D4.3: Dependency Models Iter. 2

WP 4 – Processes and Methods for Digitally Preserving Business Processes

Delivery Date: 29/03/2013

Dissemination Level: Restricted

TIMBUS is supported by the European Union
under the 7th Framework Programme
for research and technological development and demonstration activities (FP7/2007-2013)
under grant agreement no. 269940

Deliverable Lead		
Name	**Organisation**	**e-mail**
Gonçalo Antunes	INESC-ID	goncalo.antunes@ist.utl.pt

Contributors		
Name	**Organisation**	**e-mail**
Artur Caetano	INESC-ID	artur.caetano@ist.utl.pt
Marzi Bakhshandeh	INESC-ID	Marzieh.bakhshandeh@ist.utl.pt
Rudolf Mayer	SBA	mayer@ifs.tuwien.ac.at
Hossein Miri	KIT	hossein.miri@kit.edu
Mykola Galushka	SAP	mykola.galushka@sap.com
Daniel Draws	SQS	daniel.draws@sqs.com
Carlos Coutinho	CMS	carlos.coutinho@caixamagica.pt

Internal Reviewer		
Name	**Organisation**	**e-mail**
Rudolf Mayer	SBA	mayer@ifs.tuwien.ac.at
José Barateiro	LNEC	jbarateiro@lnec.pt

Document History			
Version	**Date**	**Author**	**Changes**
V1.0	22/03/2013	Gonçalo Antunes	First Version

Disclaimer

Table of Contents

List of Figures

List of Tables

List of Acronyms

BCM	Business Continuity Management
BPMN	Business Process Model and Notation
BWW	Bunge-Wand-Weber
CAD	Computer-Aided Design
CEO	Chief executive office
CUDF	Common Upgradeability Description Format
D	Deliverable
DIO	Domain Independent Ontology
DSO	Domain Specific Ontology
ERM	Enterprise Risk Management
FIPA	Foundation for Intelligent Physical Agents
FOSS	Free and Open Source Systems
iERM	intelligent Enterprise Risk Management
IT	Information Technology
ITIL	IT Infrastructure Library
ITSMO	IT Service Management Ontology
M	Month
OAIS	Open Archival Information System
OMG	Object Management Group
OWL	Web Ontology Language
PMO	Patent Metadata Ontology
PREMIS	PREservation Metadata: Implementation Strategies
SoaML	Service-oriented architecture Modelling Language
SUMO	Suggested Upper Merged Ontology
SWO	Software Ontology
T	Task
TOGAF	The Open Group Architecture Framework
TOVE	TOronto Virtual Enterprise
UML	Unified Modelling Language
VRDF	Virtual Resource Description Framework
WP	Work Package
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
Y	Year

1 Executive Summary

The WP4 of the TIMBUS project investigates, identifies, and captures what is required for digital preservation to be performed in an enterprise system, resulting in Dependency and Context models. WP5 develops a digital preservation system architecture that is implemented by the WP6 tools. The tools that are implemented particularly as a result of T4.2 and later on T4.4 include T6.2 (Dependencies Monitoring & Reasoning & Solving Tools) and T6.5 (Context Capturing and Dependencies Extracting Tools).

More specifically, the aim of T4.2 is to develop a means of describing the dependencies between different components of business processes through different layers of an enterprise architecture. The first iteration identified the types of dependencies required, categorized the layers in an enterprise, and modelled the contexts and dependencies of the business processes pertaining to the digital preservation domain. It also determined what the important components that were needed for digitally preserving the business processes in focus were. Essentially, it served to provide a common understanding of the required concepts as well as identify areas of application. As expected, however, the formalism and associated technical solutions that resulted from the first iteration were mainly exploratory in nature, and thus only served the purpose of creating a view on the problem and some potential solutions. Furthermore, not all the requirements from the use-cases were elicited and gathered before the first iteration.

Thus, as was pointed out in D4.2, this second iteration is concerned with extending and refining the basis for describing those components and layers, and consequently re-structuring and improving the Dependency and Context models. The result is a comprehensive and integrative model developed with industrial use-cases in mind, and an extensible architecture along with a governance method for evolving it further. It has been iteratively revised since D4.5 was delivered in M12, and any further refinements and improvements to it and its associated tasks will be reported in D4.9 due in M36.

2 Introduction

Context is a crucial aspect of business process preservation. On the one hand, it supports the redeployment of a process into a suitable IT infrastructure, and on the other hand, it provides the semantics necessary to understand the process. As such, the TIMBUS context model assumes central importance in the preservation of business processes, providing a means for modelling context and dependencies so that all the information required for preserving and redeploying a process is captured.

During the work in Y1 of the project it became clear that the dependency model and the context model, worked on in Tasks 4.2 and 4.4, complement each other, and thus it was decided to move to one integrated model. The main reason for doing so was that the context of the dependencies also needs to be captured to enable the preservation of a process, and thus the dependencies would also have to be captured in the context model. As we are dealing with a unified model for context and dependencies, this deliverable will report on all the developments and achievements made during Y2 of the project concerning the context model. An updated report will be given in Y3 of the project, in D4.9.

The first iteration of the TIMBUS context model served to provide a common understanding of the required concepts as well as to identify areas of application. As expected, the technical solution that resulted from this iteration was mainly exploratory in nature and served the purpose of creating a view on problem and on potential solutions that is now shared between the relevant stakeholders of the TIMBUS project. However, such exploratory solution was not suitable to actually model the context of a business process, namely the context of a business process pertaining to the digital preservation domain. The technical unsuitability derived from the complexity and ambiguity of the context model that is a consequence of the high number of classes and relationships represented in it. Such complexity also led to concepts duplication and therefore to modelling ambiguity. Furthermore, not all the requirements from the use cases were elicited before the first version of the unified context model. Thus, as was pointed out in D4.2, the TIMBUS context model would be refined, restructured and improved.

The result of the restructuring efforts was a comprehensive model developed based on best-practices, standards, and industrial case stakeholder's requirements, with an extensible architecture, and a governance method for evolving the model and adapting it for whatever preservation scenario. The model also supports reasoning and inference, which can be used for checking inconsistencies on the model and inferring information that might be particularly useful for the TIMBUS preservation processes. The developed context model was applied to the industrial cases of WP8 and WP9 with the outcomes being reported on this deliverable. Additionally, for comparison reasons, the context model was also applied to the music classification process developed for Y1.

This deliverable is structured as follows. Section 3 describes the consolidation of requirements into the context model, which had two sources: (*i*) architectural principles, which specify the principles to which the model should comply according to best-practices; and (*ii*) stakeholder requirements, which were gathered from the industrial cases' stakeholders. Section 4 specifies the architecture behind this revision of the

context model, which was developed according to the requirements of the previous section, describing how the model can be extended for including domain specific concepts and how reasoning can be incorporated and performed across different aspects of the extended model. Section 5 describes a governance process for governing the application and extension of the context model when applying to a specific domain. Section 6 describes the domain independent ontology, which forms the core of the context model, and Section 7 describes the first domain specific ontologies that are being integrated within the context model. In Section 8, we describe the results of applying the context model to the industrial cases of WP8 and WP9, as well as one of the scenarios developed in the first year. In Section 9, the related work which inspired the work reported in this deliverable is described. In Section 10 we describe the main tools used to perform the work reported in this deliverable. Finally, in Section 11, we provide the conclusions and provide an outlook on the work to be done on the context model for the remainder of the project.

3 Consolidation of Requirements and Analysis

In this section, we describe the process of requirements gathering and analysis performed to obtain the new version of the context model. As such, an overview will be given on the architecture principles that guided the final design, and subsequently the requirements gathering approach will be described.

3.1 Architecture Principles

Ontology engineering is a difficult task, especially within a setting such as the one faced in TIMBUS, where there is the need to arrive at a representation of knowledge not from a single domain, but deal with concepts that cut across several different domains. The model derived in Y1 of the project therefore faced the challenge of the large domain of aspects that may be potentially relevant to the context of a process.

To allow for a more structured and extensible ontology, the following design principles to constrain the context model were considered:

- Concern-orientation. The context model shall represent the concepts necessary and sufficient to address an explicit set of modelling concerns. This means that the model shall be derived from the questions that need to be addressed and to provide answers to those questions. This also means that the model shall not support any concepts that are not explicitly derived from concern. The principle of concern-orientation and the principle of viewpoint-orientation (below) are described in detail in the ISO 42010:2011 standard (ISO, IEC and IEEE, 2011). This standard defines requirements on the description of systems and enterprise architecture.
- Expressiveness. The context model shall be able to represent the domain concepts without ambiguity. This entails defining the minimum set of types and relationships to describe a domain.
- Extensibility. The model must cope with extensions because context modelling entails using multiple concurrent perspectives on the same problem. This derives from being able to answer to multiple concerns. Therefore, domain-specific and domain-independent models must coexist and the overall context model must cope with multiple model transformation and integration. A specific concern is that the model is extensible to new application domains, beyond the ones that are the focus of the use cases in the TIMBUS project.
- Viewpoint-orientation. The model must support defining views over subsets of its concepts. This serves to facilitate the communication and the management of the models as viewpoints act as a separation of concerns mechanism. Viewpoints will facilitate addressing multiple concerns and managing the multiple extensions required to handle these concerns.
- Modularity. The models must follow the principles of high-cohesion and low-coupling. Observing these principles contributes to expressiveness and extensibility of the context models. It is especially important that adding new domain-specific aspects to the model does not interfere with the ontologies already present in the context model.

3.2 Stakeholder Requirements

In order to derive the requirements, it is necessary to determine the stakeholders' needs concerning the information they wish to obtain from the context model.

In year one, with the use cases not yet fully defined, a mixture of a top-down and bottom up approach was taken to scope and to structure the exploration of relevant context parameters, akin to the suggested middle-out approach taken by Uschold and Gruningers' methodology (Uschold and Gruninger, 2006). The top-down approach was based on using the Zachman framework (Zachman, 1987) as top-level ontology, to structure the lower levels of the hierarchy. The bottom-up approach consisted of scenarios that were developed by the individual partners, describing different processes that were deemed relevant for digital preservation. These scenarios, and the aspects of the processes that were identified relevant for digital preservation, formed the basis of the more detailed levels of the ontology.

During the course of the second year, two of the use cases have been specified in great detail, and thus became available to a more detailed analysis. Thus, these formed the base for the refined context model. The approach for identifying aspects relevant for the digital preservation, and thus elements of the context model, was slightly different, however. Instead of directly identifying these aspects, the industrial use case stakeholders were asked for elaborating a set of questions that they considered relevant to be answered if the process was to be preserved, as well as the expected outcomes. Those questions were then processed to find entities that should be in the context model, and classified into different groups.

Table 1 depicts an example of the questions obtained along with the expected outputs. Concepts were highlighted in green colour and instances were highlighted in blue colour.

Table 1: Stakeholder's Questions to the Context Model

Question	Expected Output
Which **business processes** depend on **business process BP**?	List of Business Processes
Which business actors **BA** are required to execute business process **BP**?	List of Business Actors
What are the technological entities **T** supporting **business process BP**?	List of structural and behavioural technological entities
What **application components C** support **business process BP**?	List of application components
What **application components C** depend on **requirement R**?	List of application components
What **legal requirements R** are verified by **business process BP**?	List of legal requirements
What are the **licenses L** required to execute **software application SA**?	List of licenses
Which **sensor types ST** can measure the **physical quantity PQ**?	List of sensor types

What are the calibration constants and sensor properties required to convert raw data into the **physical quantities PQ** for **sensor type ST**?	List of properties for sensor type ST
What **constraints** C exists on the access to the medical data?	List of constraints
Who has access to the medical data?	List of persons/roles

In total, we obtained more than 110 questions, the full list of which can be found in Appendix A. After gathering these questions, we identified which ones were domain independent, and which ones touch a specific domain. Some of the questions were indeed generic and applicable to potentially many use cases, while others were rather specific to a certain domain, such as the ones dealing with specific hardware, e.g., sensors. This is an indication that our context model should be able to provide generic elements that are used in a wide range of cases, and allow for extension for specific domains. These domain specific ontologies are detailed in Section 7.

3.3 Requirements Consolidation

The TIMBUS context model aims at supporting the representation of the information required to preserve, redeploy and analyse business processes. As such, its main requirements are:

- Represent domain-specific business processes. The context model must support the generic description of business processes plus the domain-specific features of each approached scenario.
- Integrate multiple representations. Representing the context of a business process implies capturing the processes and their environment. This implies that the representation used for organising that information will have intersecting aspects that need to be integrated. These aspects may include:
    - Strategy (e.g. requirements, rules, drivers, principles, indicators),
    - Organization (e.g. people, locations, roles),
    - Operations (e.g. processes, services, products),
    - Business support systems, including the application infrastructure (e.g. applications, software) and the technological infrastructure (e.g. hardware nodes, communication devices).
    - Domain-specific aspects related to the approached scenarios.
- Provide the means to analyse the representations of business processes. The context model representations are used to facilitate the assessment of business process preservation and redeployment from a conceptual and technical perspective. The verification and validation of both preservation and redeployment is important, and the context model shall provide a basis for performing such tasks.

4 The TIMBUS Context Model Architecture

Based on the requirements identified in the previous chapter, and based on the decision made in Y1 to ground the context model in ontologies, the proposed context model architecture is established on the following concepts:

- Domain-Independent Ontology (DIO).
- Domain-Specific Ontology (DSO).
- Ontology Integration.
- Model Transformation.

The domain-independent ontology (DIO) represents a neutral, domain-independent language that is able to represent the core concepts of the context model. As indicated previously, these concepts span the domain of enterprise architecture. As such, the DIO represents a minimum set of concepts pertaining to enterprise architecture. The DIO is designated domain-independent since it does not address any specific domain-dependent concerns. In ontology engineering, sometimes such an ontology is referred to as an "upper level ontology".

A domain-specific ontology (DSO) represents a domain-specific language that addresses a particular set of concerns. For example, a Software Licensing DSO would describe the concepts required to model the universe of licenses, and may include concepts that cover licensing models, licensing agreements, copyrights, license types (e.g. free software, open source), etc. The TIMBUS context model will comprise a set of DSOs. Each DSO should be designed with the minimum set of concepts required to describe a given domain. The context model should also be easily extended, so that an additional DSO is added to the model without affecting the existing DSOs. However, the number of DSOs that will be part of the TIMBUS context model will depend on the actual domains needed to represent all the concerns of stakeholders of the addressed scenarios.

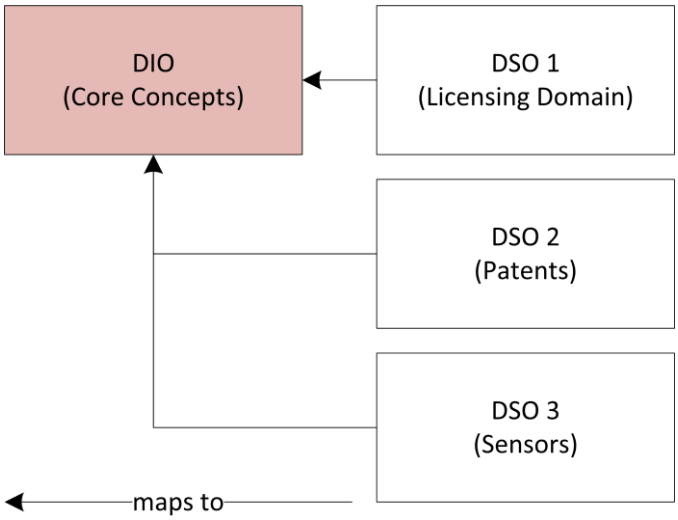

Figure 1: Simple DIO-DSO integration.

For the questions gathered (cf. Annex A), we can identify which ones can be answered already solely by the information present in the Domain Independent Ontology. For the other questions, we performed a grouping into different domains, according to the type of information missing to answer that specific question. This forms the basis of identifying suitable DSOs, as will be detailed in Section 7.

Ontology integration deals with the combination of the different ontologies in such a way that the overall context model is consistent and able to address the domains covered by each ontology. In the simplest case, each DSO needs to be integrated with the core concepts represented in the DIO as depicted in Figure 1. Several DSOs can also be integrated in order to add more expressive power to specific domains. For instance, Figure 2 depicts a scenario where a DSO for the Licensing domain integrates with two more specific DSOs for the Free and Open Source Systems (FOSS) Licensing and Commercial Licensing domains. Another case is DSO 4 that is mapped to DSO 2 and DSO 3 from different domains.

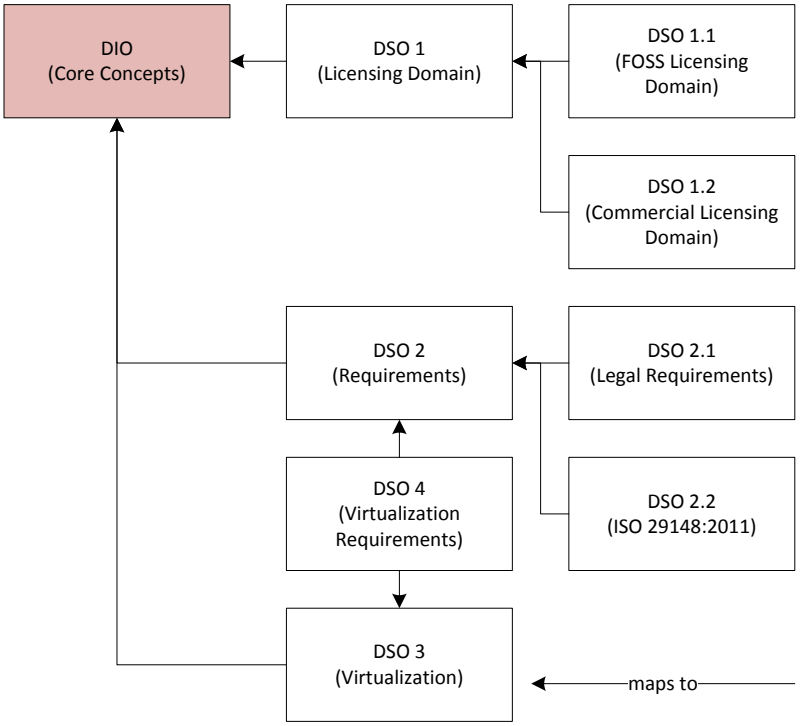

Figure 2: Hierarchical DSO integration.

This approach facilitates layering multiple DSOs according to the modelling needs. Figure 3 presents another example where an ontology to represent Civil Engineering applications builds on a CAD application ontology, a virtualization ontology and CUDF, whereas a Pharma application ontology uses only the virtualization ontology and CUDF.

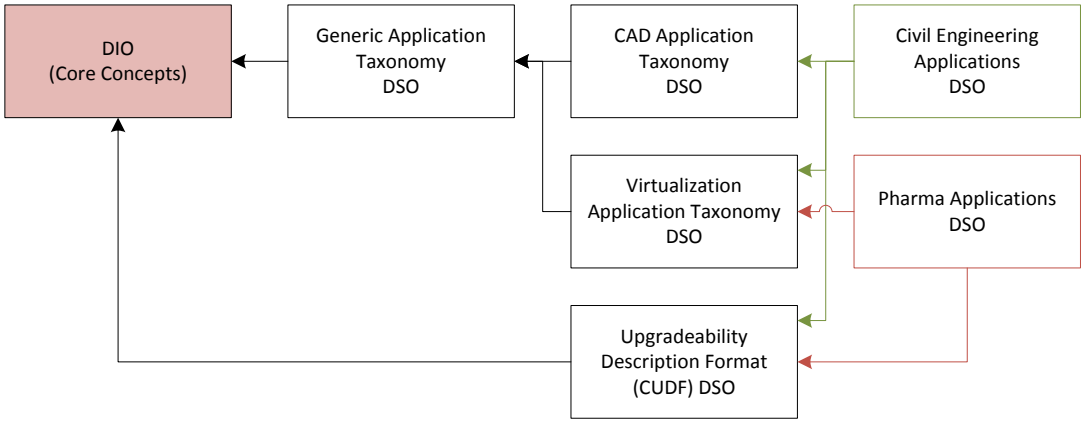

Figure 3: Hierarchical DSO integration.

