Model driven approach for open distributed systems using an Enterprise Architecture Framework

Melanie Langermeier

Angaben zur Veröffentlichung / Publication details:

Langermeier, Melanie. 2013. “Model driven approach for open distributed systems using an Enterprise Architecture Framework.” Augsburg: Universität Augsburg.

UNIVERSITÄT AUGSBURG

Model driven approach for open distributed systems using an Enterprise Architecture Framework

(Based on the master thesis of Melanie Langermeier)

Melanie Langermeier

Report 2013-03

January 2013

Institute for Software & Systems Engineering

INSITUT FÜR INFORMATIK

D-86135 Augsburg

Abstract

Open distributed systems are complex systems, which contain a lot of components provided by different vendors and built with different technologies. The use of well-established and standardized modelling techniques is one way to deal with the problems that occur in the specification and development process of these systems. Enterprise Architecture Frameworks provide a good foundation to structure the various required modelling techniques. Existing modelling solutions in the context of Enterprise Architecture Frameworks like UML4ODP, UPDM and ArchiMate do not provide optimal support for the specification of open distributed systems. In this report a coherent modelling approach for open distributed systems using an enterprise architecture framework (MODEA) is presented. In the approach the latest OMG standards like UML, SoaML BPMN and BMM are used. As enterprise architecture framework the Reference Model for Open Distributed Systems (RM-ODP) has been chosen. Finally MODEA is illustrated through the specification of two case studies.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In open distributed system a lot of different parties have to work together. The issue that “each domain has its own description techniques” (Lan09) and following “different fields speak their own languages, draw their own models, and use their own techniques and tools” (Lan09) affects the current practice in architecture specification. Nevertheless the various components of such a system have to interact to provide the required functionality although communication and decision making between is vendors gets really hard.

To avoid a Babylonian confusion it is important that the various vendors of the open distributed system agree on one language used for the specification and documentation of the system (ISO98a). Such a language should enable the creation of good and pragmatic models using techniques that are defined as formal as possible.

The use of well-established and standardized modeling techniques in the context of an Enterprise Architecture Framework is one way to reach this. Several publishers in the scientific context point out the necessity of specific techniques to deal with the growing complexity of information systems. Zachman said, that “the increased scope of design and levels of complexity of information systems implementations are forcing the use of some logical construct (or architecture)” (Zach99). And also Leist and Zellner (2006) stated that “As information systems and technologies grow in complexity and scope, the need for a coherent and comprehensive modeling approach becomes of paramount importance.” (Lei06).

Frameworks for enterprise architecture “structure architecture descriptions techniques by identifying and relating different architectural viewpoints and the modeling techniques associated with them.” (Lan09) Therewith they do not propose actual modeling concepts, but generally define the elements that are part of the architecture in quite precise way. “Modelling languages are an essential instrument for the description and communication of architectures.” (Lan07)

When choosing a set of actual modeling techniques for an enterprise architecture framework, especially in an open distributed context, well-established and standardized techniques like UML and BPMN should be chosen. Well-established modeling techniques have greater acceptance and the risk of misunderstanding their specification is lower. Using standardized modeling techniques provides the advantage that they keep stable and mostly have a good tool support.

Current approaches in this context are for example ArchiMate, UML4ODP or UPDM. ArchiMate is a very comprehensible approach, but it uses a completely new modeling techniques. UPDM is originally made for the military domain and therefore contains domain specific practices. UML4ODP is a complex approach, which is also overweighed in some parts. Additionally it does not provide explicit support for service-oriented architectures and also for different communication styles like RPC.

Service orientation is a paradigm that supports the developer by handling the complexity through structuring a system in services. “SOA represents a set of design principles that enable units of functionality to be provided and consumed as services. This essentially simple concept can and should be used not just in software engineering, but also at all other levels of the enterprise architecture, to achieve ultimate flexibility in business and IT design." (Ste05) The specification of the UML profile SoaML enables a new way to "define SOA concepts in terms of existing UML concepts". (OMG12a)

An initial comparison and evaluation of various Enterprise Architecture frameworks has been done, including Zachman, TOGAF with Archimate, DODAF/MODAF with UPDM, and ISO RM-ODP with UML4ODP. RM-ODP has been selected as the foundation for this work, as there already is an established history of applying RM-ODP within the geospatial and environmental domain. It is also possible to map the models of RM-ODP into other EA frameworks, like TOGAF or DOFAF/MODAF.

The goal of the report is to specify a coherent modeling approach for open distributed systems, illustrated by the use of an enterprise architecture framework and using standardized, well-established modeling techniques. For evaluation the defined approach is applied to two environmental systems. As practical examples two environmental systems pilot cases from running projects are chosen. The first one is the Personal Environmental Information System (PEIS) from the ENVIROFI Project. The second one is the Oil Spill pilot from the ENVISION Project. Both are large, distributed systems which contain a lot of services and data services to be integrated.

1.2 Method of work

The research method in this report is based on the concepts of the basic engineering paradigm described by Denning et al. (Den89). He proposes the following four steps to solve a given problem:

1. State requirements
2. State specification
3. Design and implement the system
4. Test the system

This method will be adapted in the report. Following the first step is to introduce the pilot cases and formulate the problems that come the open distributed context and also the ones from the examples. The next step is to define requirements for a model based approach for open distributed systems based on the identified problems. Thereby a requirements matrix will be established as foundation for further analysis and comparisons.

As second starting basis the existing model based approaches for enterprise architecture will be examined and evaluated with use of the requirements matrix. The identified gaps in the analysis of the existing solutions specify the issues that have to be considered, when defining the model driven approach for open distributed system using an enterprise architecture framework (MODEA) in the next step. The dependencies around MODEA are shown in Figure 1.

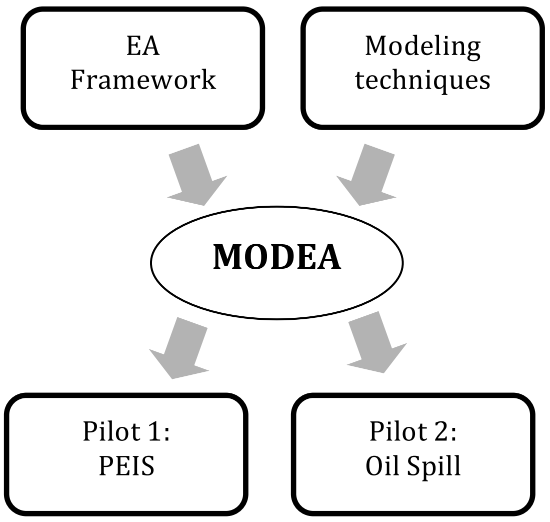


Figure 1 MODEA and its dependencies

With use of the structure and the defined concepts in an Enterprise Architecture Framework MODEA is introduced. This refers to the design and implement step of in (Den89).

Since the product is a specification approach, the testing step contains the application of the approach to the two example cases and a verification of the defined requirements in the beginning. Finally the results will be compared to the existing solutions and further work will be described. The following figure summarizes the method of work in this report.

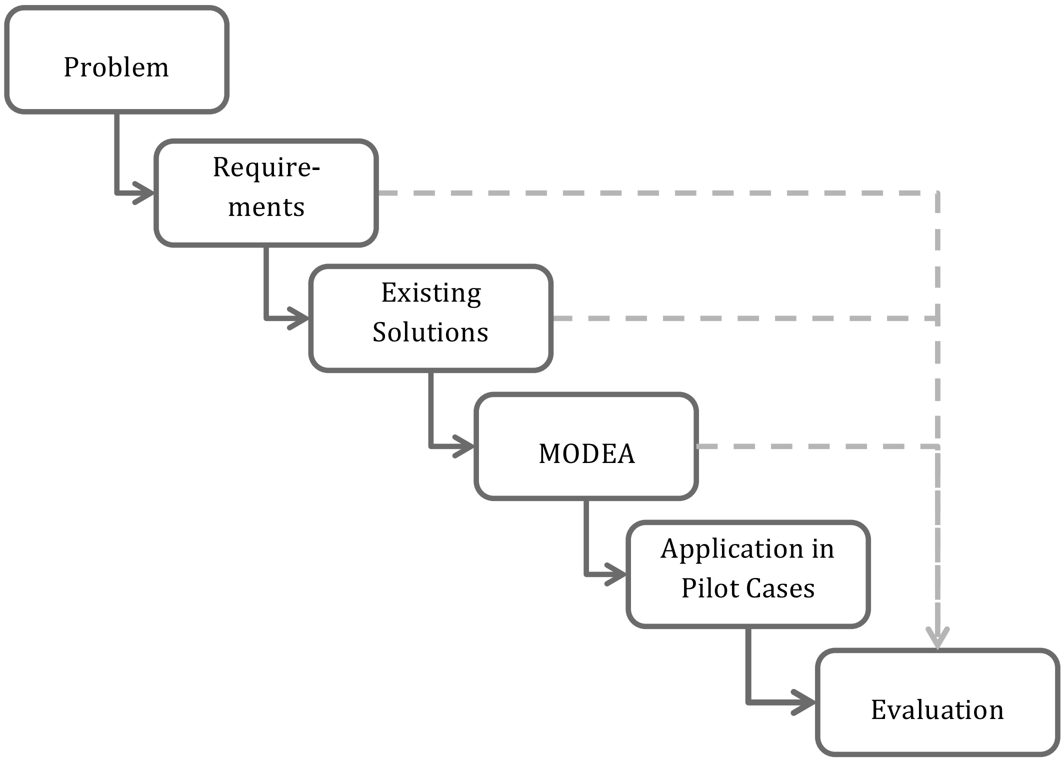


Figure 2 Method of Work

The following research questions will be answered in the report:

1. What are the most important Enterprise Architecture Frameworks and which model based approaches exist for their application?
2. How to specify an open and heterogeneous system with continuous modeling using an enterprise architecture framework?

    a. How to model the concepts of an Enterprise Architecture Framework using existing standardizes modeling techniques?
    b. How to integrate the various views of the models?
3. How to apply the approach to an environmental system?
    a. How can modeling tools support the application of the approach?

This concludes into three results. The first one is an analysis of the various existing Enterprise Architecture Frameworks and a validation of their corresponding modeling approaches. The second one is the model-driven approach for open distributed systems MODEA. And at least the specification of this continuous model based application in two pilot cases.

2 Problem definition

In this chapter at first the two pilot cases are introduced and their current challenges are outlined. The projects are the Personal Environmental Information System PEIS and the Oil Spill Decision Support System. Both systems are open distributed system and currently under development. In the following Open Distributed System in general will be defined. Through analyzing the pilot cases with respect to the characteristics of open distributed systems, the problems in developing and maintaining such systems are identified. At least Enterprise Architecture Framework are introduced, especially the Reference Model for Open Distributed Processing RM ODP. It is described how RM ODP can be used to deal with the defined problems.

2.1 Pilot 1: Personal Environmental Information System

2.1.1 Introduction

The Personal Information System for air pollutants, allergens and meteorological conditions (PEIS) is one of the two pilot cases. The aim of the system is to provide data about the atmospheric conditions to sportspersons, allergic persons or environmental observers. (PEIS 2.1)

PEIS is part of the project **Enviro***nmental Observation Web and its Service Applications within the* **F***uture* **I***nternet (ENVIROFI)*. This project is one of the eight research projects within the Future Internet Public Private Partnership (FI PPP) program of the EU. This program is funded by the European Commission with the goal to “advance a shared vision for harmonised European-scale technology platforms and their implementation, as well as the integration and harmonisation of the relevant policy, legal, political and regulatory frameworks.” (FI-PPP12)

The FI WARE project of the FI PPP program provides a Core Platform for the Future Internet. It offers a set of reusable components for all usage areas. These so called Generic Enablers capture the domain-independent functionality required for the Future Internet.

ENVIROFI deals with the environmental usage area of the Future Internet and provides generic, but environmental domain-specific functionalities, so called environmental enablers. “ENVIROFI’s vision is to establish an Environmental Observation Web in which all environmental data, whether from sensors, citizens, or models, are available anytime anywhere through the Internet in a standardized, usable format.” (ENV12b)

ENVIROFI consists of three use cases with the topics biodiversity, human/environment interaction and collaborative usage of marine data. Based on the requirements of the three pilots in the ENVIROFI project generic use cases and architectural styles for the environmental domain are specified. It also defines specific enablers needed for the pilots and gives input for the functionality required by the generic enablers from FI WARE. (ENV12c)

PEIS as one of the pilots provides requirements for ENVIROFI and then adapt the architectural styles with usage of the established specific and generic enablers to implement them in further iterations.

The PEIS project consist of three parts:

1. Personal Assessment System
2. Notification and Early Warning System

3. User Input Interface

The Personal Assessment System should provide meteorological, air quality and air pollution information about past, present and future conditions. The Notification and Early Warning System offers functionality to inform the user about specific predefined environmental situations. The User Input Interface enables the interactive part of the system. The user can provide environmental data and interact with other users. (PEIS 2.1)

2.1.2 Challenges in the project

Since the project contains generic and specific services in the environmental context it has to deal with a huge application landscape. Therefore support for the implementation is needed to identify the required services. Existing capabilities in the environmental and generic enablers of FI WARE and ENVIROFI should be reused as much as possible when implementing the PEIS system.

In the existing deliverables of ENVIROFI, PEIS and FI WARE projects various modeling approaches are used at the moment. These are for example:

- FMC Diagrams for FI WARE enablers
- UML Use Cases and Interaction Diagrams for the description of the Use Cases in the PEIS Specification
- Box Diagram for describing the ENVIROFI architecture

FMC stands for Fundamental Modeling Concepts and “primarily provide a framework for the comprehensive description of software-intensive systems.” (FMC12) Therewith also a graphical notation for creating models is included.

The term Box Diagrams is used in the report for referring to diagrams, which consist of boxes and lines and have their own syntax. They are not compatible with standardized techniques and typically do not have a syntactically support in modeling tools. Examples of a so called Box Diagrams are shown in Figure 3 and Figure 5. They are describing the Overall ENVIROFI architecutre and a ENVIROFI instance architecture.

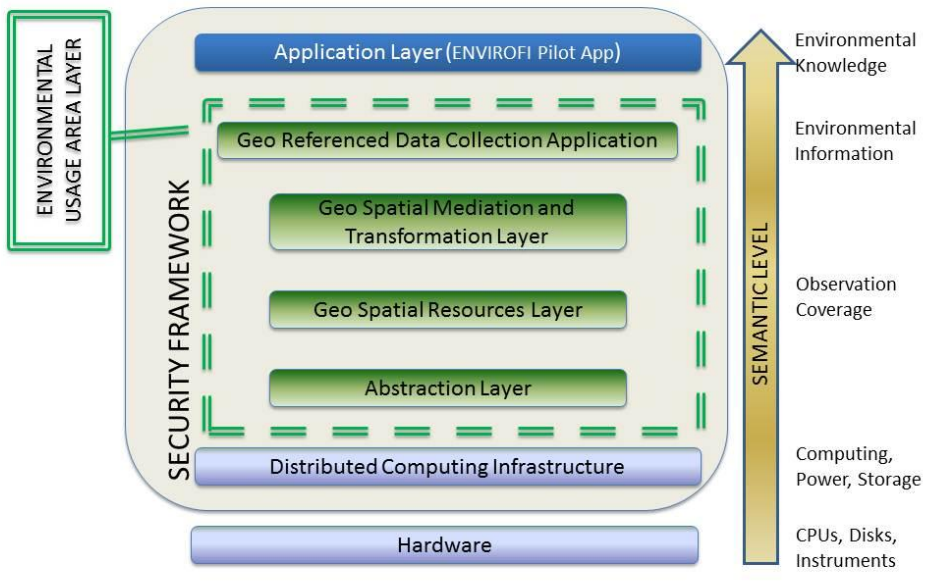

ENVIRONMENTAL USAGE AREA LAYER
Application Layer (ENVIROFI Pilot App)
Geo Referenced Data Collection Application
Geo Spatial Mediation and Transformation Layer
Geo Spatial Resources Layer
Abstraction Layer
SECURITY FRAMEWORK
Distributed Computing Infrastructure
Hardware
SEMANTIC LEVEL
Environmental Knowledge
Environmental Information
Observation Coverage
Computing, Power, Storage
CPUs, Disks, Instruments

Figure 3 ENVIROFI Overall Architecture (PEIS 5.2.2)

Figure 4 shows a Box Diagram describing the basic data flow, when a user interacts with the system.

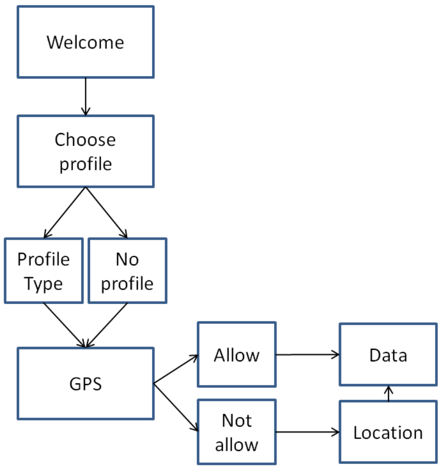


Figure 4 Box Diagram for Data Flow (PEIS 2.3.2)

Most of the diagrams stand for themselves and relationships between them are not shown up. For example in the Deliverable 5.2.2 from PEIS the Overall architecture (Figure 3) is described with its layers and afterwards the ENVIROFI instance (Figure 5) is introduced. But theres no information about how elements of the both diagrams relate to each other.

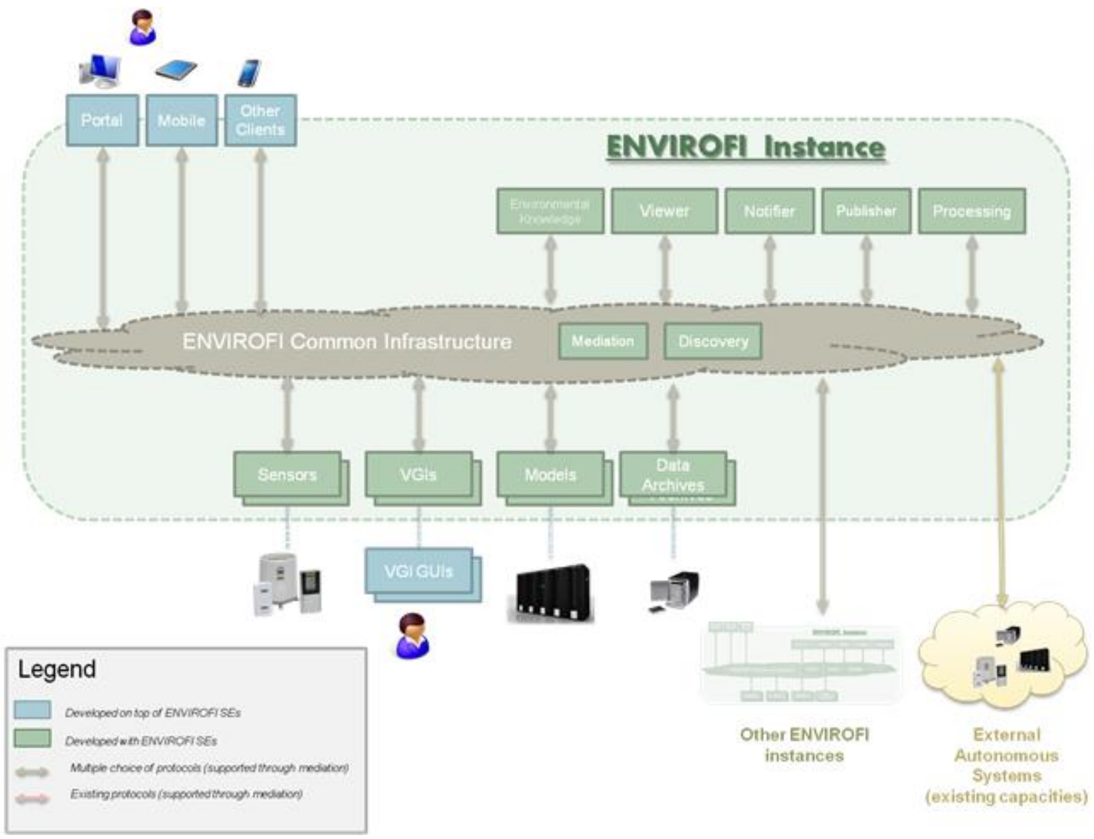

Portal
Mobile
Other Clients
ENVIROFI Instance
Environmental Knowledge
Viewer
Notifier
Publisher
Processing
ENVIROFI Common Infrastructure
Mediation
Discovery
Sensors
VGIs
Models
Data Archives
VGI GUIs
Legend
Developed on top of ENVIROFI SEs
Developed with ENVIROFI SEs
Multiple choice of protocols (supported through mediation)
Existing protocols (supported through mediation)
Other ENVIROFI instances
External Autonomous Systems (existing capacities)

Figure 5 ENVIROFI instance (PEIS 4.2)

Though the iterative approach used in the project there is one major challenge to keep the deliverables and specification documents consistent. The huge scope with its separation into several work packages makes this task even more difficult. For example the Geo Referenced Data and Application Layer in Figure 3 is called Environmental Information Layer in the description of the diagram. Both names are used in the several deliverables without mentioning, that they refer to the same layer. (PEIS 5.2.2)

In D5.2.1 from PEIS a mapping of the enabler to the layers defined in Figure 3 was made. Thereby the use-dependencies between the layerns and the use-dependencies between the enabler categories are inconsistent. This is illustrated in Figure 6.

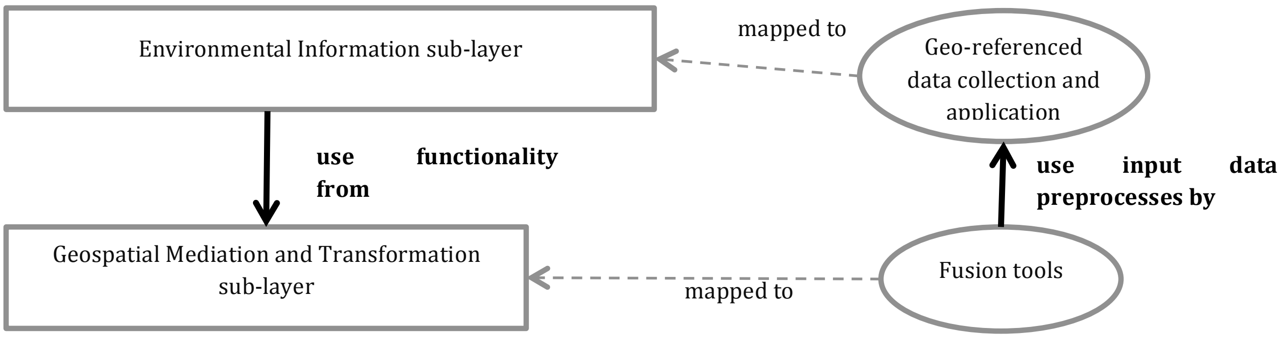

Environmental Information sub-layer
mapped to
Geo-referenced data collection and application
use functionality from
use input data preprocesses by
Geospatial Mediation and Transformation sub-layer
mapped to
Fusion tools

Figure 6 Inconsistencies in layer and enabler description (Own contribution based on PEIS 5.2.1)

The Environmental Information Layer is provided through usage of Geo-referenced Data Collection and Application Enablers. The Geospatial Mediation and Transformation Layer containts the Fusion Tools Enablers. The description of the Geo-referenced Data Collection and Application Enablers specifies the relationship between this two enabler categories as follows: "The services in this thematic class provide ways [...] for later use by other specific enablers such as fusion services." (PEIS 5.2.1) But the Environmental Information Layer is on top of Geospatial Mediation and Transformation Layer. This means that it uses the functionality of the lower layer.

2.2 Pilot 2: Oil Spill Decision Support System

2.2.1 Introduction

The second pilot case is taken from the ENVISION project. "The envision project provides an **envi**ronmental **s**ervices **i**nfrastructure with **on**tologies that aims to support non ICT-skilled users in the process of semantic discovery and adaptive chaining and composition of environmental services" (ENV1.2). The ENVISION infrastructure will be validated with usage in the three environmental pilot cases Landslide, Oil Spill and Floods. In this report the Oil Spill Pilot is used.

The Oil Spill System supports decision making in the case of accidental oil releases the sea. Therefore the prediction of the drift and the effects on cods are main functionalities of the system. To provide the functionality external data services and prediction models will be composed together at design time. The prediction is made available to the user at runtime via a scenario website. An overview of the usage of ENVISION to provide a Oil Spill Scenario Website is shown in Figure 7. (ENV 1.1)

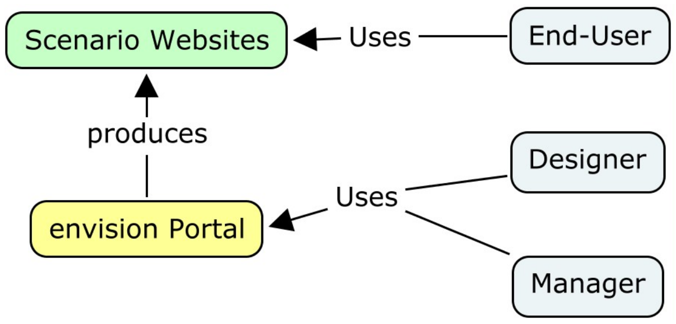


Figure 7 Usage of ENVISION (ENV 1.2)

The required services are provided as Models-as-a-Service (MaaS) chaining processing services, data and sensor services. A MaaS in this project is understood as "A model made available as a web service. A composition the user can interact with." (ENV 1.4) A Model is a "computer simulation of real world processes to make forecasts of a certain behavior of natural phenomena." (ENV 1.4) The oil spill pilot provides a decision support portal based on the knowledge provided by two web services.

- The Oil Drift Prediction calculates the oil spill concentration in three dimension plus time. Therefore a prediction model service and data sources for Spill data, Wind forecast, Sea depth data and Coast Line Data are required.
- The Cod Effect Prediction calculates the effects for cod population of an oil spill. The Oil spill prediction data as well as data about the species and populations of the cods a required.

The overall Workflow is shown in Figure 8. (ENV1.1)

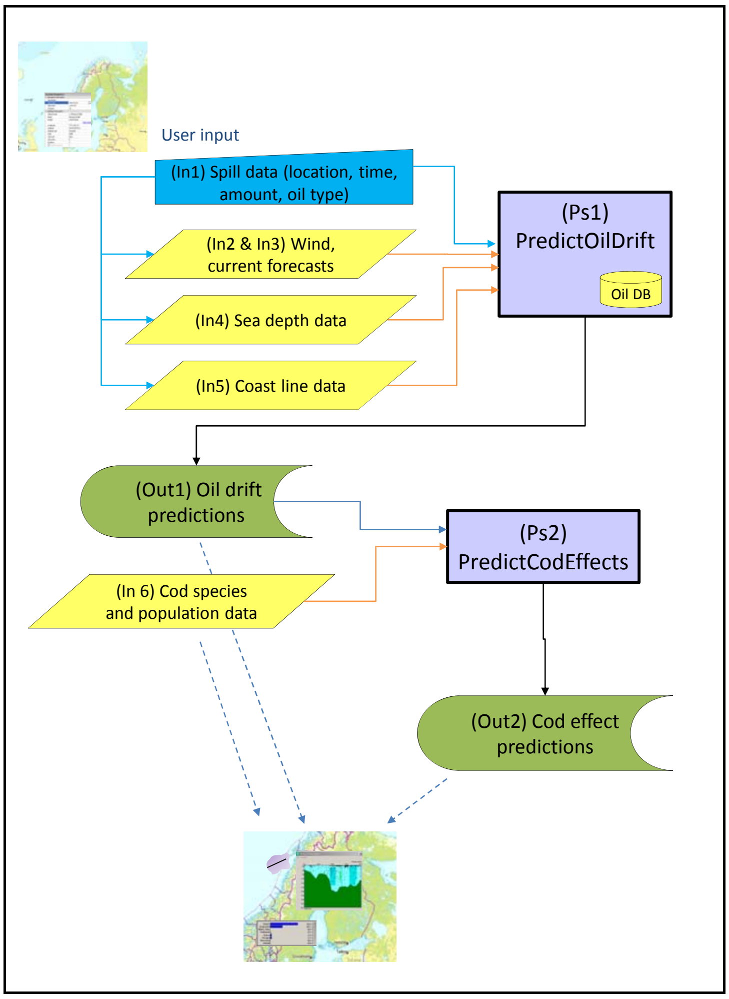

User input
(In1) Spill data (location, time, amount, oil type)
(In2 & In3) Wind, current forecasts
(In4) Sea depth data
(In5) Coast line data
(Ps1) PredictOilDrift
Oil DB
(Out1) Oil drift predictions
(In 6) Cod species and population data
(Ps2) PredictCodEffects
(Out2) Cod effect predictions

Figure 8 Workflow of Oil Spill Decision Support (ENV 1.1)

With support of the ENVISION infrastructure the two Models-as-a-Service will be developed to provide the functionality for the Oil Spill Decision Support. They will be provided "through an automated request system for model runs, with online visualization and analysis tools and through standard data formats for simulation data interoperability" (ENV 1.1)

The ENVISION infrastructure provides functionality directly required to provide the scenario website like the Map Controller or an Execution Environment for MaaS. It provides also functionality used at design time to discover and chain services using semantic technologies. (ENV 1.2)

ENVISION has the goal to provide this functionality for the use non ICT-skilled users. A non-ICT skilled Workflow- and Web-Site-Designer as well as a Resource Manager use the ENVISION Portal to create the Oil Spill Decision Support System. The creation of the scenario website is done at design time using ENVISION with its infrastructure. ENVSISION provides functionality to manage the required resources, ontologies and compositions and create models on line as Model-as-a-Service. Following the designer is able to configure web services and scenario websites for specific communities. ENVISION provides also solution to design the workflow and execute as required in the oil spill modeling domain. (ENV 1.2, ENV 1.4)

2.2.2 Challenges in the project

The technical part of the functionality in the Oil Spill System is provided by using the ENVISION portal and infrastructure. Using external processing, data and sensor services provides the domain relevant part of the system. Following there are many different vendors providing components to the system, i.e. SINTEF ICT provides a Composition Module, SINTEF Met provides the Oil Spill and Cod Effects Prediction Models and cost line data is provided by the Norwegian Mapping Authority. All the vendors have to collaborate with each other to and share their knowledge and capabilities to provide the Oil Spill Decision Support System. Overall the system is more or less a composition or chaining of existing services, which are then executed at runtime. (ENV 1.4)

In the current deliverables there are only informal architectural descriptions of the Oil Spill Decision Support following the five viewpoints of RM-ODP. Most of the time textual descriptions are used to describe each viewpoint and simply referring to parts of the ENVISION infrastructure. An overall oil spill specific picture of the architecture is missing. If there are architectural descriptions related to Oil Spill they are very generic in the case of ENVISION-oriented. (ENV 1.2, ENV 1.4)

Also a problem in the current specifications is a missing explicit differentiation between run-time and design-time aspects for the oil spill system. At run-time the user accesses the scenario website and executes the models. Then he wants to analyze the result using graphical simulations, changing the time or map section. At design-time a manager administrates the resources and ontologies and a designer configures the website and creates and deploys service compositions. Following it has to be differed between components used for run-time and these ones used for design-time in the Oil Spill Project. Also there must be a possibility to integrate the specification of the ENVISION components used to get an overall picture of the system.

The overall ENVISION infrastructure is also specified using the RM ODP viewpoints. A further description of this approach will be made in chapter 4.3.1.

The single Components like the Decision Support Portal or the MaaS Composition Portal are each described without the use of any framework (Deliverables 2.x until 6.x, ENV12a). Most components use Concept Maps to describe the overall architecture and their relations to other components of ENVISION. Figure 9 shows such a concept map, in this case from the MaaS Composition Portal.

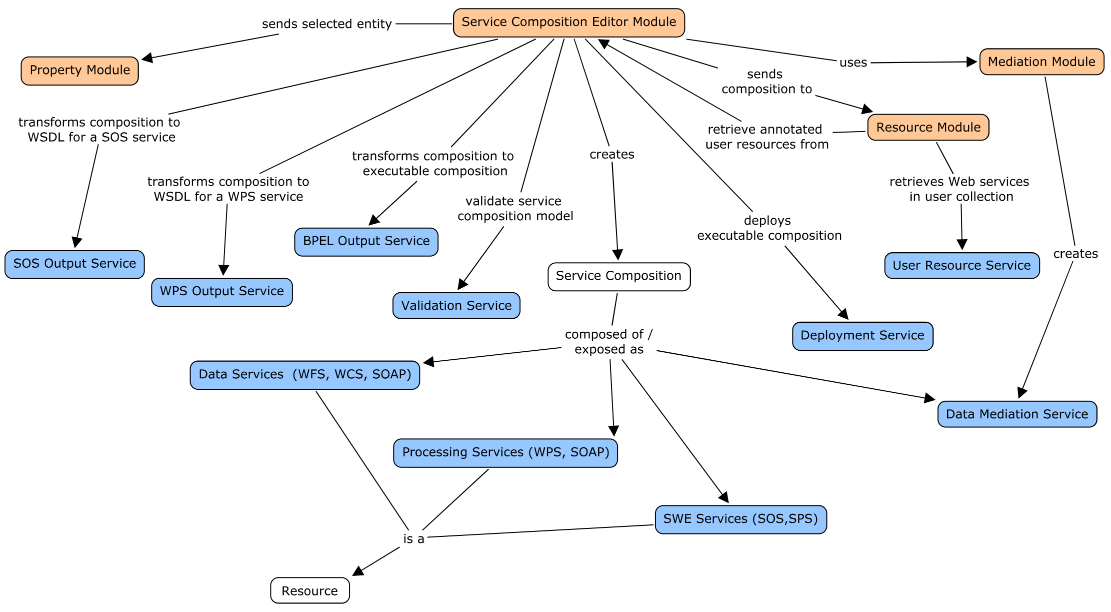

Service Composition Editor Module
sends selected entity
Property Module
transforms composition to WSDL for a SOS service
SOS Output Service
transforms composition to WSDL for a WPS service
WPS Output Service
transforms composition to executable composition
BPEL Output Service
validate service composition model
Validation Service
creates
Service Composition
uses
Mediation Module
sends composition to
Resource Module
retrieve annotated user resources from
retrieves Web services in user collection
User Resource Service
deploys executable composition
Deployment Service
creates
composed of / exposed as
Data Services (WFS, WCS, SOAP)
Processing Services (WPS, SOAP)
SWE Services (SOS,SPS)
Data Mediation Service
is a
Resource

Figure 9 Overall Architecture of the MaaS Composition Portal (ENV 3.1)

Despite Concept Maps and various kinds of process diagrams, also simple box-and-lines diagrams with their own syntax are used. One example for this is the description of the architectural components of the Envision Execution Infrastructure shown in Figure 10.

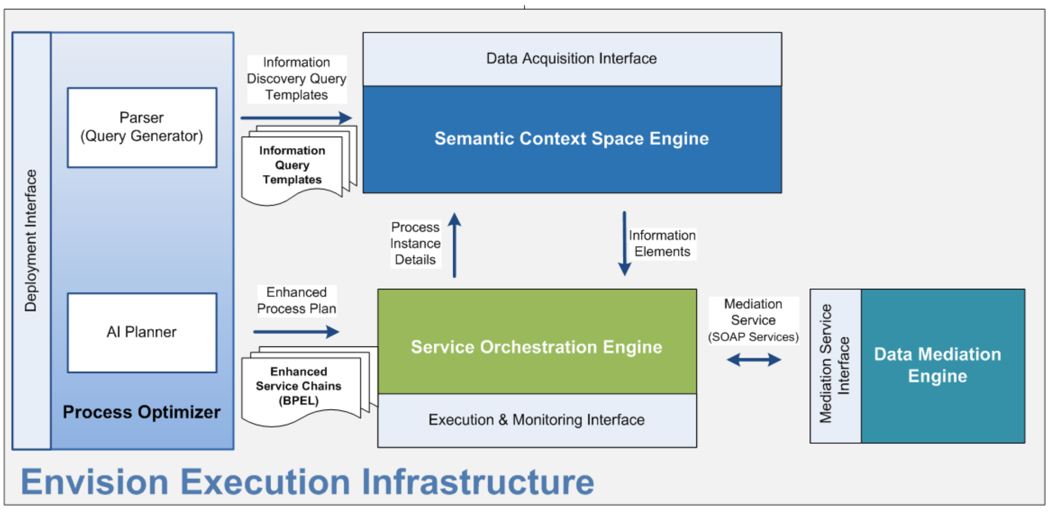

Deployment Interface
Parser (Query Generator)
AI Planner
Process Optimizer
Information Discovery Query Templates
Information Query Templates
Data Acquisition Interface
Semantic Context Space Engine
Process Instance Details
Information Elements
Enhanced Process Plan
Enhanced Service Chains (BPEL)
Service Orchestration Engine
Execution & Monitoring Interface
Mediation Service (SOAP Services)
Mediation Service Interface
Data Mediation Engine
Envision Execution Infrastructure

Figure 10 ENVISION Execution Infrastructure Architectural Components (ENV 6.1)

Another example for a simple box-and-line diagram is the overview of the Envision Focus areas in Figure 11.

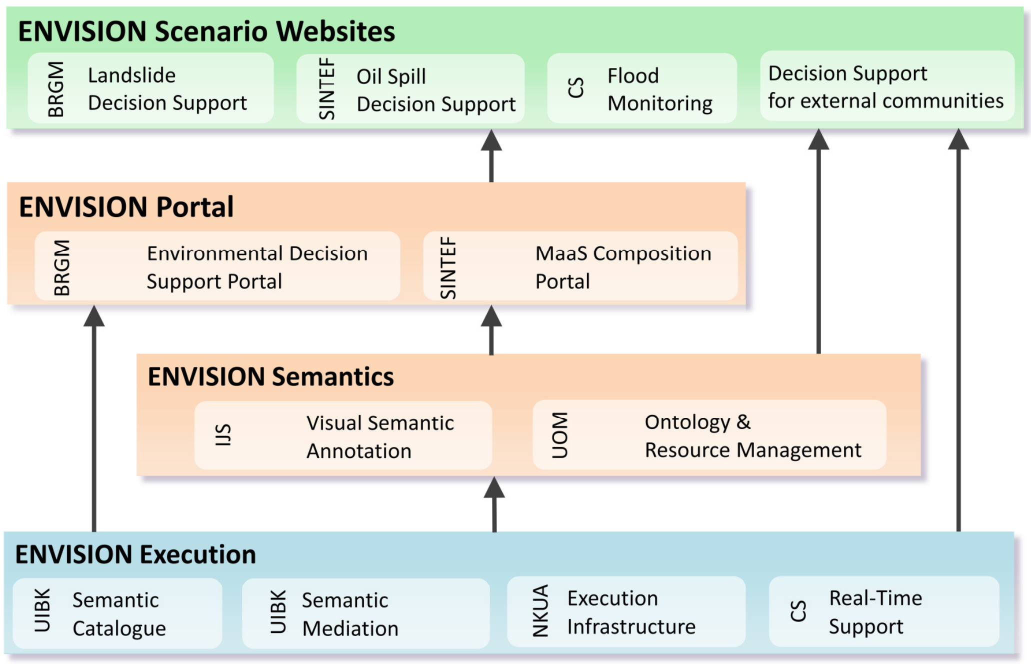

ENVISION Scenario Websites
BRGM Landslide Decision Support
SINTEF Oil Spill Decision Support
CS Flood Monitoring
Decision Support for external communities
ENVISION Portal
BRGM Environmental Decision Support Portal
SINTEF MaaS Composition Portal
ENVISION Semantics
JSI Visual Semantic Annotation
UOM Ontology & Resource Management
ENVISION Execution
UIBK Semantic Catalogue
UIBK Semantic Mediation
NKUA Execution Infrastructure
CS Real-Time Support

Figure 11 ENVISION Focus areas (ENV 1.4)

This is one of the few diagrams giving some kind of architectural overview over Envision. Some parts of the system are described using UML Use Cases, some kind of UML Collaboration diagram and UML Interactions Diagram (ENV 1.2). A diagram how all the components work together on a higher abstraction level is missing. Especially between the specifications in the various Work Packages the models are not connected to each other. This concludes to a high risk of inconsistencies, when changes are made.

2.3 Open distributed systems

In the following open distributed systems are introduced. First they are defined and the main characteristics are described. In the following problems occurring during the development and maintain process of them, especially when trying to specify the system are illustrated.

2.3.1 Definition

A distributed system consists of a number of “hardware and software components located at networked computers [which] communicate and coordinate their actions only by message passing” (Cou05). There are two different reasons for distribution. First a system is “inherently distributed and in connecting its systems into a seamless whole, a distributed systems appears” (Cro96). Second an “inherently centralized information processing system [is taken] and distribute[d ...] to achieve higher reliability, availability, safety or performance, or all of the above” (Cro96).

Such distributed systems are significantly more complex than centralized systems. An increasing scope leads to an increasing number of involved people and components and often concludes in a complex definition of the system and communication problems (Lei06). In distributed system you also have to care about the synchronization of processes and the consistency of data, since there is no global clock and no global state relating the whole system. (Cro96)

There are several characteristics of distributed systems you have to deal with. The following table gives an overview of them according to Cou05.

Heterogeneity	Manage the heterogeneity of Networks, Computer Hardware, Operating Systems and Programming Languages in distributed systems.
Openness	It must be possible to integrate components written by different programmers. The specifications and documentations of the various interfaces from the components must be made available.
Security	This includes Confidentiality, Integrity and Availability of the System. Protect the system from attacks against communication channels and processes.
Scalability	The system should remain effective although there is a increase in the number of participating components or users.
Failure Handling	Take of an increased number of failure rate due to more components. Deal with detection, masking, recovery and tolerating of failures.
Concurrency	Each resource must be designed to be safe in a concurrent environment. It must ensure the integrity of the data and the consistency of information all the time.
Transparency	Hide a specific characteristic of the system with respect to cost and performance trade offs. This can be Access, Location, Migration, Relocation, Replication, Concurrency, Failure or Persistence.

Table 1 Challenges in distributed systems (Own contribution based on Cou05)

In the ongoing report we will concentrate on the heterogeneity, openness and transparency characteristics, since they require concepts in the approach to deal with them. Security, Scalability, Failure Handling and Concurrency are more concerned with the design of the system.

"An Open Distributed System is made up of components that may be obtained from a number of different sources, which together work as a single distributed system" (Cro96). Components participating in Open Distributed System are not only from one vendor, they can be provided from several ones. Thereby the system compasses heterogeneous IT resources and multiple domains. Open distributed systems become important because of an increasing demand on information exchange between cooperating organizations and a growing need of interconnect information processing services to provide the required functionally (ISO98a).

2.3.2 Problems

The following figure shows the seven main problems, which have to consider when specifying and open distributed systems. Each problem is described further in detail and illustrated through linking to the example cases introduced above.

