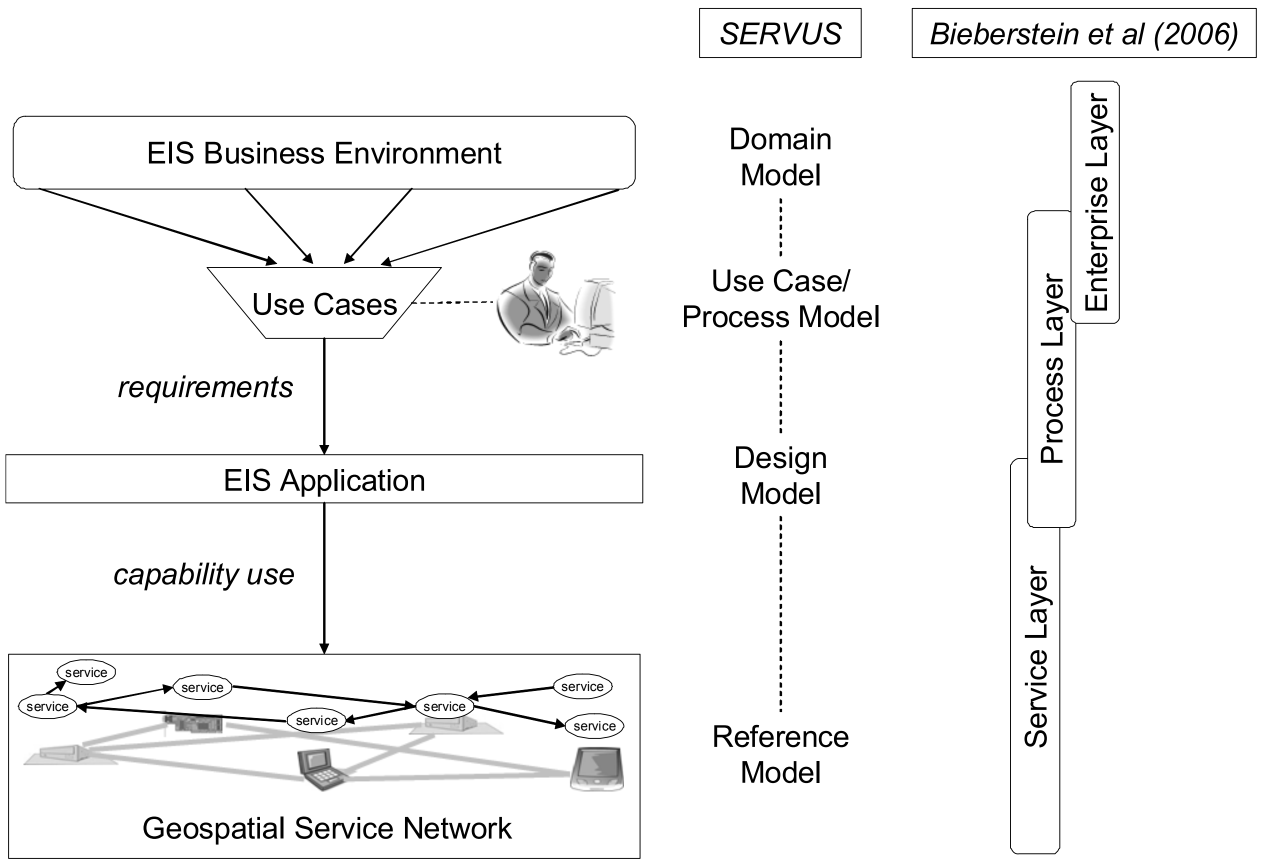


Figure 4-2: SERVUS Model Hierarchy

4.2.1 Domain Model

The SERVUS **Domain Model** belongs to the Enterprise abstraction layer of Bieberstein et al (2006) and describes "the way an enterprise (here: organization, institution, agency,…) operates in a particular industry" (here: EIS business environment). However, the SERVUS Design Methodology focuses on the informational aspect of the Enterprise layer and neglects the organisational aspects. The SERVUS Domain Model represents the thematic domain to which the problem belongs to. It formally defines the part of the world that comprises the universe of discourse between the user and the system designer, i.e. it comprises the shared knowledge about the application domain. Typically such shared knowledge is represented by the specification of an ontology (see section 3.3.1).

The SERVUS Design Methodology assumes that a domain model is either already available (e.g. from previous design projects or from the community) or it is drafted or refined during the analysis phase. It shall be documented as part of the Enterprise Viewpoint of the architecture document (section 6.2). The format of the document, the language and notation used (e.g. textual and/or graphical) and the degree of formalism may vary and is not prescribed by SERVUS.

Examples of thematic domains are scientific disciplines such as hydrogeology, weather forecasting or legislative frameworks like the European Water Framework Directive (see section 2.1.3.3). The domain model is generated in the domain modelling activity (see section 4.3.2). At least for the core part of a domain model, it is assumed that domain modelling has already taken place beforehand by the community of the thematic domain. In this case, a suitable domain model just has to be selected as part of the analysis activity (see section 4.3.2). However, If none exists or the existing ones are insufficient or incomplete for the given design task, they have to be generated or improved as part of the analysis activity.

Domain models are conceptual models, i.e. they abstract away irrelevant details, and thus allow more efficient examination of the current, past, and projected future states of the universe of discourse (Borgida and Brachman, 2007). Depending on the level of expressiveness required they may but need not necessarily be defined in an ontology language such as the Web ontology language (OWL) (Bechhofer et al, 2004; Stuckenschmidt, 2009). Domain models for OGC-compliant geospatial applications are typically applications of the OGC General Feature Model. The General Feature Model is defined in the general-purpose Unified Modeling Language (UML) (Rumbaugh et al, 1998). It adopts a **(geographic) feature** as the basic modelling unit (Percivall (ed.), 2008). A feature is an abstraction of a real world phenomenon, whereas a geographic feature is a feature with a location relative to the Earth. Features have thematic, spatial and temporal attributes, and may be related to each other in associations and inheritance relationships

4.2.2 Use Case and Process Model

Use case and process modelling is part of the Process abstraction layer of Bieberstein et al (2006). Use case and process models are the result of the requirements analysis. However, in the literature, the relationship between and the importance of these two types of models in a design methodology is not consistent. It depends on how use cases are defined. Proponents of process-orientation, such as Zimmermann et al (2004), state that "processes are more suitable for the identification of services than use cases, since one has to analyse more than one system at a time to derive adequate services". Proponents of use case-orientation recognize the importance of use cases for the sake of the stakeholders.

As explained below, SERVUS follows an approach that combines use cases and processes. We adopt the definition of use case and use case models as defined by Jacobson and Ng (2005) in the course of aspect-oriented software development.

Definition (4.1): A **use case** models the behaviour of a system. A use case is a sequence of actions performed by the system to yield an observable result that is typically of value for one or more actors or other stakeholders of the system (Jacobson and Ng, 2005).

Transferred to our context, a use case expresses the functional, informational and qualitative requirements of a user (i.e. an actor or a stakeholder), whereby the functional requirements are represented by the "sequence of actions", the informational requirements cover the content of the "observable result" and the qualitative needs encompass all the non-functional aspects how the result is yielded and the quality of the result which is important for the decision if the result is “of value” to the user.

Therefore, the degree of abstraction and formalism, and the language should be such that it is adequate to the domain of expertise of the stakeholders. To serve as an agreement, it shall be understandable to the stakeholders but also precise enough. Often this means that use cases are specified in a non-technical way, normally achieved using plain text in natural language. However, in order to reduce the ambiguities and impreciseness of descriptions in natural language, a structured textual description is preferred, i.e. the use case description may be structured according to a given template, e.g. an application form associated with code lists.

Definition (4.2): A **use case model** of a system serves as an agreement between the stakeholders (or customers) of the system and the developers on what the system should do and what qualities the system should have (Jacobson and Ng, 2005).

Using these definitions, the role of process models is to specify the “sequence of actions” of use cases, thus, on the one hand, process specifications refine and formalize use case specifications. On the other hand, an individual action as part of a process specification may again be seen as an individual use case. Consequently, both models may be used complementarily. We classify both of them being part of the Process abstraction layer. Model mapping, either manually or automated by tools, is applied in order to govern the transition from the process layer to the service layer of abstraction.

Notes:

1. It may be discussed if the use of the term “system” in the definition of uses cases and use case models is adequate in the context of an SOA as, in order to fulfil the SOA design principles of “service reusability” and “service composability” (section 2.2.2; Erl, 2008a), there should be no explicit system boundary that constrains the applicability of the resulting services. However, in the domain of environmental management, which is the primary target of SERVUS, there is rather a trend towards **systems-of-systems** with defined responsibility domains for the SOA governance than thinking of environments without well-defined system limits (section 2.2.4). The main reasons for this tendency are the needs to guarantee the fulfilment of non-functional requirements (NFR) such as performance, dependability and access control.
2. In systems-of-systems all individual systems as well as the encompassing systems have basically their own use case model associated to them although this use case model may not be explicitly available in written form, e.g. in case of legacy systems.

For defining the scope of the SERVUS methodology, in particular the mapping of the use cases to the service layer, use cases have to be classified further. Jacobson and Ng (2005) distinguish between application use cases and infrastructure use cases. As the name suggests, **application use cases** deal with application concerns and reflect the functional requirements. They refer to what users can do with the system. In SERVUS we extend this definition and explicitly include the informational requirements into the application use cases due to their importance and complexity in geospatial applications. They refer to the information elements which are handled by the functions in terms of input and output parameters.

Infrastructure use cases deal with infrastructure concerns and reflect the non-functional requirements. NFR impose some behaviour on the infrastructure (in our case the geospatial service platform) such as the way authorization, performance or reliability is handled. The analysis of the NFR results in a quality of service (QoS) model of the system to be designed.

The Human Computer Interaction (HCI) community has put effort in exploring similarities and differences between use case based analysis and task analysis, the latter being a widely-used technique to capture user-interface requirements and model them in the form of end-user tasks (Kim and Carrington, 2002). They came to the conclusion that,

- from the HCI viewpoint, use cases may be considered as a form of hierarchical task analysis favoured by the Software Engineering community, and
- use case driven analysis is also applicable to user-centred design and may integrate human factors in the development process.

Kim and Carrington (2002) propose an integrated use case-based approach, however, distinguishing between a **user-interface view**, and a **system-centric view**, covering the functional aspects of the system. The user-interface view covers the end-user tasks describing the requested behaviour of the system at the end-user interface. This analysis results in a task model. The system-centric view described the requested behaviour of the system at the technical interfaces. This use case analysis results in an analysis model. There is a scaling-up problem to maintain the consistency between the two variants of the use case models, i.e. the analysis and the task model, in larger applications.

Figure 4-3 summarizes the use case variants and the resulting models of Jacobson and Ng (2005) and Kim and Carrington (2002).

The primary intention of SERVUS is to cover infrastructure use cases as well as application use cases, the latter, however, restricted to the system-centric view. The reason for this scope of SERVUS is that this spectrum of use cases is the most important and decisive one to assure reusability and interoperability in large-scale EIS and system-of-systems in the environmental domain based upon service platforms. In these system designs, the user-centric view is an add-on and typically not unique. On the contrary, the system (of systems) architecture should be generic in order to enable the design of various user interfaces tailored to the individual needs of the end-user.

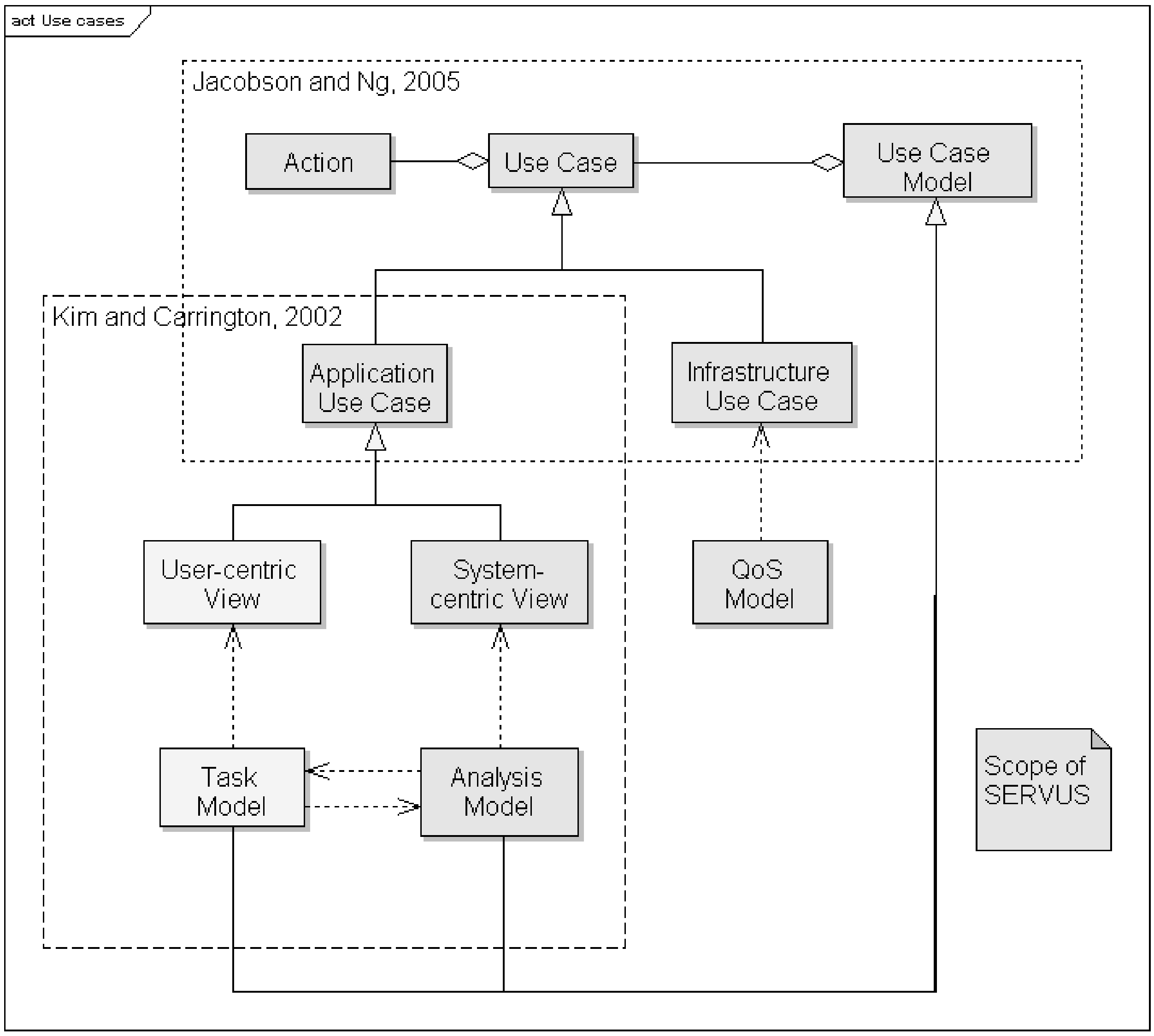


Figure 4-3: Use Cases and their resulting Models

4.2.3 Design Model

The **Design Model** is the core of the SERVUS Design Methodology and documents both the starting point and the results of the design process. The design model is positioned in the "gray area" (Bieberstein et al, 2006) between the Process and the Service layer of abstraction. The SERVUS Design Model is a conceptual model that comprises requirements and capabilities such that a mapping between both is facilitated. The result of the mapping is documented in the design model, too.

The design model is an answer to the observation that information systems, especially EIS, do not vary very much in the way they retrieve and process information. EIS typically follow the same type of business process (see section 2.1.2) but vary in the type of information objects handled and the form and format in which these information objects enter (through sensors or import interfaces) and leave (through user or export interfaces) the boundaries of the system. Furthermore, the actions that are to be performed upon the information objects may be mostly mapped to classical create, read, update and delete operations upon information objects as known from persistent storage.

The basic idea is to leverage this observation and to use **resource model**s as realization of design models. A resource model acts as common modelling language to which both use cases of the Process abstraction layer and capabilities of the Service layer may be mapped. A resource in the resource model is basically an information object that is uniquely identified, may be represented in one or more representational forms and support methods that are taken from a limited set of operations whose semantics are well-known (uniform interface). A resource has own characteristics (attributes) and is linked to other resources forming a resource network. Furthermore, resource descriptions may refer to concepts of the domain model using the principle of semantic annotation (see section 3.3.3), yielding so-called "semantic resources".

The basic idea of the resource model is derived from the Representational State Transfer (REST) architectural style for distributed hypermedia systems as conceived by Fielding (2000). The resource model is the core upon which the SERVUS design activities are based. Its formal definition is contained in section 5 as part of the definition of the SERVUS Reference Model.

The design model contains the following parts:

- A model of the requirements (req's model in Figure 4-4). This part reflects the requirements to be fulfilled. They are specified in the language of the design model as a result of the rephrasing activity (see section 4.3.4.2).
- A model of the capabilities (cap's model in Figure 4-4) consisting of a description of the existing capabilities of the service platform. The capabilities are structured into abstract capability units (e.g. resources and their representations) such that they may be semantically linked to the concepts of the requirements model. This is the result of the publishing activity (see section 4.3.3). Furthermore, it is distinguished between
    - capability types, i.e. specifications of capabilities (e.g. specification of interface and service types as well as information model schemas), and
    - capability instances, i.e. deployed and operational instances of capability types (e.g. a running Web service with a known URL delivering data sets defined with its thematic, spatial and temporal characteristics).
- A model of the mapping between the requirements and the capabilities (req2cap model in Figure 4-4). This model represents the result of the design process and is therefore produced in the discovery and matching activity.

Furthermore, the design model contains references to concepts of the domain model, e.g. for environmental risk management, and to concepts of the reference model.

4.2.4 Reference Model

Although it is widely recognized that information systems engineering needs a co-development of requirements (dealing with the problem space) and architectural concepts (dealing with the solution space) in a iterative manner, there are only rare solutions supporting such an interlinked design approach (Pohl and Sikora, 2007). We argue that an important aspect for such a co-development is a reference model that enables an iterative specification and documentation of the design and development artefacts.

The **reference model** sets the terminological and architectural foundation for the design of the geospatial service-oriented architecture. It specifies the major concepts and terms and their relationships that are essential for the architecture. For example, it defines what is meant by "service", "interface", "feature" or "resource". Standardization organizations such as OGC, OASIS and The Open Group published reference models and ontologies for SOAs. These reference models were presented in detail in section 3.2.

SERVUS is based upon the Reference Model for the ORCHESTRA Architecture (RM-OA) whose conceptual model has been developed as part and foundation of this thesis (Usländer (ed.), 2007). It is described in section 5.

4.3 Design Activities

The SERVUS Design Process consists of the following activities as illustrated in Figure 4-4: problem analysis, publishing, domain modelling, rephrasing, discovery, matching, grounding and feedback generation. These design activities are using information elements of the models that have been presented above in the SERVUS modelling approach, i.e. they rely upon the reference model, the domain model and the design model. The use of these models by SERVUS design activities is illustrated in Figure 4-4. This figure also indicates by whom the activity is carried out in principle. Two roles are basically distinguished: a **user** that defines the problem on behalf of a stakeholder, e.g., an environmental agency, and a **system designer** that is responsible for the design of the system architecture supporting the problem solution.

In the following just a textual introduction of these design activities is provided. Section 6 provides a formal UML-based description of the individual actions in these activities and the documentation of their design artefacts according to the viewpoints of ISO RM-ODP. The result is the process model of SERVUS.

4.3.1 Problem Analysis

The objective of the problem analysis is to derive the set of functional, informational and qualitative requirements from the problem to be solved and document them in natural language in electronic format (marked as "req's" in Figure 4-4). Usually, in an SOA project there are conventions set up by the project management and documented in a quality handbook that prescribe the format in which the requirements and uses cases must be described. For instance, see the requirements specification of the SANY project (Williams (ed.), 2008). Basically, the SOA design process must not prescribe a pre-defined format but just assumes that the requirements are textually specified in machine-readable documents and expressed in human language.

Problem analysis is an inherent activity of the user domain. However, in the spirit of a spiral co-development of requirements and system architecture (see section 2.3.3), the problem analysis activity may also be carried out jointly by the user and the system designer.

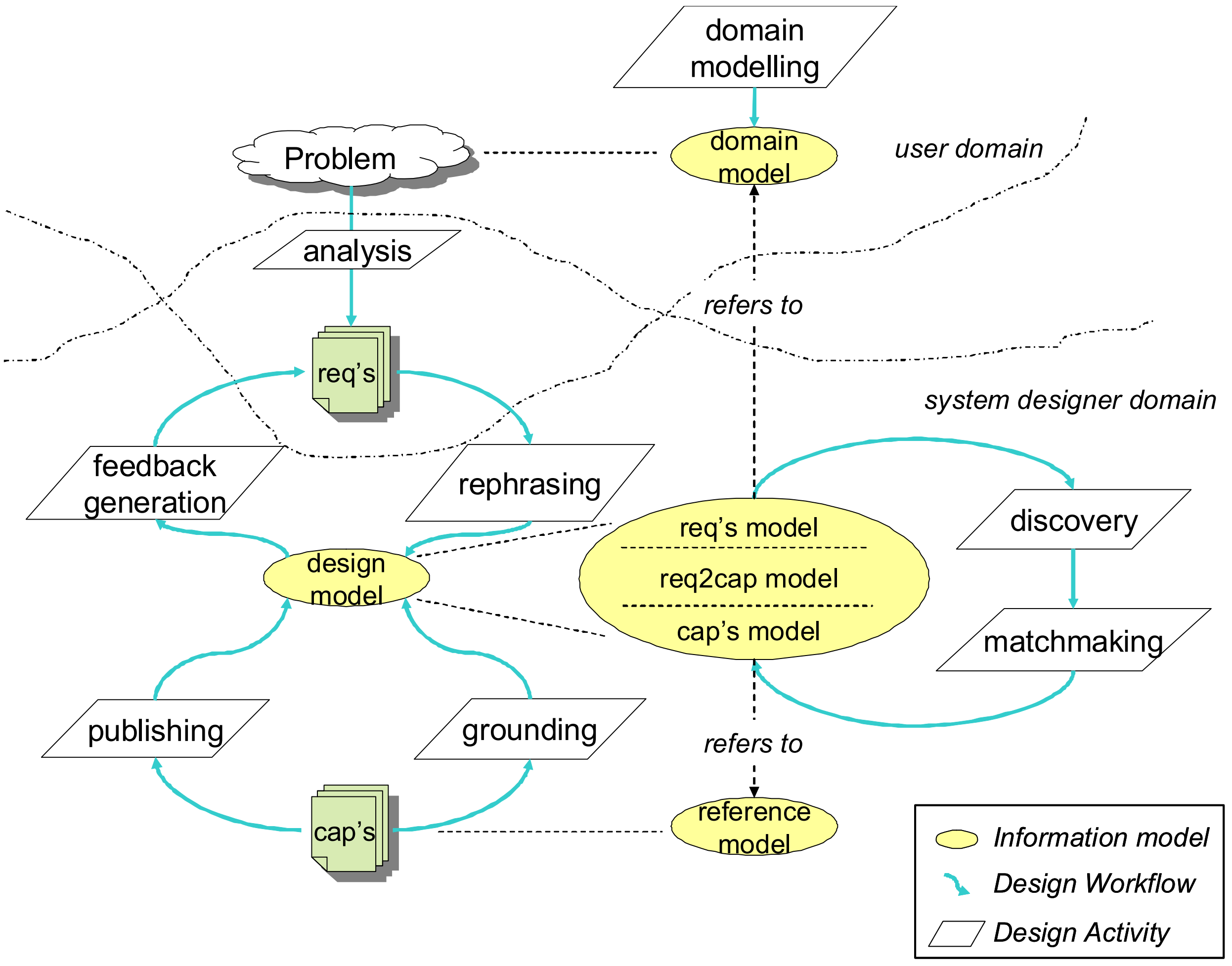


Figure 4-4: SERVUS Models and Design Activities

4.3.2 Domain Modelling

The objective of the domain modelling activity is to define the basic concepts of the thematic domain to which the problem belongs, and their interrelationships. Usually, this activity is carried out by experts of professional organizations representing a thematic community or outstanding institutions such as universities or research institutes.

Domain modelling may be a very tedious and time-consuming task as domain experts with different thematic background and objectives have to agree upon a common set of terms and concepts and their relationships. For geospatial domain modelling this means that domain experts identify geographic features as abstractions of real world phenomena perceived in the context of geospatial applications, classify them in feature types and refine them by specifying an application schema, i.e. by adding feature properties such as attributes and relationships according to the ISO General Feature Model (ISO 19109:2005).

Any feature may have a number of attributes, some of which may be numeric, a spatial geometry, meta-information, temporal information, etc. Examples of features types are earthquake, forest fire, road, building, water protection area, and monitoring station, but also sensor observation, measurement value or document. 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".

Today, domain models are increasingly specified as formal ontologies, for example, expressed in description logics as formalism for representing knowledge (Baader et al, 2007)[20]. In such cases, domain modelling activity is called ontology engineering and is a research topic of its own. Borgida and Brachman (2007), for instance, propose a methodology for conceptual modelling based upon description logics. A specific challenge in geospatial domain modelling is to map and translate between different ontologies in the same application domain or in multi-domain environments including spatial and temporal aspects (Visser, 2004.

For simplicity of the illustration the domain modelling activity is located in the user domain. However, it may also be performed or at least supported by an external thematic expert or a group of such experts. There is a multitude of national and international efforts to define domain models for the environmental field, typically in form of thesauri (Bandholtz et al, 2009) or logic-based ontologies based upon the Semantic Web for Earth and Environmental Terminology (SWEET[21]) (Raskin and Pan, 2005). When existing and fitting, these domain models may be used for the design process as a common knowledge foundation within the user community of the EIS to be designed.

[20] The definition and roles of ontologies are further discussed in section 3.3.

[21] Available at http://sweet.jpl.nasa.gov/ontology/ .

4.3.3 Publishing

The design process usually starts with the publishing activity. The publishing activity represents the step in which the capabilities of the selected platform are entered into the capability model as part of the design model. This requires that the capabilities are translated into the language and adapted to the scheme of the design model. When applying a resource-oriented approach, existing capability descriptions containing functional, informational and qualitative characteristics are transformed into “offered resources”.

Specifications of capability types (e.g. models of standardized OGC service specifications) as well as information about capability instances, usually called meta-data or meta-information, are entered into the design model. Instance information is typically entered into geospatial catalogue systems whose schemata are structured according to the capability types. This publishing step is either performed manually, or supported by some automated harvesting process.

Lutz (2007) proposes that ontological descriptions of standard services should even be specified as part of the domain model. In our distinction between a domain and a design model (see section 4.2), we consider them to be part of the design model. The reason is that the domain model represents the thematic domain of the user whereas the standard services are usually independent of the application domain. Thus, they are considered to be generic building blocks that have to be turned into thematic services through tailoring and configuration.

The next activities are part of the design iteration process and are carried out by the system designer.

4.3.4 Rephrasing, Discovery and Matching

4.3.4.1 Overview

SERVUS distinguishes between rephrasing, service discovery and matching as three related but sequential activities carried out by the system designer. In a first step, use case specifications expressing functional, informational and qualitative requirements are transformed (rephrased) into requested resources. In a second step, candidate services that may fulfil the request are searched for (or discovered), and in a third step, the best-fitting one is selected (matching).

Due to the resource-oriented approach discovery and matching of requirements against capabilities is facilitated as these activities are transformed into a searching activity within one common solution domain, a network of semantic resources, as illustrated in Figure 4-5. This facilitates the discovery of possibly fitting capabilities (candidates) and the matching of requirements with capabilities (selection) as the notion of resources is closer to the “universe of discourse” of the user than technical signatures of specific services. The basic design problem boils down to the problem of “resource matching”, i.e. a search for “similar” resources that may become candidate resources (discovery) within a network of semantic resources and a selection of the most fitting one (matching). The term **matching** is used here in the broad sense of Euzenat and Shvaiko (2007) who define matching in the context of "ontology matching" as "finding correspondences between semantically related entities in different ontologies". In SERVUS, correspondences have to be found between requested and offered resources both of which may be represented as concepts in different ontologies. In section 6.3.3.2 we will discuss how existing ontology matching approaches may be applied to the semantic resource network.

The semantic resource network models the requirements of an EIS application to be designed as a network of interlinked requested and offered resources. The links between the requested and the offered resources represent the state of the design process.

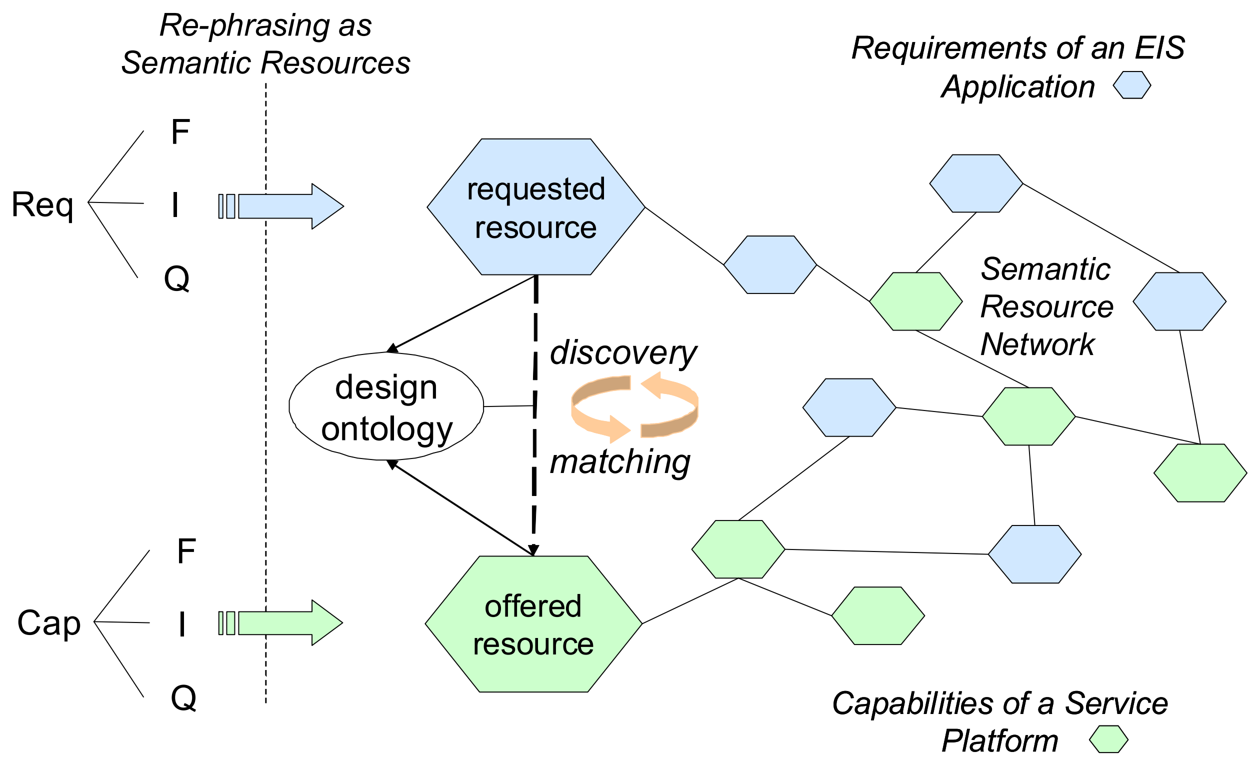

Re-phrasing as Semantic Resources
Requirements of an EIS Application
Req
F
I
Q
requested resource
design ontology
discovery
matching
Semantic Resource Network
Cap
F
I
Q
offered resource
Capabilities of a Service Platform

Figure 4-5: Principle of Rephrasing, Resource Discovery and Matching

The schematic Figure 4-5 shows that the cardinality of the relationship between requested and offered resources is m:n, i.e.

- A requested resource is not linked to any offered resource. In this case, there is either no mapping possible, i.e. this resource need to be developed from scratch, or the design process did not yet cover this requested resource,
- A requested resource is related to exactly one offered resource. This is the easy case in the sense that the offered resource completely corresponds to the requested resource.
- A requested resource is linked to more than one offered resources. In this case there are several candidate offered resources which correspond to the required

resource to a certain degree. In this case, an evaluation activity is necessary to finally select the best-fitting offered resource.

- More than one requested resource is linked to the same offered resource. This is the ideal case as it is the ultimate motivation for an SOA: The offered resource may be re-used several times, i.e. it is generic enough to fulfil the needs of several requested resources.

There are offered resources which are not (yet) linked to any requested resource. This is normal as the use cases do not always require the use of all offered resources.

4.3.4.2 Rephrasing

The system designer selects subsets of the requirements and feeds them into a rephrasing activity. The purpose of the rephrasing activity is to translate and relate the elements of the requirements to the concepts of a design model. It may be carried out manually or semi- automatically supported by a design tool, e.g. an annotation tool.

When applying the resource-oriented approach, it is the objective of the rephrasing activity to reformulate the requirements as requested resources, their representations and required methods and interlink them with concepts of the domain model as well as the design model.

Note: There is an analogy between the set of requested resources resulting from the rephrasing activity and the so-called "**blueprint of a service inventory**" resulting from the overarching service inventory analysis explained in the Mainstream SOA Methodology (Erl, 2008a; Appendix B: Process Description). Here the service inventory blueprint[22] is the "primary target deliverable of repeated iterations through the service-oriented analysis process" which can then be "further analysed and refined as necessary before committing to the actual creation of a physical service inventory". The resource model may be seen as a specific representation form of a service inventory blueprint.

4.3.4.3 Type and Instance Discovery

Following the glossary and Web service architecture of the W3C (W3C, 2004a) discovery is the "act of locating a machine-processable description of a resource that may have been previously unknown and that meets certain functional, informational or qualitative criteria. It involves matching a set of functional and other criteria with a set of resource descriptions." Applied to the context of SERVUS, the objective of the discovery activity is to search for capabilities that are candidates to fulfil the requirements. The search query must be derived from the requirements model and may be improved by the knowledge stored in the design model. The discovery activity is divided into two steps:

[22] See also http://www.soaglossary.com/inventory_blueprint.asp.

1. **Type Discovery**: Type discovery corresponds to a search within the design model. The objective is to find candidate capability types whose instances may potentially fulfil the requirements, i.e. the actions of the use cases (see section 4.2.2). The result of this activity may be fed into the second step of discovery, which is the search for possible instances of candidate capability types.
2. **Instance Discovery**: Instance discovery corresponds to a search in deployed meta-information systems that store information about the actual capabilities of service networks. Examples of such meta-information systems are catalogues of geospatial resources (i.e. data sets, documents and services) or Internet search engines. A major task of the instance discovery is to classify or even rank the search results according to their semantic proximity to the query and the domain model. The result is then a ranked set of candidate capabilities that are proposed to the system designer according to their degree of match. It is then up to the system designer to decide which capabilities to take. These are fed into the matching activity.

The key problem in instance discovery is that both the requirements and the capabilities are described on a similar semantic level based on common terms and concepts in order to support the system designer when formulating the queries and to make automatic classification and ranking of search results realistically possible.

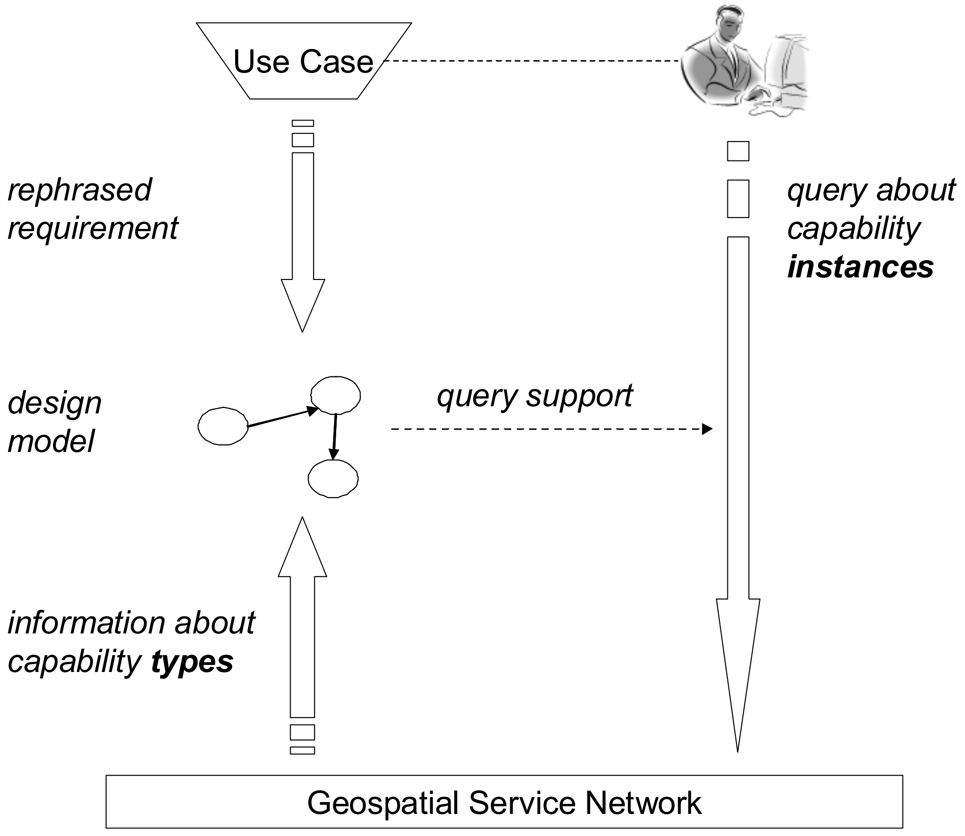

Use Case
rephrased requirement
query about capability instances
design model
query support
information about capability types
Geospatial Service Network

Figure 4-6: Basic Principle of the SERVUS Design Methodology

4.3.4.4 Query Support

SERVUS is based upon the assumption that a system designer can exploit discovery capabilities provided by geospatial service networks to learn about available and possibly reusable capability instances. The idea of "query support" is to support the discovery activity in formulating the queries through the knowledge that is stored as resource in a design model. This design model contains specification of capability types of geospatial service networks, e.g., type specifications of standard services and information models of the OGC.

Geospatial service networks are built by instances of these capability types. They comprise implementations of service types (service components), deployments of service components (service instances), as well as datasets structured according to specified information models and accessible through service instances.

As illustrated in Figure 4-6 it is assumed that a system designer may retrieve information about capability instances (capability meta-information) by querying the geospatial service network itself. The basic idea of the SERVUS methodology is to use the knowledge stored in the design model to support the formulation of the queries.

In order to enable the possibility to query the geospatial service network about available capability instances, information about these capability instances (capability meta-information) has to be provided. In geospatial service networks, this possibility is typically offered through so-called (geospatial) catalogues that are accessible by one or more catalogue services (Nebert, Whiteside and Vretanos (eds.), 2007). Catalogues are intrinsic capabilities of the geospatial service network itself (Figure 4-7).

In this case techniques such as interactive or automatic expansion of geospatial queries based upon semantic bounding boxes (Hilbring and Usländer, 2008) may be applied. They delimit the scope of the search space as illustrated in the SERVUS Implementation Architecture (section 7). "Semantic bounding boxes" are an application of query expansion techniques to ontologies, traditionally known in information retrieval as "query expansion based on similarity thesaurus" (Baeza-Yates and Ribeiro-Neto, 1999).

There are basically two interaction schemes with the catalogue from the viewpoint of the service network:

1. Capability meta-information is actively sent by the services themselves to the interface of the catalogue service for storage in the catalogue (publishing).
2. The catalogue retrieves the catalogue meta-information through a defined interface at the service (harvesting).

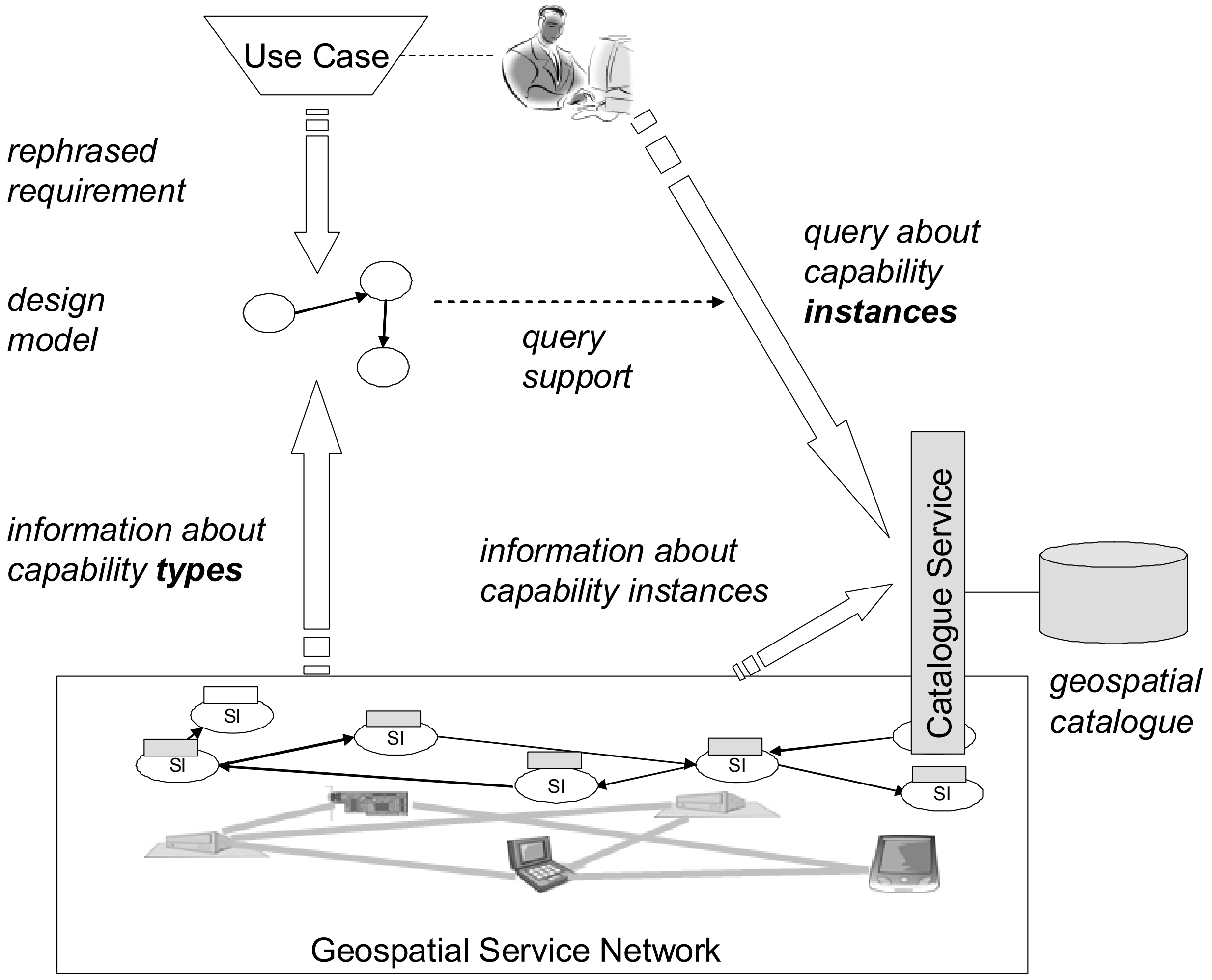

Use Case
rephrased requirement
design model
query support
query about capability instances
information about capability types
information about capability instances
Catalogue Service
geospatial catalogue
SI
Geospatial Service Network

Figure 4-7: Provision of Capability Meta-Information through Geospatial Catalogues

The harvesting scheme is particularly interesting when replacing dedicated geospatial catalogue services by established Internet search engines such as Google or Yahoo. In this case, capability meta-information is directly provided in the form of Web resources that is discoverable by Internet search engines (see Figure 4-8). Although conceptually supported by the SERVUS Design Methodology, this approach is not yet supported by the current implementation architecture of the SERVUS Design Methodology as described in section 4.