The ontology integration described above makes use of **model transformation** to relate a DSO to the DIO or to relate multiple DSOs to each other. Model transformation entails defining a mapping strategy from a source model to a destination model (Guizzardi, 2006) (Rosemann et al., 2004).

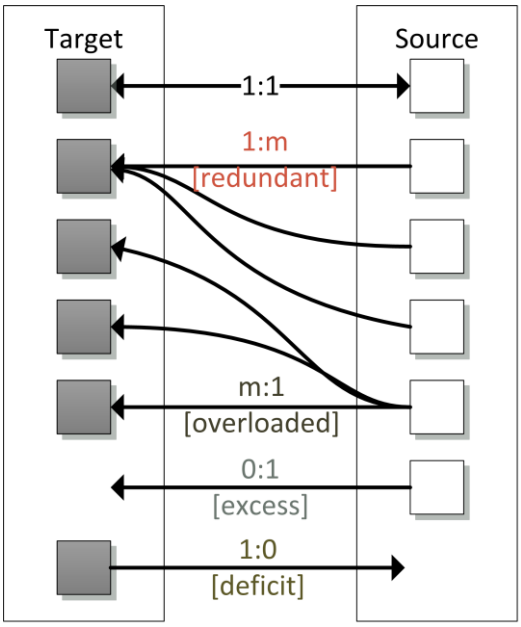

Figure 4: Types of potential representational deficiencies (Weber, 1997).

Depending on the DSO to be integrated, the mapping might create different types of representational deficiencies, which are of course expected since the DSO might address very specific concepts not present in the DIO. Any deviation from a 1:1 mapping should be considered such a deficiency. Two aspects might be analysed: ontological completeness and ontological clarity. The Bunge-Wand-Weber (BWW) representation model (Bunge, 1977) can be used as an inspiration in the study of ontological completeness by analysing the extent to which a source modelling language has a deficit of entities mapping to the set of entities proposed in target modelling language. Ontological clarity might be analysed by determining the extent to which the

source modelling language constructs are overloaded (i.e. they map to two or more constructs in the target), redundant (i.e. two or more language constructs map to the same construct in the target model), or excess (i.e. they map to none of the constructs in the target model) (v. Figure 4).

4.1 Transformation and Mapping

The ontology architecture is designed to adhere to the principles of high cohesion and low coupling. High cohesion means that each architectural module deals only with a set of related domain-specific concerns. Low coupling means that the number of dependencies between architectural modules is designed to be minimal. Together, these two properties promote modularization along with the ability to incrementally extend the architectural modules. The ontology architecture comprises a core domain-independent ontology (DIO). This core ontology is able to provide a high-level description of a system and to support inference around the core structure, behaviour and consistency. Domain-specific concepts are introduced into the architecture through domain-specific ontologies (DSO). Each DSO is designed to be highly cohesive, meaning that it is limited to describing the concepts, relationships and rules pertaining to a single domain. Therefore, each DSO addresses a limited set of concerns.

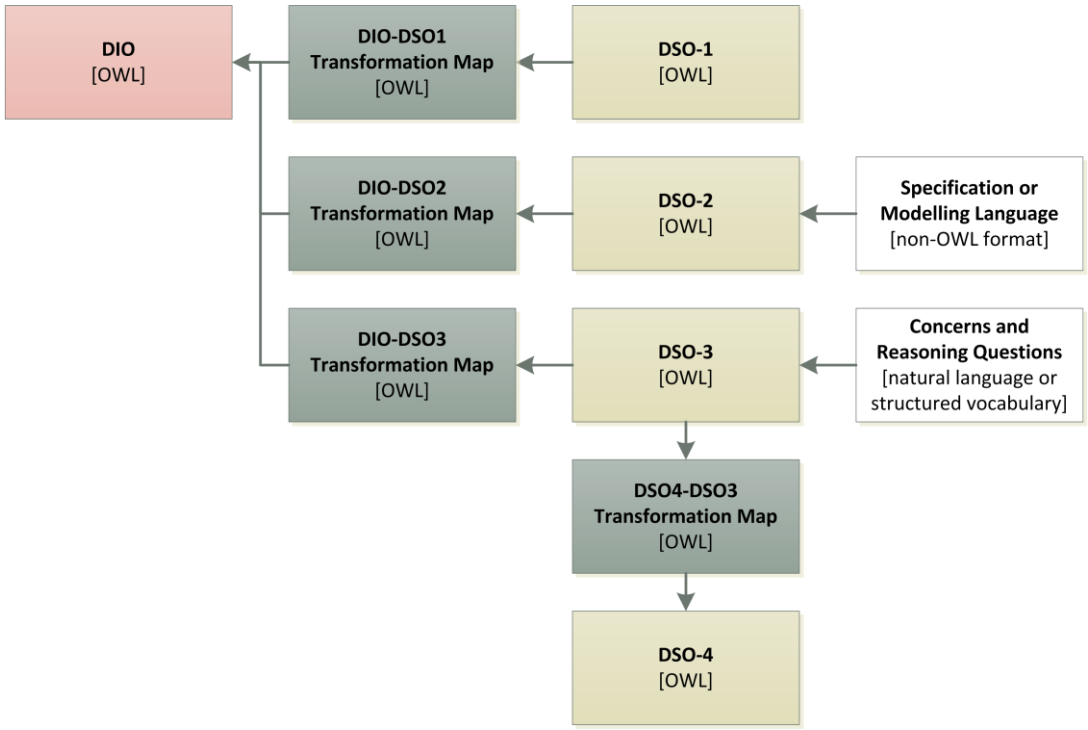

Figure 5. Relationship between DIO, DSO and transformation maps in the context model architecture. Each relationship indicates the mapping of concepts from a source to a target ontology.

A DSO has to be related to the high-level concepts of the DIO. Creating these relationships implies transforming the concepts and relationships of the DSO to the concepts and relationships of the DIO. This transformation process is straightforward when there is a one-to-one relationship or map between the concepts of the DSO and the DIO. As a result of this approach, each DSO relates to the DIO through one map.

A DSO can also be mapped to another DSO through a map (v. Figure 5). As decided in Y1, OWL was chosen as the representation language for the context model. Each map is actually an OWL ontology that specifies the transformation rules from the source to the target ontology.

4.2 Reasoning

According to the principles and requirements defined so far, the DIO should be concern-oriented. Let us imagine, for instance, that the core ontology comprises three layers, i.e., Layer1, Layer2 and Layer 3. The classes and properties of each DSO are mapped onto the classes and properties of the DIO.

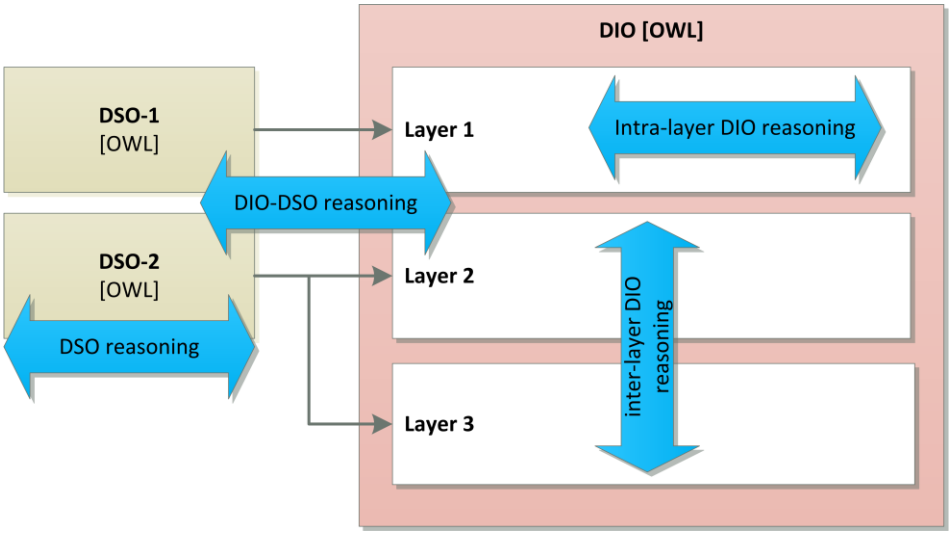

Figure 6: Reasoning configurations.

This allows for the following reasoning configurations (v. Figure 6):

- DIO reasoning. Inference is exclusively based on the DIO concepts. Considering the three layers of the DIO, two options exist:
    - Intra-layer DIO reasoning, when inference is limited to the concepts of just one of the DIO layers.
    - Inter-layer DIO reasoning, when inference concepts related to two or more DIO layers.
- DSO reasoning. Two options exist:
    - Intra DSO reasoning, when inference is exclusively based on a single DSO.
    - Inter DSO reasoning, whenever transformations between two or more DSOs exist, then reasoning may span several DSOs without interfering with the DIO.
- DIO-DSO reasoning. Inference is based on the DIO concepts plus the concepts of one or more DSOs. Such configuration would require a transformation map between each DIO-DSO pair. The resulting reasoning may span more than one DIO layer, depending on the transformation map.

5 The Governance Method for the TIMBUS Context Model

This section specifies the method for governing the TIMBUS context model according to the practices of Situational Method Engineering (Henderson-Sellers and Ralyté, 2010), where the method is adapted to different situations, with each situation being described in a method fragment. The governance method specifies the process for applying and governing the extension of the context model, with the addition of DSOs. All models presented in this section use the ArchiMate language. ArchiMate (The Open Group, 2012) is a simple, comprehensive, and extensible enterprise architecture language, which is becoming the de-facto standard in enterprise architecture modelling. The method comprises four major functions as depicted in Figure 7, namely:

- **Determine Context** assesses the relevant aspects of context that need to be modelled according to the concerns of the stakeholders.
- **Capture Context** instantiates the context model.
- **Use Context** queries the model and obtains answers that satisfy the concerns of the stakeholders.

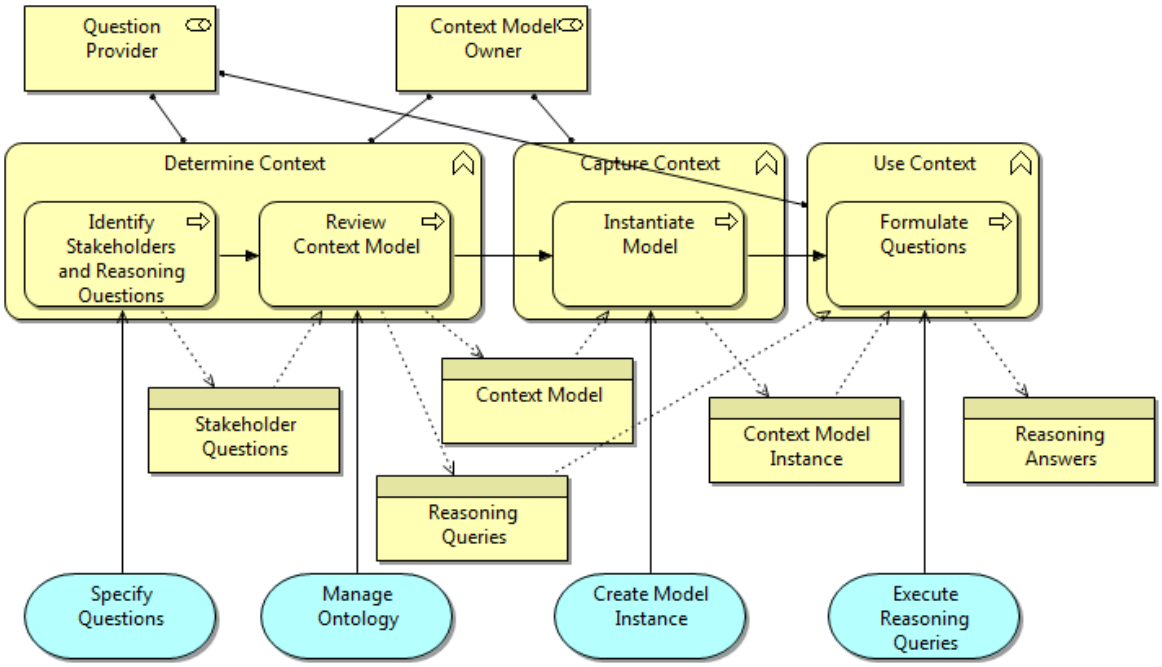

Figure 7: A business description of the governance method describing the Roles, Functions, and Objects.

The roles associated with the major functions of the method are the following:

- **Question Provider** is responsible to provide reasoning questions and examples.
- **Context Model Owner** is responsible for formalizing the reasoning questions, specifying the ontologies and integrating the ontologies.

The reason for having such a detailed governance process is related with the need for the context model evolution to be a fully controlled process. The next subsections detail each process.

5.1 Identify Stakeholders and Reasoning Questions Sub-Process

This process identifies the stakeholders, their concerns and the reasoning questions. This process concludes when each stakeholder has conveyed the information he would like to obtain from the context model.

There are two approaches to achieve this objective which are modelled as separate method fragments.

5.1.1.1 Template-Based Elicitation (Method Fragment #1)

This fragment gathers the stakeholder questions using a template provided to the actors fulfilling the business owner role. This template might be implemented using a spreadsheet and, ideally, it should be collaboratively edited, so that stakeholders can cross validate the posed questions. The template serves to capture reasoning questions pertaining to the context model. It should be filled in as follows:

- A Concerns column indicates the primary domain area that the question relates to. One question may cross-cut multiple domains.
- A Question column is where the reasoning question should be described.
- An Output column describes the results that questions should produce. This is particularly useful to understand the domains and ranges of the question.

Figure 8 depicts the fragment and Table 2 describes the fragment.

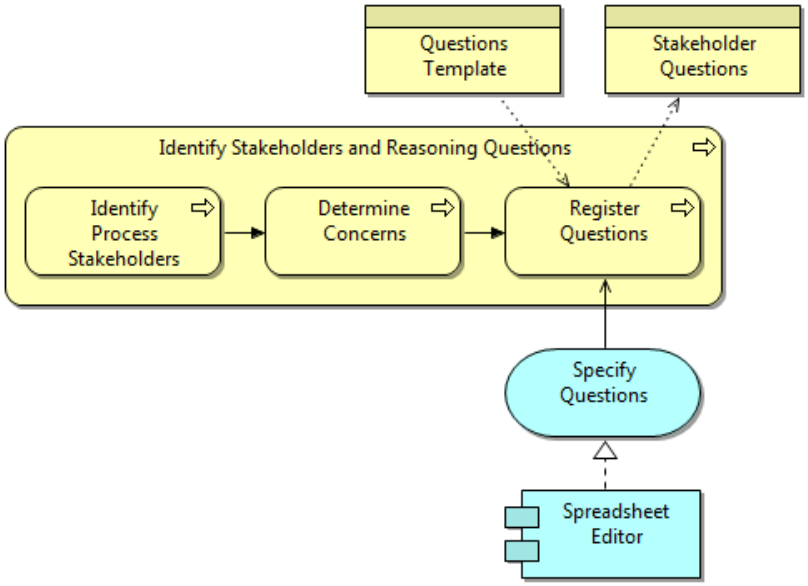

Figure 8: Identify Stakeholders and Reasoning Questions (Fragment #1)

Table 2: Identify Stakeholders and Reasoning Questions Fragment #1

Fragment	Template-based identification of stakeholders and reasoning questions
Roles	Business Owner
Pre-conditions	None
Process Steps	**Identify Process Stakeholders**: In this step, the main stakeholders of the to-be preserved process are identified. Stakeholders might range from technical (e.g., IT administrator) to organizational (e.g., head of department). The identification might be carried out through interviews or through available documentation.
	Determine Concerns: In this step, the identified stakeholders are interviewed and/or relevant documentation is consulted. The aim is to determine the main concerns of the involved stakeholders towards the preservation and future redeployment of a process.
	Register Questions: In this step, reasoning questions are elaborated in line with the identified concerns. The reasoning questions should be defined together with the expected output, in other words, the answers that the stakeholders hope to receive from the questions.
Input	The questions template which should be accessed by the register questions step.
Output	The reasoning questions which should be written in the register questions step.
Post-conditions	The reasoning questions are elaborated in a way that can be effectively used in later processes.
Tools	A browser for online editing the template.

5.2 Tool-Based Elicitation (Method Fragment #2)

The second method fragment deals with the elaboration of the stakeholder questions using a specialized tool with a controlled vocabulary specifically for expressing the reasoning questions. This tool is not available by now but could be a desired artefact to be developed within the TIMBUS project. Figure 9 depicts the fragment and Table 3 describes the fragment.

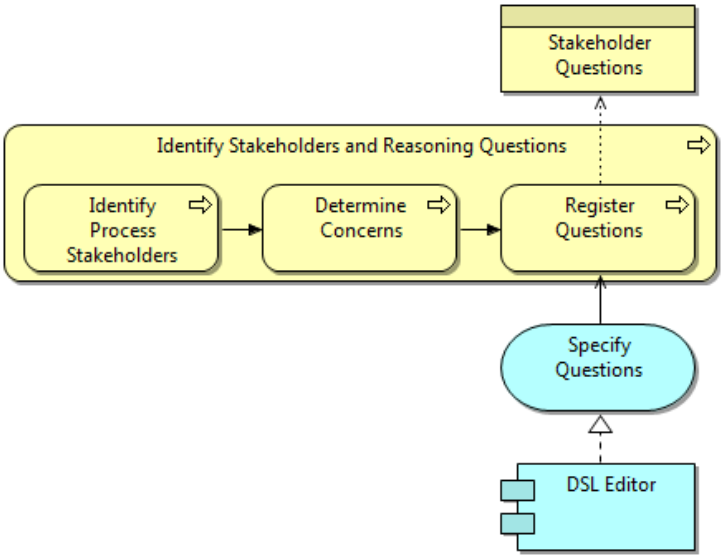

Figure 9: Identify Stakeholders and Reasoning Process (Fragment #2)

Table 3: Identify Stakeholders and Reasoning Questions Fragment #2

Fragment	Tool-based identification of stakeholders and reasoning questions
Roles	Business Owner
Pre-conditions	None
Process Steps	**Identify Process Stakeholders:** This step is exactly the same as in the previous fragment.
	Determine Concerns: This step is exactly the same as in the previous fragment.
	Register Questions: In this step, reasoning questions are elaborated in line with the identified concerns, using a Domain Specific Language (DSL) Editor. This tool will allow the expressing of reasoning questions using a controlled vocabulary, which will facilitate the transformation of the questions into queries that can be given as input to the reasoners.
Input	None
Output	The reasoning questions which should be written in the register questions step.
Post-conditions	The reasoning questions are elaborated in a way that can be effectively used in later processes.
Tools	A DSL editor which should allow the specification of the questions using a controlled vocabulary.

5.3 Review Context Model Process

This process determines what is required from the model and, based on that, elaborates a model that can be later instantiated when capturing context. Two method fragments are described for achieving this objective: the first fragment for the case when the core model is sufficient and the second fragment for the case when domain specific models are needed for being able to capture all the required knowledge.

5.3.1 Add Reasoning to Context Model (Method Fragment #1)

This method fragment deals with adding the reasoning queries to the existing context model for later instantiation on a specific scenario. In this case, the existing ontologies (DIO and DSOs) are sufficient to provide answers to address the stakeholders concerns. Therefore, no new concepts will be added to the ontologies. After the process concludes, the reasoning queries will be supported using the DIO and/or existing DSOs. Figure 10 depicts the fragment and Table 4 describes the fragment.