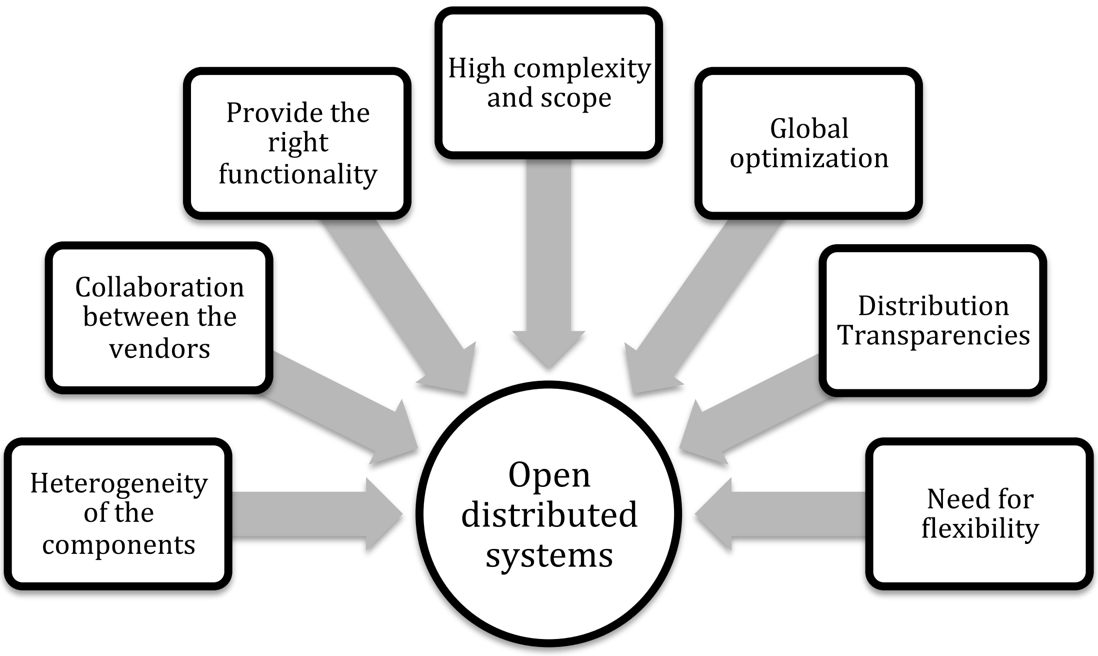

High complexity and scope
Provide the right functionality
Global optimization
Collaboration between the vendors
Distribution Transparencies
Heterogeneity of the components
Open distributed systems
Need for flexibility

Figure 12 Challenges for the architecture of an open distributed system

Problem 1: Heterogeneity of the components

A major challenge in these kinds of systems is their realization in an "environment of heterogeneous IT resources and multiple organizational domains". (ISO98a) The fact that the "components [...] may be obtained from a number of different sources" (Cro96) increases the complexity when dealing with such a system. The variety of vendors and components lead to a broader scope of the system, complicating its structure and make it more complex to see the system as a whole.

For example in the PEIS project there are 45 environmental enablers, which have to be taken into account in the design process to explore reuse facilities. On top there are 46 generic enablers as well as various sensor and data sources, which have to be considered (PEIS 4.2). Each of these components is provided using different technologies, since they are provided by different organizations and developed in different projects.

Also in the Oil Spill project several sensor and data sources are required. Most of the components are implemented using different implementation languages and technologies. For example "Bathymetry data (depths) are made available as a Web Coverage Service, the Coastline is made available as a Web Feature Service" in the Oil Spill System. (ENV 1.4) Even so the components have to collaborate with each other to provide the required functionality of the open distributed system. The major challenge is to manage the interaction between the different components and to deal with the dependencies between them.

Problem 1
Open distributed systems contain a huge number of components interacting together. How to cope with the heterogeneity of them?

Problem 2: Orchestration of a multi-vendor environment

A special characteristic of open distributed systems is, as already described in the above problem, that different vendors can provide parts of the overall functionality of the open distributed system. The vendors have to cooperate to provision the open distributed system, because “a single vendor will not have all of the answers”. (ISO98a)

Complicating in current practice the vendors or different domains often use their own techniques including languages, models and tools for creating a specification (Lan09). Concluding there is huge variety of modeling techniques and tools, which makes it very difficult to build one single consistent specification of the whole open distributed system.

In the PEIS projects the specification of the environmental enablers and the geospatial enablers is completely different. Environmental enablers are described using a predefined template. (PEIS 5.2.2) Geospatial enablers are described with FMC diagrams and informal text. (FIWiki12) The observation data sources for air quality, pollen or meteorological data are only described with informal text. (PEIS 2.3.2) Nevertheless both of the enabler groups as well as the observation data sources should be used and integrated to provide the PEIS.

The involvement of different domains provides also challenges with the used vocabulary. For example in the Oil Spill project the term “model” leads to misunderstanding between the domain specialists and the IT specialists. The domain specialist providing the oil drift prediction logic understands under the term “model” a “computer simulation of real world processes to make forecasts of a certain behaviour of natural phenomena.” (ENV 1.4). The IT specialist understands a model as visualization of an existing or required system.

Problem 2
Various vendors from different domains, using mostly different specification techniques, have to collaborate for the provisioning of the open distributed systems. How to provide an efficient communication and collaboration environment?

Problem 3: Provide the right functionality

An important aspect in every IT system is the alignment of the Business requirements and the IT System implementation. “Architectural Alignment, and business and IT alignment in particular, have proved to be difficult problems in enterprise architecture” (Ste05). The authors define two reasons for this issue: One are the differences in architectural modeling methods used from Business analysis and IT architects. The other one is a lack of an “overarching set of design rules governing the structuring of the various architectures making up the enterprise architecture” (Ste05).

Especially if the provided functionality is distributed to various components from different vendors, it is more complicated to ensure that the system provide the right functionality in the end. Essential for this is a specification of the goals and the vision for the system as well as the usage of techniques, which is understood by both, the users defining the requirements and the designers and implementers providing the system. Especially the tracing of goals to the fulfilling components is a very difficult task.

This concerns a major actual question in the PEIS project. At the moment there is a specification of the user requirements as well as a large number of components providing functionality to meet them. Now the task is to create a system design with use of the existing functionality to fulfilling the user requirements.

Problem 3
It is necessary to ensure that the open distributed system provides the required functionality for the user.   How can this be ensured?

Problem 4: High complexity and scope

As seen in the PEIS and Oil Spill projects in open distributed system often a system-of-system approach is supported. (PEIS5.2.2) It enables a structuring of the variety of components through partitioning the overall systems into smaller ones, each one providing and requiring specific functionality. A system of systems describes the "integration of many independent, autonomous systems, frequently of large dimensions, which are brought together in order to satisfy a global goal and under certain rules of engagement." (Kar10) The huge amount of systems, components and vendors working together in such system increase the complexity and scope of it. "The complexity involved in the specification of large, open distributed systems is constantly growing, due to the increasing size of software applications and the increasingly stringent requirements on their functionality, performance, reliability, security, availability, etc." (Rom12)

To describe such complex structures adequate solutions for establishing the specification and in the best case also a support for the implementation is required. (Kai05, Lei06).

In the PEIS project in Figure 6, chapter 2.1.2, inconsistencies detected in the current deliverables are presented. The defined use-relationships between the layers and enabler categories are contradictory. In the establishing process of the specification of open distributed systems several groups work on different parts. Uncontrolled changes or updates lead to an increasing complexity of the specification and also the corresponding system. "During the enterprise architecting process, changes and updates are likely to occur quickly" (Kai05). This concludes in different versions and the artifacts, which have to be managed to keep the overall specification consistent. If the changes "are not applied under the control of well-organized architecture, they will lead to more complexity and inefficient software systems" (Kho09).

Problem 4
A high number of components and vendors as well as changes and updates increase the complexity and scope of a system.   How to manage the complexity?

Problem 5: Overall optimization

A large system with many participating parties has several stakeholders and "different stakeholders require different perspectives" (Kai05). For example a database owner is interested in the "structure and location of specific databases, while a sales executive may be focused on the location and movement of data through multiple information systems" (Kai05).

Steen et al. examined that "each type of architecture is supplemented with guidelines and best practices for optimal design" (Ste05), but "concepts for expressing global optimization and criteria that guide this optimization across different architectures [are lacking]" (Ste05) in enterprise architectures.

Each of these architectures has its best practices and techniques for the design of it. Thereby "architectures within this domain may be optimal" (Lan09) but not necessarily for the overall organization.

Problem 5
Techniques for optimizing the architecture of one domain are well established unlike optimization techniques covering all parts and domains. How can an overall optimization be supported?

Problem 6: Distribution Transparencies

"The openness is a requirement of the autonomy of different users to acquire, install and operate different appropriate systems while maintaining consistent distribution mechanisms across all users' systems." (Cro96) The distribution mechanisms required for the collaboration between the different systems should be therewith hidden from the user. "A transparency is some aspect of the system that is hidden from the user." (Cro96)

The overall architecture from ENVISION (Figure 9) and also the ENVIROFI instance architecture (Figure 5) only show the components participating in the system and the relationships between them. Details showing how the communications over the distributed network works are hidden. They are defined at a different section in the specification. This is conform to the definition of Crowcroft, who said that "a transparency is provided by including some set of mechanisms in the distributed system at a layer below the interface where the transparency is required." (Cro96)

It is necessary to define to which scope the user or parts of system have to deal with the distribution or if they can use or participate in the system regarding it as a centralized one. Otherwise at some point you have to define how the distribution will be implemented. Examples for transparencies are: Access, Location, Relocation, Replication, Concurrency, Failure and Persistence Transparencies. (Cou05)

Problem 6
Usage of Distribution Transparencies is necessary when implementing a distributed system. How can this task be supported?

Problem 7: Need for flexibility

As the term open distributed systems says, these systems open for the integration of new components. Through ad-hoc relationships with new partners or through recombination of existing services these systems are able to provide new functionality. To enable this it is important to specify a flexible architecture, where it is easy to substitute or integrate components. (Ste05) Also Khoshnevis et al. point out the importance of "reusability and flexibility in dealing with changes" (Kho09)

But especially this issue is lacking in incurrent enterprise software development, after Kraftzig et al. it “always suffers from a lack of agility and from inefficiency” (Kra05). Concluding enterprises are not able to align fast enough their business requirements to the IT infrastructure.

“Enterprises have to be increasingly efficient, flexible and innovative to be successful-” (Ste05) New products or service must be integrated in the existing systems and structures fast and easily. Drivers for flexibility are for example a high interoperability and a loose coupling between the components. They enable easy a replace and integration and therewith an effective cost and resource management. (Ste05)

Problem 7
Flexible architectures for easy adoption, reuse, substitution and integration are required. How to enable the design and development of such architectures?

2.4 Architecture Framework RM ODP

In the previous chapter the problems occurring during the design and specification of open distributed systems are described. In the following Enterprise Architecture Frameworks are introduced. They provide concepts describing how to deal with these kinds of problems. As foundation in the report the Reference Model for Open Distributed Processing RM ODP is then introduced and it is shown how it addresses the defined problems.

2.4.1 Definition Enterprise Architecture Framework

The architecture of a software system is defined as the “fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” (ISO11). This architecture description facilitates a good communication in the development projects by providing a common understanding of the system. Thereby the focus lies on the main part of the system and its structure; irrelevant details are hidden. (Lan09) The systematic approach for the establishment of the architecture is called an architecture framework. Usually a framework contains different viewpoints on a system to enable the description of different perspectives. (Tan04)

Frameworks exist on the software level but also on the enterprise level. Enterprise, as defined in TOGAF from the Open Group, is “any collection of organizations that has a common set of goals” (TOGAF9). The Enterprise Architecture compasses the architecture of the whole organization. That includes the structure of the organization, the business processes, their application support and also the technical infrastructure. Through a coherent description of the different parts of the enterprise architecture it becomes an instrument in controlling the complexity of enterprises ant its processes and system (Lan09).

2.4.2 Introduction to RM-ODP

In the report the RM-ODP framework is used as foundation for the modeling approach. The Reference Model for Open Distributed Processing (RM-ODP) is a framework for specifying and building large or complex systems. It is published as standard by ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) in cooperation with ITU-T (the telecommunications standards forum). (ISO98a, ISO98b, ISO10a, ISO10b)

"The aim of the Reference Model for Open Distributed Processing (the RM- ODP) is to provide a framework for specifying and building large or complex systems" (Lin11). The systems are called Open Distributed Processing systems and their can be amongst others classical IT systems, information systems, embedded systems or business systems. The objective of the framework is to "allow the benefits of distributing information processing services to be realized in an environment of heterogeneous IT resources and multiple organizational domains" (ISO98a). Especially the variety of vendors and technologies requires consistent concepts and rules for the description of the architecture.

RM-ODP is a framework, which describes a possibility of how to think about a system and how to structure its specification. It is not a methodology. Typically applying RM-ODP is an iterative approach, where details or parts will be filled out when the requirements evolved or better understood. But the framework can be used within almost every design process. (Lin11)

The architecture of RM-ODP "provides a complete and consistent model for the specification of system architecture design" (Tan04). The specification is structured into five viewpoints, "covering all the domains of architectural design" (ISO98a): the Enterprise, the Information, the Computational and the Engineering viewpoint, A short summary of each Viewpoint is given in Table 2; further details about the Viewpoints will be given in chapter 5.

Enterprise Viewpoint	Deals with the scope, policies and requirements for the system
Information Viewpoint	Describes the semantics of the information dealt within the system and the information processing
Computational Viewpoint	Describes the functional decomposition of the system
Engineering Viewpoint	Defines how the distribution works with the used components types
Technology Viewpoint	Specifies the actual implementation with the component instances and standards

Table 2 Viewpoints of RM ODP (Own contribution based on ISO98a)

RM ODP was chosen as foundation for this report because of several reasons, which are explained in the following.

The single viewpoints are well defined in a formal way using a set of formal concepts as foundation and specific language for each viewpoint on top of them. The connections between the viewpoints are defined and there is not much overlapping between the viewpoints. These characteristics make RM-ODP be a good foundation for a modeling approach. (ISO98a)

Anther reason for this selection is the domain of open distributed systems, which required the provision of concepts to deal with the high variety of vendors and the heterogeneity of components. The framework is also already used in the current deliverables of two pilot cases introduced in chapter 2. This enables the integration of the experiences already made within the two projects. For each project the challenges are described in chapter 2.1.2 and 2.2.2. The particular use of RM-ODP in the two projects in illustrated in chapter 4.3.1 for PEIS and 4.3.2 for Oil Spill.

Since the RM-ODP is an ISO standard it will remain stable. Because of the international proven processes needed for changes, partial changes by some individuals or private groups are not possible. (Lin11)

2.4.3 Usage of ODP

The concepts of the enterprise architecture framework RM ODP addresses the problems defined above. The framework provides concepts of how they can be solved, but does not provide a specific set of techniques for the application of them. In the report the framework is used as a foundation and the model-driven approach will be built upon it. Therefore in the following the relations between the identified problems in open distributed systems and the five viewpoints of RM ODP are illustrated. Figure 13 describes how the five viewpoints of RM ODP relate to the problems occurring during the specification of an open distributed system.

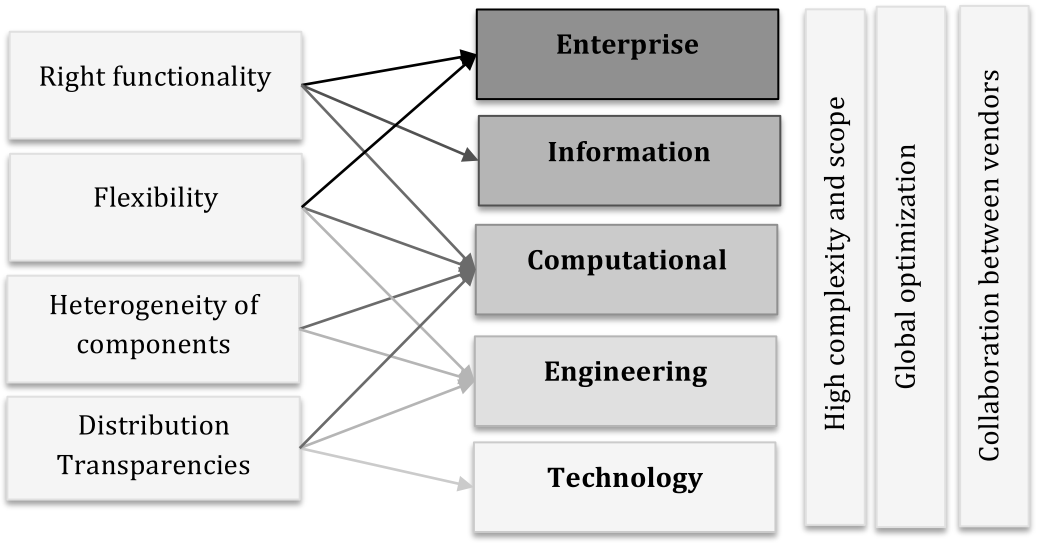

Right functionality
Flexibility
Heterogeneity of components
Distribution Transparencies
Enterprise
Information
Computational
Engineering
Technology
High complexity and scope
Global optimization
Collaboration between vendors

Figure 13 Viewpoint of RM ODP and their mapping to the problems

Providing the right functionality is concerned in the Enterprise, Information and Computational Viewpoint. In the Enterprise Viewpoint the “What” is described in terms of user requirements, in the Computational Viewpoint the “How” is described in terms of functional components collaborating together. In the Information Viewpoint the information required for the processing is described, this includes the user input information and output information. (ISO98a, ISO10b)

To gain an open distributed system with a flexible architecture it is important to separate concerns and keep them as independently as possible from each other. This is most important in the Enterprise, Computational and Engineering Viewpoints, since here the what, the how and the type of realization are specified.

The heterogeneity of the various components participating in the open distributed system is dealt within the Computational and Engineering Viewpoint. In the Computational Viewpoint the components are defined focused on the functionality they provide and require. Technical details about how the functionality is provided and how the collaboration between the components will take place are hidden in this viewpoint. The Engineering Viewpoint will handle these topics. (ISO98a, ISO10b)

A similar concern is the distribution transparencies. But here the focus lies not on the communication techniques it is about how the distribution will be realized. In the Computational Viewpoint the functional components are defined, the Engineering Viewpoint describes how they will distribute in the system using specific components types. The Technology Viewpoint defines then the infrastructure required to realize the distribution. (ISO98a, ISO10b)

At least there are also problems, which are concerned with the overall framework. The high complexity and scope of the system enhance the need for the usage of a framework, which provides a structured approach to describe the system. RM ODP provides such a structure with its five viewpoints Enterprise, Information, Computational, Engineering and Technology. Each viewpoint hast its own language describing the content it contains. The relationships between the viewpoints are also described in the RM ODP specification. (ISO98a, ISO10b)

This provides a good foundation for the global optimization problem. The defined relationships between the viewpoints as well as the foundation on one set of object modeling concepts enable the description of architectures with coherent connected elements. On top of this overall architecture-wide optimization strategies can be defined. (ISO10a, ISO10b)

The vendors of the functionality provided in an open distributed system have to collaborate through the whole developments and maintaining process. Therefore all viewpoints are concerned with this problem. The vendors have to agree about the requirements and scope of the system. For a good collaboration it is recommended to agree on one information model used in the overall system. In the computational viewpoint the vendors have to agree about how the required functionality is provided. And in the engineering and technology viewpoint they must work together to specify how they will handle the distribution and how the required infrastructure is provided. (ISO98a)

In this chapter the pilot cases were introduced. Based on the experiences in these projects and the general characteristics of open distributed systems the problems occurring during the development of such systems were identified. In the end it is illustrated how the concepts of the enterprise framework RM ODP provide solutions for these problems.

3 Requirements for a model-based approach

In the previous chapter the pilot cases were introduced. Based on the experiences in these pilots and the generic characteristics of open distributed systems seven problems were identified. This previous analysis will be the foundation for deriving the requirements for a model driven approach. The requirements are documented based on the five viewpoints of RM ODP, but there are also requirement concerning the overall approach.

At the end of each sub chapter a summary of the requirements in form of a table is shown. These tables are used as foundation for the analysis of existing solutions in the next chapter and for the evaluation of MODEA in chapter 7.

3.1 Requirements concerning the overall approach

Open distributed systems typically have a broad scope. "As information systems and technologies grow in complexity and scope, the need for a coherent and comprehensive modeling approach becomes of paramount importance" (Lei06). A common framework applied with a shared set of modeling techniques, which is used by all participants provides support for the problems occurring during the development of an open distributed systems. Such a common framework should be based on formal specified modeling techniques. The established diagrams, which are easy to understand but also powerful in their expressions, are structured using different viewpoints as defined in an enterprise architecture framework. The use of standards as well as a sound tool support makes the approach complete. How the necessary parts of the approach are related to the problems defined in the previous chapter is shown in Figure 14.

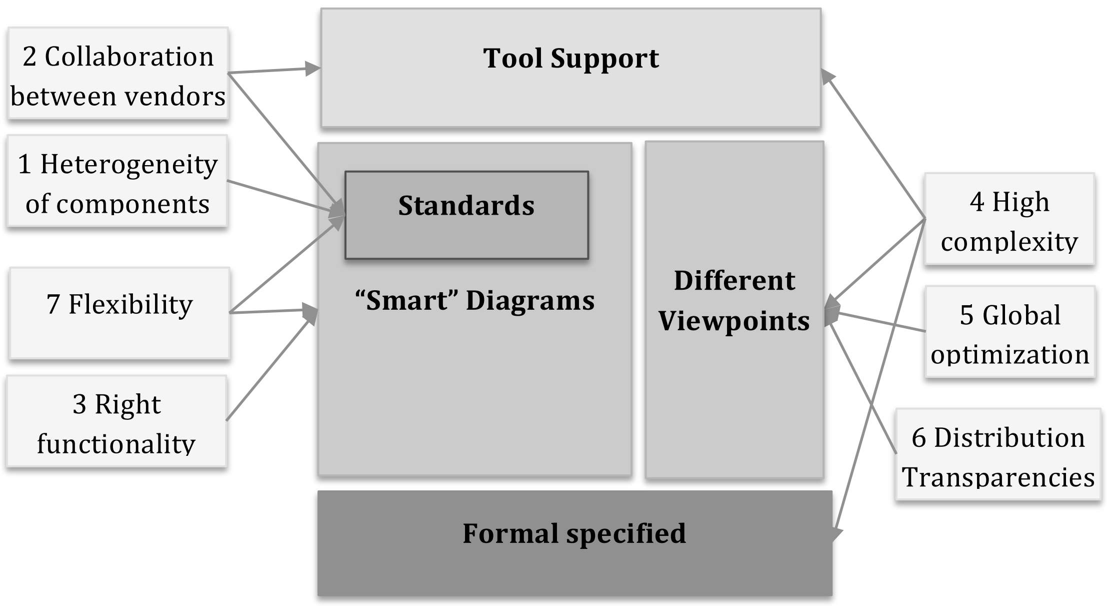


Figure 14 Relationship between the requirements concerning the overall approach and the problems

Such a tool-supported approach, fulfilling the mentioned requirements, supports in overcoming the variety of techniques and tools used by the various vendors (Lan09). In the following the requirements shown in Figure 14 are described in more detail.

Standards

To establish the system specification the information about the components from the various participants has to be integrated. Therefore the participants have to agree on one way to specify it. Using existing and standardized modeling techniques and approaches provides a stable foundation for the specification and communication (Lin11). The use of such well-defined approaches and techniques avoids misunderstandings and also work for defining an own one. Standards are also important since the specification and documentation of the components must be made available for the public. Standards enable a system to be integrated, open, flexible and modular. (Cou05, ISO98a) Following standards are one answer to deal with the problems of heterogeneity (Problem 1), providing an efficient communication and collaboration environment (Problem2) and enabling a flexible architecture (Problem 7).

"Smart" Diagrams

The whole architecture of the open distributed system should be kept as simple as possible, although the system structure can get very complex. "An EA needs to be simple enough for everybody to understand and get the gist of which system connects to which system, where applications reside, and how data and control flow through the system." (Kai05)

The simpler the architecture and the more understandable it is, the less failures due to misunderstandings occur and it is more likely to provide the right functionality (Problem 3). Simple architectures can be achieved through establishing "smart" diagrams. "Smart" diagrams are easy to understand for the user, but keep powerful in their expression. Well-known and well-established techniques are supporting the creation of good and pragmatic diagrams. Since the semantics of those diagrams are often already known, the user can concentrate on the content.

Different Viewpoints

During the specification and design of open distributed systems one problem is the high complexity (Problem 4). "One common way to cope with this complexity is by dividing the design activity according to different areas of concerns". (Rom12) The introduction of abstraction layers and different viewpoints enables the focus on the main issues of a diagram and hide the irrelevant details. Different viewpoints enable also taking into account that "different stakeholders require different perspectives" (Kai05)

Enterprise Architecture Frameworks identify different architectural viewpoints to structure the architecture description. Each viewpoint deals with one particular aspect or domain of the enterprise (Lan09). Thereby it is important to interconnect the model elements in the various established diagrams. For example a business owner is interested in how his requirements are met in the system. (Kai05) Therefore the modeled requirements have to be connected to Use Cases and these have to be mapped to Components, which implement the required functionality. This is possible if a coherent and comprehensive modeling approach is used with explicit defined relationships between the artifacts (Kai05, Lei06).

"To keep the enterprise architecture coherent, the relations between the different types of architecture must be clear, and a change should be carried through methodically in all architectures." (Ste05)

Enterprise Architectures take into account all activities of an enterprise through combination and relation of different architectures. Each architecture describes a particular aspect of the organization. Also the connections between the elements of the different architectures should be well described. Therewith they support the alignment of all the various architectures, for example they relate the strategy to the business processes and to IT resources. Such an alignment is necessary to optimize the architecture with respect to the goals and also infrastructure restrictions (Problem 5). The structure provided Enterprise Architecture Framework also supports the management of the complexity of open distributed systems (Problem 4). The goal is to create a big picture of all domains in an enterprise. (Ste05)

Modeling languages are an essential instrument for description and communication of architectures, but mostly they provide only concepts to model a specific domain. An "overarching set of design rules governing the structuring of the various architectures" is missing (Ste05). Such a concept would be essential for addressing the problem of Business and IT-Alignment (Problem 3) and also the optimization of the overall architecture (Problem 5) (Ste05, Lan09). For this reason only the use of modeling techniques does not bring the expected benefits, the model elements in the various diagrams required to specify the system, should have define relationships between each other.

Using different viewpoints to describe the architecture enables also the specification of the of distribution transparencies (Problem 6). The idea of transparencies is more or less the same as that one from viewpoints. Both want to abstract from specific details to describe a specific part of the system. This issue is further described in chapter 3.2.3, when describing specific requirements for the Computational and Engineering Viewpoint.

Tool Support and formal specification

It is important that the enterprise architecture specification is kept consistent during the whole development process. This can be supported with an adequate management of the artifacts using modeling tools. These tools should provide functionality for team support, especially change and version management, and also for establishing the models in a comfortable and user-friendly way. To keep the models consistent "a change should be carried through methodically in all architectures" (Ste05). Thereby the use of tools addresses problem 2 and 4 with providing functionality for an effective communication and collaboration between the vendors and also for managing the complexity of the system due to changes.

Following it is important to choose modeling techniques, where there is already a good tool support for the single diagrams available. Furthermore the modeling techniques must allow the specification of relationships between the model elements of the different diagrams to be able to keep the overall specification consistent. (Problem 4) A good tool should also enable tracing between the various viewpoints. For example it is important to be able to trace use cases to requirements and to the components, which will perform the functionality specified in the use case.

In the best case the tool also enables model-to-model transformations and model-to-code transformation to support the development process. For such a transformation it is necessary, that the used modeling techniques and the used methodology are specified in a formal way.

The requirements concerning the modeling techniques and their related problems are summarized in the following table.

	Requirements	Prob
1	**Use of different Viewpoints**	3,5,6

1.1	Relationships between model elements of one viewpoint	
1.2	Relationships between viewpoints	
2	**“Smart” Diagrams**	3,7
2.1	Well-known and well-established	
2.2	Readability	
3	**Use of existing standards**	1,2,7
3.1	Existing standards for modeling techniques	
3.2	Existing standardized enterprise architecture frameworks	
4	**Formal specified modeling techniques**	4
5	**Tool support for modeling techniques**	2,4
6	**Tool support for model transformation, code generation**	2,4

Table 3 Requirements dealing with the used modeling techniques

3.2 Requirements concerning the single viewpoints

In the following the requirements concerning one or several viewpoints are specified. Eight requirements are identified.

Figure 15 shows how the requirements were met in the RM ODP framework.

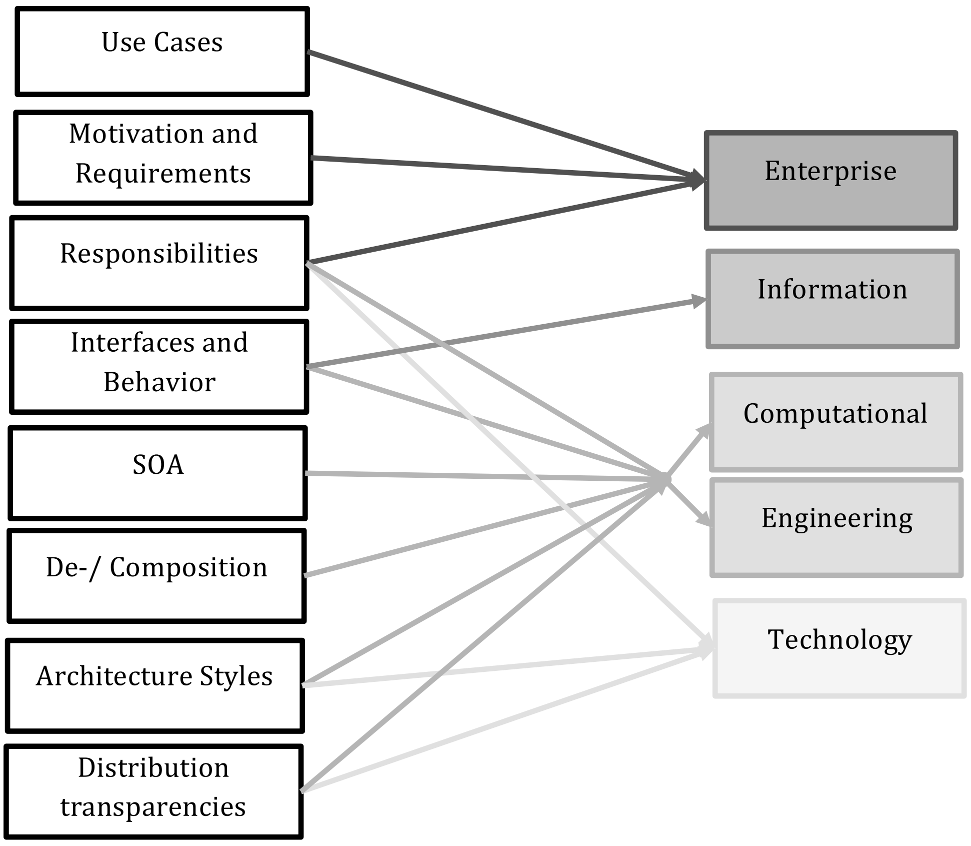


Figure 15 Requirements concerning the single viewpoints

The support of Use Cases, the definition of motivation and requirements and the assignment of responsibilities are met in the Enterprise Viewpoint. The assignment of responsibilities is also required in the Computational, Engineering and Technology Viewpoint. The specification of interfaces and behavior provides requirements to the Information Viewpoint but also to the Computational and Engineering Viewpoint. The use of a service-oriented architecture, the possibility to specify composition and decomposition, the integration of different architecture styles and patterns as well as support for distribution transparencies are all also relevant in the Computational and Engineering Viewpoint. The last two ones, architecture styles and distribution transparencies, have to be met in the technology viewpoint too.

In the following the single requirements are further detailed and their relation to the problems is shown. Since most of the requirements concerning the Computational Viewpoint are also related to the Engineering Viewpoint, these two viewpoints will be examined together.

3.2.1 Enterprise Viewpoint

Use Cases are a "popular and widely used technique for capturing and describing the functional requirements of a software system" (And01). They also support the communication with the stakeholders, especially the later users of the software system. They support the development process but are also simple enough to be understood by the users. Therefore use cases support the alignment between the user requirements and the system implementation with providing a shared communication foundation between the various stakeholders (And01). Also with the spreading of agile software engineering methods and their concept of Backlogs and User Stories, which is very similar to Use Cases, the need for explicit support of Use Cases in a coherent modeling approach becomes more and more important. Due to their characteristics Use Cases address problem 2, effective communication environment, and problem 3, providing the right functionality.

To ensure that the system provides the right functionality the specification of the motivation and of the requirement is important (Problem 3). "Requirements modeling help to understand, structure, and anlayse the way business requirements are related to information technology requirements, and vice-versa, thereby facilitating the business-IT alignment" (Eng11). Often the consideration of the motivation for the system, i.e. the goals and requirements, is a missing topic in the specification of distributed systems. (Eng11)

It is important that a modeling approach supports the specification and integration of goals and requirements and includes a possibility to see how they are realized in the system. Especially this is important to ensure a Business IT Alignment, one of the major challenges in the development process of an IT system (Problem 3). "Business process owners need to see how their requirements are met across multiple information systems" (Kai05). The results of an analysis of the motivation and requirements should be connected to elements of the application architecture. (Ber08)

In open distributed systems it is important "that it be possible to assign responsibility for any failure to meet the system's specifications." (ISO98a). It should be clear at any point in time which participating party is responsible for which part. Therefore also in the Enterprise Viewpoint it is important to be able to specify who is responsible for achieving specific goals or requirements. Without such an assignment, nobody will be responsible for reaching a specific goal with the result that nobody cares about it. Especially in the case of failures this becomes relevant.

The following table provides an overview of the requirements, which will be met in the Enterprise Viewpoint with the problems they relate to.

	Requirements	Prob
7	**Assignment of responsibilities**	3,4
8	**Integrate motivation and requirements**	3
9	**Support Use Cases**	2, 3

Table 4 Requirements concerning the Enterprise Viewpoint

3.2.2 Information Viewpoint

In an open distributed system there are a lot of participants interacting together as well as components processing the data. To ensure that the system provides the expected functionality it is important that there is a "common understanding of the information they communicate when they interact" (ISO98a). This avoids failures due to misunderstandings of single information items. A shared language in an open distributed system supports the communication and collaboration between the participants (Problem 2). Participants in this context include human actors, organizations as well as processing components.

In such a common set of information items, the information required and provided by the processing components as well as by participating users is defined. It also supports the understanding of how data entities are utilized by business functions, processes, and services" (TOGAF9) The agreement on one set of information items supports capturing the heterogeneity of the components and also the origin from different domains. When dealing with an interface of a component it is clear what information is exactly required and provided (Problem 1). Misunderstandings can be avoided when using a shared language for the information items.

The requirements, which should be met in the Information Viewpoint, are summarized in Table 5. The interface and behavior specification will be further examined in the next chapter and will be listed with a number there.

	Requirements	Prob
10	**Set up a system-wide set of vocabulary**	2
	Interface and behavior specification of components	1

Table 5 Requirements concerning with the Information Viewpoint

3.2.3 Computational and Engineering Viewpoint

Architectural styles and patterns

Especially in open distributed system there exists often several architecture styles. They are caused through the variety of components and subsystems participating in the system (Problem 1). To capture the variety of them the modeling approach has to support different architectural styles and also make the difference between them visible. Especially in the geospatial domain there must be a lot of data sources handled, since "different styles fit better for different user and system requirements" (Cen12). The use of the resource oriented architectural style is very common here. But for processing or composition services often a standard web service with message passing is used. The idea of a system delivering its functionality as services to other system also supports the integration of various architectural styles during implementation.

Different styles should also be supported in the domain-specific or general context. These generic solutions for such problems are called design patterns (Erl08). A modeling approach should support the usage of patterns on all levels.

Composition, Decomposition and Interface and Behavior specification

An open distributed system is characterized by the fact that "it is made up of components that may be obtained from different vendors" (Cro96). The components are then integrated and composed together to a whole system. When using the composition concept there are a lot of single components using different technologies to manage (Problem 1). To capture the occurring problems several issues have to be considered:

- Parts of an open distributed system can be purchased independently from each other. Following it is "very important that the behaviours of the different parts of a system be clearly defined" (ISO98a). Other parts of the system must be able to rely on the specification of the behavior, since there is no central control unit in distributed systems. A careful test and validation phase helps ensuring that the components fulfill their specifications. (Cou05, Cro96) There must also be a possibility to define responsibilities for modules and their functionality, especially for the case of failures (ISO98a)
- Especially in open distributed systems it is important that the components have published interfaces and their functionality is accessible over these interfaces. The interfaces and protocol used in the components and for communication between them have to be explicit defined (Cro96).

"Currently, developers widely use the terms 'orchestration' and 'choreography' to describe business interaction protocols that coordinate and control collaborating services." (Pap07). Both should be supported in a modeling approach for open distributed systems. The Choreography describes the collaboration between the services from a global point of view. The message exchange, rules of interactions and agreements between the multiple parties are defined. The internal action of the parties is not described and all services are treated equally. (Pap07, Bar06)

An Orchestration defines how services interact at the message level describing the business logic and execution order of the interactions. The result is a executable process from the perspective of one participating party which typically control the process-interactions. (Pap07, Bar06)

A system can also be specified the other way round, through a step-wise refinement of a system into modules. This decomposition concept of a huge system into smaller modules is an often-used concept to capture the complexity of the system structure and to keep the architecture of the open distributed system as simple as possible (Problem 4). (Cro96)

A modeling approach for open distributed systems should support both the decomposition and composition concept. For an adequate specification of them there also have to be techniques defined to specify the interface and behavior of the components as well as the interaction between them.

Service Orientation

The service-oriented style addresses several of the problems occurring in open distributed system. Above all it enables the establishment of a flexible architecture (Problem 7). It also supports in capturing the heterogeneity of the various components (Problem 1) and provides a well-understood concept, the service concept, which is understood by nearly all participants (Problem 2). The service concept tackles the alignment problem between the different architectures, which is a main factor for providing the required functionality (Problem 3). At least the integration of services enables a possibility for linking different architectures. Such a concept of linking the different architecture is necessary to tackle problem 5, the overall optimization of the enterprise architecture. The way a service-oriented style will deal with these problems is examined in the following paragraphs.

Services are characterized by an internal and external behavior. They are "self-contained and have a clear purpose from the perspective of its environment" (Ste05). For the consumer the internal behavior is irrelevant, he is only interested in the functionality and quality provided. There is only little ongoing effort to revise methods for enterprise architecture in the light of service orientation (Ste05).

Furthermore service oriented architectures "enables new and existing enterprise systems to share services, information and data across technical platforms, departments and ultimately across organizational and regional boundaries" (Cen12). Therewith it is an important concept when it comes to manage the heterogeneity of the various components in an open distributed system.

Service Orientation is a set of principles, which support to bridge the business and IT worlds in the context of enterprise architectures. It provides concepts to comprehensively describe business-critical functionalities of enterprise IT-System, but also efficient ways to cope with integration and interoperability problems. Therewith the major challenge to align business requirements and the IT system implementation can be addressed, which supports in providing the required functionality. (Sad10, Ste05).

When establishing enterprise architectures, there are a lot of stakeholders from different domains participating in the process. Services are a well-understood concept in the different domains making up enterprise architectures. SOA "views each system or business as a collection of service providers" (Kho09). This approach is very similar to what is practically used in business and organizations. Currently there is an ongoing concentration on services in organizations, i.e. delivering services as products to the customer. Services provide a common and understandable language for Business and IT, which facilitates the communication between these two main domains included in enterprise architectures. (Kho09, Ste05).

Despite the service-concept as common language between the collaborators, it also enables an adequate separation of concerns. Through the agreement on service contracts between two parties smaller projects can be established and also be assigned to different development groups. (Kho09)

For support of the interoperability of the components and to gain a flexible architecture (Problem 7) a service oriented architectural style should be supported by the modeling approach. Khoshnevis et al. (2009) stated that the "Independence of services from each other and from technology helps having higher degrees of reusability and flexibility" (Kho09). Following the organization is able to take advantage of reuse, substitution and recombination possibilities when using the service concept.

A high interoperability requires a precise service description and service protocol and also a loose coupling between the components. Therefore it must be clearly differentiated between the external and internal behavior and techniques supporting this have to be proposed. (Ste05, Kho09)

Only enabling a high interoperability does not conclude in a high rate of reuse. Therefore concepts for identification of required services functionalities are needed. In the further development process there must also be support for an identification of existing services for reusing them as implementation for the required one.

Introducing a classification approach as proposed in TR 15449 can support the identification of services. In the technical report a service-centric view for spatial data infrastructures is introduced. Thereby the report proposes a life cycle based approach for the classification of services. The service-centric part of this “lifecycle-based perspective for the identification of enablers” (Cen12) can be applied to any service-oriented system. The main components in the service centric view are Register, Discovery, View, Download, Invoke, Orchestration and Composition and Security and Risk Management.

Despite this approach in TR 15449 is also an architecture-based identification of services proposed. The main components here are

- Boundary Interaction
- Composition & Workflow
- Processing Services
- Data and Model Management Services
- Communication Services
- Management of metadata
- Security and Privacy.

They can be organized in a bus or a layered architecture.

These two approaches are examples for classification possibilities, which support the designer in service identification and enable the reuse of them. A grouping of services, together with a former grouping of requirements, supports the design of stable and loosely coupled services. This helps to “increase their adaptation to changes and better control their evolution on the basis of changing tactics” (Ber08). A model-driven approach for an open distributed should support such a classification possibility.

Distribution Transparencies

“Virtual enterprises are dynamic entities that seek to create transparency of services’ location”(Kai05). Following a further requirement for the modeling approach for open distributed systems is the support of distribution transparencies (Problem 6). That means that the approach has to provide mechanisms, which enable hiding of the distribution of the system but also a possibility, were the way transparencies are implemented is specified. In the computational viewpoint the system should be described without caring about the distribution of the system. The Engineering Viewpoint defines then how this abstraction from the characteristics of the distribution will take place. (ISO98a, Cou05) The several types of transparencies with its characteristics are explained in chapter 2.3.1.

Assignment of responsibilities

As already mentioned in the requirements for the Enterprise Viewpoint, the assignment of responsibilities is important in open distributed systems. In the computational viewpoint, there must be an assignment of responsibilities to the required functionality of the system. In the engineering viewpoint for each component, which will be realized, there must be a responsible actor ensuring its provision. The assignment of responsibilities becomes more important the more different vendors participate in providing the functionality of the open distributed system. (ISO98a)

Since this requirement is already listed in the requirements for the Enterprise Viewpoint, it will be an unnumbered entry in the following table (Table 6) summarizing the requirements concerning the Computational and Engineering Viewpoint.