4.3.4.5 Matching

The objective of the matching activity is to map requirements with capabilities. It comprises the evaluation of the adequacy of the candidate capabilities (i.e. types or instances) proposed by the discovery activity, the selection of one or more candidate capabilities, and finally the documentation of the mapping in the design model for traceability.

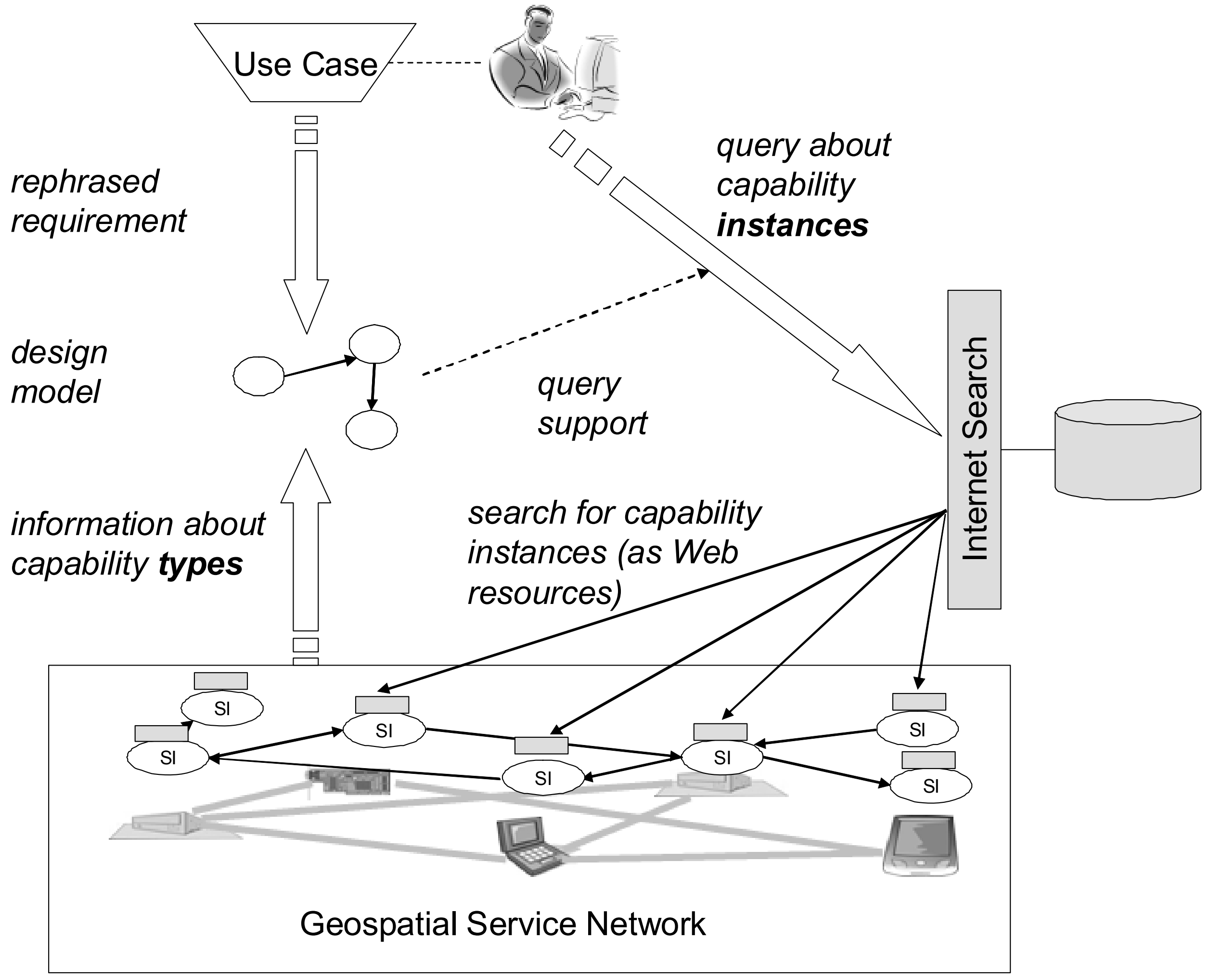

Use Case
rephrased requirement
query about capability **instances**
design model
query support
Internet Search
information about capability **types**
search for capability instances (as Web resources)
SI
Geospatial Service Network

Figure 4-8: Provision of Capability Meta-Information as Web Resources

There are two basic possibilities:

- A mapping of the requirements to one or more candidate capabilities is possible.
- No mapping of the requirements to the capabilities is possible, i.e. the degree of match is not sufficient. This may have the following reasons:
  - The requirements are too vague or too generic and need to be refined such that they may match existing capabilities, e.g.
    - If the requirements were weakened, alternative capabilities would match.
    - If the capabilities were extended, they would match the requirements.
  - Additional contextual information is required in order to better classify the requirements.

- The requirement is new in the sense that it may be turned into a new capability of the system (which has to be specified by the system designer).
- The requirement domain is not reflected well enough in the design model.

The system designer may then turn to the next requirements or continue with the next step, the grounding activity.

4.3.5 Grounding

The grounding activity maps the (new or extended) capabilities to the specification style and language of the service platform and adds it to the set of platform capabilities (marked as “cap’s” in Figure 4-4). The grounding activity is usually supported by engineering tools.

Note: This activity is out of scope of this thesis and is only mentioned for the sake of completeness.

4.3.6 Feedback Generation

The feedback generation activity is an optional validation activity. It re-formulates the formal specification of the capability into the original language of the user and explains how the original user requirements have been satisfied. Its purpose is to let the user, possibly supported by the system designer, judge whether the user requirements have been correctly “understood”. The feedback to the user is an important milestone in order to detect possible misunderstandings already in an early phase of the system design and implementation. It gives the user the chance to adapt or refine the requirements. The feedback generation activity may re-use the concepts of the design model.

Note: The resource model is a first step to support the feedback generation activity. However, feedback generation itself is out of scope of the present thesis and is only mentioned for the sake of completeness.

5 Reference Model

5.1 Overview

The definition of a design methodology needs a "modelling language" for the design artefacts, i.e., an approach and a notation to specify them. SERVUS defines this modelling language in terms of a SERVUS Reference Model that comprises an architectural framework and a conceptual model. The architectural framework provides the guidelines and rules as how to structure the system specification. It is specified in section 5.2. The conceptual model identifies and specifies the major entity types (concepts) of the system and their relationships. It is specified in section 5.3. As open geospatial service platforms lay the technological foundation of SERVUS, the reference model relies upon geospatial standards and recommendations of relevant standardization organizations wherever possible and adequate.

5.2 Architectural Framework

5.2.1 Interpretation of the RM-ODP Viewpoints

SERVUS requires is a clear structure for the documentation of the design artefacts in the individual design activities. For this purpose, the Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer (ed.), 2007) has been used as foundation. The RM-OA provides a platform-neutral abstract specification of a geospatial service-oriented architecture that responds to the requirements of environmental risk management applications but is not exclusively tied to this application domain. It comprises generic architecture services and information models based on and extending existing OGC specifications.

The RM-OA has adopted ISO/IEC 10746 Reference Model for Open Distributed Processing (RM-ODP) to structure the system documentation. As elaborated in section 3.2.2 RM-ODP is an international standard for architecting open, distributed processing systems that proposes to subdivide the specification of a complete system into so-called **viewpoints**. However, as the RM-ODP has been originally conceived in the spirit of distributed object-oriented middleware, the RM-OA process model has interpreted the RM-ODP viewpoints driven by the characteristics of geospatial service networks as described in Table 5-1. In order to reflect the spirit 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 the RM-OA and in SERVUS.

All major elements and concepts of SERVUS are oriented at this interpretation, especially the model of the SERVUS Design Process that is presented in section 6. The key aspect for the design of service-oriented architectures is the observation that the information and service viewpoints are tightly coupled and may not be handled individually. This is reflected in the conceptual model of SERVUS that defines the relationship between the concepts "feature", "service" and "resource" (section 5.3). This conceptual model is an enhancement of the conceptual model defined by the RM-OA.

RM-ODP Viewpoint	RM-OA Interpretation (applied to SERVUS)	Example
Enterprise	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.
Information	Specifies the modelling approach of all categories of information including their thematic, spatial, temporal characteristics as well as their meta-data.	Information objects specified in UML class diagrams and referred to by the specification of the processing service (e.g. as parameter types).
Computational	(in the RM-OA and in the SERVUS Design Methodology referred to as Service Viewpoint) Specifies the Interface and Service Types	Specification of the externally visible behaviour of a service type, e.g. UML specification of the interface types of the processing service including the possibility to perform statistics Service support for service orchestration and choreography.
Technology	Specifies the technological choices of the platform, its characteristics and its operational issues.	Specification of a W3C Web Services platform and a GML profile.
Engineering	Specifies the mapping of the service specifications and information models to the chosen platform. Considers the characteristics and principles for (geospatial) service networks.	Provision of the service implementation specification, incl. mapping of the UML specification to WSDL and functional service properties (e.g. persistency). Decision on access control policies.

Table 5-1: Interpretation of the RM-ODP Viewpoints for SERVUS according to Usländer (ed.) (2007)

5.2.2 Relationship to International Standards

The specification of the RM-OA required a detailed consideration of the work of standardization bodies which resulted in a complex braiding (Usländer and Denzer, 2009) as illustrated in Figure 5-1 and explained below.

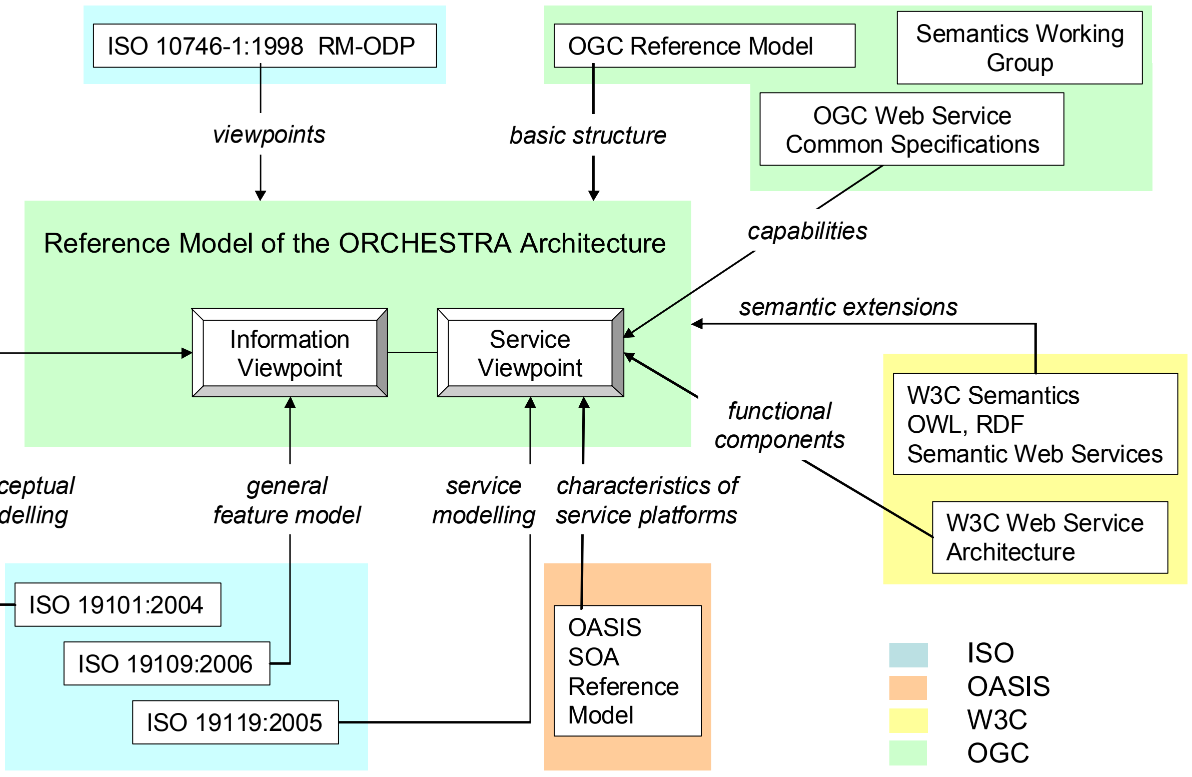


Figure 5-1: Influences of Standards to the Specification of the RM-OA

The following ISO and OGC standards heavily influenced the specification of the RM-OA models:

- The ISO/IEC 10746 Reference Model for Open Distributed Processing (RM-ODP) provided the structuring into **viewpoints** (as explained above).
- The OGC Reference Model (see section 3.2.3) influenced the **basic structure** of the RM-OA document and the usage of the pertinent ISO standards.
- The **conceptual modelling** of the RM-OA Information Viewpoint was performed according to the basic concepts (such as a "feature") of ISO 19101:2004 Geographic information - Reference model.
- The meta-model for information is an evolution of the **General Feature Model** as defined in ISO 19109:2005 Geographic information - Rules for application schema.
- The meta-model for services defined in the RM-OA Service Viewpoint is derived from ISO 19119:2005 Geographic Information - Services but harmonized with the meta-model for information (ISO 19109:2005) as described in section 5.3.
- The OpenGIS® Web Service Common Implementation Specification (OGC, 2005) details many of the aspects that are, or will be (because harmonization efforts are under way), common to all OGC Web Service interface Implementation Specifications. This idea was adopted for the specification of common

service characteristics in terms of reusable interfaces, for example, for the specification of their **capabilities**.

Furthermore, standards of the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS) were applied in the following way:

- Although not applied identically, the RM-OA meta-model for services reuses some of the concepts and their relationships as identified in the W3C Web Services Architecture (W3C, 2004a). The Web Services Architecture identifies the **functional components** and defines the relationships among those components necessary to achieve the desired properties of the overall architecture.
- The OASIS Reference Model for Service Oriented Architecture (OASIS, 2006) (see also section 3.2.4) specifies the common **characteristics** of so-called "SOA execution contexts", independent of a particular **service platform** implementation. The RM-OA assumes these characteristics as requirements for service platforms upon which a platform-neutral architecture may be mapped.
- Furthermore, there is ongoing research work in the field of **semantic extensions** of the Web as described in section 3.3.1. The RM-OA contains some initial considerations of how to integrate a semantic level into the hierarchy of the RM-OA models. Currently, there is no standardized architecture that unifies the approaches of OGC, W3C, and OASIS for spatial and non-spatial information in a harmonized and consistent way. There are partial solutions addressed by various projects, for example, in the context of the OGC, Semantic Web technologies were applied to geospatial applications in a Geospatial Semantic Web Interoperability Experiment (Lieberman (ed.), 2006). The RM-OA is a test case or architectural blueprint for such a harmonization activity.

In 2007 the RM-OA was accepted as a best-practices architectural framework for geospatial applications by the OGC Technical Committee (OGC document 07-097). Due to its generic and standards-based approach the author proposes it as foundation for system-of-systems engineering projects in the environmental science application domain. One recent example of RM-OA usage is the design and development of e-Science and technology infrastructures for biodiversity data and observatories in the LifeWatch project (Giddy et al, 2009).

The RM-OA was refined and extended towards a Sensor Service Architecture (SensorSA) (Usländer (ed.), 2009b) in the course of the European Integrated Project SANY (Klopfer and Simonis, (eds.) 2009). In addition to the RM-OA, the SensorSA includes the access to sensor observations (e.g. measurement values) and the management of sensors and sensor service networks. Sensors provide the input data for environmental monitoring as well as for risk management of natural, technical and man-made hazards. The SensorSA is based upon the services and information models of the OGC Sensor Web Enablement architecture (Simonis (ed.), 2008) but puts them into the context of the RM-ODP viewpoints.

5.2.3 Abstract Service Platform

The specifications of the Service and Information Viewpoint of the RM-OA and the SensorSA provide a specification of an abstract service platform that already contains a high percentage of the required functionality of an environmental information system. Their interfaces and services are listed in Table 5-2. They are based upon international standards but extend them where necessary. For illustrative purposes, they are organised in the following functional domains (see Figure 5-2):

- Services in the **Sensor Domain** cope with the configuration and the management of individual sensors and their organization into sensor networks. Examples are services that support communication between the sensors themselves, e.g. a take-over service in case of an impending sensor battery failure. Services in this domain are abstractions from proprietary mechanisms and protocols of sensor networks.
- Services in the **Acquisition Domain** ("AC" in Table 5-2) deal with access to observations gathered by sensors. This includes other components in a sensor network (e.g. a database or a model) that may offer their information in the same way (as observations) as sensors do. They explicitly deal with the gathering and management of information coming from the source system of type "sensor". The information acquisition process may be organized in a hierarchical fashion by means of intermediate sensor service instances (e.g. using data loggers).
- Services in the **Mediation and Processing Domain** ("MP" in Table 5-2) are specified independently of the fact that the information may stem from a source system of type "sensor". They mediate access from the application domain (see below) to the underlying information sources. They provide generic processing capabilities such as fusion of information, the management of models and the access to model results. In addition, services dedicated to resource discovery and event management are grouped in this domain.
- Services in the **Application Domain** ("AP" in Table 5-2) comprise the application-specific functions of the applications that are not yet covered by the generic service infrastructure underneath, e.g. in the area of decision support. Other examples are services that support the rendering and visualisation of information in the form of maps, diagrams and reports directly to the end-user in the user domain.
- The functionality of the **User Domain** is to provide the system interface to the end user. Usually, open generic architectures do not specify dedicated services for this domain. Usually this functionality is specified in dedicated implementation architectures that also take proprietary components, end-user devices (e.g., mobile phones) and products into account. A further important functional category is the personalisation of the application, i.e. the enablement of user role-specific views.

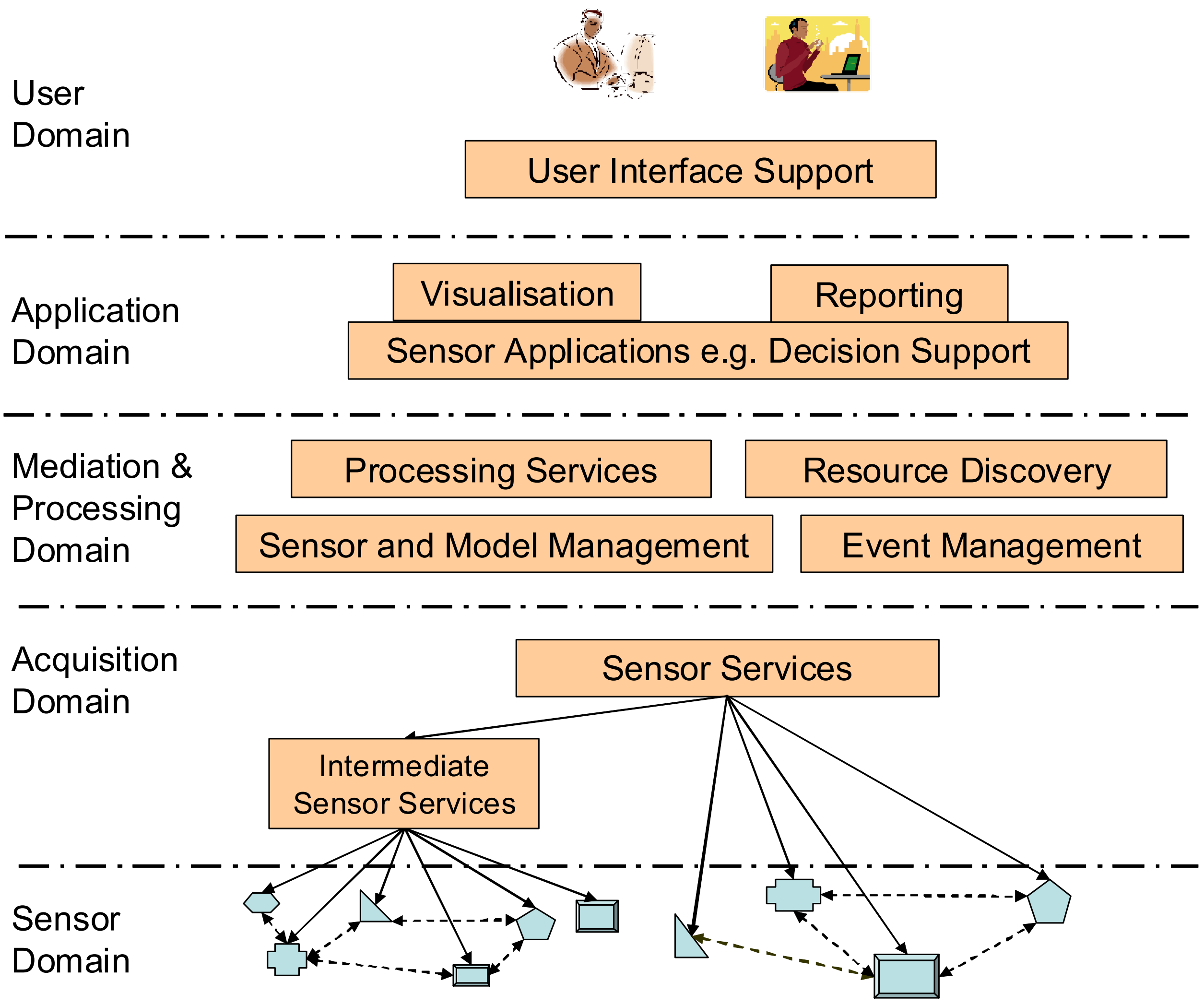


Figure 5-2: Functional Domains of an Open Sensor Service Architecture

In the SANY project a subset of this open sensor service architecture was mapped to a concrete service platform based upon the Web service environment of the W3C Services Architecture (W3C, 2004a). Its usability was demonstrated in three application pilots that covered the thematic domains of geo-hazards, marine risks and air quality (Klopfer and Simonis (eds.), 2009).

Name of Service and Interface Type [functional domain]	Description
Basic Interface Types [all]	Enable a common architectural approach for all architecture services, e.g. for the capabilities of service instances
Annotation Service [MP]	Relates textual terms to elements of an ontology (e.g. concepts, properties, instances).
Authentication Service [MP]	Proves the genuineness of principals (i.e. the identity of a subject) using a set of given credentials.

Authorization Service [MP]	Provides an authorization decision for a given context.
Catalogue Service [MP]	Ability to publish, query and retrieve descriptive information (meta-information for resources of any type. Extends the OGC Catalogue Service by additional interfaces for catalogue cascade management and ontology-based query expansion.
Coordinate Operation Service [MP]	Changes coordinates on features from one coordinate reference system to another.
Document Access Service [MP]	Access to documents of any type (e.g. text and images).
Feature Access Service [MP]	Selection, creation, update and deletion of features available in a service network. Corresponds to the OGC Web Feature Service but is extensible by schema mapping.
Map and Diagram Service [AP]	Enables geographic clients to interactively visualize geographic and statistical data in maps (such as the OGC Web Map Service) or diagrams.
Ontology Access Interface [MP]	Supports the storage, retrieval, and deletion of ontologies as well as providing a high-level view on ontologies.
Service Monitoring Service [MP]	Provides an overview about service instances currently registered within service network incl. status and load.
User Management Service [MP]	Creates and maintains subjects (users or software components) including groups (of principals) as a special kind of subjects.
Sensor Observation Service [AC]	Provides access to observations from sensors and sensor systems in a standard way that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors.
Sensor Alert Service [AC]	Provides a means to register for and to receive sensor alert messages.
Sensor Planning Service [AC]	Provides a standard interface to task any kind of sensor to retrieve collection assets.
Web Notification Service [all]	Service by which a client may conduct asynchronous dialogues (message interchanges) with one or more other services.

Table 5-2: Abstract Geospatial Service Platform for SERVUS (Usländer (ed.), 2007; Usländer (ed.), 2009b)

5.3 Conceptual Model

5.3.1 Overview

The conceptual model of SERVUS was developed in the course of the specification of the information and service viewpoints of the RM-OA (Usländer (ed.), 2007) and the Sensor Service Architecture (SensorSA) (Usländer (ed.), 2009b).

The SERVUS conceptual model is a meta-model following the principles of the model-driven architecture (MDA) (section 2.3.4). It specifies the relationships between the basic design elements such as features, interfaces, services and resources as UML meta-classes. This meta-model includes the resource model[23] as an extension of the meta-model for information and services defined by the OGC and the RM-OA for the information and service viewpoints, respectively (see Figure 5-3).

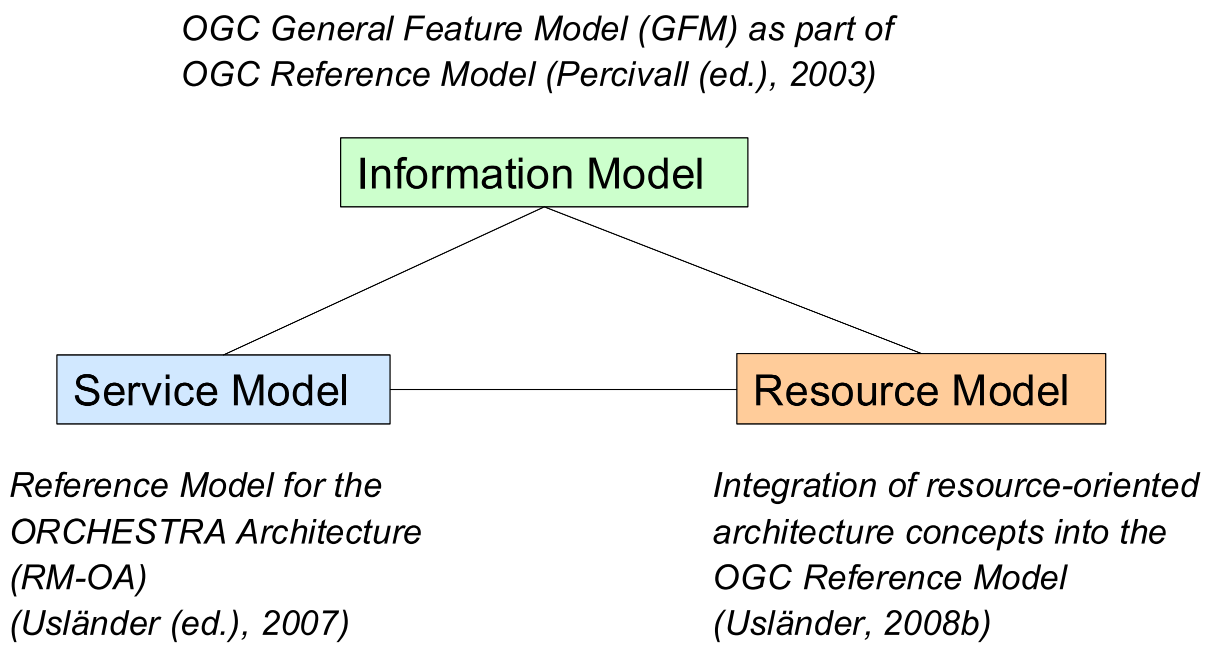

OGC General Feature Model (GFM) as part of OGC Reference Model (Percivall (ed.), 2003)
Information Model
Service Model
Resource Model
Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer (ed.), 2007)
Integration of resource-oriented architecture concepts into the OGC Reference Model (Usländer, 2008b)

Figure 5-3: Information, Service and Resource Model

For the specification of the meta-model for information and services the reader is referred to the full RM-OA specification as published by the OGC (Usländer (ed.), 2007) including the UML model diagrams. These diagrams involve an extended number of classes, attributes and relationships, resulting in an overall view that is obscured by the details. Therefore, Figure 5-4 illustrates the meta-model layer in a simplified form that elicits the relationships between the design elements. Furthermore, a UML specification of the resource model (Usländer (ed.) 2009b) is provided in section 5.3.2.

[23] This resource model specification has been submitted to OGC Architecture working group as a contribution to the discussion on abstract service models and the relationship between services and resources (Usländer, 2008b).

The basis for the modelling of the RM-OA Information Viewpoint is the ISO General Feature Model (GFM) (ISO 19109:2005). The modelling unit of the GFM is the concept of a **feature**. Features play a very important role in the design of geospatial applications as they represent entities in the universe of discourse of the users and stakeholders. In general, a feature is an abstraction of a real world phenomenon (e.g. a river or a forest). Features have **properties** which are usually **attributes** that describe thematic, spatial or thematic characteristics of a feature. Although rarely used in practice of designing GFM-based application schemas, features may also have **operation** properties. This allows a system designer to associate dynamic behaviour to features. Features may be associated to each other. This is expressed in terms of **role** properties of features. For instance, a feature "water body" may be associated to another feature "gauge" with the role "monitors" on the gauge side and the role "is monitored by" on the water body side. If required the act of "monitoring" may itself be modelled as an **association** feature in order to describe monitoring properties, e.g. to start/stop monitoring or to configure monitoring periods.

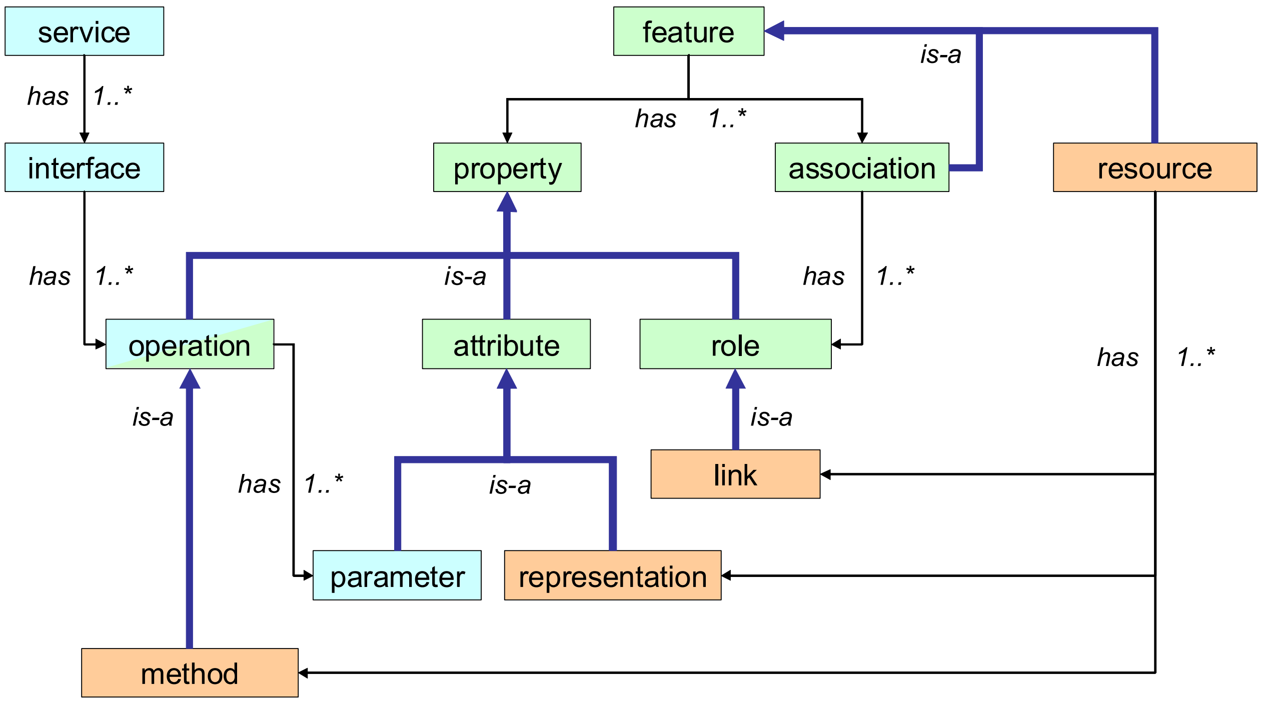


Figure 5-4: SERVUS Conceptual Model

The basis for the modelling of the RM-OA Service Viewpoint is provided by the two concepts **service** and **interface**. The modelling unit for services is the concept of an **interface**. A service may have several interfaces and one interface may be applied in several services. For instance, the meta-information of services may be specified in so-called capabilities interface which is common to all services. It delivers a self-description of a deployed service component. Examples are the OGC Web Service Common Implementation Specification (OGC, 2005) or the capabilities interface and the application schema for meta-information defined the Sensor Service Architecture (Usländer (ed.), 2009b). In such capability specifications also more sophisticated concepts (e.g., cost models) may be included such that negotiations between service requestors and providers are enabled based upon cost-benefit analysis methods (Schönbein, 2006).

An interface has one or more **operations** which in turn may have one or more (input and output) **parameters**. The operations and the parameters provide the link to the GFM as both may be properties of features.

A further modelling bridge between the information and the service viewpoint is the concept of a **resource** as a key modelling approach of SERVUS already introduced in section 4.2. On the one hand, it may be modelled as a feature whereby resource **representations** are attribute properties and **links** between resources are a specialization of role properties. On the other hand, the set of pre-defined **methods** that comprise the uniform interface of the resource model are specializations of operations in the service model.

5.3.2 Resource Model

The basic idea of this resource model is derived from the Representational State Transfer (REST) architectural style for distributed hypermedia systems as conceived by Fielding (2000). REST was used to guide the design and development of the architecture for the modern Web. Engineers of distributed systems usually discuss resource-orientation together with its realization in a Web environment, i.e., based on the basic technologies of the World Wide Web. Richardson and Ruby (2007) describe a resource-oriented architecture as a "way of turning a problem into a RESTful Web service: an arrangement of URIs, HTTP and XML that works like the rest of the Web". In contrast, the SERVUS Resource Model abstracts from the technological implications, and understands resource-orientation as a modelling approach to support the analysis and design phase.

The SERVUS Resource Model is an abstraction from the technological aspects of RESTful Web services. It is described in the following as a series of term definitions including references to the UML model shown in Figure 5-5. The description is restricted to those elements which are relevant for the understanding of the SERVUS Design Methodology. See Usländer (2008b) for the specification of the complete model.

There is a growing interest in REST-based service paradigms in the community that defines SOA design patterns[24] (Erl, 2008b). Five so-called "REST-inspired" candidate patterns were proposed by Balasubramanian (2008) with the aim of formalising the REST architectural style in the context of the SOA principles. The relationship of the following definitions with these candidate patterns is indicated below.

[24] The Internet site http://www.soapatterns.org/ is dedicated to the "on-going development and expansion of the SOA design pattern catalogue".

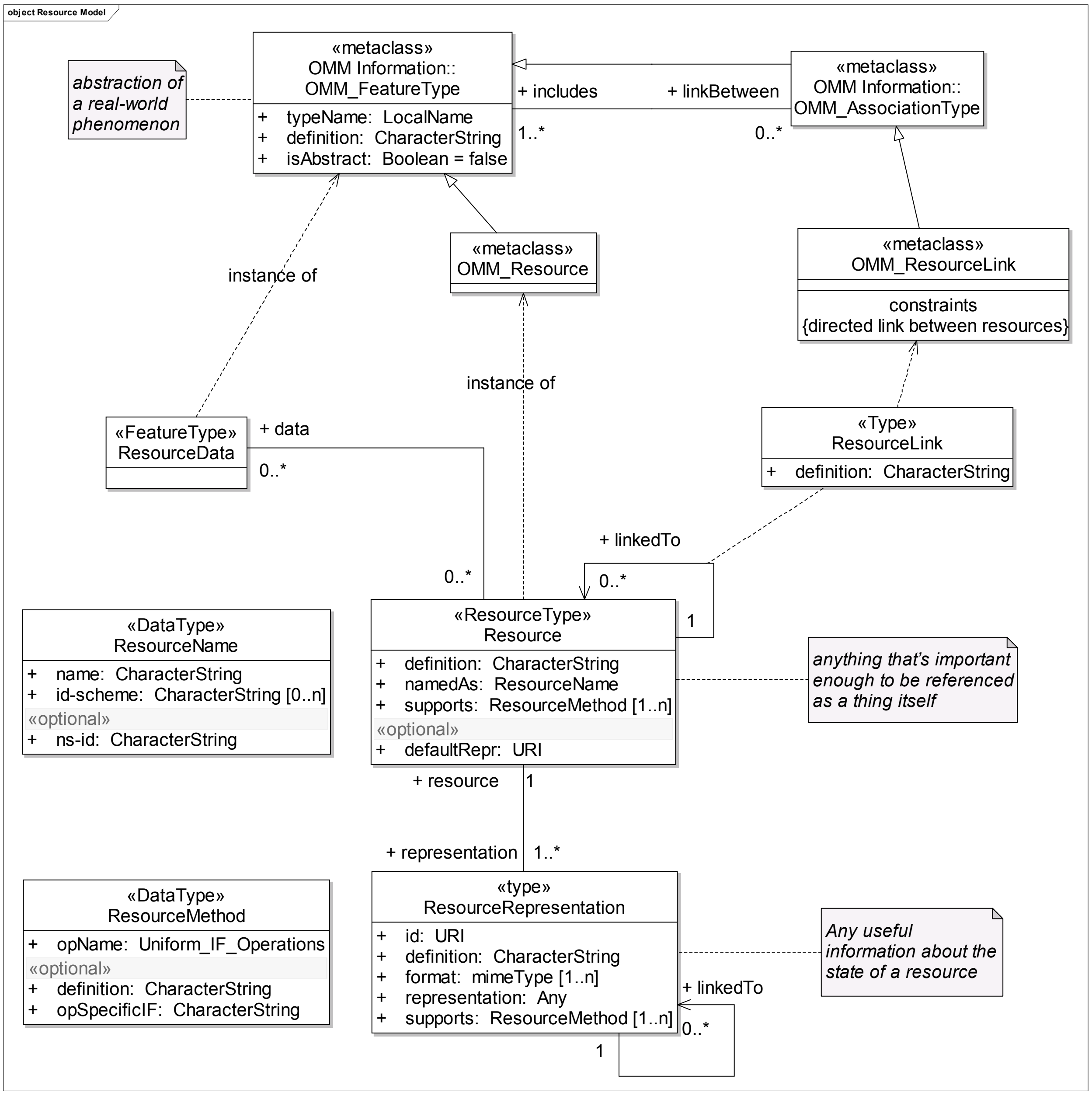

object Resource Model
abstraction of a real-world phenomenon
«metaclass» OMM Information:: OMM_FeatureType
+ typeName: LocalName
+ definition: CharacterString
+ isAbstract: Boolean = false
+ includes
+ linkBetween
1..*
0..*
«metaclass» OMM Information:: OMM_AssociationType
«metaclass» OMM_Resource
«metaclass» OMM_ResourceLink
constraints {directed link between resources}
instance of
instance of
«FeatureType» ResourceData
+ data
0..*
«Type» ResourceLink
+ definition: CharacterString
+ linkedTo
0..*
0..*
1
«DataType» ResourceName
+ name: CharacterString
+ id-scheme: CharacterString [0..n]
«optional»
+ ns-id: CharacterString
«ResourceType» Resource
+ definition: CharacterString
+ namedAs: ResourceName
+ supports: ResourceMethod [1..n]
«optional»
+ defaultRepr: URI
anything that's important enough to be referenced as a thing itself
+ resource
1
+ representation
1..*
«DataType» ResourceMethod
+ opName: Uniform_IF_Operations
«optional»
+ definition: CharacterString
+ opSpecificIF: CharacterString
«type» ResourceRepresentation
+ id: URI
+ definition: CharacterString
+ format: mimeType [1..n]
+ representation: Any
+ supports: ResourceMethod [1..n]
Any useful information about the state of a resource
+ linkedTo
0..*
1

Figure 5-5: Resource Model

<u>**Definition (5.1):**</u> A **resource** is anything that's important enough to be referenced as a thing itself. A resource has a unique identification. A resource is understood as a specialization of a feature type and has the following properties:

- *definition*: Human-readable description of the purpose of the resource.
- *namedAs*: Name of the resource. It is modelled by the type *ResourceName* which indicates the intended purpose of the resource to a human user. It has to be distinguished from the identifier of the representations of the resources which also provide an address (a path) by which to access the resource
- *supports*: Provides a list of those methods of the uniform interface that are supported by the resource.

- *providesView (optional)*: list of feature instances that provides the underlying data of the resource.
- *linkedTo:* list of zero or more resource types to which representations of this resource types may be potentially linked (modelled by the association class *ResourceLink).*
- *representedBy*: List of identifiers of possible representations of this resource. One resource may have one or more possible representations. This property is modelled by the type *ResourceRepresentation*.
- *defaultRepr*: identifier of the default representation of the resource.

Note: As a resource is modelled as a feature it may also contain one or more further attributes that define its thematic, temporal or spatial characteristics.

Definition (5.2): The **representation** of a resource comprises any useful information about the current state of a resource. A resource may have (and usually has) several representations. A representation of a resource may contain one or more links to another representation of the same or another resource.

It has the following properties:

- *id*: unique identification of the resource representation.
- *definition:* Human-readable description of the purpose of the resource representation.
- *format*: MIME-type format in which the information is presented to the client.
- *representation:* Information that is returned to the client when the representation is retrieved.
- *supports:* Provides a list of those methods of the uniform interface that are supported by the resource.
- *linkedTo:* Identifier of zero or more resource representations to which may be navigated from the resource representation.

This definition corresponds to the idea of the candidate SOA Design Pattern called "Alternative Format" (Balasubramanian, 2008).

Definition (5.3): The access to and the manipulation of resources is based on a **uniform interface** with commonly agreed, well-defined semantics.

This definition corresponds to the idea of the candidate SOA Design Pattern called "Uniform Contract" (Balasubramanian, 2008). The uniform interface is an instance of the meta-class "InterfaceType" (Figure 5-6).

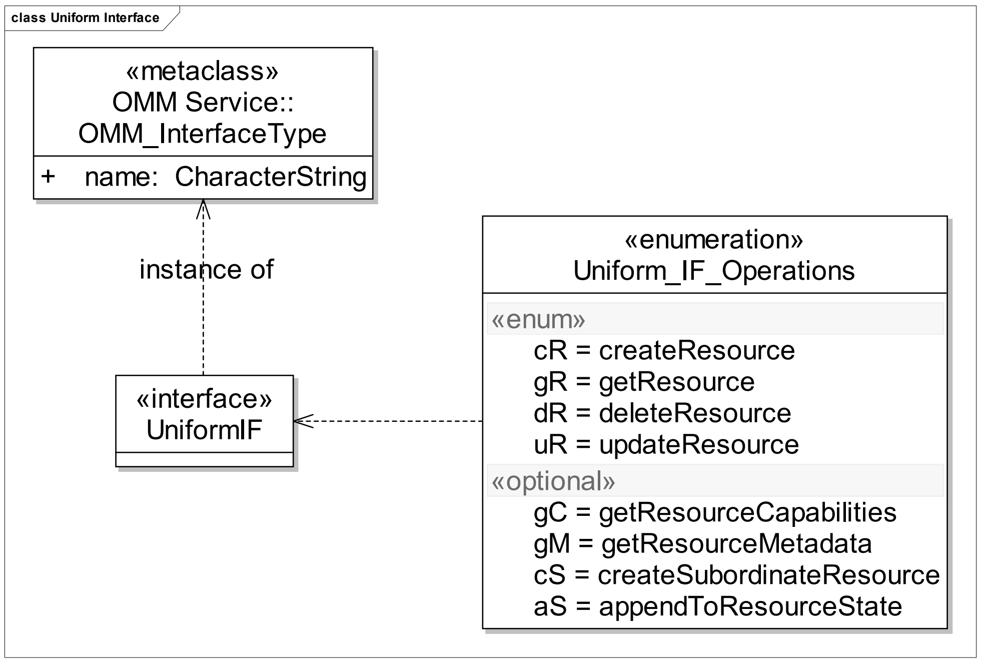


Figure 5-6: Model of the Uniform Interface

The following operations are mandatory for a given platform:

- *createResource*: create a new resource
- *getResource*: retrieve a representation of a resource
- *deleteResource*: delete an existing resource
- *updateResource*: modify an existing resource

The following operations are optional:

- *getResourceMetadata*: retrieve descriptive information about a representation
- *getResourceCapabilities*: check which methods are supported by a resource
- *createSubordinateResource*: create a resource representation in the context (e.g. namespace) of a given representation
- *appendToResourceState*: create additional information to the state of a resource

Definition (5.4): **A resource-oriented architectural style** is a coordinated set of architectural constraints that restricts the identification, characteristics and allowed links and methods of resources and their representations.