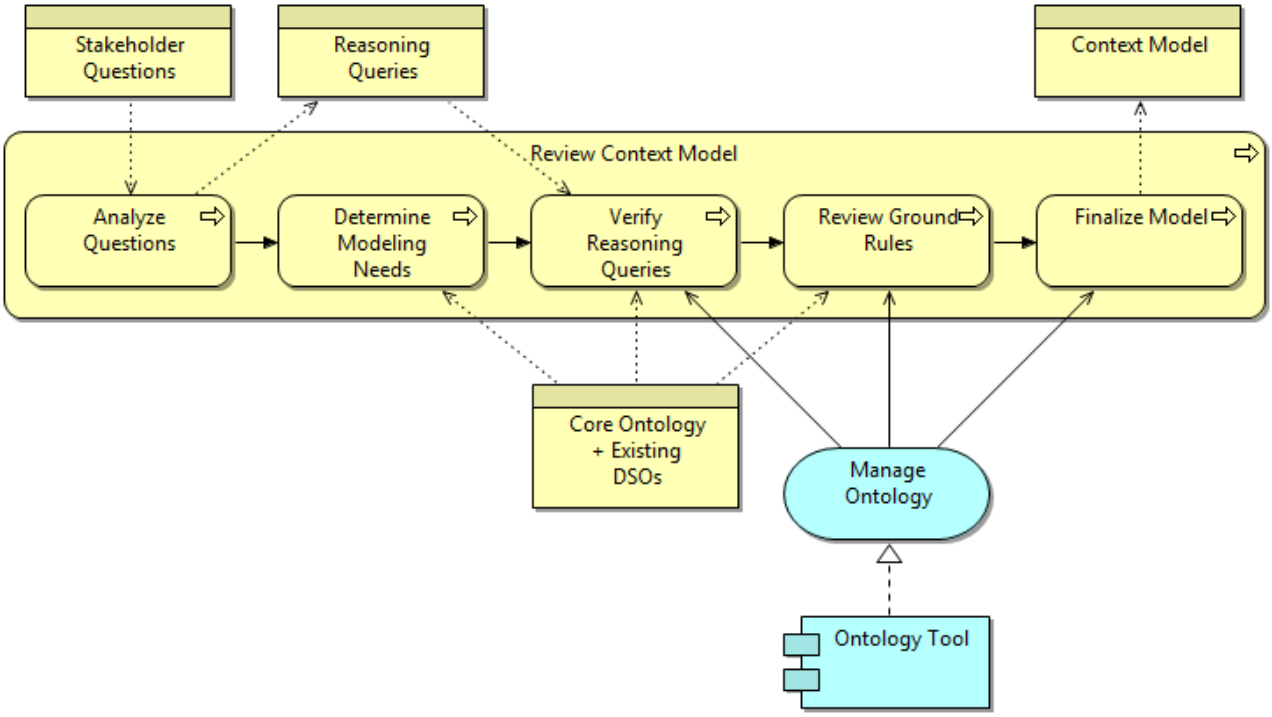

Figure 10: Review Context Model (Method Fragment #1)

Table 4: Review Context Model Fragment #1

Fragment	Add Reasoning to the Context Model
Roles	DP Analyst
Pre-conditions	Reasoning questions
Process Steps	Analyse Questions: In this step, the DP analyst performs the analysis of the questions elaborated by the business owner. The outcome of this step is the elaboration of reasoning queries that can be directly used with the reasoners.
	Determine Modelling Needs: In this step, the needs of the stakeholders are determined in terms of the core ontology and existing DSOs.
	Verify Reasoning Queries: In this step, the reasoning queries resulting from the analyze questions step are verified and validated in terms of the core ontology and existing DSOs.
	Review Ground Rules: In this step, the ground rules of the core ontology and existing DSOs are also verified, or in other words, the model is validated in terms of the existence of exceptions.
	Finalize Model: In this step, the finalized context model is released.
Input	The stakeholder questions which are given as input to the analyse questions step; The Core ontology and existing DSOs, which are given as input to the determine modelling needs step, to the verify reasoning queries step, and to then review ground rules step.
Output	The reasoning queries which is an output of the analyse questions step.
Post-conditions	The context model is ready for being instantiated.
Tools	Ontology Tool, for manipulating the core context ontology and existing DSOs, testing the queries and ground rules.

5.3.2 Extend and Add Reasoning to Context Model (Method Fragment #2)

This method fragment deals as well with adding the reasoning queries to the existing context model for later instantiation on a specific scenario. However, in this case the existing domain-specific ontologies aren't expressive enough to cover the stakeholders concerns. Therefore, new concepts are required. This process entails the following steps:

- Determine the actual gaps and modelling needs.
- Create a new DSO or update existing DSOs in order to address the new concerns.
- Create a set of transformation maps between the DSO and DIO. This will link the new concepts to the core DIO concepts. The new DSO can also be mapped to other DSOs.
- Specify the reasoning queries according to the new DSO(s).

Figure 11 depicts the fragment and Table 5 describes the fragment.

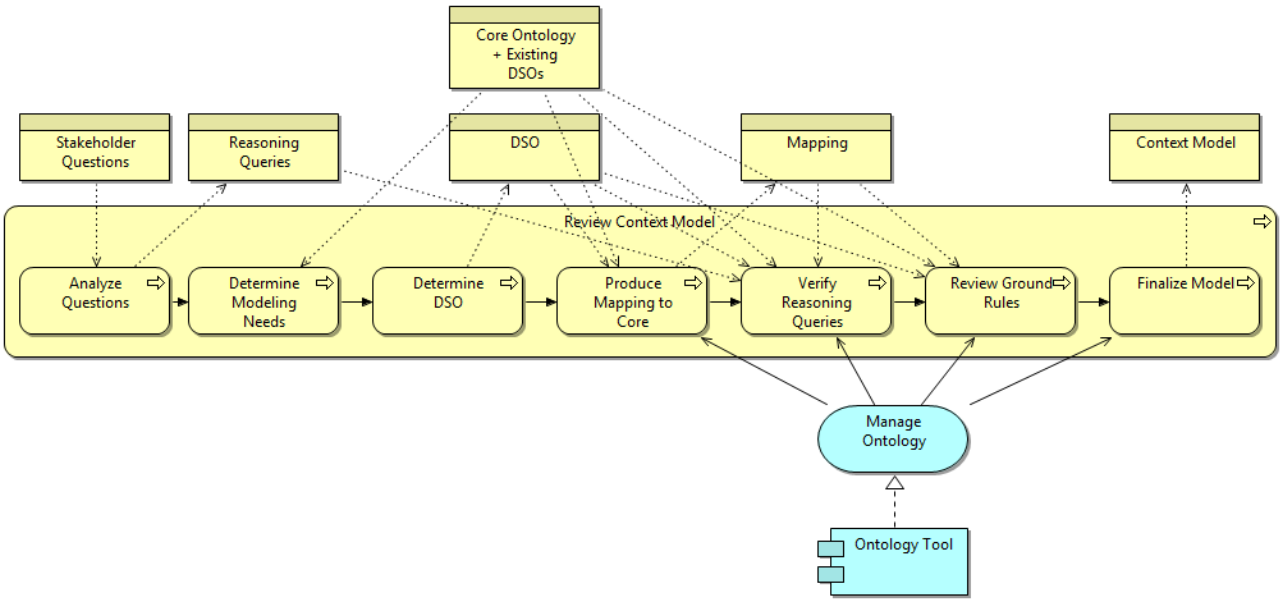

Figure 11: Review Context Model (Method Fragment #2)

Table 5: Review Context Model Fragment #2

Fragment	Extend and Add Reasoning to Context Model
Roles	DP Analyst
Pre-conditions	reasoning questions
Process Steps	Analyse Questions: In this step, the DP analyst performs the analysis of the questions elaborated by the business owner. The outcome of this step is the elaboration of reasoning queries that can be directly used with the reasoners.
	Determine Modelling Needs: In this step, the needs of the stakeholders are determined in terms of the core ontology and of the DSOs.
	Determine DSO: In this step, the necessary DSOs are determined with basis on the reasoning queries.

	Produce Mapping to Core: In this step, the mappings between the core ontology and the DSOs are produced.
	Verify Reasoning Queries: In this step, the reasoning queries resulting from the analyse questions step are verified and validated in terms of the core and domain specific ontologies.
	Review Ground Rules: In this step, the ground rules of the core and domain specific ontologies are also verified, or in other words, the model is validated in terms of the existence of exceptions.
	Finalize Model: In this step, the finalized context model is released.
Input	The stakeholder questions which are given as input to the analyse questions step; The Core ontology and existing DSOs, which are given as input to the determine modelling needs step, to the produce mapping to core step, to the verify reasoning queries step, and to the review ground rules step; the DSO which is given as input to the produce mapping to core step, to the verify reasoning queries step, and to the review ground rules step; the mapping which is given as input to the verify reasoning queries step, and to the review ground rules step.
Output	The reasoning queries which is an output of the analyse questions step; the context model, which is the output of the finalize model step.
Post-conditions	The context model is ready for being instantiated.
Tools	Ontology Tool, for manipulating the core and domain specific ontologies, testing the queries and ground rules.

5.4 Instantiate Model

This method fragment deals with the creation of a model instance for a specific scenario. Depending on the scenario and respective needs of the stakeholders, the instance might make use of the DIO or of the DIO + DSOs + Mappings. Table 6 describes the fragment.

Table 6: Instantiate Model Fragment

Fragment	Instantiate Model
Roles	DP Analyst
Pre-conditions	reasoning questions
Process Steps	n/a
Input	The Context Model (DIO + DSOs + mappings)
Output	The Context Model instance.
Post-conditions	The context model is ready for inference.
Tools	Tool for instantiating the core part of the context model. Tools for instantiating the DSOs of the context model. Automatic and semi-automatic transformation tools between models (e.g. xquery and xslt based)

5.5 Perform Inference

This method fragment deals with the posing of the reasoning queries to the context model instance, by using different reasoners and query engines according to the needs. Table 7 describes the fragment.

Table 7: Perform Inference Fragment

Fragment	Perform Inference
Roles	Business Owner
Pre-conditions	None
Process Steps	n/a
Input	The context model instances, reasoning questions
Output	The reasoning answers.

Post-conditions	None
Tools	Ontology Tool Reasoning Tools

6 Domain Independent Ontology

To ground our approach to context modelling and address the aforementioned principles, we decided to use the ArchiMate 2.0 language (The Open Group, 2012) as domain-independent language. ArchiMate is an international standard that covers the domain of enterprise architecture. Therefore, it can be considered a domain-independent language in the setting of enterprise architecture. The motivation to select the ArchiMate language as the core of the context model is that its design principles largely overlap with those of the TIMBUS context model. Namely, ArchiMate is a language that provides a high-level of abstraction, is concern-oriented and viewpoint-oriented and was designed with extensibility in mind. However, and as a domain-independent language, ArchiMate does not address domain-specific concerns that were identified as stakeholder requirements, such as licenses, patents, legal requirements, sensors, and so on. This is why the principle of extensibility proves to be important.

Therefore, the ArchiMate language meta-model was converted to an OWL representation so that inference can be applied to its models. The resulting core ontology is extended through a set of DSOs tailored to address explicit modelling concerns. Inference (reasoning) will be used, for example, to assess the consistency of models against rules, verify the completeness of models, or produce reports based on the contents of the model. In this section, we briefly describe the ArchiMate language and framework and the OWL language.

6.1 ArchiMate

The ArchiMate modelling language represents the culmination of years of work in the area of enterprise architecture modelling languages and frameworks. The language includes a minimum set of concepts and relationships and the framework includes a minimum set of layers and aspects to enable modelling of the majority of cases (The Open Group, 2012).

6.1.1 Framework and Meta-model

The framework organizes the modelling language in a three by three matrix: the rows capture the enterprise layers, i.e., business, application, and technology, and the columns capture cross layer aspects, i.e., active structure, behaviour and passive structure. Figure 12 depicts the framework.

The business layer is concerned with products and services offered to external customers, realized by the business processes of the organization, which are performed by business cases. The application layer is concerned with the application services, which support the business layer and are realized by software applications. The technology layer is concerned with the infrastructure services offered to applications, realized by hardware and system software. Regarding aspects, the active structure contains entities capable of performing behaviour; the behaviour, contains elements defined as units of activity performed by one or more active structure elements; and the passive structure contains objects on which behaviour is performed. Figure 13 depicts the different concepts and relationships of the language organized as follows: the concepts

belonging to the passive structure in green, the concepts belonging to the active structure in blue, and the behaviour concepts in yellow.

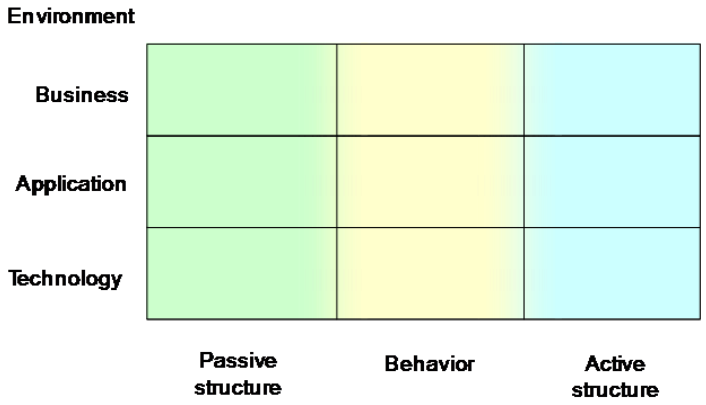

Figure 12: The ArchiMate Framework (The Open Group, 2012)

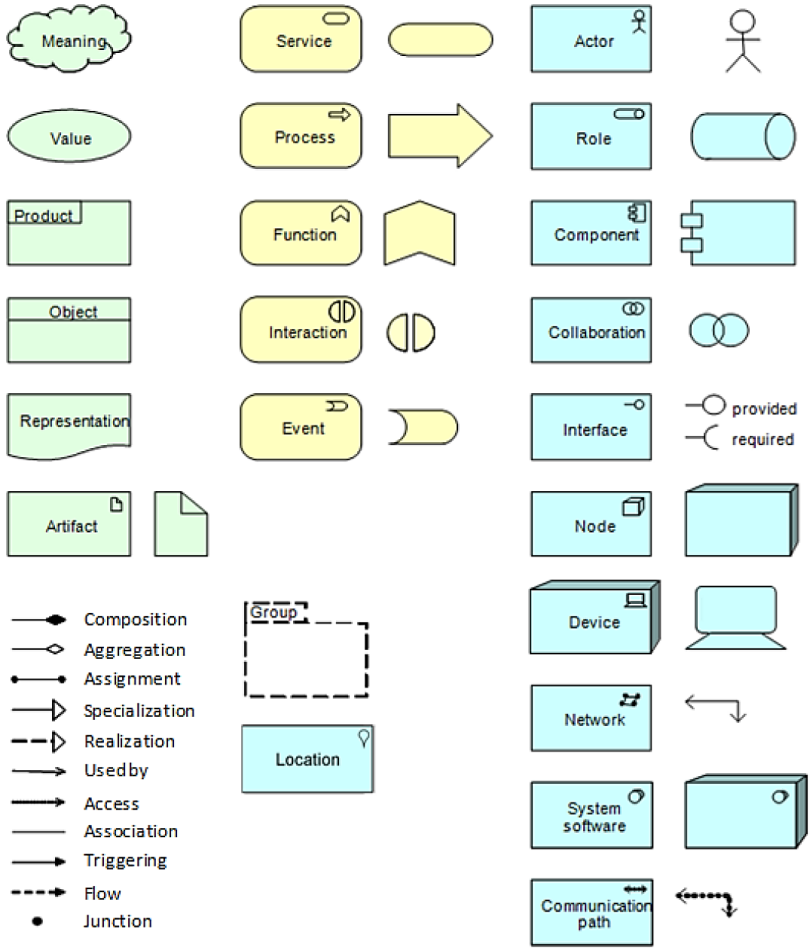

Figure 13: ArchiMate's Concepts and Relationships (The Open Group, 2012)

While it is already possible to envision the relationships existing between elements pertaining to different columns from the description just given, relationships between elements pertaining to different layers also exist. In fact, ArchiMate is able to model intra- and inter-layer dependencies. Inter-layer dependencies between two layers are usually fulfilled by the "Used By" relationship where the lower-level layer usually provides a service which is used by elements at the higher level layer. Other types of inter-layer dependencies can also occur, such as when an element at a higher layer is realized by an element at a lower layer, or when a lower layer element is assigned to a higher layer element (for instance, when a business process, function, or interaction is fully automated, an assign relation is used in conjunction with the respective application component; the same also happens between business service and application interface). The full meta-model of the language can be consulted in Annex B.

6.1.2 Viewpoints

Besides providing a framework and a modelling language, and in line with the recommended practice on architecture descriptions described in ISO 42010, ArchiMate also provides a set of viewpoints that can be used to accommodate different concerns. The viewpoints act as filters on the model and are used to specify different views upon the model, highlight different aspects that matter to different stakeholders. Some viewpoints display intra-layer concepts and dependencies, while others display cross layer concepts and relationships. Currently, the following standard viewpoints are part of ArchiMate: Introductory, Organization, Actor Co-operation, Business Function, Business Process, Business Process Co-operation, Product, Application Behaviour, Application Co-operation, Application Structure, Application Usage, Infrastructure, Infrastructure Usage, Implementation and Deployment, Information Structure, Service Realization, Layered, and Landscape Map. Table 8 lists the viewpoints and the layers/aspects that they cross.

Table 8: ArchiMate Viewpoints Description

Viewpoint	Layers	Aspects
Introductory	Business, Application, Technology	Active Structure, Behaviour, Passive Structure
Organization	Business	Active Structure
Actor Co-operation	Business, Application	Active Structure, Behaviour
Business Function	Business	Behaviour, Active Structure
Business Process	Business	Behaviour
Business Process Co-operation	Business, Application	Behaviour
Product	Business, Application	Behaviour, Passive Structure
Application Behaviour	Application	Passive Structure, Behaviour, Active Structure
Application Co-operation	Application	Behaviour, Active Structure
Application Structure	Application	Active Structure, Information
Application Usage	Business, Application	Behaviour, Active Structure
Infrastructure	Technology	Behaviour, Active Structure
Infrastructure Usage	Application, Technology	Behaviour, Active Structure
Implementation and Deployment	Application, Technology	Passive Structure, Behaviour, Active Structure
Information Structure	Business, Application, Technology	Passive Structure

Service Realisation	Business, Application	Behaviour, Active Structure, Passive Structure
Layered	Business, Application, Technology	Passive Structure, Behaviour, Active Structure
Landscape Map	Business, Application, Technology	Passive Structure, Behaviour, Active Structure

According to the ArchiMate specification, viewpoints are classified in two axes: purpose and content. Possible purposes are designing, deciding, and informing, whereas content can be divided in different levels of abstraction: overview, coherence, and detail. Designing viewpoints has the aim of supporting the design of the system from the initial conception to the detailed design, according to the concerns of architects, software developers, process designers, etc. In that sense, the views developed on the basis of designing viewpoints should consist of diagrams using formal modelling languages. Deciding viewpoints should assist in decision making, addressing the concerns of decision making stakeholders, such as CEOs. In that sense, views developed according to the viewpoint should only address factors that enable the discussion of important issues, offering less complexity than the views developed according to the designing viewpoint. For its turn, informing viewpoints should be used to inform any stakeholder about the enterprise architecture in order to promote awareness, commitment, etc., addressing the concerns of customers, employees, and other stakeholders. Views developed according to this viewpoint should rely on informal pictures or diagrams that should be easy to understand for these stakeholders. Figure 14 depicts the described classification.

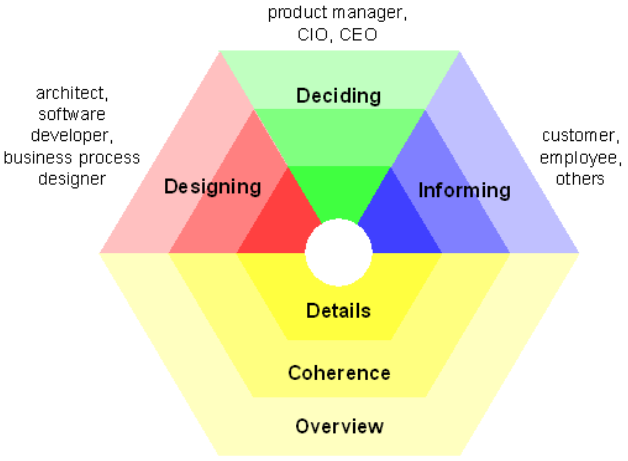

Figure 14: Viewpoint Classification (The Open Group, 2012)

6.1.3 Extensions

ArchiMate is also extensible at different levels: (*i*) at the level of the properties of concepts and relationships; (*ii*) at the level of specialization of concepts already existing in the meta-model; (*iii*) and at the level of the addition of new concepts to the meta-model and respective notation. However, the specification also claims that any extension should comply with the design restriction of keeping the language as small as possible.