	Requirements	Prob
11	**Specification and integration of different architectural styles and patterns**	1
12	**Support for Decomposition and Composition**	4, 1
12.1	Support for Choreography and Orchestration	
12.2	Interface and behavior specification of components	
13	**Support a service oriented architectural style**	1,2,3,5,7
13.1	Specification of services	
13.2	Identification of Services	
13.3	Reuse of services	
13.4	Classification of Services	
14	**Support specification of distribution transparencies**	6
14.1	Communication styles	
	Assignment of responsibilities	3,4

Table 6 Requirements concerning the Computational and Engineering Viewpoint

3.2.4 Technology Viewpoint

All the requirements concerning the Technology Viewpoint are already mentioned in the chapter defining the requirements for the computational and engineering viewpoint. For this reason only a short comment how these requirements are affecting the Technology Viewpoint specification is given.

Also in the Technology Viewpoint, there is an assignment of responsibilities required. Each element of the technical infrastructure in the system has to be managed by a specific actor.

Architectural styles and patterns play a role in the technology viewpoint too. Each architectural style used in the Computational and Engineering Viewpoint derives requirements for the Technology Viewpoint. Their realization as well as their link to the Engineering/Computational Viewpoint should be made visible in the modeling approach. But also the other way round, the existing infrastructure provides requirements for the usage of architecture styles and patterns. For example patterns can encompass the parallelization of operations of the scalability of the system. The modeling approach should be able to adapt such patterns, when specifying the technology viewpoint.

The use of Distribution Transparencies has also influences on the Technology Viewpoint. Here the infrastructure of the distributed system is defined from whom the transparencies should abstract.

Table 7 gives an overview over the requirements concerning also the Technology Viewpoint.

Requirements	Prob
Assignment of responsibilities	3,4
Support for different architectural styles and patterns	1
Support specification of distribution transparencies	6

Table 7 Requirements concerning the Technology Viewpoint

4 Existing Solutions

4.1 Modeling approaches relating to Enterprise Architecture Frameworks

There are several Architecture Frameworks in current practice and theory. The most known frameworks are Zachman, TOGAF and DODAF/MODAF/NAF. Despite these frameworks there exists a lot of other frameworks like the RM-ODP for open distributed systems and the FEAF/TEAF for US federal agencies and other governmental agencies. (Tan04, Lei06, Lan07)

In the following a short introduction to Zachman, TOGAF, DOFAD/MODAF and RM ODP as the most known architectural frameworks will be given and modeling approaches related them are described.

4.1.1 Zachman

The Zachman Framework is the best-known Enterprise Architecture Framework. It provides a structure for classifying and organizing the elements of an enterprise and its systems that are interesting to the management or development process. (Lan09)

In a 6x6-cell matrix the resulting artifacts from the different perspectives are defined. The columns define the five perspectives What, How, Where, Who, When and Why to describe the enterprise architecture. In the rows six classifications for the artifacts are described. These are Executive Perspective, Business Management Perspective, Architect Perspective, Engineer Perspective, Technician Perspective and Enterprise Perspective. In Figure 16 the Architecture with its classification and perspectives is shown. (Zach12).

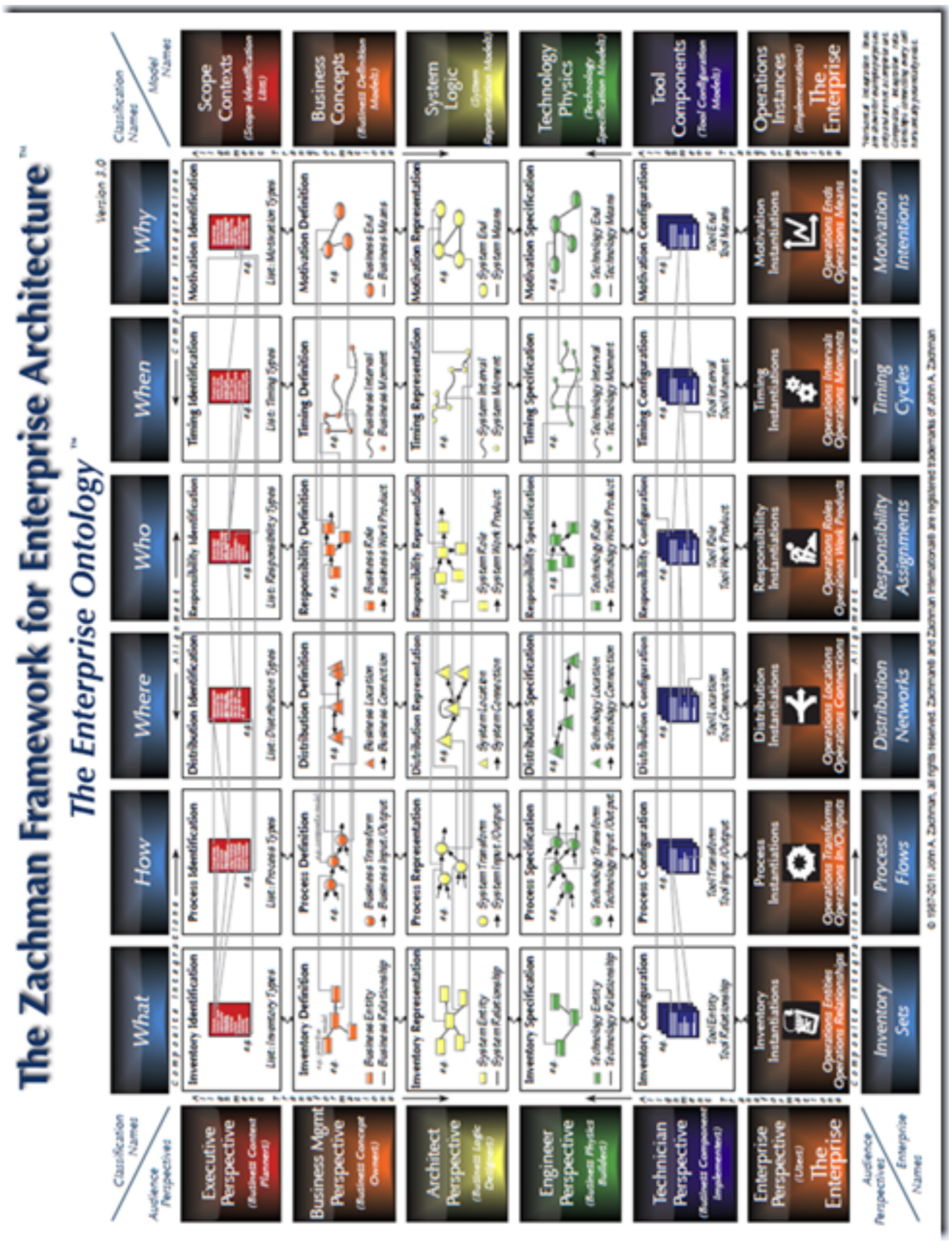


Figure 16 Zachman Framework for Enterprise Architecture, Version 3.0 (Zach12)

The Zachman framework is not a methodology, it typed itself as ontology and metamodel. Hence it doesn't provide a support for the implementation process of the enterprise architecture. The framework provides a structure to ensure that all views are well established. The concepts of the Zachman Framework are also used in frameworks like FEAF, DoDAF and TOGAG. (Zach87)

The Zachman Framework is easy to understand and provides a specification for the enterprise at a whole. It can be implemented with any tool or methodology, since its specification is made independent from this. But 36 cells is for a practical usage a large number of cells, which makes the architectural specification more difficult Also the relations between the single cells are not that well specified in the framework. (Lan09)

4.1.2 ArchiMate and TOGAF

TOGAF is a standardized architecture framework from the Open Group that "provides the methods and tools for assisting in the acceptance, production, use, and maintenance of an enterprise architecture." (Open11) The framework is a quasi industry standard which supports the design, evaluation and building of the right architecture for any organization. TOGAF contains an architectural framework but also an Architecture Development Method (ADM). The ADM is a a "reliable, proven approach for developing enterprise architecture descriptions that meets the needs of the specific business" (Lei06). This approach includes:

- establish an architecture framework,
- develop the architecture content,
- transitioning and governing the realization of the architecture. (Open11)

The ADM describes how to derive organization-specific enterprise architecture from the TOGAF architecture framework that addresses the business requirements. The Architecture Development Cycle describing the ADM with its single steps is shown in Figure 17.

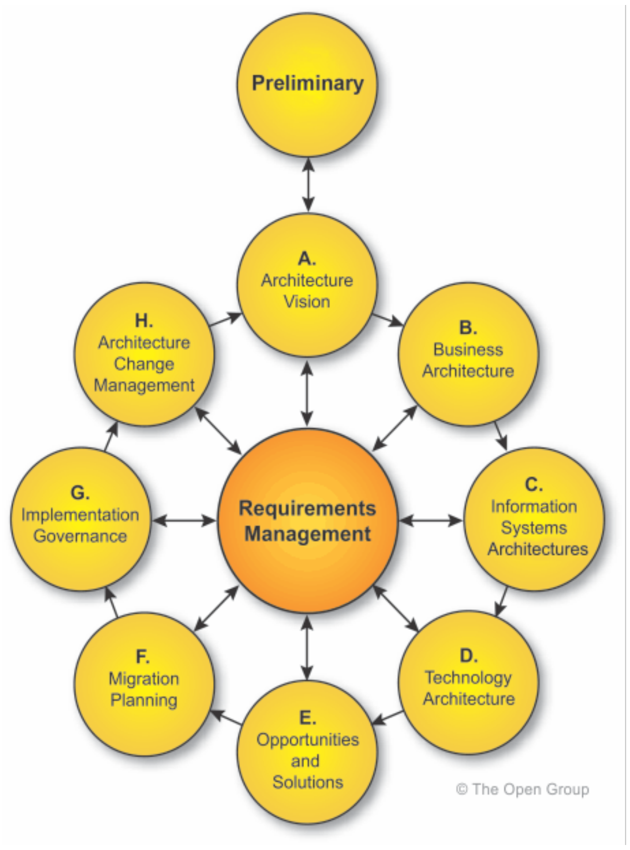

Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management
© The Open Group

Figure 17 TOGAF Architecture Development Method Cycle (Open11)

The steps B, C, D in the ADM (Figure 17) deal with the architecture domains. They are the Business Architecture, the Information System Architecture consisting of the Data Architecture and the Application Architecture and the Technology Architecture.

- The Business Architecture deals with the Business strategy, the governance, the organization, and the key business processes.
- The Data Architecture describes the logical and physical data assets as well as the data management resources.
- In the Application Architecture the individual applications systems for deployment are designed as well as their interactions and relationships to the core business processes.
- At least the Technology Architecture describes logical software and hardware capabilities that are required for the deployment of business, data, and application services. The architecture includes the IT infrastructure, use of middleware, networks, communications and standards. (Open11, Lan07)

ArchiMate is a modeling approach for TOGAF and also standardized by the Open Group. It's goal is to "provide a graphical language for the representation of enterprise architectures over time" (Open12a). Additionally is a way to capture the domain-based language barriers with only using a single set of icons. The current version was published in January 2012 (Version 2). The most important concepts in this version are illustrated in Figure 18.

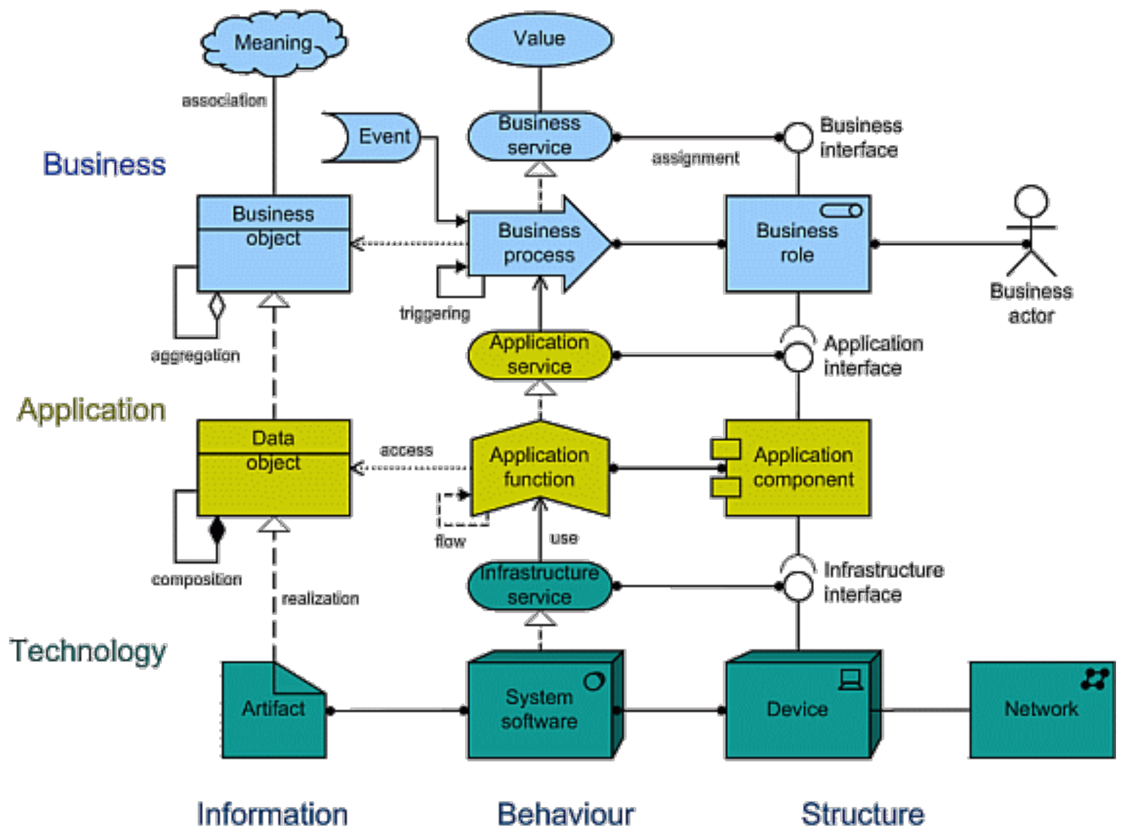


Figure 18 Main Concepts of ArchiMate Version 2 (Open12b)

The architectural description is divided in three layers. They are Business, Application and Technology. Services and Interfaces are the linking objects between the layers. The idea is, that functionality is provided through services to the upper layer. In each layer the approach "distinguish[s] between the structural or static aspect and the behavioural or dynamic aspect" (Lan07) Structural aspects are further refined in active and passive ones. The last ones are also referred as Information. The three layers can be further extended through the Motivation Extension or through the Implementation and Migration Extension. (Open12a)

Examples for tool support for ArchiMate are the BiZZdesign Architect, Archi (ArchiMate Modeling Tool) and the Enterprise Architect from Sparx Systems. (Rom12)

ArchiMate is a well-connected and well-understandable modeling approach. But this language is provides only support for a visualization of an enterprise architecture. It is not intended to use as language for a model driven development with model and code generation.

4.1.3 UPDM and DoDAF/MODAF/NAF

There exist three architecture frameworks, which are specific for the military domain and very close to each other. The Department of Defence Architectural Framework (DoDAF), the Ministry of Defence Architectural Framework (MODAF) and the NATO Architecture Framework (NAF). MODAF was established based on DoDAF. NAF was built upon MODAF and there are only minor differences between the two frameworks. (OMG12b) Some processes and taxonomies are domain independent, but the frameworks also contain elements specific for defense operations and. (Tan04) The goal of them is enable the description of interrelated architectures and realization of system that can interoperate (Lei06).

Table 8 shows the viewpoints of the current version of DoDAF 2.0 and MODAF 1.2. It also illustrates how the MODAF Viewpoints relate to the DoDAF Viewpoints.

DoDAF 2.0	MODAF 1.2
Capability Viewpoint	Strategic Viewpoint (Service Oriented Viewpoint)
Operational Viewpoint	Operational Viewpoint
Services Viewpoint	Service oriented Viewpoint
Systems Viewpoint	Systems Viewpoint
All Viewpoint	All Viewpoint
Data and Information Viewpoint	Parts from Operational Viewpoint/ System Viewpoint
Standards Viewpoint	Technical Viewpoint
Project Viewpoint	Acquisition Viewpoint

Table 8 Viewpoints in DoDAF 2.0 and MODAF 1.2 (Own contribution based on OMG12b)

The concepts of the Capability Viewpoint in DoDAF can be found in the Strategic Viewpoint of MODAF but also parts of them in the Service Oriented Viewpoint. The Operational Viewpoint, the Systems Viewpoint and the All Viewpoint contain mainly the same concepts in both frameworks. The concept of services in MODAF and DoDAF used in the Service Oriented Viewpoint (MODAF) and Services Viewpoint (Systems Viewpoint) differ significantly. The Data Information Viewpoint in DoDAF is realized within the Operational and System Viewpoint in MODAF. The concepts of the Standards Viewpoint in DoDAF can be found in the Technical Viewpoint in MODAF and the Project Viewpoint in DoDAF corresponds to Acquisition Viewpoint in MODAF. Further details about the relationships between the two frameworks can be found in the UPDM specification. (OMG12b)

The Unified profile for DoDAF/MODAF (UPDM) is an initiative for the development of a modeling standard that support both Enterprise Architecture Frameworks. In the new release in January 2012 (Version 2) UPDM also supports NAF. The goal of UPDM is “to specify a UML 2, and optional SysML, profile to enable practitioners to express DoDAF and MODAF model elements and organize them in a set of specified viewpoints” (OMG12b)

UPDM is provided as a UML Profile based on the UML, SysML and SoaML specifications. Therewith it includes a service oriented component modeling and a SysML system modeling to represent the DoDAF and MODAF architecture views. The profile defines a set of elements and the relationships between these elements and also a number of views and viewpoints. These concepts are primarily used to support the development of Enterprise Architecture in the Military domain (OMG12b).

Tool support for UPDM is for example provided in Artisan, Enterprise Architect and Rational System Architecture. (Rom12)

4.1.4 UML4ODP and RM-ODP

A short introduction to RM-ODP was already given in chapter 2.4.2. There is also an ISO standard, which defines a UML profile to model the five viewpoints of RM ODP. This approach is called UML4ODP and standardized in ISO/IEC 19793 (ISO09). The standard encompass "the expression of a system specification in terms of RM-ODP viewpoint specifications using defined UML concepts and extensions" (ISO09)" and the "relationships between the resultant RM-ODP viewpoint specifications" (ISO09). UML4ODP is an UML profile realizing the concepts of RM ODP.

The five viewpoints are described as packages structured through sub-packages and containing the corresponding model elements and diagrams for this viewpoint. Figure 19 shows such a package for the Information Viewpoint of a specification from a Library System.

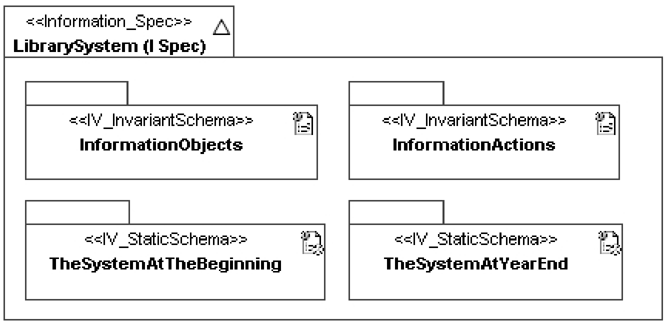


Figure 19 Example Structure of the Information Viewpoint Specification (ISO09)

Each package contains in this case a diagram (like in the InformationObjects package) or the specification of model types (like Action Types in the InformationActions package).

A lot of diagrams are very similar to corresponding UML diagrams but with use of ODP special stereotypes. An example for these diagrams is the "TheSystemAtBeginning" diagram. It describes the initial state of the library system with a static schema of the information objects. This diagram is mainly an UML Object Diagram with the use of *<<IV_Object>>* stereotypes for the objects. (ISO09)

Another diagram used in the Enterprise Viewpoint to further define the interaction between two roles is the one in Figure 20.

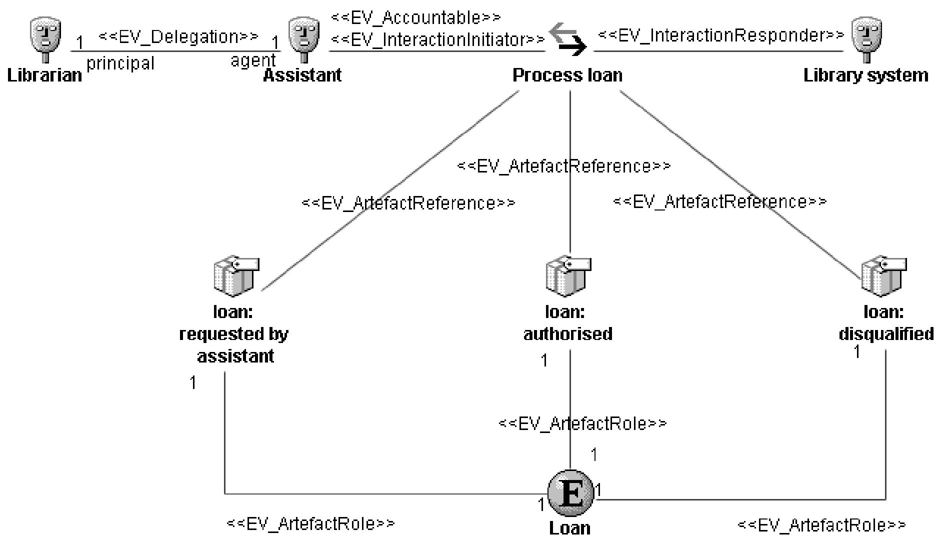


Figure 20 Process loan interaction (ISO09)

This diagrams shows that the Assistant initiates the Process loan interaction and the Library system responds to it. Three signals, stereotypes with *<<EV_Artefact>>* express the artifact roles of Loan involved in the interaction. (ISO09)

UML4ODP is a very formal specified modeling approach since it is defined as UML profile and built completely upon the formal RM-ODP specification. But this concludes also in over-weighted diagrams at some points. To capture all the concepts of RM ODP the diagrams differ a lot from the original UML diagram. Therewith they are more difficult to understand and also to establish.

4.2 Other service-oriented modeling approaches

One major requirement for the modeling approach is the use of a service oriented architectural style. This year the SoaML specification of the OMG group (OMG12a) was published as a standard for modeling services and there already exist modeling approaches using SoaML to capture the service-oriented paradigm in a software specification. The approaches are not integrated in any enterprise architecture framework and therefore include not all relevant viewpoints. But they give input to at least a few viewpoints.

Sadovykh et al. (2010): ES Modeling with SoaML using BMM and BPMN

Sadovykh et al. (2010) describe a Model Driven Architecture (MDA) based methodology for modeling enterprise architecture with BMM, BPMN and SoaML. In the paper they "present[…] [their] practical experience with SoaML in attempt to align business models and enterprise IT systems implementation" (Sad10).

MDA specifies three levels a model passes in the development cycle:

- Computational Independent Model
- Platform Independent Model
- Platform Specific Model

According to MDA Sadovykh et al. define a Business Architecture Model (BAM), a System Architecture Model (SAM) and the Implementation level. Figure 21 shows the levels in the approach with their concepts.

For the BAM the modeling languages BMM and SysML Requirements (Business Scope), UML Class Diagrams (Information Semantics), BPMN (Business Processes) and SoaML (Capabilities, Service Contracts, Choreographies and Architectures) are used.

For SAM mainly SoaML is used to establish the models. The implementation models skeletons can be generated from the SAM SoaML models. In their research project they developed transformations from the SAM SoaML models to various implementation platforms. They are for example Web Services, Multi-Agent Systems and Semantic Web Services platforms.

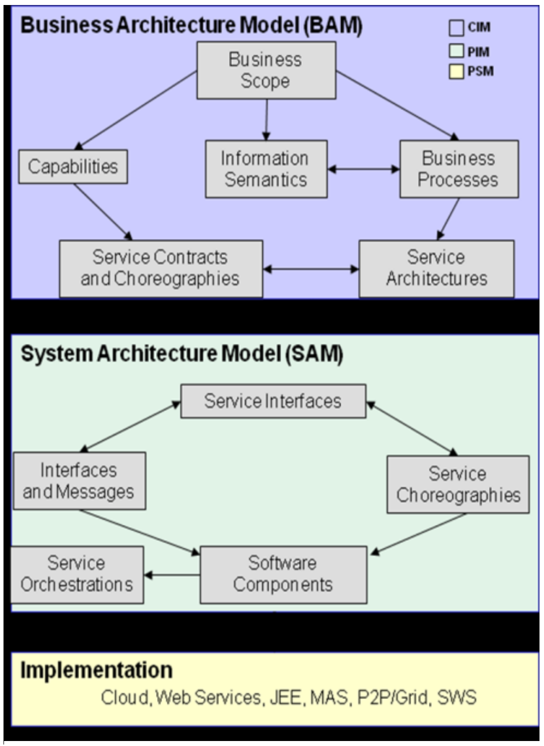


Figure 21 Enterprise Architecture Modeling Levels according to (Sad10)

Berkem (2008): Goal-Driven SOA

Berkem (2008) propose a possibility of how to link the elements of the Business Motivation Model BMM to the components of a service oriented architecture. Therewith he wants to enable the alignment of the IT according to the goals and directives. Applying the "layered architecture of the 'Goal-Driven SOA' Process permits to identify traceability relationships that permits to connect such high level business goals and directives toward elements of the software level specifications" (Ber08).

The important concepts in BMM for a bridging to SOA are the business goals, the courses of actions (strategy and tactics), directives (rules and policies) and Business Processes.

Berkem (2008) identifies three necessary layers for a mapping. These layers with its main concepts are shown in Figure 22. In addition he also mentions a Deployment layer, dealing with the Where-question.

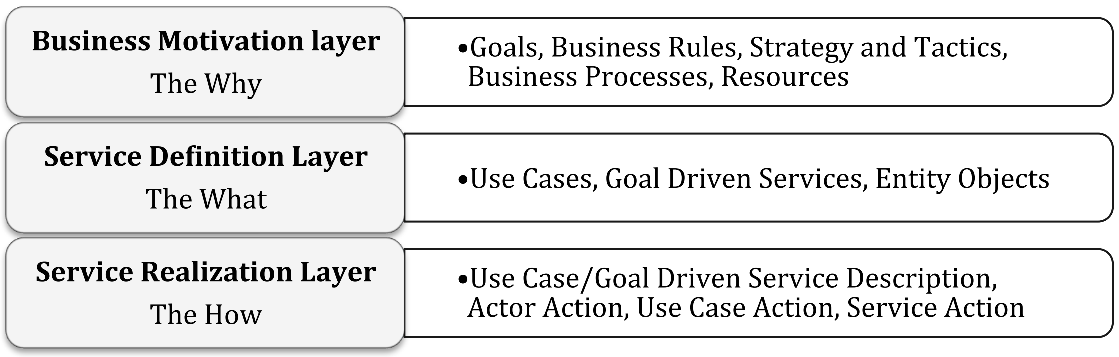

Business Motivation layer
The Why
•Goals, Business Rules, Strategy and Tactics, Business Processes, Resources
Service Definition Layer
The What
•Use Cases, Goal Driven Services, Entity Objects
Service Realization Layer
The How
•Use Case/Goal Driven Service Description, Actor Action, Use Case Action, Service Action

Figure 22 Layers in SOA Framework from (Own contribution based on Ber08)

The Tactics and Business rules in the Business Motivation layer are traced towards the Service Definition layer to drive services and derive requirements for them. A goal-driven-service is defined for each Business Process. Tactics act as a façade to "drive" a set of goal-driven services. In the approach of Berkem (2008) there is a one-to-one relationship between use cases and goal-driven services. The realization layer provides a description of the Tactics, Services and Use Cases determined in the Definition layer. A Use Case or a service can be determined by an activity (composite structure) or a single action. An activity contains further activities or actions.

TR15449: Service-centric view of a Spatial Data Infrastructure

Despite these two partial approaches for enterprise architecture modeling there is also a recommendation in TR 15449, which will be further established in ISO 19154 this summer, for modeling the five viewpoint of RM ODP (Cen12). The standard describes a service-centric view of Spatial Data Infrastructure (SDI) from the perspective of the RM ODP viewpoints. SDIs can be seen as a set of interconnected, distributed, information systems. The understanding of the five viewpoints in the TR 15449 and the relevant standards defined in the report are shown in Table 9.

Enterprise Viewpoint	Service aspects from an organizational, business and user perspective (Use Cases and external functionality related to services)
	BPMN, Use Cases, Agile requirements with user stories, BMM, UML4ODP Enterprise Specification
Information Viewpoint	Service aspects from a geospatial information expert perspective (Information being used and provided by services)
	UML, XML, GML, potentially semantic technologies like OWL and Linked Open Data with RDF
Computational Viewpoint	Service aspects from a system architect perspective
	SoaML, USDL, UML4ODP Computational Specification
Engineering Viewpoint	Service aspects from a system designers perspective
	Service-oriented Architecture, REST, Web 2.0
Technology Viewpoint	Service aspects from a system builder and implementer perspective
	Cloud Computing

Table 9 Usage of the RM ODP Viewpoints in TR 19154 (Own Contribution based on Cen12)

UML4ODP is mentioned as relevant standard in the context of modeling RM ODP. But since it provides a full support for RM ODP it has been preferred to use more light weighted techniques like UML and SoaML in the SDI community. (Cen12)

4.3 Current approach in the pilot cases

4.3.1 Pilot 1: Personal Environmental Information System

The Oil Spill project has already started and there are deliverables, mainly for specification of requirements and use cases. There are also some architectural diagrams for PEIS, but they are not included in any framework.

But there are deliverables starting to define the architecture of ENVIROFI. The ENVIROFI architecture will be used as foundation for the PEIS architecture and adapted to the specific needs of the project. The architecture is described using the five RM-ODP viewpoints Enterprise, Information, Computational, Engineering, and Technology. This is done mainly textual, with tables and some Box Diagrams. The term Box Diagram is used for any diagram consisting of some boxes and lines with no specific or some self-defined syntax.

In Table 10 the existing specifications from the latest deliverables concerning the ENVIROFI architecture are shown.

	Environmental Architecture (PEIS 4.2)	**Sketch of the ENVIROFI Architecture (PEIS 6.1.1)**
Enterprise	Use Cases and Requirements	
	Template, Use Case Diagrams	

Information	General Feature Model, Service Model, Resource Model Approaches to semantic interoperability	Data models and metadata models that must be supported
	UML Class Diagram, textual	Table
Computational	Enabler Architecture, Categorization of Enablers	Description of system requirements and computational objects, Layered Architecture
	Box Diagram	Textual, Box Diagram
Engineering	Relevance of FI WARE enablers for ENVIROFI	Distributed Architecture, Interoperability agreements, Security Framework Mapping enablers with engineering objects
	Table	Box Diagram, Textural, Table
Technology	Used technologies and standards in FI WARE	Description of principles that should be applied
	Table	Textual

Table 10 Specification Approach in the ENVIROFI project (Own contribution based on PEIS 4.2, PEIS 6.1.1)

For each viewpoint the specified content is shortly described and also the used modeling techniques are named in the table.

Despite the specification documents for PEIS and the ENVIROFI architecture, there are also specifications of the generic and environmental enablers, which have to be considered. The enablers are organized in categories. Each of the environmental enablers is specified through a predefined template (PEIS 5.2.2). Each of the general enablers is specified textual and in the most cases also with FMC diagrams (FIWiki12). FMC stands for 'Fundamental Modeling Concepts' and is "primarily a consistent and coherent way to think and talk about dynamic systems" (FMC12). There are also considerations in the project of using USDL for the description of all enablers.

4.3.2 Pilot 2: Oil Spill Decision Support System

The requirements for the Oil Spill Decision Support System are already specified. Also a first description of the operational system is done. At the moment the implementation of a first pilot is in progress.

The requirements from the three case studies for the ENVISION projects are already established. The specification of the different parts of the ENVISION infrastructure is also done. The several components of the ENVISION architecture, like the Composition Portal, the Semantic Catalogue or the Execution Infrastructure, are specified separately from each other in their own work packages. At the beginning of the specification it is shown how the components are integrated in the overall ENVISION infrastructure. The following specification does not follow any methodology or framework. Whereas the requirements for the ENVISION infrastructure derived from the case studies are structured using the RM ODP viewpoints.

This approach and the specification approach for the operational system of the Oil Spill System are described in Table 11.

	Requirements specification for ENVISION (ENV 1.2)	**Operational System for Oil Spill (ENV 1.4)**
Enterprise	ENVISION and Pilot Definition, Actors and Roles, User activities	(General System Description) Composition Workflow, Service Description (Input, Output)
	Table, Use Cases, Textual, Box Diagrams, Use Case Diagrams	Box Diagram, Table, Textural
Information	Input and Output Data with types, Used Models	Objectives and operational system components to fulfill them Link to ENVISION work packages and components
	Table	
Computational	Portlets, Web Services, Libraries, Applications Collaboration between them	
	Table, Mix of UML Collaboration and Component Diagram	
Engineering	Use of Ontologies, Key activities	Textual, Table
	Activity Diagram, Component Diagram, Textual Templates with UML Interaction Diagrams	
Technology	Standards and used technologies	
	Short textual description	

Table 11 Specification approach for ENVISION and the Oil Spill project (Own contribution based on ENV 1.2, ENV 1.4)

As illustrated in the table above, the specification of the Oil Spill System is almost completely done using informal descriptions and links to other specifications. The specification of the ENVISION infrastructure, describing the requirements for it, is done very structured and with the use of modeling techniques.

4.4 Comparison and Evaluation of the Modeling approaches

In the previous chapter several modeling approaches relating to Enterprise Architecture Frameworks and Service-oriented Architectures as well as the recommendations in TR 15449 for RM ODP are presented. In this chapter an overview over all these approaches with their used modeling techniques is presented. The overview is done based on the five viewpoints of RM ODP. The layers and viewpoints of other approaches than UML4ODP are mapped to the five ones of RM ODP based on the concepts they contain.

The overview in Table 12 will be used to see how other approaches model the aspects from the Enterprise, Information, Computational, Engineering and Technology Viewpoint. This enables the definition of a modeling approach using the experiences made within the other approaches. For each approach the corresponding viewpoint or layer to the RM ODP viewpoint is mentioned and then the modeling concepts used to describe the viewpoint are shown.

Viewpoint / Approach	Enterprise	Information	Computational	Engineering	Technology
UPDM (Modea/ Dodaf)	Corresponds to Strategic Viewpoint /Operational Viewpoint	Corresponds to parts of the Operational Viewpoint	Service oriented Viewpoint and parts of System Viewpoint	Corresponds to parts of System Viewpoint	Corresponds to Technical Viewpoint (Standards profile)
(OMG12b)	Class Diagrams, Matrix, Activity Diagram	Class Diagrams, Composite Structure, State machines	Matrix, Table, Class/ Activity/Component and Composite Structure Diagram	Class/Component and Composite Structure Diagram, Matrix, Table	Table
UML4ODP (RM ODP) (ISO09)	Community Contracts with Enterprise Objects, Roles, Processes, Policies and Objectives	Invariant, static and dynamic Schema of the information objects	Software Architecture and Behavior	Distribution to processing nodes with interaction over channels	Technology objects, implementable standards, Technology processes
	Diagrams, State Machines, Hierarchies	Class Diagrams, State Machines, Object Diagram	Component Diagram: Ports and Interfaces, Interaction Diagram	Component diagrams with use relationships to channels	Class Diagram for technology objects, Activity Diagram
ArchiMate (TOGAF) (Open12a)	Corresponds to Business Architecture	Corresponds to Information Systems Architecture/Data	Information Systems Architecture/ Application	Corresponds to Technology Architecture	Corresponds to Technical Architecture
	Roles and Actors, Process, Business Service, Business Objects	UML Class Diagram	Components, Application Services, Internal Function (Process), Data Objects	System Software providing Infrastructure Services and Artifacts (like Database tables)	Devices connected through Network (at the physical layer)
Sadovykh et al. (2010)	Corresponds to Business Architecture Model	Corresponds to Business Architecture Model	Corresponds to Business Architecture Model	Corresponds to System Architecture Model	Corresponds to Implementation
	SysML Requirements, Business Processes	UML Class Diagram	SoaML Capabilities, Service Contracts and Architectures, Choreographies	Service Interfaces, Software Components, Orchestration	
Berkem (2008)	Corr. to Business Motivation, Service Definition		Corresponds to Service Definition, Service Realization		
	Business Motivation Model, Use Cases, Service		Service, Use Case Description, Actions		
Rec. in TR 15449 (Cen12)	BPMN, UML Use Case (with Templates), User Stories, BMM, UM4ODP	UML, XML, GML	SoaML, USDL	SOA (Web Services), Rest, Web 2.0	Cloud Computing

Table 12 Summary of the Modeling Approaches (Own contribution based on OMG12b, ISO09, Open12a, Sad10, Ber08, Cen12

The presented modeling approaches ArchiMate, UPDM and UML4ODP, which are built upon an Enterprise Architecture Framework, are now analyzed by the 14 requirements defined in chapter 3. For the evaluation a three-step scale is used. 1 means that there is no support of this requirement in the modeling approach. 2 means partially supported and 3 means fully supported. First the requirements concerning the overall approach are presented in Table 13.

	ArchiMate (Open12a)		**UPDM (OMG12b)**		**UML4ODP (ISO09)**	
1 Use of different Viewpoints		3		3		3
1.1 Relationships between model elements of one viewpoint	One diagram for each layer	3	Meta Models for each Viewpoint	3	Concept models, each layer one package	3
1.2 Relationships between viewpoints	All layers connected	3	Not visible in a paper form	2	Defined in specification	3
2 "Smart" Diagrams		2		2		2
2.1 Well-known and well-established	New language	1	New UML profile	2	New UML profile	2
2.2 Readability	Few concepts, same concepts in all layers	3	Lots of diagrams, easy to loose overview	2	Over weighted diagrams through stereotypes	2
3 Use of existing standards		2		2		2
3.1 Existing standards for modeling techniques	New Open group standard	1	New UML profile	2	New UML profile	2
3.2 Existing standardized enterprise architecture frameworks	TOFAF	3	DODAF, MODAF	1	RM ODP	3
4 Formal specified modeling techniques	Concept meta models for each layer	2	UML Profile, Metamodel for each layer	3	UML profile, Meta model for each layer	3
5 Tool support for modeling techniques		3		3		3
6 Tool support for model transformation, code generation	Only visualization, structuring	1	Support for UML and SysML	2	Support for UML, xODP (M2Maude)	2

Table 13 Evaluation of the requirements concerning the overall approach (Own contribution based on Open12a, OMG12b, ISO09)

ArchiMate, UPDM and UML4ODP contain all different viewpoints, which have defined relationships to each other. All the three modeling approaches for enterprise architecture frameworks create their own set of modeling techniques for the documentation. ArchiMate is built upon a complete new set of language concepts. UPDM and UML4ODP are built defining a new UML profile. No modeling approach for enterprise architectures is based only upon existing techniques through linking the concepts of them.

Especially in UPDM and UML4ODP the diagrams are not always intuitive and comprehensive for the user, but they are formally defined and very powerful in expressing the concepts of the used frameworks. With ArchiMate good understandable diagrams can be established, but the language is only a visualization techniques and does not include a formal defined set of modeling constructs, which can be used for code generation.

Only ArchiMate and UML4ODP are domain independent approaches. UPDM is build upon a domain dependent and not standardized enterprise architecture framework. The tool support for modeling is quite good for all of the frameworks. ArchiMate does not support a model or code generation. UML4ODP and UPDM are formally defined and existing approaches for code and model generation for UML diagrams could be reused and adapted.

In the following the three approaches will be evaluated using the requirements concerning the single viewpoints of RM ODP. Table 14 provides an overview of the evaluation.

	ArchiMate (Open12a)		**UPDM (OMG12b)**		**UML4ODP (ISO09)**	
7 Assignment of responsibilities	No explicit support	1	i.e. Capability to Org. View	3	Through links between the Viewpoints	3
8 Integrate motivation and requirements	Not in core, but in motivation extension	2	Capability/ Strategic Viewpoint	3	Objective	2
9 Support Use Cases	Not	1	Not	1	Not	1
10 Set up a system-wide set of vocabulary	Included in Information Systems Arch.	3	Included in All-Viewpoint	3	Included in Information VP	2
11 Specification, integration of different architectural styles and patterns	Not explicit, but possible	2	Not explicit, but possible	2	Applied in engineering viewpoint	2
12 Support for Decomposition and Composition	Analog UML components	2	(Internal) Block Diagrams	3	Analog UML components	2
12.1 Support for Choreography and Orchestration	No explicit techniques	1	In MODAF, but not explicit in UPDM	2	No support of composition	1
12.1 Interface and behavior specification of components	Interfaces for components, specification of collaboration and interaction	3	Refinement of communicatio n paths possible	3	Interface templates and signatures, internal behavior, but not external behavior (collaboration)	2
13 Support a service oriented architectural style	Use of service concept	3	Service View	3	No concept of services ("object that interact at interfaces")	2
13.1 Specification of services	Interfaces, Roles, Service Behavior, but no information flow, service interface specification	2	Fully supported in Service View	3	no service concept	1
13.2 Identification of Services	No explicit techniques	1	Capability to service mapping	2	Group predefined operations to interfaces	2

13.3 Reuse of services	Reuse of components and services	3	No support	1	Not explicit, reuse of interfaces possible	2
13.4 Classification of Services	Application, Business, Infrastructure Services	3	Not defined in specification	1	Not defined in specification	1
14 Support specification of distribution transparencies	Services as concept for distribution transparencies	2	Not directly through a viewpoint	2	Engineering VP, node concept	3

Table 14 Evaluation with the requirements concerning the single Viewpoints (Own contribution based on Open12a, OMG12b, ISO09)

None of the three modeling approaches provides full support for the assignment of responsibilities and the integration of architectural styles and patterns on all architectural layers. Only UPDM provides full support for modeling the motivation and requirements of the system and for the composition and decomposition concepts, the two other approaches provide only partial support in these issues. The concept of use cases is not supported by any of these approaches, although this is a quite often used and good understandable concept to describe the functionalities of a system.

Since the agreement on a shared information model is very important all of the three modeling approaches provide full support for this requirement through an own viewpoint. It is the Data part in the Information Systems Architecture from ArchiMate, the All-Viewpoint in UPDM and the Information Viewpoint in UML4ODP.

ArchiMate, UPDM and UML4ODP provide only partial support for requirement 13, the support of a service oriented architectural style. The first two ones contain both a service concept in contrary to UML4ODP, where a service concept is missing. The system is seen here as various objects interacting together. ArchiMate lacks of concepts for full specification and identification of services and UPDM does not provide good support for classification and reuse of services.

The specification of distribution transparencies is only directly supported by UML4ODP in the Engineering Viewpoint. The two other approaches UPDM and ArchiMate do not explicitly define concepts for integrating distribution transparencies. But both approaches enable the specification of them through already existing ones, like the service concept that abstracts from the underlying infrastructure in ArchiMate.

The gaps identified in the above evaluation of the existing approaches point out the issues MODEA should concentrate on. For example the reuse of existing and standardized languages as techniques in the approach is missing in UPDM, ArchiMate and UML4ODP. But also issues were frameworks provide very high advantages are important to be considered when developing MODEA. An example for this is the good readability and understandability of ArchiMate specifications.