Furthermore, a resource may include references to semantic models in analogy to the semantic annotation of Web services and XML schema (see the SAWSDL approach described in section 3.3.3.1). Inspired by Toch et al (2008)'s definition of a semantic operation and a service network in the context of the semantic Web service framework OWL-S (see section 3.3.2), a Semantic Resource and a Semantic Resource Network are defined as follows.

<u>Definition (5.5)</u>: A **Semantic Resource** is a quadruple SR = ⟨P, M, Rep, l⟩ such that:

- P is a set of resource properties including a unique identifier. Examples of resource properties are:
  - The unique identifier of the resource. This is a mandatory resource property.
  - An indication if it is a requested or an offered resource. This is an optional resource property. If omitted, it indicates an offered resource.
  - Other attributes that characterize the resource type and hold the state of a resource instance.
  - Associations to other resources, thus forming a resource network in the form of a directed graph (see below). Such associations are used in the SERVUS Design Methodology to document the matching of requirements against capabilities, i.e. to match a requested resource with an offered resource.
- M is the list of supported methods of the resource.
- Rep is the list of representations of the resource.
- $l: P \cup SR \rightarrow SM$ is a labelling function that associates properties, as well as the semantic resource itself, with concepts taken from a semantic model SM.

Notes:

- The semantic resource definition is based on the terms defined by the W3C recommendation SAWSDL (Farrell and Lausen (eds.), 2007), see section 3.3.3. This means that the range of the labelling function are elements of a semantic model (called concepts), i.e. machine-interpretable representations of an area of knowledge or some part of the world.
- A semantic model may be an ontology, e.g., specified in OWL (section 3.3.1), or another conceptual model such as the Web Resource Space Model defined by Zhuge (2008).
- Not all properties must have a semantic reference.
- A property may also have more than one semantic reference, possibly with a different uncertainty value.
- As the semantic resource belongs to the domain of the labelling function l, too, a semantic reference may also be attached to the resource itself.

- An implementation of the labelling function l in an XML-based environment shall be based upon the SAWSDL *modelReference* for semantic annotation[25] as described in section 3.3.3.

A simplified version of the UML representation of a Semantic Resource is shown in Figure 5-7. A Semantic Resource inherits all properties of a *Resource* and adds an array of *conceptualLink* attributes which corresponds to the labelling function l introduced above. The *SemanticLink* basically comprises a Universal Resource Identifier (URI) to a concept in a semantic model.

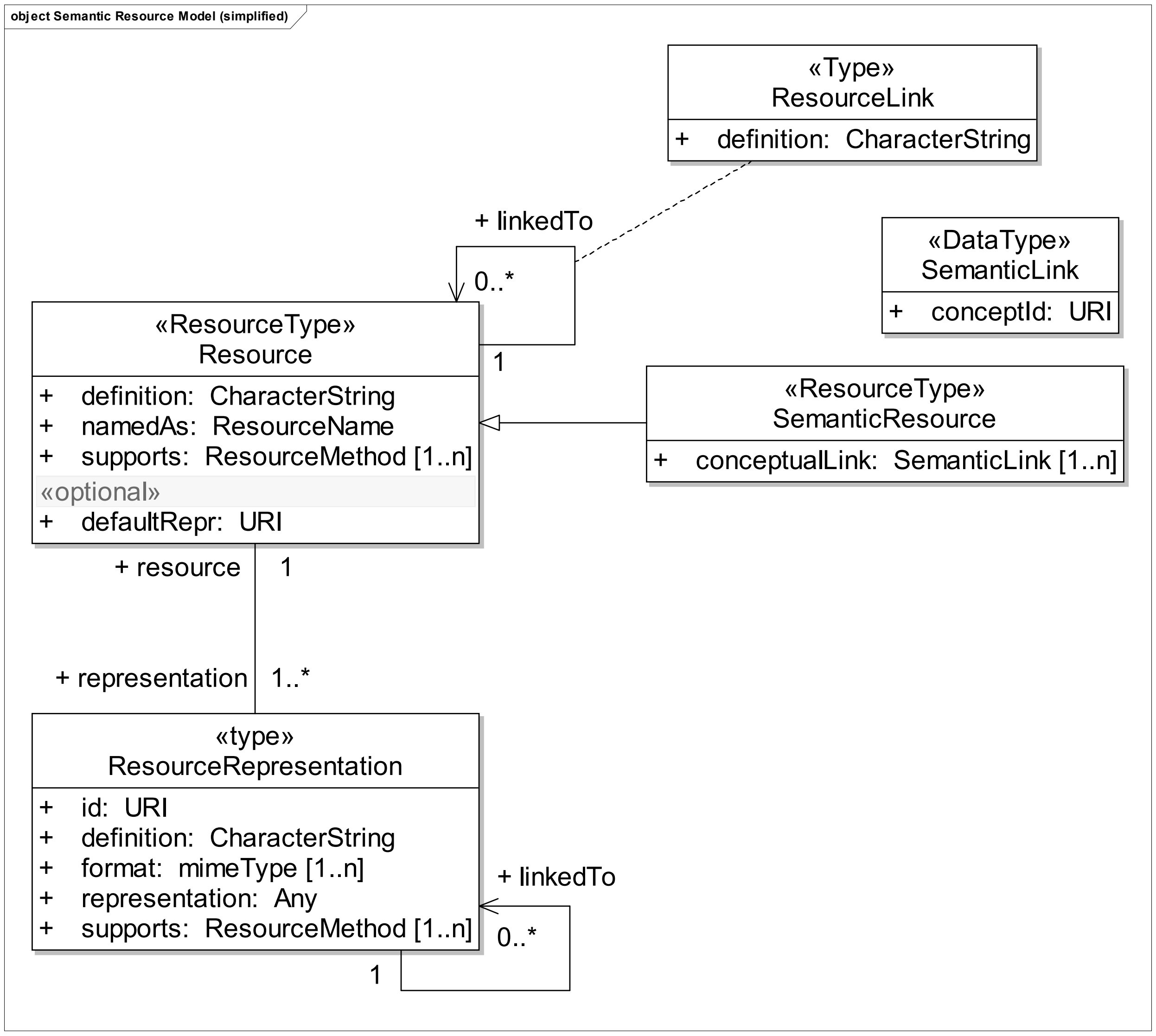

object Semantic Resource Model (simplified)
«Type» ResourceLink
+ definition: CharacterString
+ linkedTo
0..*
1
«DataType» SemanticLink
+ conceptId: URI
«ResourceType» Resource
+ definition: CharacterString
+ namedAs: ResourceName
+ supports: ResourceMethod [1..n]
«optional»
+ defaultRepr: URI
«ResourceType» SemanticResource
+ conceptualLink: SemanticLink [1..n]
+ resource
1
+ representation
1..*
«type» ResourceRepresentation
+ id: URI
+ definition: CharacterString
+ format: mimeType [1..n]
+ representation: Any
+ supports: ResourceMethod [1..n]
+ linkedTo
0..*
1

Figure 5-7: Semantic Resource Model

[25] Semantic annotation of services that follow the resource-oriented architectural style is also proposed by Gomadam et al (2008) in their work about SA-REST. However, they work on the concrete platform level and aim at semantically annotating RESTful Web services embedded in HTML-based Web page descriptions.

Finally, based upon the concept of a Semantic Resource the Semantic Resource Network (SRN) is defined. It is one of the major concepts of the SERVUS Design Methodology.

<u>Definition (5.6)</u>: A **Semantic Resource Network** (SRN) is a connected graph SRN = {D} such that

- D: SR × SR is a finite set of dependencies, namely directed relations, indicating relations between semantic resources (SR).

Notes:

- The relations are expressed by association properties of resources to which the labelling function l may be applied, i.e. the relations may be semantically annotated.
- In the SERVUS Design Methodology, the relations may consist between "requested resources" resp. between "offered resources" in order to represent structural relationships between the requirements resp. capabilities. However, they may also exist between requested and offered resources to represent the mapping of requirements to capabilities (Figure 4-5).

6 Specification of the Design Process

6.1 Multi-step Design Approach

According to Bieberstein et al (2006), a multi-step breakdown process across several layers of abstraction is necessary for an SOA design process (see section 3.4). The SERVUS Design Process follows an iterative approach to analyse the problem domain, express it in the form of user requirements, to map these to the capabilities of service platforms and to feed the results back into the next refinement step of architectural design. The artefacts resulting from the work in the design steps feed an architecture document. As illustrated in Figure 6-1 SERVUS distinguishes between an abstract service platform that is specified independently of a middleware technology and a concrete service platform that is based upon a chose middleware technology.

In the analysis phase, the "problem" is analysed together with the user and transformed into a set of requirements. The abstract design phase leads to platform-neutral specification following the rules of the abstract service platform provided by the SERVUS Reference Model in the form of meta-models (section 5). They represent the functional requirements (e.g. abstract service specifications), informational requirements (e.g. information model) and non-functional requirements (e.g. specification of the quality of service) of the problem domain.

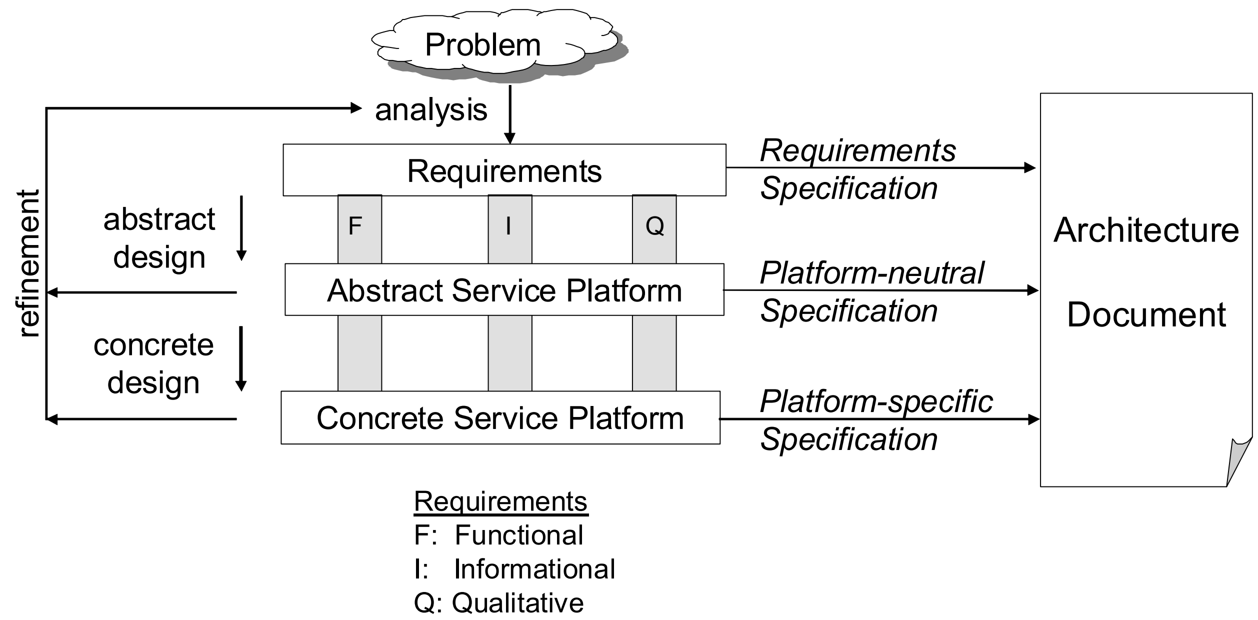


Figure 6-1: Multi-step Design across Service Platforms

The concrete design phase maps the abstract specifications to a chosen concrete service platform, e.g. the Web Services platform consisting of the rules of W3C (2004a), or OGC Web services and a profile of the OGC Geography Markup Language (GML) as the current mainstream service platform technologies for geospatial applications. Furthermore, platform-specific components are identified, arranged in implementation architectures and documented in platform-specific specifications taking into account also the qualitative requirements such as access control and dependability.

In practice, these individual phases are often interlinked and repeated in an iterative fashion. Sometimes the abstract design phase is not required in the first place. Furthermore, existing service and OGC service standards for Web services make a pure top-down approach improper. Thus, in practice, a middle-out design approach with a co-development of requirements and architecture is often the appropriate method (section 2.3.3).

Figure 6-2 presents an iterative SERVUS Design Process model in the form of an UML activity diagram that considers the edition of all viewpoints of the reference model (section 5) in an architecture document. This design process distinguishes between the design activities of problem identification, requirements analysis, capability analysis, abstract design and concrete design. The SERVUS Design Methodology described in this thesis (section 4) focuses on the abstract design. The refinement of the abstract design by means of the SERVUS design activities is contained in section 6.3.

6.2 Activities of the Design Process

A project starts with the **problem identification** activity. It analyses and describes the problem to be solved. This includes goals and objectives to be met (see section 2.1.1). Furthermore, this activity determines the type of documentation that is required during the architectural design. We assume that at least the edition of an **architecture document** is foreseen, and that this document is roughly structured according to the RM-OA viewpoints (see above). Further descriptions of the project context (e.g. the management environment of the project and its financial conditions) are outside of our scope.

The problem identification leads to the start of three design activity lines which are carried out in parallel:

1. The incremental edition of the architecture document (AD),
2. the analysis of the capabilities, and
3. the sequence of the requirements analysis and design activities, continuously fed by the results of the capability analysis and the latest version of the architecture document.

The explicit consideration of the **capability analysis** acknowledges the ambition of an SOA-based solution to re-use existing service-oriented capabilities and the need to co-develop requirements and architecture (see section 2.3.3). This approach provides both guidance and constraints: On the one hand, it reduces the design and development costs because existing and possibly deployed building blocks may be re-used, on the other hand, it restricts the freedom of the system architect in his/her design.

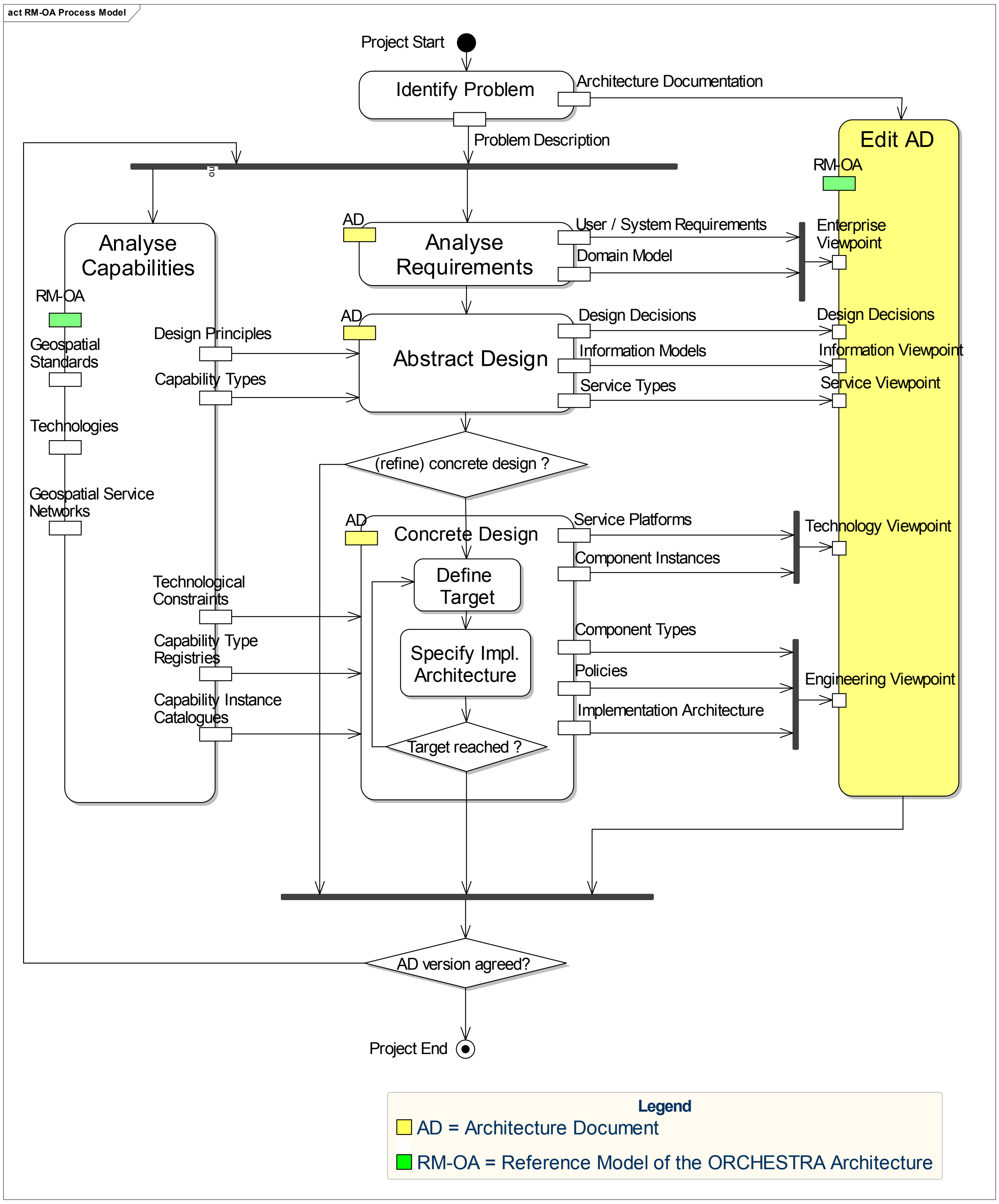


Figure 6-2: SERVUS Design Process Model

The input to the capability analysis in a geospatial SOA environment comprises the generic elements of the RM-OA, geospatial standards (e.g. interface and information model specifications), existing and emerging ICT and software technologies (e.g. Web service platforms), and the capabilities of deployed geospatial service networks (e.g. a satellite image ordering system) which may or even have to be integrated due to business requirements. Capabilities exist on conceptual, type, instance and technological level and feed the abstraction layers of the analysis, design and engineering activities, accordingly. Capabilities may be re-used as they are, but also adapted or tailored depending on their nature and the requirements.

Based upon the problem description and conceptual constraints it is the objective of the **requirements analysis** to refine the problem description into concise user requirements. Very often, the users or stakeholders of the system to be designed already prescribe a system environment. These should be explicitly written down as system requirements that encompass all constraints on technological and system level that are known at this time of the analysis and the design. The results of the requirements analysis shall be documented in the Enterprise Viewpoint of the architecture document.

The specification of the Information and Service Viewpoint resulting from requirements of the Enterprise viewpoint is the major task of the **abstract design** activity. Abstract here means that the service and information models are neutral with respect to a specific service platform and do not contain any particular dependencies on the peculiarities of a given platform. The RM-OA provides significant help in this design phase as it provides a generic modelling toolbox in terms of pre-defined but generic information and service types upon which the functional and informational user requirements may be mapped. Furthermore, experience in the edition of the RM-OA and the Sensor Service Architecture (Usländer (ed.), 2009b) showed that there is a need for a section in the architecture document that is dedicated to overall design decisions or major concepts before presenting the information and service models in detail. Examples of such design decisions are architectural styles and patterns used in the design of the system (e.g. request-reply interactions, event-driven processing, naming and identification of resources or management concepts).

Depending on the nature of the project or the current phase of the project the **concrete design** activity follows, or the next iteration step may be entered. The results of the concrete design are documented in both the Technology and the Engineering Viewpoint (see below). The concrete design starts with a definition of the target to be met in the concrete design step as usually only a subset of the service and implementation models identified on the abstract level are mapped to the concrete level in one iteration loop. The specification of the implementation architecture is tailored to this target. The individual tasks of the concrete design activity are:

- Specification of the characteristics of the concrete service platforms and the platform components (possibly including gateways between platforms) in the Technology Viewpoint.
- Specification of interfaces and characteristics of platform-specific component types in the Engineering Viewpoint. There are two ways for this task:
    1. Mapping the service and information models of the abstract design activity to the selected platform.
    2. Searching for fitting capability types in capability registries.

- Identification of the components according to the characteristics of the concrete capability types to be documented in the Technology Viewpoint. There are two ways for this task:
    1. Searching for instances of concrete capabilities deployed in operational geospatial service networks and use them as components in the system to be designed. An example is an instance of an OGC Sensor Observation Service (Na and Priest, 2007) that provides observation collections (time-series) about environmental phenomena (e.g. wind speed and wind direction) in a defined area. These instances may be found through an entry in a (geospatial) catalogue, or directly if they expose their capability documents as part of the interface of the capability instance.
    2. Specifying corresponding implementation components.
- Specification of policies for geospatial service network in the Engineering Viewpoint, i.e. guidance and rules how to combine components in a geospatial service network in order to fulfil a given task (e.g. access control or discovery).
- Specification of an implementation architecture of a geospatial service network as an arrangement of components in the Engineering Viewpoint.

6.3 Specification of the SERVUS Design Activities

This section specifies the SERVUS design activities **publishing**, **rephrasing**, **discovery** and **matching** in form of UML activity diagrams. These specifications refine the abstract design activity of the overall SOA Design Process. Therefore, these activities are carried out in a project and system context whose Enterprise viewpoint was already defined and documented in an architecture document. The semantic resource network that is set-up as part of the design model and filled by each of the following activities has to be considered specific to a project context. Note, however, that the result of the publishing activity may possibly be re-used across several design projects if they share the same service platform environment.

6.3.1 Publishing Activity

The publishing activity as introduced in section 4.3.3 represents the step in which the capabilities of the selected platform are entered into the design model. Concretely, this means to take the capabilities and publish them as offered resources according to the resource model. Three cases are to be distinguished:

1. The capabilities are not structured in a resource-oriented form. This is the typical case for OGC services as their standard capability schemata do currently not take resource-oriented approaches into account. This case is referred to as **OGCcap** case below.

2. The capabilities are already structured in a resource-oriented form, e.g. in terms of a Web Application Description Language (WADL) document (Hadley, 2006).
3. The capabilities are specified and implemented in a resource-oriented form, e.g. as RESTful Web services according to Richardson and Ruby (2007). RESTful geospatial services are currently scrutinized in the geospatial community. For instance, there is a proposal of a RESTful Sensor Planning Service by Cappelaere et al (2009) and a prototype of a RESTful geospatial catalogue service (ERDAS, 2009). Furthermore, there are ongoing discussions about the future role of the resource-oriented architectural style in OGC service specifications (Reed, 2009).

For the publishing activity cases 2 and 3 are discussed together. They are referred to as **RESTfulCap** case below. The publishing activity is illustrated as actions in the UML activity diagram in Figure 6-3. It comprises the following two steps whereby the OGCcap case is handled differently than RESTfulCap case:

Step 1: Publishing of Capability Types

Step 2: Publishing of Capability Instances

6.3.1.1 Step 1: Publishing of Capability Types

The publishing of capability types is a preparatory step before meta-information about capability instances may be harvested in step 2. This step is carried out in advance by the system designer for each known service type (see e.g. the service types specified in the abstract service platform). Examples of important service types used in this thesis are listed in Table 5-2.

Its major action is the definition of semantic resources derived from the capability schema of a service type (action *DefineRM*). The result is stored in the design model in an *EditDesignModel* action. Albeit recent discussions about semantic annotations of OGC Web Services (Maué (ed.), 2009), these capabilities are not yet available in a resource-oriented form for all geospatial service types. Thus, in order to prepare the publishing of capability instances in step 2, rules of how to map capability documents to the resource model have to be defined. These rules allow the designer to express capabilities of a service as offered resources in a semantic resource network. They are expressed in a so-called **capability lifting schema** by the action DefineLiftingSchema. Furthermore, the mapping shall maintain backward references from the offered resource to the original service operations. This backwards mapping is carried out in the action DefineLoweringSchema, defined in a so-called capability lowering schema and stored in the design model using the EditDesignModel action.

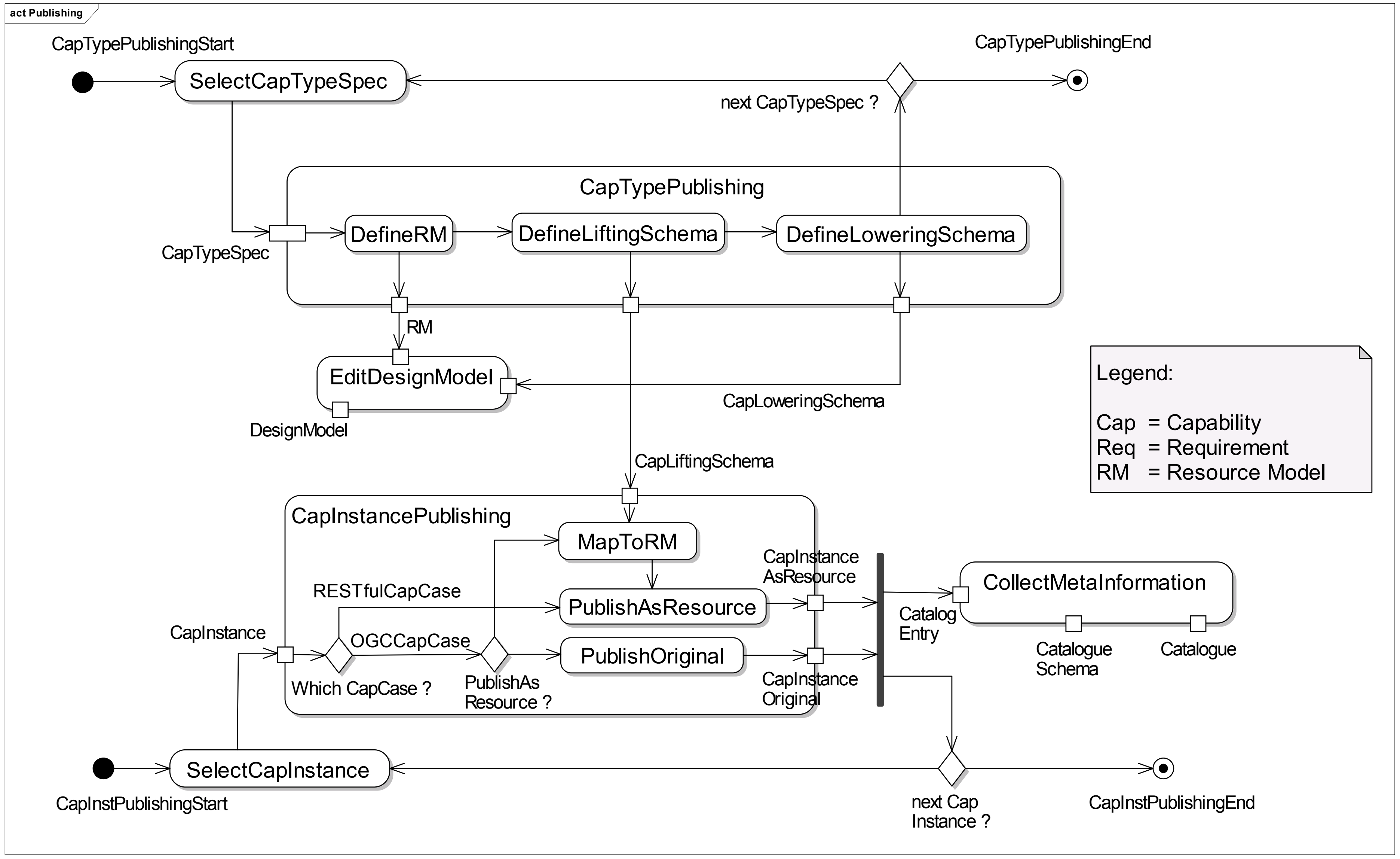


Figure 6-3: SERVUS Publishing Activity

The ultimate purpose of this mapping is that the discovery and matching activity (see section 6.3.3) may be performed in a uniform semantic resource network consisting of requested and offered resources (Figure 4-5).

6.3.1.2 Step 2: Publishing of Capability Instances

The second step is to retrieve meta-information about capability instances of a given geospatial service network and publish it in a geospatial catalogue system. Basically, either the services themselves call the publication operations of the catalogue and store their meta-information, or there is a harvesting application that reads out the capabilities of the services (e.g. by calling the *getCapabilities* operation of the OGC services) and writes them into the geospatial catalogue (action *CollectMetainformation* in Figure 6-5). In order to follow changes in the geospatial service network, this harvesting step has to be repeated (possibly in regular time intervals) in order to have the most current information in the catalogue (Hilbring and Schleidt, 2009).

The description of the detailed sequence of actions requires the distinction between the RESTfulCap case and the OGCcap case:

- In the RESTfulCap case, the service capabilities are already provided (solely or additionally) in a resource-oriented form. Thus, they may be directly stored in the catalogue assuming that its meta-information schema supports the resource model (action *PublishAsOfferSR*). If this is not the case, there is an alternative way of publishing as described in the note below.
- In the OGCcap case, the capability model of the services is not resource-oriented. In this case, the mapping rules of the capability lifting schema defined in step 1 have to be applied (action *MapToSR*) before the information may be stored in the geospatial catalogue in a resource-oriented form as in the RESTfulCap case above. Alternatively, the capabilities could also be published in their original form (action *PublishOriginal*). In this case, it must be considered in the formulation of the query as part of the discovery activity (see section 6.3.3).

Furthermore, if the source information is not complete, additional information may be added in external documents. An example scenario where this is necessary in practice is when capabilities of the OGC Sensor Web Enablement (SWE) services have to be exposed in an INSPIRE-compliant catalogue (Hilbring and Schleidt, 2009). In addition, the semantic references to ontological concepts must be entered by a human unless the resource descriptions contain already enough machine-readable information or even semantic references such that automatic semantic annotation is possible.

Note: In the RESTfulCap case there is an alternate way of publication. The resource-oriented capabilities could be made available to (Internet) search engines as Web resources in an adequate representation format (Figure 4-8). An example is a human-readable representation format such as an HTML page that may be accessed by issuing a HTTP GET operation to the URL of the Web service. However, this alternative is not yet considered in the SERVUS Implementation Architecture described in this thesis.

6.3.2 Rephrasing Activity

The purpose of the rephrasing activity as introduced in section 4.3.4.2 is to translate and relate the elements of the requirements expressed as use cases to the concepts of the design ontology. Its realization as part of the SERVUS Design Methodology is illustrated as an UML activity diagram in Figure 6-4. It is assumed that a use case consists of a set of actions that have to be carried out in a defined sequence (see the definition 4.1 in section 4.2.2).

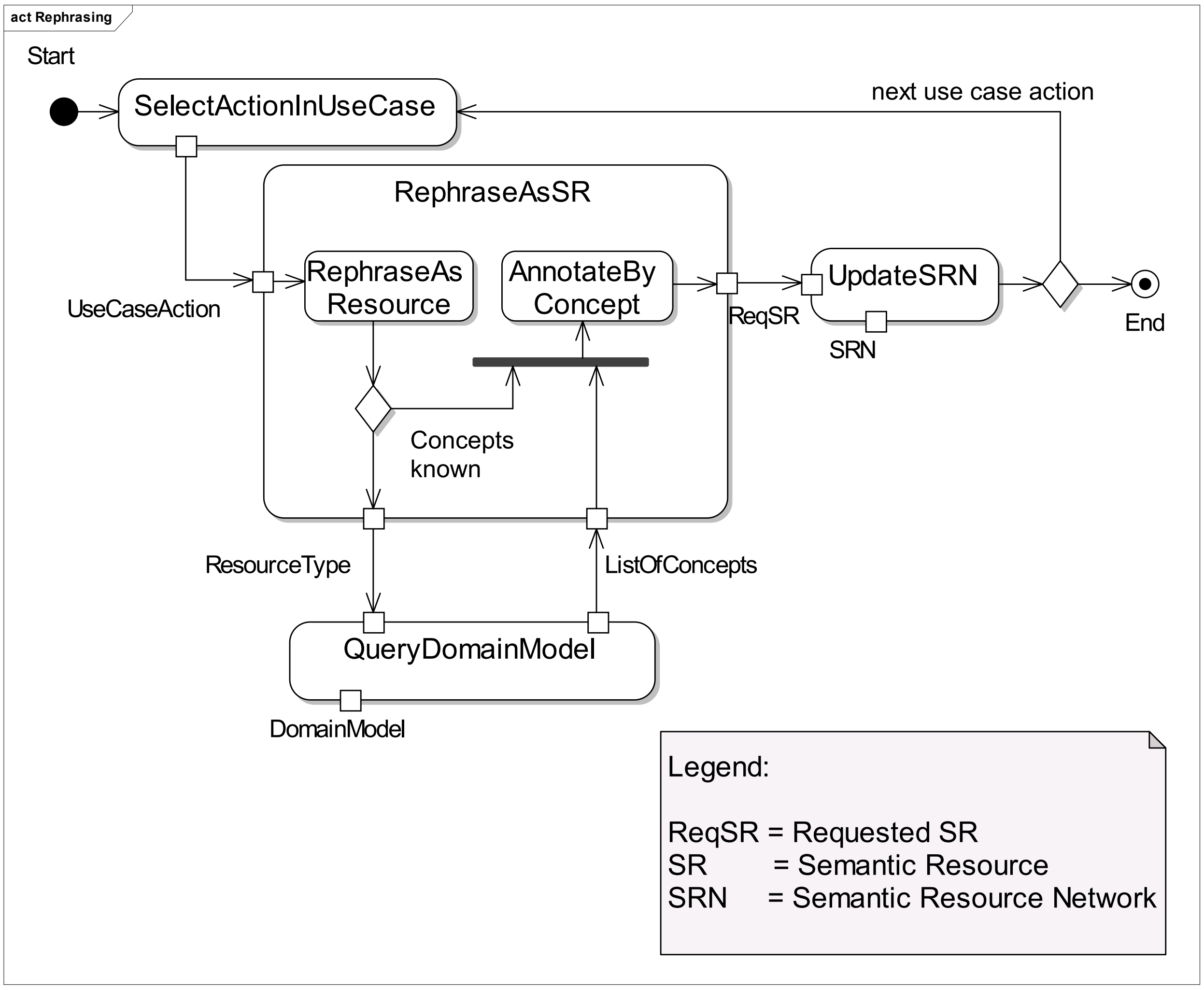


Figure 6-4: SERVUS Rephrasing Activity

The rephrasing starts with the reformulation of the use case actions to the structure of the resource model (action *RephraseAsResource*). This means that the use case actions have to be rephrased into a "normalized" form in terms of requested resources, resource representations and supported methods of the uniform interface. Furthermore, the resources may be annotated by links to concepts of the domain ontology (action *AnnotateByConcept*). If these conceptual links are not known, the domain model has to be queried for the best-fitting concept (action *QueryDomainModel*). Finally, the requested resources are then added to the semantic resource network (action *UpdateSRN*).

Rephrasing is an iterative activity that is carried out per action in each use case action (action *SelectActionInUseCase*) and finally leads to a semantic resource network that represents the set of requirements.

6.3.3 Discovery and Matching Activity

The discovery and matching activity as introduced in section 4.3.4 consists of two sub-activities:

1. The discovery sub-activity with the objective is to search for capabilities that are candidates to fulfil the requirements.
2. The matching sub-activity with the objective to evaluate the matching degree of candidates with the requirements.

6.3.3.1 Discovery Sub-activity

Discovery means to search for fitting elements in an SRN comprising a linked set of offered resources as result of the publishing activity (action *DiscoverSemanticResources*). Note that discovery in the traditional SOAD approach (see section 3.4) means to map elements of "service chains" or "workflows" to services in an SOA. The present resource-oriented approach is one possible answer to the recommendation in the outlook of Lutz (2007) to consider "higher-level more coarse-grained entities instead of complex and fine-grained service matching".

As illustrated in Figure 6-5 the discovery activity starts with the selection of a requested resource (action *SelectReqSR*) and its indication as such (action *MarkReqSR*). The next action is to discover and to select the type of the offered resource (action *DiscoverOfferSRType*) whose instances may be candidates for a subsequent matching[26]. The search for such resource instances is supported by a catalogue and therefore requires the formulation of a catalogue query (action *FormulateCatQuery*). This action needs information about the mapping from the resources to the service operations which is defined by the capability lowering schema (see step 1 of the publishing activity, section 6.3.1.1). The query is issued by the action *GetMetainformation.* If there are several candidates they must be ranked according to a ranking policy[27] (action *RankResultSetElements*). If the entries in the catalogue are not resource-oriented, the capabil-

[26] If the requested resource is of generic nature without detailed temporal, spatial or property requirements (e.g. the need to "retrieve sensor observations" instead of requesting a "nitrate time-series in the years 2002-2005 in the Upper Rhine Water Body") there is no need to search for instances. This case is modelled by a default instance the offered resource type that represents a generic implementation of the resource type.

[27] A future refinement may allow the system architect to configure the ranking policy. Ontology-based ranking of search results is a future research topic (section 9.3.2).

ity lifting schema defined in step 1 of the publishing activity (see section 6.3.1.1) shall be used in order to rephrase them into a resource-oriented form (action *LiftResultSet*). The resulting list of candidate resources is then added to the SRN in the action *AddCandSRtoSRN* (supported by the generic action *UpdateSRN*) and marked as offered resource in the action *MarkOfferSR*.

6.3.3.2 Matching Sub-activity

The objective of the matching sub-activity (action *MatchSemanticResources*) is to match candidate offered resources with the requested resources. The input to this sub-activity is provided by the SRN with resources marked as *requested* (result of action *MarkReqSR*) and *offered* (result of action *MarkOfferSR*).

For each selected offered resource (action *SelectOfferSR*) a matching degree is calculated (action *EvaluateMatchingDegree*). Once the best-fitting candidate has been selected, the mapping of the requested resource to the offered resource is documented in the Semantic Resource Network (action *CreateSRLink* supported by the generic action *UpdateSRN*). The documentation consists of creating a resource link from the requested to the offered resource.

This matching problem, and especially the sub-problem of calculating matching degrees between requested and offered resources, corresponds to the challenge of "mapping discovery" in the research domain of semantic integration (Noy, 2004): Given two ontologies, how to find similarities between them and to determine which concepts and properties represent similar notions. An example of an algorithm that assesses concept similarity based upon semantic distances in ontologies is provided by Ge and Qiu (2008). Noy distinguishes between two basic approaches: 1) using a shared ontology, and 2) using heuristics and machine-learning.

The first approach requires that a "general upper ontology is agreed upon by developers of different applications who then extend this general ontology with concepts and properties specific to their applications". Transferred to the semantic resource network in SERVUS, this means that the Reference Model plays the role of this upper ontology such that concepts of the Domain Model ("developed" by the user) are refinements of the generic concepts of a geospatial SOA of the Reference Model ("developed" by the system architect).

Techniques of the second approach often relies upon natural-language processing techniques which is not applicable to our matching problem as concepts in the Domain Model are linguistically not related to concepts in the Reference Model. The more promising techniques are those that aim at determining that a concept in one ontology (here: domain model) is a specialization of a concept in another ontology (here: the reference model) which is quite similar to the approach based upon a shared ontology (see above).

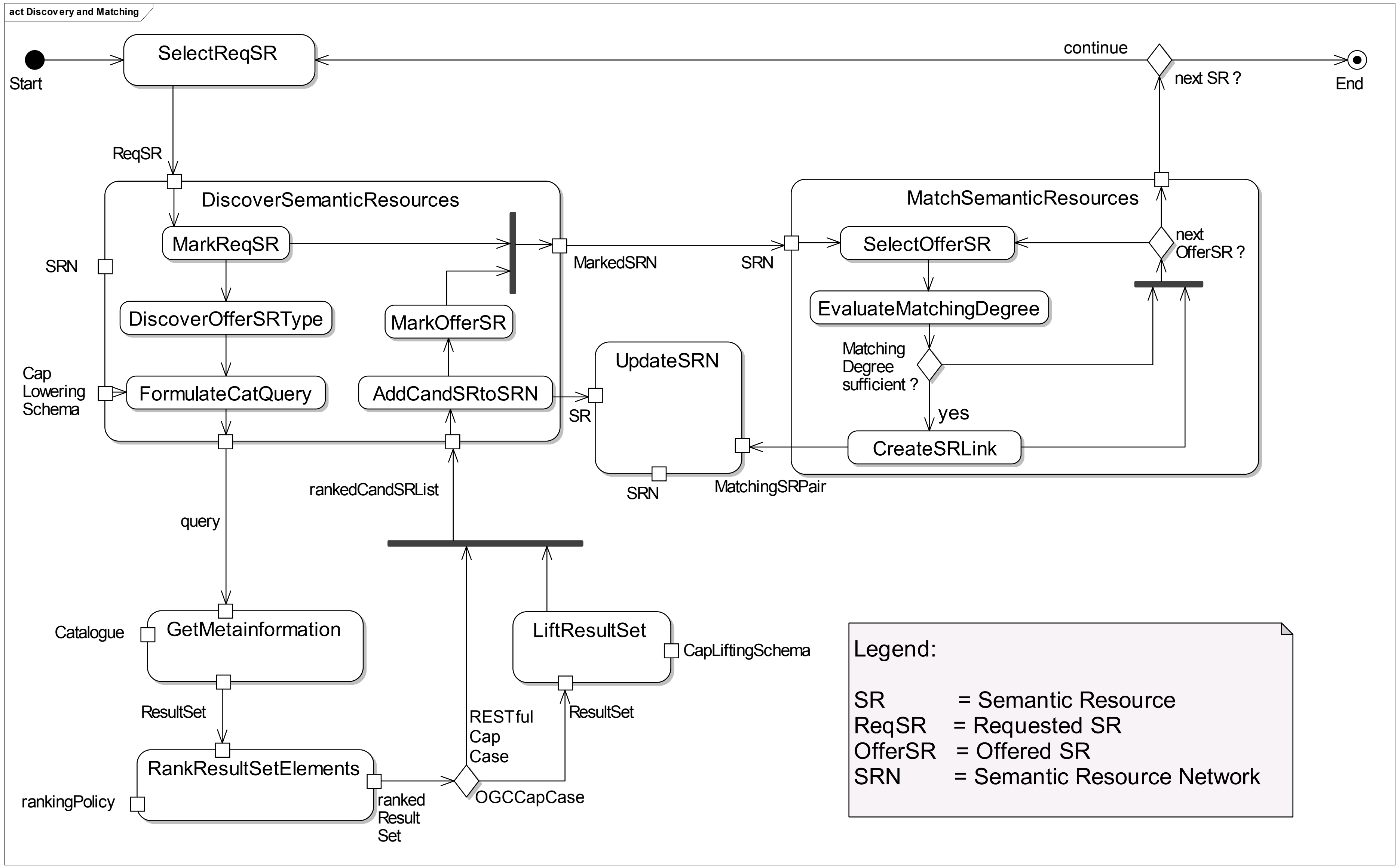


Figure 6-5: SERVUS Discovery and Matching Activity

Noy (2004) concludes that "tools for automatic and semi-automatic ontology alignment use the following definitions (to various extent)" (in SERVUS applied to the Resource Model):

- Concept names and natural-language descriptions (in SERVUS: names of resources, resource descriptions, use case resp. service descriptions from which the requested resp. offered resources were derived)
- Class hierarchy (in SERVUS: specialization and generalization relationships between resource types)
- Property definitions (in SERVUS: resource type properties, especially the links between resources and the resource representations)
- Instances of classes (in SERVUS: requested and offered resources)
- Class descriptions (e.g. in description logics) (in SERVUS: only applicable if the Domain Model is defined in description logics).