Concerning (*iii*), two official extensions are present in the current specification: the Motivation Extension, and the Implementation and Migration Extension. The Motivation extension adds motivational or intentional

concepts, i.e., goals, principles, requirements and constraints, and the sources of such intentions or motivations, i.e., stakeholders, drivers and assessments. Along with the concepts, new viewpoints are also added: stakeholder viewpoint, goal realisation viewpoint, goal contribution viewpoint, principles viewpoint, requirements realisation viewpoint, and motivation viewpoint. Dependencies from the motivation extension to the core ArchiMate meta-model are materialized through the concepts of requirement and constraint, and through the realisation relationship. Figure 15 depicts the Motivation extension meta-model.

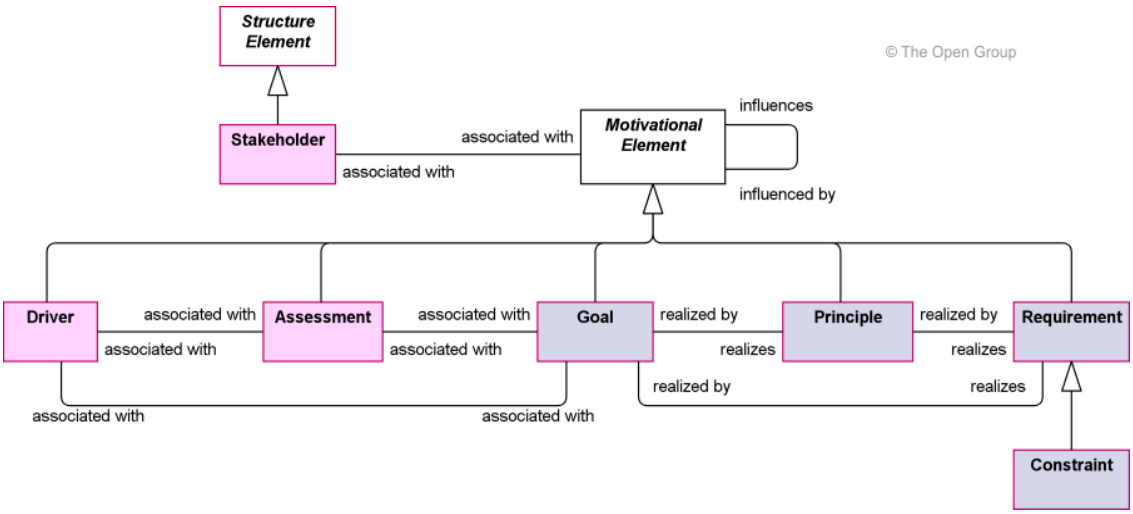

Figure 15: Motivation Extension Meta-model (The Open Group, 2012).

The Implementation and Migration extension includes concepts for modelling implementation programs and projects to support the program, portfolio, and project management, i.e., work package, deliverable. Concepts are also included for supporting the planning of migrations, i.e., gap, plateau. Three additional viewpoints are included: project viewpoint, migration viewpoint, and implementation and migration viewpoint. Dependencies to the core meta-model are enforced through the assignment of business roles to work packages and of locations to work packages and deliverables, and also through the association of gaps to core elements and the aggregation of core elements in plateaus. Figure 16 depicts the meta-model.

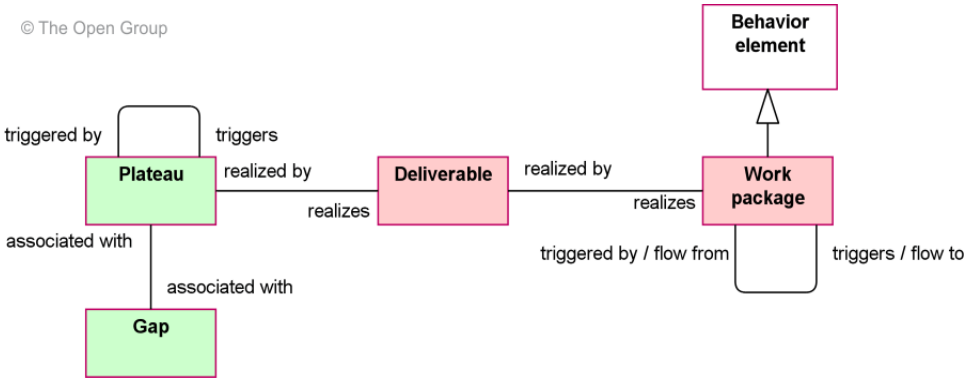

Figure 16: Implementation and Migration Extension Meta-model (The Open Group, 2012)

6.2 OWL

OWL (W3C, 2012) is the latest ontology language presented by the World Wide Web Consortium (W3C). It is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things.

6.2.1 Components of an OWL Ontology

An ontology consists of axioms that place constrains on classes and relationships permitted between them. These axioms allow the systems to infer additional information based on the data explicitly provided. The data described by an ontology specified using one of the languages of the OWL family is interpreted as a set of "individuals" and a set of "properties" which relate these individuals to each other. The components are now described in increasing detail:

- Classes: Are the basic building blocks of OWL ontology. Every individual in the OWL world is a member of the class `owl:Thing`. Thus each user-defined class is implicitly a subclass of `owl:Thing`. Domain specific root classes are defined by simply declaring a named class. OWL also defines the empty class, `owl:Nothing`. OWL supports six main ways of describing classes; the simplest of these is a Named Class. The other types are: Intersection classes, Union classes, Complement classes, Restrictions, and Enumerated classes.
- Individuals: In addition to classes, we want to be able to describe their members. We normally think of these as individuals in our universe of things. An individual is minimally introduced by declaring it to be a member of a class.
- Properties: Properties are used to state relations between individuals or between an individual and a data value. There are two main categories of properties, Object properties and Datatype properties, which can be described as follows:
    - Object properties, which link individuals to individuals.
    - Datatype properties, which link individuals to datatype values.

  Furthermore, in order to restrict the relation, properties can have a specified domain, which specifies which individuals from specific classes can make use of it, and range, which specifies that the values that the property can take. It is also possible to specify property characteristics, which provides a powerful mechanism for enhanced reasoning about a property. The following characteristics are possible:
    - Functional, which for a given individual, the property takes only one value. In other words, there cannot be two distinct values that are instances of such a property.
    - Inverse functional, which for a given property value there might be only a unique individual.
    - Symmetric: If a property links A to B then it can be inferred that it links B to A.
    - Transitive: If a property links A to B and B to C then it can be inferred that it links A to C.

Figure 11 depicts an example of the basic elements of OWL, of which person, country and pet are classes. Inside the classes are the individuals which are shown by the symbol ◊ and the arrows between them are the properties.

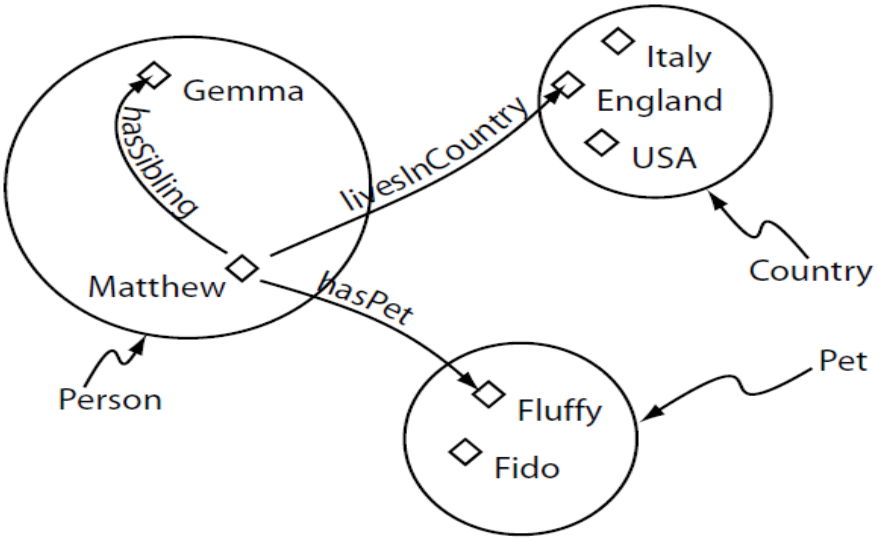

Figure 17: OWL Components Example (Horridge, 2011)

6.2.2 OWL Restrictions

Restrictions describe a class of individuals based on the type and possibly number of relationships that they participate in. Restrictions can be grouped into three main categories:

- Quantifier Restrictions (Existential ∃, Universal ∀)
- Cardinality Restrictions (Min ≥, Equal =, Max ≤)
- Has Value Restriction (∋)

The existential restriction means 'some values from', or at least one. An existential restriction describes the class of individuals that have at least one kind of relationship along a specified property to an individual that is a member of a specified class. The Universal restriction ∀ describes that a set of individuals, for a given property, only have relationships to other individuals of a specific class. Cardinality restrictions allow the specification of the number of relationships that a class of individuals participates in with other individuals or data types. Finally, "Has Value" restrictions allow us to specify the class of individuals that participate in a specified relationship with a specific individual (Horridge, 2011).

6.2.3 Reasoning in OWL

One of the key features offered by ontologies that are described using OWL-DL is that they can be processed by a reasoner. One of the main aspects offered by a reasoner is to test whether or not one class is a subclass of another class. By performing such tests on the classes in an ontology, it is possible for a reasoner to compute the inferred ontology class hierarchy. This is particularly useful when dealing with cases where

classes can have more than one parent. Another standard feature offered by reasoners is consistency checking. Based on conditions of a class, the reasoner can check whether or not it is possible for the class to have any instances. A class is deemed to be inconsistent if it cannot possibly have any instances (Horridge, 2011). The use of all the described features can aid in ensuring compliance with the ArchiMate standard.

6.3 OWL Representation of ArchiMate

As already mentioned, an OWL representation of the ArchiMate meta-model including the Motivation extension and the Implementation and Migration extension was created. ArchiMate itself is grounded in the entity-relation paradigm, providing specialization of these generic concepts into enterprise architecture concepts and also into domain-specific concepts, as shown in Figure 18.

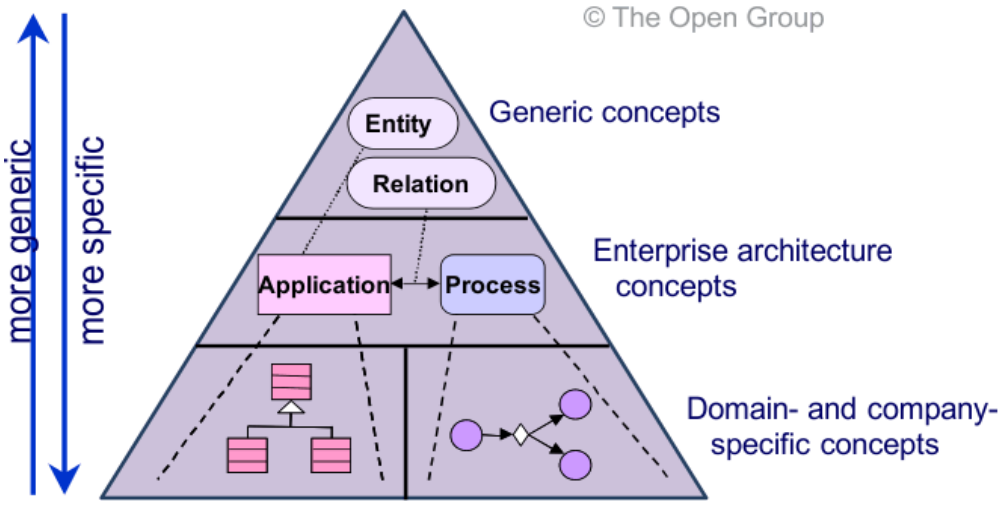

Figure 18: Meta-models at Different Levels of Specificity (The Open Group, 2012)

As such, the creation of an ontological representation of the ArchiMate meta-model would involve mapping between the concepts and relations of ArchiMate and the classes and properties of OWL. Such mapping involves analysing ArchiMate's meta-model concept by concept, including the relations with other concepts and the constraints existing in those relations. Concepts were mapped into OWL classes, relations were mapped into OWL ObjectProperties, and restrictions were added into those properties: InverseObjectProperties and SuperObjectProperties axioms were added to the OWL ontology, so that derived relationships can be extracted through the use of reasoners. Cardinalities were also added to reinforce the coherence of the ontology and its compliance to the ArchiMate meta-model. For instance, since each concept in the core ArchiMate meta-model is part of exactly one layer and one structure, such coherence was enforced through cardinality restrictions. Figure 19 shows the OWL representation of ArchiMate with the Business Function class highlighted on the left pane and respective properties, including restrictions on the right pane. The OWL representation of the ArchiMate meta-model can be found at:

`https://timbus.teco.edu/ontologies/DIO.owl`

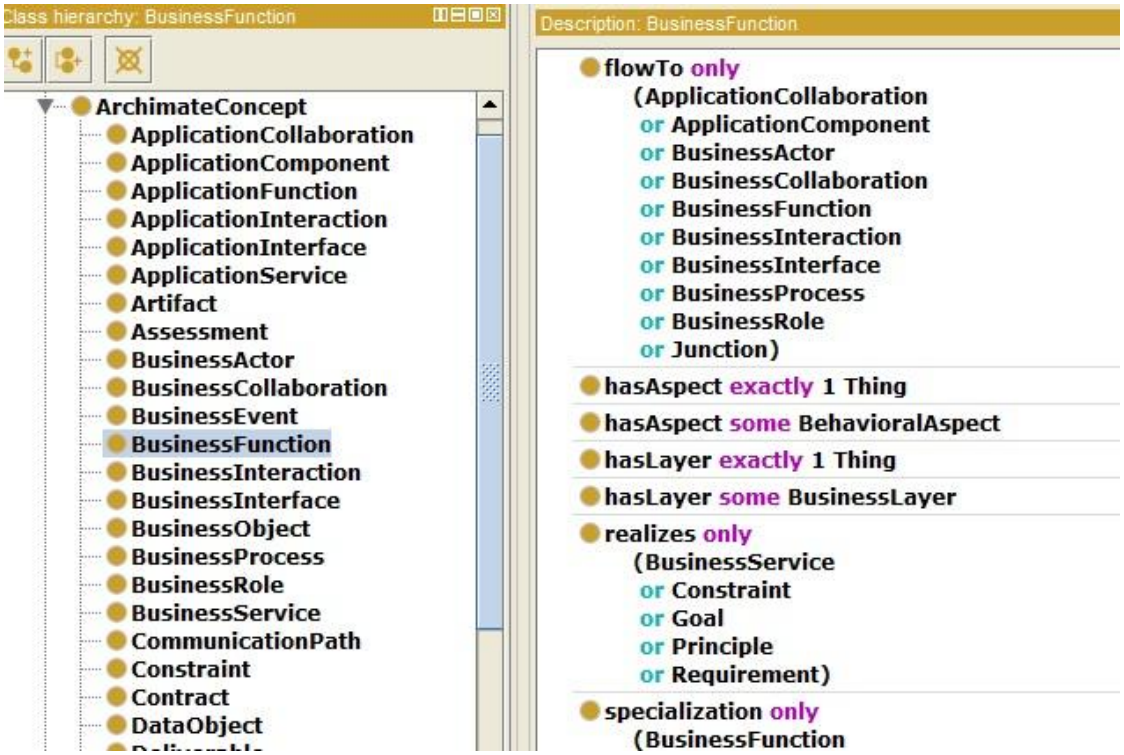

Figure 19: Business Function Class and Respective Properties

Figure 20, Figure 21, Figure 22, and Figure 23 showcase different aspects of ArchiMate's OWL representation, particularly, how the ontology representation still enforces layers and aspects.

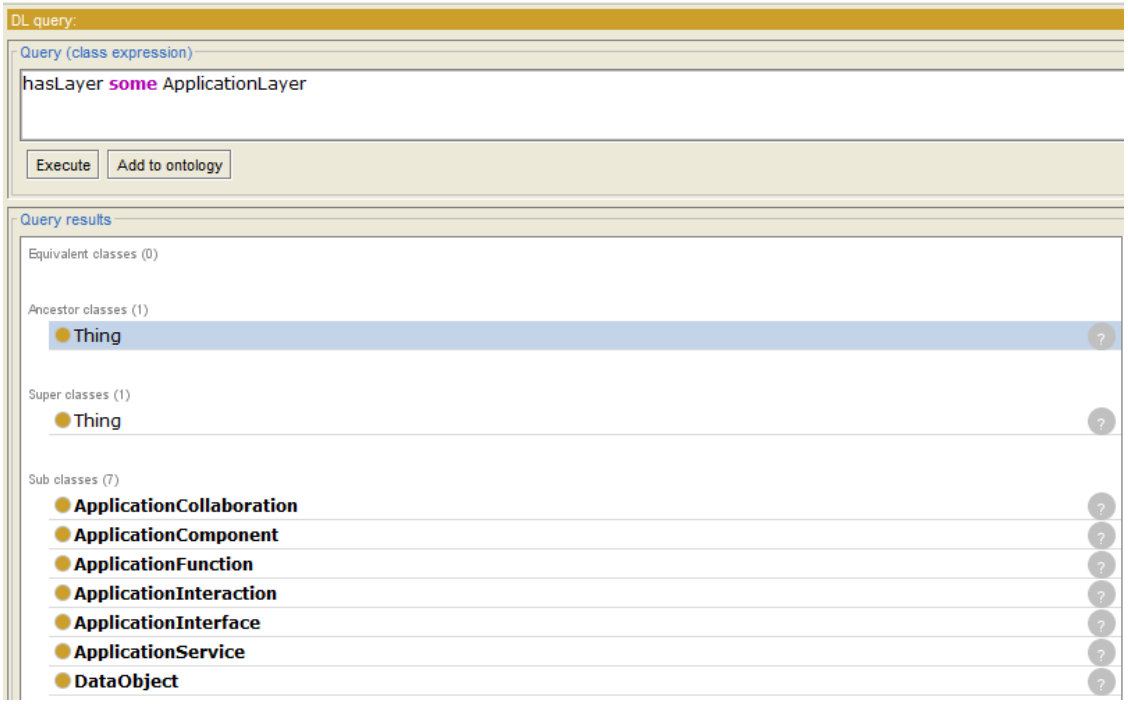

Figure 20: What ArchiMate concepts belong to the Application Layer?

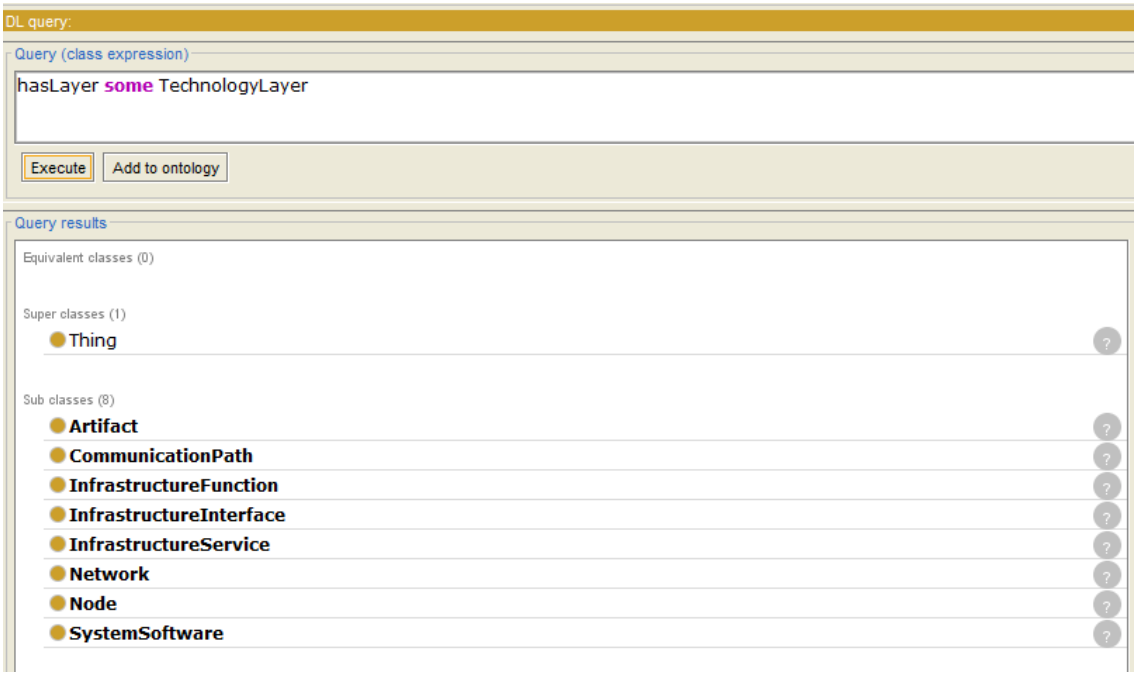

Figure 21: What ArchiMate concepts belong to the Technology Layer?