5 MODEA

5.1 Vision of MODEA

In the previous chapter an introduction to the domain of open distributed systems as well as enterprise architecture frameworks was given. Thereby seven important problems, that occur in the development and specification process of open distributed systems are examined. These are:

- Heterogeneity of the components
- Collaboration between the vendors
- Providing the right functionality
- High complexity and scope
- Global optimization
- Distribution transparencies
- Need for flexibility

To cope with these problems a modeling approach based on an enterprise architecture framework can be used. In chapter 3 several requirements for such a modeling approach are defined. Among these are smart diagrams, the use of viewpoints and existing standard as well as a formalized modeling approach. Additionally more detailed requirements for the approach are the integration of Use Cases, support for decomposition and composition, support for a service-oriented style and also for the specification of distribution transparencies.

Existing modeling approaches based on enterprise architecture frameworks like UPDM, ArchiMate and UML4ODP are examined in the previous chapter. In a following analysis of these three modeling approaches using the requirements in chapter 3 strengths and weaknesses of them are identified. Important to notify here is that all of these approaches define their own modeling language or UML profile for the architecture description. Although all approaches enable the specification of architecture with a service-oriented style, some techniques required like identification, support for reuse or classification are not well established. Additionally especially UML4ODP and UPDM are very extensive approaches, which are often too complex and over-weighted. Furthermore UPDM contains specific aspects for the military domain. The main problem with ArchiMate is, that it is only created for visualizing architectures, and not for model driven development.

With MODEA a model driven approach based on existing and well-established modeling techniques will be created. Thereby the increasing use of web-enabled functionality in current enterprise architectures will be considered. The goal is to create good and pragmatic models based on existing standards to support the development and specification process of open distributed systems. Through the use of formalized modeling techniques and a formal defined framework as foundation, the approach should provide support for model and code generation in the future.

To reach the described benefits MODEA will be specified upon an Enterprise Architecture Framework, using its concepts to structure the overall specification. These concepts will be modeled with use of existing and well-established standards. The techniques should have a good tool support and should be widely used in industry and research. Furthermore the techniques should allow the specification of the concepts defined in the requirements (chapter 3), like use cases, motivation and requirement, architectural styles and patterns, decomposition and composition as well as a service oriented style. The idea for the provisioning of MODEA through combining existing modeling techniques with an enterprise architecture framework is illustrated in Figure 23.

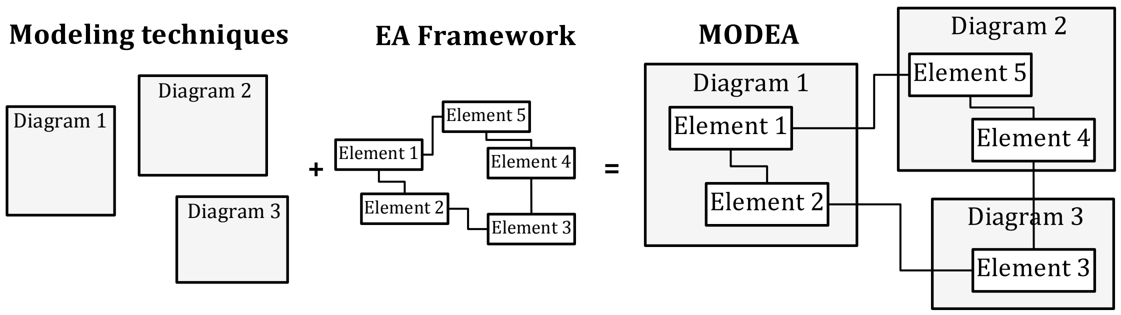


Figure 23 Provisioning of MODEA using modeling techniques and an EA Framework

The Enterprise Architecture Framework used in MODEA as foundation is the Reference Model for Open Distributed Systems (ODP) defined in ISO/IEC 10746-1 – 10746-4 (ODP98a, ODP10a, ODP10b, ODP98b). A first introduction and reasons for this selection were already given in chapter 2.4. Advantages on RM ODP are the formal definition of the viewpoints and their relationships, the domain independency of open distributed systems as well as the issue that it is an ISO standard which will remain stable. Additionally the framework supports in dealing with a high heterogeneity in a system and it also already used in the two example cases used in this report.

The concepts described in this standard will be the modeled using UML, SoaML, BMM and BPMN. SoaML and BMM are both UML profiles. The following figure provides an overview of the framework and the modeling techniques used to define MODEA.

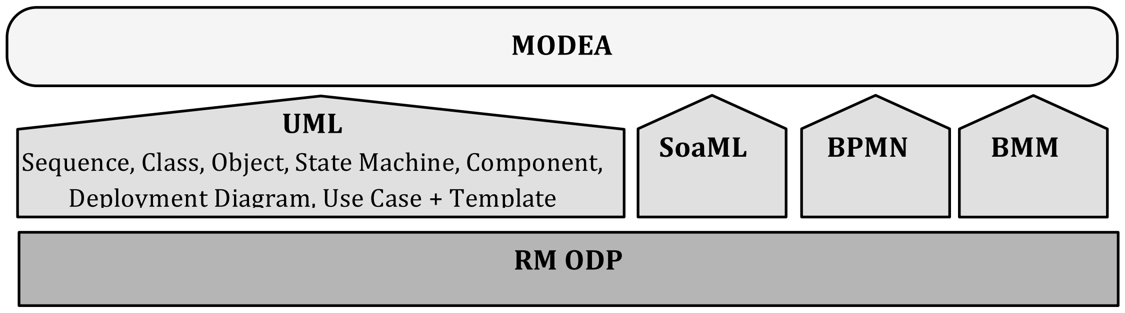

MODEA
UML
Sequence, Class, Object, State Machine, Component, Deployment Diagram, Use Case + Template
SoaML
BPMN
BMM
RM ODP

Figure 24 Basic principle for MODEA

For MODEA UML is chosen, since it “is the most frequently used language for visualizing static and dynamic aspects of software- intensive systems” (Bro08). Furthermore it is “widely used by organizations, and supported by more than a dozen different product offerings”(Bro08).

A lot of domains can be captures with plain UML and defined UML profiles. But UML lacks of an overarching concept of how to link the various single diagrams. In MODEA such a linking of the different diagrams with respect to the specification of the RM ODP framework will be made. This enables to use UML with its strong support in tools and industry for creating a coherent specification of an open distributed system. “The standardization of the UML notation has helped the software industry to communicate understanding of software artifacts using a commonly understood visual language.” (Bro08) With an integration of UML into the context of enterprise architecture modeling, these benefits of UML should be also earned in EA modeling.

Use Cases are important to integrate, because they are a widespread technique used to specify the functional requirements. (And01) The UML Use Case diagram enables the specification of actors, a system subject and the use cases they participate in. Further description of Use Cases in UML can be done through further diagrams but also a textual description. In MODEA we will use templates for this, since “use case models constructed using the Template or Style guidelines are easier to understand” (And01).

Further diagrams used from the UML specification are Sequence Diagrams, State Machines, Class and Objects Diagrams, Deployment Diagrams and Component Diagrams.

A service-oriented style addresses several of the identified problems in open distributed systems (chapter 3). Therefore and with the increasing importance for the integration of web-enabled functionality, this style should be supported in MODEA. A new modeling standard called SoaML was published this year by the OMG. The Service oriented architecture Modeling Language (SoaML) enables the identification and specification of services as well as defining service consumers and producers. It provides also support for describing the policies for using and providing a service and for defining classification schemas. (OMG12a) With respect to the requirements set in chapter 3, SoaML would be a good choice for realizing a service oriented architecture style. Additionally SoaML provides another important benefit with its definition of a metamodel and UML profile through being able to link services to model elements of the OMG Business Motivation Model, UML Use Cases and also to some process notations. (OMG12a) Furthermore through the specification of an UML profile, modeling SoaML diagrams is possible with at least any tool providing support for UML 2.

The Business Motivation Model is a business modeling specification, published by the OMG group, to define the motivation e.g. “to be able to say ‘way’” (OMG10) for certain business activity. On advantage of BMM is the very simple definition of it, since the few concepts only have basic attributes and most of the associations are unconstrained (OMG10). Together with the provided tool support for BMM, also in the context of UML diagrams, BMM would be a good choice for modeling the goals in MODEA. A lot of modeling tools like Select Architect, Sparx Enterprise Architect, IBM Rational Software Architecture and partially also Modelio, provide support for requirements modeling using the BMM standard. Although BMM does not contain an UML profile in its specification all mentioned tools enable the definition of links between the elements of BMM and thus of UML or BPMN. (Ams08, Sel12, Sparx12b, Mod12)

At least Collaboration and Process Diagrams from the Business Process Modeling Notation (BPMN) are chosen as technique for defining business processes. This notation is also supported in nearly all modeling-tools, which provide support for UML 2. Furthermore BPMN has two more advantages:

- It is a “notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes” (OMG11).
- And it “creates a standardized bridge for the gap between the business process design and process implementation” (OMG11).

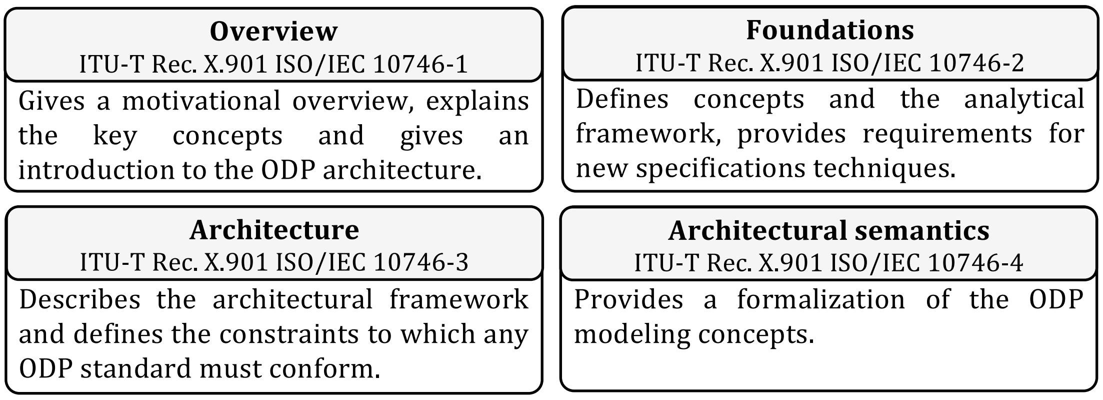
Thus BPMN would be sound selection for MODEA, since enterprise architecture typically encompasses several domains with several stakeholders. Additionally BPMN would support the future goal for implementation support.

In the following the used framework RM ODP will be explained more detailed and then the MODEA is introduced through explaining the modeling approach in each viewpoint. Afterwards the connections between the viewpoints and how to model them in the UML context are shown up. At least the current tool support is described.

5.2 RM ODP

5.2.1 Introduction

A short introduction to RM ODP was already given in chapter 2.4. Now the concepts and main elements are illustrated. The RM ODP standard is published in four parts:



Overview
ITU-T Rec. X.901 ISO/IEC 10746-1
Gives a motivational overview, explains the key concepts and gives an introduction to the ODP architecture.

Foundations
ITU-T Rec. X.901 ISO/IEC 10746-2
Defines concepts and the analytical framework, provides requirements for new specifications techniques.

Architecture
ITU-T Rec. X.901 ISO/IEC 10746-3
Describes the architectural framework and defines the constraints to which any ODP standard must conform.

Architectural semantics
ITU-T Rec. X.901 ISO/IEC 10746-4
Provides a formalization of the ODP modeling concepts.

Figure 25 Parts of the RM ODP Specification (Own contribution based on ISO98a)

In the four documents, mainly the first three, the main elements of RM ODP are specified. These are

- Object Modeling,
- Viewpoint Specification,
- Distribution transparency and
- Conformance.

Each of them will be described in the following.

5.2.2 Object Modeling

In the part 2 of the ODP Specification, ITU-T Rec. X.901 ISO/IEC 10746-2 (ISO10a), modeling concepts as foundation for building the architecture of ODP system are defined. This object modeling approach is reused and refined when specifying the viewpoint languages in part 3 of the specification. All viewpoint languages are based on the same concepts. Therewith the correspondences between the viewpoints can be defined without using the same notation for all viewpoints.

There are three categories of modeling concepts used in the specification.

- The Basic modeling concepts describe the set of elements that are the basis for the system description with ODP.
- The specification concepts describe elements, which are required for reasoning about the ODP system and for defining requirements on the specifications languages.
- The structuring concepts deal with notions and structures that are generally applicable in the design and description of distributed systems.

The basic set of elements for a system description after the RM ODP encompass Objects and the Environment, Interfaces, Interaction Points, Behavior, Actions, and State. (ISO10a) Figure 26 provides a simplified overview about how these elements are related to each other.

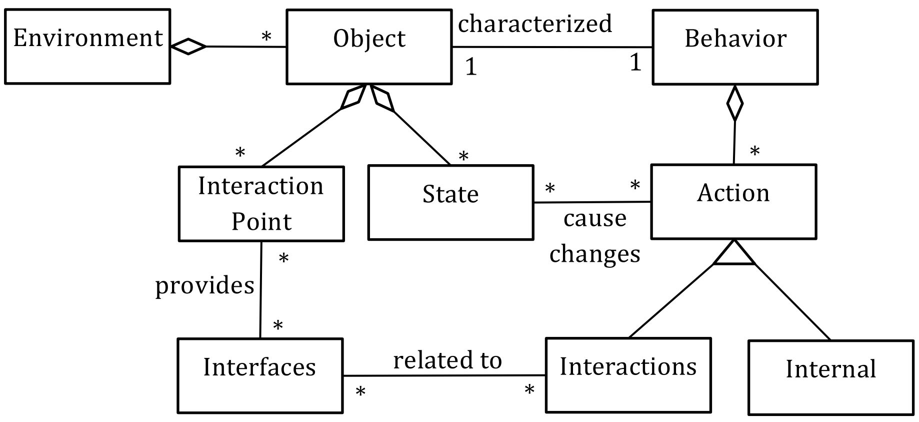


Figure 26 Elements of the basic modeling concept (Own contribution based on ISO10a)

The most important concept is the object. "ODP system specifications are expressed in terms of objects" (ISO98a). Every object interacts in an environment, which contains also other objects. Through interaction points, which can contain several interfaces, objects are able to interact with other objects of the environment. Each object can have several states. A State describes conditions, which determines a possible future behavior. Each object is characterized by a specific behavior, which is a collection of actions. An action can be internal or external. External actions are called Interactions. An action also cause changes in the states of the participating objects. In the updated version of the Foundation (ISO10a) also a service concept is integrated. A service is understood as "A behaviour, triggered by an interaction, that adds value for the service users by creating, modifying, or consuming information" (ISO10a). Those services are associated with an interface and defined by an interaction. Furthermore a service can be characterized by a service type and also be composed of other services. At least "the provision of a service involves a collaboration between its provider and user. This collaboration may involve a complex series of interaction" (ISO10a).

The second concepts are the specification concepts. Here are elements defined which are needed for reasoning about the specification of and ODP system and they also define requirements on specification languages used for the specification of an ODP system. This includes the concept of composition and decomposition of objects, which is "used to organize the specification of a distributed system as a set of specifications, each on dealing with a different level of abstraction" (ISO98a). Also the behavioral compatibility between objects and the concepts of types, classes and templates belong to the specifications concepts.

One important concept for MODEA is roles. "A role my correspond to a subset of the total behavior of a component object." (ISO98a). That means when considering an object in terms of a specific role, only "a named subset of its actions is of interest" (ISO98a). If an object is composite by several components objects a role could be the behavior identified with one them. Often roles are related to interfaces.

At least there are the structuring concepts. They include grouping concepts like Domains and Groups as well as a Naming concept to refer entities in a given context. In this category also the Contract concept can be found. It is defined as "a general concept for characterizing and regulating the cooperation of objects" (ISO98a). In the collaboration context of objects there exist also the concepts of binding and liaison to define the contractual context and the relationship between cooperating objects.

5.2.3 Architectural Framework: Viewpoints

Part 3 of the RM ODP specification (ITU-T Rec. X.901 ISO/IEC 10746-3) defines an architectural framework for structuring the specification of ODP systems (ISO10b). The specification of an ODP system is done using the concepts of viewpoints, viewpoint specifications and distribution transparencies. The viewpoint specifications are expressed using a set of language terms. In RM ODP five viewpoints are identified. Figure 27 gives an overview of them together with a short explanation what they are about.

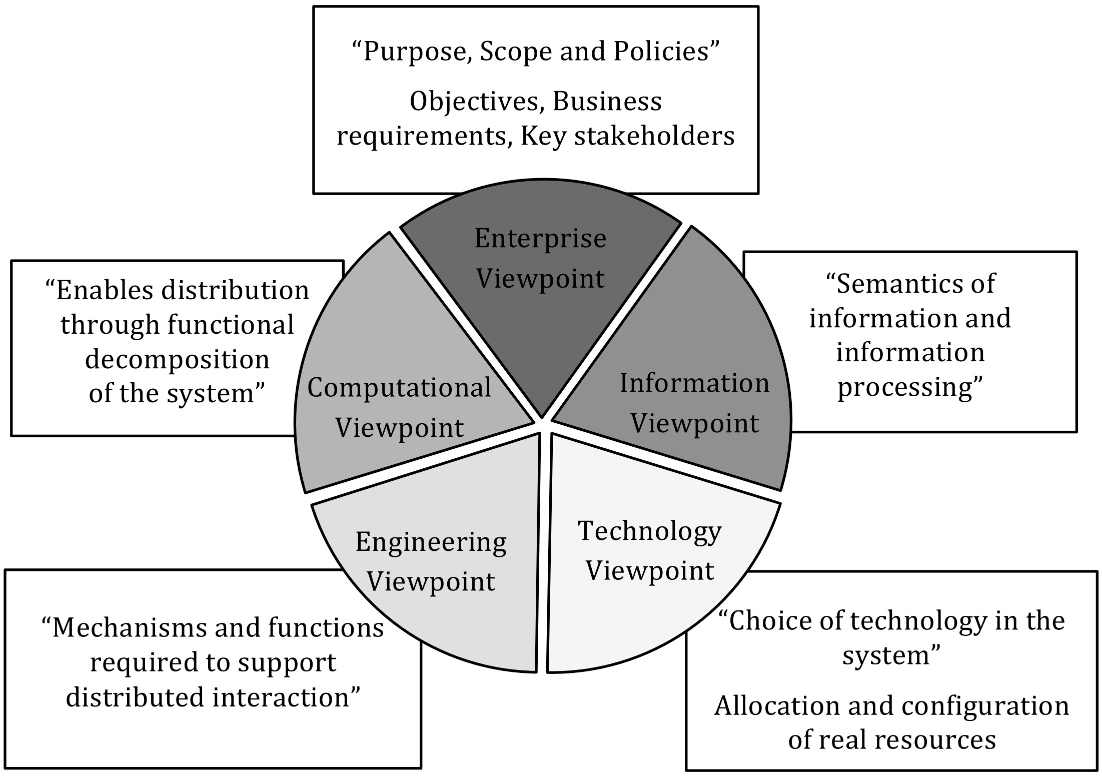

"Purpose, Scope and Policies"
Objectives, Business requirements, Key stakeholders
Enterprise Viewpoint
"Enables distribution through functional decomposition of the system"
Computational Viewpoint
Information Viewpoint
"Semantics of information and information processing"
Engineering Viewpoint
Technology Viewpoint
"Mechanisms and functions required to support distributed interaction"
"Choice of technology in the system"
Allocation and configuration of real resources

Figure 27 Viewpoint Overview of RM ODP (Own contribution based on ISO98a)

The single viewpoints with their concepts will be described shortly in the following. The concepts used in each viewpoint are introduced in chapter 3.2, when explaining the modeling approach for MODEA.

The **enterprise viewpoint** "focuses on the organizational situation in which the design activity is to take place" (Lin10) In the specification the purpose, scope and policies of the system are described and it deals with the objectives, the business requirements and business rules. (ISO98a, Lin10)

The enterprise specification is expressed using the roles played and the activities undertaken by the system as well as policy statements about the system. The key stakeholders for this viewpoint are the owners of the business processes and the managers responsible for the policies (ISO10b).

The **information viewpoint** is responsible for the modeling of the shared information in the system. Therewith a common understanding of the information and a consistent interpretation of those can be ensured.

In this viewpoint the configuration of the information objects, the behavior of the objects, the actions that can happen and also the constraints that should always hold are specified. The focus lies on the structure of the information and the information flow. Therefore three schemas exist.

- In the invariant schema a configuration of information object types is specified, which must always be satisfied.
- The static schema defines the state of information objects at one point in time.
- The dynamic schema specifies the allowed state changes. This can also be modeled as transition from one static schema to another. (ISO98a, ISO10b)

The goal in the information viewpoint is to "avoid the divergence of use and incomplete collection of information" (Lin10).

In the **computational viewpoint** the system is decomposed "into objects performing individual functions and interacting at well defined interfaces" (ISO98a). The functional decomposition is described in a distribution transparent manner in this viewpoint, it is not specified where and how these objects are implemented. The focus of the specification lies on "the object model which defines the form of interface that an object can have; the way that interfaces can be bound and the forms of interaction which can take place at them" (Rom05). The environment contracts for the interfaces of the objects play also a major role in the specification of the computational viewpoint. (ISO10b)

The computational viewpoint is linked to the information viewpoint in terms of the action types used to specify the interactions. Whereas the engineering viewpoint specifies the communication mechanisms, that are required to support the behavior at the interfaces in the computational viewpoint. (Rom05, ISO98a)

The **engineering viewpoint** "defines the mechanisms and functions required to support distributed interaction between objects of an ODP system" (ISO10b).

In the specification the required infrastructure to support the functional distribution of an ODP system is required. Therefore the configuration of engineering objects with the interaction channels between them is specified. Engineering Objects can provide application functionality but also functionality required for the physical distribution, communication, processing and storage. It is important to notice that the engineering specification is independent from a specific platform or technical infrastructure. (ISO98a, ISO10b)

Nowadays the specification of the engineering viewpoint can be simplified through the integration of standard middleware or web service components. They provide mechanisms required to support the distribution of the system and can easily be referenced in the specification of the engineering viewpoint (Lin10)

The **technology viewpoint** "expresses how the specifications for an ODP system are to be implemented" (Lin10). The viewpoint is concerned with restrictions to the hardware due to existing platforms or budget requirements. In the specification the allocation and configuration of the real resources like hardware, software and communication technology of the OPD system are described. It also includes extra information for testing and specifies processes and activities for provision, deployment, maintenance and evolution of the system.

The concepts to describe the viewpoint are a configuration of technology objects and the interfaces between them as well as a selection of implementable standards for the ODP system. (ISO10b)

5.2.4 Distribution transparency

Distribution transparencies in RM ODP are defined as "the property of hiding the properties of distribution from end users and specifiers in the enterprise, information, and computational languages." (ISO98a)

The required functionality for this aspect is identified and specified in the engineering viewpoint. For example engineering objects can move from one location to another. This requires functionality to record and discover the current location of an object. The engineering objects work together to provide such transparencies to the upper viewpoints. (ISO98a)

Possible transparencies, which can be supported in ODP systems, are Access, Failure, Location, Migration, Persistence, Relocation, Replication and Transaction Transparencies. Further information about the transparencies can be found in the RM ODP specification. Since transparencies are related to performance and cost trade-offs, typically not all transparencies are fully supported in an ODP system. (ISO98a)

5.2.5 Conformance

The RM ODP specification provides also a framework for assessing system conformance. This is important in an open distributed system, since different vendors can provide parts of the system. "Conformance is a relation between a specification and a real implementation […]. It holds when specific requirements in the specification […] are met by the implementation" (ISO98a).

In the specification of the five viewpoints reference points, which can be declared as conformance points, are identified. These are the points at which conformance will be tested and therefore have to be accessible for test. (ISO10a)

Since the framework for conformance assessment will not play a major role in the ongoing report, this concept is not explained in detail here. Further information can be found in ISO98a and ISO10a.

5.3 Modeling the ODP Viewpoints

In this chapter the model driven approach for open distributed system using an enterprise architecture framework (MODEA) is introduced. This is done based on RM ODP, introduced in the previous chapter. First the modeling approach for the concepts of each viewpoint is presented. Then the connections between the models of the single viewpoints are examined. In the end the support for the approach in current modeling tools is described.

5.3.1 Enterprise Viewpoint

The Enterprise Viewpoint in RM-ODP answers the following questions:

- "What is the purpose of the system?"
- "What are the business requirements for the system?"
- "Who are the key stakeholders and how do they interact?" (Lin11)

The purpose and motivation for the system is shown in a Business Motivation Model. The requirements of the system are captured in Use Case templates and their corresponding

Business Processes are described using BPMN. The stakeholders and their interactions with the system are shown in an UML Use Case Diagram.

The base concept in the enterprise viewpoint of RM ODP is a community. A system consists of several communities containing roles to represent the participating parties. The roles collaborate together to reach a specific objective. The behavior itself is described through a composition of several activities and each activity is linked to a responsible role. In addition a definition of policies allow a dynamic reaction to changing circumstances in the business context modeling. Following the system is described as an overlapping set of rules, described by policies and business processes. This enables more flexibility than the specification of one single algorithm (Lin11, ISO10b). Figure 28 represents a simplified conceptual model of the elements in the RM ODP Enterprise Viewpoints. Furthermore the figure shows a mapping of these main concepts to the modeling concepts used in MODEA.

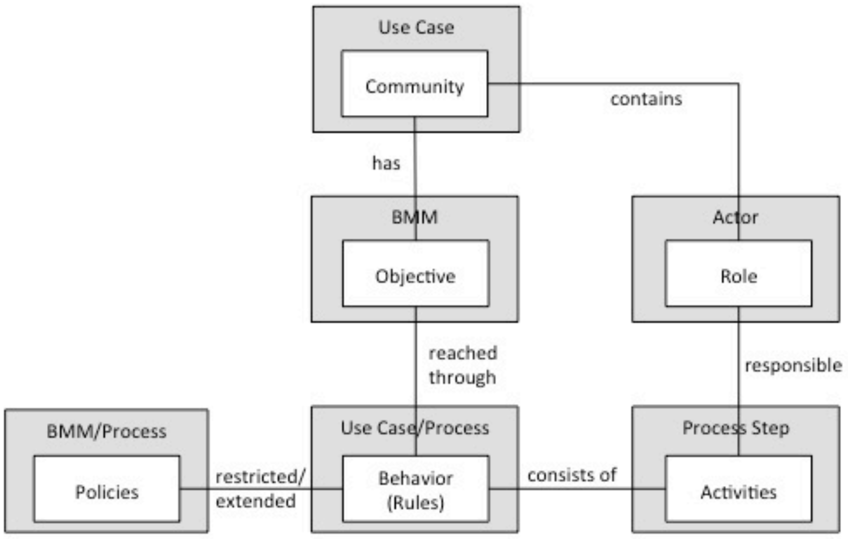


Figure 28 Mapping of RM-ODP concepts in the Enterprise Viewpoint to MODEA modeling concepts (Own contribution based on ISO10b)

A community in RM ODP can be seen as the overall system or any subsystem in the given context. In MODEA this is represented as the subject in the UML Use Case Diagram. An UML Use Case Diagram typically describes, "what the system is supposed to do" (OMG07). Thereby it abstracts from technical details and focuses on the required usage of the system. The *Subject* is the system to which the use cases apply. The roles played by users and other interacting systems or persons, are represented as *Actors*. Actors are always external from the system. The interactions between these actors are specified through *Use Cases*. The Use Cases can be further specified by "some kind of Behavior […], such as interactions, activities, and state machines, or by pre-conditions and post-conditions as well as by natural language text where appropriate". (OMG07)

The behavior of a community, with its goal to reach a specific objective, is represented as a Use Case and further described through a Business Process using BPMN in MODEA. "Use case modelling has become a popular and widely used technique for capturing and describing the functional requirements of a software system." (And01)

The Business Process Modeling Notation BPMN, also an OMG standard, is a graphical specification language for modeling business processes and work flows. The main concepts used are

- Pools and Lanes
- Flow Objects (Gateways, Events, Activities)
- Connecting Objects (Sequence Flow, Message Flow)
- Artifacts (Groups, Data). (OMG11)

The objective, which should be achieved by the Use Cases, is described in the Business Motivation Model. Policies and rules are also captured in the Business Motivation Model, where rules are some kind of actionable policies. The Business Motivation Model is an OMG standard describing the why of any business activity. It defines the results that the approach should achieve. Therefore the following core concepts are defined:

Ends: Describes the *Vision*, which is amplified by *Goals*, which are quantified by *Objectives*

Means: Describes the *Missions* and its *Strategies* and guiding Business *Policies*. Strategies implemented by *Tactics*, Policies get actionable through Business *Rules*(OMG10)

The following figure describes the modeling concepts of MODEA and how the different diagrams are related. From the BMM the higher level concepts like Vision, Goals, Mission, Strategy and Business Policy are not mentioned, since only the low-level concept which implement the higher level ones are important to the other diagrams.

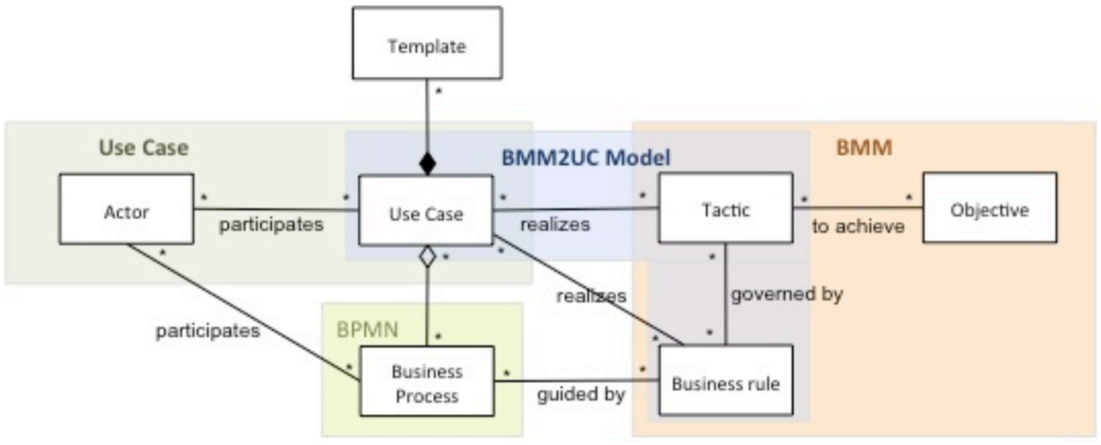


Figure 29 Concept Model of the Enterprise Viewpoint

In MODEA the tactics defined in the BMM are realized by Use Cases. A business rule can also be realized directly in a Use Case or only indirectly. In the last case no explicit relationships to other model elements are possible. If a business rule is realized in a Use Case it also has effects on the business process describing this Use Case. A specific diagram called BMM2UC Model is visualizing the relationship between Tactics and Rules with Use Cases.

Use Cases, as representations of interactions between the stakeholders, are summarized in a UML Use Case Diagram. The role a stakeholder plays in a specific use case is represented as Actor. The behavior of a use case will be defined through a Business Process using BPMN. Business rules guide the Business Process, but there will be no special view showing this relationship, since you can trace the corresponding Business Rules also via Use Cases and Tactics. The pools in a Business Process represent the Actors that participate in the described Use Case.

Use Cases and Business Processes are supplementary concepts. As Nawrocki et al. (2006) found out in an experiment, "Use cases are easier to understand than BPMN diagrams" but also that Business Process diagrams support the understanding of use cases. They conclude with the proposition that the “description of business processes should be based on use cases”. Also Lübke et al. (2008) analyze the relationship between Use Cases and Business Processes and present an algorithm of how to generate Business Processes out of Use Case Templates.

With the Business Processes and the Use Case Diagram one part of the use case modeling is specified. The second part is a use case description containing detailed requirements (And01). This can be done informally and unstructured but also in a formal style or with pseudo code. In experiments Anda (2001) comes to the result that “use case models constructed using the Template or Style guidelines are easier to understand”. Adolph et al. (02) also point out that “a well-written use case is relatively easy to read”, which is “one reason for their popularity”. Therefore he describes patterns for well-written use cases (Ado02) and the co-author Cockburn gives also further advices of how to write effective use cases (Coc00).

In MODEA we will use a predefined template to further describe Use Cases. The exact structure of the template is dependent from the project and the application of the Use Cases in the development process. The main issues that should be included in every template after (And01, Coc00) are: Title, Actor, Trigger, Summary, Preconditions, Basic flow of events, Extension Points, Alternate courses and Post-conditions.

Figure 30 gives an overview of the used models and how they are connected to each other.

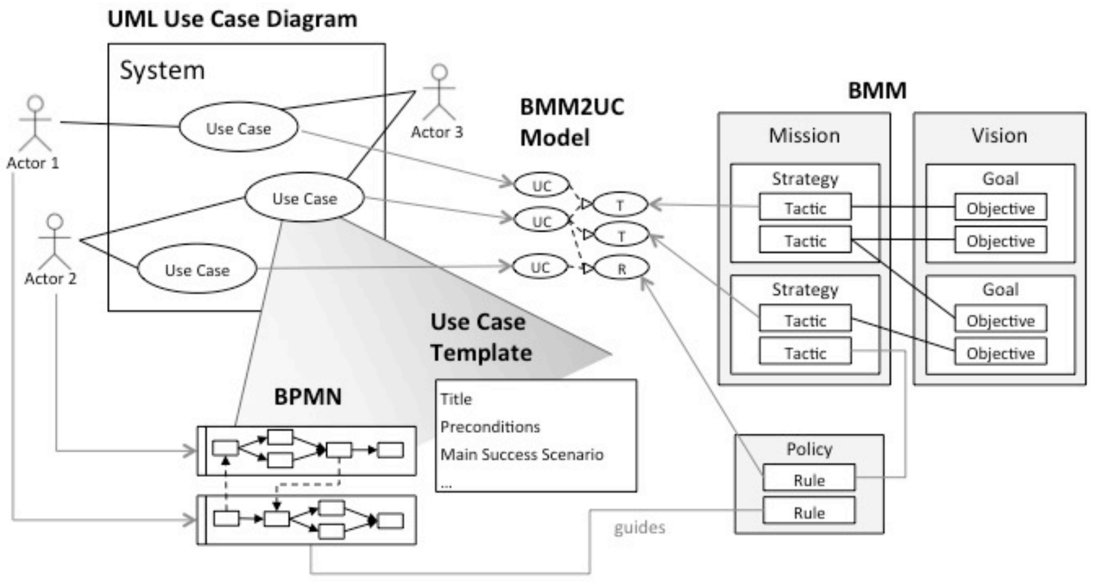


Figure 30 Schematic Overview of the Enterprise Viewpoint

Above one possible way to model the Enterprise Viewpoint using BMM, Use Cases and BPMN is shown. The links between the single diagrams are pointed out with grey arrows. The actors participate in Use Cases and are linked to the pools of the BPMN diagrams. The BPMN diagram as well as the Use Case Template provides further information about a Use Case. A Use Case realizes specific Tactics and Business Rules defined in the BMM. If a Business Rule is realized by a specific Use Case it also as effects on the corresponding Business Process.

5.3.2 Information Viewpoint

The Information Viewpoint in RM-ODP answers the following questions:

- “What are the data types of the information that the system will handle?”
- “What are the relationships between these types?”

- "How will the state of the data in the system evolve as the system operates?"
- "What are allowable actions and how will they affect the state of the data?"

(Lin11)

The main concepts in RM ODP used for the description of the information specification are the information object and their types as well as the relationships between the various objects and the various types. Every information object has a specific information object type and a specific state. Action can cause changes to the state of one or more objects. In the definition of an action also the information types, which participate in the action are defined. (Lin11, ISO10b)

A simplified model showing the main concepts that are dealt within the Information Viewpoint can be seen in Figure 31.

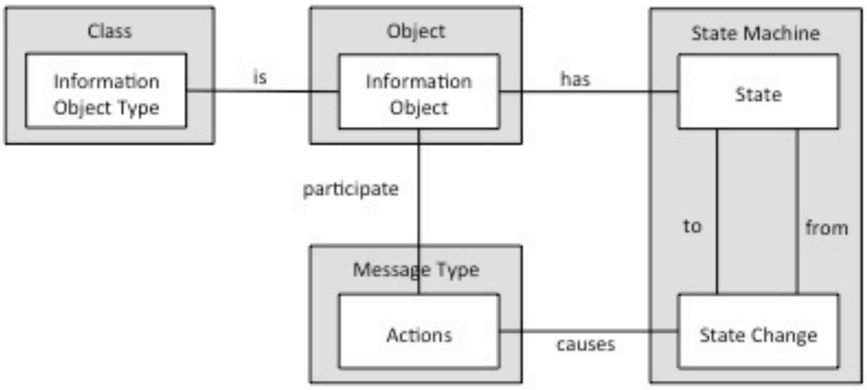


Figure 31 Mapping of RM-ODP concepts in the Information Viewpoint to MODEA modeling concepts (Own contribution based on ISO10b)

The data types with their relationships to each other are described using an UML Class diagram in MODEA. A specific state of the data can be described with an UML Object Diagrams. At least the state of the data and allowable actions can be described using UML state machines.

Following the concepts used in the Information Viewpoint can be easily mapped to concepts in the UML specification. Information Objects in RM ODP can be modeled as UML objects. Their structure is specified using an UML Object Diagram. The Object diagram is a variation of the class specification of UML. It contains the instance specifications (objects) and links between them. (OMG07)

These UML objects are instantiated with UML classes, which represent the Information Object types. The classes itself and relationships between them are modeled using the UML Class Diagram. Here concepts like association, aggregation and generalization can be used to describe the structure of the information types represented as classes. (OMG07)

These classes are also referenced as attributes when specifying RM ODP actions. These actions are modeled with the Message Type concept of SoaML. MessageTypes are a construct from the SoaML specification. They extend the UML metaclasses DataType, Class and Signal with the goal to "specif[y] [the] information exchanged between service consumer and providers" (OMG12a). A message type can contain classes as attributes or aggregated associations of data types.

The state and state changes of information objects can be modeled using UML Behavioral State Machines. The State Machines can be used for "modeling discrete behavior through finite state-transition systems" (OMG07). The most important concepts in this kind of diagrams are states and transitions. A state represents a static situation like waiting for an event, but also a dynamic one like performing a specific behavior. A transition is a directed relationship between modeling elements of the state machine. (OMG07)

Following there are four different diagrams in the MODEA approach for the information viewpoint. Figure 32 illustrates how these diagrams with their containing concepts, relate to each other.

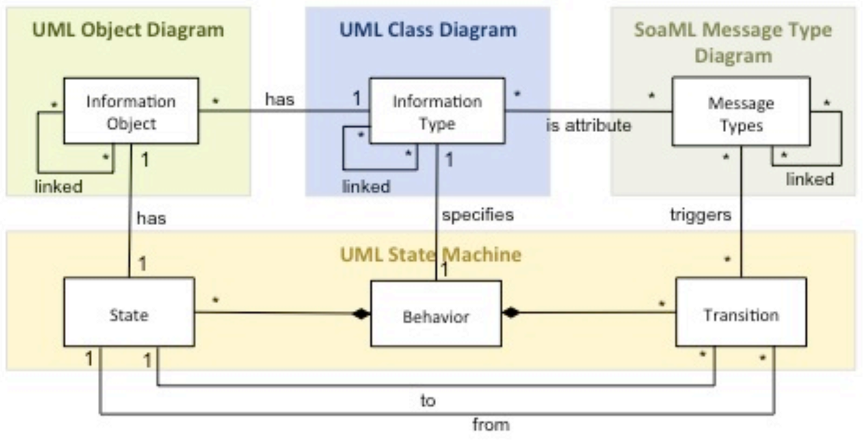


Figure 32 Concept Model of the Information Viewpoint

Each information object has a specific type and a specific state. The type is represented through instantiation of the object with the corresponding class. The state is represented through an attribute. The information type specifies the possible state changes of an information objects through a linked UML State Machine. The outgoing transitions in one state represent the exits that could be used to change the state. Special message types can trigger one or more transitions. The parameters used to define the message types have to specify as information type.

Figure 33 provides a schematic overview of the diagrams and how they relate to each other. It illustrates the theoretical concepts described above.

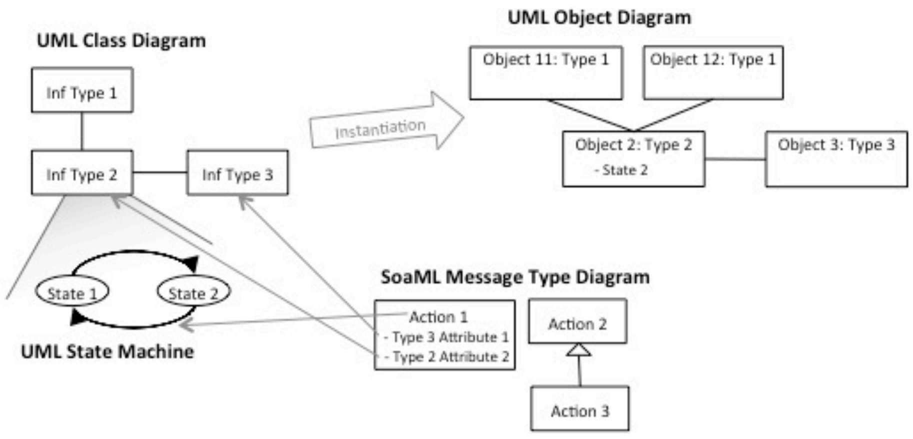


Figure 33 Schematic Overview of the Information Viewpoint

The diagram contains three different information types in UML Class Diagram. For Type 2 also a corresponding State Machine is given. Action 1, which has Type 2 and Type 3 as Attributes, triggers the upper transition. The corresponding Object Model contains 4 information objects. Two Objects are from Type 1, one object is from Type 3 and one from Type 2. The last one, object 2, is in state 2 from the state machine of its information type.

5.3.3 Computational Viewpoint

The Computational Viewpoint in RM-ODP answers the following questions:

- What is the "basic functionality of the system"?
- How are the services offering these functionalities?
- "How [are] these services [...] realized internally in terms of components and connectors"? (Lin11)

The main concept in RM ODP to describe the computational specification is the computational object. "Computational Objects model the basic functional elements of the system" (Lin11). The computational objects provide their functionality over interfaces to other objects. Each interfaces fulfills therefore an environmental contract, which specifies a set of Quality of Service constraints. The behavior at any interface is described through interactions between the participating parties. Each computational object also realized a defined behavior. A behavior consists of actions, which can be internal or external. External actions are called interactions (ISO10a, Lin11). In the case of specifying Human/System interaction the "enterprise interactions and information objects are modelled in terms of computational objects providing services" (ISO10b). The concepts of the computational specification in RM ODP with their relationships to each other are shown in Figure 34. Thereby also the mapping to the concepts, which will be used in MODEA is shown.