The investigation of adequate algorithms for the resource matching problem exceeds the scope of this thesis and may be a topic for future research (see the outlook section 9.3).

6.3.4 Documentation of the Design Process

The rephrasing, discovery and matching activity is carried out for each design step. At each time, the Semantic Resource Network (SRN) represents the state of the design process. It comprises

- the requested semantic resources representing the requirements as a result of the rephrasing activity,
- the candidate offered semantic resources, representing the available capabilities (services and their information models), and
- the links between requested and offered resources for those requested resources that are already matched with one or more offered resources.

As a very important side-effect this approach enables an instant documentation of the current state of the design process (design requirement R.11, section 1.4.3). With respect to the RM-OA Process Model and the structure of the architecture document in the SERVUS Design Process (see section 6.2 and Figure 6-2), the documentation of the SRN is split between the Enterprise and the Information Viewpoints:

- The offered semantic resources shall be documented in the Information Viewpoint (whereby the services and interfaces from which they are derived are part of the Service Viewpoint).
- The requested semantic resources shall be documented in the Enterprise Viewpoint (as they are rephrased user requirements).

- The links between requested and offered resources shall be documented in the Enterprise Viewpoint, too. The reason is these links turn the user requirements into **system requirements** that "add detail and explain how the user requirements should be provided by the system" (Sommerville, 2007). They may be "used by software engineers (in SERVUS: service engineers) as the starting point for the system (in SERVUS: service) design".

Furthermore, the SRN provides a means to realize the traceability of requirements and capabilities (R.8) which is an important aspect to generate a feedback to the user of how the requirements are fulfilled by the system (R.1).

7 Implementation Architecture

7.1 Overview

This section describes an implementation architecture of the SERVUS Design Methodology. The SERVUS Implementation Architecture and its core components are illustrated in Figure 7-1. It distinguishes four functional domains:

- User Interface,
- SERVUS Design Environment,
- Geospatial Services, and
- Source systems.

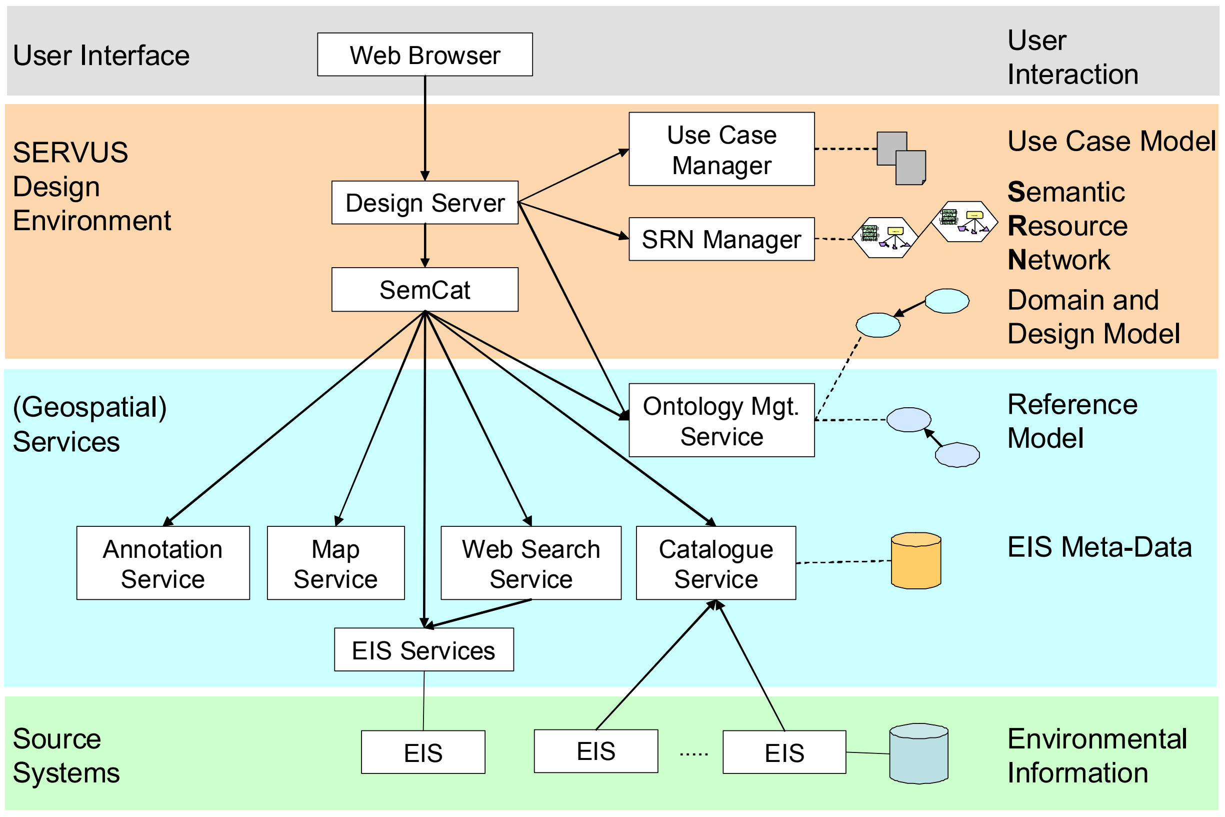


Figure 7-1: SERVUS Implementation Architecture

The **User Interface** comprises a standard Web browser and is the point of interaction of the user with the SERVUS Design Environment. The typical user role that is addressed and supported by SERVUS is the system architect or system analyst who is responsible for the architecture of the EIS to be built. His/her task is to mediate between the representatives of the relevant stakeholders who are experts in the application domain and express requirements, and the EIS developers who are experts in the programming and integration of the EIS components.

The **SERVUS Design Environment** represents the working environment for the user. It comprises four components: At its core is the *Design Server* which controls the execution of the rephrasing as well as the design and matching activities. It is a functional extension of the *Semantic Catalogue (SemCat)* through which it may access generic services, e.g. for geospatial search, semantic annotation, map rendering or ontology management. It uses a *Use Case Manager* to access and manage use cases and an *SRN Manager* to access and manage the semantic resource network that is incrementally developed during the design process. These components are described in more detail in section 7.2.

The functional domain of the **Geospatial Services** supports the activities carried out in the design environment. The types of these services are generic in the sense that they are not specific to the design process itself but belong either to the set of standard geospatial services of the OGC, or were identified and specified as architecture services for EIS applications and related application domains. Examples are contained in Table 5-2 of section 5.3.2 which lists the service and interface types defined in the RM-OA (Usländer (ed.), 2007). Thus, these services already comprise the set of capabilities of existing service platforms upon which the EIS applications to be designed may be built upon.

At the bottom of the SERVUS Implementation Architecture are the **EIS Source Systems** which offer the environmental information upon which the EIS applications rely. They are accessed through the geospatial services as described above. The environmental information of the source systems may either be offered through proprietary interfaces by components that are embedded into the EIS, or there are dedicated EIS services with interfaces that comply with the rules of the geospatial services domain.

Note: The SERVUS Design Methodology may be implemented in various forms. The implementation architecture presented in this thesis tries to re-use as much as possible multi-purpose modelling tools as well as generic components and services of the European research projects ORCHESTRA (Klopfer and Kannellopoulos (eds.), 2008) and SANY (Klopfer and Simonis (eds.), 2009). This implementation approach demonstrates the basic idea of SERVUS to exploit as much as possible existing capabilities for the realization of new use cases. The purpose and focus of the present implementation architecture is to demonstrate the benefits of the SERVUS design principles instead of providing a full-fledged design and development environment that may compete with SOAD environments on the market (see section 3.4) which, however, do not consider the side-constraints of geospatial standards.

7.2 Components of the SERVUS Design Environment

7.2.1 The Semantic Catalogue

The Semantic Catalogue (in short: **SemCat**) provides the foundation of the SERVUS Design Environment, especially in the support of the design and matching activity. It was specified and implemented as a semantically-enhanced geospatial catalogue service (see section 7.3) for environmental risk management applications (Hilbring and Usländer, 2008).

The main differences of the SemCat catalogue service and the OGC Catalogue Service are:

- SemCat provides two additional optional interfaces:
  - The semantic interface which provides basic means for ontology-based query extension and result assessment.
  - The management interface provides operations for the management of the underlying catalogue services of a cascading catalogue scenario.
- SemCat is not tied to a particular schema of a meta-information standard (e.g. ISO 19115 or ebRIM as defined by OASIS (2003)). Instead it supports application schemas for meta-information that are designed according to the rules of the meta-model for information (section 5.3).

The advantage of this approach is a higher flexibility:

- The meta-information schema may be adapted to the needs of a particular service network. For instance, for sensor-based applications there is a need to search for sensor types, observations and properties which is currently not supported in the standard meta-information schemas. These extensions were specified in the Sensor Service Architecture (Usländer (ed.), 2009b).
- It is possible to also include other catalogue types and search engines such as UDDI (OASIS, 2004), Internet search engines or catalogues of Earth observation data.
- It facilitates the adaptation to new search and discovery strategies and interfaces (e.g. OpenSearch) currently discussed under the ad hoc name CSW 3.0 in OGC.

Thus, an instance of SemCat may act as a portal catalogue with increased and innovative functionality that hides but integrates the possibly heterogeneous catalogue landscape underneath. Figure 7-2 shows the SemCat implementation architecture. It basically consists of a SemCat client (which is a Web-based application) and one or more SemCat servers that mediate a search request to underlying possibly heterogeneous geospatial catalogues.

The characteristics of SemCat are well suited to support SERVUS design activities:

- The semantic query expansion enables to broaden the search space for the discovery of offered resources that could potentially match the requested resources.
- The portal approach of SemCat, the support of cascaded catalogues and its mapping to various underlying search engines and geospatial catalogues enables to search for capability instances (resources) in several and possibly different service platforms. Furthermore, the management interface of SemCat

allows for a tailored usage of the catalogues cascade according to the scope and the needs of the user and the targeted applications.

- The annotation service (Bügel and Usländer, 2009) may be integrated in order to support the detection of resources in textual descriptions of applications, application processes and use cases and rephrase them to requested resources.
- Furthermore, the SemCat component provides a built-in harvesting functionality that may be used to support the publishing activity. Harvesting in the context of a geospatial catalogue is the procedure to retrieve meta-information about services, process it and store it as meta-information entries in a catalogue store. In an operational service network, the harvesting procedure shall be automated as much as possible in order to keep the entries in the catalogue up-to-date. It is either carried out periodically or event-driven, i.e. when a service indicates by an event notification that its capability document has changed. An implementation of a harvester for the SemCat is described by Hilbring and Schleidt (2009).

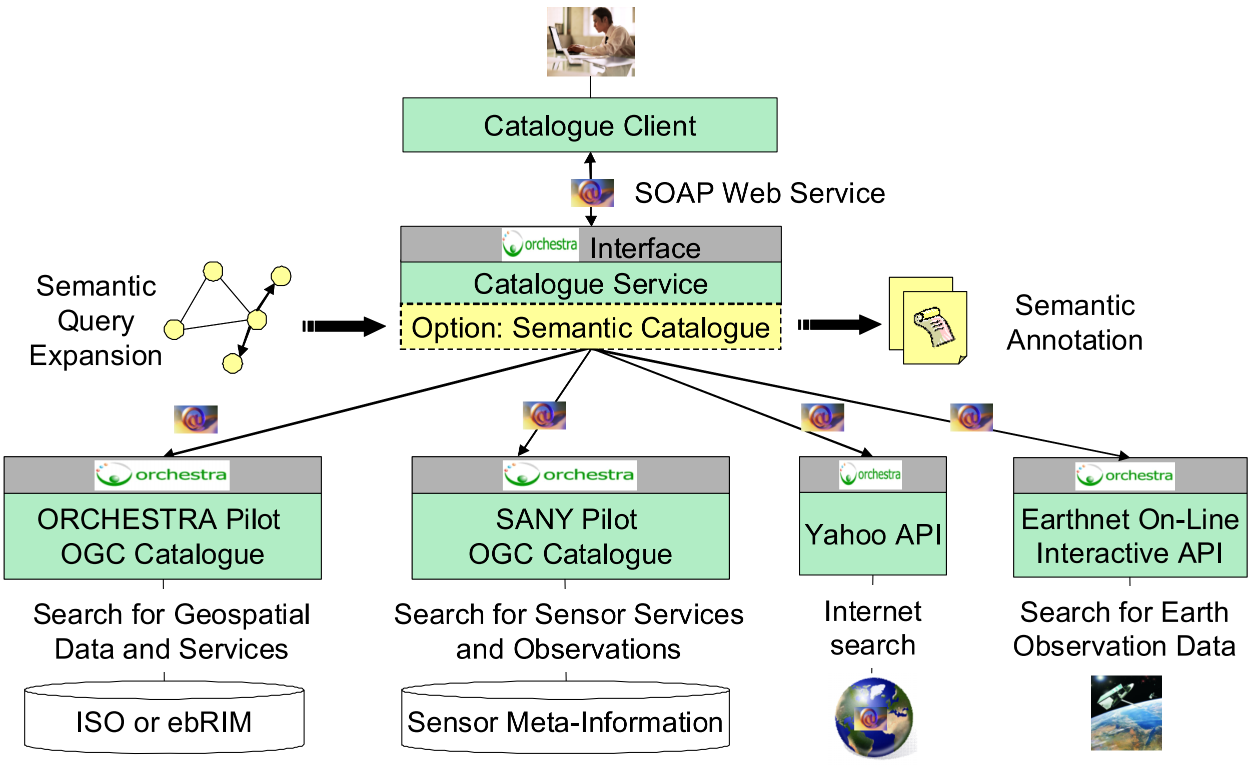


Figure 7-2: Architecture of the Semantic Catalogue (SemCat)

7.2.2 The Design Server

The Design Server is the central access point to the SERVUS Design Environment. Its objective is to control the user interactions and the execution of the SERVUS design activities. It also acts as the interface to the supporting actions such as the management of use case models, the management of the semantic resource networks and the management of the ontologies.

The implementation environment for the design server is a Web-based application that supports workflows, interfaces to Web services as well the management and persistent storage of information objects. In order to avoid the need to use multiple tools a SERVUS design server in a productive environment should also encompass the ability to manage ontologies on concept and instance level. This may happen by embedded functions or by calling ontology management operations through an API or through a remote service.

7.2.2.1 The Use Case Manager

The design server manages use case models. It is preferable to group use cases according to the structure of a project or the application domains covered by the project. Furthermore, each use case should have a unique identifier and some additional administrative information that fits to the management structure of the project. The precise definition of such administrative use case information is out of the scope of the present thesis. Furthermore, SERVUS does not require formal use case specifications (e.g. in UML) as it exploits textual use case descriptions delivered in structured format, sometimes also called "structured English".

For the SERVUS Implementation Architecture we adopt the basic structure of the methodology that was applied in the SANY project to document requirements and use cases (Williams (ed.), 2008). Note that, according to the process model described in section 6.2, this documentation belongs to the Enterprise Viewpoint of the architecture document.

We assume that the following information is available and documented as a result of the requirements analysis phase:

- Textual description of a specific problem to be solved, captured in a so-called **application process**. An application process comprises "a series of activities, events, decisions which take place in the overall system addressing a specific problem. This application process can be described by a workflow in terms of logic and interactions between the users (more generally, external actors) and the system. The term system means a combination of users, other stakeholders and machines."
- A list of use cases that are derived from the description of the application process.
- The description of the use cases in a common format comprising the following elements:
    - Scope: Short textual description of the use case.
    - Primary Actor: Role of a user that interacts with the system for this use case.
    - Stakeholder: Organization that has an interest in the solution of the problem.

- Pre-conditions: Conditions that must hold before the use case is carried out.
- Post-conditions: Conditions that must hold after the use case was carried out.
- Main requirements: Numbered sequence of actions of the use case.
- Extensions: Optional actions as a refinement of main actions.

An example of such a use case is given in Table 7-1.

Scope	
	A beach safety manager who is using the system will need to be able to select data sources which he would like to subject to a fusion process. He will need to be able to select the fusion process and will require notification that fused data has been added to the catalogue.
Primary actor	
	Statutory authority
Stakeholder	
	Beach Users *Data providers*
Preconditions	
	The user has access to charts and data sets from sensors in the area of interest
Post conditions	
	none
Main requirements	
1	*The user is able to select the data for fusion processing*
2	*The user is able to select the required fusion process from a list*
3	*The user is able to initiate the selected required fusion process*
4	*The system processes the data and adds the result of the fusion to the data catalogue.*
5	*When the data is ready, the system sets a flag to notify user.*
Extensions	
	none

Table 7-1: Example of a Use Case description according to Williams (ed.) (2008)

The Use Case Manager shall be capable of managing application process and use case descriptions according to this structure. However, the rephrasing activity requires two extensions:

1. For each action it must be possible to associate one or more resource types and resource instances according to the SERVUS Resource Model.
2. It is preferable to have an interface to the Annotation Service such that the textual description of an application process, the scope of a use case and an action description may be annotated with concepts of the domain model[28].

[28] In a fully integrated design environment the call to the Annotation Service shall be embedded into the Use Case Manager. However, a simple copy-and-paste mechanism to a text window whose contents may be annotated may also be sufficient.

7.2.2.2 The Semantic Resource Network Manager

The Semantic Resource Network Manager (SRN Manager) is the component that is responsible for the management of the Semantic Resource Network[29]. It shall offer the following means for the management of resources on both type and instance level:

- Create, read, modify and delete types and instances of resources.
- Search for resource types with given search criteria according to the resource characteristics.
- Search for instances of a given resource with given search criteria according to the resource characteristics.

7.3 Components of the Geospatial Services Domain

The idea of the geospatial services domain is to re-use existing generic services of an open geospatial service platform to support the design process. The following list that is also illustrated in Figure 7-1 provides a not exhausted overview of the most important service types and their use in the design environment:

- Catalogue Service: Geospatial catalogue services provide access to (meta-) information about available geo-spatial resources, e.g. topographical data or geostatistical processing capabilities. In contrast to Internet search engines such as Google or Yahoo, these catalogue services take the specifics of geospatial information processing such as spatial queries and geo-referencing of resources explicitly into account. Implementations of the OGC Catalogue Services Specification (Nebert, Whiteside and Vretanos (eds.), 2007) are widely used for this application field.

  The OGC Catalogue Service realizes the publish-find-bind pattern described in the OGC Reference Model (Percivall (ed.), 2008). This basic pattern supports the dynamic binding between resource providers and requestors because sites and applications may frequently change in a service network environment. In a design environment the Catalogue Service may be used to support the discovery of capability types and instances.
- Map Service: A Map Service enables geographic clients to interactively visualize geographic data. It transforms geographic data (vector or raster) into a graphical representation using symbolization rules. The main output of this service is an image document which may be a map or a thematic map that

[29] In a fully integrated design environment the SRN Manager shall be embedded into the design server including an interface to the Ontology Management Service. However, it may also be a separate component (e.g. a modelling tool such as an ontology editor) that is connected to the design server with a simple copy-and-paste mechanism.

visualizes spatial distribution of one or more statistical data themes. A standard example is the OGC Web Map Service (OGC, 2006).

In a design environment the Map Service may be used in the discovery activity 1) to define the area of interest in the visual context of other map layers, e.g. by drawing a geographic bounding box, and 2) to visualize the results of the search process in the visual context of a map.

- Ontology Management Service: The Ontology Management Service supports the storage, retrieval, and deletion of ontology modules as well as the access and the management of ontology elements. More sophisticated functions comprise the support of ontology reasoning (Baader et al, 2007). In the Design Environment the Ontology Management Service may be used to support the management of the models of the SERVUS Modelling Framework in case these are realized as ontologies (section 7.4). Furthermore, this service may support the search for concepts and instances within these ontologies.
- Annotation Service: The Annotation Service relates textual terms to elements of a semantic model, e.g. concepts, properties and instances. An example is the Annotation Service of the ORCHESTRA project (Bügel and Usländer, 2009). It may be used to support the SERVUS rephrasing activity by the detection of resource types and instances in textual descriptions of applications, application processes and use cases.
- EIS Service: This service category is a placeholder to call arbitrary functions provided by EIS source systems as services. Examples are calls to sensor observations provided by EIS in terms of services of the OGC Sensor Web Enablement architecture (Simonis (ed.), 2008). In a design environment they may be used to support the matching activity. In a validation step it may be checked if the capability description of a service (in form of an offered resource) is consistent with the service itself.

7.4 Implementation of the SERVUS Modelling Framework

7.4.1 Overview

The SERVUS Modelling Framework that comprises the use case, domain, design and reference models (section 4.2) is currently implemented as a modular ontology represented in OWL (section 3.3.1). This implementation approach is beneficial because wide-spread OWL-based knowledge management tools may be used to handle all models of the SERVUS Modelling Framework. For instance, the domain modelling, rephrasing and the publishing and discovery of capability types may be performed in a stand-alone ontology editor such as Protégé (Stuckenschmidt, 2009). Furthermore, the implementation of the modelling framework as ontology basically enables the easy integration of reasoning techniques (Baader et al, 2007) to support the SERVUS design activities, especially the discovery and matching activity.

Another implementation option is to seamlessly integrate the access and the management of this modular ontology into the SERVUS Design Environment presented in section 7.2. This approach enables the use of the semantic services of the geospatial service domain such as the Ontology Management Service or the Annotation Services.

7.4.2 Relationship to the Ontology Types of Kolas, Hebeler and Dean

Kolas, Hebeler and Dean (2005) propose a modular ontology for the representation of geospatial system concepts. They come up with the first five ontology types[30] shown and explained in Table 7-2. The third column of this table indicates the usage and meaning of these ontology types in the SERVUS Modelling Framework. As these ontology types do not cover the design aspects of geospatial systems an additional design process ontology that contains the concepts of an SOA design methodology is proposed. These may then provide the conceptual structure for the design artefacts generated during the design process. In the case of SERVUS, for instance, these concepts comprise the set of use cases and the requested resources as an artefact of the rephrasing activity and the mapping to the offered resources as an artefact of the discovery and matching activity.

[30] Kolas, Hebeler and Dean (2005) state that the use of the term "ontology" does not necessarily imply a specification in formal logics.

Ontology Type	Contents according to Kolas, Hebeler and Dean (2005)	Meaning in the SERVUS modelling framework
Domain Ontology	Thematic concepts and their relations (view of the expert)	dito (= domain model
Base Geospatial Ontology	Essential terms and concepts required to describe geospatial data and services.	dito (= reference model)
Geospatial Filter Ontology	Geospatial and logical relations between geospatial objects used in queries	dito (= reference model)
Feature Data Source Ontology	Schema of underlying databases, possibly converted to ISO/OGC feature instances	Types and instances of the capabilities in a geospatial service network published as offered resources in the design model
Geospatial Service Ontology	Geospatial services	
Design Process Ontology	---	Contains the concepts (design artefacts) of an SOA design methodology, e.g. use cases and requested resources

Table 7-2: Geospatial Ontology Types

7.4.3 Elements of the SERVUS Design Process Ontology

The SERVUS Design Process ontology comprises the following concepts and attributes:

- Application
    - list of actors (user roles) involved
    - list of application processes
- Application Process
    - List of use cases
- Use case (UC)
    - identifier
    - UC type (application UC and infrastructure UC, see section 4.2.2)
    - list of actors (= user roles)
    - pre-condition (optional)

    - post-condition (optional)
    - sequence of actions
- Action
    - “reference to subordinate UC” (optional)
    - list of requested resources
- Requested Resource
    - properties according to the SERVUS Resource Model (e.g. list of representations)
    - property “mapsTo”: link to offered resources that are candidates to fulfil the requested resource

7.5 Implementations of the Design Activities

7.5.1 Publishing activity

The SERVUS publishing activity as described on an abstract level in section 6.3.1 distinguishes between two sub-activities (see Figure 6-3): in the first place, to publish the capability types in resource-oriented form to the design ontology (*CapTypePublishing)*, and in the second place, to publish the capability instances to the catalogue (*CapInstancePublishing).* The current implementation architecture considers the “OGCCapCase”, i.e. the publishing of the capabilities in their original form to the catalogue.

7.5.1.1 *CapTypePublishing* Sub-Activity

The geospatial capability type model is provided by the system designer in the *CapTypePublishing* sub-activity which is performed in advance of the design process. The implementation of its three actions (Figure 6-3) is as follows:

<u>DefineRM (Resource Model):</u>

The purpose of the *DefineRM* action is to define a resource-oriented type model for the capabilities of the service types and store it in the design model using an implementation of the *EditDesignModel* action. Resource models for some of the OGC service types are currently defined, e.g., Cappelaere et al (2009).

As part of this thesis a resource model for the OGC Sensor Observation Service (SOS) (Usländer, 2008b) is proposed. This resource model is oriented at the OGC Observations and Measurements (O&M) model which is the basic information model for the access and management of observations originating from sensors of arbitrary types (Cox (ed.), 2007).

DefineLiftingSchema/DefineLoweringSchema:

These actions cover the definition of the information that is required to interpret the results of a catalogue query as an offered resource (lifting schema) as well as to support the system architect in the formulation of a query to a geospatial catalogue (lowering schema). The basic idea here is to link the offered resources to the "queryable" resources of the OGC catalogue service (lowering schema) as well as to maintain the backwards link (lifting schema). For instance, the offered resource "service instance" (e.g. an instance of the OGC Sensor Observation Service) may be linked to the queryable "type" with an indication of the name of the service type that is typically used in the catalogue entries (e.g. "SOS") as result of the *CapInstancePublishing* sub-activity (see next section).

7.5.1.2 *CapInstancePublishing* Sub-Activity

The purpose of the *CapInstancePublishing* sub-activity is to harvest the service meta-information and store it in the geospatial catalogue. In the "OGCCapCase" no mapping to the resource model is necessary, thus the standard transactional interface of the catalogue service (Nebert, Whiteside and Vretanos (eds.), 2007) may be used for this sub-activity. Here the SemCat component (section 7.2.1) is used to perform the *CapInstancePublishing* sub-activity.

7.5.2 Rephrasing activity

The rephrasing activity is performed per action for each use case. Each action has to be rephrased into a "requested resource" and entered into the semantic resource network (SRN). Based on the components described above the individual sub-activities of the rephrasing activity (see Figure 6-4) may be implemented as follows:

SelectActionInUseCase:

Use the Use Case Manager to first select a use case and then an action in the use case.

RephraseAsResource:

1. Enter one or more "requested resource" for the action. This step needs support to search in the SRN for a fitting resource type.
2. If no fitting resource type has been found, a new resource type has to be defined.
3. Fill out the properties of each requested resource
    a. Select resource method
    b. Provide resource method parameters (optional)
    c. Provide representation form. This step is supported by a list box that shows the possible representations of this resource type that have already been defined in the SRN. If no representation form is fitting, a new representation form is added.

AnnotateByConcept:

Provide references to fitting concepts of a domain model. This step may be performed manually or it may be supported by a query to the Ontology Management Service. The formulation of the query may be supported by the Annotation Service that searches for textual matches of the action description with concepts and instances in the design ontology.

UpdateSRN:

Finally, the requested resource is added to the SRN using the functions of the SRN Manager.

7.5.3 Discovery and Matching activity

The discovery and matching activity that is described on an abstract level in section 6.3.3 and illustrated in Figure 6-5 is carried out per requested resource in the SRN.

SelectReqSR:

The first action is to select the requested resource for which a mapping to an offered resource is to be investigated. This selection is typically based upon priorities set by the user and is supported by the functions of the SRN Manager. The discovery and matching activity is divided into the two sub-activities discovery and matching (action *MatchSemanticResources*) which may be implemented as follows.

7.5.3.1 ***DiscoverSemanticResources*** Sub-Activity

MarkReqSR

First, the requested resource that is handled by this sub-activity is marked using the functions of the SRN Manager.

DiscoverOfferSRType

The discovery of the type of the offered resource that fits to the requested resource is supported by the query functions of the SRN Manager. In case the SERVUS Modelling Framework is implemented as a modular ontology (section 7.4), the ontology editor or the ontology management service may be used for this action.

FormulateCatQuery

The formulation of the catalogue query is determined by the *CapLoweringSchema* associated with the type of the offered resource. It is carried out by the design server based upon the functions of the SRN Manager. The query is then issued to the SemCat component by calling the *GetMetainformation* action.

The SemCat delivers a result list whose elements are ordered according to the ranking policy used (action *RankResultSetElements*). It may range from a simple classification of the elements according to the ontological concepts (as currently implemented) up to sophisticated ontology-based ranking algorithms (a future research activity as discussed in the outlook section 9). In the “OGCCapCase” the result set has to be transformed into an offered resource according to the *CapLiftingSchema*.

<u>AddCandSRtoSRN and MarkOfferSR</u>

The offered resources that have been discovered are added to the SRN and marked as such using the functions of the SRN Manager (action *UpdateSRN*).

7.5.3.2 ***MatchSemanticResources*** Sub-Activity

The matching sub-activity starts with the SRN that contains marks of both the requested resource and the candidate offered resources. Currently, this computation of the matching degree (action *EvaluateMatchingDegree*) is not automated, i.e. the selection of the best-fitting offered resource is carried out on a manual basis. In case of the implementation of the design model as an ontology, the computation of the matching degree corresponds to the problem of concept similarity in ontologies, see e.g. Ge and Qiu (2008).

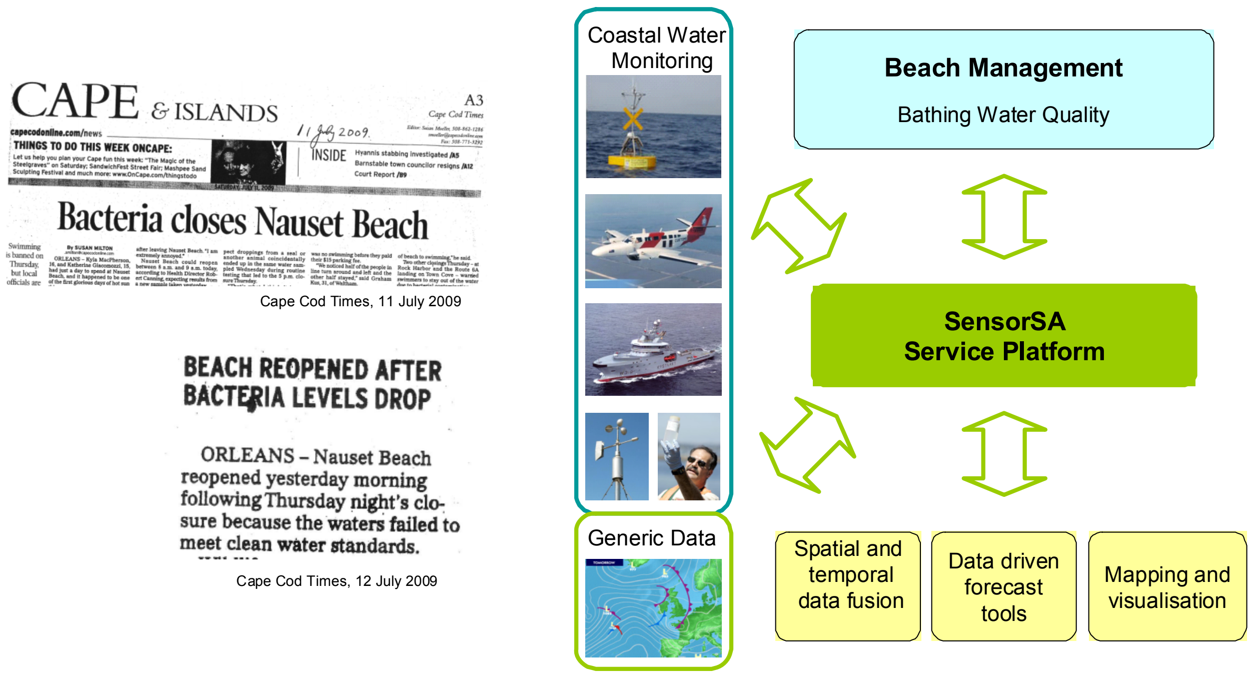
<u>CreateSRLink</u>

Finally, the link between the requested resource and the selected offered resource has to be established. This action is supported by the functions of the SRN Manager.

8 Application Example

8.1 Introduction

This section provides an example of the use of the SERVUS Design Methodology for a selected environmental information system. The information system addresses the marine risks application domain, in particular the quality of bathing water and related beach management issues (Figure 8-1).


CAPE & ISLANDS

Bacteria closes Nauset Beach

Cape Cod Times, 11 July 2009

BEACH REOPENED AFTER BACTERIA LEVELS DROP

ORLEANS – Nauset Beach reopened yesterday morning following Thursday night's closure because the waters failed to meet clean water standards.

Cape Cod Times, 12 July 2009
Coastal Water Monitoring
Generic Data
Beach Management
Bathing Water Quality
SensorSA Service Platform
Spatial and temporal data fusion
Data driven forecast tools
Mapping and visualisation

Figure 8-1: Marine Pilot Application (derived from Williams (ed.), 2008)

This application domain was tackled by the European Integrated Project SANY and implemented using the geospatial service platform defined by the Sensor Service Architecture (SensorSA) (Usländer (ed.), 2009b). The marine risks application domain is motivated by Williams (ed.) (2008) as follows:

"The application domain concerns the management of marine risks in coastal waters, using sensor networks coupled with web services in a way to improve decision-making processes and respond in the most efficient way to risk arisings. Marine risks can have many different causes: natural events, man-activity causes and a combination of both. In almost all cases, such marine risks have an economic, human and environmental impact. The selected application themes concern both natural and human geo-hazards."

For this example the application theme **Bathing Water Risk Management** is selected:

"Supporting the reduction of hazards to public health through contamination of bathing waters. Such waters are subject to extensive regulatory standards, established via an EU Bathing Water Directive which is itself currently undergoing revision. Failure to meet regulatory stan-

dards can have significant impacts on tourism revenues, as well as on the health of individual citizens.”

The requirements are available in form of use cases which are specified in structured English according to a common template (see Table 7-1). In total, this application theme comprises 19 application use cases categorized into three application processes:

1. Forecasting risk of water contamination exceeding threshold level.
2. Assessing quality of forecasts to determine robustness of decision-making.
3. Access all data to select action required.

All three application processes were analysed for this application example. However, in order to reduce the complexity, this example focuses on the first application process which is further described as follows by Williams (ed.) (2008):

“Some bathing beaches suffer periodic incidents of poor water quality due to microbial contamination of bathing waters, while others suffer periods when water quality falls from good to merely satisfactory. These incidents are usually caused by run-off or overloading of urban waste water treatment works after heavy rainfall. Where diffuse pollution sources are the cause, several run-off locations may be suspected, but the actual point of contaminated run-off is likely to vary from incident to incident.

The statutory authority needs to improve its ability to forecast these incidents in order to avoid failure to meet water quality criteria defined in the Bathing Water Directive. The authority is required to take bathing water samples on pre-specified dates during the bathing season. In order to minimize the risk of exceeding a threshold, the authority must decide by 0900 on every sampling day whether to close the beach in order to ‘discount’ the sample from its annual compliance assessment. This decision can only be based on a forecast of the risk of adverse water quality on the day in question.

Without a forecasting tool, the decision to close a beach must be based on well-informed guesswork. In some situations, it may be possible to use an in-situ assay technique (…) to measure microbial contamination levels, but this takes 24 hours to achieve a result. This method is therefore only useful to confirm a contamination incident retrospectively.”

The use of SERVUS is presented in terms of the three major design artefacts resulting from the SERVUS design activities (Figure 4-4):

1. The **Domain Model** for this application domain derived from the textual application process and use case descriptions, and
2. The **Reference Model** as the foundation of the design activities,
3. The **Design Model** resulting from the publishing, the rephrasing and the discovery and matching activity. The design model encompasses
   - the network of requested resources derived from the use case actions,
   - the network of offered resources derived from the reference model, and
   - the mapping of the requested to the offered resources.

All three models were implemented in form of OWL ontologies. Excerpts of them are described and visualized in the following sections[31].

8.2 Domain Model

The domain model was generated by analyzing the major thematic terms and their relationships in the textual descriptions of the application theme and the use cases. Basically, the resulting domain model distinguishes between the **actors** and the **marine risk** concepts. The actor concepts are presented in Figure 8-2 and comprise the following list:

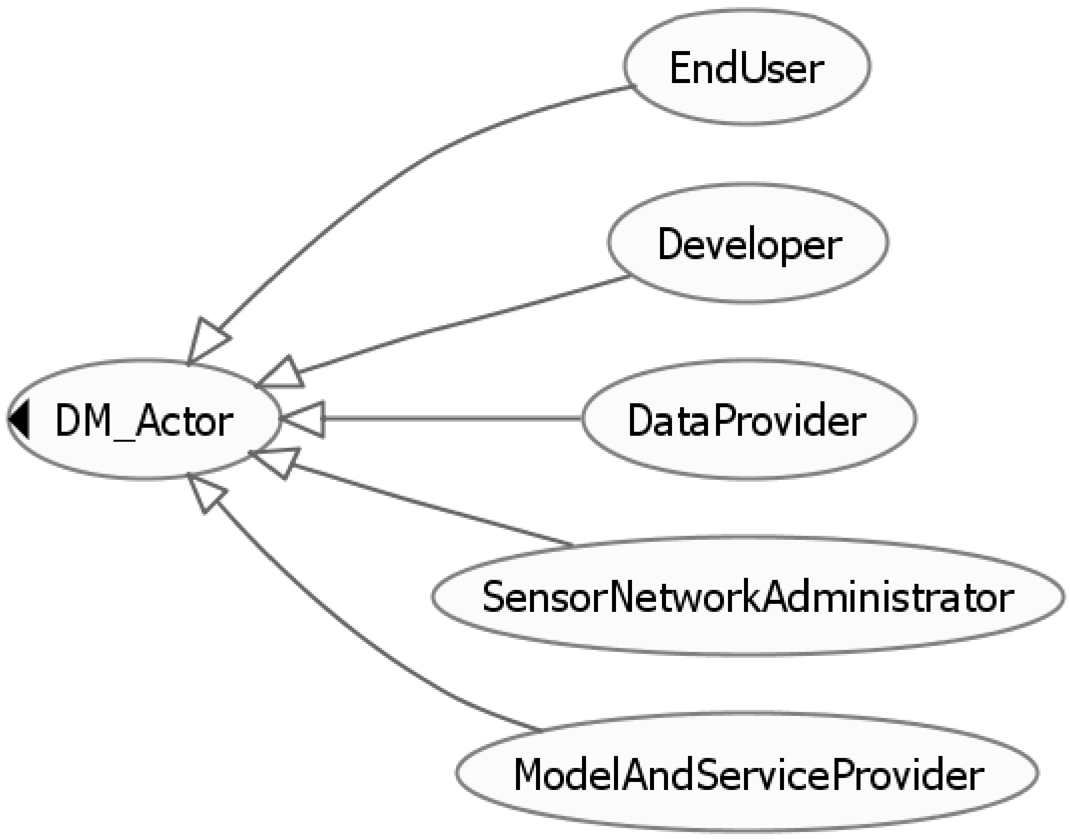


Figure 8-2: Actors in the Bathing Water Risk Management Application

- **End user**: Organisations benefiting from use of the system, but not involved in configuring the system. Such users might include: urban or regional authorities responsible for public health in specific waters, or a government department responsible for legal compliance of a member state with the Bathing Water Directive.

[31] The conceptual hierarchies were rendered by the OWLviz tool built into version 3.4.1 of the OWL editor Protégé, whereas the RDF Gravity tool (RDF Graph Visualization Tool) was used for the visualisation of the graphs (Goyal and Wesenthaler, 2004), however, partly annotated in order to increase the readability.

- **Developer**: Organisations configuring and running systems on behalf of themselves or end user clients.
- **Data provider**: Users who are providing or selling data through the application. They make a commercial use of the application.
- **Sensor network administrator**: Users who administer the application and the underlying system. They need a management view upon sensors or sensor networks.
- **Model and service provider**: Users who are providing or selling models or Web services. They can use their own data or process external data for their developments.

Most of the actors in the use cases are "statutory authorities" which fall into the actor class "end user". Excerpts of the domain ontology of the bathing water risk application domain are contained in the following figures[32].

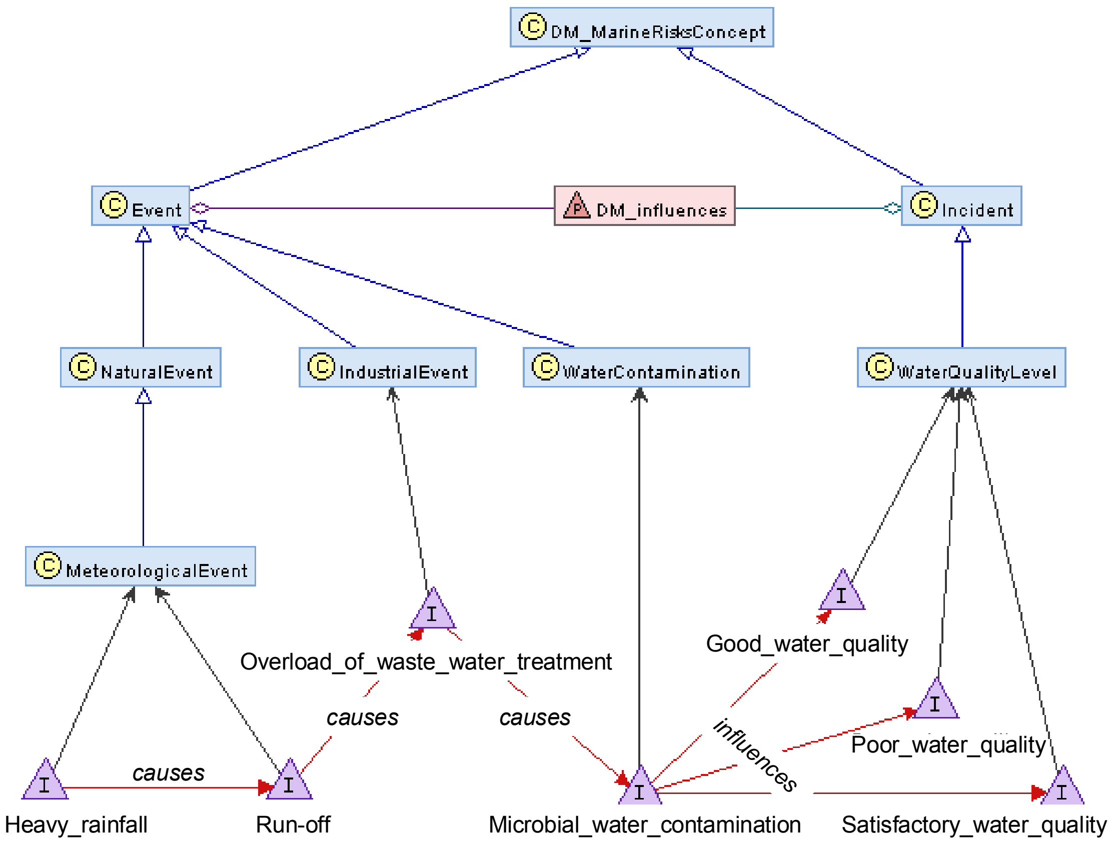


Figure 8-3: Ontology for the Application Theme "Marine Risks – Bathing Water Quality" focussing on Events and Incidents

[32] RDF Gravity supports the visualisation of concepts Ⓒ, instances △, conceptual properties △, as well as sub-type relations between concepts, property relations between instances and type relations between instances and concepts, indicated as arrows, respectively.