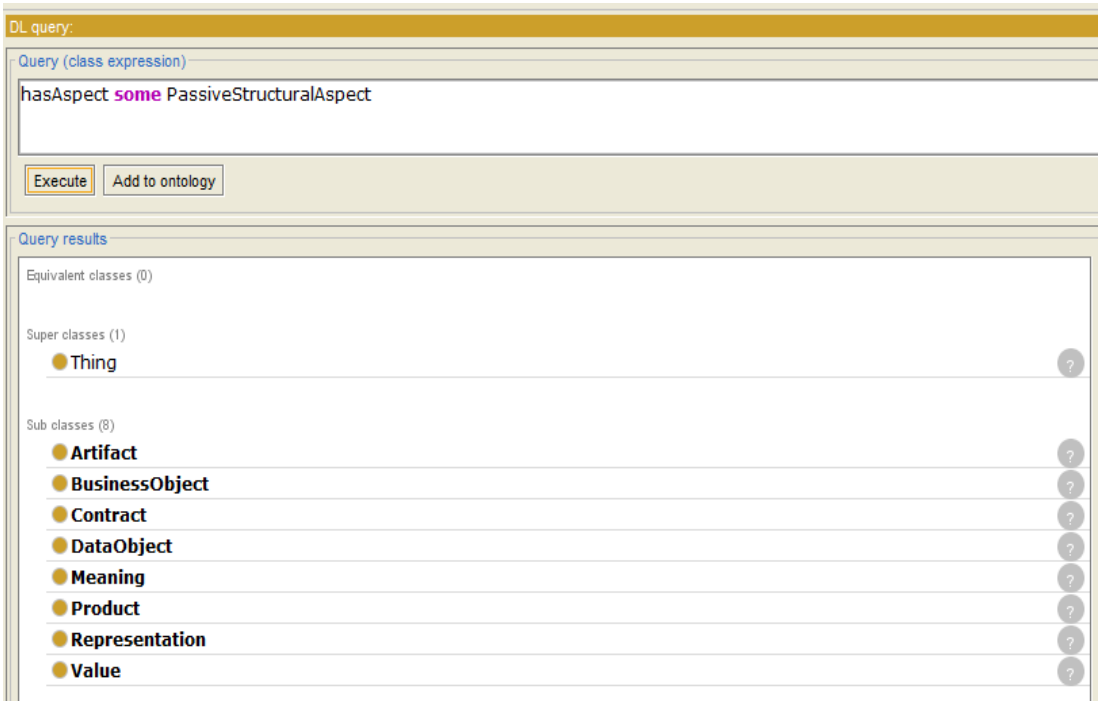

Figure 22: What ArchiMate concepts are Passive Structural Aspects?

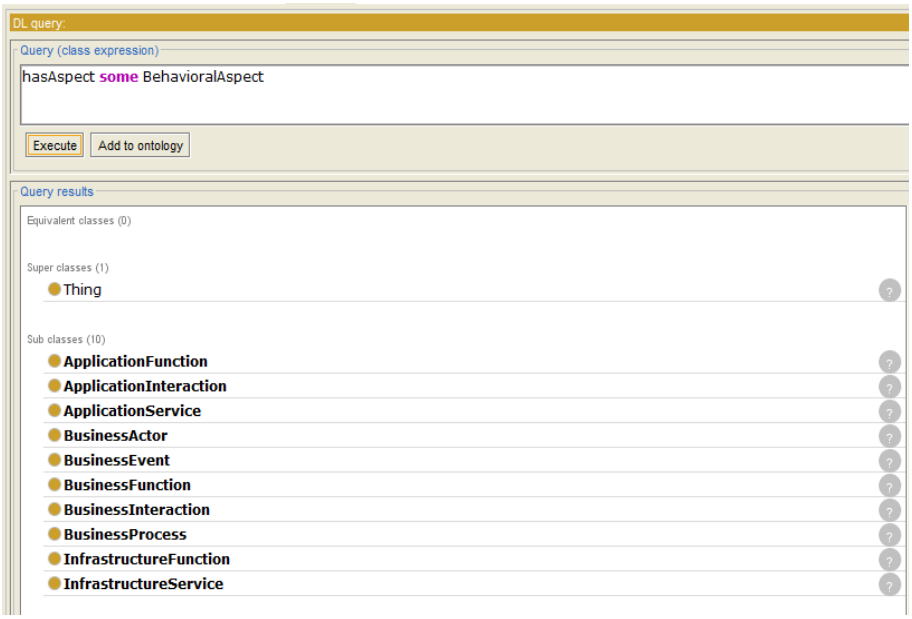

Figure 23: What ArchiMate concepts are Behavioural Aspects?

6.4 Reasoning on the DIO

This section depicts how derived relationships can be used for inferring dependencies. The ArchiMate model depicted in Figure 24 shows the relationships between two business actors Actor 1 and Actor 2, three business processes Process 1, Process 2, and Process 3 and the supporting application concepts, namely application services Service 1 and Service 2 that are realized by application components Component 1, Component 2, Component 3, and Component 4. The example is used to show how derived dependencies can be inferred.

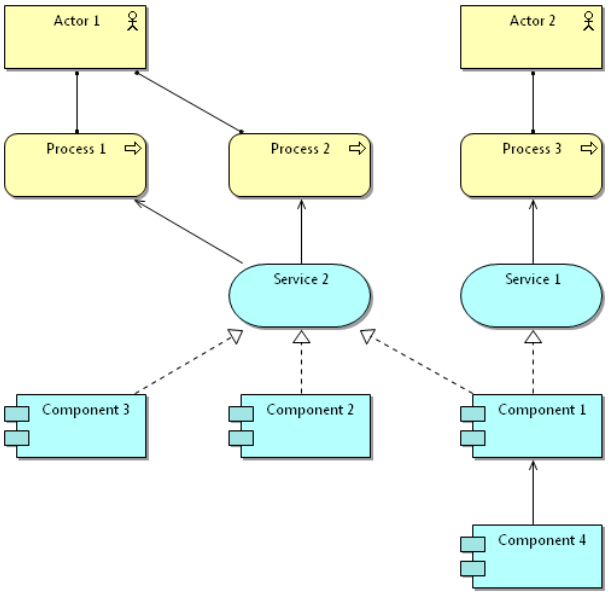

Figure 24: Example with uses and realizes relationships between business and application layer concepts

When converted to the core ontology, the ArchiMate model results in the individuals and relationships shown in Figure 25.

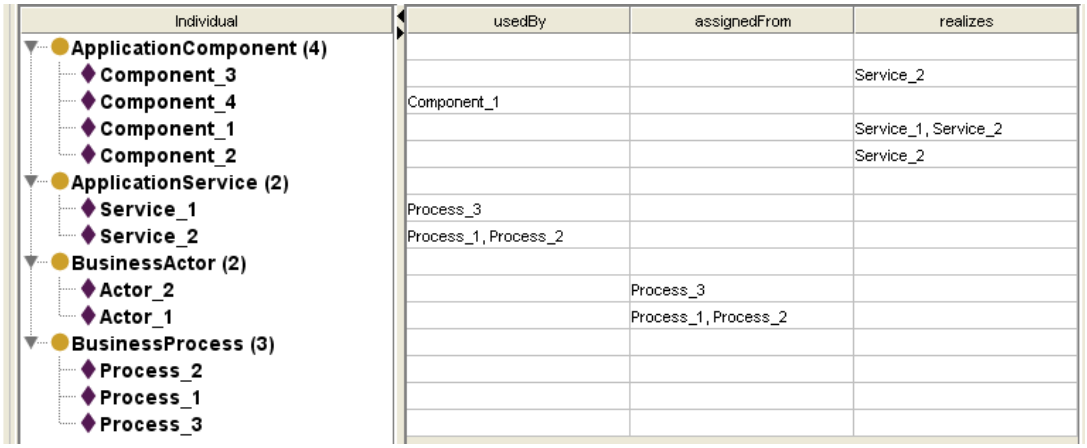

Figure 25: Individual specification of the above model in OWL

Let us consider the reasoning goal is identifying all **Application Components that Business Actor "Actor 2" depends on**. The inference process is as follows:

- Business Actor "Actor 2" is *assigned From* Business Process "Process 3"
- Application Service "Service 1" is *used by* Business Process "Process 3"
- Application Component "Component 1" realizes Application Service "Service 1"
- Application Component "Component 4" is *used by* Application Component "Component1"
- Business Actor ba1 *depends on* Application Component "Component 1", "Component 4"

Figure 26 shows the reasoner explanation that leads to reasoning that Application Components "Component 1" and "Component 4" support Business Actor "Actor 2".

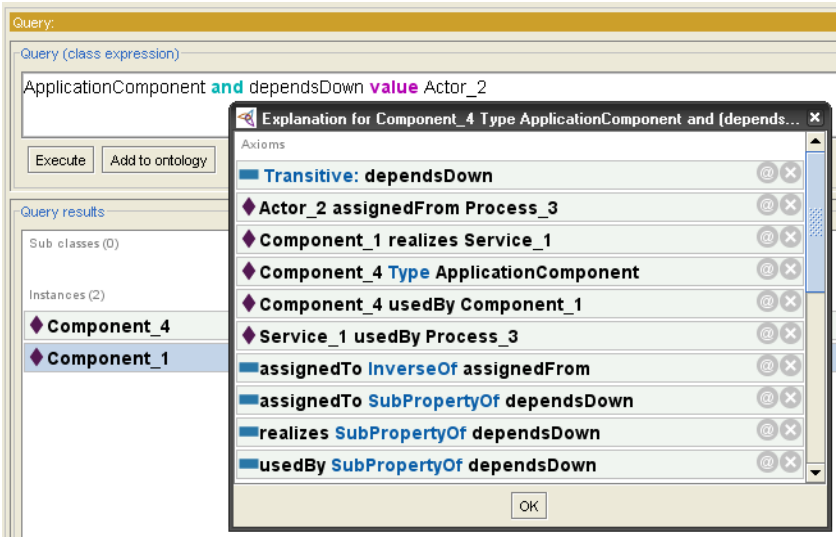

Figure 26: Reasoning question "Application Components that Business Actor 'Actor 2' depends on".

7 Domain Specific Ontologies

In this section, the first domain-specific ontologies identified as relevant for the use cases in the TIMBUS project are described. This list of DSOs is therefore not complete for all possible digital preservation scenarios, but the process described in this section is easily applicable to provide other domain specific ontologies that might be needed for capturing the context of other use cases.

The basis for the identification of the domain specific ontologies were the questions identified from the stakeholders, as detailed in Section 3, which can be consulted in Annex A. After an analysis step of grouping those questions that cannot be answered by the DIO itself due to that particular information being missing from the DIO, we arrived at the following grouping of potential DSOs

- Data: This grouping includes information on data used in the process (created, read or modified, by users or software), such as information on the input data of an application. This also includes metadata on the data, describing e.g. if the data contains personalised information. Further, information on the encoding can be provided here.
- Data Formats: This grouping includes information on which data format documents used in the process (created, read or modified, by users or software) adhere too. This type of information is the main concern of traditional digital preservation activities, and thus a tight interlinking to the many related projects is aimed for. Data Formats are tightly linked to the data group mentioned above.
- Legal: This grouping includes all legal requirements imposed on the processes and surrounding context. For example, this can be regulation on how long certain parts of a system need to be preserved.
- License: This grouping includes all aspects related to licenses, and concentrates initially on software licenses. Information captured in this DSO is on the types of licenses available, and the clauses they contain. These license clauses then pose restrictions on what can be performed with the software. Licenses are to a certain point a specialisation of legal requirements.
- Patents: This grouping contains aspects on patents, e.g. who is the owner of a specific patent, what the patent covers, or when it was granted. Patents also imply a restriction on how a software, hardware or method can be used. As with licenses, these are to a certain part a specialisation of legal requirements.
- Hardware: This grouping includes all aspects related to hardware, from desktop systems, computational and storage server infrastructure, to customised devices, such as handheld devices employed in the use case of work package 8.
- Sensors: This grouping is a specialisation of hardware, mainly dealing with sensors employed in the use case of work package 8. Sensors may differ in their appearance, from basic systems that need to be read via special instruments, to complex devices that have embedded software for processing.

- Software (applications): This grouping deals with software components and their relations to each other. These relations describe which other software components are required to run a certain software, or which software is conflicting. This group also includes information on configuration and the data required for a certain piece of software.

In our approach of extending the domain independent ontology, we aimed at reusing existing domain specific ontologies whenever available and fit. This allows for a greater interoperability of our context model with other standards, and decreases the overall risks in engineering an ontology for each domain from scratch. As some of the identified ontologies are very complex, we opted for an approach of mapping some specific elements, identified important to answer the questions, from these source ontologies to the domain-independent ontology.

7.1 Patent Ontology

7.1.1 Description

This domain-specific ontology shall describe information on patents (or more general, intellectual copyright) that are relevant to the preservation of business processes. The question is thus whether specific algorithms, software solutions, or hardware components are affected by patents. If this is the case, it could have implications on whether, or to what level of completeness, the preservation of the processes could be performed. Interesting aspects are thus information on which components of a process are affected, for which time periods the patents are granted.

7.1.2 Reasoning Questions

Some of the questions identified regarding patents are given in Table 9: Reasoning questions regarding patents

Table 9: Reasoning questions regarding patents

Question	Expected Output
Which patents are required for a certain component C?	List of patents.
What patents are used by application A during sequence discovery process?	The application A uses the patent “X”. The patent was granted on “1.1. 2010”.
How long is the patent p valid for?	The patent p is valid for X years. The date of expiry is “31.12. 2019”.

7.1.3 Ontology Structure

The most suitable candidate we identified for this domain is a result of the PATExpert project [1], funded by the European Union in the 6th Framework Programme. PATExpert defined a suite of ontologies that describe

[1] http://cordis.europa.eu/ist/kct/patexpert_synopsis.htm

patent documents, covering aspects such as the structure of documents and content they provide. It is mapped against the "Suggested Upper Merged Ontology"[2] (SUMO). An overview on the modules of the suite is given in Figure 27.

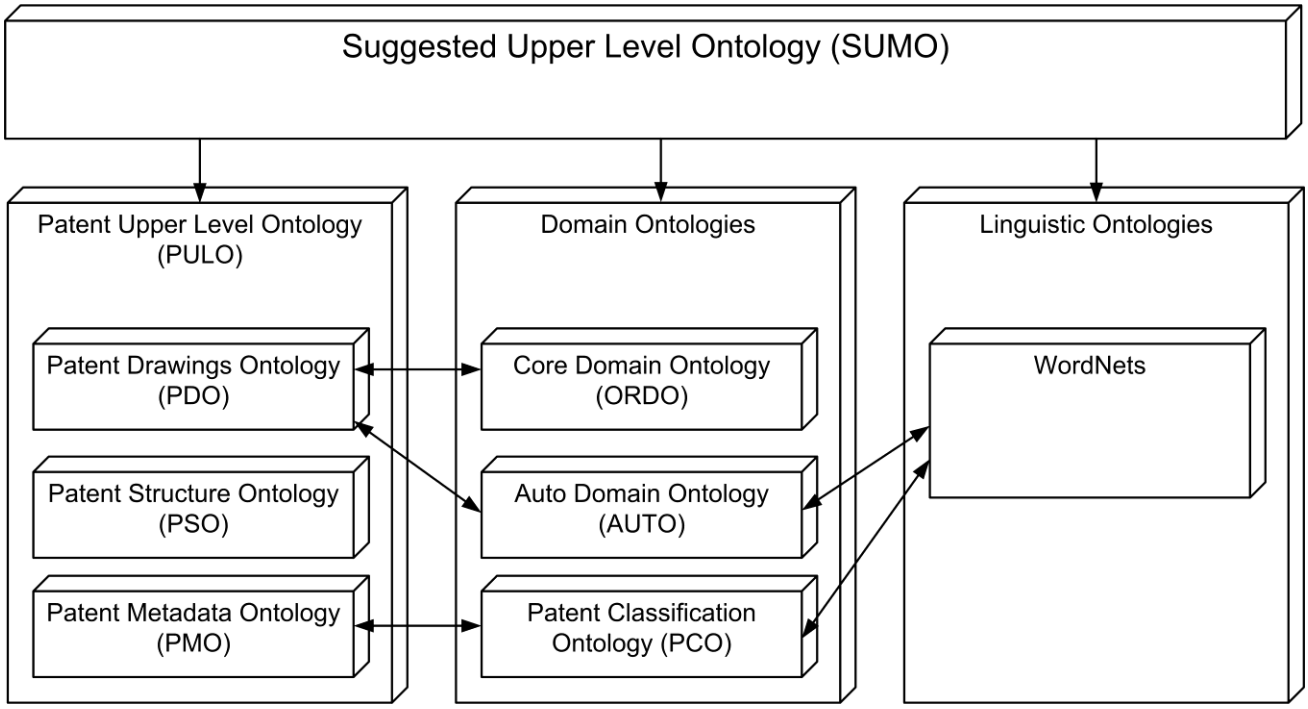

Figure 27: Structure of the PATExpert ontologies, and their integration with SUMO and other external ontologies.

For the purpose of answering the questions identified, we identified the Patent Metadata Ontology (PMO) to be relevant. An overview on the structure of the PMO is provided in Annex C, Figure 71.

An important aspect in the ontology is the pmo:PatentDocument, including the subclasses pmo:PatentPublication and pmo:GrantedPatent. These are classified into categories (pmo:classifiedAs relation to a pmo:PatentClassificationCategory), and described in detail by pmo:IntellectualPropertyDocument.

7.1.4 Ontology Mapping

For the initial version of the mapping, we opted for a simple approach that considers the pmo:GrantedPatentDocument class to be a specialised version of a Constraint in ArchiMate, which is indicated in the DIO by the property hasType:Patent. Using pmo:GrantedPatentDocument is sufficient for answering the above presented question, as the pmo:GrantedPatentDocument respectively its super-classes pmo: PatentDocument and the related pmo:IntellectualPropertyDocument contains information on the

[2] http://www.ontologyportal.org/

owner of the patent, and the publication of the patent. This mapping is shown in Figure 28, and can be found at the following address:

`https://timbus.teco.edu/ontologies/DSOs/PatentsMapping.owl`

For the subsequent D4.9, we will consider if this mapping is sufficient, or a more refined one is required.