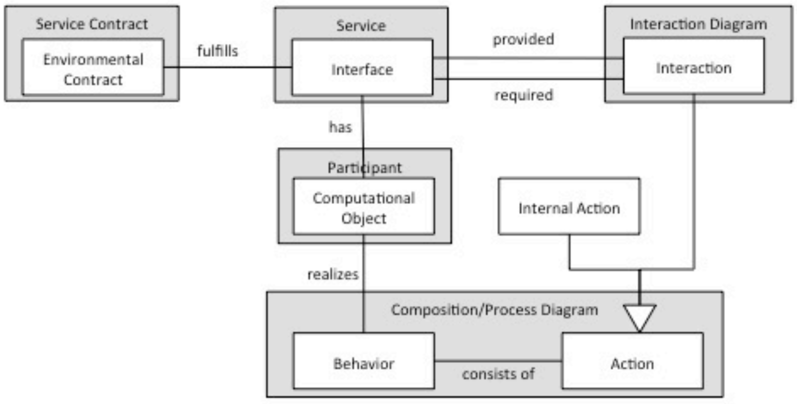


Figure 34 Mapping of RM-ODP concepts in the Computational Viewpoint to the MODEA modeling concepts (Own contribution based on ISO10b)

In open distributed system many vendors work together through providing and requiring functionality from each other. When using a service-oriented architecture (SOA) approach these functionalities are provided as a service. A SOA "is a way of describing and understanding organizations, communities, and systems to maximize agility, scale, and interoperability" (OMG12a). This paradigm works for integrating existing functionality as well as for creating new one.

In MODEA mainly SoaML is used as modeling technique to specify the Computational Viewpoint. Thereby not only the Human/System interaction are modeled as Services, but also the interaction between different parts of the system. SoaML "provides a standard way to architect and model SOA solutions using the Unified Modeling Language" (OMG12a). The current modeling approach for OPD, UML4ODP, enables a RPC-based architectural style only. With the use of SoaML also support for other interaction paradigms like document centric messaging or publish/subscribe are given. (OMG12a). The characteristics of services in RM ODP, described in chapter 5.2.2 are also adapted in MODEA. The main concepts used in SoaML are Participants, Service Architectures and Service Descriptions. A service can be specified in three ways. The simplest one is the use of a UML interface. They define one-way services, which do not require a protocol and represent in most cases RPC-style web services. To specify two-way services with a specific protocol the concept of ServiceInterfaces or ServiceContracts can be used. Such bi-directional services are characterized by the fact that "both the provider and the consumer have responsibilities to invoke and respond to operations, send and receive messages or events" (OMG07).

In the Computational Viewpoint the Service Architectures with its containing services and participants are specified. Therewith the ServiceContract based specification approach for services is used. The SoaML Diagrams in the Computational Viewpoint are extended through UML Sequence Diagram and BPMN. The mapping of the participants to system components and the specification of the provider and consumer interfaces are done in the Engineering Viewpoint.

The Environment Contract can be roughly mapped to a Service Contract. The interface in RM ODP will be represented as SoaML ServiceContract in MODEA. A computational object is modeled as Participant. The behavior of a participant can be modeled using two different ways. One way is decomposition and the other one is using behavioral diagrams like BPMN. Since the concept of decomposition is supported in the approach, process diagrams are used to illustrate how the different parts will be composed together. The interaction that takes place between the interfaces of different participating parties is specified using UML Sequence Diagrams. UML Sequence Diagrams enable the specification of the visible aspect of interactions, the messages. The provide concepts to "describe a sequence of messages" (OMG07) between several lifelines. (OMG07)

The concept model for the computational viewpoint of the MODEA approach is provided in Figure 35.

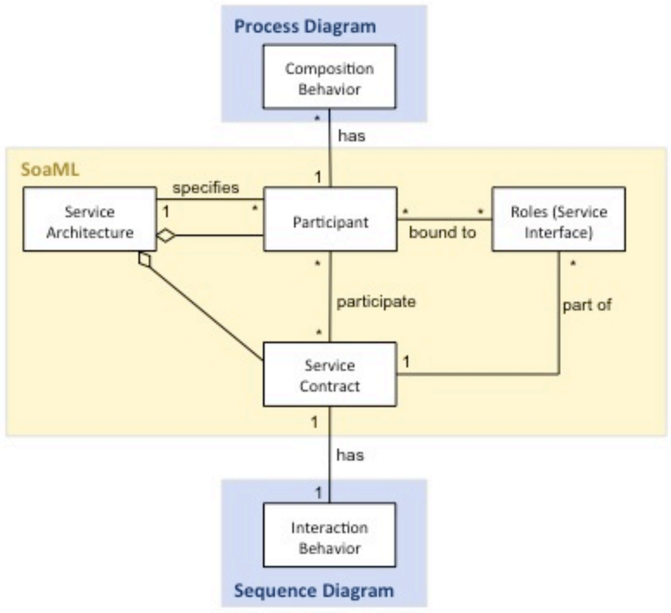


Figure 35 Concept Model of the Computational

The specification of the Computational Viewpoint in MODEA is completely conform to the contract-based SOA with its concepts of Participants, Service Architectures, Service Contracts, Providers and Consumers and Sequence Diagrams. Therewith Figure 35 is more or less a simplified conceptual model of the used part of SoaML.

The overall system is described as Service Architectures, which contains Participants and the Service Contracts between them.

The Service Contracts specifies the roles that can be played within the service. A role is represented as interface and can be either a provider or a consumer. The Participants collaborating in the Service have to bound to defined role of the Service Contract. A Service, specified through a Service Contract, is also related to an interaction behavior of the participating parties. These parties, represented as lifelines, are the roles that participate in the Service.

A Service Contract can also be composed of other Service Contracts. These nested Service Contracts provide a more fine-granular description of the service. Such a compound service does not represent an implementation through calling other service. This can be specified in a Participant Architecture or Process Diagram. (OMG12a)

The participants can be further described through a service architecture defining how the services are provided through internal parts or use of other services. Another possibility to describe the internal behavior of a participant is through the use of a process diagram like BPMN. Here the composition of required services and required internal actions are defined to describe how the functionality is provided. A schematic diagram illustrating the modeling approach in MODEA for the computational viewpoint is presented in Figure 36.

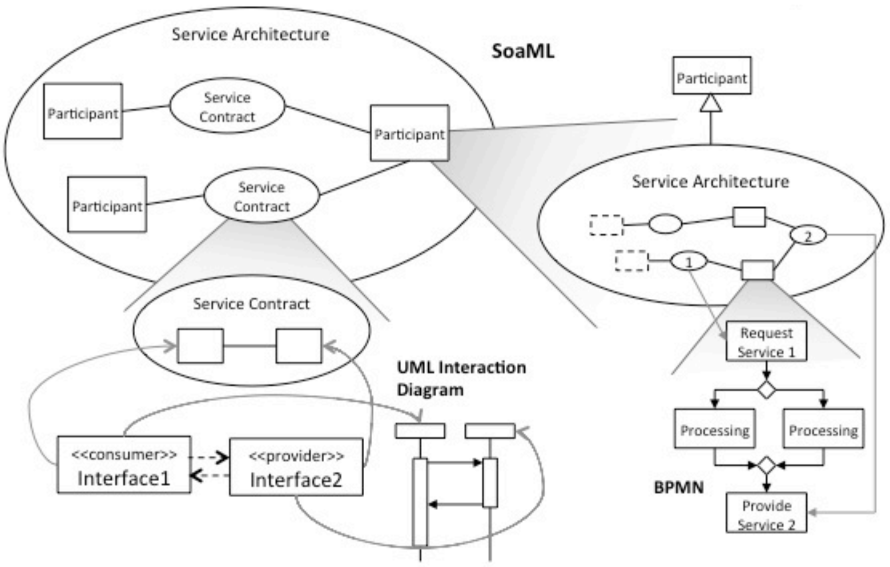


Figure 36 Schematic Overview of the Computational Viewpoint

The SoaML specification contains an interface-based approach and a service-contract-based approach. The difference between these two approaches is "whether the interaction between participants are defined separately from the participants in a ServiceContract [...] or individually on each partipiants' service and request" (OMG12a)

The use of the contract-based approach for SoaML enables a definition and usage of patterns of services. (OMG07) Compound services enable the specification of general design patterns, which can be then adapted to a specific context in the service architecture. For example a generic Service Contract for a Sale can be specified. This contract can then be used either for a retail sale or a whole sale, dependent on the participants that are bound to the buyer and seller role.

5.3.4 Engineering Viewpoint

The Engineering Viewpoint in RM-ODP answers the following questions:

- "How [does] distribution work[...]?"
- "How [are] objects distributed to nodes?"
- What is the structure of the nodes and the linking channels?
- What functions are required to support the required distribution transparencies?(Lin11)

The engineering viewpoint describes how the interaction between the objects defined in the computational viewpoint is achieved and what resources are required for this. These can be for example discovery services or request brokers. In this viewpoint the "supporting mechanisms for distributed interactions between objects [are identified and specified]" (Lin11) With the concepts of engineering objects, nodes, capsules, clusters and channels a technology-neutral architectural framework or reference architecture should be described. (ISO98a, Lin11)

Each application functionality identified as object in the computational viewpoint is represented as Basic Engineering Object in the engineering specification. Transparencies objects represent the platform functionality required to enable the distribution. The engineering objects communicate between each other through channels, which contain a specific protocol. In the engineering specification the distribution of these engineering objects to processing units, the so-called nodes, is defined. Inside these nodes engineering objects can be further grouped in capsules and clusters. Capsules own storage and a part of the node's processing resources, which are shared among the contained engineering objects. Clusters are the smallest grouping possibility for engineering objects with the goal to reduce the costs of manipulating the contained engineering objects. (ISO10b, Lin11) Figure 37 summarized the main concepts that are dealt within the engineering specification. It also illustrated a mapping to the modeling concepts used for them in MODEA.

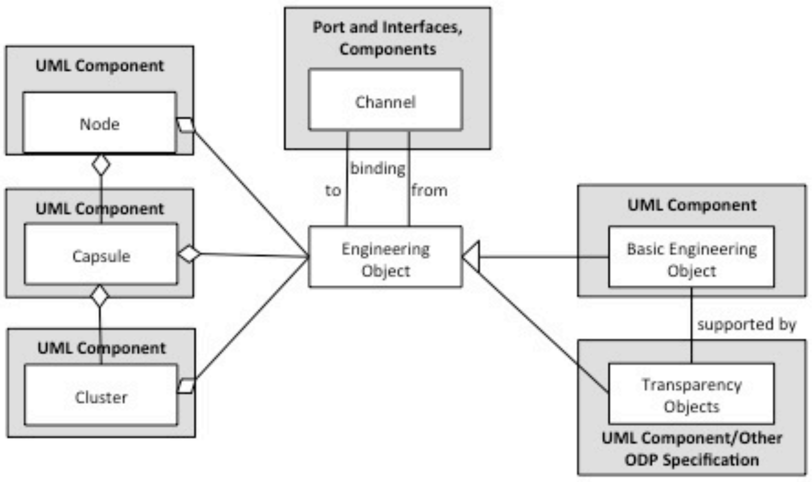


Figure 37 Mapping of RM-ODP concepts in the Engineering Viewpoint to MODEA modeling concepts (Own contribution based on Lin11, ISO98a)

In MODEA nodes, capsules, clusters and engineering objects are modeled as UML components. They are differentiated through their use on different abstraction layers. Their structure with the corresponding communication channels is modeled using UML Component diagrams. UML Component Diagram enables the specification of "software systems of arbitrary size and complexity" (OMG07). A component in this context is defined "as a modular unit with well-defined interfaces that is replaceable within its environment". There are two kinds of components: A Basic Component represents an executable element in a system. It has provided and required interfaces as well as at least one classifier realizing its behavior. The second one is Packaging Components. These components extend the Basic Components to be able to represent "'building block[s]' that may own and import a set of model elements." Components are connected through ports and interfaces as well as realization and usage dependencies. The component diagram is enables the specification of logical components but also physical components. Following a link to the logical description of the system as well as the deployment can be made. (OMG07)

Nodes represent the highest abstraction layer. If required they can be decomposed into capsules and clusters. The components representing basic engineering objects and transparencies objects are on the lowest abstraction level in the engineering specification. The communication channels between the various components are modeled through the use of ports and assembly connectors, linking required and provided interfaces. If necessary for the communication, components fulfilling any protocol processing steps or other communication functionality can also be included. The diagrams used in the MODEA approach to specify the engineering viewpoint as well as their main concepts and the relations between them are illustrated in Figure 38.

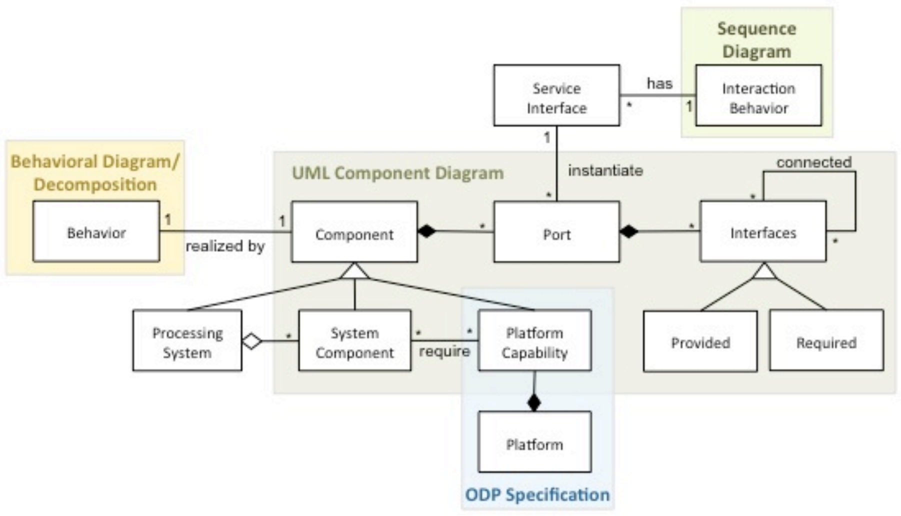


Figure 38 Concept Model of the Engineering Viewpoint

The essential element in the engineering viewpoint in MODEA is the component diagram. Here the system distribution using components is described. Components represent Processing Systems, Platform Capabilities from external Platforms and System Components. System components provide application functionality but also necessary functionality to support the distribution or communication in the OPD system. For example the replication transparency can be modeled with distributing the replicated system component on two processing systems.

Often when using Cloud Computing as infrastructure, external organizations or companies provide the distribution infrastructure. In this case "engineering specifications should be mapped to the specifications of the transparency mechanisms and common functions implemented and offered by the cloud provider." (Lin11) This is realized in MODEA through representing these platform capabilities from external providers as components. The platform itself can be described with an own enterprise architecture specification. The provided capabilities are integrated with use-relationships to internal system components.

The service contracts of the computational viewpoint are defined more technically in the Engineering Viewpoint using SoaML Service Interfaces or simple UML Interfaces in case of a one-way service. This is similarly to the approach from Sadovykh et al. (2010) described in chapter 4.2. In his approach the services with their contracts and choreography are defined in the Business Architecture Model. The Service Interfaces and Software Components are described in the System Architecture Model.

The communication between the components is modeled using UML Ports and UML Interfaces as well as SoaML Service Interfaces. A port describes a communication point at a component and is typed with an Interface, if it is a one-way-service, or with a Service Interface, if it is a two-way-service. The port at the component providing the service is labeled as <<service>>-Port, the port at the component requiring the service is labeled as <<request>>-Port. The provided and required interfaces at the port are then typed with the same interface as the port in case of a one-way-service and with the provider and consumer interfaces in case of a two-way service. The communication channels between the provided and required interfaces are modeled using assembly connectors. Ports can be also connected through a simple connector. For each port, that means for the interface that types this port, a more technically UML Sequence Diagram can be specified. This modeling approach relates to the Participant Architecture as defined in SoaML. (OMG12a)

The internal behavior of a component can be modeled with two ways. Either it is described using a behavioral diagram like State Machines or a Process Diagram or the structure is further refined through a decomposition of the component.

A schematic overview of how the viewpoint is specified is given in Figure 39.

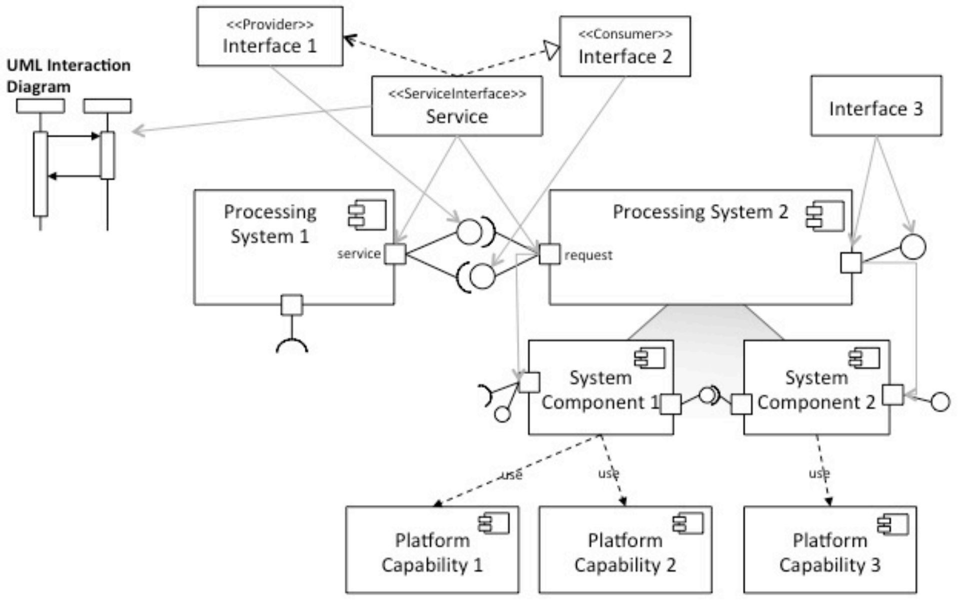


Figure 39 Schematic Overview of the Engineering Viewpoint

The schematic overview represents two processing systems, which communicate with each other through a two-way-service. The service and request ports at the processing systems are typed with the Service Interface of the Service. System 1 provides the service and therefore provides Interface 1 and required Interface 2, the consumer. System 2 is the consumer of the service. It required the providing Interface 1 and provides the consuming Interface 2. System 2 also provides a second service, typed by Interface 3, which is a simple one-way-service. Processing System 2 can be further decomposed using two system components. These components must together have the same ports as the processing system. The System Components use several external Platform Components to provide their functionality.

In the current UML specification the communication between the components works over interfaces and ports. Port, provided and required interfaces can be specified and connected through the assembly connector. This enables only the visible specification of one communication type, which represents a one-way communication. Two-way communications between components have to be modeled at the moment using a required and a provided interface at each component. This leads to a lot of labels and connections and blow the diagrams out of proportion. The diagrams with the labels as specified in the UML and SoaML Specification are shown in Figure 40. On the left side a one-way-service is shown and on the right side a two-way-service.

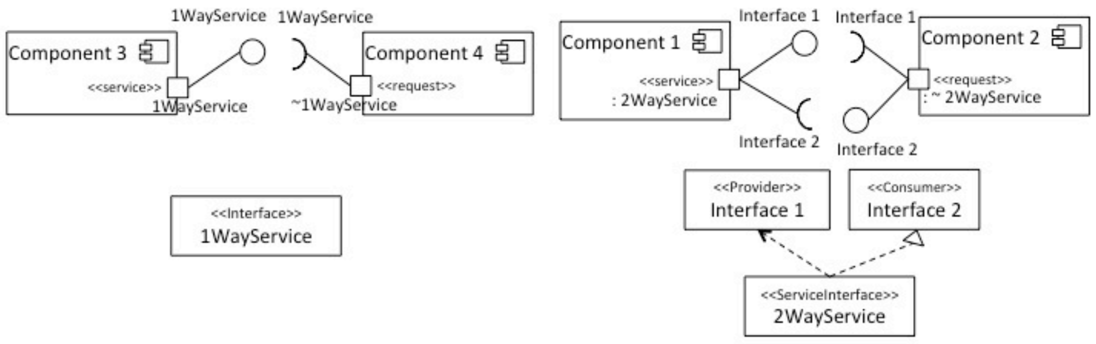


Figure 40 Labeling in the Component Diagram

Through labeling all visible elements the diagrams get hard to understand. Therefore the recommendation, especially for more complex diagram containing a lot of components, ports and interfaces, is to hide the interface and port labels. Only the assembly connector will be labeled with the service name and in the case of an two-way-service the service and request ports have to labeled. This recommendation is presented in Figure 41.

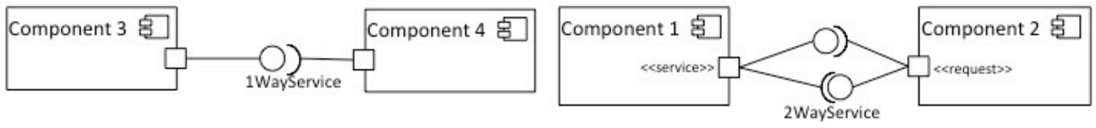


Figure 41 Proposed labeling in the Component Diagram

Therefore the recommendation is not to label all single interfaces.

5.3.5 Technology Viewpoint

The Technology Viewpoint in RM-ODP answers the following questions:

- How to consider "the IT infrastructure already available in the company [and] their budget requirements"?
- How to align to the "existing commercial policies or strategies that might force (or forbid) the use particular vendor technologies"?

(Lin11)

The technology viewpoint provides the link between the other four viewpoints and the real implementation. It describes the hard- and software components of the implementation as well as the constraints in terms of costs and availability of existing products. Also possible standards the system should be conform to are integrated. (ISO98a)

The four main concepts are shortly explained in the following table.

Technology Objects	Hardware devices like PCs, servers, ATMs, printers Operating Systems and applications like browsers, text editors Connections like LANs, WANs, intranets
Implementable standards	Templates for the technology objects Represent standards to which the objects must be conform

Implementation	Defines the activity of instantiating the specification Includes development, deployment, configuration and evolution processes
IXIT	Provides extra information for conformance testing

Table 15 Concepts of the Engineering Viewpoint in RM ODP (Own contribution based on ISO10b)

The implementation processes can be specified using process diagrams like BPMN or UML Activity Diagrams. Both enable the specification of single actions and the integration of different process flows. With regard to a possible automation of these processes BPMN should be preferred, since there support for automation is given using BPEL.

The technology objects and their structure are represented in MODEA with the UML Deployment Diagram. The Deployment package of UML provides "constructs that can be used to define the execution architecture of systems that represent the assignment of software artifacts to nodes" (OMG07). With the concept of Nodes, with possible nesting, communication channels and artifacts the technical specification will be described.

Each technology object is represented as a Node in a UML Deployment Diagram, which is very similar to the approach described in UML4ODP. The interfaces between the technology objects are represented through communication paths between the nodes. The implementable standards can be integrated as artifacts, which have dependencies to the nodes the must be conform to them. Figure 42 gives an overview of the diagrams and concepts used in MODEA to specify the Technology Viewpoint. The process diagram for the implementation process is not included in the figure.

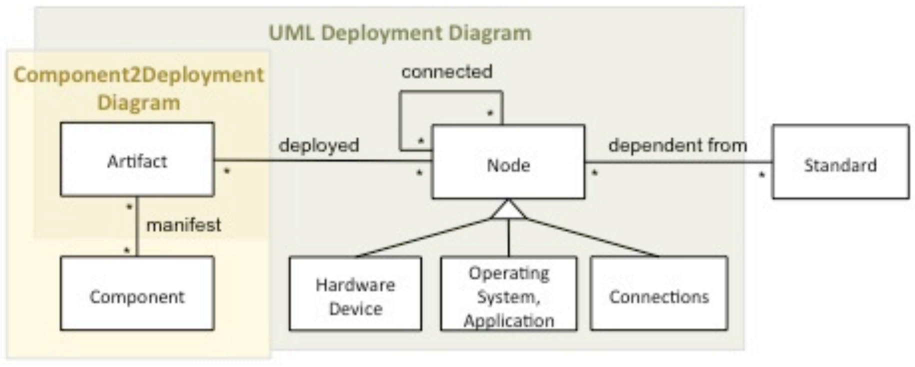


Figure 42 Concepts of the Technology Viewpoint

At this moment we will not go in detail of the specification for this viewpoint, since it should be generated in main parts out of the other specifications in the future. The concepts related to conformance testing are not examined in this report.

A schematic overview of the specification of the deployment structure and the implementation process is given in Figure 43.

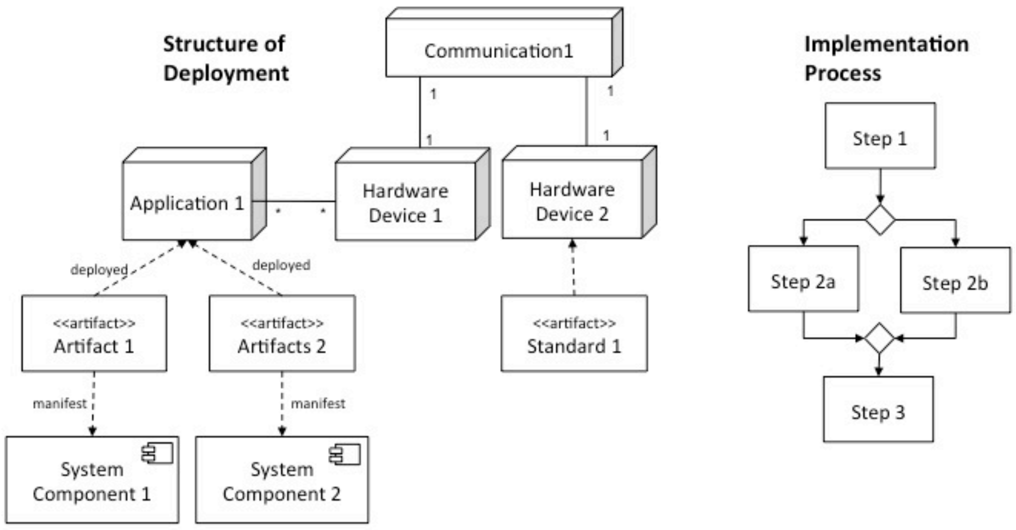


Figure 43 Schematic Overview of the Technology Viewpoint

Each system components defined in the Engineering Viewpoint is manifested through an Artifact. The Artifacts are then deployed to Nodes, which can be for example Applications, Hardware Devices or Communication Channels. Standards are represented as artifacts, since they are also “concrete elements in the physical world that are the result of a development process.” (OMG07). A dependency-connector links the standard to the appropriate node.

5.4 Connections between the viewpoints

If a system is specified with the use of different viewpoints and different language for each viewpoint the consistency between the various diagrams are a major issue. Therefore it is necessary to define the relationships between the viewpoints. They will support the designers to keep the overall specification consistent. The relationships between the model elements in MODEA are identified based on that ones defined in the RM ODP specification and on the relationships between the different UML and SoaML packages.

First an overview of the relationship between all the model elements defined in the five viewpoints above is given in Figure 44. Specialized children of the elements are hidden as well as the multiplicities and labels between the elements of one viewpoint.

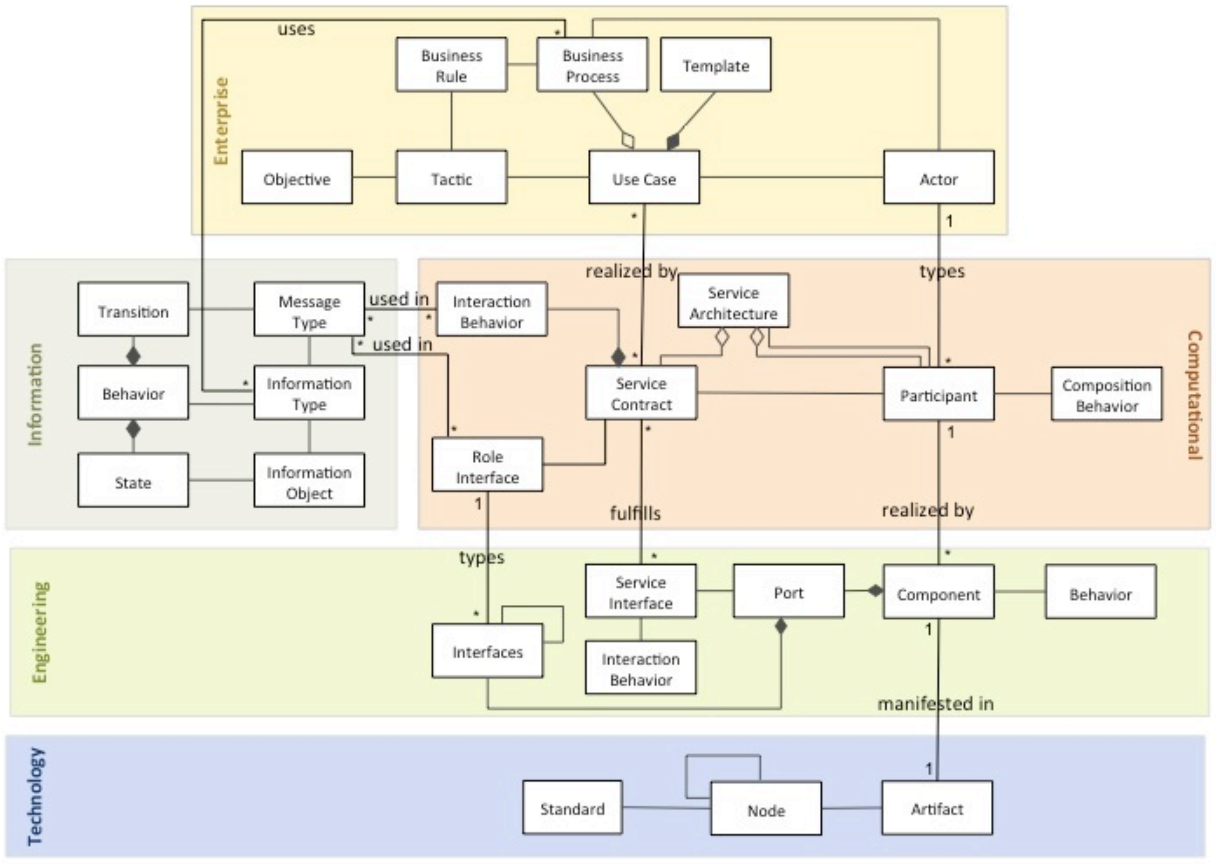


Figure 44 Relationships between the Viewpoints in MODEA

The model elements between the different viewpoints are connected through the use-connector, the realization-connector or through typing of the element.

Enterprise Viewpoint

The Enterprise Viewpoint is related indirectly to all other viewpoint. The goals and objectives described here must be considered in the overall specification. A viewpoint is “consistent with an enterprise specification if all roles, activities and policies described in the enterprise specification are correctly reflected”. (ISO98a)

For example the behavior of an information type or a participant must always obey the policies and rules defined in the Enterprise Specification. Flexibility requirements or policies must be considered in the choice of technology for the implementation of the system. Transparency needs as well as security and performance issues defined as requirements in the Enterprise

Specification have to be considered in the Engineering Viewpoint. (ISO98a) If any object is related to a specific requirement, specified as business rule, use case or tactic in the Enterprise Specification, then these two elements can be connected with the Realization or Dependency-Connector in UML.

Enterprise, Information and Computational Viewpoint

But there are also direct links from the Enterprise Viewpoint to the Information and Computational Viewpoint. In the Business Process the Information Types are used for the data objects and also to annotate the message flows with objects.

An actor in the Enterprise Specification, which is also linked to a pool in the Business Process, will be mapped to a participant on the highest abstraction levels. These participants can then be further refined. The same will be done with a Use Case in the Enterprise Specification. At the highest abstraction level in the Computational Specification, this will be directly mapped to a service. In the most cases this is then a Business Service. Further refinement steps will identify more technical services required to implement the business service.

Berkem (2008) also proposes a one-to-one mapping between use cases and services. He also includes the Business Processes in the one-to-one mapping. Tactics act as a façade for the goal-driven-services that are mapped to use cases in his approach. In MODEA Tactics, Use Cases, Business Processes and Business Services are linked in a similar manner to each other. The exact cardinalities between these concepts are dependent from the project type and development process. Following MODEA supports a one-to-one mapping between these four elements as it is in Berkem (2008), but there is also support for specifying one Business Process for several Use Cases or link a Use Case to more than one service.

The information viewpoint provides a shared vocabulary for all the other viewpoints through the definition of the used information and message types. The information types are important for the Business Processes in the Enterprise Specification. Each state change in the Information Specification, triggered by some Messages Types, also known as actions, has some correspondence in the computational viewpoint. In most cases it corresponds to some message flow in the sequence diagram of a service contract. But they can also be linked to some internal actions of a participant.

The Computational Viewpoint, typically created when some first results for the Enterprise and Information Viewpoint are specified, has links to the Enterprise, Information and also Engineering Viewpoint (Lin11). An schematically overview of the explained relationships between the Enterprise, Information and Computational Viewpoint is given in Figure 45.

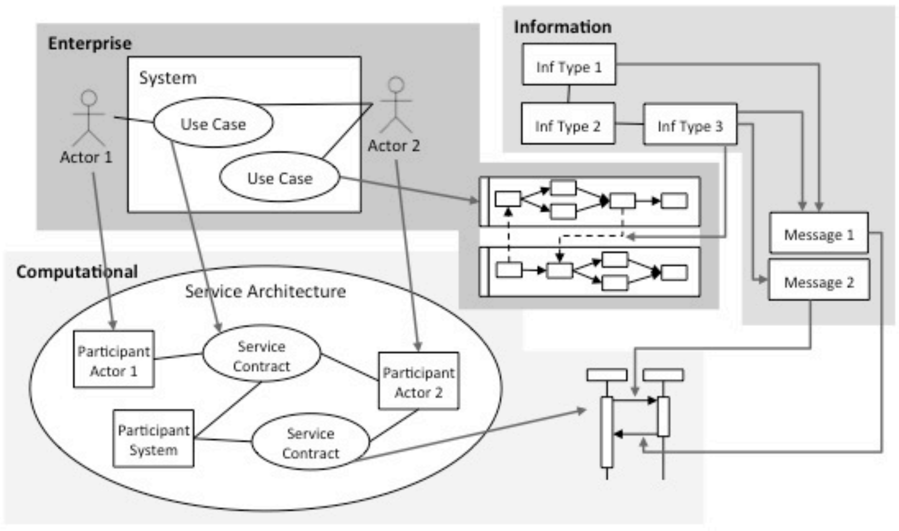


Figure 45 Schematic relationships between the model elements of the Enterprise, Information and Computational Viewpoint

To ensure that the computational specification fulfills the enterprise specification, the use cases and actors are mapped in a first step to participants and service contract using a one-to-one mapping. Then this system overview can be further detailed to specify the functional decomposition required to fulfill the requirements. This support the top-down approach to define the enterprise specification presented in (Lin11). When existing components or functionality has to be integrated in the system a Bottom-up approach is recommended (Lin11). This is support through encapsulate the existing component or functionality using a participant and providing it over a service. The so defined components are then composed together to fulfill the requirements in the Enterprise Specification.

With both approach the used information in messages of the sequence diagrams and also in the interface definition for the roles has to be defined in the information specification.

Computational and Engineering Viewpoint

On the RM ODP specification each computational object corresponds to at least one basic engineering object in the engineering specification. The process of getting from the computational specification to the engineering specification “may simply consist of the identification of suitable supporting objects to populate channels that represent binding objects in the computational specification”. (ISO98a)

In MODEA this concept is realized through the introduction of one system component for each participant at the lowest abstraction level. The distribution of the components in the engineering specification can be the same as the functional decomposition hierarchy in the computational specification but they can also be different from each other.

The relationships between the interactions parts of the two specifications are done as followed. Each service contract will be fulfilled by a service interface. This service interface types the port from the component representing a participant of the service. The provided and required interfaces at this ports define the role, the component plays, through implementing the provider and consumer roles defined in the computational specification. This mapping of service architectures and participants to components and the mapping of the interaction elements is shown in Figure 46.

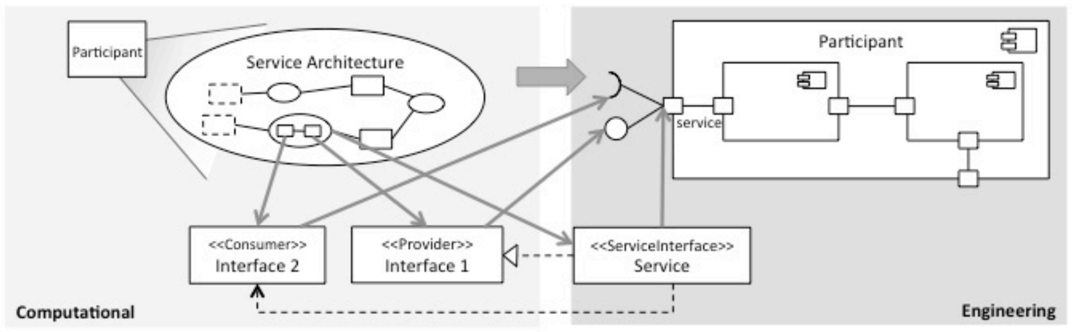


Figure 46 Schematic relationship between Computational and Engineering Viewpoint

The sequence diagrams defined in the computational viewpoint for the interaction in one service can be extended in the engineering viewpoint with more technical details or protocol information. Therefore also components with functionality for the communication or transparency specification can be introduced in the engineering viewpoint.

In the RM ODP specification the interaction defined in the engineering specification is restricted by the fact that is has to "start[...] and end [...] with an interaction involving one or more of the basic engineering objects corresponding to the interacting computational objects" (ISO10b). This means that the sequence diagram defined in the engineering specification for a service interfaces must at least contain the same lifelines and messages as the corresponding sequence diagram from the service contract. It also has to start and end with an interaction from these "original" participators. But with considering these restrictions the sequence diagram can be extended with further lifelines, for example a broker, a further interactions between the lifelines. This relationship is illustrated in Figure 47. The grey shaded lifelines represent the participants of the service contract in the computational viewpoint. In the engineering specification a third actor will be included. This could be for example necessary for some transparency or security reasons.

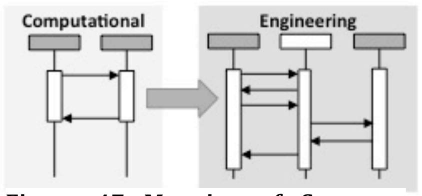


Figure 47 Mapping of Sequence Diagrams

Engineering and Technology Viewpoint

In the technology specification each technology object has a corresponding atomic or composite engineering object or a channel in the computational specification (Lin11). In MODEA each component from the computational specification corresponds to an artifact in the technology specification using the manifest-relationship. The service interfaces of the computational viewpoint, specifying the communication between the components, are also mapped to artifacts using the manifest-relationships. The artifacts are then deployed to appropriate nodes, which represent Applications, Hardware Devices or Communication Mediums.

5.5 Methodology for MODEA

In the previous chapters the modeling approach is introduced through an introduction of the concepts used to describe each of the five RM ODP viewpoints. Now one methodology of how to create the specifications in a development process is recommended in this chapter. This basic methodology can be supported by most of the development processes for software engineering like agile methods, iterative methods or waterfall methods. Therewith the propositions given by Linington et al. (12) will be taken into account.

The process typically starts with the definition of a first draft of the enterprise and information specification. After that or in parallel with them, the computational specification is created. (Lin11)

The computational specification can be defined using a bottom-up approach or a top-down approach. When using a top-down approach the computational objects, that mean the services and participants in MODEA, are derived from the enterprise and information specification. In case of a bottom-up approach the computational objects encapsulate existing functionalities and are then composed together. (Lin11)

The next step is to define the engineering specification. Therefore a first version of the computational specification is required, since the engineering viewpoint is build based upon the identified system parts in the computational specification. For the creation of the engineering specification first the mapping of the elements in the computational specification to components, ports and interfaces has to be done. Then additional required functionality for communication, security and distribution can be integrated.

At least the elements of the engineering specification have to mapped to that ones in the technology specification. The parts of technology specification describing hardware and communication mediums can also be defined earlier in the development process, as the provide also restrictions to the upper viewpoints.

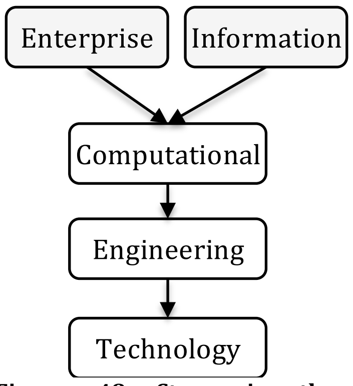


Figure 48 Steps in the development process

Figure 48 shows the sequence of creating the different viewpoints. The development process described here can be supported through model generation mainly at three points:

- Generate a high-level service architecture of the computational specification
- Generate a first draft for the component model in the engineering specification
- Generate the technology specification

After defining the Use Case Diagram in the Enterprise Specification, a generation of the high-level service architecture in the Computational Viewpoint can be automatically generated. Therefore for each actor as well as the system subject a participant in the service architecture is created. For each use case a service contract is defined. The system participant attends in each of the defined service contract. The corresponding participant of an actor takes part in those service contracts, where the actor is linked to the appropriate use case.

If the computational specification is finished, a first draft of the component diagram in the engineering specification can be generated. Each service architecture is mapped to a component; the contained participants are mapped to subcomponents. For each service contract a service interface is created which is used to type the port at the corresponding component. The roles defined in the service contracts are used to define the provided and required interfaces at the ports. This generated component diagram can be used to integrate the required components and functionalities for communication, distribution and security.

At least the technology specification can be generated from the engineering specification, a specification of the used infrastructure and some deployment rules. First the components in the engineering specification are represented as artifacts. Using the defined rules the artifacts are linked to the corresponding elements of the specified infrastructure.

5.6 Tool Support

In MODEA only a small number of different modeling techniques is used. These are UML 2.0 and BPMN as well as the UML profiles BMM and SoaML. Since UML 2.0 and BPMN are well-established modeling techniques there is a quite good tool support for at least the modeling part of the diagrams. An overview over the used diagrams in each viewpoint of MODEA is given in Table 16.

Viewpoint	Used Modeling Technique
Enterprise	OMG Business Motivation Model, UML Use Case Diagram, BPMN
Information	UML Class Diagram, UML Object Diagram, UML Behavioral State Machines, SoaML Message Types
Computational	SoaML, UML Sequence Diagram, BPMN
Engineering	UML Component Diagram, UML Sequence Diagram
Technology	UML Deployment Diagram, BPMN

Table 16 Overview of the diagram types used in MODEA

Each tool that provides at least support for UML 2.0 and BPMN can be used for MODEA. SoaML is an UML profile and if a modeling tool does not provide a direct support, it can be easily integrated through the use of the appropriate stereotypes. Also BMM, although there is no UML profile, can be easily used through serotyping since there are only a few simple concepts and associations. However to choose a tool that provides also support for BMM and SoaML makes it easier. Examples for tools providing support for UML 2.0, BPMN and BMM are Modelio, Sparx System Enterprise Architect, IBM Rational Architect or Select Architect. The first three one provide also support for SoaML. (Mod11, Sparx12a, Sel12, IBM12)

At the moment there is no tool providing directly support for the model generation proposed in the previous chapter or enables code generation out of the specification.