They focus on concepts and relation properties that represent cause-effect chains. One of the most important concepts is the **event** (Figure 8-3)**.** An event characterizes what happening may endanger the water quality of a bathing beach, e.g. a natural event such as heavy rainfall, an industrial event such as an overload of a waste water treatment, or a microbial water contamination event. Events may cause other events, e.g. heavy rainfall may cause a run-off that leads to an overload of the waste water treatment which, in turn, may cause microbial contamination of the coastal water. Finally, events influence **incidents** of good, satisfactory or poor **water quality.**

Incidents are determined by the authorities through chemical and biological analysis of water probes and the result data of model forecasts[33]. They are finally the decision basis for the authorities to keep a beach open or to close it (Figure 8-4). These decisions have an environmental impact, an economic impact (e.g. decrease of tourism revenues) or a human **impact** (e.g. risk for the public health) which may be in conflict. Furthermore, these decisions have associated social and economic **costs**.

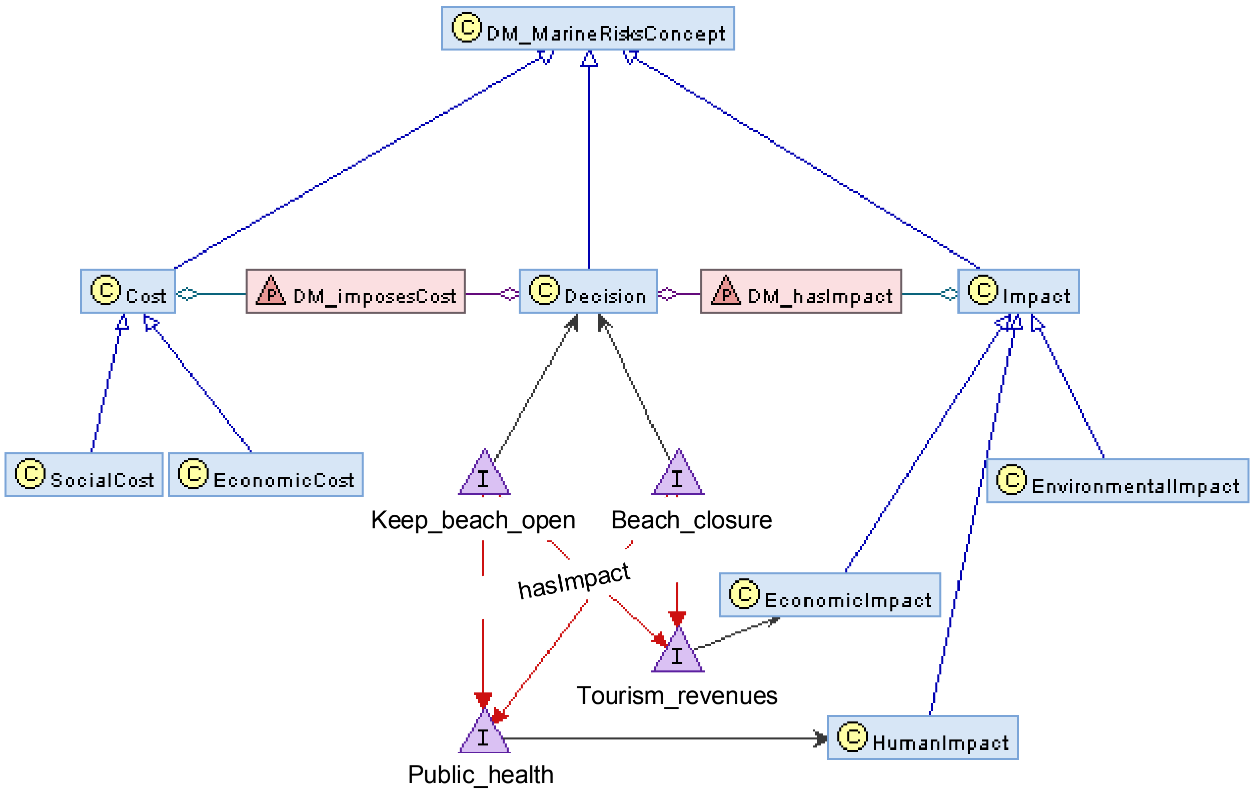


Figure 8-4: Ontology for the Application Theme "Marine Risks – Bathing Water Quality" focussing on Decisions, Impacts and Costs

[33] These concepts and relations are not shown in Figure 8-3 for the sake of readability of the figure.

8.3 Reference Model

The reference model introduced in section 5 is used as a foundation for this application example. This means, in particular, that the interface and service types of the service platform that are listed Table 5-2 comprise the capability types to which the requirements are mapped. For this application example the reference model was encoded as an OWL ontology.

Figure 8-5 shows the conceptual hierarchy of the major concepts of this ontology. It results from the SERVUS conceptual model (Figure 5-4) that integrates features and their properties, services and their interfaces as well as resources and their representations as presented in section 5.3. In addition, a service classifier (*RM_ServiceClassifier*) is introduced to enable the categorization of the service types according to the INSPIRE Network Services (EC, 2007), ISO 19119:2005 or the functional domains (Figure 5-2) of the Sensor Service Architecture (Usländer (ed.), 2009b).

The ontology was refined and populated with the objective to include all those interfaces and services that may be relevant for the realisation of the application example. Furthermore, as shown in Figure 5-2, it foresees sub-concepts *to RM_ResourceType* that represent the resource models of the services involved. They are refined in section 8.4.3 when the network of offered resources will be presented.

Figure 8-6 shows an excerpt of the resulting ontology with a focus on the capabilities of the Catalogue Service (*RM_CatalogueService*) and the Sensor Observation Service (*RM_SensorObservationService*). Their interfaces are structured as follows:

- Both services share the *RM_ServiceCapabilities* interface that enables to get descriptions (meta-information) about the capabilities of a running service instance. This interface may be used for the publishing activity for capability instances in order to create and update the entries in the catalogue (section 6.3.1.2).
- The OGC Sensor Observation Service (Na and Priest, 2007) comprises three interfaces: *RM_CoreOperationProfile* providing the operations to retrieve observations and sensor descriptions, *RM_EnhancedOperationProfile* providing operations related to the feature of interest as well as convenience operations, and *RM_TransactionOperationProfile* providing operations to insert new observation values into the underlying observation data store and to register sensors.
- The Catalogue Service comprises interfaces to search for catalogue entries (*RM_CatalogueSearchInterface*), if required enhanced by ontology-based query formulation (*RM_SemanticInterface*), to publish capability descriptions (*RM_Catalogue PublicationInterface*) and to retrieve collections of entries (*RM_CatalogueCollection Interface*). Furthermore, there is a management interface (*RM_Catalogue_Management Interface*) that enables to configure cascades of catalogues (see, e.g., the SemCat catalogue architecture as illustrated in Figure 7-2).

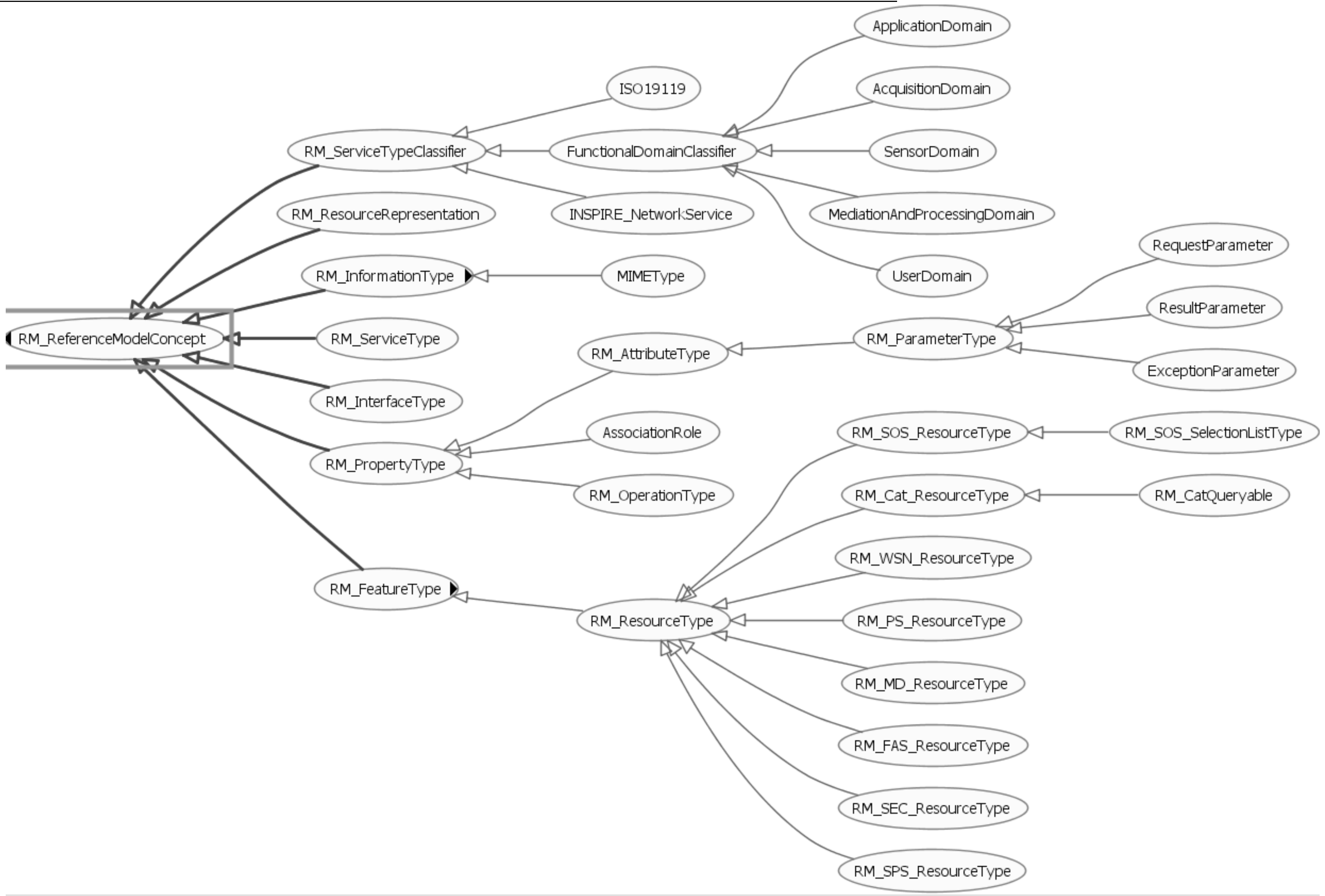


Figure 8-5: Conceptual Hierarchy of the Reference Model Ontology

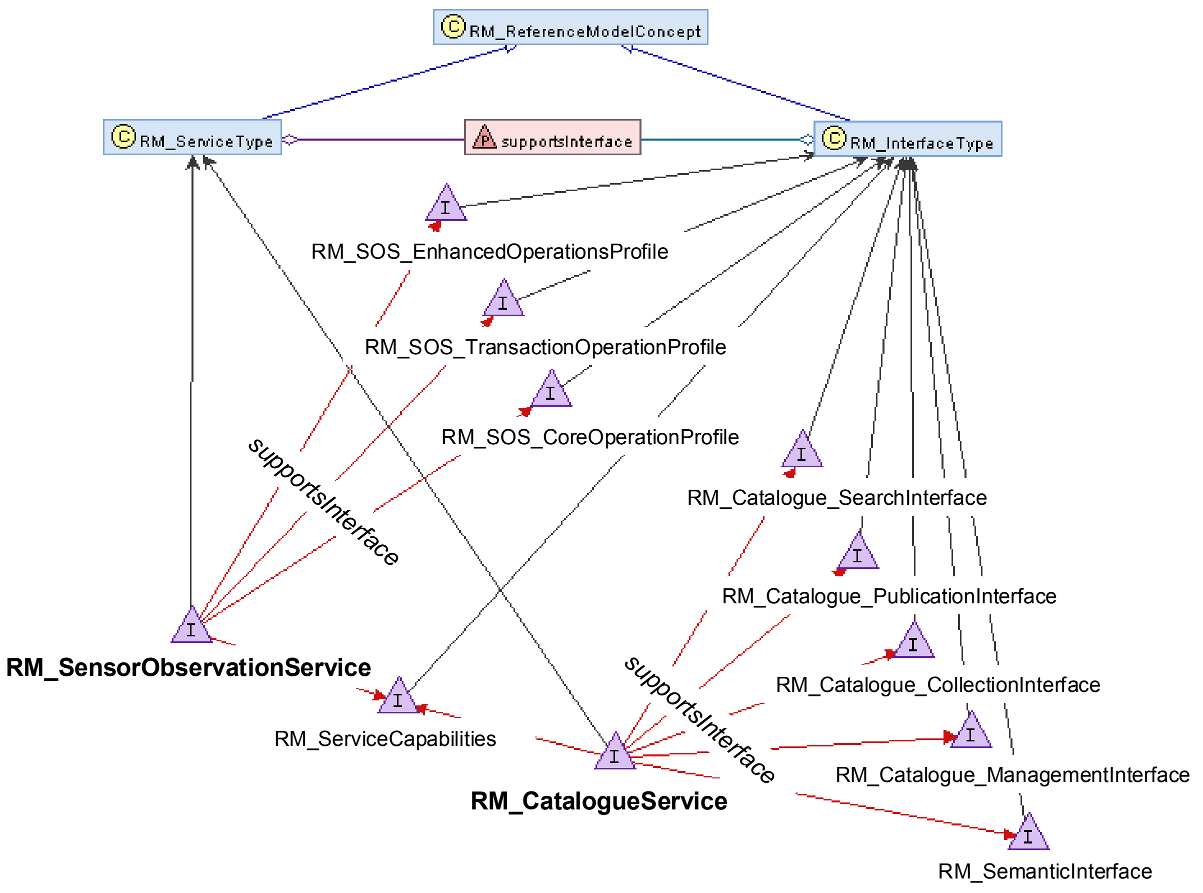

RM_ReferenceModelConcept
RM_ServiceType
supportsInterface
RM_InterfaceType
RM_SOS_EnhancedOperationsProfile
RM_SOS_TransactionOperationProfile
RM_SOS_CoreOperationProfile
supportsInterface
RM_SensorObservationService
RM_ServiceCapabilities
RM_Catalogue_SearchInterface
RM_Catalogue_PublicationInterface
RM_Catalogue_CollectionInterface
supportsInterface
RM_Catalogue_ManagementInterface
RM_CatalogueService
RM_SemanticInterface

Figure 8-6: Reference Model with a Focus on Catalogue and Sensor Observation Service

Figure 8-7 shows examples of resource representations that are relevant for the rendering of information through resources that are offered by the OGC Sensor Observation Service. It includes representations such as *document* (e.g. to get a collection of observations in form of a coverage file), *map* (e.g., to visualize features of interests in a cartographical context), *diagram* (e.g., to render observation collections as a time series diagram), *legend* (of a map or a diagram), *or styled layer descriptor* (i.e., a formal representation of the structure and the contents of a map layer). These resource representations are linked to standard Multipurpose Internet Mail Extensions (MIME) formats[34].

[34] See the list of MIME Media Types assigned by the Internet Assigned Numbers Authority (IANA) at http://www.iana.org/assignments/media-types .

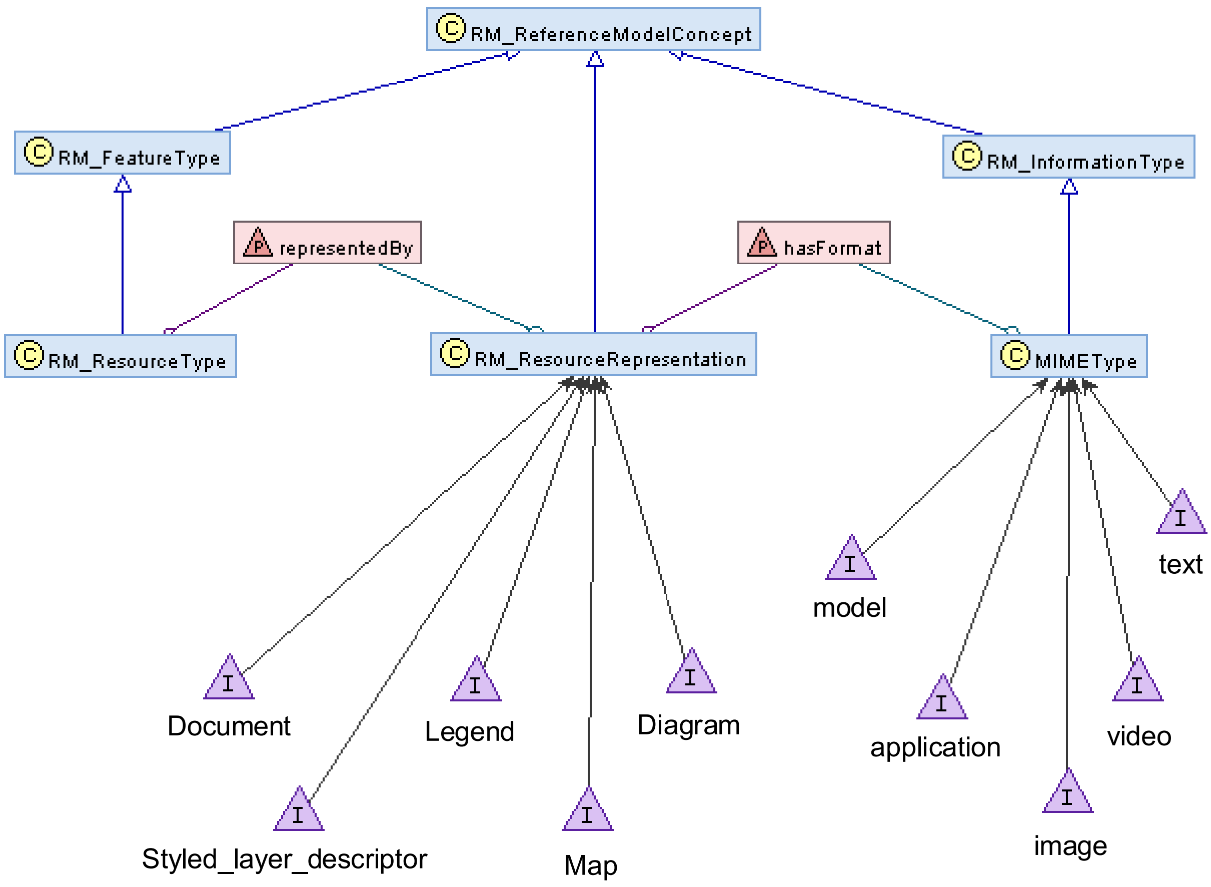


Figure 8-7: Reference Model with a Focus on Resource Representations dedicated to the OGC Sensor Observation Service

8.4 Design Model

8.4.1 Rephrasing of the Use Case Actions

Of the selected application process on the forecasting of water contamination risks, one use case is investigated in detail to illustrate the SERVUS rephrasing activity. As described in section 6.3.2 the purpose of rephrasing is the systematic derivation of the requested resources from the textual description of the use case. According to the activity diagram (Figure 6-4) it comprises an iteration loop consisting of the sub-activities *RephraseAsResource*, *AnnotateByConcept* (which is an optional sub-activity that may include a query to the domain model and results in a *model reference*) and the *update* of the resource network. This iteration loop is carried out for each use case action.

<u>Description of the use case "Select & Run Available Fusion Processes" (Table 7-1):</u>

> *A beach safety manager who is using system will need to be able to select data sources which he would like to subject to a fusion process. He will need to be able to select the fusion process and will require notification that fused data has been added to the catalogue.*

Note: **Fusion** processes enable the predictions of environmental parameters and their respective uncertainties in time and space when or where sensing measurements are not available. In EIS, fusion processes are used to integrate observation data, contextual data and phenomenological models from different sources in order to obtain new environmental information. Three types of fusion processes may be distinguished (Klopfer and Simonis (eds.), 2009):

- Spatial fusion services using e.g., Kriging or Bayesian Maximum Entropy.
- Causal fusion services using e.g., multi-linear regressions or neural networks.
- Temporal fusion services using e.g., state-space modelling and Kalman filters.

In the following, each use case action and their rephrasing approach is presented individually.

8.4.1.1 Action 1: Selection of Data for Fusion Processing

This action is rephrased into a requested resource *Sensor_data* that supports a *getResource* operation in the representation format that is suitable as input data format for the software environment in which the fusion processing is running (Figure 8-8).

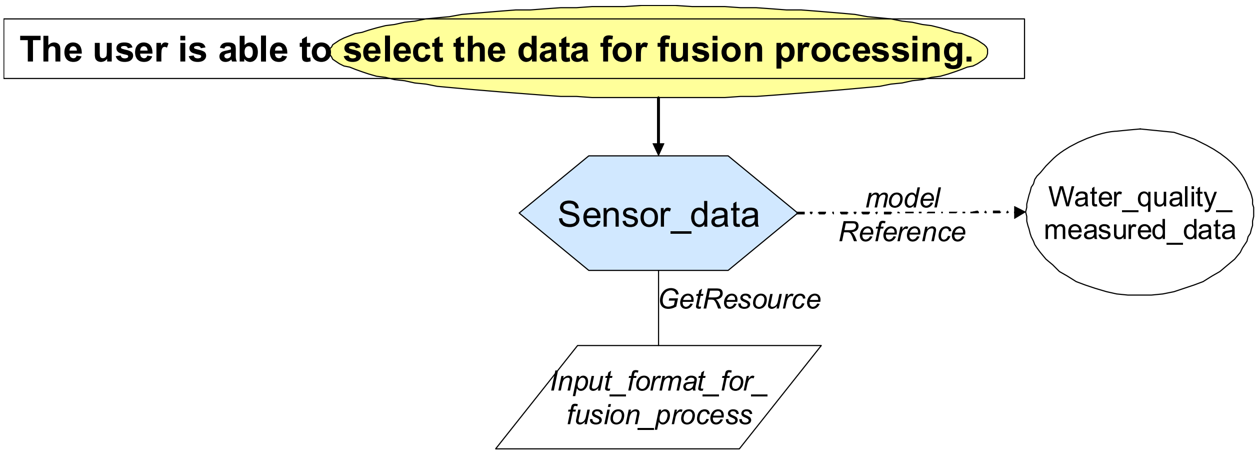

The user is able to select the data for fusion processing.
Sensor_data
model Reference
Water_quality_ measured_data
GetResource
Input_format_for_ fusion_process

Figure 8-8: Rephrasing of Use Case Action "Selection of Data for Fusion Processing"

According to the textual description of the application process in which this use case is defined the semantics of the resource *Sensor_data* is semantically annotated by a model reference to the domain model concept *Water_quality_measured_data*.

8.4.1.2 Action 2: Selection of the Fusion Process

This action is rephrased into a requested resource *Fusion_process* that supports a *getResourceMetadata* operation to retrieve the characteristics of the available fusion processes in a *list* representation format (Figure 8-9). It contains descriptive information for each process, such as the type and parameters of the fusion algorithms, their availability and response times. Such meta-data is typically stored in service inventories (Erl, 2008a). Following the vocabulary of the geospatial community (Percivall (ed.), 2008) these inventories were called catalogues by the editors of the use case. Thus, the resource *Fusion_process* is linked to a further requested resource, the *Service_catalogue*. The *getResource* operation supports the selection of the fusion process. The requested representation format is a *Process_form* that enables to set the parameters of the fusion process, e.g. a reference to the sensor data including the uncertainty of the observations, and the sampling points at which the fusion process is to estimate a value of its property (Klopfer and Simonis (eds.), 2009). The *Fusion_process* resource refers to the model that enables a forecast of the water quality on the basis of the observed input data. Thus, it is semantically annotated by a model reference to the domain concept *Water_quality_model.*

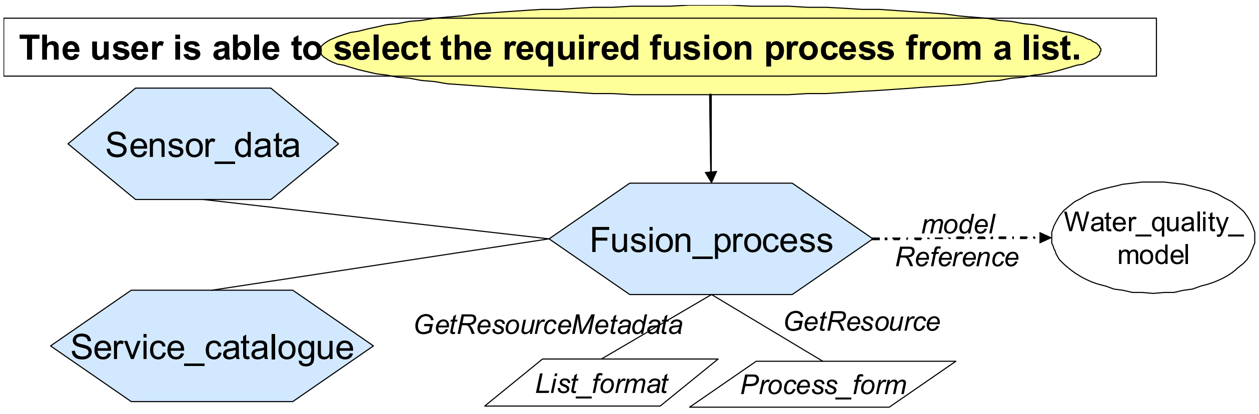


Figure 8-9: Rephrasing of Use Case Action "Selection of the Fusion Process"

8.4.1.3 Action 3: Initiation of the Fusion Process

This action builds upon the two requested resources *Fusion_process* and *Sensor_data* identified in the actions 1 and 2. The initiation of the selected fusion process is rephrased into a requested resource *Fusion_process_instance* that is linked to the other two (Figure 8-10).

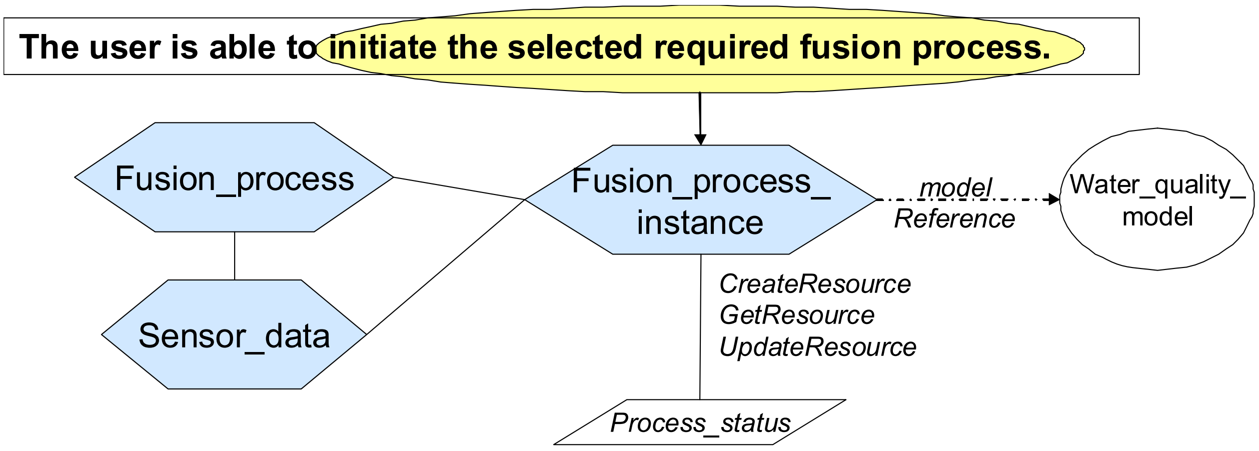


Figure 8-10: Rephrasing of Use Case Action "Initiation of the Fusion Process"

It represents the status of the running fusion process. This process needs to be started (*CreateResource* operation), its operational status needs to be surveyed (*GetResource* operation) and possibly re-configured (*UpdateResource*). In the domain model there is no distinction between the concept of the water quality model and its operational instance. Thus the model reference is identical to that of the *Fusion_process*.

8.4.1.4 Action 4: Data Processing and Storage of the Fusion Result

The access to the result of the fusion processing is rephrased by a *GetResource* operation to the resource *Fusion_process_result_data* (Figure 8-11). The result is delivered in the output format of the fusion process, e.g. a coverage file that contains the set of the estimated property values for the sampling points together with a quantified description of their uncertainty (Klopfer and Simonis (eds.), 2009). The result data refers to the *forecast data of the water quality concept* of the domain model. It is requested by this action to add the result to the data catalogue. This requirement is rephrased into an *UpdateResource* operation upon the requested resource *Data_catalogue*.

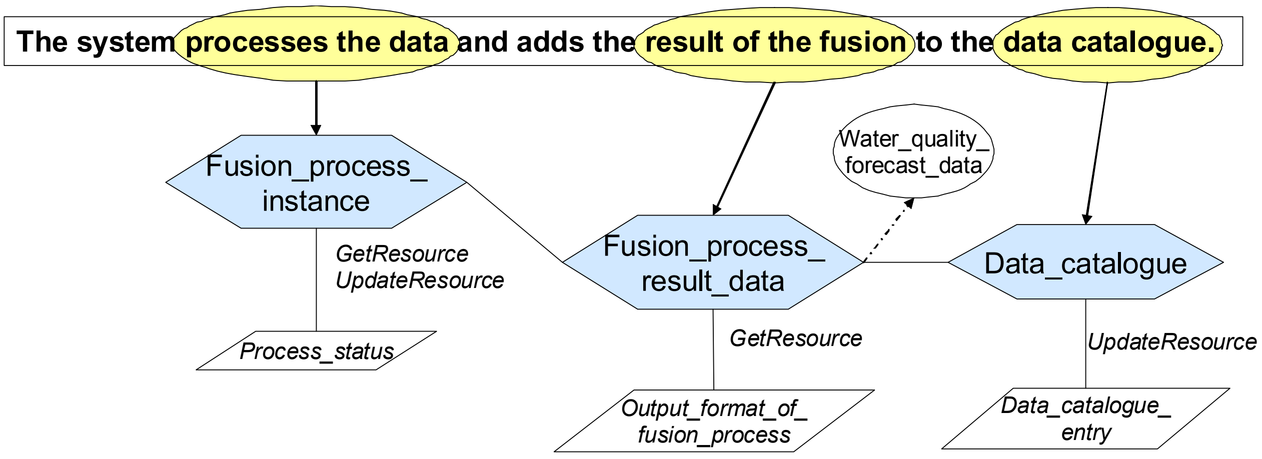

The system processes the data and adds the result of the fusion to the data catalogue.
Fusion_process_instance
GetResource
UpdateResource
Process_status
Water_quality_forecast_data
Fusion_process_result_data
GetResource
Output_format_of_fusion_process
Data_catalogue
UpdateResource
Data_catalogue_entry

Figure 8-11: Rephrasing of Use Case Action "Data Processing and Storage of the Fusion Result"

8.4.1.5 Action 5: Notification of the User

Usually, the fusion process runs asynchronously such that the user activity is not suspended during the fusion process run. However, this requires a notification to the user when the fusion process has terminated. This requirement is rephrased into a resource called *User_notification_fusion_result* whose creation in the representation format *Notification_flag* signals the availability of the fusion result and the notification of the user (Figure 8-12).

Note that the communication means (e.g., by e-mail or short message service to a mobile device) is not specified in the action. This resource is linked to another requested resource called *User_object* that represents the authenticated user of the marine application. Its characteristics are specified by a further use case called "Launch bathing water application" that is dedicated to user registration and authentication.

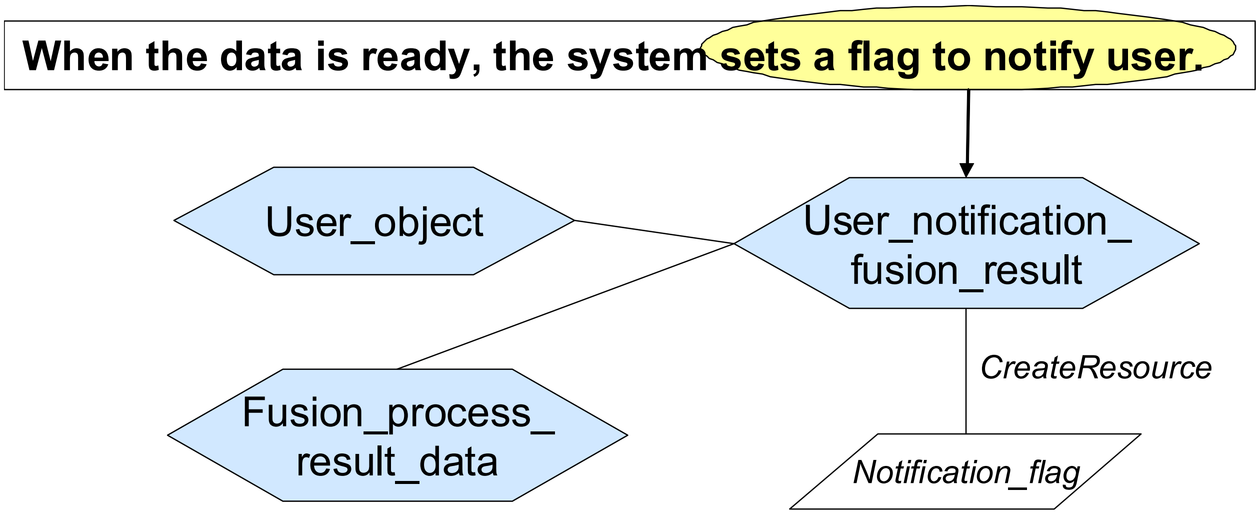

When the data is ready, the system sets a flag to notify user.
User_object
User_notification_ fusion_result
Fusion_process_ result_data
CreateResource
Notification_flag

Figure 8-12: Rephrasing of Use Case Action "Notification of the User"

8.4.2 Requirements Model - Network of Requested Resources

The selected application process contains 19 use cases structured into 53 actions. Their complete analysis and rephrasing leads to a network of 44 requested resources structured into seven resource sub-types (Figure 8-13) and 19 resource representations (Figure 8-14).

Note: For the sake of readability only those elements of these two ontology figures are highlighted that are relevant for the understanding of this application example. An illustration and discussion of the complete resource network exceeds the scope of this thesis.

The purpose of these two figures is two-fold: In the first place, it illustrates the degree of complexity resulting from the analysis of just this application process. In the second place, it shows the need to discuss the use cases and the individual actions with the users in order to clarify terminological and semantic issues. Such an iteration step helps in getting a more concrete understanding of the user requirements and, as a side effect, to detect commonalities and redundancies and thus to reduce the complexity.

The first example refers to the use case descriptions that talk about a "data catalogue" (*Data_catalogue* in Figure 8-11). On the one hand, a data catalogue shall store both the measured data (*Sensor_data*) and the output data (forecast results, *Fusion_process_result_data*) of the fusion process but, on the other hand, it is also referred to when searching for historical data sets. It is the task of the system architect to decide in which case this requested resource is mapped to offered resources that provide data storage and retrieval functions (e.g. a database that is wrapped by a Feature Access Service (Usländer (ed.), 2007)) and/or to the resources of a geospatial catalogue service that is dedicated to the access and management of meta-data, i.e. descriptions about data sets.

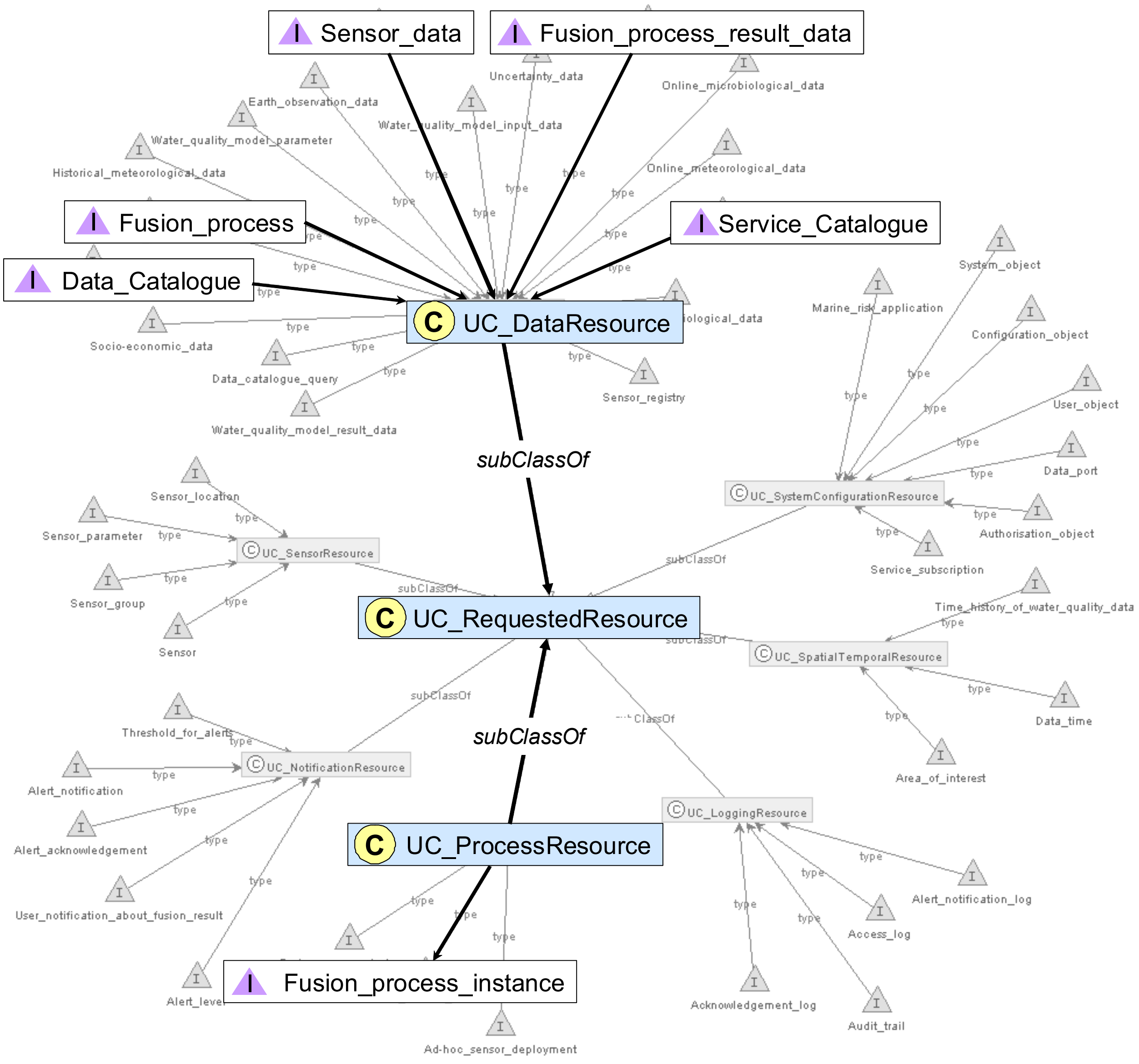


Figure 8-13: Types and Instances of Requested Resources

The second example refers to the use case description that requests "to select the required fusion process from a list" (section 8.4.1.2). This requirement is rephrased into the requested resources *Fusion_process*, *Service_catalogue* and *Fusion_process_instance*.

The third example refers to the representations of the requested resources (Figure 8-14). In the use case descriptions the format requirements range from concrete (e.g. display as a histogram), fuzzy (e.g. chart layer) up to open (e.g., display of socio-economic data). Furthermore, there are use case actions that request application forms with user interactions. These are translated into corresponding requested resources such as *Configuration_editor* and *Login_screen* without refining at this point in time how they should concretely look like. The structured results of the rephrasing activity in terms of resources help in the clarification and concretization of the requirements and the disambiguation of terms when discussing them with the users.

This iteration step should happen before the mapping to the offered resources is started.

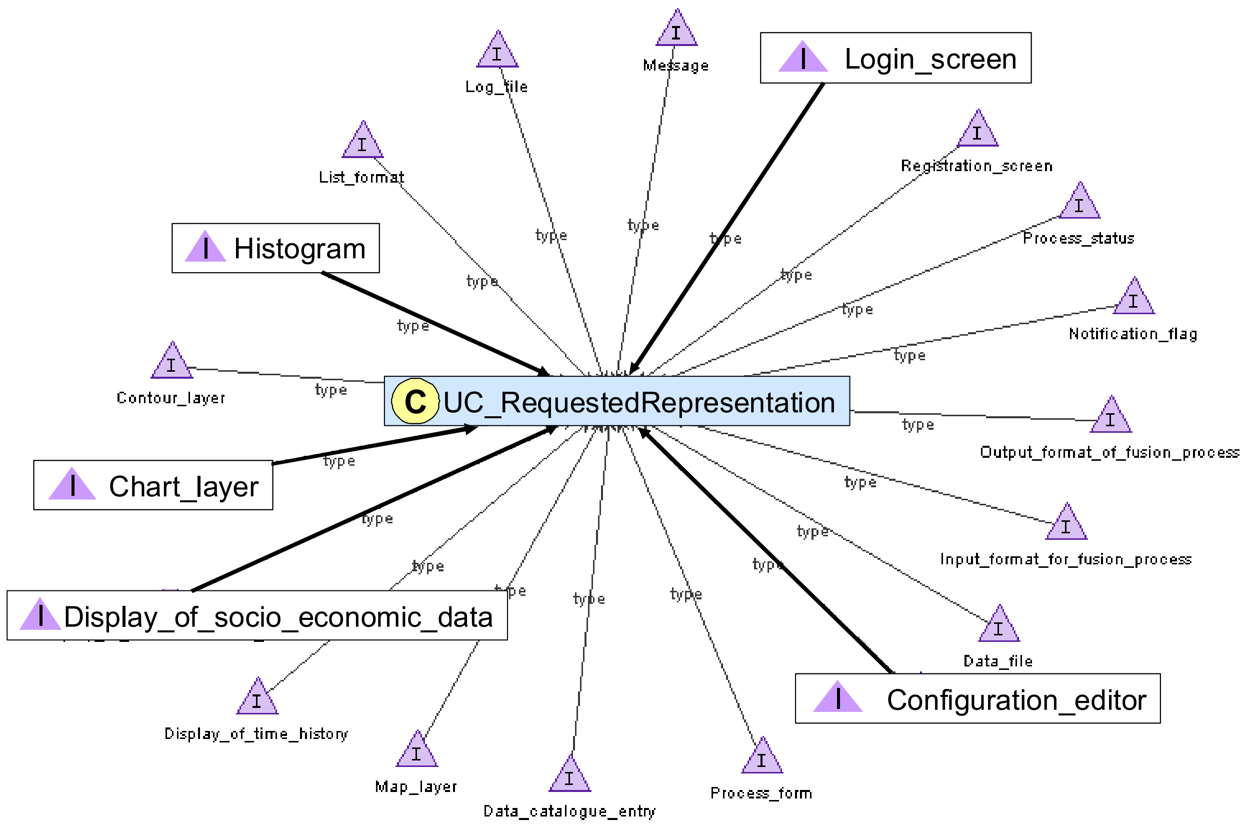


Figure 8-14: Representations of Requested Resources

8.4.3 Capability Model - Network of Offered Resources

The SERVUS publishing activity foresees the edition of a capability model on both type and instance level based upon and referring to the concepts of the reference model (section 6.3.1). The capability type model is presented in form of a network of offered resource types in Figure 8-15. They result from the definition of a resource-oriented view upon those capabilities of the major interface and service types of the reference model that may be of interest for the realisation of the bathing water risk management application.