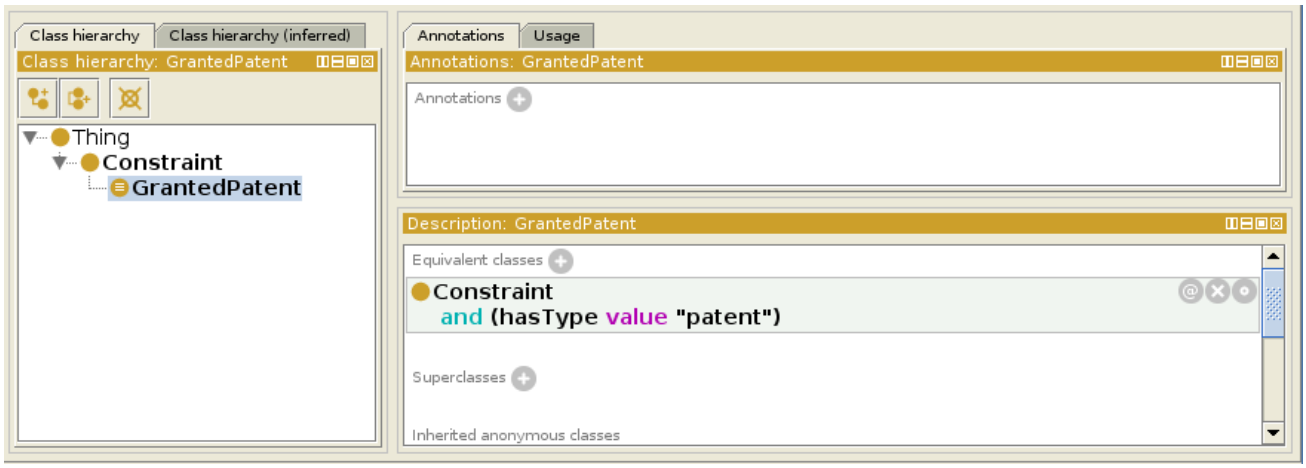

Figure 28: Mapping of the pmo:GrantedPatent to the domain-independent ontology

7.2 Software Licences

7.2.1 Description

In many cases, companies do not develop (all of) the software components they use to support their processes by themselves, but acquire them from third parties. This can be so-called commercial off-the-shelf software, or customised software.

Software licenses concern the rights and obligations a party has regarding these acquired software applications and components. The license in this case is a specific kind of contract that grants certain rights to the license taker regarding the usage of the software, e.g. as a component his own applications use. It defines for example whether the customer can get access to the source code, modify it, redistribute the software, etc.

7.2.2 Reasoning Questions

Some of the questions identified regarding patents are given in Table 10.

Table 10: Reasoning questions regarding licenses

Question	Expected Output
Which legal requirement R is a license?	List of legal licenses
Which licenses L are open-source?	List of licenses
What are the licenses L required to execute software application SA?	List of licenses
What restrictions on preservation actions are allowed according to license L?	List of actions applicable to the source code

7.2.3 Ontology Structure

A suitable candidate for this domain-specific ontology was identified in a subsection of "The Software Ontology[3] (SWO). The SWO is an ontology for describing software tools, their types, tasks, versions, provenance and associated data. SWO was originated in a project between the European Bioinformatics Institute and the University of Manchester, and has thus a focus on this domain, with many of its classes tailored to it. The ontology is structured in many different components, concerning e.g., versions, organisations, algorithms, or interfaces. One of these components is dedicated to licenses, and suits the needs of the reasoning questions outlined above. An overview on this ontology is given in Annex D, Figure 72.

The ontology models two important concepts – Software licenses, and License clauses. License clauses define properties and restrictions on what can be done with the software, e.g. whether redistribution is allowed, and in what form (with or without notice), or whether there is a restriction on the number of users that can use the software. Software licenses are then a composition of these clauses. Some abstract classes exist, e.g. the abstract class "Open source licenses" defines that the source code is available. Specific licenses are subclasses of a software license. The ontology pre-defines a set of these, but is not complete on commonly used free open source software licenses.

7.2.4 Ontology Mapping

The ontology mapping is relative straightforward, and allows both a *Software license* and a *License clause* to be specified as a subclass of a constraint. This way, we can profit from the pre-defined standard licenses in case we use such a license, but can easily define our own license as a composition of clauses, without having to modify the domain-specific ontology. This mapping is illustrated in Figure 29, and can be found at the following address:

`https://timbus.teco.edu/ontologies/DSOs/LicensesMapping.owl`

[3] http://theswo.sourceforge.net/

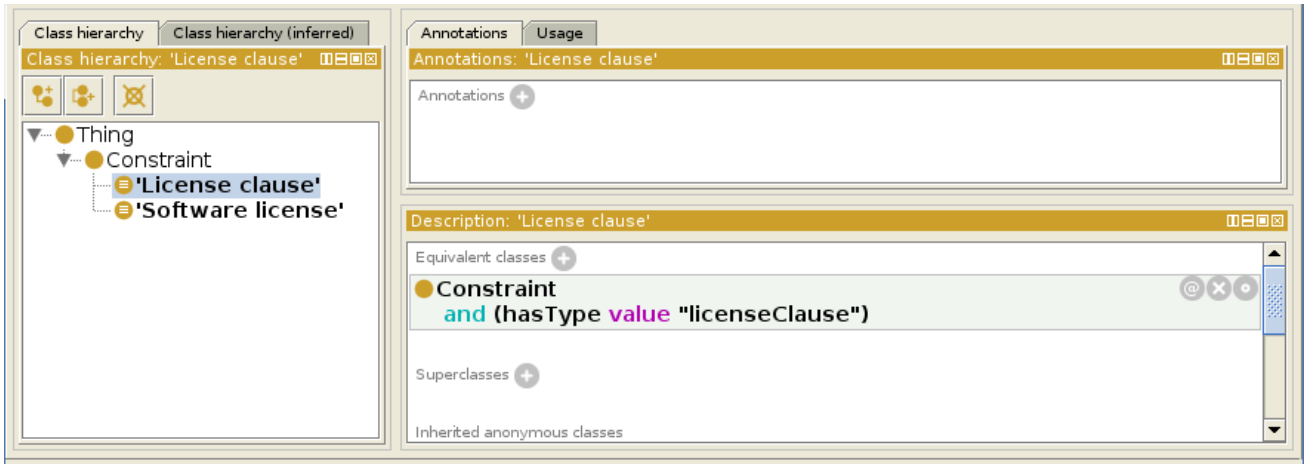

Figure 29: Mapping of the Software Ontology License clause and Software license classes to the DIO

7.3 Sensors Ontology

7.3.1 Description

Sensors are a very important element in civil engineering structural monitoring and safety. Sensors measure values that can be then processed and analysed, so that the structural behaviour is predicted, and safety measures are taken, if needed. Different types of sensors exist for measuring different types of engineering quantities, which will be derived according to a determined algorithm and respective calibration constants. Sensors are installed in determined locations on the structure and in absolute terms, and have a determined acquisition rate.

7.3.2 Reasoning Questions

Some of the questions identified regarding sensors are given in Table 11.

Table 11: Reasoning questions regarding sensors

Question	Expected Output
Which sensor types can measure the physical quantity Y?	List of sensor types
What is the observation plan for dam X?	List of sensors organized by sensor type
What are the measurement units for sensor X?	List of measurement units
What is the acquisition frequency for sensor X?	Acquisition frequency for sensor X

7.3.3 Ontology Structure

Different alternatives available for modelling sensors were analysed, such as SensorML[4] and TransducerML.[5] However, due to the complexity demonstrated in both specifications, it was thus decided as a first step to develop a simple sensor ontology that accounts for the reasoning questions dealing with sensors. Later, a migration from this sensor ontology to an available alternative can be performed, if desired. An overview on this ontology is given in Annex E, Figure 73.

7.3.4 Ontology Mapping

For the mapping between this DSO and the DIO we are considering that the sensors:Sensor is an equivalent class to a specialized version of a Node in the DIO, indicated by the datatype property hasType:sensor. The class sensors:GeoLocation and sensors:StructuralLocation are equivalent to the specialized Location element in the DIO with a datatype property hasType:sensor_location. Finally, it is also considered that the class sensors:Value is equivalent to the class Artifact in the DIO with the property hasType:sensor_value. This mapping is illustrated in Figure 30 and can be found at the following address:

`https://timbus.teco.edu/ontologies/DSOs/sensorsMapping.owl`

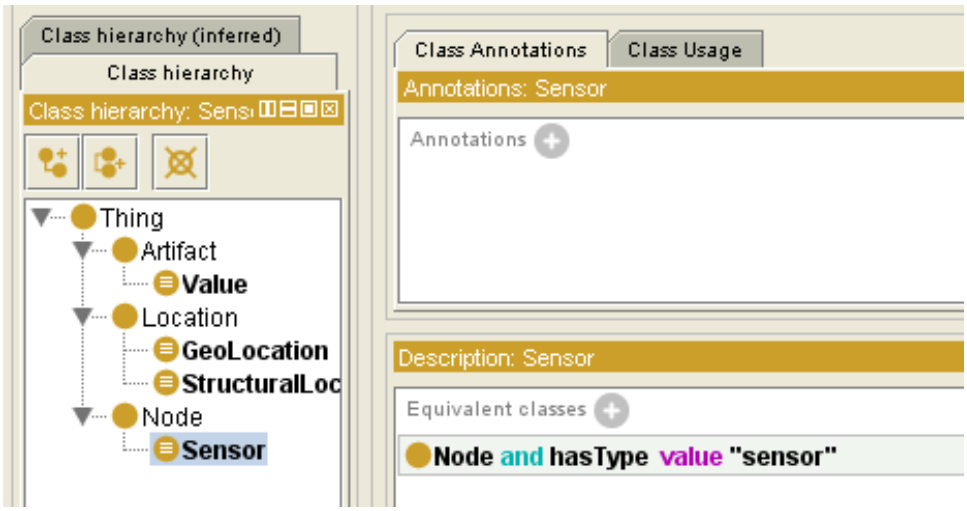

Figure 30: Mapping of the Sensor, StructuralLocation, GeoLocation, and Artifact classes to the DIO.

[4] http://www.ogcnetwork.net/SensorML

[5] http://www.ogcnetwork.net/infomodels/tml

8 Case Studies

This section will exemplify the application of the Context Model to different use cases: the WP8 and WP9 industrial cases and the Music Classification Process presented during Y1 of the project. For each case there will be a brief description of the scenario, a description of the instantiation of the DIO, and a description of the instantiations of the DSOs. For each case, a predefined set of reasoning queries will be posed to each DIO, as shown in Table 12, although considering concrete individuals of each case.

Table 12: DIO Questions and Queries

Question	Formalized Query
Which Service Level Agreements (SLAs) are associated with external systems?	Contract and (aggregatedBy some Product)
What business actors are assigned to business process BP?	BusinessActor and (assignedFrom value BP)
What business objects are being used by business process BP?	BusinessObject and (accessedBy value BP)
What application components support business process BP?	ApplicationComponent and (dependsDown value BP)
What are the technological entities supporting business process BP?	Thing and hasLayer some TechnologyLayer and hasAspect some BehavioralAspect or hasAspect some ActiveStructuralAspect and (dependsDown value BP)
What are the application dependencies of application component C?	Thing and hasLayer some ApplicationLayer and hasAspect some BehavioralAspect or hasAspect some ActiveStructuralAspect and (dependsDown value C)
What is the input data to Component C	DataObject and (hasAccessTypeReadBy value C)
What is the output data of Component C	DataObject and (hasAccessTypeWriteBy value C)

For the DSOs, a set of questions will be demonstrated specifically for each specific case making use of it.

8.1 Music Classification Process

The music classification process was presented during Y1 and was demonstrated using the first version of the context model. In order to show that the new version of the context model goes beyond the previous version of the context model, it was decided to present the application of the model to this case.

The case itself has some particularities that can be used to show the potential of the approach taken to the context model. It depicts the process dynamics and the dependencies between each step of the process and the applications and technology supporting it. It captures the service agreements which are associated with services and product offerings. It also captures software licences, which are mainly associated with elements at the level of the technology layer, and patents, which in this case is associated with the mp3 format and is also depicted at the level of the technology layer. These particularities will involve the usage of different DSOs integrated with the DIO: the Patent DSO and the Software Licenses DSO.

8.1.1 Brief Description of the Scenario

The music classification process deals with conducting an experiment involving the verification and validation of the usefulness of a method for automatically classifying items in a music collection into a set of predefined categories corresponding to music genres. It is performed by a researcher which aims to collect performance metrics for classification and make comparisons to the state of the art. The motivation for performing the preservation of such a process is related to any possible challenges to the results that can be made by members of the research community. Thus, by preserving such process, the provenance and authenticity of the results can be proven.

The process consists of the following steps: (*i*) Get Music Data, which uses an external service for acquiring training and test data; (ii) Get Groundtruth, which uses an external service for getting meta-data about genre classification for the input files; (*iii*) Extract Features, which uses an external service for getting numerical features from the input files; (*iv*) Combine Groundtruth with Features, which combines the features with the genre assignments; (*v*) Classify, which uses machine learning to train a model and assign new meta-data (genre labels) to unknown data; and (*vi*) Present Results, in which the results are obtained and presented, typically under the form of a publication. The process uses the following application components:

- WEKA machine learning toolkit, version 3.6.6; employed for the learning of a predictive model and assigning of labels to unknown data.
- Java SOMToolbox, version 0.7.5.1; used for format conversions.
- Taverna Workflow Engine, version 2.3.0; used to execute beanshell scripts and to provide the process workflow.
- Java Development Kit / Java Runtime Environment version 6.0; use as runtime environment for the Taverna Workflow Engineering.
- Ubuntu Linux version 11.04; used as platform to run the JDK / JRE.
- AudioFeatureExtraction REST Service which provides the extraction of numerical features from MP3
    - CGI parameters:
        - voucher={authentication key}
        - music={mp3 file Base64 encoded}
    - Return value: Vector in SOMLib format.
- MP3 Data provider Service; provides the audio files.
- Genre assignment (ground truth) provider; provides the assignment of the audio files to a specific genre.

8.1.2 The DIO Instance

The music classification process was completely modelled in ArchiMate, using the Archi tool, and then converted into OWL using the conversion tools described in Section 10.3. Annex F, Figure 74 depicts the ArchiMate model from which the DIO was derived. Figure 31 depicts the DIO instance with the individuals and properties, and Figure 32 to Figure 39 depict the results of the general queries made to the DIO. The DIO with the individuals for this case can be found at:

`http://timbus.teco.edu/ontologies/Scenarios/MusicClassification.owl`

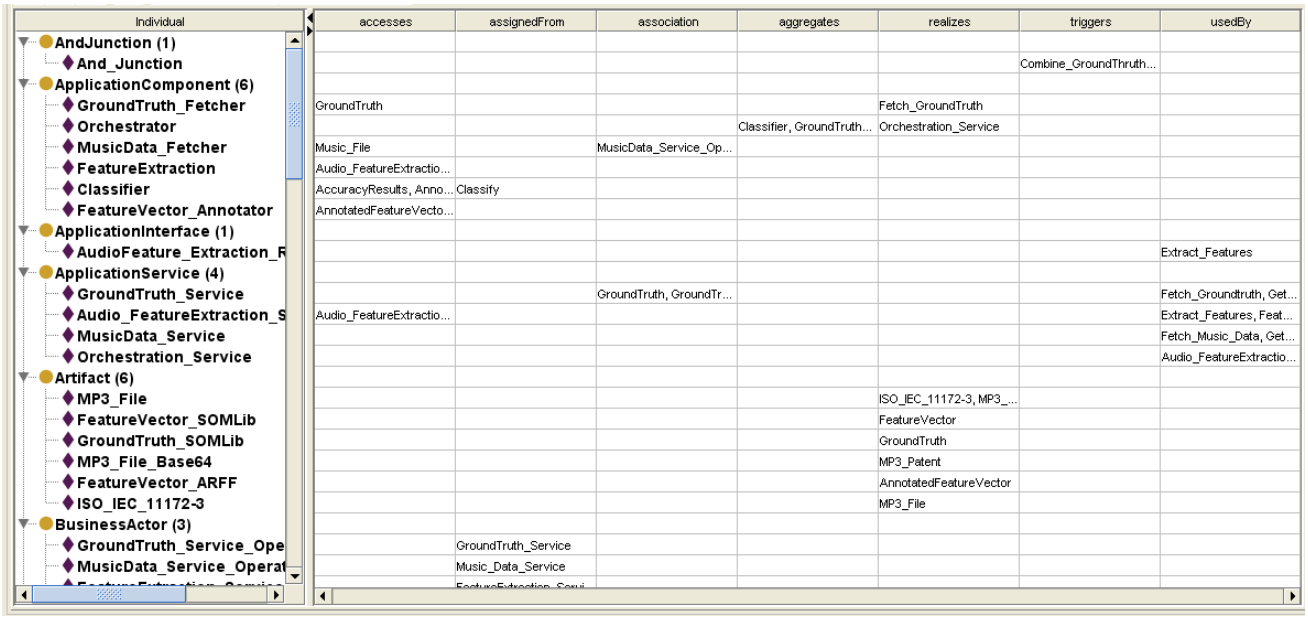

Figure 31: Music Classification Process DIO Instantiation

- Which Service Level Agreements (SLAs) are associated with external systems?

Figure 32: "Which Service Level Agreements (SLAs) are associated with external systems?" Query Results

- What business actors are assigned to business process *Experiment*?

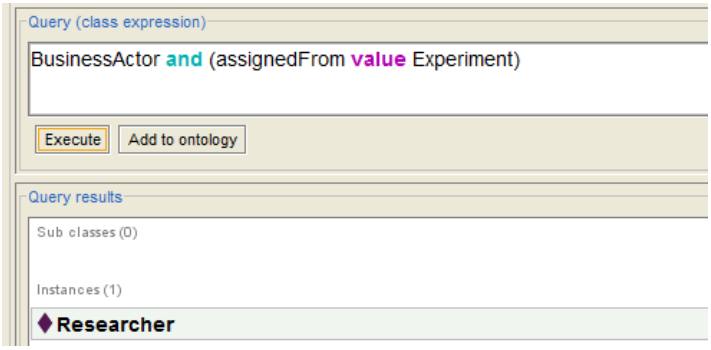

Figure 33: "What business actors are assigned to business process *Experiment*?" Query Results

- What business objects are being used by business process *Classify*?

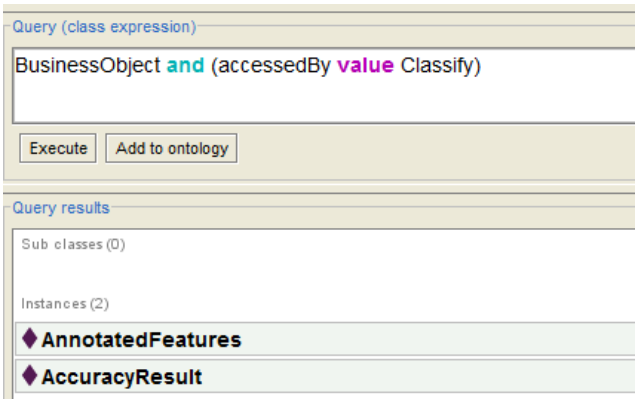

Figure 34: "What business objects are being used by business process *Classify*?" Query Results

- What application components support business process *Experiment*?

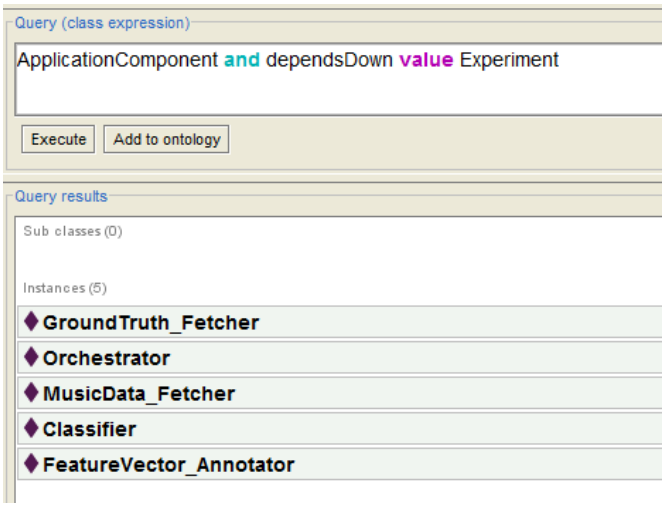

Figure 35: "What application components support business process *Experiment*?" Query Results

- What are the technological entities supporting business process *Experiment*?