Sadovykh et al. (2010) present in their paper an approach to "transform[…] from the SoaML SAM to various platforms including Web Services, Multi-Agent Systems and Semantic Web Services platforms". The System Architecture Model SAM contains definitions of the service interfaces, service choreographies, interfaces and messages, software components and service orchestrations. They use the Modelio CASE tool to generate conventional Web Services, Java Persistency and SQL.

6 Application of MODEA

In the previous chapter a model driven approach for open distributed systems using an enterprise architecture framework, MODEA, was introduced. In the following, this approach will be applied to the two example cases introduced in chapter 2.

6.1 Pilot 1: ENVIROFI PEIS

An introduction to the PEIS project was already given in chapter 2.1. In the next section some specific characteristics of the projects are shortly explained and in the following a partial specification of PEIS to illustrate the use of MODEA is shown.

6.1.1 Specific Characteristics

In the project the service-oriented architecture paradigm will be used. It will be applied with the concept of a Multi-Style-SOA. This enables the usage of multiple architectural styles and communication patterns, like event-driven communication, synchronous request/reply messaging, asynchronous message oriented, stream oriented and resource-oriented communication. (PEIS 4.2)

The PEIS system is provided through the usage of functionalities from existing platforms. These are the FI Ware platform and the ENVIROFI platform. They provide their functionality with use of so called enablers. (PEIS 4.2)

An enabler is “a software component in implementation architecture with a well-defined interface that fulfills a given set of functional, informational and qualitative requirements” (PEIS 4.2). An enabler can be a specific enabler for a domain or a generic domain-independent enabler. The ENVIROFI platform provides the specific enablers, whereas the FI WARE platform provides the generic enablers. For performing the functionality the enablers require features, which are mainly resources of environmental data. They are often provided or referenced in the operation parameters. An enabler can also provide its functionality by using or composing other enablers. In the PEIS project the existing enablers should be reused and integrated to implement the use cases. (PEIS 4.2)

The specific enablers provided by the ENVIROFI Platform are classified in environmental enablers and geospatial enablers. The geospatial enablers relate to geospatial services and data models, which are existing or emerging standards from OGC and ISO/TC211. Environmental enablers are built on top of them, tailored to the various environmental disciplines. Both enabler categories provide its functionality using the generic enablers from the FI WARE core platform. The relationship between these three enabler categories is shown in Figure 49. The arrows symbolize use-relationships between the different categories. Environmental Enablers use Geospatial Enablers and Generic Enablers. Geospatial enablers refer also to Generic Enablers. (PEIS 4.2)

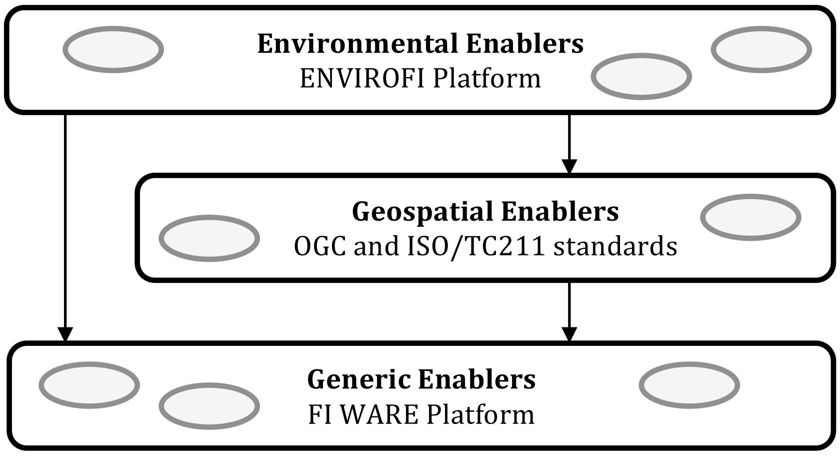


Figure 49 Enablers in ENVIROFI (Own contribution based on PEIS 4.2)

The three types of enablers are each categorized by thematic issues. At the moment there is no common approach for classifying the enablers in ENVIROFI and FI WARE. Possible suggestions for a classification are made in D4.1.1:

- Lifecycle based Approach derived from CEN Technical Report TR15449
- Bus or layered Architecture-based Approach based on ISO 19119.

In the following the classification approach based on ISO 19119 is used. These different groups of services in this classification approach are shown in Figure 50. The different colors proposed in this figure will be used to categorize the service based on this classification.

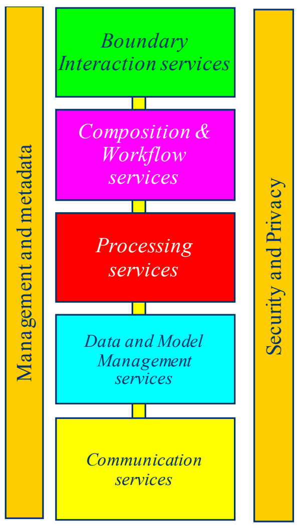

Management and metadata
Boundary Interaction services
Composition & Workflow services
Processing services
Data and Model Management services
Communication services
Security and Privacy

Figure 50 Architecture-Based Classification Approach (ISO05)

6.1.2 Architecture Specification

The architectural specification of PEIS is made with use of the deliverables D4.2 describing the environmental architecture, D2.1 defining the scenarios and use cases for PEIS as well as D2.3.2 containing the functional and organizational specification of PEIS.

Enterprise Viewpoint

In the following the Enterprise Viewpoint of the MODEA approach is applied to the PEIS Pilot. First the diagrams from the enterprise perspective are introduced. Figure 51 shows the Business Motivation Model for the PEIS System.

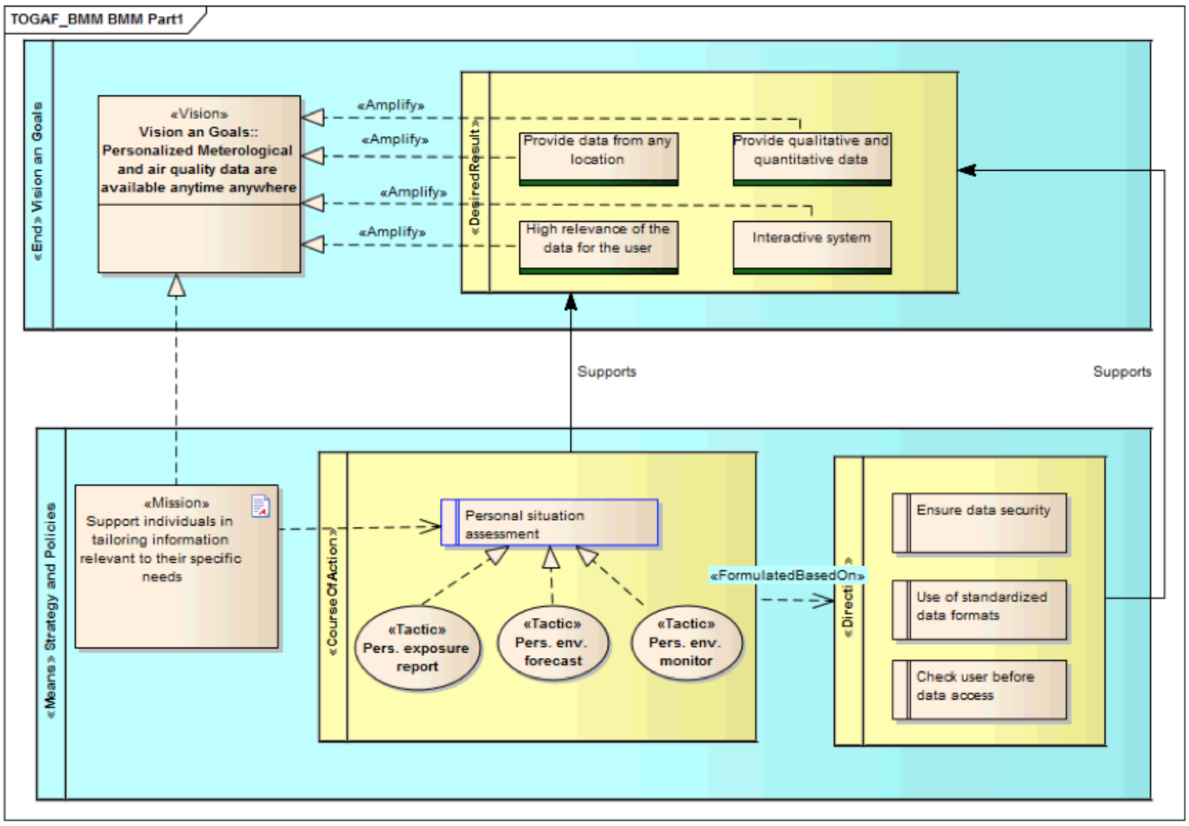

TOGAF_BMM BMM Part1
«End» Vision an Goals
«Vision» Vision an Goals:: Personalized Meterological and air quality data are available anytime anywhere
«Amplify»
«Amplify»
«Amplify»
«Amplify»
«DesiredResult»
Provide data from any location
Provide qualitative and quantitative data
High relevance of the data for the user
Interactive system
Supports
Supports
«Means» Strategy and Policies
«Mission» Support individuals in tailoring information relevant to their specific needs
«CourseOfAction»
Personal situation assessment
«Tactic» Pers. exposure report
«Tactic» Pers. env. forecast
«Tactic» Pers. env. monitor
«FormulatedBasedOn»
«Directi
Ensure data security
Use of standardized data formats
Check user before data access

Figure 51 PEIS - Business Motivation Model

The overall vision of the system is to make personalized meteorological and air quality data available anytime and anywhere. More details are given through the four goals, which refine the overall vision. For example one goal is to provide data from any location another one is to provide qualitative and quantitative data. Additionally the data should have a high relevance for the user and the system should be designed to be interactive.

To reach the vision with its goals, the system should provide support for individuals in tailoring information to their specific needs. This is the mission of the system. One strategy for this mission is to create a personal situation assessment for the user. This will be done by the creation

- of a personal exposure report, which is concerned with past events and data,
- of a personal environmental forecast of future events and data and
- of a personal environmental monitoring of the current events and data.

When realizing these three tactics there a several business rules, which must be taken into account. One is to ensure data security, especially for the user data. Furthermore the use of standardized data formats and a mandatory check of the user data before data access are business rules, which have to be considered in design and implementation.

The business rules and tactics are realized in the first step through use cases. For example the tactic of 'creating a personal exposure report' is realized by the two use cases 'display past meteorological conditions and events' and 'display past exposure to air pollution and pollen'. This relationship as well as all other ones that exist between tactics, business rules and use cases is shown in Figure 52.

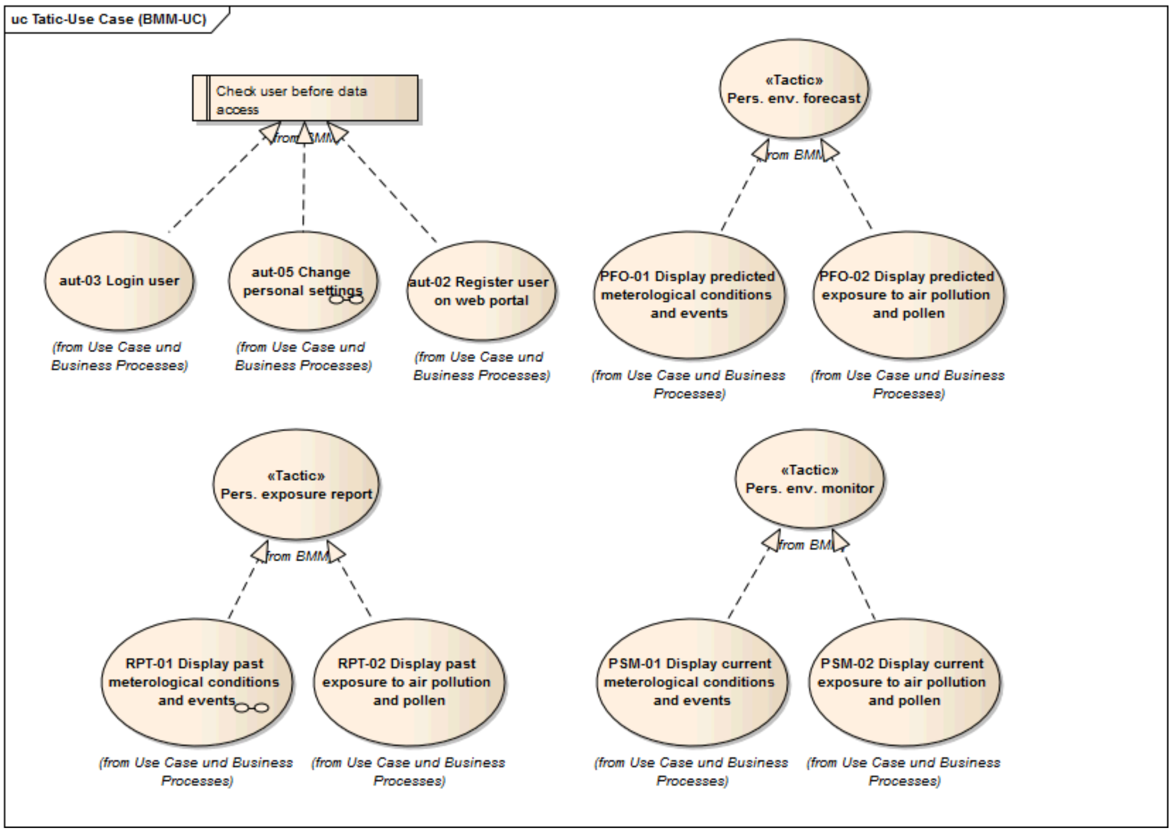

uc Tatic-Use Case (BMM-UC)
Check user before data access
«Tactic» Pers. env. forecast
from BMM
aut-03 Login user
aut-05 Change personal settings
aut-02 Register user on web portal
PFO-01 Display predicted meterological conditions and events
PFO-02 Display predicted exposure to air pollution and pollen
(from Use Case und Business Processes)
«Tactic» Pers. exposure report
«Tactic» Pers. env. monitor
RPT-01 Display past meterological conditions and events
RPT-02 Display past exposure to air pollution and pollen
PSM-01 Display current meterological conditions and events
PSM-02 Display current exposure to air pollution and pollen

Figure 52 PEIS – BMM2Use Case Diagram

This diagram is not a UML 2.0 specified diagram, it is just for making the realization-relationship between tactics, business rues and use cases visible. For example the use cases 'Login User', 'Change personal settings' and 'Register user on web portal' are realizing the 'Check user before data access' business rule. The tactic to create personal environmental reports is realized by the use cases 'Display past meteorological conditions and events' and 'Display past exposure to air pollution and pollen'. The business rule 'check user before data access' becomes necessary, when retrieving environmental data. Therefore the rule is realized in the use cases 'display past, predicted and current meteorological conditions and events' as well as the 'exposure to air pollution and pollen'.

The defined use cases with their relationships to each other and the participating actors are shown using a UML Use Case Diagram. Figure 53 shows this diagram for the PEIS System.

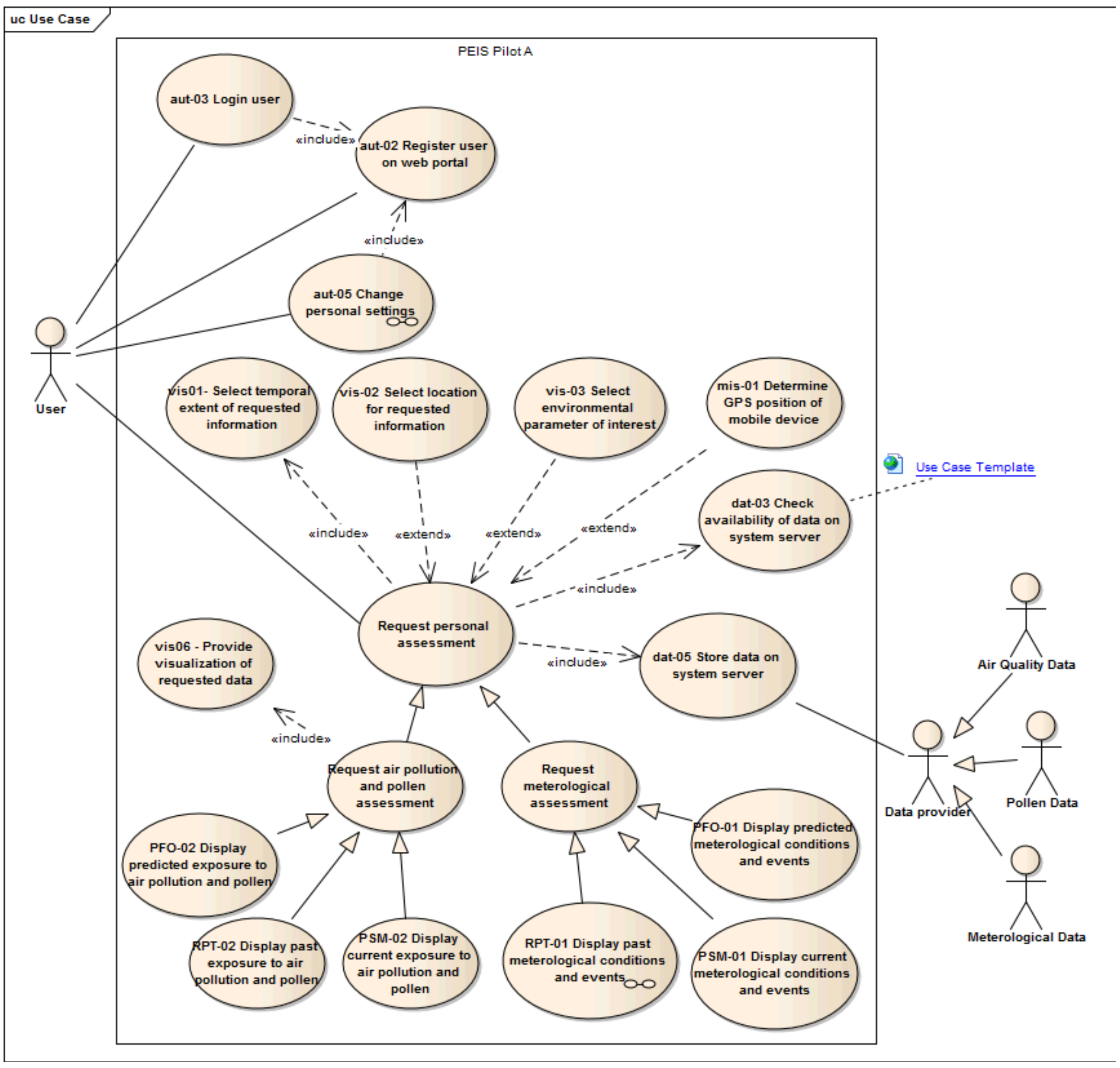

uc Use Case
PEIS Pilot A
aut-03 Login user
«include»
aut-02 Register user on web portal
«include»
aut-05 Change personal settings
User
vis01- Select temporal extent of requested information
vis-02 Select location for requested information
vis-03 Select environmental parameter of interest
mis-01 Determine GPS position of mobile device
Use Case Template
dat-03 Check availability of data on system server
«include»
«extend»
«extend»
«extend»
«include»
Request personal assessment
«include»
dat-05 Store data on system server
vis06 - Provide visualization of requested data
«include»
Request air pollution and pollen assessment
Request meterological assessment
Air Quality Data
Data provider
Pollen Data
PFO-02 Display predicted exposure to air pollution and pollen
PFO-01 Display predicted meterological conditions and events
RPT-02 Display past exposure to air pollution and pollen
PSM-02 Display current exposure to air pollution and pollen
RPT-01 Display past meterological conditions and events
PSM-01 Display current meterological conditions and events
Meterological Data

Figure 53 PEIS - UML Use Case Diagram

In PEIS there are mainly two groups of actors. One is the application user and ones are the environmental data providers. These data providers comprise meteorological, air quality and pollen data. For example the use case 'RPT-01 Display past meteorological conditions and events' is generalized by the 'request meteorological assessments' use case, which is itself a specialization of the 'request personal assessment use case'. This use case hierarchy is only used for structuring the use cases and makes the diagram more readable. This is also the reason why this generalized use cases do not have a use case identifier. The general use case 'report personal assessments' includes several use cases that deal with the retrieval of the required user data and the handling with the environmental data.

Each of the use case in the above diagram with an identifier is described more detailed using a predefined use case template. For example a cut-off from template from the use case dat-03 'Check availability of data on system server' is shown in Figure 54.

Goal	Check availability of data on system server for required temporal and spatial extent
Summary	While making use of PEIS, the user requests data of one or more atmospheric parameters for a specific temporal and spatial extent. The system checks the available internally.

Category	Data Access
Actor	All
Primary Actor (initiates)	User
Stakeholder	
Preconditions	User is logged in UC-ENV1.1-auth-01-V01
Triggers	User requests data of one or more atmospheric parameters for a specific temporal and spatial extent
Main success scenario	• The system checks the availability of the requested data • The system informs the user of the outcome
Extensions	The data originator is informed that their data has been accessed.
Alternative paths	
Post conditions	The system provides the user the requested data
Author and date	UBIMET, 2011-09-09

Figure 54 PEIS - Use Case Template dat03 Check availability of data on system server (Excerpt from PEIS 2.3.1)

Main parts describing the details of the use case are the goal, the summary, the actors, the preconditions, triggers and the main success scenario. At least also the author of the use case and the date are provided. In the next step a use case will be further defied using collaboration diagrams. Figure 55 visualizes the business process in the Use Case 'RPT-01 Display past meteorological conditions and events'. Included use cases are referenced through sub processes.

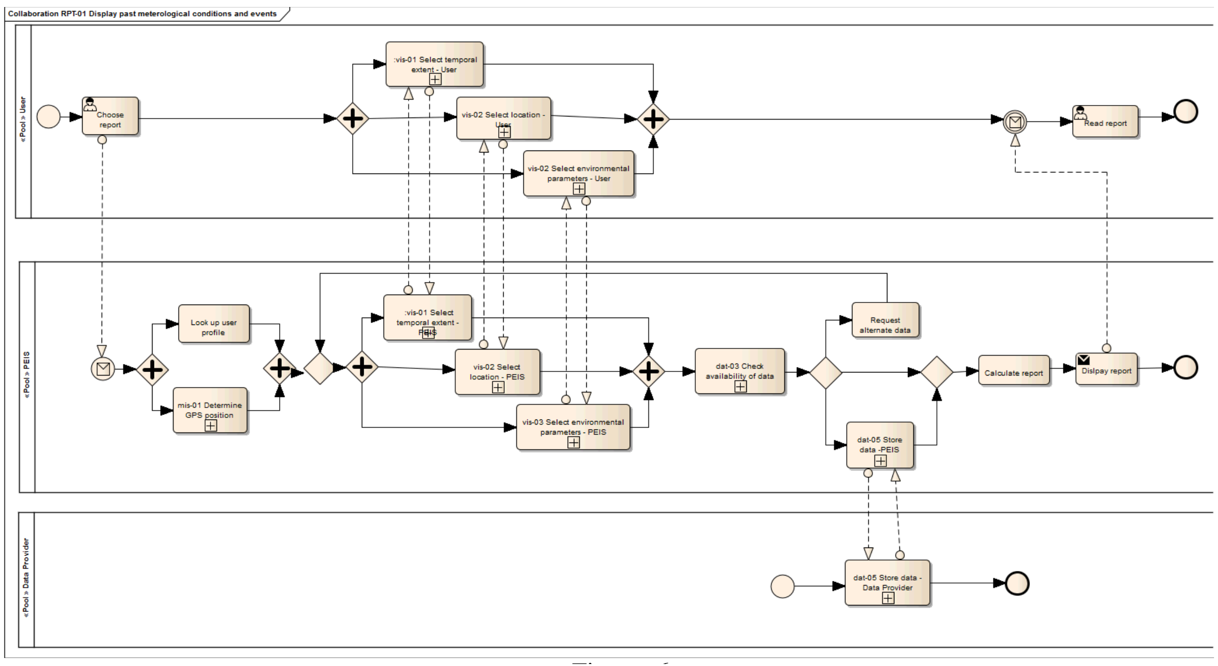

Collaboration RPT-01 Display past meterological conditions and events
«Pool» User
Choose report
:vis-01 Select temporal extent - User
vis-02 Select location - User
vis-02 Select environmental parameters - User
Read report
«Pool» PEIS
Look up user profile
mis-01 Determine GPS position
:vis-01 Select temporal extent - PEIS
vis-02 Select location - PEIS
vis-03 Select environmental parameters - PEIS
dat-03 Check availability of data
Request alternate data
dat-05 Store data -PEIS
Calculate report
Dislpay report
«Pool» Data Provider
dat-05 Store data - Data Provider

Figure 55 PEIS - BPMN Collaboration RPT-01 Display past meteorological conditions and events

The pools in the collaboration diagram represent the participating actors in the use case. They are the user, the data provider and the PEIS system itself. The basic flow is to retrieve first the required user data as well as time location and parameters. Then the availability of the required environmental data is checked. If necessary the data is imported into the system and then processed to provide the report to the user.

Several sub processes referencing other included or extended use cases have to be integrated in the process of Figure 55. Therefore a BPMN collaboration for the included Use Case is created. The activities of each actor in a pool are grouped together in an actor-specific sub process for this use case. In the Use Case that includes the other one, these sub processes can then be reused. For example the Use Case ‘Select temporal extent’ is included in the Use Case ‘Display past meteorological conditions and events’. To make this visible the process steps are hidden through usage of a sub process. The definition of the sub processes is shown in Figure 56.

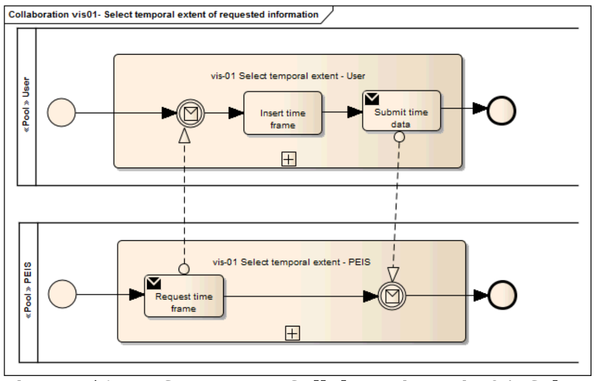

Collaboration vis01- Select temporal extent of requested information
«Pool» User
vis-01 Select temporal extent - User
Insert time frame
Submit time data
«Pool» PEIS
vis-01 Select temporal extent - PEIS
Request time frame

Figure 56 PEIS – BPMN Collaboration vis-01 Select temporal extent

Information Viewpoint

The information types used in PEIS are shown in the class diagram in Figure 57.

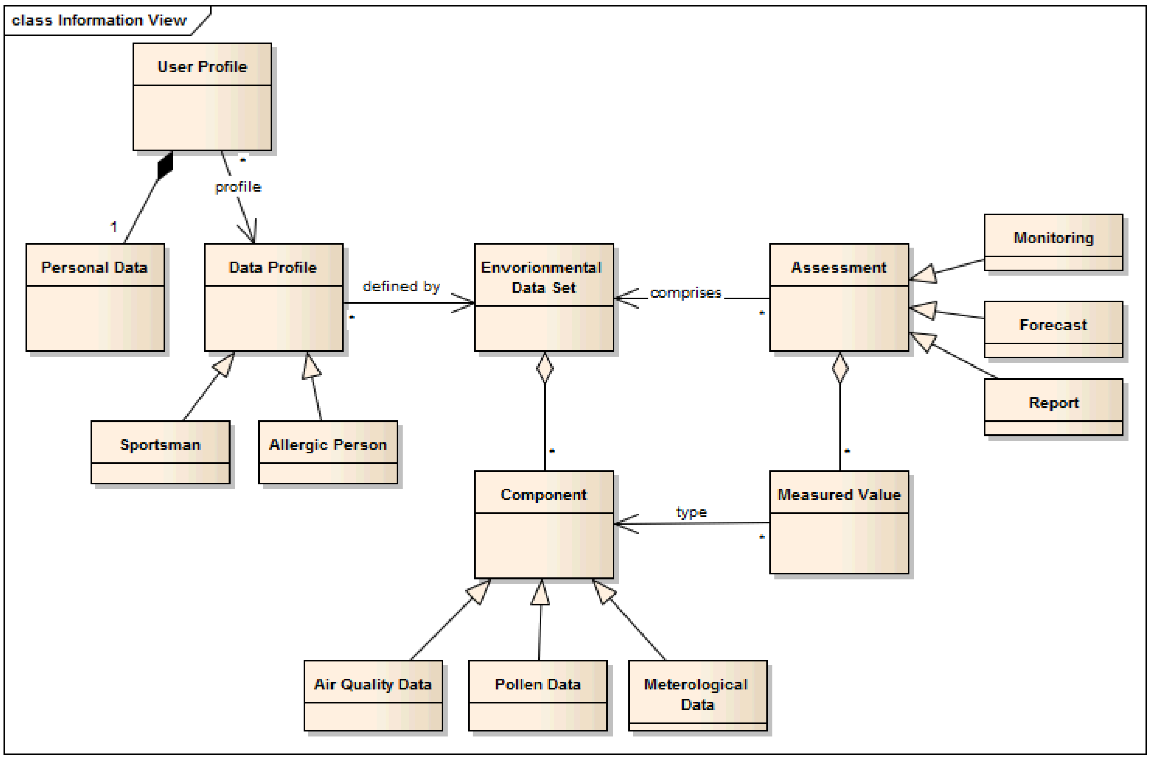


Figure 57 PEIS - Information Type Model

The information types are identified with help of the process diagrams in the enterprise viewpoint. The information types will be derived from the information objects sent on the messages flows between the tasks in the process diagram in Figure 55. In PEIS these are the user profile with personal data and an environmental data profile. In the current version this can be a sportsmen profile or an allergic person profile. Furthermore there is the type of an environmental data set, which consists of several components, which can be Pollen Data, Meteorological Data or Air Quality Data. An assessment is linked to an environmental data set and consists of several measured values. Each value is related to a component, which is part of the related data set. An assessment can be a repot about past data, a forecast for future data and also a monitoring of the current situation.

Figure 58 represents the invariant schema of the User Profile Information Type with an UML State Machine.

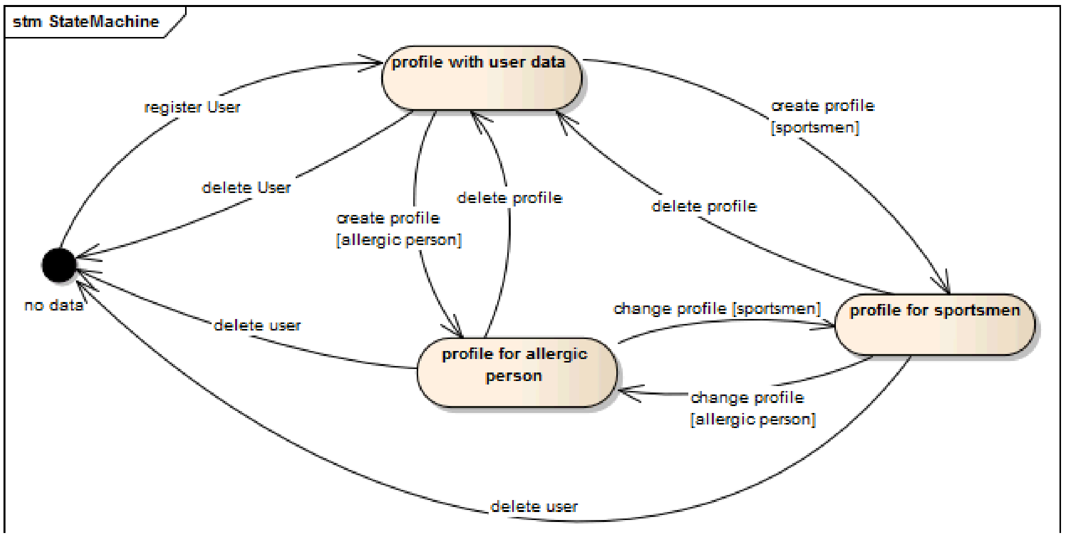

stm StateMachine
profile with user data
register User
create profile [sportsmen]
delete User
delete profile
create profile [allergic person]
delete profile
no data
change profile [sportsmen]
profile for sportsmen
delete user
profile for allergic person
change profile [allergic person]
delete user

Figure 58 PEIS - State Machine for Information Type User profile

There are three types of user profile in the system: A profile only containing the user data, a profile specific for allergic persons and a profile specific for sportsmen. The transitions represent the allowable actions to get from one profile type to another. The corresponding message type diagram for the actions is shown in Figure 59.

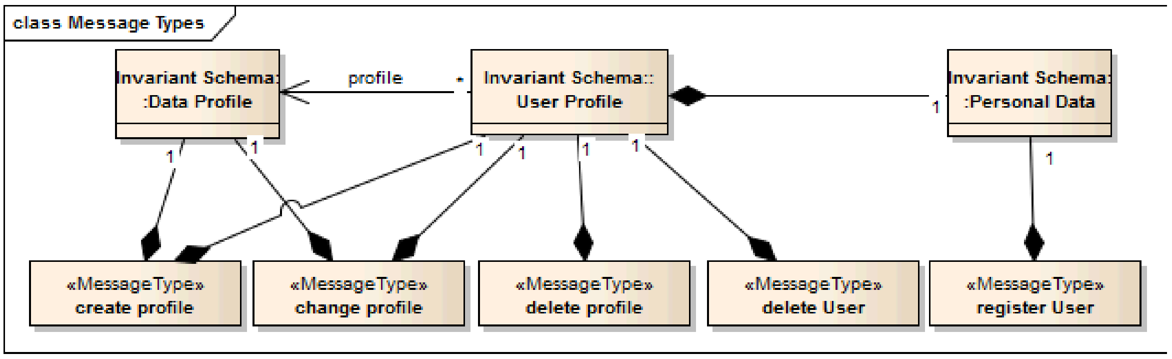

class Message Types
Invariant Schema::Data Profile
profile
Invariant Schema::User Profile
Invariant Schema::Personal Data
«MessageType» create profile
«MessageType» change profile
«MessageType» delete profile
«MessageType» delete User
«MessageType» register User

Figure 59 PEIS - Message Type Diagram

Each action defined on a transition in the State Machine has a corresponding message type in the above diagram. The message types are defined through a composition of information types. For example the message type "regist er User" consists of a Personal Data Information Object.

The static structure of the information objects after creating a user profile is represented in Figure 60 using an UML Object Model. The user profile consists of a specific personal data objected and is linked to the sportsmen profile. This profile is described through Data Set 1, containing a CO2, an Ozon and a Temperature component.

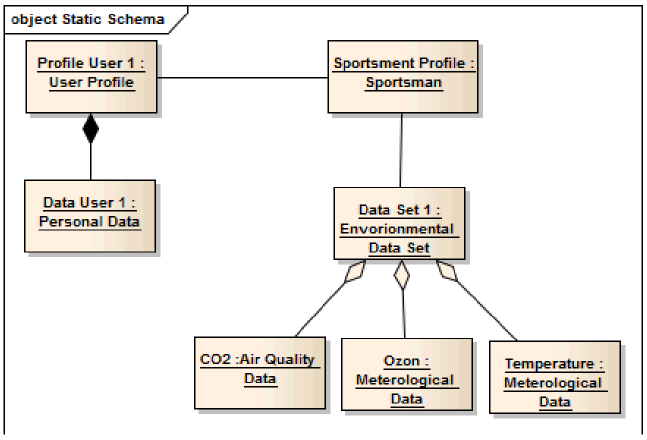

object Static Schema
Profile User 1 : User Profile
Sportsment Profile : Sportsman
Data User 1 : Personal Data
Data Set 1 : Envorionmental Data Set
CO2 :Air Quality Data
Ozon : Meterological Data
Temperature : Meterological Data

Figure 60 PEIS - Object Model

Computational Viewpoint

In the Computational Viewpoint the functional decomposition of the PEIS system is specified. The starting point is the service architecture at the highest level, describing collaborations between PEIS and external participants. This diagram is derived from the Use Case Diagram in Figure 53. For each actor as well as the PEIS system a participant will be created and for each viewpoint a service. This service architecture describing PEIS in its environment is shown in Figure 61.

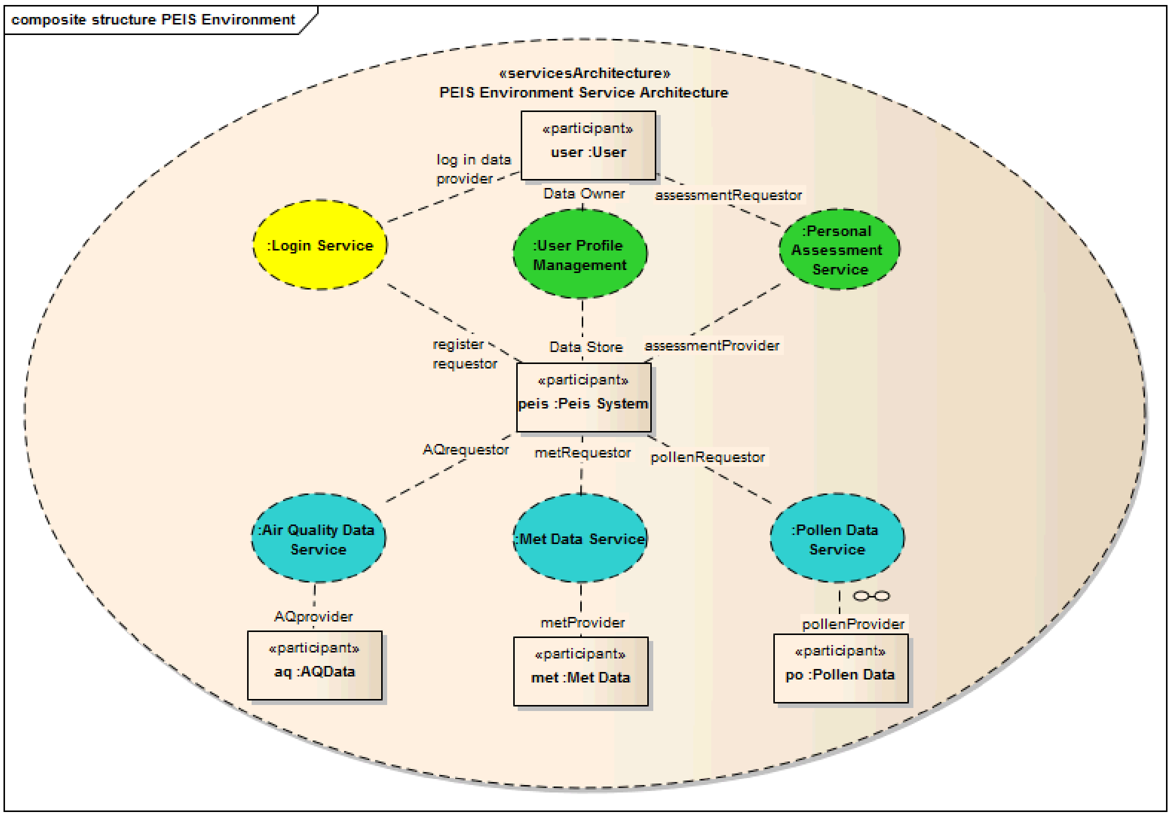


Figure 61 PEIS - Service Architecture PEIS Environment

The assessment creation is provided through the Personal Assessment Service from the PEIS system. The PEIS System and the User collaborate also in the Login Service and in the User Profile Management Service. The first one represents the user login. The last one is for creating and changing the user profile. Also the three different kinds of environmental data services provided by an Air Quality Provider, a Meteorological Data Provider and a Pollen Data Provider are included.

The next step is to specify the PEIS System itself in more detail. Therefore the PEIS System Service Architecture is defined, which describes the internal roles of the PEIS system and how they collaborate to provide the services. The PEIS System Service Architecture is shown in Figure 62.

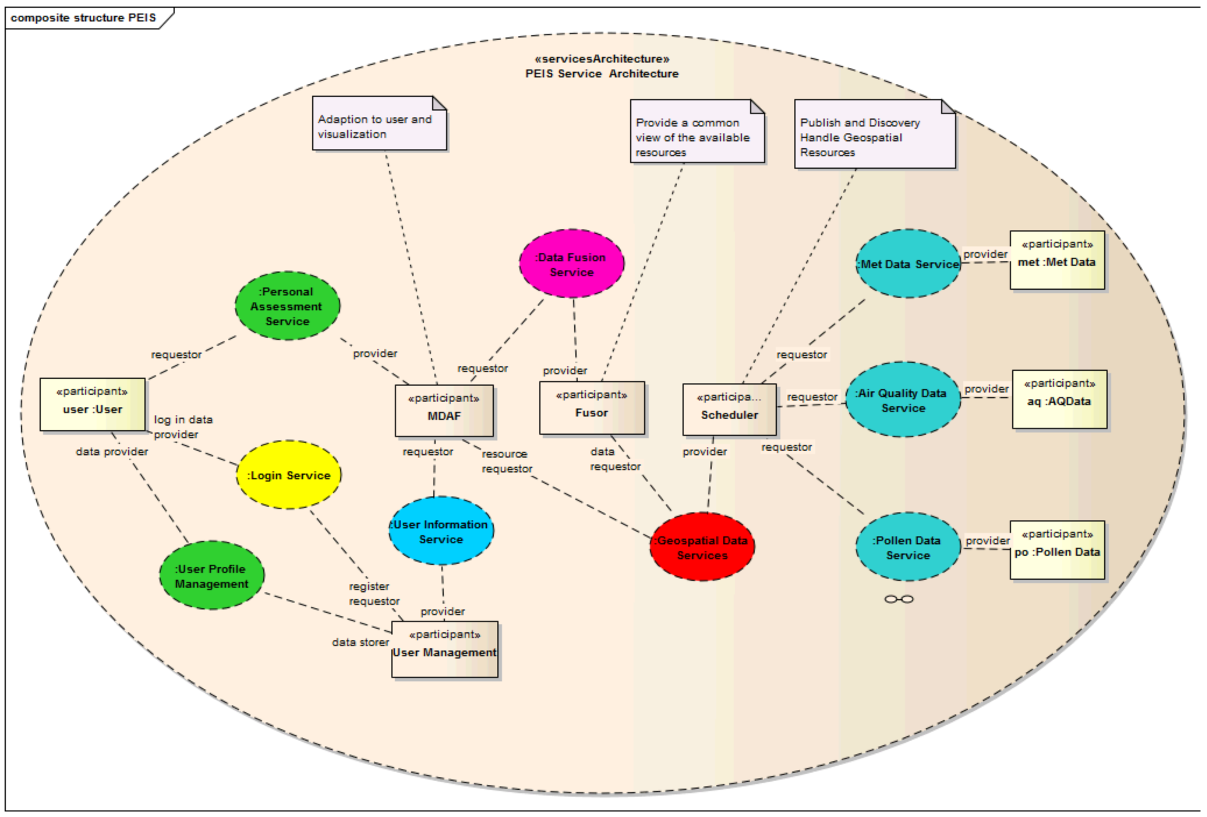

composite structure PEIS
«servicesArchitecture» PEIS Service Architecture
Adaption to user and visualization
Provide a common view of the available resources
Publish and Discovery Handle Geospatial Resources
:Data Fusion Service
:Personal Assessment Service
:Met Data Service
«participant» met :Met Data
«participant» user :User
«participant» MDAF
«participant» Fusor
«participa... Scheduler
:Air Quality Data Service
«participant» aq :AQData
:Login Service
:User Information Service
:Geospatial Data Services
:Pollen Data Service
«participant» po :Pollen Data
:User Profile Management
«participant» User Management
requestor
provider
log in data provider
data provider
resource requestor
data requestor
register requestor
data storer

Figure 62 PEIS - Service Architecture PEIS System

To make external parts visible they are shaded in yellow. The PEIS System contains four parts. They are the Mobile Data Acquisition Framework (MDAF), the Fusor, the Scheduler and the User Management.