Resource models are specified for eight service types and consequently classified into eight resource sub-types. Those which are relevant for the chosen application example are described in more detail and highlighted in the figure:

- *RM_SOS_Resource*: Resources of this type represent the major elements of the OGC Observation Schema (Cox (ed.), 2007) as they are accessible and manageable through the operations of the OGC Sensor Observation Service. An example is the observation collection resource (SOS_Observation_collection) that refers to the getObservation operation and enables the retrieval of result data of a procedure which may be a sensor device or a phenomenological model, e.g. a water quality forecast model. Furthermore, there is the offered resource *SOS_Offering* that specifies what an SOS offers to its consumers in terms of the procedures that are available as well as the spatial and temporal coverage of the observations per observed properties (e.g., "nitrate concentration values between 2003 and 2008 in the Upper Rhine Valley measured by the groundwater monitoring stations GW1, GW42 and GW43").

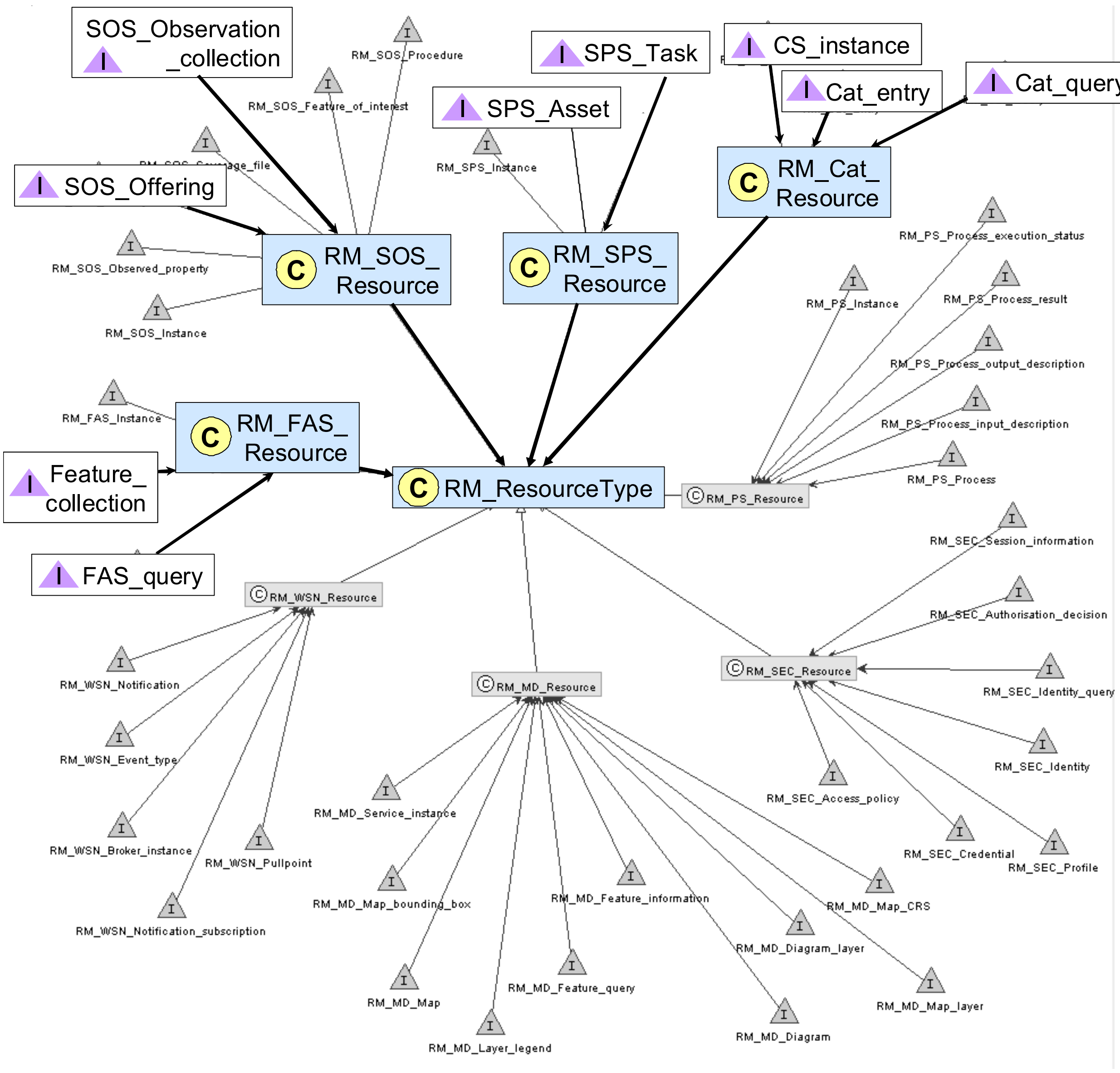


Figure 8-15: Types and Instances of Offered Resources[35]

[35] Meaning of the acronyms: RM = Reference Model, Cat = catalogue, FAS = Feature Access Service, MD = Map and Diagram, PS = Processing Service, SEC = Security, SOS = Sensor Observation Service, SPS = Sensor Planning Service, WSN = Web Services Notification.

- *RM_SPS_Resource*: Resources of this type represent the major elements of the OGC Sensor Planning Service (Simonis (ed.), 2007) whose operations enable to task any kind of sensors (here also called *Asset*) to retrieve observation collections. The major offered resource is the *Task* which enables the configuration of the observation collection behaviour of an asset in terms of time (e.g., when the observation should happen), space (e.g., which location is of interest) and contents (e.g., what environmental parameter should be observed).
- *RM_Cat_Resource*: Resources of this type represent the major elements of a geospatial catalogue. The major resource types distinguish between the meta-data entries of a catalogue (*Cat_Entry*) and the catalogue service instance itself (*CS_instance*). The catalogue query resource type (*Cat_Query*) enables to formulate queries.
- *RM_FAS_Resource*: Resources of this type represent the major elements of the Feature Access Service (Usländer (ed.), 2007) that enables read and write access to geospatial features (objects), usually stored in relational databases. Important resource types are the query (*FAS_Query*) that enables to formulate queries in an SQL-like fashion, and the feature collection (*Feature_collection*) in order to retrieve (*GetResource* operation) and change (*UpdateResource* operation) feature instances. The selection of the features is realized by linking them to the query resources.

8.4.4 Network of Requested and Offered Resources

The SERVUS design activity that follows the requirements analysis (rephrasing activity) and the provision of the capability model (publishing activity) is the discovery and matching activity. Here, the network of requested resources is to be mapped to the network of offered resources. For the given application example this activity was carried out manually. Figure 8-16 shows the resulting resource network for all the resources that result from the rephrasing of the use case described in section 8.4.1:

- The requested resources *Data_catalogue* and *Service_catalogue* are both mapped to the offered resource *CS_instance* as the geospatial catalogue service of the reference model can store meta-data of both data and services.
- Consequently, the requested resource *Fusion_process* which is linked to the *Service_catalogue* is mapped to an entry resource of the geospatial catalogue (*Cat_Entry*).
- Both the *Sensor_data* and the *Fusion_process_result_data* requested resources are mapped to the offered resource *Observation_collection* as this resource does encompass result data of any kind of procedure, being a sensor device or a model.
- However, there is an alternative mapping. *Fusion_process_result_data* may also be considered as a feature collection. In this case, it may be mapped to the offered

resource *Feature_collection*. This is a design decision that has to be taken by the system architect taking into account, for instance, further non-functional aspects such as performance or security.

- The Fusion_process_instance is mapped to the offered resource *Asset* that is related to the Sensor Planning Service (SPS) and represents any "means of collecting information" (Simonis (ed.), 2007).

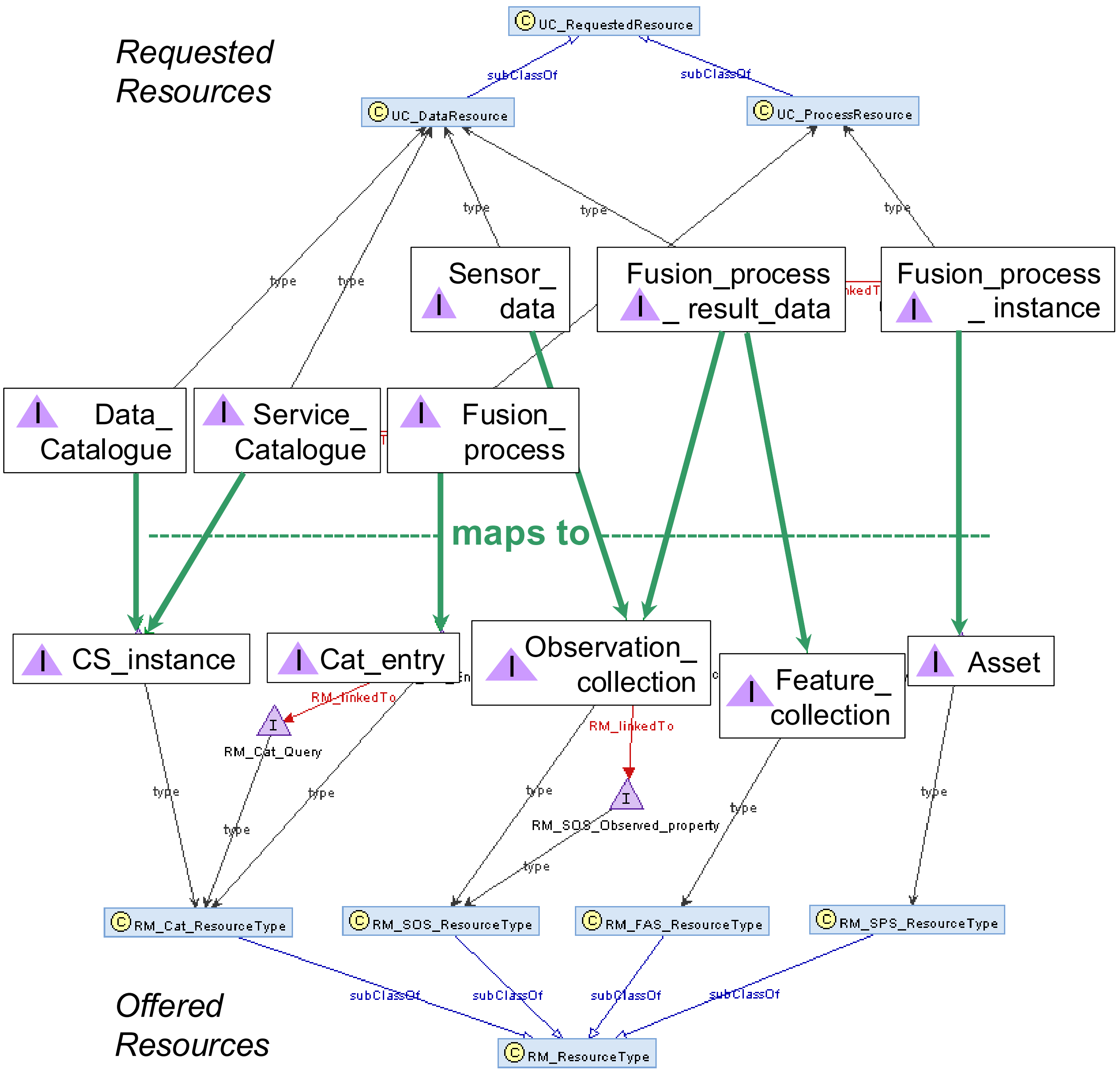


Figure 8-16: Resource Network with Requested and Offered Resources

8.4.5 Semantic Resource Network

The SERVUS Resource Model supports semantic annotation which allows the meaning of a resource not only to be given by its name and its textual description but also by a formal reference to a concept in a semantic model, for instance, an ontology. This semantic model reference is exploited when thinking of how the SERVUS discovery and matching activity may be supported by a software tool.

Two variants of resource coupling are distinguished: direct and indirect coupling (Figure 8-17). Two resources are said to be **directly coupled** through a concept if they both own a model reference either to the same concept or to two concepts that are themselves linked by a model reference (left side of Figure 8-17). They are said to be **indirectly coupled** if they own model references to two distinct concepts which are, however, related by an is-a property (subsumption relationship, right side of Figure 8-17). Both variants may also be combined and applied in multiple steps.

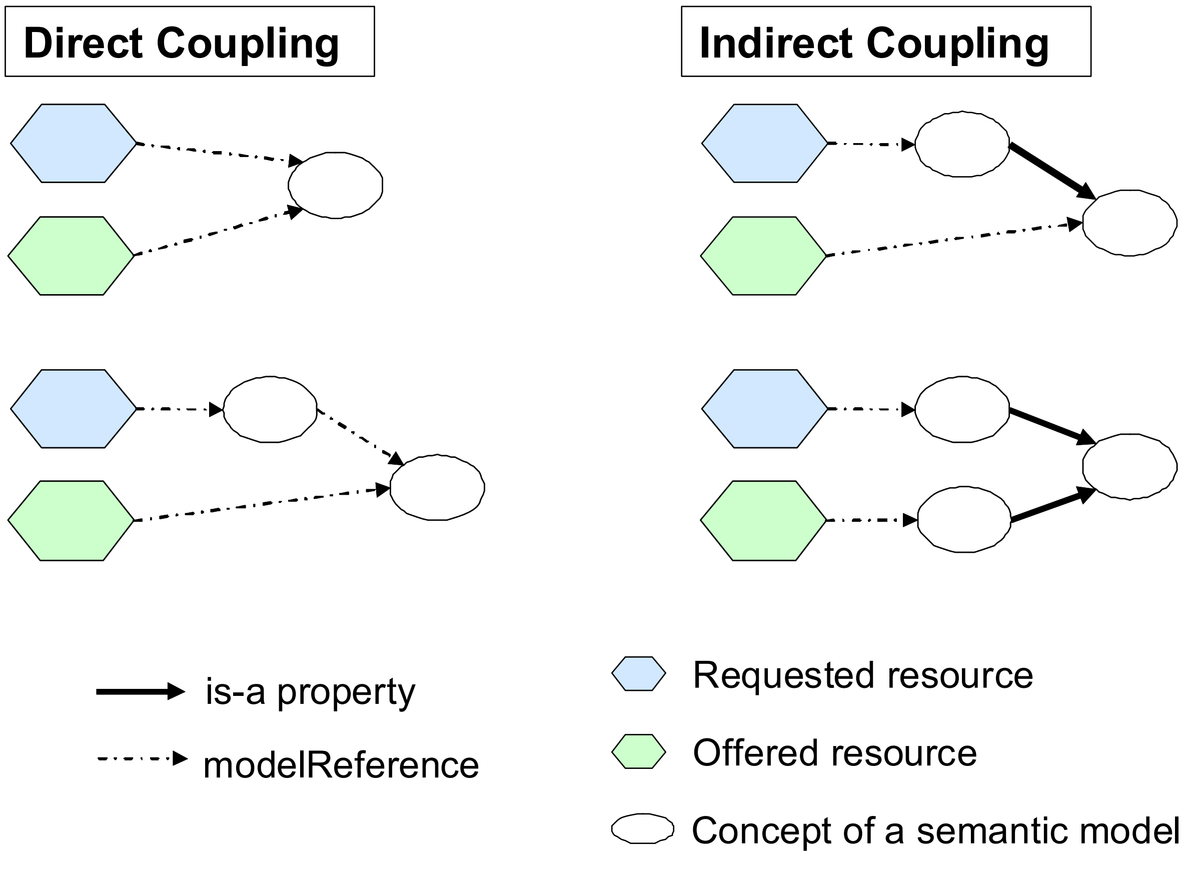


Figure 8-17: Direct and Indirect Resource Coupling

Based upon these definitions, the following rule may be specified for the matching:

> *An offered resource is a candidate resource for a matching with a requested resource, if the offered resource is directly or indirectly coupled through an identified concept in a semantic model.*

Figure 8-18 shows how this rule may be applied to the requested resource *Sensor_data* of the chosen application example.

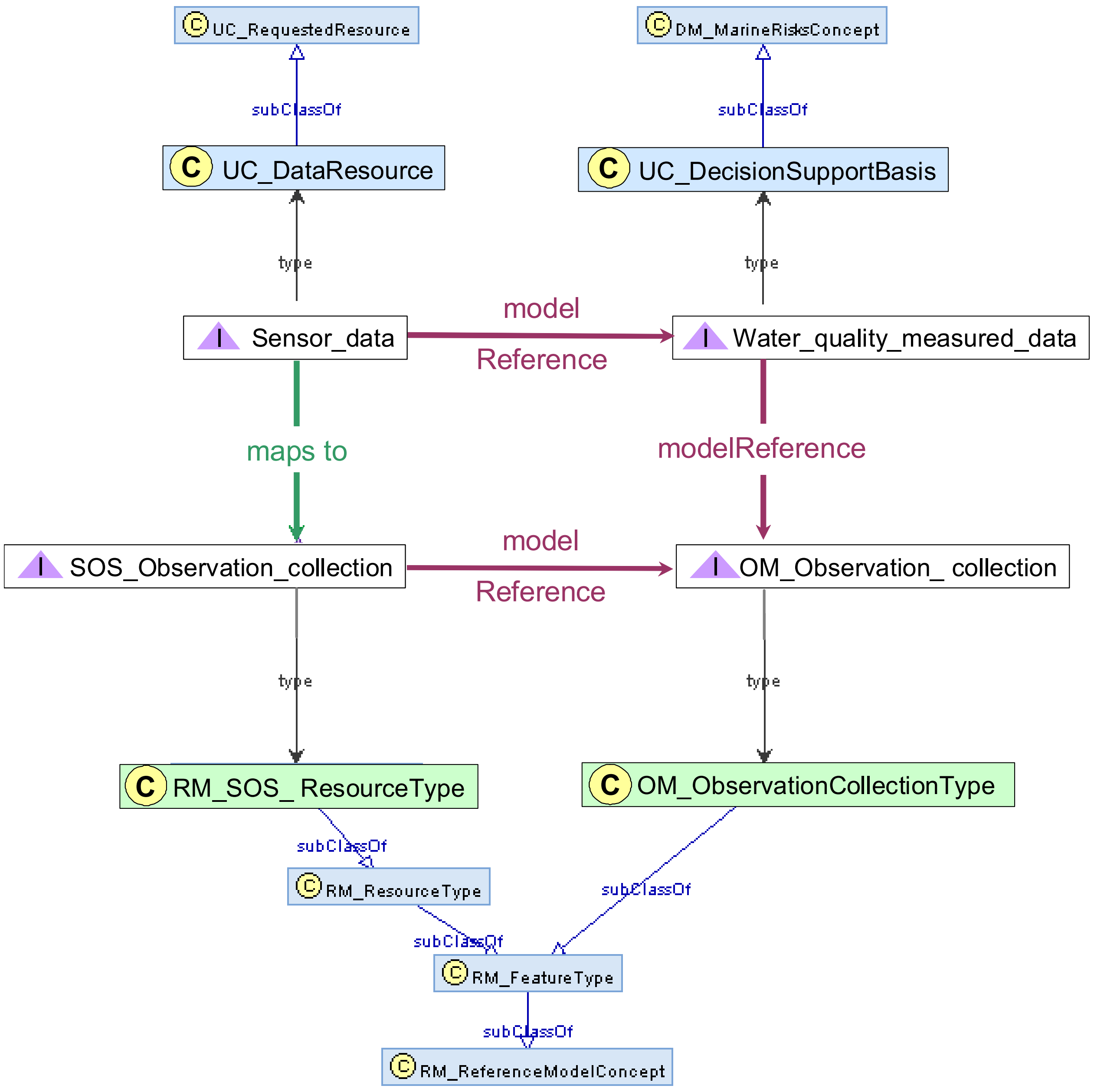


Figure 8-18: Mapping of Requested Resource "Sensor Data" including Model References

As a result of the rephrasing activity *Sensor_data* was defined with a model reference to *Water_quality_measured_data* which is a concept of the domain model (section 8.4.1.1). Furthermore, *Water_quality_measured_data* has a model reference to *OM_Observation_collection* which is a concept of the reference model (section 8.3). This second model reference bridges the semantic gap between the vocabulary of the marine risks application and the terms used in the geospatial standardisation community, here the OGC Observations and Measurements (O&M) model (Cox (ed.), 2007).

It expresses the fact that the data set that results from a "measurement of the water quality" is a specialization of a generic "observation collection". An observation collection is an estimate of a set of values about an environmental phenomenon provided by any kind of procedure, e.g. a sensor or, in this case, a chemical and biological analysis of a water probe. As a consequence, the requested resource *Sensor_data* is indirectly coupled to the offered resource *OM_Observation_collection*. This means that *OM_Observation_collection* is a candidate resource for the requested resource *Sensor_data*. Furthermore, the capability lifting schema of the OGC Sensor Observation Service (SOS) delivers that *OM_Observation_collection* is the published resource type of the SOS *getObservation* operation. As a consequence, it may be inferred that the use case action "Selection of Data for Fusion Processing" may be implemented by means of this SOS operation.

9 Evaluation and Outlook

The framework of information systems research that was adopted for this thesis (section 2.3) proposes seven guidelines (Table 9-1) for effective design-science research (Hevner et al, 2004). These guidelines are applied as structuring means for a general evaluation of the thesis results (section 9.1). A more specific evaluation based upon a comprehensive list of SOAD methodology criteria follows in section 9.2. Finally, section 9.3.2 elaborates on future research topics and concludes with a revisit of the motivation section 1.1 in the introduction of this thesis.

No.	Guideline	Guideline Description
1	Design as an Artefact	Design-science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation.
2	Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
3	Design Evaluation	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.
4	Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies
5	Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.
6	Design as a Search Process	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7	Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 9-1: Guidelines for Design-Science Research (Hevner at al, 2004)

9.1 Evaluation by Design-Science Research Guidelines

9.1.1 Design as an Artefact

The research guideline “design as an artefact” requires to identify the design artefacts (DA) of this thesis and to classify them in terms of constructs, models, methods, or instantiations according to March and Smith (1995):

- **Constructs** provide the language in which problems and solutions are defined and communicated.
- **Models** use constructs to represent a real world situation, i.e. the design problem and its solution space (Simon, 1996).

- **Methods** define processes and provide guidance on how to search the solution space. They range from formal, mathematical algorithms that explicitly define the search process to informal, textual descriptions of "best practices" approaches, or some combination thereof.
- **Instantiations** are implementations of constructs, models and methods in working systems. They demonstrate feasibility, enabling concrete assessment of an artefact's suitability to its intended purpose.

Table 9-2 lists the five design artefacts that result from this thesis. They contribute to the "knowledge base" of the design-science research (Figure 2-9) covering the design of EIS applications based upon the principles of geospatial S. The column "Design Requirement" refers to the requirements for service-oriented design that were set-up in section 1.4.3. An entry R.x resp. (R.x) means that the design artefact fulfils (resp. partially) fulfils the design requirement R.x.

Artefact Category	No.	Design Artefact Name	Justification	Design Requirement (section 1.4.3)	Reference
Construct	DA.1	Reference Model (UML meta-model)	Language to formally express the functions and information models of open geospatial service platforms	R.4, (R.5), R.6, (R.7)	Section 5
	DA.2	SERVUS Resource Model	Common language in which both requirements and capabilities may be expressed	R.2, R.3, R.4, (R.5), R.6, R.7, R.8, R.11, R.12	Section 5.3.2
Model	DA.3	RM-OA	Provides a reference model for risk management applications based upon DA.1 and DA.2	R.4, (R.5), R.6, R.7, R.11, R.12	Section 5
Method	DA.4	SERVUS Design Methodology	Identifies design activities and orders them in a design process based upon a modelling framework	R.1, R.2, R.3, R.4, R.5, R.6, R.7, R.8, (R.9), R.11, R.12	Sections 4 and 6
Instantiation	DA.5	SERVUS Implementation Architecture	Describes an option to implement the SERVUS Design Methodology (DA.4) on the basis of a geospatial service platform	R.5, R.6, R.7, (R.10), (R.11), R.12	Section 7

Table 9-2: Design Artefacts contributed by this Thesis

9.1.2 Problem Relevance

The design-science research guideline "problem relevance" requests that the design artefacts listed in Table 9-2 enable solutions to **important and relevant business problems**.

The first business problem tackled by this thesis is to effectively **design architectures for large-scale EIS** that, on the one hand, fulfil the requirements expressed by users and stakeholders in a traceable manner, and, on the other hand, exploit and reuse as much as possible and adequate existing capabilities of service platforms based upon geospatial standards. The importance of this problem is demonstrated by the multitude of national and international initiatives and projects (section 2) aiming at designing systems-of-systems in the environmental and associated thematic domains. The relevance of this problem is made visible by the increasing adoption of service-oriented computing principles and uptake of corresponding geospatial standards by these initiatives and projects (section 2.2), the non-existence of adequate SOAD methodologies (section 3.4.5) and the continued identification of SOAD as future research topic (Kontogiannis et al, 2007). The design artefacts DA.2, DA.4 and DA.5 contribute to a solution for this business problem.

The second business problem encompasses the provision of an **agreed architectural framework** for the category of systems described above. The importance and relevance of this problem was stressed by the European Commission in its call of 2003 for research projects[36] with the aim

- to "develop open platforms, integrated systems and components for improved risk management, civil security applications (…) and environmental management", and
- to "foster the emergence of a European info-structure and service platforms in order to facilitate the use of interoperable components and sub-systems".

ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) (Klopfer and Kannellopoulos (eds.), 2008) was one of the strategic projects funded by the European Commission as a response to this call. The design artefacts DA.1 and DA.3 constitute core elements of the Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer (ed.), 2007) which is one of the key results of the ORCHESTRA project (section 9.1.7).

9.1.3 Design Evaluation

This guideline requires that the "utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods". Hevner et al (2004) describe a spectrum of design evaluation methods. Table 9-3 indicates which

[36] Framework Programme 6 - Information Systems Technologies (IST) strategic objective *2.3.2.9 – Improving Risk Management*

of these evaluation methods were applied to which design artefact with which level of rigor and provides a short reason:

- X = low level of rigor
- XX = medium level of rigor
- XXX = strong level of rigor

For this evaluation, DA.1 and DA.3 are combined to one design artefact "Reference Model", whereby the design artefacts DA.2, DA.4 and DA.5 are discussed together under the umbrella term "SERVUS". A more detailed justification to the major assessments of Table 9-3 is given below.

Evaluation Method	Description	Reference Model	SERVUS	Reason
1. Observational	Case Study: Study artefact in depth in business environment	X		use in research projects
	Field Study: Monitor use of artefact in multiple projects	X		re-use by other projects
2. Analytical	Static Analysis: Examine structure of artefact for static qualities (e.g., complexity)			
	Architecture Analysis: Study fit of artefact into technical IS architecture	XXX	XX	geospatial architectures
	Optimization: Demonstrate inherent optimal properties of artefact or provide optimality bounds on artefact behaviour			
	Dynamic Analysis: Study artefact in use for dynamic qualities (e.g., performance)			
3. Experimental	Controlled Experiment: Study artefact in controlled environment for qualities (e.g., usability)	XXX	XX	evaluation in research projects
	Simulation: Execute artefact with artificial data		X	study with SANY use case

4. Testing	Functional (Black Box) Testing: Execute artefact interfaces to discover failures and identify defects			
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artefact implementation			
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the utility of the artefact	XX	XXX	based upon open geospatial standards and related research results
	Scenarios: Construct detailed scenarios around the artefact to demonstrate its utility	X	X	ORCHESTRA and SANY pilots

Table 9-3: Application of the Design Evaluation Methods of Hevner et al (2004)

9.1.3.1 Evaluation Methods applied to the Artefact "Reference Model"

The primary evaluation method applied to the reference model is the **architecture analysis**. Not only has the reference model been systematically derived from the needs of environmental applications and geospatial SOAs in systems-of-systems, it was also applied and refined in four iteration steps in the course of the development of the technical EIS architectures for the ORCHESTRA application pilots (Klopfer and Kannellopoulos (eds.), 2008) and for the follow-on research project SANY (Klopfer and Simonis (eds.), 2009). The latter application resulted in the specification of the Sensor Service Architecture (Usländer (ed.), 2009b) dedicated to Sensor Web environments. Although driven by real-world needs, the application of the reference model in these research projects is a study of its usability in a **controlled experiment**.

However, after the submission of the RM-OA including its UML meta-models to the OGC Architecture Working Group and its final acceptance as OGC best-practice document in 2007, an increasing uptake in other projects and business environments is noticed. An example is that the design of an e-Science and technology infrastructures for biodiversity data and observatories in the LifeWatch project (Giddy et al, 2009) is based upon the RM-OA.

9.1.3.2 Evaluation methods applied to the Artefact "SERVUS"

The SERVUS Resource Model and the associated design methodology were primarily evaluated in a "descriptive" form by **informed argument**, i.e. the use of available information (here from the geospatial standardization and research community) to build a convincing argument for the utility of the artefact. Furthermore, the utility of SERVUS is assessed by applying existing evaluation criteria developed in comparison studies between SOAD methodologies (see section 9.2 below).

Architectural analysis was carried out by the conception of a SERVUS Implementation Architecture (section 7) in such a way that essential capabilities of existing architectures may be re-used and also applied to the design process itself (although originally not conceived for such a purpose). A controlled experiment with "artificial data" was performed by rephrasing the requirements of the SANY application pilot "marine risks" (Williams (ed.), 2008) to a network of requested semantic resources, and by matching them with a set of offered resources derived from the geospatial services of the Sensor Service Architecture (Usländer (ed.), 2009b).

9.1.4 Research Contributions

The guideline "research contribution" demands that "clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies" are delivered. Table 9-4 demonstrates compliance with this guideline by relating selected research topics for the engineering domain identified by Kontogiannis et al (2007) to the design artefacts described in section 9.1.1.

9.1.5 Research Rigor

Rigorous methods have to be applied for both the construction and evaluation of the design artefacts in order to follow the design-science guideline of "research rigor". The construction of the design artefacts described in section 9.1.1 relies upon the principles of model-driven architecture (MDA) (section 2.3.4) and international standards of ISO and the OGC. In particular, the information and process models of the artefacts DA.1, DA.2 and DA.4 were specified in the general-purpose modelling language UML and related to the meta-model for information, i.e. the ISO General Feature Model (ISO 19109:2005). The RM-OA (DA.3) was iteratively constructed and evaluated in a consensus-driven process that started within the consortium of the ORCHESTRA project and continued in the Architecture Working group of the OGC.

The construction of the SERVUS Implementation Architecture (DA.5) is derived from the UML-based specification of the design process model (section 6) and relies upon the established principles of service-oriented computing (section 2.2), ontology modelling (Stuckenschmidt, 2009) and Web-based applications. Its rigorous evaluation, e.g. in the form of a case or field study in a business environment, would require a full implementation in an integrated design environment which is one of the possible follow-on research topics described in section 9.3.2.

Research Topic	Relationship to the Design Artefacts
Process and Lifecycle support, e.g. by development processes and methodologies for service-oriented systems	SERVUS is a proposal for a design methodology for geospatial SOA (see section 4).
Requirements analysis support, e.g. management of non-functional, soft, evolvable and possibly conflicting requirements from the perspectives of service providers, service consumers and infrastructure	The SERVUS Modelling Framework provides a uniform modelling approach that encompasses the perspectives of users (by rephrasing uses cases into requested resources) and service providers (by rephrasing service capabilities into offered resources).
Service selection support, e.g. – Models to support strategic reuse – Techniques and models for the syntactic and most importantly, semantic description of services	The SERVUS Modelling Framework facilitates the discovery and matching (selection) of existing services by the publication of their capabilities in terms of offered resources. The resources provide the option for semantic annotations which provides the foundation for a matching with requirements on the semantic level.
Architecture and Design support, e.g. – Architectural styles for service-oriented systems – Architectures for data integration in service oriented environments – Architectures for service types, including Data services (Information as a service), Business services, and Infrastructure services – Design for run-time semantic-based discovery and composition	The Reference Model provides a harmonized meta-model for services, interfaces, features and resources. The SERVUS Design Methodology is based upon this reference model and may cope with multiple architectural styles in a distributed system, e.g. remote invocation or resource-oriented architectural styles. All types of services are supported. The RM-OA (DA.3) contains a series of service and interface specifications that comprise a high-level of generic functionality of a geospatial service platform. The publication of service capabilities as semantic resources enables run-time semantic-based discovery.
Implementation support, e.g. model-driven approaches and template-based code generation	The SERVUS Design Methodology (DA.4) follows a model-driven approach.
Tools and Products, e.g. integrated development environments to support service-oriented development	The SERVUS Implementation Architecture (DA.5) provides an option to implement the SERVUS Design Methodology based upon existing standard components of a geospatial service platform.

Table 9-4: Selected Research Topics of Kontogiannis et al (2007) and their relationships to the Design Artefacts

9.1.6 Design as a Search Process

"Design as a search process" denotes a guideline that is dedicated to the "search for an effective artefact". It requests that "available means to reach desired ends" are utilized "while satisfying laws in the problem environment". In the problem environment of EIS design these laws encompass the design constraints on both organizational and technological level and their interdependencies (section 1.2). It is one of the key assumptions of the service-oriented design of EIS that the problem environment mandates the use of international standards of the geospatial community. Thus, the resulting reference model and the design methodology heavily rely upon these "laws" of the problem environment. The "search" for effective artefacts is an inherent principle of the SERVUS Design Methodology. When applying the SERVUS design activities, it corresponds to the search for offered resources that may fulfil the requirements expressed by the characteristics of requested resources.

9.1.7 Communication of Research

The guideline "communication of research" recommends that the design research "must be presented effectively both to technology-oriented as well as management-oriented audiences". The design artefacts related to the reference model (DA.1 and DA.3) were extensively presented to decision makers, system architects and technicians of the geospatial community and the users and stakeholders of EIS (Usländer, 2008a; Usländer, 2009a). Furthermore, it was communicated to related application domains such as the health domain (Skouloudis (ed.), 2009) and the risk, emergency and security management domain (Douglas et al, 2008; Usländer, 2009c; Usländer and Denzer, 2009). These activities culminated in the acceptance of the RM-OA (DA.3) as OGC best-practices architecture specification (Usländer (ed.), 2007) and its uptake by other research communities, e.g. biodiversity research (Giddy et al, 2009).

The SERVUS Resource Model (DA.2) was submitted to OGC (Usländer, 2008b) and presented in the OGC Architecture Working Group as a contribution to the ongoing discussion about the future strategy of OGC with respect to service models and architectural styles (Pautasso et al, 2008; Lucchi, Millot and Elfers, 2008; Cappelaere et al, 2009).

9.2 Evaluation based upon SOAD Methodology Criteria

9.2.1 Evaluation according to Kohlborn et al

Up to now, there is no agreed list of evaluation criteria for SOAD methodologies against which the SERVUS Design Methodology may be assessed. A survey, however, just based upon available written documentation of six SOAD methodologies, was conducted amongst graduate students of computer sciences and industrial engineering from October 2007 till February 2008 and reported by Offermann and Bub (2009). They used a questionnaire to assess the criteria "quality", "completeness", "consistency" and "applicability" and applied a five-point scale, respectively. The quality was calculated as a combined value of the sub-criteria "perceived ease of use", "perceived usefulness" and "intention to use". They concluded the survey with the statements that the methodologies "show significant differences in quality", however, "ranking of all methods is relatively close to neutral" when considering all criteria.

Kohlborn et al (2009) developed a framework of comparison criteria that were systematically derived from similar studies. This list was applied to evaluate 30 SOAD methodologies. Their comparison is just performed on a descriptive and qualitative level and does not end up in a ranking list or recommendation. As it is the most comprehensive study known to the author, their list of nine criteria is used to evaluate the SERVUS Design Methodology:

Each of them are presented and analysed with respect to SERVUS as follows:

- **SOA concept**: Reflects whether an "approach's primary focus is on the derivation of business services, software services or both"[37].

  <u>SERVUS</u>: The focus is on both the business and software services. On the one hand, the starting points are business services whose functional and informational requirements are rephrased as requested resources. On the other hand, today, the existing capabilities of open geospatial service platforms are primarily software services (e.g., the existing geospatial standard services). However, with an increasing use of SOA approaches in large-scale projects there will be an increasing potential to also publish business services as "offered resources". In the RM-OA (Usländer (ed.), 2007), this type of services is called "thematic services" or "thematic support services" depending on their degree of reusability for an application domain. In addition, this trend is emphasized by the increasing number of application-oriented working groups within the OGC, e.g., for the domains of hydrology or meteorology (OGC-Press, 2009).

- **Delivery strategy for SOA:** Reflects whether "a particular approach supports the top-down strategy, where services are derived based on the analysis of business requirements (Erl, 2005), the bottom-up strategy, which focuses on the derivation of services based on an analysis of legacy systems on an as-needed basis, or the meet-in-the-middle strategy that combines the other two strategies. This criterion expresses whether an approach addresses organization-specific requirements (e.g. the need to leverage existing legacy systems).

  <u>SERVUS:</u> The resource-oriented approach of SERVUS follows a meet-in-the-middle strategy as both the requirements and the existing capabilities (of the "legacy systems") are considered in parallel in one design iteration step.

[37] Compared to the abstraction layers of Bieberstein et al (2006) (section 3.4.1), "business services" are in the focus of the business analyst and belong to the Enterprise and Process Layer, whereby "software services" correspond to "IT services" of the Service Layer and are in the focus of the IT specialist.

- **Lifecycle coverage**: Reflects which activities of a "full SOA lifecycle" are covered by the approach. A trivalent scale (0, +, ++) is used with the following semantics:
  - 0 stands for methods that focus on service identification and analysis only.
  - + represents methods with a service analysis and design focus.
  - ++ marks more comprehensive approaches including implementation or deployment.

  <u>SERVUS</u> focuses on service identification and analysis (0) and helps in the service design (+) as existing services are explicitly taken into account.
- **Degree of prescription**: Reflects whether an approach is rather **prescriptive** and defines a rigid, heavy-weight process with lots of details, or is rather agile and describes describe a more **lightweight**, flexible, less structured process that is adaptable. Again a trivalent scale (0, +, ++) with the following semantics:
  - 0 stands for methods that are very lightweight.
  - + represents methods with a moderate degree of prescription.
  - ++ marks highly prescriptive approaches.

  <u>SERVUS</u> provides a moderate degree of prescription (+). The SERVUS Design Process is inherently agile due to its iteration support and the co-development of requirements and capabilities that enables continuous adaptation on both levels. However, it prescribes the application of a defined formalism (the SERVUS Resource Model) into which the requirements have to be transformed (rephrasing activity).
- **Accessibility and validity**: Reflects whether the approach is **well documented**, whether the documentation is **accessible**, and whether the **validity** of the approach is made clear. A three-valued tuple (degree of documentation, accessibility, validity) is applied:
  - The degree of documentation is described by a textual comment.
  - The availability is described by a trivalent scale with 0 standing for a proprietary, not openly available method, + representing a method that is at least partially documented for public use, and ++ denotes a fully open method.
  - The validity is given by a textual comment.

  <u>SERVUS</u> may be characterized as (well documented, ++, controlled experiment) because its validity in a case or a field study needs a completely implemented design environment (section 7.2).

- **Adoption of existing processes/approaches/techniques/notations:** Reflects if the "approach utilizes already existing techniques, procedures and notations that can serve as a foundation for the approach". This criterion is described by a textual comment.

  SERVUS relies upon existing basic modelling techniques (e.g. a model-driven architecture, ontologies) and notations (e.g. use of UML). Furthermore, it is one of the key characteristics of SERVUS that existing service types of geospatial service platforms (e.g. catalogue service, map service, annotation service), although originally not conceived for this purpose, are exploited for the design process of new applications.

- **Regard to stakeholders:** Reflects if the approach addresses "the requirements of potential stakeholders regarding services", i.e. if the perspective of the service consumer is included or if solely the perspective of the service provider is addressed. This criterion is given by a textual comment.

  SERVUS inherently takes both perspectives into account.

- **Service classification and clustering:** Reflects if the approach "distinguishes different kinds of services. A trivalent scale is used with the following semantics: (0, +, ++) to indicate if there is only one single service concept in the method (0), if the method distinguishes between different service types but does not provide details (+) or if the method includes a detailed definition of the different service types (++).

  SERVUS does basically not rely upon a given service classification (+), however, the service classification of the OGC (ISO 19119:2005) is the prevalent classification that is considered due to the original motivation of the methodology from the needs of the environmental domain. Up to now, resource models were specified for a selected set of geospatial services, e.g. the OGC Sensor Observation Service (Usländer, 2008b).

- **Additional characteristics:** Is a "placeholder for any other important characteristics of the analysed methods that seem important enough to be pointed out".

  SERVUS was derived from the set of requirements for a service-oriented design methodology introduced in section 1.4.3. The following section elaborates on how these requirements were met.

9.2.2 Evaluation according to the Requirements for Service-oriented Design

In section 1.4.3 twelve requirements (R.1 – R.12) for a service-oriented design methodology have been specified to provide guidance to the conception of the SERVUS Design Methodology.

Category	Requirement	Assess-ment	Comment
Human Compre-hension	R.1 User Feedback	++	Iterative design approach with a step-wise refine-ment of the user requirements
	R.2 Usability	++	To be validated in large-scale projects.
	R.3 User-near Analysis	+++	Explicit objective of the SERVUS Resource Model resulting in a network of requested and offered resources.
Analytical Power	R.4 Architec-tural Frame-work	+++	SERVUS is based upon the geospatial reference model RM-OA.
	R.5 Property Coverage	++	Qualitative requirements still to be considered.
	R.6 Standards Compliance	+++	Explicit consideration of geospatial services and information models of the OGC.
	R.7 Semantic Enrichment	++	Practicability of the approach is dependent on the effort to build an adequate ontology widely ac-cepted in the user community.
	R.8 Traceabil-ity	++	Practicability of the approach to be validated in the context of a completely integrated service design environment.
	R.9 Service Factoring Guidance and Evaluation	+	Rephrasing of use case actions into requested resources and their mapping to services provides some guidance; however, the evaluation of service design is missing.
Realism	R.10 Effi-ciency Gain	+	Requires automation support of the SERVUS design activities and validation in a large-scale project.
	R.11 Iteration and Docu-mention Sup-port	++	SERVUS Design Process is based upon iterations. Resource Model enables co-design and documen-tation of both requirements and design artefacts.
	R.12 Capabil-ity Discovery	++	Explicitly supported but needs support for the ranking of search results as well as interpretation of textual specifications.

Table 9-5: Evaluation against the Requirements of Human Comprehension, Analytical Power and Realism

Table 9-5 shows the evaluation of the SERVUS characteristics against these requirements, using a trivalent scale (0, +, ++) with the following semantics:

-	not supported by SERVUS
+	supported by SERVUS; implicit property; needs strong enhancement
++	explicitly supported by SERVUS; needs enhancement or not yet validated in practice
+++	strong support; strength of the SERVUS Design Methodology

9.3 Results and Outlook

9.3.1 Summary of the Results

This thesis provides a contribution to the complex topic of service-oriented design methodologies for information systems. As stated in several studies (Papazoglou et al, 2007; Kontogiannis et al, 2007; Offermann and Bub, 2009) this topic will remain on the research agenda for the upcoming years. Van den Heuvel et al (2009) argue that **Software Service Engineering (SSE)** is an "emerging discipline that entails a departure from traditional software engineering", which is primarily due to

- its **open-world assumption**, i.e. service applications have to be designed without any knowledge about the context in which they will be executed, and
- its **holistic approach**: SSE demands an "interdisciplinary approach towards the analysis and rationalization of business processes, design of supporting software service systems, their realization, deployment, provisioning and monitoring and adaptation. This implies that SSE concepts, models, methods are integrated and tools are interoperable, adhering to open standards and offering integrated support for several stakeholders".