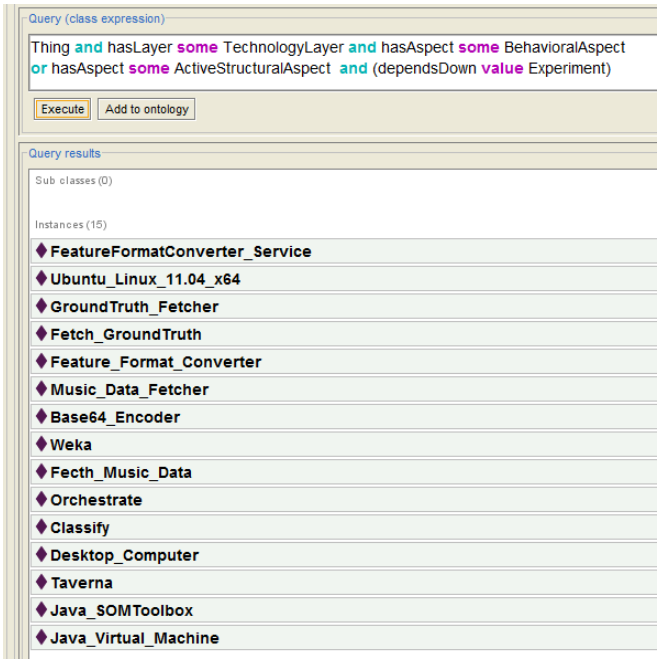

Figure 36: "What are the technological entities supporting business process *Experiment*?" Query Results

- What are the application dependencies of application component *Orchestrator*?

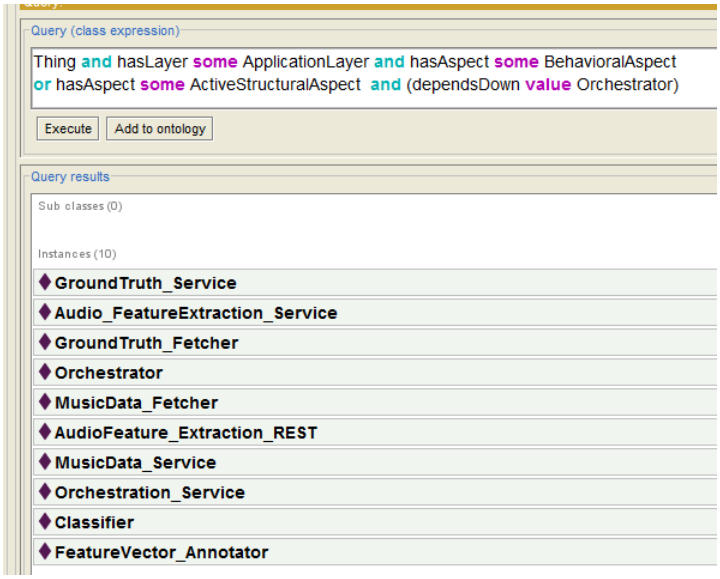

Figure 37: "What are the application dependencies of application component *Orchestrator*?" Query Results

- What is the input data to Component *FeatureVector Annotator*?

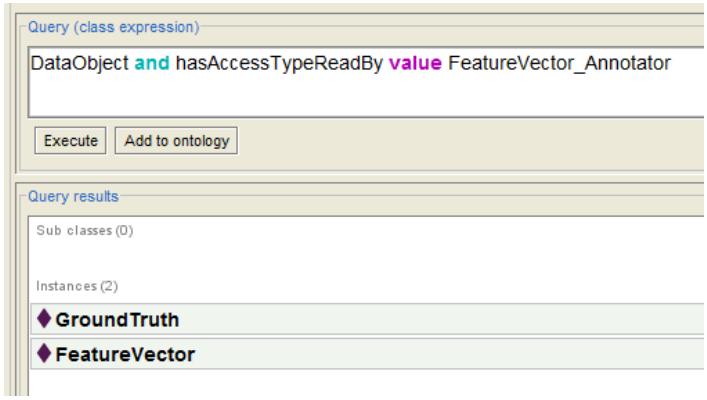

Figure 38: "What is the input data to Component *FeatureVector Annotator*?" Query Results

- What is the output data to Component *FeatureVector Annotator*?

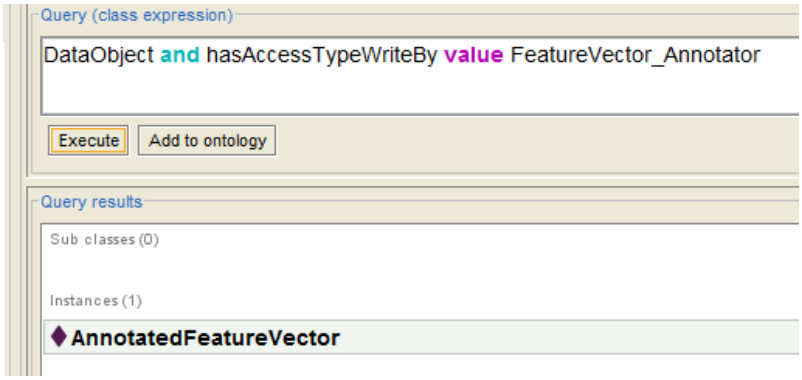

Figure 39: "What is the output data to Component *FeatureVector Annotator*?" Query Results

8.1.3 The DSO Instances

The music classification process has some particularities that are not possible to model in the DIO with the desired level of detail:

- Software licenses, with four occurrences (i.e., Apache Licence 2.0, Oracle Binary Code Licence, GNU General Public Licence GPL 2.0, and GNU Lesser General Public License LGPL 2.0)
- Patents, with one occurrence (i.e., MP3 Patent)

Those elements can be captured in an increased level of detail through the use of the Software Licences DSO and the Patent DSO.

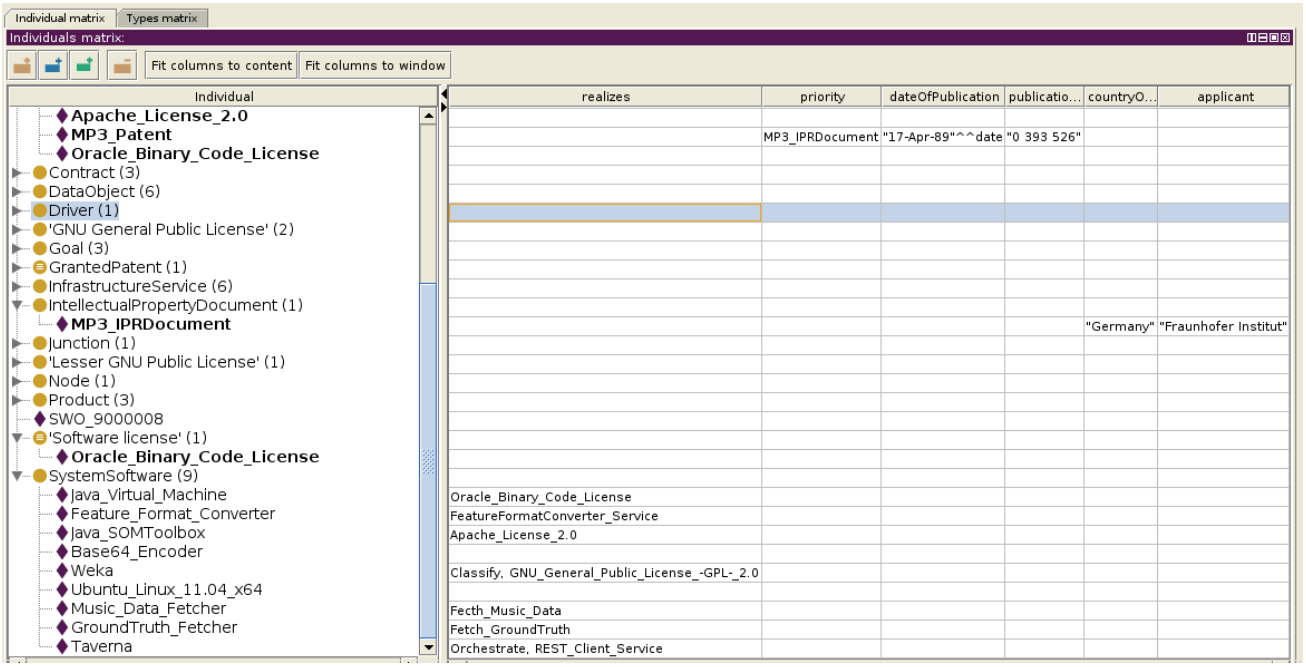

Figure 40: Licenses and Patents DSO instantiation

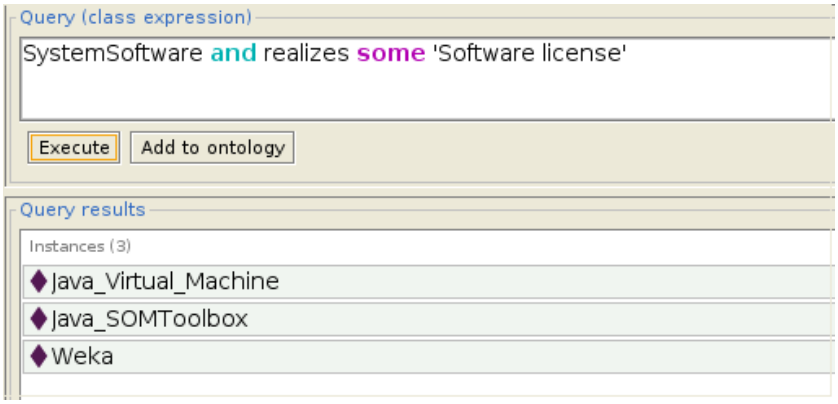

Figure 41: "What are the licenses L required to execute software application SA?" Query results

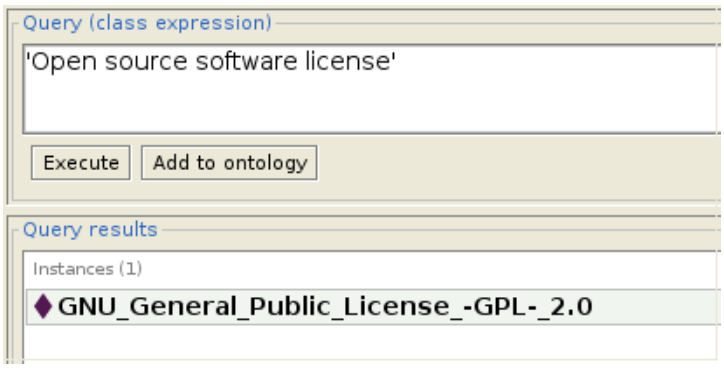

Figure 42: Which licenses L are open-source?" Query results

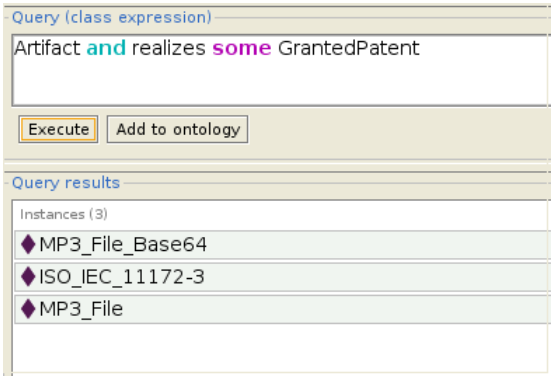

Figure 43: "Which patents are required for a certain component C?" Query results

8.2 WP8 Industrial Case

The WP8 Industrial Case description was not available during the Y1 iteration of the context model. As the description became available in Y2, it greatly influenced the new iteration of the context model, especially because some of the aspects that should be captured by the context model are specific to the domain of civil engineering and, in particular, to the domain of structural safety monitoring. This section provides a brief description of the scenario to which the context model was applied. The DIO instance and general query results are also described, along with a description of the DSO created especially for this case, including some reasoning queries.

8.2.1 Brief Description of the Scenario

The scenario explored in the context of the WP8 industrial case deals with sensor data acquisition. LNEC is mandated by law to monitor the structural behaviour of large civil engineering structures, in particular dams, in order to prevent accidents and ensure structural safety. The process to which the context model was applied deals with the manual or automatic acquisition of sensor data, which is required for analyzing the behaviour of a dam and its structural safety, from different sensors installed along the structure.

Once the data is acquired by the sensors it is uploaded to the gestBarragens information systems, directly from the sensors, using web services, or through portable devices. Once that data enters the system, it needs to be validated and then transformed from raw to engineering quantities that can be analysed by the civil engineers. After being transformed, the data is again validated and archived. More details on this process and on the motivations surrounding it can be consulted in D8.1.

8.2.2 The DIO Instance

An instance of the DIO was derived for the WP8 use case using Archi for producing an ArchiMate model, which was then converted into OWL using the conversion tools described in section 10.3. Annex G contains the ArchiMate model from which the DIO was derived. Figure 44 depicts the DIO instance with the individuals and properties. Figure 45 to Figure 52 depict the results of the general queries made to the DIO. The resulting OWL representation can be found at:

http://timbus.teco.edu/ontologies/Scenarios/wp8.owl

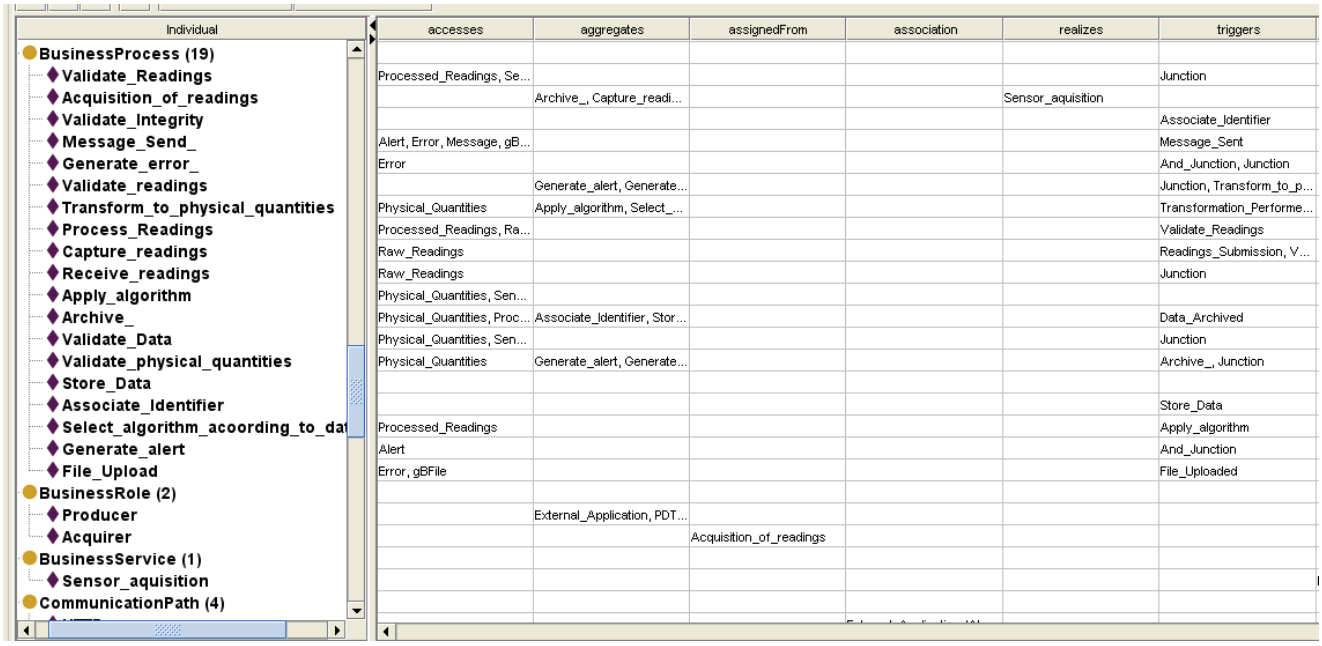

Figure 44: WP8 DIO Instantiation

- Which Service Level Agreements (SLAs) are associated with external systems?

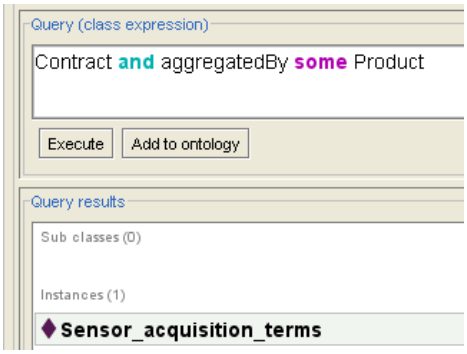

Figure 45: "Which Service Level Agreements (SLAs) are associated with external systems?" Query Results

- What business actors are assigned to business process *Acquisition of readings*?

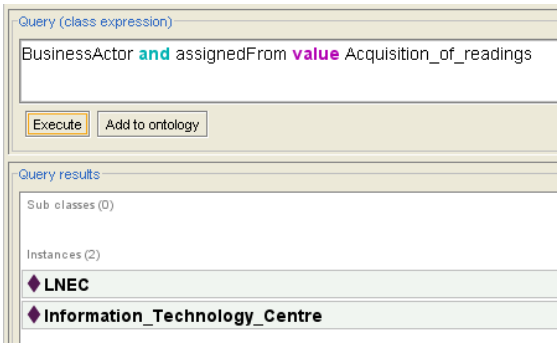

Figure 46: "What business actors are assigned to business process *Acquisition of readings*?" Query Results

- What business objects are being used by business process *Validate Readings*?

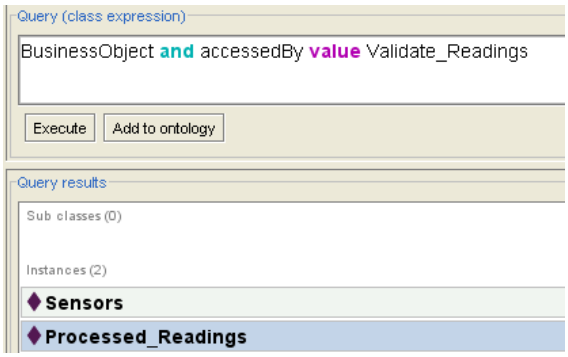

Figure 47: "What business objects are being used by business process *Validate Readings*?" Query Results

- What application components support business process *Acquisition of Readings*?

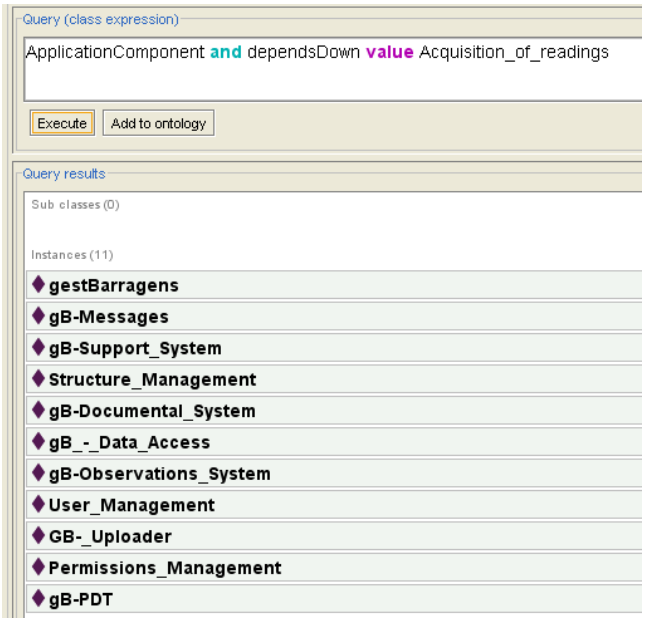

Figure 48: "What application components support business process *Acquisition of Readings*?" Query Results

- What are the technological entities supporting business process *Acquisition of Readings*?

Figure 49: "What are the technological entities supporting business process *Acquisition of readings*?" Query Results

- What are the application dependencies of application component *GB- Uploader*?

Figure 50: "What are the application dependencies of application component *GB- Uploader*?" Query Results

- What is the input data to Component *gB–Observation System*?