- The *user management* participant is responsible for user authentication and the user profile management. It requests login data from the user in order to verify it and it enables the user to create or change a profile.
- The *MDAF* is the user interface of the provided functionality. It collects the required data from the user and requests the necessary environmental data from the Fusor and visualizes it for the user.
- The *Fusor* provides a common view of the available resources with its different types of data. It also enables processing on the existing environmental measurements.
- The *Scheduler* handles the geospatial resources. It enables discovery of the available environmental data sources and provides the access to them.

The different colors of the services show their category according to the classification shown in Figure 50. The Login Services belongs to the Category 'Secruity and Privacy'. The User Profile Management and the Personal Assessment Service are Boundary Interaction Services. Furthermore there is one data processing service, the Geospatial Data Services and one Composition and Workflow Service, the Data Fusion Service. At least there a four services in the category 'Data and Model Management Services'. These are the User Information Service, the Met Data Service, the Air Quality Data Service and the Pollen Data Service. This classification approach is used is all service architectures in the context of PEIS.

Each of the four participants in the PEIS System Service Architecture is further refined with an own Service Architecture. The Service Architecture for the Fusor Participant is shown in Figure 63.

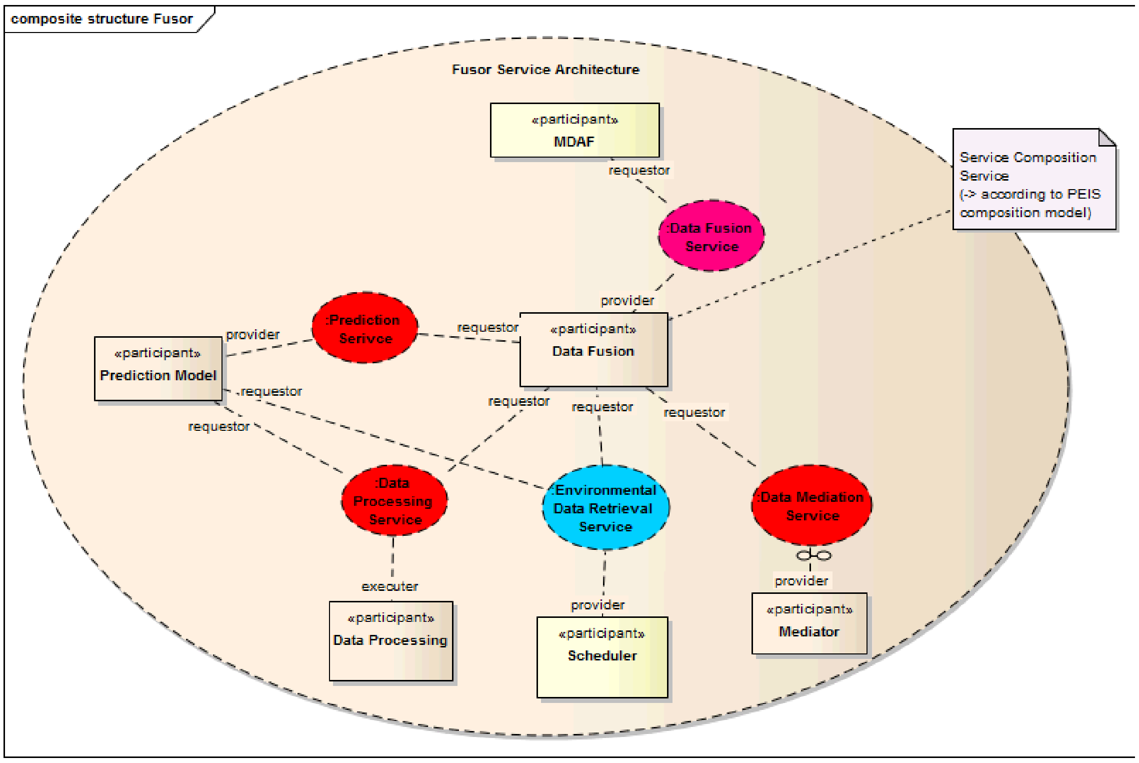


Figure 63 PEIS - Participant Service Architecture Fusor

The two participants MDAF and Scheduler, that request and provide services from the Fusor are also represented in the Participant Service Architecture. Additionally internal parts of the Fusor and their interactions are specified. The Prediction Model, Data Processing and Mediator provide required functionality for the composition in the Data Fusion. The way the service composition in the Data Fusion Participant takes place to provide the Data Fusion Service is defined in a BPMN process diagram. Figure 64 shows this process diagram describing the behavior of the Data Fusion Participant.

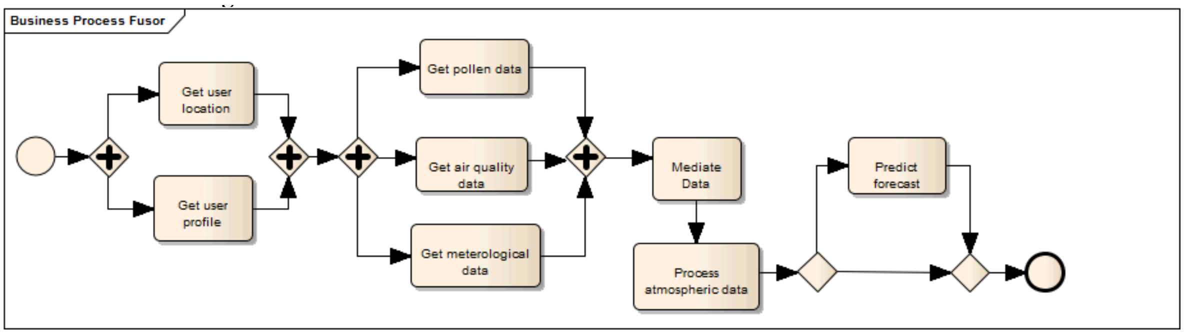


Figure 64 PEIS - Service Composition with BPMN

Each incoming fusion request has attached information about the location and the profile. With use of them the required environmental data concerning pollen, air quality or meteorological conditions are requested. Afterwards the various data types are mediated and in the following processed according to the users needs. If the required assessment is a forecast, a prediction of the required values will be calculated.

To illustrate the specification of services the service contract of the Geospatial Data Services is shown in Figure 65. The Scheduler provides the Geospatial Data Service and the MDAF as well as the Fusor requests this service, but with two different roles. The first one is the Discover Resource Requestor and the second one is the Data Requestor.

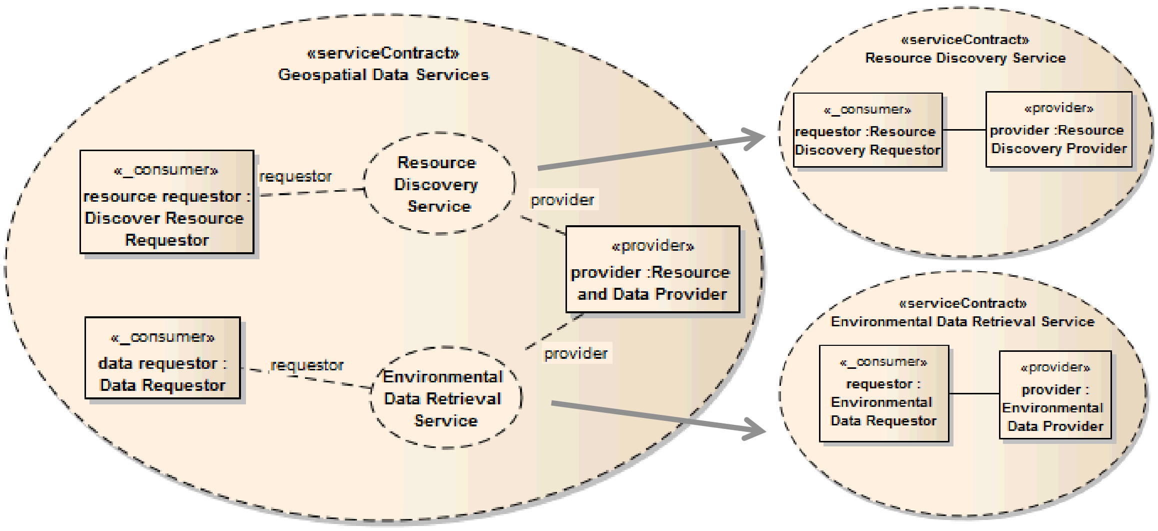

«serviceContract»
Geospatial Data Services
«_consumer»
resource requestor : Discover Resource Requestor
requestor
Resource Discovery Service
provider
«provider»
provider :Resource and Data Provider
«_consumer»
data requestor : Data Requestor
requestor
Environmental Data Retrieval Service
provider
«serviceContract»
Resource Discovery Service
«_consumer»
requestor :Resource Discovery Requestor
«provider»
provider :Resource Discovery Provider
«serviceContract»
Environmental Data Retrieval Service
«_consumer»
requestor : Environmental Data Requestor
«provider»
provider : Environmental Data Provider

Figure 65 PEIS - Service Contracts

The Geospatial Data Service is a composite service, which contains two other services: The Environmental Data Retrieval Services and the Resource Discovery Service. The first one provides access to the environmental data available, which is requested by the Data Requestor Role, played by the Fusor. The second is about the available access to environmental data, requested by the Discover Resource Requestor and played by the MDAF. These two services are also described using a service contract, where the participating roles are defined.

Each role in the service contract is typed with a defined consumer and provider interface. The interfaces for roles in the Geospatial Data Service as well as for the two nested services are shown in Figure 66.

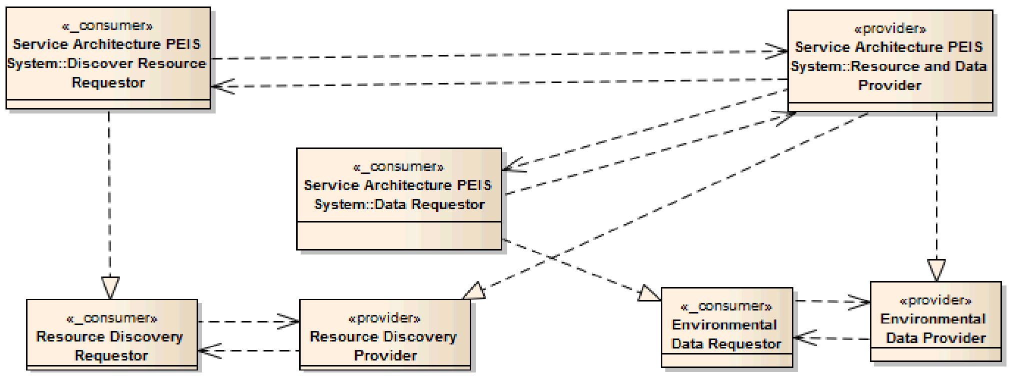

«_consumer»
Service Architecture PEIS System::Discover Resource Requestor
«provider»
Service Architecture PEIS System::Resource and Data Provider
«_consumer»
Service Architecture PEIS System::Data Requestor
«_consumer»
Resource Discovery Requestor
«provider»
Resource Discovery Provider
«_consumer»
Environmental Data Requestor
«provider»
Environmental Data Provider

Figure 66 PEIS - Consumer and Provider Interfaces

The three upper interfaces belong to the Geospatial Resource Service. The four lower services belong to the two nested services of the Geospatial Resource Service. The roles of the Geospatial Resource Service have to realize the roles defined in the two nested services. In a two-way-service each consumer interface uses the provider interface and also vice versa. In a one-way service there will be no consumer interface defined, since the provider does not need a consumer interface.

Engineering Viewpoint

In the Oil Spill Pilot we will have four systems interacting together. The structure is the same as already specified in the Computational Viewpoint. Figure 67 shows the four systems interacting together for Oil Spill as well as the external required ones.

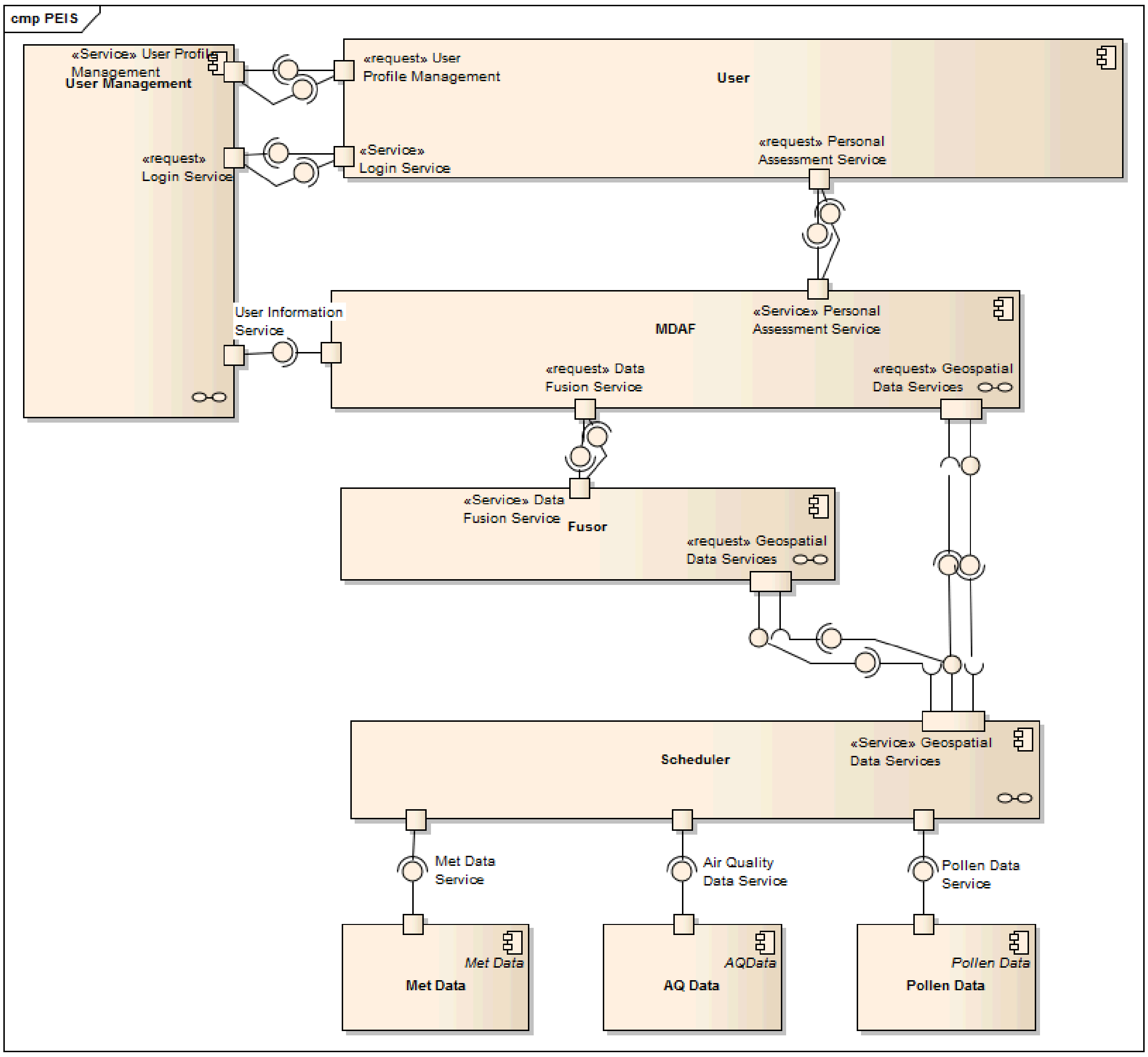

cmp PEIS
«Service» User Profile Management
User Management
«request» Login Service
User Information Service
«request» User Profile Management
«Service» Login Service
User
«request» Personal Assessment Service
MDAF
«Service» Personal Assessment Service
«request» Data Fusion Service
«request» Geospatial Data Services
«Service» Data Fusion Service
Fusor
«request» Geospatial Data Services
Scheduler
«Service» Geospatial Data Services
Met Data Service
Air Quality Data Service
Pollen Data Service
Met Data
Met Data
AQData
AQ Data
Pollen Data
Pollen Data

Figure 67 PEIS - Component Diagram PEIS and Environment

The interaction between the components is specified using ports and exposed provided and required interfaces. The assembly connector is used to link the interfaces and ports. The interfaces for typing the ports with their provided and required interfaces are shown in Figure 66.

The ports at the components are typed with service interfaces. These service interfaces fulfill the appropriate services contracts in the computational specification. For example the Service Contract for the Geospatial Data Services (Figure 65) will be fulfilled in the engineering specification through the service interface Geospatial Data Service (Figure 68).

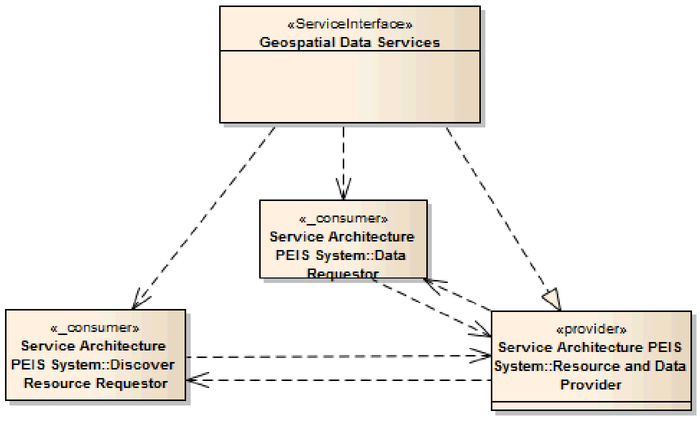

«ServiceInterface»
Geospatial Data Services
«_consumer»
Service Architecture PEIS System::Data Requestor
«_consumer»
Service Architecture PEIS System::Discover Resource Requestor
«provider»
Service Architecture PEIS System::Resource and Data Provider

Figure 68 PEIS - Service Interface

The provided and required interfaces at the ports are typed with the consumer and provider roles defined in the computational specification (Figure 66).

Each of the subsystem components of PEIS can be further refined with an own component diagram. The composite structure diagram of the Scheduler is shown in Figure 69 as an example for those.

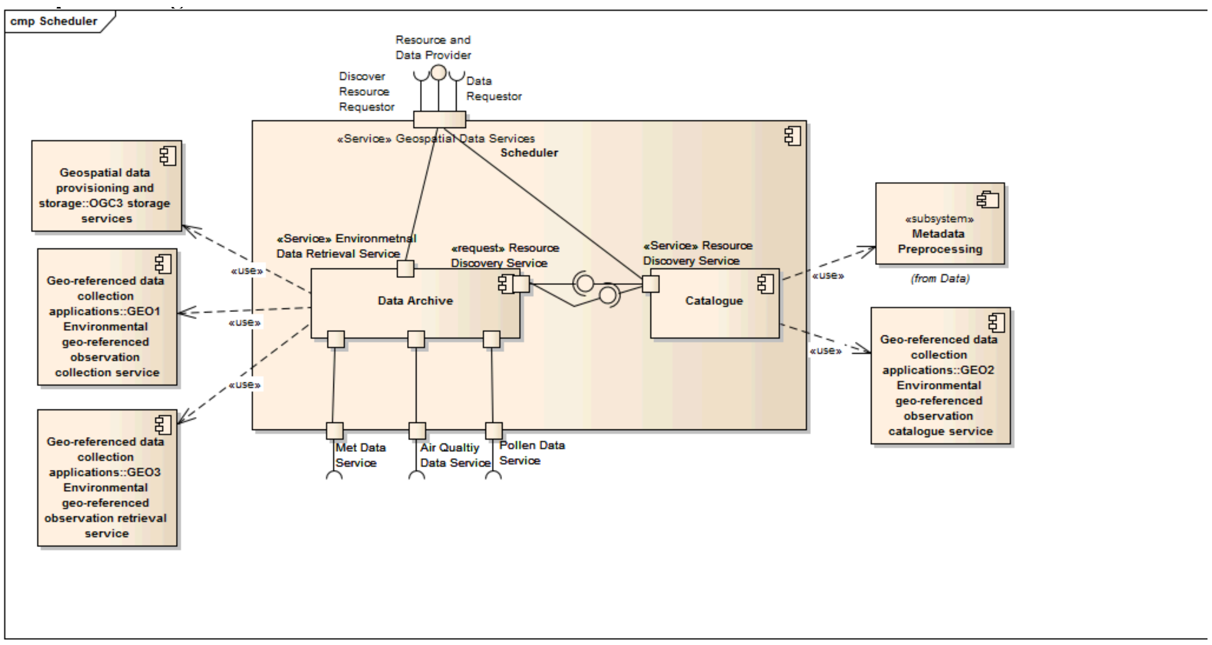

cmp Scheduler
Resource and Data Provider
Discover Resource Requestor
Data Requestor
«Service» Geospatial Data Services
Scheduler
Geospatial data provisioning and storage::OGC3 storage services
Geo-referenced data collection applications::GEO1 Environmental geo-referenced observation collection service
Geo-referenced data collection applications::GEO3 Environmental geo-referenced observation retrieval service
«use»
«use»
«use»
«Service» Environmetnal Data Retrieval Service
«request» Resource Discovery Service
«Service» Resource Discovery Service
Data Archive
Catalogue
Met Data Service
Air Qualtiy Data Service
Pollen Data Service
«subsystem» Metadata Preprocessing
(from Data)
«use»
«use»
Geo-referenced data collection applications::GEO2 Environmental geo-referenced observation catalogue service

Figure 69 PEIS - Component Diagram Scheduler

The Scheduler consists of the Data Archive and the Catalogue, corresponding to the participants in the Service Architecture in the Computational Viewpoint.

The Data Archive requires data from the Met Data Service, the Air Quality Data Service and the Pollen Data Service. The two components collaborate together in the Resource Discovery Services, which is provided by the Catalogue also to Scheduler-external components. Together with the Environmental Data Retrieval Services, provided by the Data Archive the Geospatial Data Services is provided by the Fusor.

Furthermore at this abstraction level of the component diagram a mapping of the specified components with platform capabilities is done. The PEIS system will be based upon the ENVIROFI and FI WARE platform. These platforms provide both, processing capabilities but also capabilities to deal with the distribution of the overall system. Use-relationships from PEIS components to capabilities from these two platforms are defined to make the dependencies between them visible. For example the Data Archive uses OGC3 data storage services to be enable to store the environmental data

The current deliverables of the PEIS project do not provide further details about the distribution mechanisms in the system as well as the used technology.

6.2 Pilot2: ENVISION Oil Spill

The Oil Spill project was already introduced in chapter 2.2. The current specification approach was illustrated in chapter 4.3.2. The Oil Spill Decision support system is built in two steps. In the first step the ENVISION portal is used to create Models as a Service for calculating the Oil Spill and the effects on the Cod population. In the next step these deployed services are used to provide the Oil Spill Decision Support System with use of the ENVISION infrastructure. Therefore the specification of the project in the next chapter is also divided in a Run-Time-Part showing the provisioning of the Oil Spill System and a Design-Time-Part showing the provisioning of the two Model-Services.

6.2.1 Oil Spill Decision Support System (Runtime)

In the following the provisioning of the Oil Spill Decision Support System is specified using MODEA. The models are created based on the Oil Spill specification in ENV 1.1, ENV 1.2 and ENV 1.4.

Enterprise Viewpoint

In the Enterprise Viewpoint there are no big differences between the Oil Spill and the PEIS project. The Oil Spill Project will be also defined with the use of a BMM to define the motivation and furthermore Use Cases and Business Processes to show it will be realized. The overall vision of the Oil Spill System is to support the decision-making in case of an oil spill on the operational level.

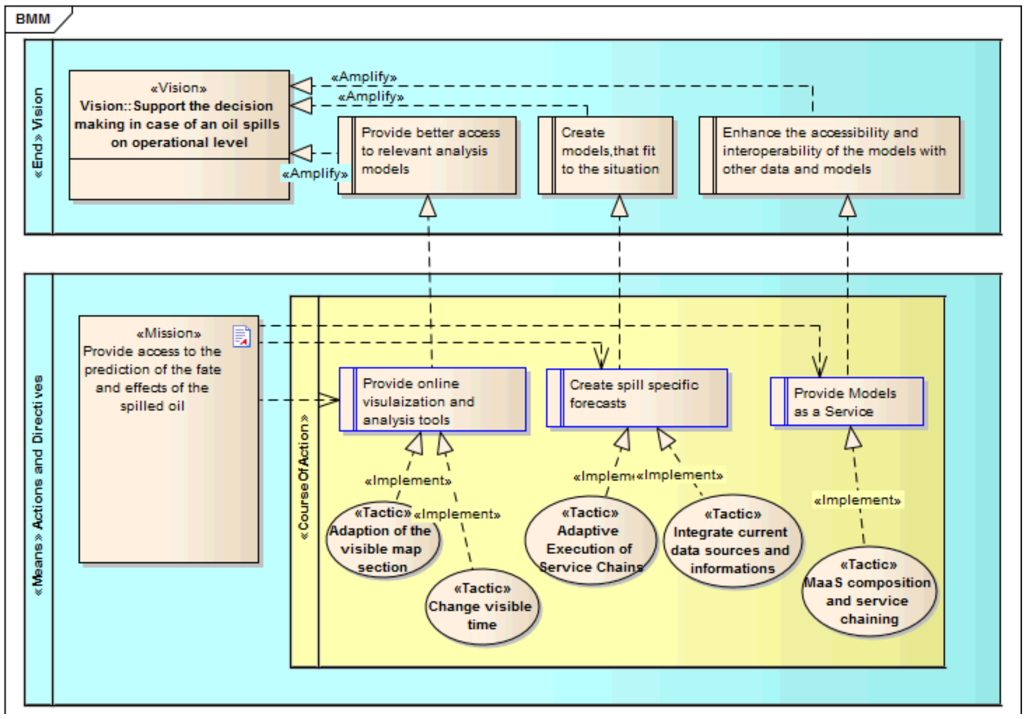

BMM
«End» Vision
«Vision»
Vision::Support the decision making in case of an oil spills on operational level
«Amplify»
«Amplify»
«Amplify»
Provide better access to relevant analysis models
Create models,that fit to the situation
Enhance the accessibility and interoperability of the models with other data and models
«Means» Actions and Directi ves
«Mission»
Provide access to the prediction of the fate and effects of the spilled oil
«CourseOfAction»
Provide online visulaization and analysis tools
Create spill specific forecasts
Provide Models as a Service
«Implement»
«Implement»
«Implem«Implement»
«Implement»
«Tactic»
Adaption of the visible map section
«Tactic»
Change visible time
«Tactic»
Adaptive Execution of Service Chains
«Tactic»
Integrate current data sources and informations
«Tactic»
MaaS composition and service chaining

Figure 70 Oil Spill - Business Motivation Model

The vision will be realized by providing access to the prediction of the fate and effects on cod population of the spilled oil. Therefore an online visualization and analysis tool will be provided, which enables a tailoring of the map section as well as the visible time. To enable well-fitting models a spill specific forecast should be create. This should be done by an adaptive execution of a service chain and the integration of current data sources and information about the oil spill. At least the required models should be provided as a Service (MaaS) to enhance the interoperability between them. Therefore the system should support a MaaS composition and service chaining.

The identified uses cases and actors for the Oil Spill system are presented in Figure 71.

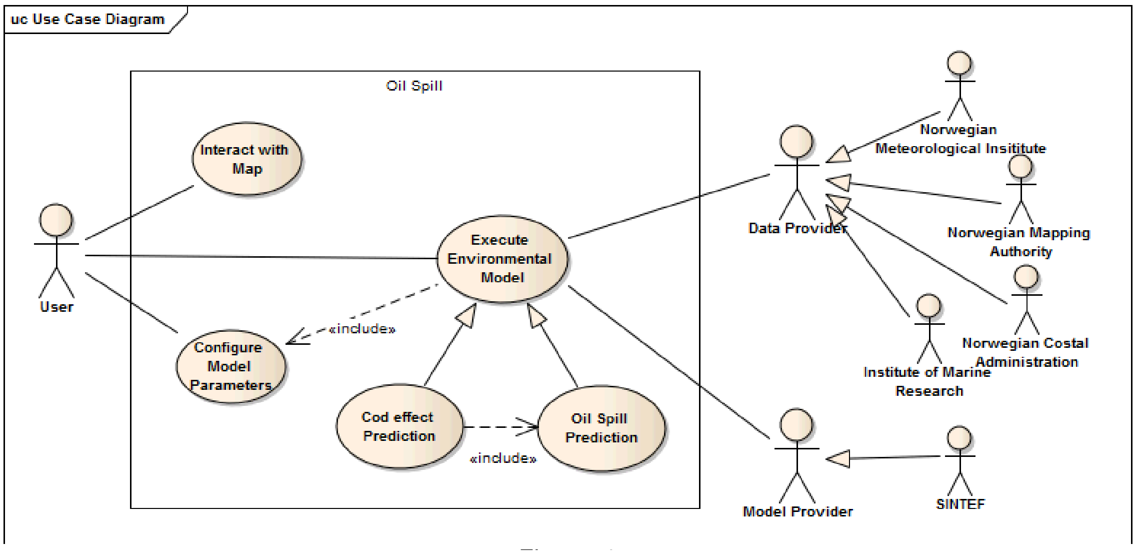

uc Use Case Diagram
Oil Spill
Interact with Map
User
Execute Environmental Model
«include»
Configure Model Parameters
Cod effect Prediction
Oil Spill Prediction
«include»
Data Provider
Norwegian Meteorological Insititute
Norwegian Mapping Authority
Norwegian Costal Administration
Institute of Marine Research
Model Provider
SINTEF

Figure 71 Oil Spill - Use Case Diagram

In the use cases of the oil spill systems three types of actors participate in. There is the user, who interacts with the scenario website in two ways. He can execute an environmental model and he can interact with the map. The model execution can be a Code Effect Prediction or an Oil Spill prediction. Both require the configuration of model parameters from the user. In the model execution also Data Providers and a Model Provider participate. In the first step the predictions made in the Oil Spill System are restricted to spills in the Norwegian Sea. Following the Data Providers include the Norwegian Meteorological Institute for wind and current forecasts, the Norwegian Mapping Authority for sea depth and costal line data as well as sanctuaries, the Norwegian Costal Administration for ship location data and the Institute of Marine Research for cod species and location data. The prediction models required for a simulation of oil spill or the effects on the cod population are both provided by SINTEF MET.

The relationship between the Use Cases and the tactics defined in the BMM are illustrated in the BMM2Use Case Diagram in Figure 72.

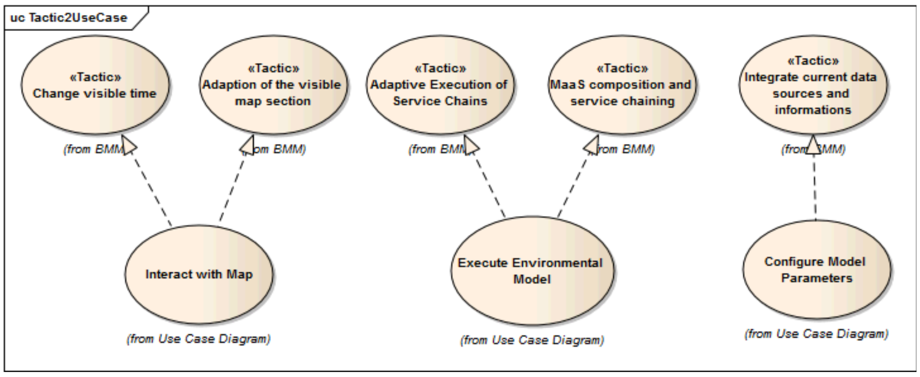

uc Tactic2UseCase
«Tactic» Change visible time
(from BMM)
«Tactic» Adaption of the visible map section
(from BMM)
«Tactic» Adaptive Execution of Service Chains
(from BMM)
«Tactic» MaaS composition and service chaining
(from BMM)
«Tactic» Integrate current data sources and informations
(from BMM)
Interact with Map
(from Use Case Diagram)
Execute Environmental Model
(from Use Case Diagram)
Configure Model Parameters
(from Use Case Diagram)

Figure 72 Oil Spill - BMM2Use Case Diagram

The Use Case Interact with Map realizes the two tactics Change visible time and adaption of the visible map section. The tactic adaptive execution of service chains as well as MaaS composition and service chaining are both realized in the Execute Environmental Model. With a configuration of the model parameters the tactic to integrate current data sources and information is realized.

The use cases of the Oil Spill project are also further refined using BPMN Collaborations in the same manner as in the PEIS project. Thereby for each use case one business process is defined, despite of the generalized use case execute environmental model. This use case has not a direct corresponded business process. Included use cases are realized through the use of sub processes as it is shown in the PEIS project in Figure 55 and Figure 56.

Information Viewpoint

The Information Viewpoint in Oil Spill can be modeled in the same way as the one for PEIS. For a better understanding of the ongoing example the class diagram representing the information types used in Oil Spill is shown in Figure 73.

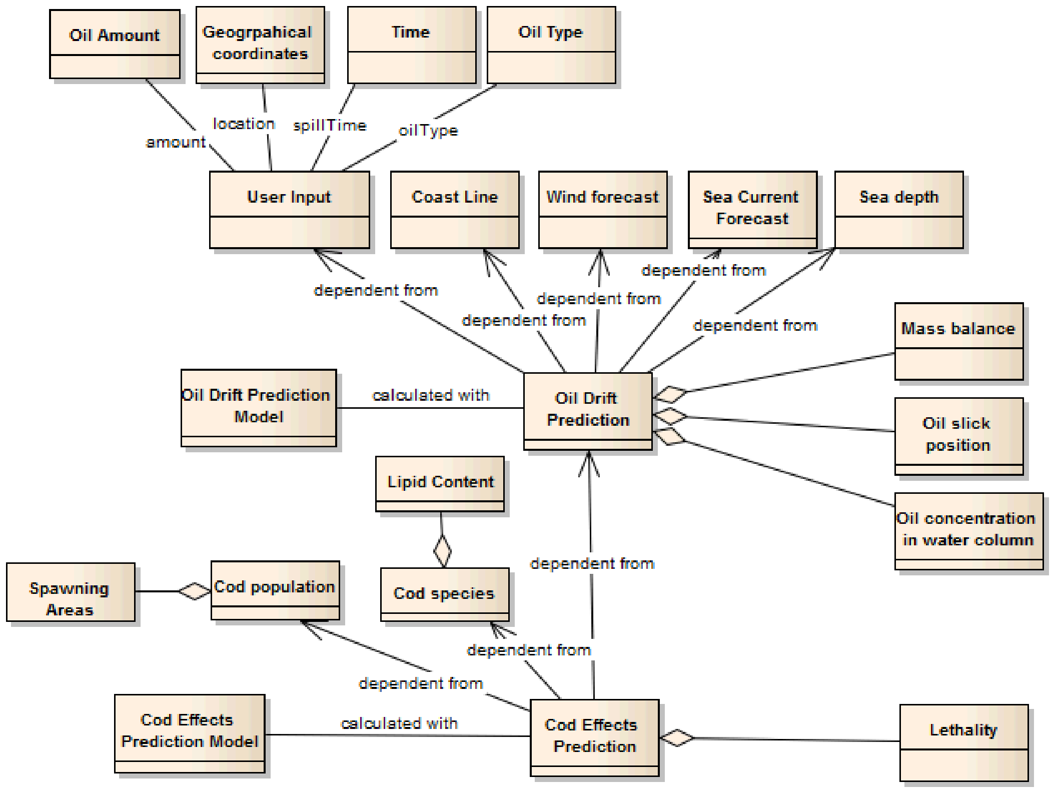


Figure 73 Oil Spill - Invariant Schema Information Type Diagram

This diagram is created based on the Oil Spill Ontology defined in Deliverable 4.3 (ENV4.3). The prediction of the oil drift consists of three parts. These are the mass balance, the oil slick position and the oil concentration in water column. Therefore information from the user about the amount of spilled oil, the geographical coordinates of the spill, the time and the oil type are required. Additionally coastline data, wind and current sea forecast as well as sea depth data are required to create the prediction. Each predication of an oil spill is calculated with the use of a oil spill prediction model. For the cod effect prediction data about the cod population, the cod species as well as the oil spill prediction is required. The prediction is calculated based on a Cod Effects Prediction Model and consists of the lethality.

Computational Viewpoint

The high level service architecture of the computational specification is derived from the use case diagram in Figure 71 using the same methodology as in PEIS, which is described in chapter 5.5. The user, the data provider and the model provider as well as the Oil Spill System are represented as participants. The Oil Spill Prediction, Cod Effects Prediction and Map Interaction Use Cases are represented as Service Contracts. The overall service architecture showing the collaborations of the oil spill system with external participants is shown in Figure 74.

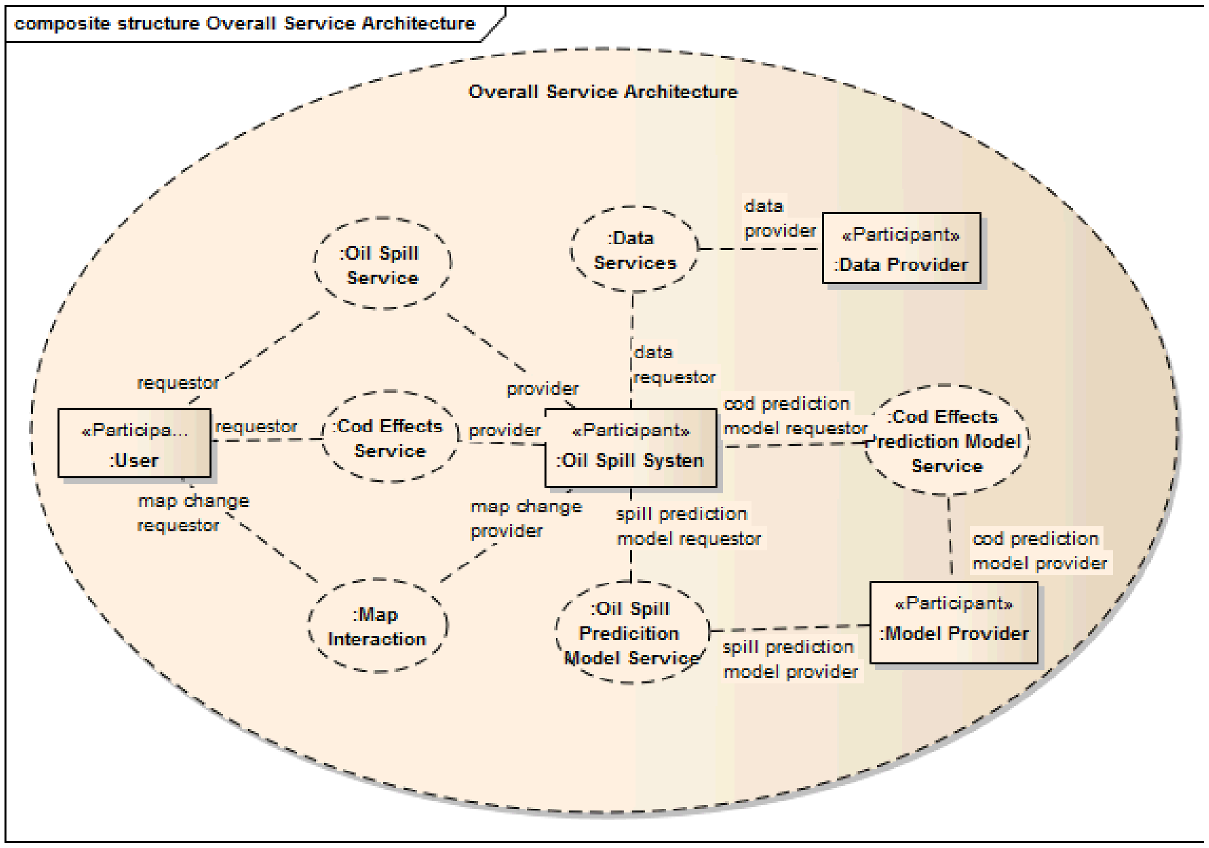

composite structure Overall Service Architecture
Overall Service Architecture
:Oil Spill Service
:Data Services
data provider
«Participant» :Data Provider
data requestor
requestor
provider
«Participa... :User
requestor
:Cod Effects Service
provider
«Participant» :Oil Spill Systen
cod prediction model requestor
:Cod Effects Prediction Model Service
map change requestor
map change provider
spill prediction model requestor
cod prediction model provider
:Map Interaction
:Oil Spill Predicition Model Service
spill prediction model provider
«Participant» :Model Provider

Figure 74 Oil Spill - Overall Service Architecture

In the following the Oil Spill System will be further refined through the definition of a service architecture for the Oil Spill System Participant. Therefore the internal parts Web Site Navigation, Composition Execution, Oil Spill Prediction Executable and Cod Effects Prediction Executable are specified. The Web Site Navigation is responsible for the visualization of the predicted model as well as for the user interactions with the visualized prediction. The composition execution triggers and monitors the execution of the two prediction compositions. These compositions are represented in the Oil Spill Prediction Executable as well as in the Cod Effects Prediction Executable. The two parts specify the composition of data and model provider for calculating the required prediction. Figure 75 shows the service architecture of the oil spill system with these internal components and the service contracts that exist between them.