The SERVUS approach claims to deliver innovative ideas to this new service engineering discipline, especially for the geospatial software community. As a conceptual foundation it provides a reference model (DA.1) that is based upon geospatial standards and reached best-practices status within the OGC. A core concept of this reference model is the Semantic Resource (DA.2) that allows the system designer to bridge the conceptual gap between the requirements of the expert user and the technical service infrastructure. The SERVUS Design Methodology (DA.4) leverages this uniform modelling approach. On the one hand, it enables to exploit the capabilities of existing service networks to support the design of new applications. On the other hand it facilitates the stepwise tracing of the SOA design.

Although originally tailored to the application domain of environmental management, the SERVUS approach may also be applied to other application domains, especially if geospatial resources and the side-condition to rely upon OGC standards play a major role.

9.3.2 Future Research Topics

Based upon the experience with the SERVUS Design Methodology, the following research topics are worthwhile to be tackled in the future:

- Ontology-based Ranking of Search Results

  As discussed in section 4.3.4.3 the SERVUS discovery activity may heavily benefit from the possibility of query expansion techniques based upon domain ontologies. Query expansion is also supported in the SERVUS Implementation Architecture by the integration of the SemCat component (section 7.2.1) in the form of the possibility to express so-called "semantic bounding boxes". Although this technique helps in the recall of queries the subsequent matching activity needs knowledge about the relevance of each item in the result set, or, in more detail, which item in the result set is due to which query element. Usually, this relevance is expressed in form of a ranking of the result set. The computation of the ranking list should take the ontology and especially the semantic similarity to the original query of the user into account. It would be beneficial to investigate if existing approaches to ontology-based ranking that were mainly conceived for the ranking of Web search results (e.g., Tran et al, 2007; Sriharee, 2006; Fang et al, 2007; Shamsfard, Nematzadeh and Motiee, 2006) could also be applied to the ranking in ontology-based geospatial catalogue systems.

- Domain Modelling during the Problem Analysis

  One of the biggest problems in all ontology-based approaches is the challenge and the costs to define an adequate ontology that is shared by the community of the users of the system (Stuckenschmidt, 2009). Based upon the experience on building a topographic ontology Hart et al (2008) devise a "methodology that enables domain experts working with ontology engineers to construct ontologies that have both a conceptual, human readable aspect and a computation aspect that is interpretable by machines". A key part of their methodology is "Rabbit" which is a controlled natural language based on English. Rabbit allows the domain expert to easily understand the ontology whilst supporting all the OWL-DL language features.

  It would be worthwhile to scrutinize if such ontology engineering methodologies could be combined with semantic annotation techniques already embedded within the SERVUS Implementation Architecture (Bügel and Usländer, 2009). They may help to detect ontological concepts in textual descriptions of applications, application processes and use cases. The challenge is to link such concepts to domain ontologies and to overcome term ambiguities and inconsistencies that are inherent in documented user requirements.

- Application of Methods for Semantic Integration

  The SERVUS Design Methodology (DA.4) relies upon a matching between requested and offered resources which are both specified in a uniform manner according to the SERVUS Resource Model (DA.2). Tools for automatic or semi-automatic resource matching are indispensable in order to increase the efficiency of the design methodology. As a conceptual basis for such tools its should be investigated if existing algorithms that were developed for mapping discovery (Noy, 2004) and the analysis and discovery of concept similarity in ontologies (e.g. Ge and Qiu, 2008) could be re-used and adapted to the specific structure and properties of resources, their representations, their semantic model references and their surrounding context in a resource network.

- Uncertainty of References in the SERVUS Resource Model

  The current SERVUS Resource Model allows the system architect to semantically annotate a resource. This is accomplished by a reference of the resource and/or the resource attributes to a concept in a semantic model (ontology), following the basic idea of the SAWSDL technology (Farrell and Lausen (eds.) 2007). However, very often there is an uncertainty associated with this semantic reference if the concepts in the ontology do not exactly fit or if there are multiple possibilities. This uncertainty could be formalized by extending the definition of a semantic resource (definition 5.5 in section 5.3.2) towards a probabilistic semantic resource as follows:

  Definition (9.1): A **Probabilistic Semantic Resource** is a 5-tuple SR = ⟨P, M, Rep, l, y⟩ such that:

  - P is a set of resource properties including a unique identifier.
  - M is the list of supported methods of the resource.
  - Rep is the list of representations of the resource.
  - l: P ∪ SR → SM is a labelling function that associates properties, as well as the semantic resource itself, with concepts taken from a semantic model SM.
  - y: l → [0,1] assigns a value that signifies the certainty of the concept mapping, i.e. the belief of the system designer on whether the resource or the properties belong to the concepts of the semantic model SM.

  Further research is required to investigate the consequences for the SERVUS Design Methodology. For instance, as the set of resource properties belongs to the domain of the labelling function l, it may also be used to express uncertainty of resource associations, in particular, uncertainties in the matching of requested to offered resources. This may help in the selection of the best-fitting matches.

- Application of SERVUS in Case or Field Studies

  As mentioned in the evaluation section 9.1, it would be beneficial to study the applicability of the SERVUS-related design artefacts DA.2, DA.4 and DA.5 in business environments. However, as a pre-requisite to carry out case or field studies, there is a need for a complete implementation of the SERVUS Design Methodology in an integrated design environment with a high-quality user interface in order to be accepted by system architects. The current SERVUS Implementation Architecture described in section 7 provides one promising option for such an implementation as it is based upon standard components already proven in the practice of geospatial and environmental applications.
- Definition of the SERVUS Design Process in terms of a Standard Engineering Meta-model

  As mentioned in section 2.3.1 the Object Management Group (OMG) has recently specified a Software and Systems Process Engineering Meta-model called SPEM (OMG, 2008d). If this meta-model gets acceptance in the research community and/or the software engineering industry, it would be beneficial to define the SERVUS design artefacts, especially the SERVUS Resource Model (DA.2) and the SERVUS Design Methodology (DA.4) in terms of the SPEM notation.

9.3.3 Motivation: Revisited

Imagine…the hydrologist that issued the tender for the new water information system and the computer scientist whose company won the contract for the software development agreed to apply the SERVUS Design Methodology for their project. They took this decision as they needed a tool to overcome the language barriers in talking about the thematic problem to be solved and the existing software solutions. The SERVUS Resource Model appeared to provide the adequate abstraction level for both communities and thus to better control and trace the design steps.

One the one side, it allows the hydrologist to exploit and integrate the high-level domain ontologies that have been agreed in an exhaustive effort in the hydrology community of the last years and extend them for the project.

On the other side, the computer scientist may leverage the huge base of operational environmental monitoring and modelling services resulting from the various international initiatives and projects of the last years carried out in public-private partnerships. The capabilities of these services may be directly detected by Internet search engines as they are provided in resource-oriented style, the "language" of the Internet. Both human-readable and machine-readable resource representations are available, partly semantically annotated with conceptual links to standards of the geospatial community. Thus, the system designer may quickly check the adequacy of an offered service. However, also software agents triggered by the network of requested resources may be used to automatically "crawl through" the resource offerings, match them with the requested resources and propose matching results to the system designer.

For sure, a design methodology is only one of the success factors in software engineering projects. Creativity, competence and soft skills of all members in a project team will remain decisive. However, the choice of an adequate design methodology may lay the foundation for an efficient software development.

This thesis submits a fitting proposal for the application domain of Environmental Information Systems anticipating that service networks will be developed and deployed in future as an operational and dependable infrastructure. As of today, large sums of money and human resources are invested on regional, national and international levels to conceive, set-up and run spatial data infrastructures in line with and partly extending the INSPIRE directive. It is expected that the increasing availability of geospatial data will encourage the development of higher-level thematic services, also encompassing Earth observation data and services through world-wide initiatives such as GEOSS (GEO, 2008). These developments will turn spatial data infrastructures into service infrastructures based upon international geospatial standards. This trend will have an increasing impact upon the manner how environmental information systems and geospatial software applications will be designed and structured in the future.

The SERVUS Design Methodology proposed by this thesis aims at supporting both the user and the system architect in the recognition, analysis and exploitation of this still dormant but emerging potential for the sake of new and innovative environmental applications.

10 References

10.1 Bibliography

Note: All referenced Internet sites (URLs) were accessed and checked on 26 October 2009.

Alonso et al, 2004

Alonso, G., Casati, F., Kuno, H. and Machiraju, V. "Web Services: Concepts, Architectures and Applications. Data-Centric Systems and Applications". Springer-Verlag Berlin Heidelberg, 2004.

Alexander, 2002

Alexander, D.E. "Principles of Emergency Planning and Management". Terra Publishing, Harpenden, Herts, UK, and Oxford University Press, New York. (hardback and paperback), 2002.

Alexiev et al, 2005

Alexiev, V., Beu, M., de Bruijn, J., Fensel, D, Lara, R. and Lausen, H. "Information Integration with Ontologies – Experiences from an Industrial Showcase". John Wiley & Sons Ltd., West Susses, England, ISBN 0-470-01048-7, 2005.

Annoni et al, 2005

Annoni, A., Bernard, L., Douglas, J. Greenwood, J., Laiz, I., Lloyd, M., Sabeur, Z., Sassen, A.-M., Serrano, J.-J. and Usländer, T. "Orchestra: Developing a Unified Open Architecture for Risk Management Applications". In: van Oosterom et al (eds.) (2005): "Geo-information for Disaster Management". Springer-Verlag Berlin Heidelberg, pp. 1-17, 2005.

Arsanjani et al, 2008

Arsanjani, A., Ghosh S., Allam, A., Abdollah, T., Ganapathy, S. and Holley, K. "SOMA: A method for developing service-oriented solutions". IBM Systems Journal, Vol. 47, No. 3, pp. 377-396, 2008.

Asadi and Ramsin, 2008

Asadi, M. and Ramsin, R. "MDA-based Methodologies: An Analytical Survey". In: Schieferdecker, I. and Hartman, A. (eds.): European Conference on Model-driven Architecture – Foundations and Applications (ECMDA-FA) 2008, LNCS 5095, pp. 419-431, Springer-Verlag Berlin Heidelberg, 2008.

Atkinson and Kühne, 2005

Atkinson, C. and Kühne, T. "A Generalized Notion of Platforms for Model-Driven Development". In: Beydeda, S., Book, M. and Gruhn, V. (eds.) "Model-Driven Software Development". ISBN-10 3-540-25613-X Springer, 2005.

Avison and Fitzgerald, 2003

Avison, D.E. and Fitzgerald, G. "Where Now for Development Methodologies?". Communications of the ACM, Vol. 46, No. 1, pp. 79-82, 2003.

Baader et al, 2007

Baader, F., Calvanese, D. , McGuiness, D.L., Nardi, D. and Patel-Schneider, P.F. "The Description Logic Handbook: Theory, Implementation, Applications". 2nd Edition, Cambridge University Press, ISBN 978-0-521-87625-4, 2007.

Baeza-Yates and Ribeiro-Neto, 1999

Baeza-Yates, R. and Ribeiro-Neto, B. "Modern Information Retrieval". ISBN 0-201-39829-X, ACM Press, 1999

Balasubramanian, 2008

Balasubramanian, R. "REST-Inspired SOA Design Patterns". SOA Magazine Issue XXIV: December 2008.
http://www.soamag.com/I24/1208-3.asp

Bandholtz et al, 2009

Bandholtz, T., Fock, J., Legat, R., Nagy, M., Schleidt, K. and Plini, P. "Shared Terminology for the Single Environmental Information System". Proceedings of the EnviroInfo 2009, Berlin, 2009.

Bechhofer et al, 2004

Bechhofer, S., Harmelen, F. v., Hendler, J., Horrocks, I., McGuiness, D., Patel-Schneider, P.F. and Stein, L.A. "Web ontology language reference". W3C recommendation, 2004.
http://www.w3.org/TR/2004/REC-owl-ref-20040210/

Béjar et al, 2009

Béjar, R., Latre, M. Á., Nogueras-Iso, J., Muro-Medrano, P. R. and Zarazaga-Soria, F.J. "Systems of Systems as a Conceptual Framework for Spatial Data Infrastructures". Article under Review for the International Journal of Spatial Data Infrastructures Research (IJSDIR), Vol (4), 2009.
http://ijsdir.jrc.ec.europa.eu/index.php/ijsdir/article/view/124/105

Berente and Lyytinen, 2007

Berente, N. and Lyytinen, K. "What is Being Iterated ? Reflections on Iteration in Information System Engineering Processes". In: Krogstie et al (eds.), (2007): Conceptual Modelling in Information Systems Engineering, pp. 261-278, 2007.

Bercovici, Fournier and Wecker, 2008

Bercovici, A., Fournier, F. and Wecker, A.J. "From Business Architecture to SOA Realization Using MDD". In: Schieferdecker, I. and Hartman, A. (eds.): European Conference on Model-driven Architecture – Foundations and Applications (ECMDA-FA) 2008, LNCS 5095, pp. 381-392, Springer-Verlag Berlin Heidelberg, 2008.

Berners-Lee, Hendler and Lassila, 2001

Berners-Lee, T., Hendler, J. and Lassila, O. "The Semantic Web". Scientific American, 284(5):34-43, 2001.

Bernard et al, 2005a

Bernard, L., Kannellopoulos, I., Annoni, A. and Smits, P. "The European geoportal – one step towards the establishment of a European Spatial data Infrastructure". Elsevier Journal Computers, Environment and Urban Systems 29 (2005), pp. 15-31, 2005.

Bernard et al (eds.), 2005b

Bernard, L., Fitzke, J. and Wagner, R. M. "Geodateninfrastruktur – Grundlagen und Anwendungen". Herbert Wichmann Verlag, ISBN 3-87907-395-3, 2005.

Bieberstein et al, 2006

Bieberstein, N., Bose, S., Fiammante, M., Jones, K. and Shah, R. "Service-Oriented Architecture (SOA) Compass – Business Value, Planning and Enterprise Roadmap". IBM Press developerWorks® Series. ISBN 0-13-187002-5, 2006.

Biliouris and van Orshoven, 2009

Biliouris, D. and van Orshoven, J. "WISE: Water Information System for Europe - issues and challenges for Member states". In: Hřebíček, J. et al (eds.) (2009): Proceedings of the European conference Towards eEnvironment, ISBN 978-80-210-4824-9, pp. 430-436, 2009.

Booz Allen Hamilton, 2005

Booz Allen Hamilton. "Geospatial Interoperability Return on Investment Study", National Aeronautics and Space Administration (NASA) Geospatial Interoperability Office, Booz Allen Hamilton, 2005.
http://www.egy.org/files/ROI_Study.pdf

Borgida and Brachman, 2007

Borgida, A. and Brachman, R.J. "Conceptual Modeling with Description Logics". Chapter 10 of Baader et al (2007): The Description Logic Handbook, 2007.

Bræk and Melby, 2005

Bræk, R. and Melby, G. "Model-Driven Service Engineering". In: Beydeda, S., Book, M. and Gruhn, V. (eds.): Model-Driven Software Development. ISBN-10 3-540-25613-X Springer Verlag, pp. 385-401, 2005.

Brambilla et al, 2007

Brambilla, M., Ceri, S., Facca, F.M., Celino, I., Cerizza, D. and Della Valle, E. "Model-Driven Design and Development of Semantic Web Service Applications". ACM Transactions on Internet Technology, Vol. 8, No. 1, Article 3, 2007.

Brickley and Guha (eds.), 2004

Brickley, D. and Guha, R.V. (eds.). "RDF Vocabulary Description Language 1.0: RDF Schema". W3C Recommendation, 2004. http://www.w3.org/TR/rdf-schema

Brockmans et al, 2006

Brockmans et al (eds.). "Proceedings of the 2nd Workshop on Meta-Modelling". WoMM 2006, pp. 47-59, ISSN 1617-5468, 2006.

Bügel and Usländer, 2009

Bügel, U. and Usländer, T. "Discovery and Analysis of Environmental Information based on Formalised Terminology". In: Hřebíček, J. et al (eds.) (2009): Proceedings of the European conference Towards eEnvironment, ISBN 978-80-210-4824-9, pp. 583-590, 2009.

Bubenko, 2007

Bubenko jr., J.A. "From information Algebra to Enterprise Modelling and Ontologies – a Historical Perspective on Modelling for Information Systems". In: Krogstie, J.; Opdahl A.L.; Sjaak Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering. Springer-Verlag ISBN 978-3-540-72676-0, pp. 1-18, 2007.

Busskamp, Fretter and Usländer, 2004

Busskamp, R., Fretter, K. and Usländer, T, "Reporting Schemes for the European Water Framework Directive in the context of the Internet Portal WasserBLIcK and INSPIRE". In: Proceedings of the 18th International Conference on Informatics for Environmental Protection (EnviroInfo 2004), Brno, Czech Republic, 2004.

Cappelaere et al, 2009

Cappelaere, P., Frye, S. and Mandl, D. "SPS 2.0 -- A RESTful Approach". NASA EO-1 Team, 2009.

Chang and Kim, 2007

Chang, S.H. and Kim, S.D. "A Systematic Approach to Service-Oriented Analysis and Design". In: Münch, J. and P. Abrahamsson: Product-Focused Software Process Improvement, LNCS 4589, pp. 374-388, Springer-Verlag, 2007.

Coene and Gasser, 2007

Coene, Y. and Gasser, R. "Joint Operability Workshop Report. Towards a single information space for Environment in Europe", Frascati, 2007. ftp://ftp.cordis.europa.eu/pub/ist/docs/environment/sise-workshop-report_en.pdf

Cox (ed.), 2007

Cox, S. (ed.). “Observations & Measurements - Part 1: Observation Schema”. OGC Document 07-022r1, approved as OpenGIS® specification, 2007.
http://portal.opengeospatial.org/files/?artifact_id=22466

Craglia et al, 2003

Craglia, M. and INSPIRE Framework Definition Support Group. “Contribution towards the extended impact assessment of INSPIRE”. Environment Agency for England and Wales, 2003.
http://www.ec-gis.org/sdi//ws/costbenefit2006/reference/original_xia_report.pdf

CSIRO, 2007

Commonwealth Scientific and Industrial Research Organisation (CSIRO). “Water Resources Operation Network Australia (WRON) Brochure”, 2007.
http://wron.net.au/documents/WRON_brochure_March207-print.pdf

de Amicis et al (eds.), 2009

de Amicis, R.; Stojanovic, R. and Conti, G. (eds.). “GeoSpatial Visual Analytics: Geographical Information Processing and Visual Analytics for Environmental Security”. NATO Science for Peace Security Series, Springer Science + Business Media B.V., ISBN 978-90-481-2897-6, 2009.

de la Torre et al (eds.), 2007

De la Torre, J., Sutton, T., Meganck, B., Vieglais, D., Stewart, A., Brwer, P. and Catanach, T.A. “BioGeoSDI workshop - GeoInteroperability Testbed Pilot Project Version: 0.6“, 2007.
http://www.tdwg.org/fileadmin/subgroups/meeting_reports/GIG_BioGeoSDI_Report.pdf

Delphi, 2003

Delphi Group. “The Value of Standards A Delphi Study”. Boston, 2003.
http://www.delphigroup.com

Denzer, 2005

Denzer, R. “Generic Integration of Environmental Decision Support Systems – State-of-the-Art”. Elsevier Journal Environmental Modelling and Software 20 (2005), pp. 1217-1223, 2005.

DEWS, 2009

DEWS Consortium. “2nd Flyer of the Distant Early Warning System for Tsunami”. European STREP Proposal No: 045453, 2009.
http://www.dews-online.org

Dikanski et al, 2009

Dikanski, A., Emig, C. and Abeck, S. “Integration of Security Products in Service-oriented Architecture”. Third International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2009, Athens/Glyfada, Greece, 2009.
http://doi.ieeecomputersociety.org/10.1109/SECURWARE.2009.8

Di Penta et al, 2009

Di Penta, M., Bastida, L., Sillitti, A., Baresi, L., Maiden, N., Melideo, M. Tilly, M., Spanoudakis, G., Gorroñogoitia Cruz, J., Hutchinson, J. and Ripa, G. “SeCSE – Service-centric System Engineering: An Overview”. In : Di Nitto, E., Sassen, A.-M. and Zwegers, A. (eds.): At Your Service – Service-Oriented Computing from an EU Perspective. MIT Press. ISBN 978-0-262-04253-6, pp. 241-272, 2009.

Dufourmont (ed.), 2004

Dufourmont, H. (ed.). “Results Task Force XIA”. Eurostat, Luxembourg, 2004.

Dobson, 2005

Dobson, G., Lock, R. and Sommerville, I. "QoSOnt: a QoS Ontology for Service-Centric Systems”. 31st EUROMICRO Conference on Software Engineering and Advanced Applications, pp.80-87, 2005.

Dobson and Sawyer, 2006

Dobson, G. and Sawyer, P. “Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web”. International Seminar on "Dependable Requirements Engineering of Computerised Systems at NPPs", Institute for Energy Technology (IFE), Halden, 2006.
http://www.comp.lancs.ac.uk/computing/users/dobsong/papers/dre06_ontology_based_re.pdf

Dobson et al, 2007

Dobson, G.; Hall, S. and Kotonya, G. “A Domain-Independent Ontology for Non-Functional Requirements”, IEEE International Conference on e-Business Engineering (ICEBE'07), pp.563-566, 2007.

Douglas et al, 2008

Douglas, J., Usländer, T., Schimak, G., Esteban, J.F. and Denzer, R. “An Open Distributed Architecture for Sensor Networks for Risk Management”. Sensors 2008, 8, ISSN 1424-8220, pg. 1755-1773, MDPI, 2008.
http://www.mdpi.com/1424-8220/8/3/1755/pdf

EC, 2000

Directive 2000/60/EC of the European Parliament and of the Council establishing a framework for the Community action in the field of water policy, 2000.

EC, 2003a

European Commission. "Environment and Health Strategy". COM2003-338 final, 2003.

EC, 2003b

Directive 2003/4/EC of the European Parliament and of the Council on public access to environmental information and repealing Council Directive 90/313/EEC, 2003.

EC, 2007

Directive 2007/2/EC of the European Parliament and of the Council establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), 2007.

EC, 2008

European Commission, Communication from the Commission to the Council, the European Parliament, The European Economic and Social Committee and the Committee of the Regions. "Towards a Shared Environmental Information System (SEIS)", COM (2008) 46 final, 2008.

Egenhofer, 2002

Egenhofer, M.J. "Toward the Semantic Geospatial Web", Tenth ACM International Symposium on Advances in Geographical Information Systems, McLean, Virginia, 2002.

eEnvironment, 2009

Resolution of the Delegates of the Conference "Towards eEnvironment (Challenges of SEIS and SISE: Integrating Environmental Knowledge in Europe)"., Prague, 2009.
http://www.e-envi2009.org/090327-e-environment-conference-conclusions-final.pdf

Erl, 2005

Erl, T. "Service-Oriented Architecture. Concepts, Technology, and Design". 4th edition, Upper Saddle River, NJ., Prentice Hall, 2005.

Erl, 2008a

Erl, T. "SOA: Principles of Service Design". ISBN 0-13-234482-3. Prentice Hall, 2008.

Erl, 2008b

Erl, T. "SOA design patterns". ISBN 0-13-613516-1. Prentice Hall, 2008.

Emig, Weisser and Abeck, 2006

Emig, C., Weisser, J. and Abeck, S. "Development of SOA-Based Software Systems - an Evolutionary Programming Approach". IEEE Conference on Internet and Web Applications and Services ICIW'06, Guadeloupe, 2006.

Emig et al, 2007

Emig, C., Krutz, K., Link, S., Momm, C. and Abeck, S. “Model-Driven Development of SOA Services”, Research Report University of Karlsruhe, 2007. http://www.cm-tm.uka.de/CM-Web/07.Publikationen/%5BEK+07%5D_Model_Driven_Development_of_SOA_Services.pdf

Emig et al, 2008

Emig, C., Kreuzer, S., Abeck, S., Biermann, J. and Klarl, H. “Model-Driven Development of Access Control Policies for Web Services”. 9th IASTED International Conference on Software Engineering and Applications SEA 2008, Orlando/Florida, 2008.

ERDAS, 2009

ERDAS APOLLO Catalog, 2009 http://apollo2.erdas.com:8081/erdas-apollo/catalog/

Euzenat and Shvaiko, 2007

Euzenat, J. and Shvaiko, P. “Ontology Matching”. Springer, Heidelberg (DE), 2007.

Fabbri and Weets, 2005

Fabbri, K. and Weets, G. “ICT for Environmental Risk Management in the EU Research Context”. In: van Oosterom et al (eds.) (2005): “Geo-information for Disaster Management”. Springer-Verlag Berlin Heidelberg, pp. 51-56, 2005.

Fang et al, 2007

Fang, J., Guo, L., Wang, X.D. and Yang, N. ”Ontology-Based Automatic Classification and Ranking for Web Documents”. Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), 2007.

Fensel et al, 2006

Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D. and Dominigue, J. “Enabling Semantic Web Services – The Web Service Modeling Ontology”. Springer-Verlag Berlin Heidelberg New York, ISBN-13 978-3-54034-519-1, 2006.

Fensel et al, 2008

Fensel, D., M. Kerrigan and M. Zaremba (eds.), “Implementing Semantic Web Services - The SESA Framework”. ISBN 978-3-540-77019-0, Springer, 2008.

Farrell and Lausen (eds.) 2007

Farrell, J. and Lausen, H. (eds.). “Semantic Annotations for WSDL and XML Schema”. W3C Proposed Recommendation, 05 July 2007. http://www.w3.org/TR/sawsdl/

Fielding, 2000

Fielding, R.T. “Architectural Styles and the Design of Network-Based Software Architectures”. Doctoral dissertation, University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Franceschetti and Grossi, 2008

Franceschetti, G. and Grossi, M. “Homeland Security Technology Challenges – From Sensing and Encrypting to Mining and Modeling”. ISBN-13 978-1-59693-289-0, Artech House Boston London, 2008.

Ge and Qiu, 2008

Ge, J. and Qiu, Y. “Concept Similarity Matching Based on Semantic Distance”. Proceedings of the 4th International Conference on Semantics, Knowledge and Grid. ISBN 978-0-7695-3401-5, 2008.

GEO, 2008

GEO – Group on Earth Observation. “The Global Earth Observation System of Systems (GEOSS) 10-Year Implementation Plan”. GEOSS Website, 2008. http://www.earthobservations.org

GEOSS AIP, 2008

GEO – Group on Earth Observation. “Architecture Implementation Pilot (AIP), Phase 2: IOC Augmentation - Call for Participation (CFP)”. GEO Task Team AR-07-02, 2008. http://www.earthobservations.org/documents/aip/20080626_geo_aip2_call_for_participation.pdf

Giddy et al, 2009

Giddy, J., Hardisty, A., Hernandez-Ernst, V., Poigné, A. and Voss, A. “LifeWatch Reference Model Version 0.2”. LifeWatch – e-Science and Technology infrastructures for biodiversity data and observatories, EU FP7 Infrastructures Project 211372 (INFRA-2007-2.2-01), 2009. http://lifewatch.eu/images/stories/PDFs/lw_referencemodel.pdf

Gomadam et al, 2008

Gomadam, K., Ranabahu, A. and Sheth, A. “SA-REST: Semantically enhancing REST Services”. SWS-XG position paper and discussion draft. Kno.e.sis Services Science Lab, Wright State University, Dayton, Ohio, 2008. http://knoesis.wright.edu/research/srl/standards/sa-rest/

Goyal and Wesenthaler, 2004

Goyal, S. and Westenthaler, R. “RDF Gravity (RDF Graph Visualization Tool) – User Documentation”. Knowledge Information Systems Group, Salzburg Research, 2004. http://semweb.salzburgresearch.at/apps/rdf-gravity/user_doc.html

Goodall et al, 2008

Goodall, J.L., Horsburgh, J.S., Whiteaker, T.L., Maidment, D.R. and Zaslavsky, I. "A first approach to web services for the National Water Information System". Elsevier Environmental Modelling & Software 23 (4), 404–411, 2008.

Graham et al (eds.), 2006

Graham, S., Hull, D. and Murray, B. (eds.). "Web Services Base Notification 1.3 (WS-BaseNotification)". OASIS Standard, OASIS Document identifier wsn-ws_base_notification-1.3-spec-os, 2006.

Havlik et al, 2007

Havlik, D., Schimak, G., Denzer, R. and Stevenot, B. "SANY (Sensors Anywhere) Integrated Project". In: Swayne, D.A. and Hřebíček, J. (eds.) Proceedings of the ISESS 2007, Prague, Czech Republic, 2007.

Hadley, 2006

Hadley, M.J. "Web Application Description Language (WADL)". Sun Microsystems Inc., Java GlassFish project, 2006.
http://wadl.dev.java.net/#spec

Hart and Dolbear, 2006

Hart, G. and Dolbear, C. "So what's so special about spatial ?", Workshop Terra Cognita, Directions to the Geospatial Semantic Web, ISWC 2006, Athens, Georgia, USA, 2006.

Hart et al, 2008

Hart, G., Johnson,M. and Dolbear, C. "”Rabbit: Developing a controlled natural language for authoring ontologies". In: Bechhofer, S. et al (eds.): Proceedings of the 5th European Semantic Web Conference ESWC 2008, Lecture Notes in Computer Science 5021 Springer Verlag, 2008.

Hilbring and Schleidt, 2009

Hilbring, D. and Schleidt, K. "Automatic creation of INSPIRE related metadata from Sensor Web Enablement Services". AGILE 2009 Pre-Conference Workshop "Challenges in Geospatial Data Harmonization", 2009.

Hatley, Hruschka and Pirbhai, 2000

Hatley, D.J., Hruschka, P. and Pirbhai, S. "Process for System Architecture and Requirements Engineering". Dorset House Publishing Company, Inc., ISBN-13 978-0-93263-341-5, 2000.

Henderson and Venkatraman, 1993

Henderson, J. and Venkatraman, N. "Strategic Alignment: Leveraging Information Technology for Transforming Organizations". IBM Systems Journal (32:1), 1993.

Henderson-Sellers and Gonzales-Perez, 2008

Henderson-Sellers, B. and Gonzales-Perez, C. “Standardizing Methodology Metamodelling and Notation: An ISO Exemplar”. In: Kaschek, R et al (eds.): Information Systems and e-Business Technologies, 2nd UNISCON 2008 Conference, LNBIP 5, pp. 1-12, Springer-Verlag, 2008.

Hesse, 2008

Hesse, W. “Engineers Discovering the ‘Real World’ – From Model-Driven to Ontology-Based Software Engineering”. In: Kaschek, R et al (eds.): Information Systems and e-Business Technologies, 2nd UNISCON 2008 Conference, LNBIP 5, Springer-Verlag, pp. 136-147, 2008.

Henderson and Venkatraman, 1993

Henderson, J. and Venkatraman, N.. ”Strategic Alignment: Leveraging Information Technology for Transforming Organizations”. IBM Systems Journal (32:1), 1993.

Hevner et al, 2004

Hevner A.R., March S.T. and Park, J. “Design Science in Information System Research”. MIS Quarterly Vol. 28 No. 1, pp.75-105, 2004.

High, Krishnan and Sanchez, 2008

High Jr., R., Krishnan, G. and Sanchez, M. “Creating and maintaining coherency in loosely coupled systems”. IBM Systems Journal, Vol. 47, No 3, pp. 357-375, 2008.

Hilbring and Schleidt, 2009

Hilbring, D. and Schleidt, K. “Automatic creation of INSPIRE related meta-information from Sensor Web Enablement Services”. Pre-conference workshop WS7 “Challenges in Geospatial Data Harmonisation” of the 12th AGILE International Conference on Geographic Information Science - Advances in GIScience, 2009.

Hilbring and Usländer, 2006

Hilbring, D., and Usländer, T. “Catalogue Services Enabling Syntactical and Semantic Interoperability in Environmental Risk Management Architectures”. Proceedings of the 20th International Conference on Informatics for Environmental Protection (EnviroInfo 2006), Graz, Austria, ISBN-10 3-8322-5321-1, pp. 39-46, 2006.

Hilbring and Usländer, 2008

Hilbring, D., and Usländer, T. “Ontology-Based Discovery of Geoscientific Information and Services”. Session on Semantic Interoperability, Knowledge and Ontologies in the European Geosciences Union General Assembly, Vienna, 2008.

Hill, 2006

Hill, L.L. "Georeferencing – The Geographic Associations of Information". Digital Libraries and Electronic Publishing. ISBN-13 978-0-262-08354-6. The MIT Press Cambridge, Massachusetts, London, England, 2006.

Hilty and Rautenstrauch, 1997

Hilty, L. M. and Rautenstrauch, C. "Environmental Information Systems for Production and Recycling". Keynote lecture at the 2nd International Symposium on Environmental Software Systems (ISESS), Whistler (Canada). New York: Chapman & Hall, pp. 21-29, 1997.

Hřebíček et al (eds.), 2009

Hřebíček, J. (main editor), Hradec, J., Pelikán, E., Mirovský, O., Pillmann, W., Holoubek, I., and Bandholtz, T. (eds.). "Proceedings of the European conference of the Czech Presidency of the Council of the EU: Towards eEnvironment (Challenges of SEIS and SISE: Integrating Environmental Knowledge in Europe)", Prague, 25-27 March 2009, ISBN 978-80-210-4824-9, 2009.

IEEE 1471-2000

IEEE. "Recommended Practice for Architecture Description of Software-Intensive Systems". 2000.

IEEE 610.12-1990

IEEE. "Standard Glossary of Software Engineering Terminology". 1990.

ISO 19101:2004

ISO. "Geographic information – Reference model". 2004.

ISO 19109:2005

ISO. "Geographic information – Rules for application schema". 2005.

ISO 19115:2003

ISO. "Geographic Information – Metadata". 2003.

ISO 19119:2005

ISO. "Geographic Information – Services". 2005.

ISO/IEC 10746-1:1998

ISO/IEC. "Information technology - Open Distributed Processing – Reference model". 1998.

ISO/IEC 10746-2:1996

ISO/IEC. "Information technology - Open Distributed Processing – Foundations". 1996.

ISO/IEC 19793:2008

ISO/IEC. "Information technology - Open Distributed Processing – Use of UML for ODP system specifications". 2008.

ISWC, 2007

The 6th International Semantic Web Conference (ISWC) and the 2nd Asian Semantic Web Conference (ASWC). "Service Matchmaking and Resource Retrieval in the Semantic Web". Proceedings of the Workshop 1, BEXCO Busan, Korea, 2007.

Jacobson, 1987

Jacobson, I. "Object-Oriented Development in an Industrial Environment". Proceedings of OOPSLA'87, pp. 183-191, 1987.

Jacobson and Ng, 2005

Jacobson, I. and Ng, P.-W. "Aspect-Oriented Software Development with Use Cases". The Addison-Wesley Object Technology Series, ISBN 0-321-26888-1, 2005.

Janowicz, 2006

Janowicz, K. "Similarity-based Retrieval for Geospatial Semantic Web Services Specified using the Web Service Modeling Language (WSML-Core)", Workshop Terra Cognita 2006 Directions to the Geospatial Semantic Web, ISWC 2006, Athens, Georgia, USA, 2006.

Johannesson, 2007

Johannesson, P. "The Role of Business Models in Enterprise Modelling". In: Krogstie, J.; Opdahl A.L.; Sjaak Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering. Springer-Verlag ISBN 978-3-540-72676-0, pp. 123-140, 2007.

Kanevski (ed.), 2008

Kanevski, M. (ed.). "Advanced Mapping of Environmental Data: geostatistics, machine learning, and Bayesian maximum entropy". ISBN 978-1-84821-060-8. ISTE Ltd and John Wiley & Sons, Inc., 2008.

Karakostas and Zorgios, 2008

Karakostas, B. and Zorgios, Y. "Engineering Service Oriented Systems: A Model Driven Approach". ISBN 978-1-59904-968-7, IGI Global, 2008.

Karle and Oberweis, 2008

Karle, T. and Oberweis, A. "Collaborative Model Driven Software Development for SOA-based Systems". In: Kaschek, R et al (eds.): Information Systems and e-Business Technologies, 2nd UNISCON 2008 Conference, LNBIP 5, Springer-Verlag, pp. 189-200, 2008.

Keitel, Mayer-Föll and Schultze, 2009

Keitel, A., Mayer-Föll, R. and Schultze, A. "Framework Conception for the Environmental Information System of Baden-Württemberg (Germany)". In: Hřebíček, J. et al (eds.) (2009): Proceedings of the European conference Towards eEnvironment, ISBN 978-80-210-4824-9, pp. 461-468, 2009.

Kim and Carrington, 2002

Kim, S.-K. and Carrington, D. "Integrating Use-Case Analysis and Task Analysis for Interactive Systems," Ninth Asia-Pacific Software Engineering Conference (APSEC'02), pp. 12, 2002.

Klarl and Preitsameter, 2006

Klarl, H. and Preitsameter, M. "Securing Service-oriented and Event-Driven Architectures – Results of an Evaluation of Enterprise Security Frameworks". In: IEEE Services Computing Workshops, pp. 89, 2006.

Klien, 2008

Klien, E. "Semantic Annotation of Geographic Information." PhD Thesis, Institute for Geoinformatics, University of Muenster. Muenster, Germany, 2008. http://ifgiweb.uni-muenster.de/~klien/publications/Klien_PhDThesis_full.pdf

Klyne and Caroll (eds.), 2004

Klyne, G. and Caroll, J.J. (eds.). "Resource Description Framework (RDF): Concepts and Abstract Syntax". W3C Recommendation, 2004. http://www.w3.org/TR/rdf-concepts

Klopfer and Kannellopoulos (eds.), 2008

Klopfer, M. and Kannellopoulos, I. (eds.). "ORCHESTRA – An Open Service Architecture for Risk Management", ISBN 978-3-00-024284-7, 2008. http://www.eu-orchestra.org/docs/ORCHESTRA-Book.pdf

Klopfer and Simonis (eds.), 2009

Klopfer, M. and Simonis, I. (eds.) "SANY - an open service architecture for sensor networks". ISBN 978-3-00-028571-4, 2009 http://www.sany-ip.eu/publications/3317

Kohlborn et al, 2009

Kohlborn, T., Korthaus, A., Chan, T. and Rosemann, M. "Service Analysis - A Critical Assessment of the State of the Art". European Conference of Information Systems (ECIS 2009), Verona, Italy, 2009.

Kolas, Hebeler and Dean, 2005

Kolas, D., J. Hebeler and Dean, M. "Geospatial Semantic Web: Architecture of Ontologies". In: Rodriguez et al (eds.) GeoSpatial Semantics, First International Conference GeoS, Mexico City, 2005, LNCS 3799, pp. 183-194, 2005.

Kontogiannis et al, 2007

Kontogiannis, K., Lewis, G.A., Smith, D.B., Litoiu, M., Müller, H., Schuster, S. and Stroulia, E. "The Landscape of Service-Oriented Systems: A Research Perspective". International Workshop on Systems Development in SOA Environments (SDSOA'07). IEEE Computer Society 0-7695-2960-7/07, 2007.

Koskela et al, 2007

Koskela, M., Rahikainen, M. and Wan, T. "Software development methods: SOA vs. CBD, OO and AOP". Seminar on Enterprise Information Systems: Service-Oriented Architecture and Software Engineering. Helsinki University of Technology, 2007.
http://www.soberit.hut.fi/T-86/T-86.5165/2007/final_koskela_rahikainen_wan.pdf

Kreger and Estefan (eds.), 2009

Kreger, H. and Estefan, J. (eds.). "Navigating the SOA Open Standards Landscape Around Architecture". The Open Group, Document No. W096, 2009.
http://www.opengroup.org/projects/soa/uploads/40/20044/W096.pdf

Krogstie et al (eds.), 2007

Krogstie, J., Opdahl A.L. and Brinkkemper, S. (eds.). "Conceptual Modelling in Information Systems Engineering". Springer-Verlag ISBN 978-3-540-72676-0, pp. 229-246, 2007.

Kruchten, 2000

Kruchten, P. "The Rational Unified Process – An Introduction", 2nd edition. Addison-Wesley-Longman, Reading, MA, 2000.

Krutz et al, 2007

Krutz, K., Kuhn, T., Gnad, C. and Abeck., S. "Semantic Service Description to Support Education and Training". 3rd International Conference on Web Information Systems and Technologies (WEBIST), Barcelona, 2007.

Kuhn, 2004

Kuhn, W. "Semantics of What ?" Position paper NCGIA Specialist Meeting on Spatial Webs, Santa Barbara, CA, December 2-4, 2004.

Lake and Farley, 2007

Lake, R. and Farley, J. "Infrastructure for the Geospatial Web". In: Scharl, A. and Tochtermann, K. (eds.): The Geospatial Web. ISBN-13 978-1-84628-826-5, Springer-Verlag, 2007.

Lam and Swayne, 2001

Lam, D. and Swayne, D. "Issues of EIS software design: some lessons learned in the past decade". Elsevier Journal Environmental Modelling and Software 16 (2001), pp. 419-425, 2001.

Landholm (ed.), 1998

Landholm, M. (ed.). "Defining Environmental Security: Implications for the U.S. Army". Army Environmental Policy Institute, AEPI-IFP-1298, 1998.
http://www.aepi.army.mil/internet/defining-env-sec-for-army.pdf

Lausen, Polleres and Roman (eds.), 2005

Lausen, H., Polleres, A. and Roman, D. (eds.). “Web Service Modeling Ontology (WSMO)”. W3C Member Submission, 2005.
http://www.w3.org/Submission/WSMO/

Leymann, 2003

Leymann, F. “Web Services – Distributed Applications without Limits”. Keynote at the Conference „10. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)“, Leipzig, 2003.
http://doesen0.informatik.uni-leipzig.de/proceedings/paper/keynote-leymann.pdf

Lieberman (ed.), 2006

Lieberman, J. (ed.). “Geospatial Semantic Web Interoperability Experiment Report”. OGC Discussion Paper 06-002r1, Version 0.5, 2006.
http://portal.opengeospatial.org/files/?artifact_id=15198

Lieberman, 2007

Lieberman, J. “Geosemantics Working Group”, OGC Project Document 06-179, 2007.

Lucchi, Millot and Elfers, 2008

Lucchi, R., Millot, M. and Elfers, C. “Resource Oriented Architecture and REST - Assessment of impact and advantages on INSPIRE”. JRC Scientific and Technical Reports, EUR 23397 EN, European Commission, Joint Research Centre, 2008.
http://www.ec-gis.org/sdi/publist/pdfs/lucchi-etal2008eur.pdf

Lutz and Klien, 2006

Lutz, M. and Klien, E. “Ontology-based Retrieval of Geographic Information”. International Journal of Geographical Information Science (IJGIS), Volume 20, Issue 3, pg. 233 – 260, 2006.

Lutz, 2007

Lutz, M. “Ontology-based Descriptions for Semantic Discovery and Composition of Geoprocessing Services”, Geoinformatica 11:1-36 DOI 10.2007/s10707-006-7635-9, 2007.

Maier, 1998

Maier, M.W. “Architecting Principles for Systems-of-Systems”. Journal of Systems Engineering, Vol. 1, No. 4, pp. 267-284, 1998.

Manske et al, 2009

Manske, K., Doeweling, S., Probst, F., and Ziegert, T. “SoKNOS - an interactive visual emergency management framework”. In: de Amicis et al (eds.) (2009). NATO Science for Peace Security Series, Springer Science + Business Media B.V., ISBN 978-90-481-2897-6, pp. 251-262, 2009.

Martin et al, 2004a

Martin, D., Paolucci, M., and McIlraith, S. "Bringing Semantics to web services: The owl-s approach". In: Proceedings of the Workshop on Semantic Web Services and Web Process Composition (SWSWPC), San Diego, CA, USA, 2004.

Martin et al, 2004b

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDemott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. and Sycara, K. "OWL-S: Semantic Markup for Web Services". W3C Member Submission, 22 November 2004.
http://www.w3.org/Submission/OWL-S/

March and Smith, 1995

March, S.T. and Smith, G. "Design and Natural Science Research on Information Technology", Decision Support Systems (15:4), pp. 251-266, December 1995

Mayer-Föll et al, 2009

Mayer-Föll, R., Keitel, A. and W. Geiger (eds.). "F+E-Vorhaben KEWA - Kooperative Entwicklung wirtschaftlicher Anwendungen für Umwelt, Verkehr und benachbarte Bereiche in neuen Verwaltungsstrukturen, Phase IV 2008/2009“. Scientific Reports of the Forschungszentrum Karlsruhe FZKA 7500, 2009.
http://www.lubw.baden-wuerttemberg.de/servlet/is/54863/

Maué (ed.), 2009

Maué, P. (ed.). "Semantic Annotations in OGC Standards". Version 0.3.0 OGC Discussion paper 08-167r1, 2009.
http://portal.opengeospatial.org/files/?artifact_id=34916

McGuinness, 2003

McGuinness, D.L. "Ontologies come of age". In: Fensel, D., Hendler, J., Lieberman, H. and Wahlster, W. (eds.): Spinning the Semantic Web: Bringing the Wolrd Wide Web to Its Full Potential. MIT Press, Cambrdige, MA, 2003.

Mowbray and Ruh, 1997

Mowbray, T.J. and Ruh, W.A. "Inside CORBA. Distributed Object Standards and Applications". ISBN-13 978-0-20189-540-7. Addison-Wesley Object Technology, 1997.

Na and Priest, 2007

Na, A., and Priest, M. (eds.). "Sensor Observation Service Version 1.0". OpenGIS® Implementation Standard, OGC 06-009r6, 2007.
http://www.opengeospatial.org/standards/sos

NATO, 2008

NATO. "Integration of Information for Environmental Security". In: NATO Security through Science Series, DOI 10.1007/978-1-4020-6575-0, ISBN 978-1-4020-6574-3, 2008.

National Research Council, 2007

US National Research Council. "Successful Response Starts with a Map – Improving Geospatial Support for Disaster Management". Committee on Planning for Catastrophe. The National Academies Press. ISBN-10 0-309-10340-1, 2007.

Nebert, Whiteside and Vretanos (eds.), 2007

Nebert, D., Whiteside, A. and Vretanos, P. (eds.) "OpenGIS Catalogue Services Specification, Version 2.0.2, Corrigendum 2 Release". OpenGIS® Implementation Specification, OGC 07-006r1, 2007
http://www.opengeospatial.org/standards/cat

Noran, 2003

Noran, O. "UML vs. IDEF: An Ontology-oriented Comparative Study in View of Business Modelling". In: Seruca et al. (eds.): Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004), Vol. 3. pp. 674-682. Porto, Portugal, 2003.

Noy, 2004

Noy, N. "Semantic Integration: A survey of ontology-based approaches", SIGMOD Record, Vol. 33, no. 4, pp. 65-70, 2004.

Nuseibeh, 2001

Nuseibeh, B. "Weaving Together Requirements and Architecture", IEEE Computer, 34(3):115-117, March 2001.

OASIS, 2003

OASIS. "ebXML Registry Information Model (ebRIM) v2.5", Committee Approved Specification, 2003.
http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrim-2.5.pdf

OASIS, 2004

OASIS. "Universal Description Discovery & Integration (UDDI) Version 3.0.2". UDDI Spec Technical Committee Draft, 2004.
http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

OASIS, 2006

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

OASIS, 2007

OASIS. “Web Services Business Process Execution Language Version 2.0”. OASIS Standard, 2007.
http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

OASIS, 2008

Norton, B., Kerrigan, M., Mocan, A., Carenini, A., Cimpian, E., Haines, M., Scicluna, J., and Zaremba, M. (eds.). “OASIS Reference Ontology for Semantic Service Oriented Architectures Version 1.0”. 2008.
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/see-rosoa-v1.0.doc

OASIS, 2009

Kerrigan, M., Norton, B. “OASIS Reference Architecture for Semantic Execution Environments”. Initial Draft 13 January 2009.
http://www.oasis-open.org/committees/download.php/30675/Semantic%20
Execution%20Environment%20Reference%20Architecture_20090113_BN.doc

Offermann and Bub, 2009

Offermann, P. and Bub, U. “Empirial Comparison of Methods for Information Systems Development according to SOA”. European Conference of Information Systems (ECIS 2009), Verona, Italy, 2009.

OGC-Glossary, 2009

Open Geospatial Consortium (OGC). “Glossary of Terms”. Living Document at the OGC web site, last accessed on 30/10/2009.
http://www.opengeospatial.org/ogc/glossary

OGC, 1999

Open Geospatial Consortium (OGC). “Abstract Specification - Topic 13 Catalog Services”. OGC Doc. No. 99-133, 1999.

OGC, 2002

Open Geospatial Consortium (OGC). “Abstract Specification - Topic 12 - The OpenGIS® Service Architecture”. OGC Doc. No. 02–112, 2002.

OGC, 2005

Open Geospatial Consortium (OGC). “OpenGIS® Web Service Common Implementation Specification”. OGC Doc. No. 05-008c1, 2005.

OGC, 2006

Open Geospatial Consortium (OGC). “Web Map Service (WMS) Implementation Specification V1.3.0”. OGC Doc. No. 06-042, 2006.

OGC-Press, 2009

Open Geospatial Consortium (OGC). “OGC Forms Hydrology and Meteorology Domain Working Groups”. OGC Press release, 2009.
http://www.opengeospatial.org/pressroom/pressreleases/1001

OMG, 1997

Object Management Group (OMG), Soley, M. (ed.). "Object Management Architecture Guide, Revision 3.0". OMG Document ab/97-05-05, 1997.
http://doc.omg.org/ab/97-05-05

OMG, 2002

Object Management Group (OMG). "Meta-Object Facility Version 2", OMG Document formal/02-04-03, 2002.

OMG, 2003

Object Management Group (OMG), Mukerji, I. and Miller, J. (eds.). "MDA Guide Version 1.0.1". OMG Document omg/03-06-01, 2003.
http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

OMG, 2006

Object Management Group (OMG). "UML Profile and Metamodel for Services". OMG Document soa/06-09-09, 2006.

OMG, 2007

Object Management Group (OMG). "Unified Modeling Language: Superstructure, version 2.1.1". OMG Document formal/07-02-05, 2007.

OMG, 2008a

Object Management Group. "Business Motivation Model, Version 1.0". OMG Document formal/08-08-02, 2008.
http://www.omg.org/spec/BMM/1.0/PDF

OMG, 2008b

Object Management Group (OMG). "Business Process Definition Meta Model Volume II: Process Definitions, Version 1.0". OMG Document formal/08-11-04, 2008.
http://www.omg.org/spec/BPDM/1.0

OMG, 2008c

Object Management Group (OMG). "Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS)". OMG Document ad/08-11-01, 2008.

OMG, 2008d

Object Management Group (OMG). "Software and Systems Process Engineering Meta-Model Specification", Version 2.0. OMG Document formal/08-04-01, 2008.

OpenGroup, 2008

The Open Group. "Service-Oriented Architecture Ontology". Version 2.0, Draft Technical Standard, 2008.
http://www.opengroup.org/projects/soa-ontology

Papazoglou and Georgakopoulos, 2004

Papazoglou, M.P. and Georgakopoulos, D. "Service-oriented computing". Communications of the ACM 46(10), pp. 24-28, 2004.

Papazoglou et al, 2007

Papazoglou, M.P., Traverso, P., Dustdar, S. and Leymann, F. "Service-Oriented Computing: State of the Art and Research Challenges". IEEE Computer Society 0018-9162/07, 2007.

Pautasso et al, 2008

Pautasso, C., Zimmermann, O. and Leymann, F. "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision". WWW Conference 2008, Beijing, China, ACM 978-1-60558-085-2/08/04, 2008.

Percivall (ed.), 2003

Percivall, G. (ed.). "OGC Reference Model Version 0.1.3", Open Geospatial Consortium Document 03-040, 2003.
http://portal.opengeospatial.org/files/?artifact_id=3836&version=3

Percivall (ed.), 2008

Percivall, G. (ed.). "OGC Reference Model Version 2.0", Open Geospatial Consortium Document 08-062r4, 2008.
http://orm.opengeospatial.org/

Pohl, 2008

Pohl, K. "Requirements Engineering – Grundlagen, Prinzipien, Techniken". dpunkt.verlag GmbH, 2nd corrected edition, ISBN 978-3-89864-550-8, 2008.

Pohl and Sikora, 2007

Pohl, K. and Sikora, E. "The Co-Development of System Requirements and Functional Architecture". In: Krogstie et al (eds.), (2007): Conceptual Modelling in Information Systems Engineering, pp. 229-246, 2007.

Powell, 1991

Powell, D. (ed.). "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.

Preist, 2004

Preist, C. "A conceptual architecture for semantic web services". In: McIlraith, S.A., Plexousakis, D. and van Harmelen, F. (eds.): The Semantic Web – ISWC 2004, LNCS volume 3298. Springer-Verlag Berlin Heidelberg, pp. 395-409, 2004.

Ramm and Topf, 2009

Ramm, F. and Topf, J. "OpenStreetMap – Die freie Weltkarte nutzen und mitgestalten." Lehmanns Media, 2nd Edition 2009, ISBN 978-3-86541-320-8, 2009.

Raskin and Pan, 2005

Raskin, R. and Pan, M.J. ”Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)”. In: Babaie, H.A., Ramachandran, R. (Eds.), Special Issue Applications of XML in Geosciences, Computers & Geosciences 31, pp. 1119-1125, 2005.

Reed, 2009

Reed, C. “The OGC and REST – a position statement”. OGC NetworkTM, 2009.
http://portal.opengeospatial.org/files/?artifact_id=32222

Reichardt, 2003

Reichardt, M. “The Havoc of Non-Interoperability. An Open GIS Consortium (OGC) White Paper“. Open Geospatial Consortium (OGC), 2003.
http://portal.opengeospatial.org/files/?artifact_id=5097&version=3&format=pdf

Reussner and Hasselbring (eds.), 2009

Reussner, R. and Hasselbring, W. (eds.). “Handbuch der Software-Architektur”. 2nd Edition. ISBN 978-3-89864-559-1, dPunkt.verlag GmbH, 2009

Richardson and Ruby, 2007

Richardson, L. and Ruby, S. “RESTful Web Services”. O’Reilly Media, Inc. ISBN-10: 0-596-52926-0, 2007.

Ricken, 2007

Ricken, J. “Top-down Modeling Methodology for Model-driven SOA Construction”. In R. Meersman, Z. Tari, P. Herrero et al. (eds.): On the Move to Meaningful Internet Systems OTM 2007 Workshops, Part I, LNCS 4805, pp. 323–332, ISBN 978-3-540-76887-6, Springer-Verlag, 2007.

Ricken and Petit, 2008

Ricken, J., Petit, M. “Characterization of Methods for Process-Oriented Engineering of SOA”. In: Ardagna, D., Mecella, M. and Yang, J. (eds.): Business Process Management Workshops (BPM 2008) International Workshops, Springer Berlin Heidelberg, ISBN 978-3-642-00327-1, pp. 621-632, 2008.

Rodriguez et al, 2005

Rodriguez M.A., Cruz, I.F., Egenhofer, M.J. and Levashkin, S. (eds.). “GeoSpatial Semantics“, First International Conference GeoS, Mexico City, 2005, LNCS 3799, 2005.

Rumbaugh et al, 1998

Rumbaugh, J., Jacobson, I. and Booch, G. “The Unified Modeling Language Reference Manual”. Addison Wesley Publ. Co., Reading, Massachussets, 1998.

Sawyer, 2005

Sawyer, P. (ed.) "State of the Art - Service Engineering (Specification)", Deliverable A1.D1a of the EU Integrated Project SeCSE, 2005. http://www.secse-project.eu/wp-content/uploads/2007/08/a1d1-state-of-the-art-service-engineering.zip

Schimak (ed.), 2005

Schimak, G. (ed.). "Environmental Knowledge and Information Systems". Elsevier Special Issue, Vol. 20, Number 12, 2005.

Schönbein, 2006

Schönbein, R. "Agenten- und Ontologiebasierte Software-Architektur zur Bildauswertung". PhD thesis, University of Karlsruhe (TH), Faculty of Informatics, Universitätsverlag Karlsruhe, ISBN 3-937300-98-8, 2006.

Shamsfard, Nematzadeh and Motiee, 2006

Shamsfard, M., Nematzadeh, A. and Motiee, S. "ORank: An Ontology Based System for Ranking Documents". International Journal of Computer Science, Volume 1, No 3. ISSN 1306-4428, 2006.

Sriharee, 2006

Sriharee, N. "Semantic Web Services Discovery Using Ontology-based Rating Model". Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings) (WI'06), 2006.

Simon, 1996

Simon, H.A. "The Sciences of the Artificial (3rd ed.)". MIT Press, Cambridge, MA, 1996.

Simonis (ed.), 2007

Simonis, I. (ed.). "OpenGIS® Sensor Planning Service Implementation Specification". OGC® Standard 07-014r3, 2007. http://www.opengeospatial.org/standards/sps

Simonis (ed.), 2008

Simonis, I. (ed.). "OGC® Sensor Web Enablement Architecture". OGC Best Practices Document 06-021r4, Version 0.4.0, 2008. http://www.opengeospatial.org/standards/bp

Simons, 2002

Simons, A. "The theory of classification. part 4: Object types and subtyping", Journal of Object Technology, 1(5): 27-35, 2002.

Skouloudis (ed.), 2009

Skouloudis, A. N. (ed.). "Connectivity between Environment and Health Information Systems". SMART 2006/0054 Final Study Report, 2009 (publication announced)

Sommerville, 2007

Sommerville, I. “Software Engineering”. Addison-Wesley Publishers Limited, 8th Edition, 2007.

Sowa, 2007

Sowa, J.F. “Mathematical Background”. Revised and extended version of Appendix A from Sowa, J.F.: Conceptual Structures. 2007.
http://www.jfsowa.com/logic/math.htm

Stock, 2009

Stock, K. “OGC^{TM} Catalogue Services – OWL Application Profile of CSW”. Version 1.0.0. OGC Document 09-010, 2009.

Stuckenschmidt, 2009

Stuckenschmidt, H. “Ontologien. Konzepte, Technologien und Anwendungen”. Informatik im Fokus. ISBN 978-3-540-79330-4, Springer Verlag, 2009.

Sycara et al, 2002

Sycara, K., Widoff, S., Klusch, M. and Lu, J. “Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace”. In First International Joint Conference on Autonomous Agents and Multi-Agent Systems, pg. 173–203, Bologna, Italy, 2002.

Tan et al, 2009

Tan, W., Missier, P., Madduri, R. and Foster, I. “Building Scientific Workflow with Taverna and BPEL: a Comparative Study in caGrid”. In: Feuerlicht, G. and Lamersdorf, W. (Eds.): ICSOC 2008, Springer-Verlag LNCS 5472, pp. 118-129, 2009..

Thieken (ed.), 2005

Thieken, A. (ed.). “CEDIM Glossary - Terms and definitions of risk sciences”. Center for Disaster Management and Risk Reduction Technology (CEDIM), located at the University of Karlsruhe (TH) and the Geoforschungszentrum (GFZ) in Potsdam, 2005.
http://www.cedim.de/english/905.php

Timmerman and Langaas (eds.), 2004

Timmerman, J.G. and Langaas, S. (eds.) “Environmental Information in European Transboundary Water Management“. IWA Publishing, London, UK, ISBN 1-84339-038-8, 2004.

Toch et al, 2008

Toch, E., Gal, A., Reinhartz-Berger, I. and Dori, D. “A Semantic Approach to Approximate Service Retrieval”. In ACM Transactions on Internet Technology. Special Issue on Semantic Web Services. Volume 8, Number 1, 2008.

Tran et al, 2007

Tran, T., Cimiano, P., Rudolph, S. and Studer, R. "Ontology-based Interpretation of Keywords for Semantic Search". Proceedings of the 6th International Semantic Web Conference (ISWC'07), pp. 523-536. Busan, Korea, 2007.

Trastour et al, 2001

Trastour, D., Bartolini, C. and Gonzalez-Castillo, J. "A semantic web approach to service description for matchmaking of services". In: Cruz, I., Decker, S., Euzenat, J. and McGuinness, D. (eds.): The first Semantic WebWorking Symposium (SWWS), pg. 447–461, Stanford University, CA, USA, 2001.

Usländer et al, 2003

Usländer, T., Schmid, H., Schmieder, M. and Stumpp, J. "Thematic User Environment for Water Body Information Systems and beyond". In: Proceedings of the 17th International Conference on Informatics for Environmental Protection (EnviroInfo 2003), Cottbus, Germany, 2003.

Usländer, 2005

Usländer, T. "Trends of environmental information systems in the context of the European Water Framework Directive". Elsevier Journal Environmental Modelling & Software 20, pg. 1532-1542, 2005.

Usländer (ed.), 2007

Usländer, T. (ed.). "Reference Model for the ORCHESTRA Architecture Version 2.1". OGC Best Practices Document 07-097, 2007.
http://portal.opengeospatial.org/files/?artifact_id=23286

Usländer, 2008a

Usländer, T. "The Growing Importance of Open Service Platforms for the Design of Environmental Information Systems". Proceedings of the International Congress on Environmental Modelling and Software (iEMSs 2008) Volume 3, (eds. Sànchez-Marrè et al). pp. 1628-1635, ISBN: 978-84-7653-074-0, 2008.
http://www.iemss.org/iemss2008/index.php?n=Main.Proceedings

Usländer, 2008b

Usländer, T. "Integration of Resource-Oriented Architecture Concepts into the OGC Reference Model", Version 0.0.2, OGC Document 07-156r1, 2008.

Usländer, 2009a

Usländer, T. "Architectural Viewpoints and Trends for the Implementation of the Environmental Information Space". In: Hřebíček, J. et al (eds.) (2009): Proceedings of the European conference Towards eEnvironment, ISBN 978-80-210-4824-9, pp. 130-137, 2009.

Usländer (ed.), 2009b

Usländer, T. (ed.). “Specification of the Sensor Service Architecture, Version 3.0 (Rev. 3.1)”. OGC Discussion Paper 09-132r1. Deliverable D2.3.4 of the European Integrated Project SANY, FP6-IST-033564, 2009. http://portal.opengeospatial.org/files/?artifact_id=35888

Usländer, 2009c

Usländer, T. “Towards Interoperable Environmental Security Applications – The Role of Open Geospatial Service Platforms”. In: de Amicis et al (eds.) (2009): GeoSpatial Visual Analytics: Geographical Information Processing and Visual Analytics for Environmental Security. NATO Science for Peace Security Series, Springer Science + Business Media B.V., ISBN 978-90-481-2897-6, pp. 15-27, 2009.

Usländer and Denzer, 2009

Usländer, T., Denzer, R. “Requirements and Open Architecture for Environmental Risk Management Information Systems”, Chapter 15 of Van de Walle, B., Turoff, M. and Hiltz, S.R. (eds.): Information Systems for Emergency Management. In the Advances in Management Information Systems monograph series (Editor-in-Chief: Vladimir Zwass). Armonk, NY: M.E. Sharpe Inc., ISBN 978-0-7656-2134-4, 2009.

van den Heuvel et al, 2009

van den Heuvel W. J., Zimmermann O., Leymann F., Lago P., Schieferdecker I., Zdun U. and Avgeriou P. “Software Service Engineering: Tenets and Challenges”. In: Proceedings of ICSE 2009 Workshop - Principles of Engineering Service Oriented Systems (PESOS), IEEE Computer Society, May 2009.

van Oosterom et al (eds.), 2005

van Oosterom, P., Zlatanova, S. and Fendek, E.M. (eds.) “Geo-information for Disaster Management”. ISBN 3-540-24998-5 Springer-Verlag Berlin Heidelberg, 2005.

Visser, 2004

Visser, U. “Intelligent Information Integration for the Semantic Web”, Lecture Notes in Artificial Intelligence 3159, ISBN 3-540-22993-0, Springer-Verlag Berlin Heidelberg, 2004.

Vogt (ed.), 2002

Vogt, J. (ed). “Guidance Document on Implementing the GIS Elements of the WFD”. Report of the Working Group GIS as part of the Water Framework Directive (WFD) Common Implementation Strategy. Joint Research Centre, European Commission, 166 pp. Office for Official Publications of the European Communities, ISBN: 92-894-5129, 2002.

Wahlster and Raffler, 2008

Wahlster, W. and Raffler, H. “Forschen für die Internet-Gesellschaft: Trends, Technologien, Anwendungen“. Trends und Handlungsempfehlungen 2008 des Feldafinger Kreises, 2008.
http://www.feldafinger-kreis.de

Williams (ed.), 2008

Williams, J. (ed.). “SP5 Requirements Specification Refinement, Version 1.3”. Deliverable D5.1.1.2, European Integrated Project SANY, IST-FP6 033564, 2008.
http://www.sany-ip.eu/publications/3161

W3C, 2004a

World Wide Web Consortium (W3C). “The Web Services Architecture”. 2004.
http://www.w3.org/TR/2004/NOTE-ws-arch-20040211

W3C, 2004b

World Wide Web Consortium (W3C). “Web Services Glossary”. 2004.
http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

W3C, 2007a

World Wide Web Consortium (W3C). “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”. W3C Candidate Recommendation, 2007.
http://www.w3.org/TR/2007/REC-wsdl20-20070626

W3C, 2007b

World Wide Web Consortium (W3C). “SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)”. W3C Recommendation, 2007.
http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

W3C-SWBPD, 2006

W3C Semantic Web Best Practices and Deployment Working Group. “A Semantic Web Primer for Object-Oriented Software Developers”, 2006.
http://www.w3.org/TR/2006/NOTE-sw-oosd-primer-20060309

WRON, 2006

WRON Alliance and National Water Commission of the Australian Government, “System Architecture for the Australian Water Resources Information System (AWRIS)”. Final report, 2006.
http://water.gov.au/Publications/AWRIS_SystemArchitecture_Dec06.pdf

Zachman, 1987

Zachman, J.A. “A framework for information systems architecture”. IBM Systems Journal: Volume 26, Number 3, pp. 276-292, 1987.

Zachos et al, 2007

Zachos, K., Maiden, N., Zhu, X. and Jones, S. "Discovering Web Services to Specify More Complete System Requirements". In: Proceedings of CAiSE 2007, LNCS 4495, pp. 142-157, Springer-Verlag Berlin Heidelberg, 2007.

Zhang et al, 2007

Zhang, L.-J., Cheng, S., Chee, Y.-M., Allam, A. and Zhou, Q. "Pattern Recognition based Adaptive Categorization Technique and Solution for Services Selection". IEEE Asia-Pacific Services Computing Conference, pp. 535-543, 2007.

Zhang et al, 2008

Zhang, L.-J., Zhou, N., Chee, Y.-M., Jalaldeen, A., Ponnalagu, K., Sindhgatta, R.R., Arsanjani, A. and Bernardini, F. "SOMA-ME: A platform for the model-driven design of SOA solutions". IBM Systems Journal, Vol. 47, No. 3, pp. 397-413, 2008.

Zhuge, 2008

Zhuge, H. "The Web Resource Space Model". Web Information Systems Engineering and Internet Technologies Book Series. Springer Science+Business Media, LLC, ISBN-13 978-0-38772-771-4, 2008.

Zimmermann et al, 2004

Zimmermann, O., Krogdahl, P. and Gee.C. "Elements of service oriented analysis and design. An interdisciplinary modelling approach for SOA projects", IBM developers works, 2004.
http://www-128.ibm.com/developerworks/webservices/library/ws-soad1

10.2 About the Author

Name	Thomas Usländer
Date of Birth	30 July 1961

Education

1972-1981	Kepler-Gymnasium Pforzheim, Germany, Abitur 1981
1981-1987	Study of informatics at the University of Karlsruhe (TH), Germany
	Diploma degree in informatics (Dipl.-Inform.), 1987

Professional Career

1987-1992	Research Scientist at Fraunhofer IITB, Karlsruhe, Germany
	Development of ISO/OSI network management systems
1992-1993	Leader of the development team "Network Management" at Conware Computer Consulting GmbH, Karlsruhe, Germany
1993-1994	Research Scientist at Fraunhofer IITB, Karlsruhe, Germany
	Design of Distributed Systems Architectures based upon CORBA
since 1995	Manager of the Research Group “Integrated Thematic Information Systems“, Fraunhofer IITB (since 1 January 2010 renamed to Fraunhofer IOSB), Karlsruhe, Germany

- since 1998 Project Manager for the development of Environmental Information Systems in the German federal states of Baden-Wuerttemberg, Thuringia and Bavaria
- 2004 Certificate “Management of Water Resources”, Bauhaus-University Weimar
- 2006-2009 Leader of the work package “Architecture Design” of the European Integrated Project SANY (Sensors Anywhere), FP6-IST-033564
- 2004-2008 Leader of the work package “Architecture Design” of the European Integrated Project ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management), FP6-IST-511678
- 2008-2009 Co-chair of the “services” sub-group of the OGC Architecture working group

Invited Expert of the European Commission

- Experts Consultation Workshop ”Towards a Single Information Space in Europe for the Environment (SISE)”, organised by the ICT for Sustainable Growth Unit of DG-INFSO, Brussels, 15 February 2008
- Final Workshop of “Connectivity between Environment and Health Information Systems (CEHIS): Supporting synergy between environment and health research and policies.”, on behalf of DG-INFSO (SMART 2006/0054), Brussels, 22-23 September 2008

Reviewer

- Elsevier Journals “Environmental Management” and “Environmental Modelling and Software”
- European Conference on Information Systems (ECIS), Verona, Italy, 2009
- International Congress on Modelling and Simulation (MODSIM 2009), Cairns, Australia, 2009
- International Congress on Environmental Modelling and Software (iEMSs), Barcelona, Spain, 2008
- Research Programme “Information Society”, German federal state Saxony-Anhalt, 2003-2004

Awards

- OMG Application Award 2000 in the category “Best Application utilizing reusable components leveraged from of for use in other projects”
- Best-paper award of the European conference “Towards eEnvironment”, Prague (Usländer, 2009a)

Invitations to Doctoral Consortia

- Knowledge Web PhD Symposium 2007, co-located with the 4th Annual European Semantic Web Conference, Innsbruck, Austria.
- Doctoral Consortium associated to the European Conference on Information Systems (ECIS), Galway, Ireland, 2008

11 Abbreviations, Glossary and Index

11.1 Abbreviations

AC	Acquisition Domain (of the Sensor Service Architecture)
AD	Architecture Document
AOP	Aspect-oriented Programming
AP	Application Domain (of the Sensor Service Architecture)
AWRIS	Australian Water Resources Information System
BPDM	Business Process Definition Metamodel
BPEL	Business Process Execution Language
BPM	Business Process Modelling
BPMN	Business Process Model Notation
CBD	Component-based Development
CEDIM	Center for Disaster Management and Risk Reduction Technology
CEHIS	Connectivity between Environment and Health Information Systems
CEN	Comité Européen de Normalisation (European Committee for Standardization)
CIM	Computation-Independent Model
CORBA	Common Object Request Broker Architecture
COSMOD(-RE)	sCenario and gOal based SysteM development methOD – Requirements Engineering
CSE	Cascaded Services Exploration
CSW	Catalogue Service (for the Web)
DA.x	Design Artefact number x
DEWS	Distant Early Warning System (for Tsunami)
DG-INFSO	General Directorate Information Society of the European Commission
EAI	Enterprise Application Integration
EC	European Commission
EHIS	Environment and Health Information System
EIS	Environmental Information System
EMIS	Environmental Management Information System
FAS	Feature Access Service
FOL	First-order predicate logics
FP6	6th European Research Framework Programme
GEOSS	Global Earth Observation System of Systems
GFM	General Feature Model
GIS	Geographic Information System
GMES	Global Monitoring for Environment and Security
GML	Geography Markup Language
H/H/P	Hatley, Hruschka and Pirbhai design method
HTTP	Hypertext Transfer Protocol

IANA	Internet Assigned Numbers Authority
IDEF	Integrated Definition Methods
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineeeing Task Force
INSPIRE	Infrastructure for Spatial Information in the European Community
ICT	Information and Communication Technology
IOPE	Input, Output, Precondition and Effect
IS	Information System
ISM	Implementation-Specific Model
ISO	International Organization for Standardization
IST	Information Society Technologies
IT	Information Technology
MD	Map and Diagram Service
MDA	Model-Driven Architecture
MIME	Multipurpose Internet Mail Extensions
MP	Mediation and Processing Domain (of the Sensor Service Architecture)
NASA	National Aeronautics and Space Administration
NFR	Non-functional requirement
NOx	nitrogen oxide
NWIS	National Water Information System (of the United States)
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OMG	Object Management Group
OOAD	Object-oriented Analysis and Design
ORCHESTRA	Open Architecture and Spatial Data Infrastructure for Risk Management (European FP-6 Integrated Project in the IST domain)
OWL	Web Ontology Language
OWL-S	Web service ontology based on OWL
O&M	Observations and Measurements
PIM	Platform-Independent Model
PS	Processing Service
PSM	Platform-Specific Model
R.x	Requirement number x
RBMP	River Basin Management Plans
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RM	Reference Model
RM-OA	Reference Model for the ORCHESTRA Architecture
RM-ODP	Reference Model for Open Distributed Processing

SANY	Sensors Anywhere (European FP-6 Integrated Project in the IST domain)
SAWSDL	Semantic Annotations for WSDL and XML Schema
SDI	Spatial Data Infrastructure
SeCSE	Service Centric System Engineering
SEIS	Shared Environmental Information System
SemCat	Semantic Catalogue
SensorSA	Sensor Service Architecture
SERVUS	Design Methodology for Information Systems based upon Geospatial Service-oriented Architectures and the Modelling of Use Cases and Capabilities as Resources
SISE	Single Information Space for the Environment in Europe
SOA	Service-oriented Architecture
SoaML	Service-oriented Architecture Modelling Language
SOAP	Simple Object Access Protocol[38]
SOAD	Service-oriented Analysis and Design
SOA-RM	(OASIS) Reference Model for Service-oriented Architectures
SoKNOS	Service-Oriented Architectures Supporting Networks of Public Security
SOMA (-ME)	Service-Oriented Modelling and Architecture (modelling environment)
SPEM	Software and Systems Process Engineering Meta-model
SoS	System-of-Systems
SOS	Sensor Observation Service
SPS	Sensor Planning Service
SR	Semantic Resource
SRN	Semantic Resource Network
SSE	Software Service Engineering
SSOA	Semantic Service Oriented Architectures
SSOA-RO	Reference Ontology for Semantic Service Oriented Architectures
SWBPD	Semantic Web Best Practices and Deployment Working Group
SWE	Sensor Web Enablement
SWEET	Semantic Web for Earth and Environmental Terminology
TC	Technical Committee
UDDI	Universal Description, Discovery and Integration
UML	Uniform Modelling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WISE	Water Information System for Europe

[38] Original meaning of the acronym SOAP, however, its use has been deprecated.

WFD	Water Framework Directive
WRON	Water Resources Operation Network Australia
WSDL	Web Service Description Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Execution Environment
WSN	Web Services Notification
XML	Extensible Mark-up Language
XSLT	eXtensible Stylesheet Language Transformation

11.2 Glossary

Note: If not indicated otherwise, the glossary terms are originally defined in this thesis or adopted from the Reference Model for the ORCHESTRA Architecture (RM-OA) (Usländer (ed.), 2007).

Application

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 (derived from OGC Glossary (2009)).

Application Domain

Integrated set of problems, terms, information and tasks of a specific thematic domain that an application has to cope with.

Application Schema

Conceptual schema for data required by one or more applications (ISO 19109:2005).

Application Use Case

Use case that deals with application concerns and reflects the functional requirements and the informational requirements (derived from Jacobson and Ng (2005)).

Architecture

(1) Set of rules to define the structure of a system and the interrelationships between its parts (ISO/IEC 10746-2:1996).
(2) Fundamental organization of a system embodied in its components, their relationship to each other and the environment, and the principles guiding its design and evolution (IEEE 1471-2000).

Architectural style

(1) Coordinated set of architectural constraints that restricts the roles/characteristics of architectural elements and the allowed relationships among those elements within an architecture that conforms to that style (derived from Fielding (2000)).
(2) Combination of distinctive features in which architecture is performed and expressed (OpenGroup, 2008).

Candidate Capability

Possible capability which may fulfil the requirements.

Capability

Description of service types or service instances including their functional, informational and qualitative aspects.

Catalogue
(1) Set of service interfaces which support organization, discovery, and access of geospatial information.
(2) Specialized database of information about geospatial resources available to a group or community of users. These resources are assumed to have OpenGIS feature, feature collection, catalog and metadata interfaces, or they may be geoprocessing services.
(OGC, 1999)

Component
Hardware component (device) or Software Component.

Concept
Element of a semantic model.

Conceptual model
Model that defines concepts of a universe of discourse (ISO 19109:2005; ISO 19101:2004).

Conceptual schema
Formal description of a conceptual model (ISO 19109:2005; ISO 19101:2004).

Design
(1) Process of defining the architecture, components, interfaces, and other characteristics of a system or component.
(2) Result of the process in (1).
(IEEE 610.12, 1990)

(Design) Activity
(1) Step in a design process delivering a defined set of design artefacts.
(2) Activity that creates or modifies a design (OpenGroup, 2008).

Design Process
Sequence of expert activities that produces an innovative product (i.e., the design artefact) (Hevner et al, 2004).

(Design) Artefact
Product resulting from a design process.

(Design) Methodology
Recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system, underpinned by the set of beliefs and assumptions that explains why it functions as it does (Avison and Fitzgerald, 2003).

Design Model (of SERVUS)

Conceptual model that comprises requirements and capabilities such that a mapping between both is facilitated. The result of the mapping is documented in the design model, too.

(Design) Problem

Indicates the differences between a goal state and the current state of a system (Hevner et al, 2004).

Discovery

Act of locating a machine-processable description of a resource that may have been previously unknown and that meets certain functional, informational or qualitative criteria. It involves matching a set of functional and other criteria with a set of resource descriptions (derived from W3C (2004b)).

Domain Model (of SERVUS)

Represents the thematic domain to which the problem belongs to. It formally defines the part of the world that comprises the universe of discourse between the user and the system designer, i.e. it comprises the shared knowledge about the application domain.

Environment

(1) (noun) the surroundings or conditions in which a person, animal, or plant lives or operates.
(2) (the environment) the natural world, especially as affected by human activity.
(3) (computing) Overall structure within which a user, computer, or program operates.
(Oxford Dictionary)

Environmental Information System

Geographic Information System that provides information about the past, current and future status of environment phenomena.

Feature

Abstraction of a real world phenomenon perceived in the context of an application (derived from ISO 19101:2004).
Note: As in (Usländer (ed.), 2007), our understanding of a "real world" explicitly comprises hypothetical worlds. Features may but need not contain geospatial properties. In this general sense, a feature corresponds to an "object" in analysis and design models.

Functional Requirement

Requirement that describes the functions and the processes that a system has to support.

Generic (Service, Infrastructure…)

Independent on the organisation structure and application domain, etc. For example, 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).

Geographic Information System

Application used to discover, access, manipulate, store, process and view geospatial information.

Note: The term Geographic Information System (GIS) is more popular than the term Geospatial Information System. However, in many contexts, "geospatial" is more precise than "geographic," because geospatial information is often used in ways that do not involve a graphic representation, or map, of the information (OGC Glossary, 2009).

Geospatial

Referring to a location relative to the Earth's surface.

Informational Requirement

Requirement that describes the major terms, concepts of the application domain and information elements the system has to deal with.

Infrastructure Use Case

Use case that deals with infrastructure concerns and reflects the non-functional requirements.

Interface

Named set of operations that characterize the behaviour of an entity. The aggregation of operations in an interface, and the definition of interface, shall be for the purpose of software reusability. The specification of an interface shall include a static portion that includes definition of the operations. The specification of an interface shall include a dynamic portion that includes any restrictions on the order of invoking the operations (ISO 19119:2005).

Loose coupling

Describes the configuration in which artificial dependency, i.e. the set of factors that a system has to comply with in order to consume the features or services provided by other systems, has been reduced to the minimum.
(derived from W3C, 2004b)

Matching

Finding correspondences between semantically related entities in different ontologies

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

Meta-information Model
Conceptual model for meta-information.

Multi-style Service-Oriented Architecture
Service-oriented Architecture in which the service-oriented architectural style coexists with other architectural styles.

Non-functional Requirement
Qualitative requirement or side-condition.

Observation
Act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property (Cox (ed.), 2007).

Observed Property
Identifier or description of the phenomenon for which the observation result provides an estimate of its value (Cox (ed.), 2007).

Offered Resource
Abstraction of capabilities according to the resource model.

Ontology
Conceptual model that defines concepts and their relations, together with constraints on those objects and relations (Alexiev et al, 2005). An ontology represents shared knowledge, i.e. it represents a common understanding (consensus) of the universe of discourse between the parties involved.

Open Architecture
Architecture whose specifications are published and made freely available to interested vendors and users with a view of widespread adoption of the architecture. An open architecture makes use of existing standards where appropriate and possible and otherwise contributes to the evolution of relevant new standards (Powell, 1991).

Open Geospatial Service Platform
Service platform that enables the access, management and processing of geospatial information. "Open" means that the specifications of the service interfaces are published, made freely available to interested vendors and users with a view of widespread adoption, and rely upon existing standards where appropriate and possible.

Operation

Specification of a transformation or query that an object may be called to execute. An operation has a name and a list of parameters (ISO 19119:2005).

Phenomenon

Characteristic of one or more feature types, the value for which may be estimated by application of some procedure in an observation (Cox (ed.), 2007).

Probabilistic Semantic Resource

Semantic resource that additionally includes a value that signifies the certainty of the concept mapping, i.e. the belief of the system designer on whether the resource or the properties belongs to the concepts of the semantic model.

Publishing

Design activity in which the capabilities of a platform are entered into the capability model of the design model.

Qualitative Requirement

Requirement that describes qualitative constraints upon the behaviour of the system, dealing, for instance, with dependability, performance and security aspects.

Reference Model

Abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment (OASIS, 2006).

Rephrasing

Design activity in which the requirements of a user are transformed and entered into the requirements model of the design model.

Representation (of a resource)

Comprises any useful information about the current state of a resource (Richardson and Ruby, 2007).

Requirement

(1) Condition or capability needed by a user to solve a problem or achieve an objective.
(2) Condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) Documented representation of a condition or capability as in (1) or (2).
(IEEE 610.12, 1990)

Resource

Anything that's important enough to be referenced as a thing itself. A resource has a unique identification. (Richardson and Ruby, 2007)

Requested Resource

Abstraction of requirements according to the resource model.

Resource Model

Conceptual model that is structured according to the resource-oriented architectural style.

Resource-oriented architectural style

Architectural style that restricts the identification, characteristics and allowed links and methods of resources and their representations.

RESTful Web Service

Web service defined according to the resource-oriented architectural style.

Semantic Model

Set of machine-interpretable representations used to model an area of knowledge or some part of the world, including software (Farrell and Lausen (eds.), 2007).

Semantic Resource

Resource that includes a labelling function that associates properties, as well as the semantic resource itself, with concepts taken from a semantic model.

Semantic Resource Network

Connected graph comprising of directed relations between semantic resources.

Sensor

Entity that provides information about an observed property at its output. A sensor uses a combination of physical, chemical or biological means in order to estimate the underlying observed property. At the end of the measuring chain electronic devices produce signals to be processed.

Service

Distinct part of the functionality that is provided by an entity through interfaces (ISO 19119:2005).

Service Instance

Executing manifestation of a software component that provides an external interface of a service according to a service type specification.

Service Type

Specification of the interfaces and the externally visible behaviour of a service according to a given platform.

Service Network

Set of service instances that interact in order to serve the objectives of applications. The basic unit within a service network for the provision of functions are the service instances.

Service-Oriented Architectural Style

Architectural style that restricts the roles, characteristics and allowed relationships of services and service consumers.

Service-Oriented Architecture

(1) Architectural approach that supports the creation of business processes from functional units defined as services (Zhang et al, 2008).
(2) Architectural style that supports service orientation (OpenGroup, 2008).

(Service) Platform

(1) Set of infrastructural means and rules that are applied in a multi-style service-oriented architecture.
(2) Set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented. (OMG, 2003)

Side-Condition

Describes constraints upon to the design process, dealing, for instance, with the request to use standards, to apply a given design methodology or to produce design artefacts according to a given template.

Software Component

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

System

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 (ISO/IEC 10746-2:1996).
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 Designer

Human who is responsible for the design of the system architecture supporting the problem solution.

Traceability

Possibility to derive all architecture and design artefacts from given requirements, and to retrieve all requirements from which architecture and design artefacts were derived (derived from Hatley, Hruschka and Pirbhai (2000)).

Uniform Interface

Interface with commonly agreed, well-defined semantics that is defined independently of the types of resource that are accessed and managed by the operations of the interface.

Universe of discourse

View of the real or hypothetical world that includes everything of interest (ISO 19101:2004).

Use Case

Models the behaviour of a system. A use case is a sequence of actions performed by the system to yield an observable result that is typically of value for one or more actors or other stakeholders of the system (Jacobson and Ng, 2005).

Use Case Model (of a system)

Serves as an agreement between the stakeholders (or customers) of the system and the developers on what the system should do and what qualities the system should have (Jacobson and Ng, 2005).

User

(1) Human who defines the problem on behalf of a stakeholder.
(2) Members of agencies (e.g. civil or environmental protection agencies) or private companies that are involved in an application domain and that use the applications.

Viewpoint

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 (ISO/IEC 10746-2:1996).

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.

11.3 Term Index

Note: This index references the first and/or the major occurrences of a term in the text.

Lehrstuhl für Interaktive Echtzeitsysteme
Karlsruher Institut für Technologie

Fraunhofer-Institut für Optronik, Systemtechnik und
Bildauswertung IOSB Karlsruhe

Service-orientation has an increasing impact upon the design process and the architecture of environmental information systems. This work specifies the SERVUS design methodology for geospatial applications, anticipating that service networks based upon standards of the Open Geospatial Consortium will be developed and deployed in future as an operational and dependable infrastructure.

SERVUS builds upon a reference model for geospatial service-oriented architectures. Inspired by the architectural style of Representational State Transfer for distributed hypermedia systems, SERVUS guides the system architect to rephrase use case requirements as a network of semantically-annotated requested resources and to iteratively match them with offered resources that mirror the capabilities of existing services.

ISSN 1863-6489
ISBN 978-3-86644-499-7

ISBN 978-3-86644-499-7
9 783866 444997 >