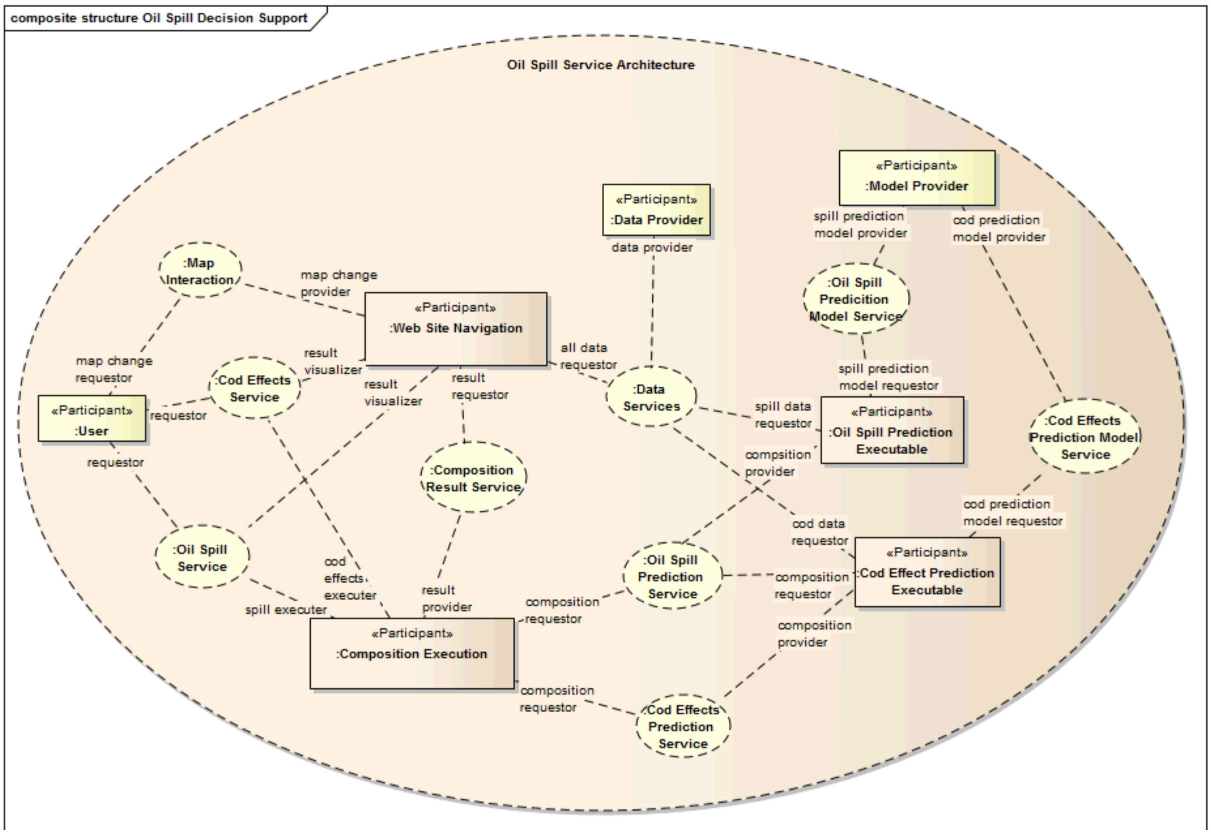

composite structure Oil Spill Decision Support
Oil Spill Service Architecture
«Participant» :Model Provider
«Participant» :Data Provider
data provider
spill prediction model provider
cod prediction model provider
:Oil Spill Predicition Model Service
:Map Interaction
map change provider
«Participant» :Web Site Navigation
all data requestor
:Data Services
spill prediction model requestor
map change requestor
result visualizer
:Cod Effects Service
result visualizer
result requestor
«Participant» :User
requestor
spill data requestor
«Participant» :Oil Spill Prediction Executable
:Cod Effects Prediction Model Service
compsition provider
requestor
:Composition Result Service
cod prediction model requestor
cod data requestor
:Oil Spill Service
«Participant» :Cod Effect Prediction Executable
cod effects executer
:Oil Spill Prediction Service
composition requestor
result provider
spill executer
composition requestor
«Participant» :Composition Execution
composition provider
composition requestor
:Cod Effects Prediction Service

Figure 75 Oil Spill - Oil Spill System Service Architecture

The participants Web Site Navigation and Composition Execution are further refined using a service architecture. The Oil Spill Prediction Executable as well as the Cod Effect Predication Executable represents deployed BPEL processes. The logic of the composition in these executables is specified using BPMN Process Diagrams. The process for the Oil Spill Prediction Executable is shown in Figure 76.

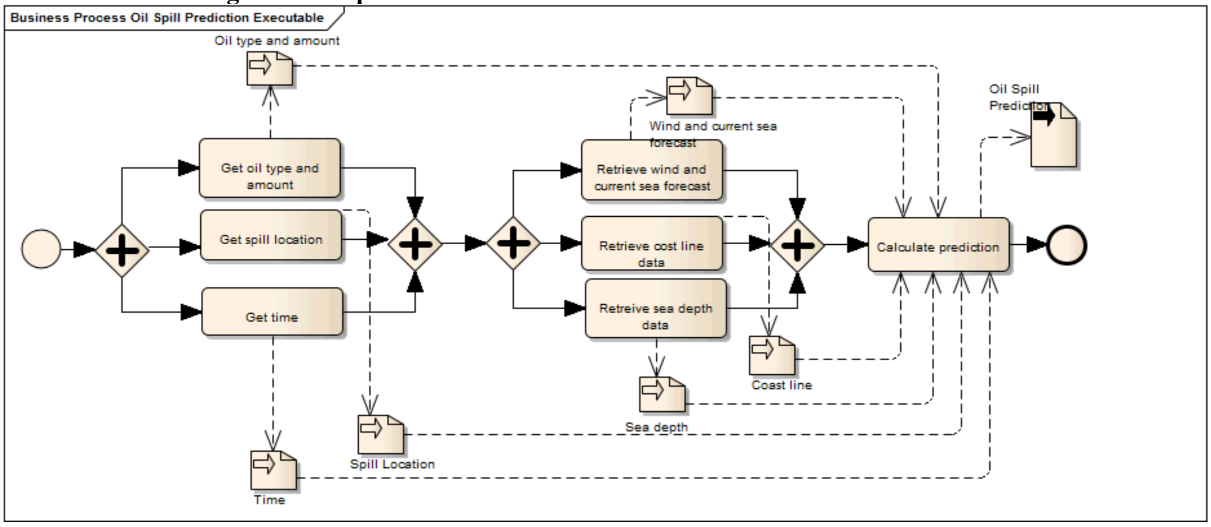


Figure 76 Oil Spill - Composition Process for the Oil Spill Prediction Executable

This diagram represents the overall workflow of PEIS shown in Figure 8, chapter 2.2.1. It shows how the various service are composed together to provide the required functionality for the Oil Spill Prediction Service. It also defines the flow of the information types within this process. The executables are created during Design Time of the Oil Spill System and with use of the ENVISION portal. The portal enables the user to create such BPMN composition diagrams and deploys them to executable BPEL processes. How this functionality is provided will be described in another MODEA specification in chapter 6.2.2.

Engineering Viewpoint

The next step, based on the methodology for MODEA (chapter 5.5) is to define the engineering specification. Figure 77 shows the component diagrams specifying the oil spill system.

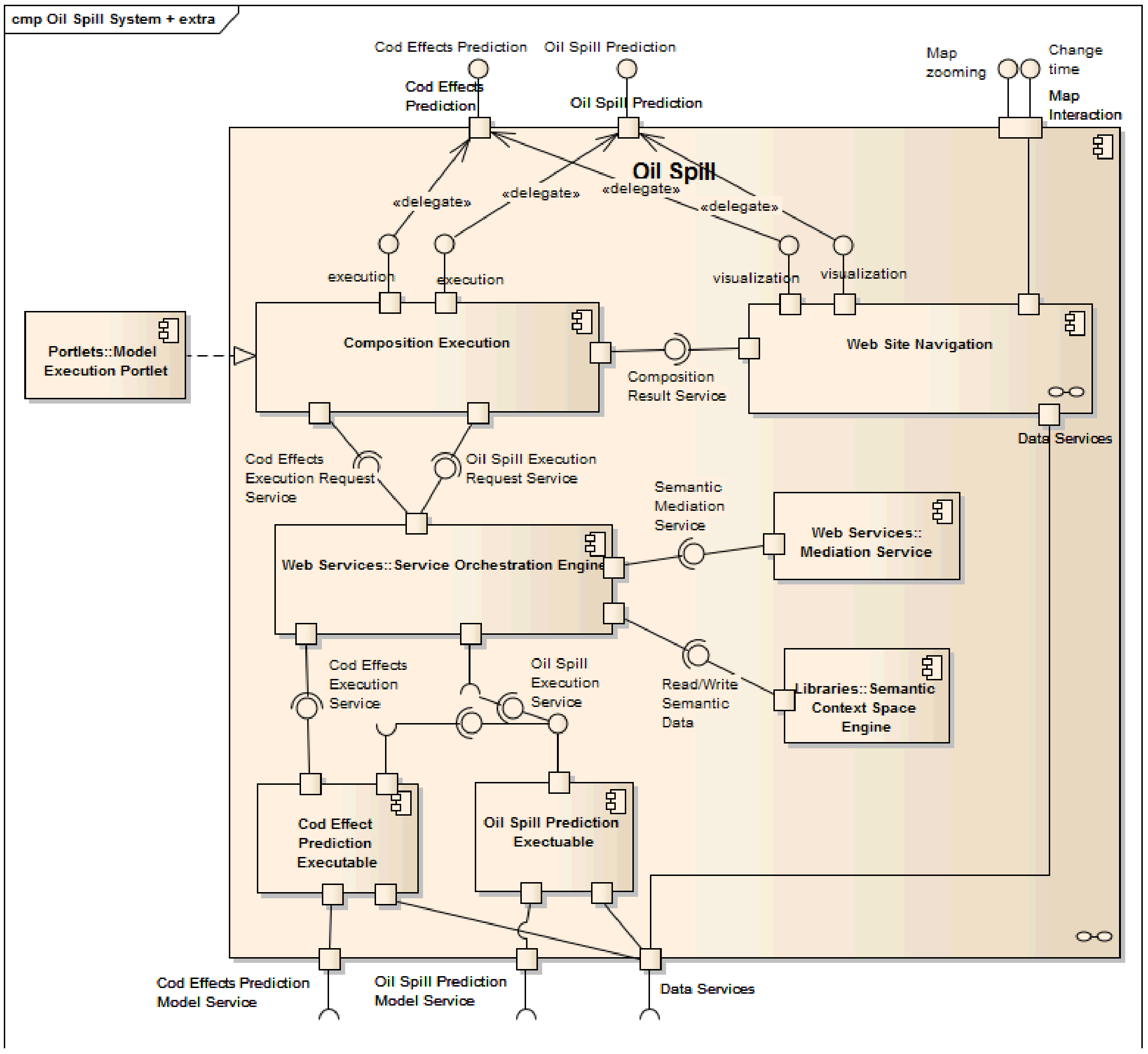

cmp Oil Spill System + extra
Cod Effects Prediction
Oil Spill Prediction
Map zooming
Change time
Map Interaction
Cod Effects Prediction
Oil Spill Prediction
Oil Spill
«delegate»
«delegate»
«delegate»
«delegate»
execution
execution
visualization
visualization
Portlets::Model Execution Portlet
Composition Execution
Composition Result Service
Web Site Navigation
Data Services
Cod Effects Execution Request Service
Oil Spill Execution Request Service
Semantic Mediation Service
Web Services::Service Orchestration Engine
Web Services:: Mediation Service
Cod Effects Execution Service
Oil Spill Execution Service
Read/Write Semantic Data
Libraries::Semantic Context Space Engine
Cod Effect Prediction Executable
Oil Spill Prediction Exectuable
Cod Effects Prediction Model Service
Oil Spill Prediction Model Service
Data Services

Figure 77 Oil Spill - Component Diagram Oil Spill System

Despite the already known parts Composition Execution, Website Navigation, Oil Spill Prediction Executable and Cod Effects Prediction Executable there are also further components integrated. These components are provided by the ENVISION infrastructure are used to cope with the distribution and therefore also heterogeneity of the various components. For example the Service Orchestration Engine "enable[s the] distributed execution of environmental models as BPEL-based service chains" (ENV6.1). Therewith it is required to execute the two executables for Oil Spill and Cod Effects. Following the Oil Spill Prediction Service from the computational viewpoint will be realized through two service interfaces. These service interfaces together with the realized and used consumer and provider interfaces from the computational specification are shown in Figure 78.

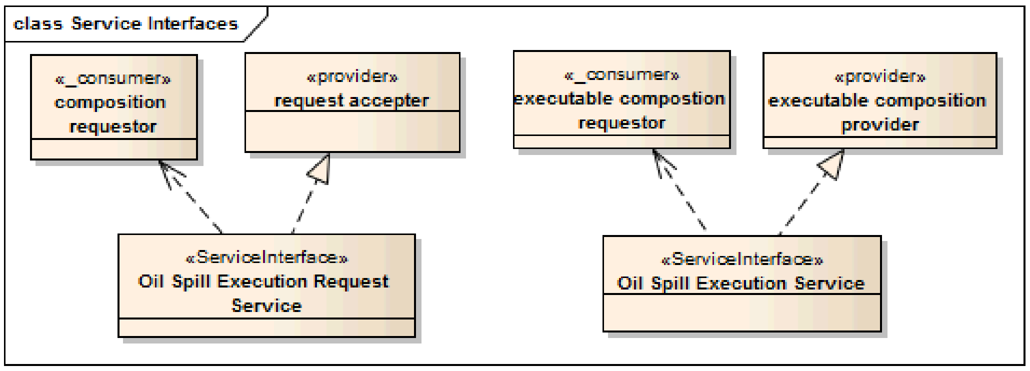

class Service Interfaces
«_consumer» composition requestor
«provider» request accepter
«_consumer» executable compostion requestor
«provider» executable composition provider
«ServiceInterface» Oil Spill Execution Request Service
«ServiceInterface» Oil Spill Execution Service

Figure 78 Oil Spill - Service Interfaces

The Oil Spill Execution Request Service and the Oil Spill Execution Service together fulfill the service contract of the Oil Spill Prediction Service. The provided and required interfaces at the Composition Execution and the Oil Spill Executables are the same as defined roles in the service contract in the computational specification.

Technology Viewpoint

Since the specification of the Oil Spill Project is more far advanced in development than PEIS a first specification of the technology viewpoint is possible. The following figure shows the deployment diagram for the Oil Spill System.

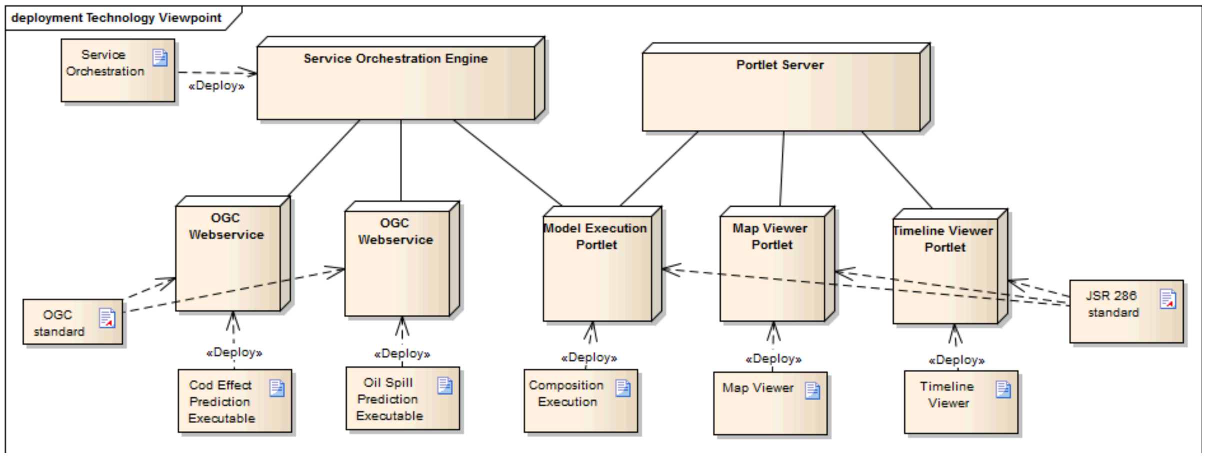


Figure 79 Oil Spill - Deployment Diagram

The components defined in the engineering specification are represented as artifacts. Each of the artifacts is then deployed to a node. For example the Code Effect Prediction Executable is deployed as an OGC Webservice. Therewith the OGC standards have to consider. The Composition Execution, the Map Viewer and the Time Line Viewer are deployed as Portlets using the JSR 286 standard. Each of these Portlets will be then run on a Portlet Server. (ENV 1.2)

6.2.2 ENVISION portal (Oil Spill Design Time)

As already mentioned above the Oil Spill System is build in two steps. The first one, the provisioning of the scenario website using deployed compositions for the functionality is explained in the previous chapter. Now the second step, i.e. the step to provision these deployed compositions, will be shortly described. Therefore only the computational viewpoint will be used to provide a short overview, how the ENVISION portal is provided. The Service Architecture describing the ENVISION portal is shown in Figure 80.

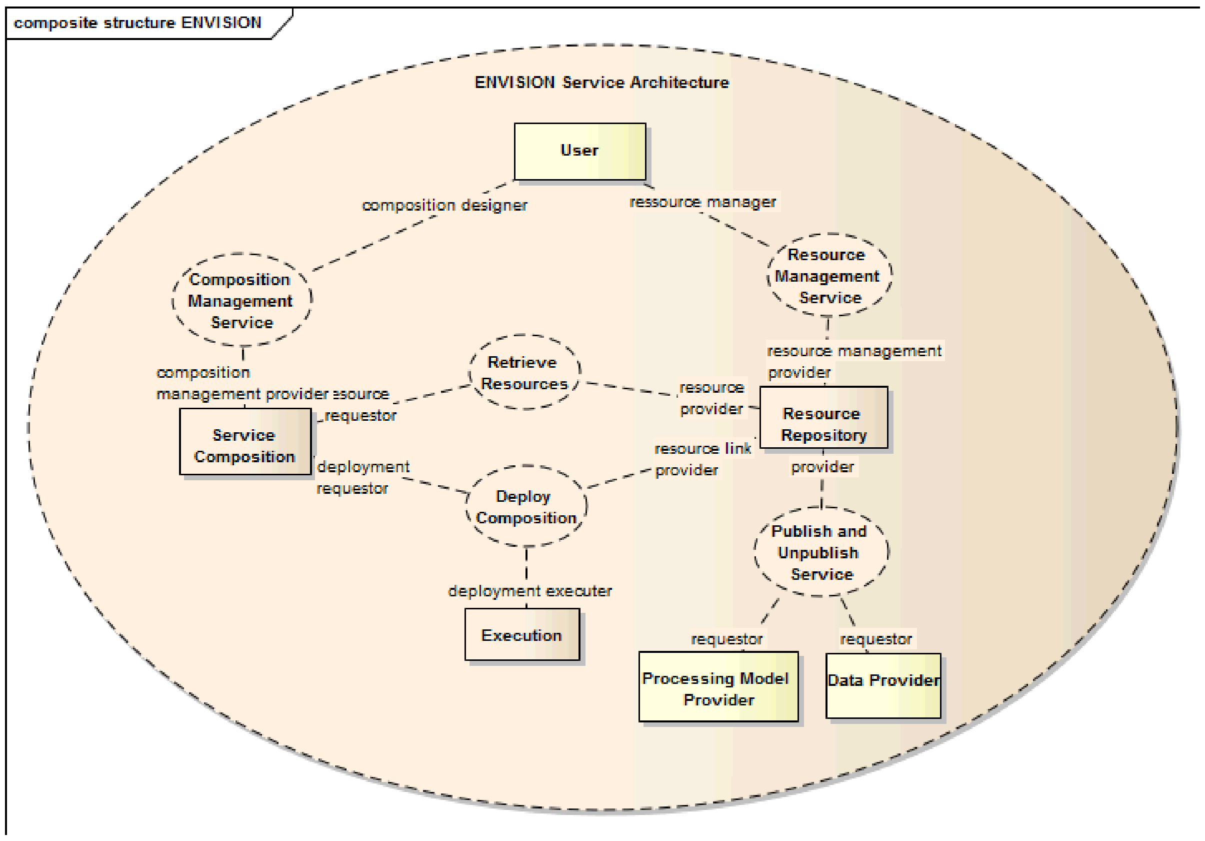


Figure 80 ENVISION Portal - Service Architecture

There are two main users roles that interact with the ENVISION portal. One is the composition designer, who creates, edits and requests the deployment of the composition. The other one is the resource manager, who import and exports resources in the repository and annotates them using ontologies. The Service Composition Module enables a visual service chaining of the resource in the repository. At least it provides support for generate executable compositions that can be deployed as a web service. The Execution Module provides the deployment. The Resource Repository administrates the resources that used in the service composition module. An interface to the user enables the management of the resources. This includes service discovery, import, export and visual semantic annotation of services using ontologies.

7 Conclusion and future work

7.1 Summary

In the previous chapters first the problems occurring in the development and specification process of open distributed systems are examined. Among these are the heterogeneity of the components and the high complexity and scope of the system. Furthermore providing the right functionality and an effective collaboration between the vendors are challenges in the development process. At least enabling global optimization, integrating distribution transparencies and the need for flexible architecture are issues that have to consider.

To cope with these problems one suggestion is the use of an enterprise architecture framework with adequate standardized modeling techniques. Thereby the frameworks provide the foundation with defining viewpoints and the concepts that are described within them as well as the correspondences between the viewpoints. Typically a framework does not specify techniques for its application.

MODEA, as a model-driven approach for open distributed systems extending an enterprise architecture framework, provides a proposal of how to use OMG standards when modeling Enterprise Architectures. The Reference Model for Open Distributed Systems with its five viewpoints Enterprise, Information, Computational, Engineering and Technology is used as foundation in MODEA.

An overview of MODEA showing the modeling techniques used to describe each viewpoint is given in Table 17. In the left column of each viewpoint the concepts described in this viewpoint are shortly mentioned. In the right column the used modeling techniques are listed.

Enterprise Viewpoint			
Goals, Strategies, Requirements and Business Process, Actors		Business Motivation Model, UML Use Cases, Use Case Templates, BPMN Collaborations	
Information		**Computational**	
Information Types, Information Objects, Actions, Semantics of Information Processing	UML Class and Object Diagrams, UML State Machines, SoaML Message Types	Functional Decompositions in terms of provided and required services	SoaML, UML Sequence Diagrams, BPMN Processes
Engineering			
Distribution of system components, communication technologies		UML Component Diagram, SoaML Service Interfaces, UML Sequence Diagram	
Technology			
Specification of the hard- and software infrastructure, deployment process		UML Deployment Diagram, BPMN Processes	

Table 17 Summary of MODEA

The Actors in the Enterprise Viewpoint, representing roles that interact with the system, are linked to Use Cases in the UML Use Case Diagram as well as to pools in the Business Process refining those Use Cases. Each use case will realize at least one tactic or business rule defined in the BMM.

Each use case will have a correspondent service in the computational viewpoint realizing the required functionality. The service can be composed of other service for providing the functionality. Such a composition will be specified with BPMN processes. The participants providing or requesting service are in the first step derived from the actors and then can then be further refined. The structure of the service and participants is defined in a service architecture, which can be used on different abstraction levels. The interaction between two participants related to a service, which is specified through a service contract, is defined in the UML Sequence Diagram.

The information viewpoint provides a common set of information types and actions and well as constraints on those. All the other viewpoints have to be consistent to this definition and using these actions to specify interfaces or information types for information flows.

In the engineering viewpoint system components are specified for the participants in the computational viewpoint. These components communicate through ports, which are realized by a service interface. At least these components are mapped to processing nodes in the technology viewpoint with use of UML Artifacts.

In chapter 6 the modeling approach is illustrated using the Personal Environmental Information System PEIS as well as the Oil Spill Decision Support System.

The current specification of MODEA does not encompass all model elements defined in the several specification of the modeling techniques. For the beginning the focus lies on the main concepts and how to enable the creation of a coherent specification with them. Further work will be required to expand this approach for a full support of the UML, SoaML, BMM and BPMN specifications.

7.2 Evaluation

The current degree of detail in the MODEA specification is enough to make a first comparison with the requirements defined in chapter 3. This evaluation of MODEA is shown in Table 18.

	Requirement	**MODEA**	**ArchiMate**	**UPDM**	**UML4ODP**
1.	Use of different Viewpoints	++	++	++	++
2.	“Smart” Diagrams	+	+	o	o
3.	Use of existing standards	++	-	o	o
4.	Formal specified modeling techniques	++	o	++	++
5.	Tool support for modeling techniques	++	++	++	++
6.	Tool support for model transformation, code generation	o	--	o	o
7.	Assignment of responsibilities	+	-	++	++
8.	Integrate motivation and requirements	++	o	++	o
9.	Support Use Cases	++	--	--	--
10.	Set up a system-wide set of vocabulary	++	++	++	++

11.	Specification, integration of different architectural styles and patterns	+	o	o	o
12.	Decomposition of Components	++	o	++	o
13.	Support a service oriented architectural style	++	o	o	o
14.	Support specification of distribution transparencies	++	o	o	++

Table 18 Evaluation of MODEA and other modeling approaches

As it can be easily seen in Table 17 MODEA provides fully support for the specification of the motivation and requirements and use cases. It is defined using different viewpoints, including the engineering viewpoint to specify distribution transparencies and the information viewpoint for setting up a system-wide set of vocabulary. With only the use of latest OMG standards all modeling techniques are standardized and with the use UML and BPMN two well-established and well-known modeling techniques are used. For all the modeling techniques there is quite good tool support also code and model generation could possible since all modeling techniques have a formal defined specification.

As well as SoaML in the Computational Viewpoint and also UML Component Diagrams in the Engineering Viewpoint enable the specification of composition and decomposition. Both approaches enable the definition of interfaces and interaction protocols to specify the behavior. With BPMN processes and collaborations the composition behavior in terms of orchestration and choreography can be described.

Nearly all requirements in the context of the support for a service-oriented architectural style are fulfilled by concepts of SoaML. SoaML provides support for identification, reuse and the specification of services. The last two ones are enabled through the service contract concept. (OMG12a) For the classification of services there are two possibilities in MODEA. One is to color the uses of a service contract in a defined color. The other one is to use compound service contracts for a classification. Such a compound service contract can also be used to specify and use design patterns.

For a full support of different architectural styles UML lacks methods for defining communication details between components. For example it is not possible to differ between a rest-oriented communication, a stream-oriented communication, an event-oriented communication or a simple request and reply.

Concluding one can see that MODEA, comparing to UPDM, UML4ODP and ArchiMate, fulfills more of the defined requirements. Especially in the fields where the three other frameworks are weak MODEA contains concepts to deal with them. These are the use existing standards, the possibility of specifying distribution transparencies as well as a full support for a service oriented architectural style. Also when dealing with the requirements, especially the popular concept of use cases, MODEA is stronger than the three other ones. In the case of support for model and code generation, MODEA is on the same stage as UPDM and UML4ODP. All the frameworks are based on formal specifications, especially UML, and therefore code generation is possible, but not yet commercially implemented.

7.3 Future Work

This concludes in one point, where future work can be done. In chapter 5.4 a first mapping of concepts from the different viewpoint to each other and possibilities of how to derive one model out of another (chapter 5.5) are introduce. Further work has to be done to first include all specification elements of the used techniques in the MODEA approach and then in a second step define formal rules for model transformation. Finally concepts have to be specified of how to generate code out of these models.

Beside this issue concerning the overall framework, there are also some aspects for future work specific for each viewpoint.

Enterprise Viewpoint

Since agile techniques are an important issue in current software development, further work can be done to include their requirements specification concepts. In Leffingwell (2011) the requirements descriptions is done in terms of Themes, Epics, Features and Stories. One proposal for their integration is to use the UML Use Case as basic modeling concept and apply it on different abstraction levels. The highest abstraction level of use cases will be Themes. They will be refined into Epics, Features and then into Stories. How this can be definitely adapted in MODEA is subject for future work.

Information Viewpoint

Another way than using loosely coupled services to gain a high interoperability in open distributed systems is the use of ontologies to enable semantic interoperability and integration. Especially in the context of heterogeneous systems ontologies provide several advantages since they contain "computer-usable definitions of basic concepts in the domain and the relationships among them" (Obr03). Such ontologies can be used in the context of service composition, as it is already done in ENVISION, but also for a model and also code generation for the overall MODEA specification. How this can be done and also integrated in the modeling approach has to be done in future.

Computational Viewpoint

The computational viewpoint provides a lot of possibilities for code generation that have to considered in future work. First it would be possible to generate the highest abstraction level of service architecture out of the use case diagram defined in the enterprise viewpoint. Afterwards, when the computational specification is finished, it could be transformed to an initial draft for the engineering viewpoints.

Engineering Viewpoint

A major issue for future work in context of the Enterprise Viewpoint is the integration of more details about characteristics of the communication between the components. Aspects to consider here are for example whether the communication is asynchronous or synchronous and persistent or transient. A second characteristic to be defined is how the communication takes place. For example the communication can take place as messaging, request/response or stream. (Cro96) The current UML specification does not allow a visible definition of those characteristics in the models. Future work has to elaborate if the introduction of new icons or connector type is necessary to specify such details.

7.4 Conclusion

In summary MODEA is a coherent model driven approach for specifying and designing enterprise architectures, especially those of open distributed systems. It is built upon the enterprise architecture framework RM ODP, which provides a sound basis of how to structure the overall specification. The concepts of the framework are modeled using the latest OMG standards and also with the integration of a service-oriented architectural style. These techniques provide a common foundation to cope with the heterogeneity and complexity and enable an effective collaboration between the various vendors and a flexible, optimized architecture. Although there is a quite good tool support for modeling, further work is required to enhance the support for model and code generation.

8 Literature

Ado02 **Steve Adolph, Paul Bramble, Alistair Cockburn, Andy Pols (2002):** Patterns for Effective Use Cases.
Addison-Wesley Professional, 1st edition

Ams08 **Jim Amsden (2008):** Capturing requirements with Business Motivation Model, IBM Rational RequisitePro, and IBM Rational Software Modeler.
In:https://www.ibm.com/developerworks/rational/library/08/0401_amsden/, accessed on 10.08.2012

And01 **Bente Anda, Dag Sjøberg, Magne Jørgensen (2001):** Quality and Understandability of Use Case Models.
In: J. Lindskov Knudsen (Ed.): ECOOP 2001 – Object oriented programming, Vol. 2072, pp. 402-428, Springer-Verlag Berlin Heidelberg

Bar06 **Alistair Barros, Marlon Dumas, Phillipa Oaks (2006):** Standards for Web Service Choreography and Orchestration: Status and Perspectives.
In: C. Bussler et al. (Eds.): BPM Workshops, Vol. 3812, pp. 61–74, Springer-Verlag Berlin Heidelberg

Ber08 **Birol Berkem (2008):** From The Business Motivation Model (BMM) To Service Oriented Architecture (SOA).
In: Journal of Object Technology, Vol. 7, No. 8

Bro08 **Alan W. Brown (2008):** MDA Redux: Practical Realization of Model Driven Architecture.
In: Seventh International Conference on Composition-Based Software Systems, ICCBSS2008, pages 174-183

Cen12 **PrfCEN/TR 15449-4**: Geographic information - Spatial Data Infrastructure (SDI) - Service centric view.
Under approval, Publication Jan 2013

Coc00 **Cockburn, A. (2000):** Writing Effective Use Cases.
Addison-Wesley, 1st edition

Cou05 **George Coulouris, Jean Dollimore, Jim Kindberg (2005):** Distributed Systems, Concepts and Design.
Addison Wesley, 4. rev. edition

Cro96 **Jon Crowcroft (1996):** Open Distributed Systems.
Artech House

Den89 **Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, Joe A. Turner, Paul R. Young (1989):** Computing as a Discipline.
In: Communications of the ACM, Volume 32, Number 1, pages 9-23

Eng11 **Wilco Engelsman, Dick Quartel, Henk Jonkers, Marten van Sinderen (2011):** Extending enterprise architecture modelling with business goals and requirements.
In: Enterprise Information Systems, Volume 4, Number 3, pages 9-36

ENV 1.1 **ENVISION Consortium (2010):** D1.1 Report presenting definition of pilot cases. In: http://www.envision-project.eu/wp-content/uploads/2010/01/D1.1-FINAL.pdf, accessed on 10.07.2012

ENV 1.2 **ENVISION Consortium (2011):** D1.2 Environmental Services and Models Scenarios and Pilots, Requirements specification. In: http://www.envision-project.eu/wp-content/uploads/2011/06/D1.2-v02.pdf, accessed on 10.07.2012

ENV 1.4 **ENVISION Consortium (2012):** D1.4 Development of an operational system by applying the ENVISION results to the oil spill and landslide pilot cases. In: http://www.envision-project.eu/wp-content/uploads/2010/01/D1.4-revised.pdf, accessed on 10.07.2012

ENV 3.1 **ENVISION Consortium (2012):** D3.1 MaaS Composition Portal – Architecture specification.
In: http://www.envision-project.eu/wp-content/uploads/2010/01/D3.1-FINAL.pdf, accessed on 10.07.2012

ENV 4.3 **ENVISION Consortium (2012):** D4.3 Ontologies and Annotations for Environmental Models. In: http://www.envision-project.eu/wp-content/uploads/2012/05/D4.3-2.0.pdf, accessed on 10.07.2012

ENV 6.1 **ENVISION Consortium (2010):** D6.1 ENVISION Adaptive Execution Infrastructure – Architecture Specification. In: http://www.envision-project.eu/wp-content/uploads/2010/01/D6.1-FINAL.pdf, accessed on 10.07.2012

ENV12a **ENVISION Consortium (2012):** Deliverables In: http://www.envision-project.eu/resources/deliverables, accessed on 14.08.2012

ENV12b **ENVISION Consortium (2012):** Frequently asked questions. In: http://www.envirofi.eu/FrequentlyAskedQuestions/tabid/4685/Default.aspx, accessed on 14.08.2012

ENV12c **ENVISION Consortium (2012):** About the project. In: http://www.envirofi.eu/AbouttheProject/tabid/3924/Default.aspx, accessed on 14.08.2012

Erl08 **Thomas Erl (2008):** SOA design patterns. Prentice Hall

FI-PPP12 **Future Internet PPP (2012):** About us. In: http://www.fi-ppp.eu/about-us, accessed on 13.08.2012

FIWiki12 **FI-Ware Forge Wiki (2012):** FI-WARE Architecture and Open Specifications.
In: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Architecture_and_Open_Specifications, accessed on 13.08.2012

FMC12 **FMC Consortium (2012):** What is FMC about?
In: http://www.fmc-modeling.org/, accessed on 13.08.2012

IBM12	**IBM (2012):** Rational Software Architect: Features In: http://www-142.ibm.com/software/products/de/de/ratisoftarch/, accessed on 18.08.2012
ISO98a	**ITU-T X.901 | ISO/IEC 10746-1 (1998):** Information Technology – Open Distributed Processing – Reference Model: Overview Version 1
ISO98b	**ITU-T X.904 | ISO/IEC 10746-4 (1998):** Information Technology – Open Distributed Processing – Reference Model: Architectural Semantics Version 1
ISO05	**ISO 19119:2005 (2005)**: Geographic information – Service
ISO09	**ITU-T Recommendation X.906 | ISO/IEC 19793 (2009):** Information technology — Open distributed processing — Use of UML for ODP system specifications Version 02.01, 2009
ISO10a	**ITU-T X.902 | ISO/IEC 10746-2 (2010):** Information Technology – Open Distributed Processing – Reference Model – Foundations Version v01.04, FDIS
ISO10b	**ITU-T X.903 | ISO/IEC 10746-3 (2010):** Information Technology – Open Distributed Processing – Reference Model – Architecture Version v01.04, FDIS
ISO11	**ISO/IEC/IEEE 42010:2011 (2011):** Systems and software engineering — Architecture description
Kai05	**Stephen H. Kaisler, Frank Armour, Michael Valivullah (2005):** Enterprise Architecting: Critical Problems. In: Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS 05), pages 224b
Kar10	**N. Karcanias, A.G. Hessami (2010):** Complexity and the notion of system of systems: Part (II): defining the notion of system of system In: World Automation Congress (WAC), pages.1-7
Kho09	**Sedigheh Khoshnevis, Fereidoon Shams Aliee, Pooyan Jamshidi (2009):** Model Driven Approach to Service Oriented Enterprise Architecture. In: IEEE Asia-Pacific Services Computing Conference (APSCC 2009), pages 279-286
Kra05	**Dirk Krafzig, Karl Banke, Dirk Slama (2005):** Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall International
Lan07	**Marc Lankhorst, Hans van Drunen (2007):** Enterprise Architecture Development and Modelling. In: via-nova-archiecture.org, Magazine, published 21.March 2007, accessed at http://www.via-nova-architectura.org/files/magazine/Lankhorst.pdf on 10.06.2012

Lan09 **Marc M. Lanckhorst (2009):** Enterprise architecture at work modelling, communication and analysis.
Springer Verlag, 2nd edition

Lef11 **D. Leffingwell (2011):** Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series).
In: Addison-Wesley Professional; 1 edition

Lei06 **Susanne Leist, Gregor Zellner (2006):** Evaluation of Current Architecture Frameworks.
In: Proceedings of the 2006 ACM symposium on Applied computing (SAC 06), pages 1546 - 1553

Lin11 **Peter F. Linington, Zoran Milosevic, Akira Tanaka, Antonio Vallecillo (2011):** Building Enterprise Systems with ODP: An Introduction to Open Distributed Processing.
Chapman & Hall/CRC Press

Lüb08 **Daniel Lübke, Kurt Schneider, Matthias Weidlich (2008):** Visualizing Use Case Sets as BPMN Processes.
In: Requirements Engineering Visualization (REV'08), pages 21 - 25

Mod11 **Modeliosoft (2011):** About Modelio: Features.
In: http://www.modelio.org/about-modelio/features.html, accessed on 10.08.2012

Mod12 **Modeliosoft (2012):** Enterprise Architecture tool: Goal diagrams – Definition and example using BMM standard.
In: http://www.modeliosoft.com/en/download/132.html, accessed on 10.08.2012

Naw06 **Jerzy Nawrocki, Tomasz Nedza, Miroslaw Ochodek, Lukasz Olek (2006):** Describing Business Processes with Use Cases.
In: Proceedings of Business Information Systems (BIS 2006), pages 13-27

Obr03 **Leo Obrst (2003):** Ontologies for Semantically Interoperable Systems.
In: Proceedings of the twelfth international conference on Information and knowledge management (CIKM'03), pages 366-369

OMG07 **OMG Group (2007):** Unified Modeling Language Superstructure
Version 2.1.2, in: http://www.omg.org/spec/ UML/2.1.2/Superstructure/, accessed on 20.04.2012

OMG10 **OMG Group (2010):** Business Motivation Model
Version 1.1, http://www.omg.org/spec/ BMM/1.1/, accessed on 20.04.2012

OMG11 **OMG Group (2011):** Business Process Modeling Notation
Version 2.0, http://www.omg.org/spec/BPMN/2.0/, accessed on 20.04.2012

OMG12a **OMG Group (2012):** Service oriented architecture Modeling Language
Version 1.0, in http://www.omg.org/spec/SoaML/1.0/, accessed on 20.04.2012

OMG12b **OMG Group (2012):** Unified Profile for DoDAF and MODAF (UPDM) Version 2.0, in: http://www.omg.org/spec/UPDM/2.0/, accessed on 20.04.2012

Open11 **The Open Group (2011):** TOGAF Version 9.1 Catalog Number: G116, accessed via https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116 on 29.05.2012

Open12a **The Open Group (2012):** ArchiMate 2 Specification Catalog Number C118, accessed via https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12480 on 29.05.2012

Open12b **The Open Group (2012):** What is ArchiMate? In:http://www.archimate.nl/en/about_archimate/what_is_archimate.html, accessed on 22.07.2012

Pap07 **Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann (2007):** Service-Oriented Computing: State of the Art and Research Challenges. In: Computer, Volume 40, Issue 11, p 38 - 45

PEIS 2.1 **ENVIROFI Consortium (2011):** D2.1 Scenario Description and Use Cases Specification, restricted document

PEIS 2.3.1 **ENVIROFI Consortium (2012):** D 2.3.1 Use Case Requirements Report WP2 Annex, restricted document

PEIS 2.3.2 **ENVIROFI Consortium (2012):** D2.3.2 Functional and Organisational Specification for PIS Pilot II. In:http://www.envirofi.eu/Portals/89/Docs/Project/Public_deliverables/ENVIROFI%20D2.3.2_Functional_and_Organisational_Specification_for_PIS_Pilot_II.pdf, accessed on 23.06.2012

PEIS 4.1.1 **ENVIROFI Consortium (2011):** D4.1.1 Environmental Requirements I. In:http://www.envirofi.eu/Portals/89/Docs/Project/Public_deliverables/ENVIROFI%20D4.1.1_Environmental_Requirements_I.pdf, accessed on 23.06.2012

PEIS 4.2 **ENVIROFI Consortium (2012):** D4.2 Environmental Architecture In:http://www.envirofi.eu/Portals/89/Docs/Project/Public_deliverables/ENVIROFI%20D4.2_Environmental_Architecture.pdf, accessed on 23.06.2012

PEIS 5.2.1 **ENVIROFI Consortium (2012):** D5.2.1 Initial Specification of the Specific Enablers for Environmental Domain I, restricted document

PEIS 5.2.2 **ENVIROFI Consortium (2012):** D5.2.2 Initial Specification of the Specific Enablers for Environmental Domain II, restricted document

PEIS 6.1.1 **ENVIROFI Consortium (2011):** D6.1.1 Sketch of the ENVIROFI Architecture, restricted document

Rom05 **J.R. Romero, A. Vallecillo (2005):** Modelling the ODP Computational Viewpoint with UML 2.0: The Templeman Library Example. In: Proceedings of the 2nd International Workshop on ODP and Enterprise Computing (WODPEC 2005)

Rom12	**José Raúl Romero, Juan Ignacio Jaén, Antonio Vallecillo (2012):** A Tool for the Model-Based Specification of Open Distributed Systems. In: The Computer Journal, 2012
Sad10	**Andrey Sadovykh, Brian Elvesæter, Philippe Desfray, Arne-Jørgen Berre, Einar Landre (2010):** Enterprise Architecture Modeling with SoaML Using BMM and BPMN – MDA Approach in Practice. In: 6th Central and Eastern Europe Software Engineering Conference (CEE-SECR 10), pages 79 - 85
Sel12	**Select Business Solutions (2012):** Select Architect (BMM, BPMN, UML) In: http://www.selectbs.com/analysis-and-design/select-architect, accessed on 14.08.2012
Sparx12a	**Sparx Systems (2012):** Features at a glance In: http://www.sparxsystems.com/, accessed on 14.08.2012
Sparx12b	**Sparx Systems (2012):** TOGAF MDG Technology: Features In:http://www.sparxsystems.com.au/products/mdg/tech/togaf/index.html, accessed on 14.08.2012
Ste05	**M.W.A. Steen, P. Strating, M.M. Lankhorst, H. ter Doest, M.-E. Iacob (2005):** Service-Oriented Enterprise Architecture. In: Zoran Stojanovic, Ajantha Dahanayake (Ed.): Service-Oriented Software System Engineering: Challenges and Practices, pages 132-154 IGI Global
Tan04	**A. Tang, Jun Han, Pin Chen (2004):** A Comparative Analysis of Architecture Frameworks. In: 11th Asia-Pacific Software Engineering Conference, pages 640-647
Zach87	**John A. Zachman. (1987):** A framework for information systems architecture. In: IBM Systems Journal, Volume 26, Issue 3,pages 276-295.
Zach99	**John A. Zachman (1999):** A Framework for Information Systems Architecture. In: IBM Systems Journal, Volume 38, Issue 3, pages 454-70
Zach12	**John A. Zachman (2008):** John Zachman's Concise Definition of The Zachman Framework In: http://www.zachman.com/about-the-zachman-framework, accessed on 21.05.2012