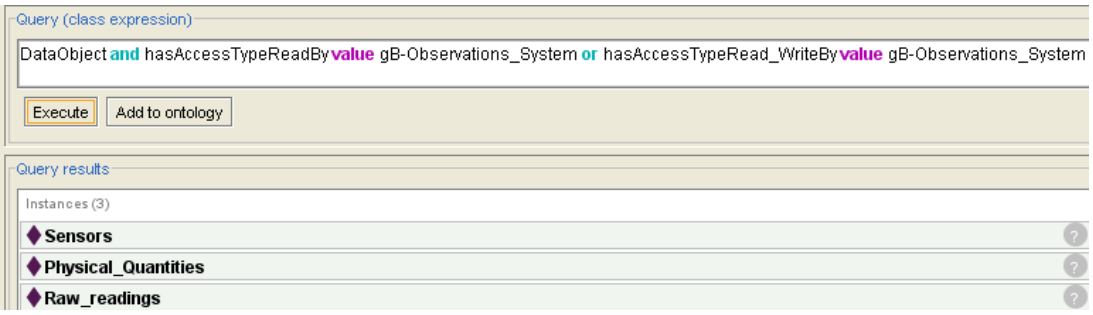

Figure 51: "What is the input data to Component *gB–Observation System*?" Query Results

- What is the output data to Component *gB–Observation System*?

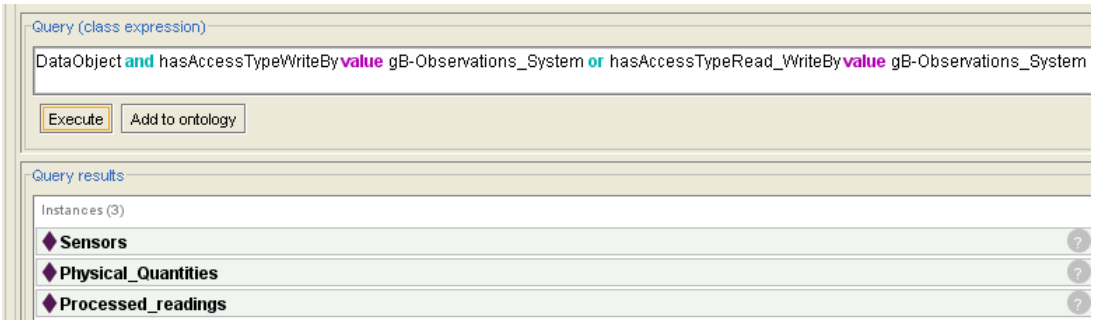

Figure 52: "What is the output data to Component *gB–Observation System*?" Query Results

8.2.3 The DSO Instances

This section shows how the sensor DSO can be used integrated with the DIO and how reasoning can be performed across the DIO-DSO. For showing this, a set of reasoning questions were posed to the model. Table 13 show the questions along with the respective formalized queries, with the first three being intra-DSO queries and the fourth being a DIO-DSO query. Figure 53 depicts the DIO instance with the individuals and properties. Figure 54 to Figure 57 depict the results from running the queries.

Table 13: Sensor DSO Questions and Queries

Question	Formalized Query
Which sensor types can measure the physical quantity Y?	SensorType and hasReading value Y
Which calibration constants are required to convert raw data into physical quantities for the type of sensor X?	Quantity and hasConstant some (SensorType and hasSensorType value X)
Which sensors (of the same type) are located in the same structural location L?	Sensor and has location value L
Which components are responsible to transform the readings for sensor type X?	ApplicationComponent and dependsUp some (Sensor and hasSensorType value X)

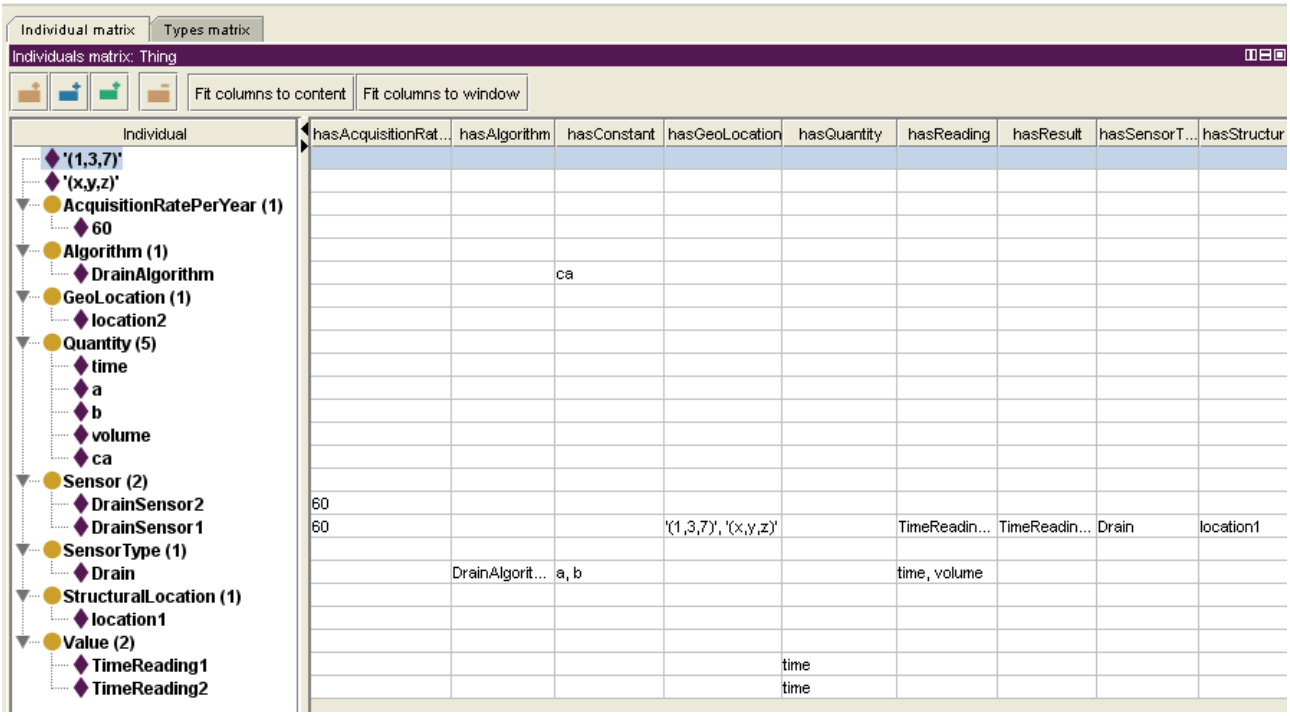

Figure 53: Sensor DSO Instantiation

- Which sensor types can measure the physical quantity *time*?

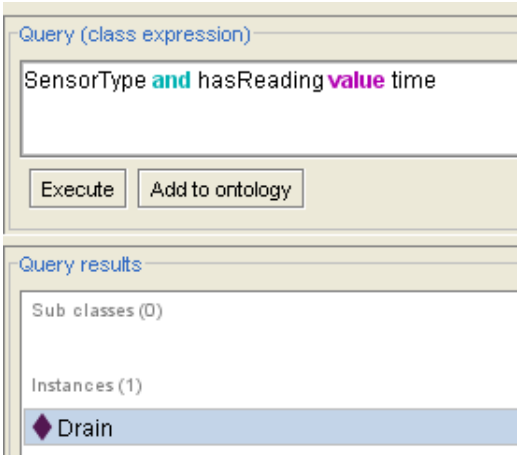

Figure 54: "Which sensor types can measure the physical quantity *time*?" Query Results

- Which calibration constants are required to convert raw data into physical quantities for the type of sensor *DrainSensor1*?

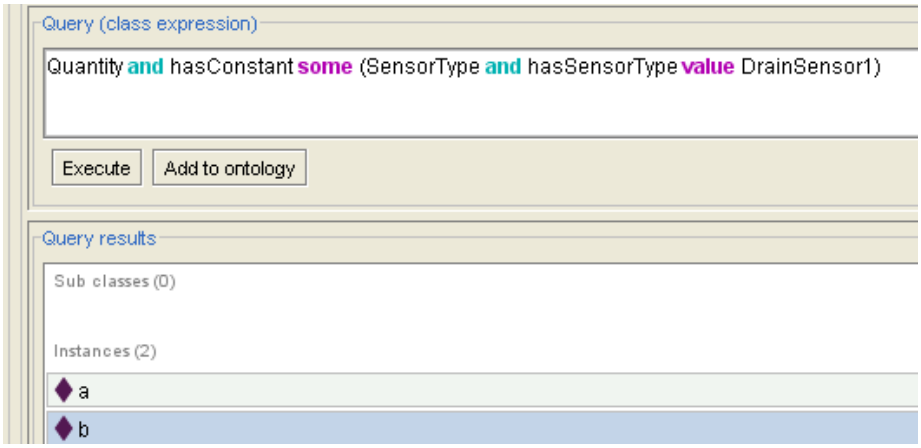

Figure 55: "Which calibration constants are required to convert raw data into physical quantities for the type of *DrainSensor1*?" Query Results

- Which sensors (of the same type) are located in the same structural location *location1*?

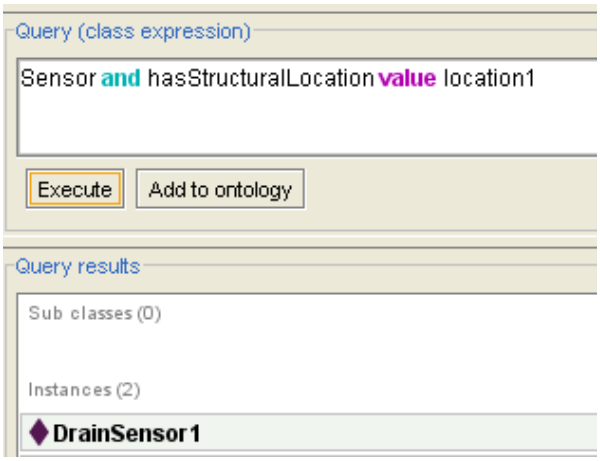

Figure 56: "Which sensors (of the same type) are located in the same structural location *location1*?" Query Results

- Which components are responsible to transform the readings for sensor type *Drain*?

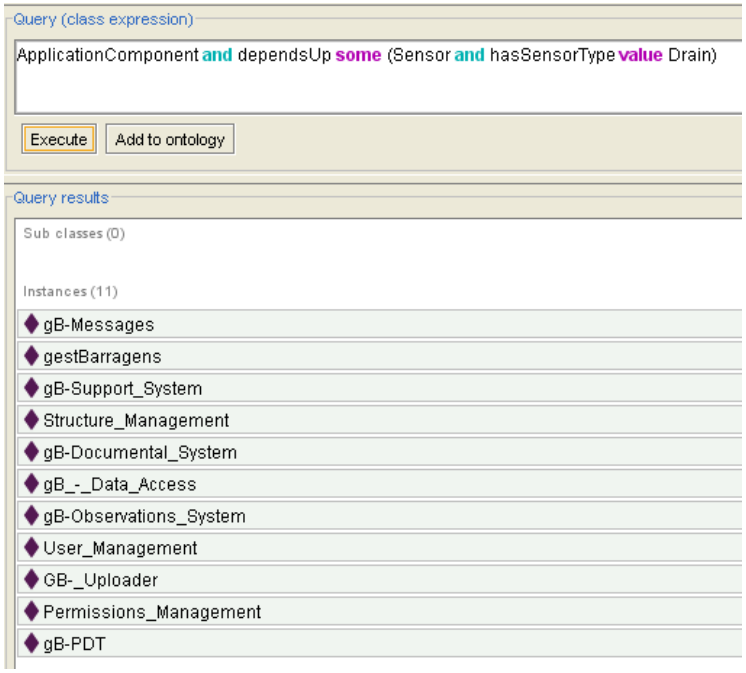

Figure 57: "Which components are responsible to transform the readings for sensor type *Drain*?" Query Results

8.3 WP9 Industrial Case

As the previous case, the WP9 Industrial Case description was also not available during the Y1 iteration of the context model. This section provides a brief description of the scenario to which the context model was applied. The DIO instance and general query results are also described. As this industrial case was the last one to be available, and since it only became available during the writing of this deliverable, no DSOs were formulated for this case for this deliverable but such analysis should take place in D4.9.

8.3.1 Brief Description of the Scenario

This scenario deals with the three companies that work together for providing a service for advising medical personal about potential adverse effects of drugs when treating patients with drug combinations: DrugFusion, which is the company that provides the service; SemanTech, which is an IT company providing services for discovering drug combinations causing potential adverse effects and for providing efficient search facilities for medical personal; and DataMole, which is a R&D company maintaining the AI algorithm for discovering the cause-effect sequences from the drug-use database provided by clinical authorities. More details on this scenario can be consulted in D9.3.

8.3.2 The DIO Instance

As in the previous case, an instance of the DIO was derived for the WP9 use case using Archi for producing an ArchiMate model, which was then converted into OWL using the conversion tools described in section 10.3. Annex H contains the ArchiMate models from which the DIO was derived. Figure 58 depicts the DIO instance with the individuals and properties. Figure 59 to Figure 66 depict the results of the general queries made to the DIO. The resulting OWL representation can be found at:

`http://timbus.teco.edu/ontologies/Scenarios/wp9.owl`

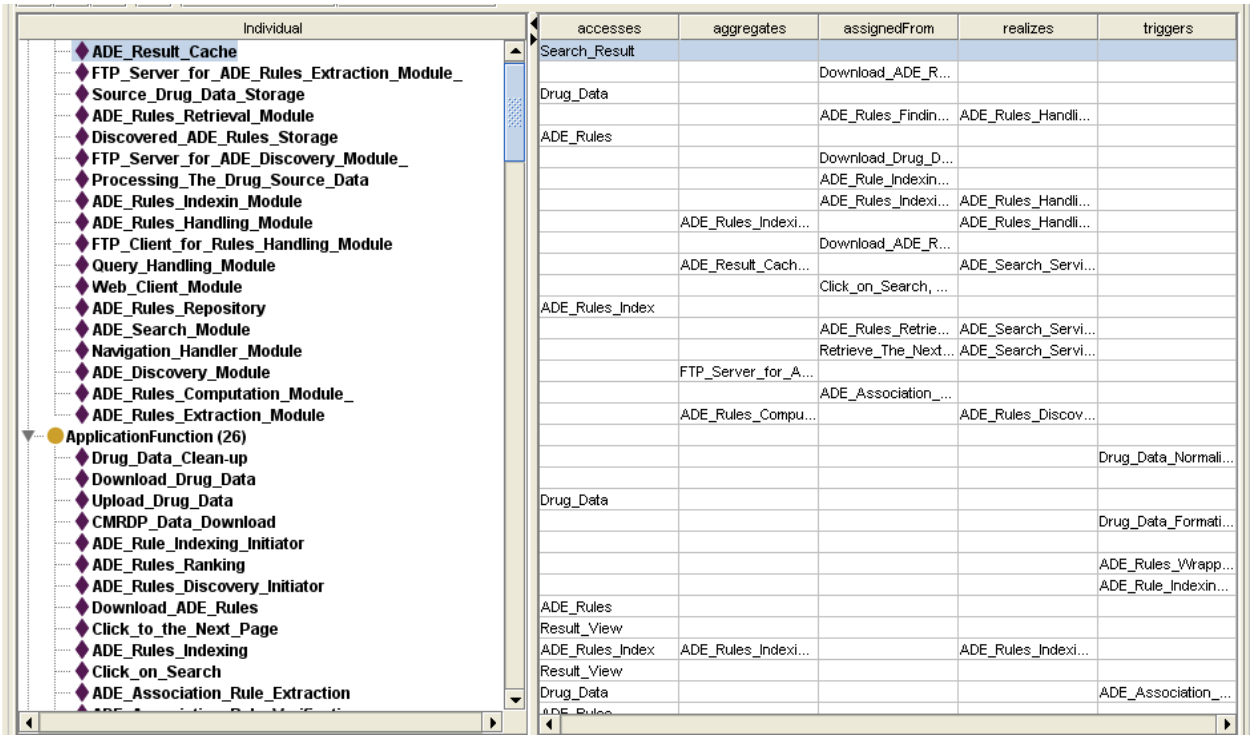

Figure 58: WP9 DIO Instantiation

- Which Service Level Agreements (SLAs) are associated with external systems?

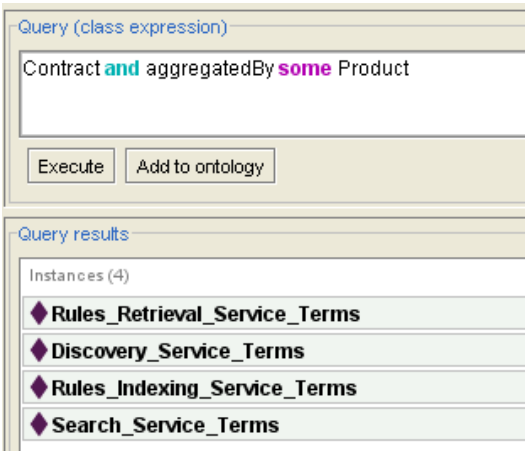

Figure 59: "Which Service Level Agreements (SLAs) are associated with external systems?" Query Results

- What business actors are assigned to business process *ADE Discovery*? *(None, as the process is fully automatic)*

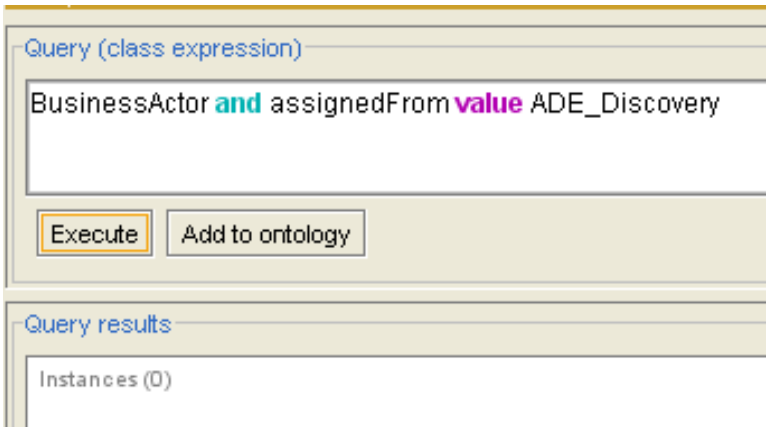

Figure 60: "What business actors are assigned to business process *ADE Discovery*?" Query Results

- What business objects are being used by business process *ADE Discovery*?

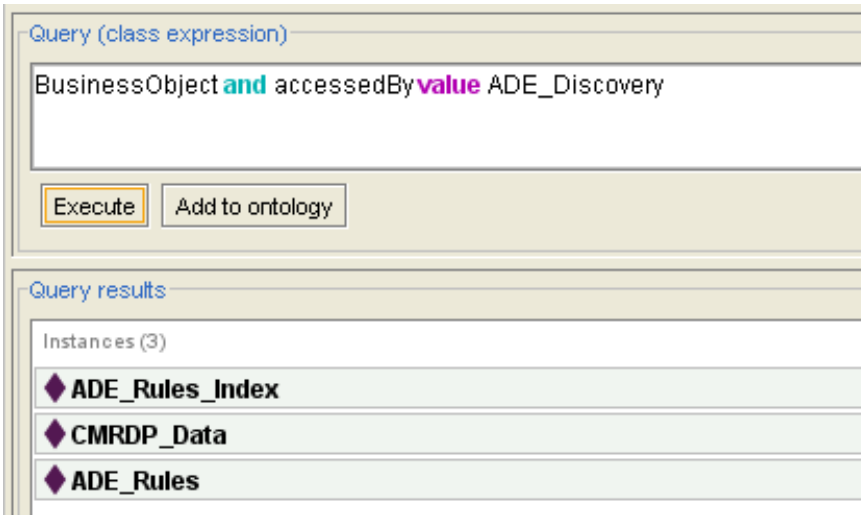

Figure 61: "What business objects are being used by business process *ADE Discovery*?" Query Results

- What application components support business process *ADE Discovery*?

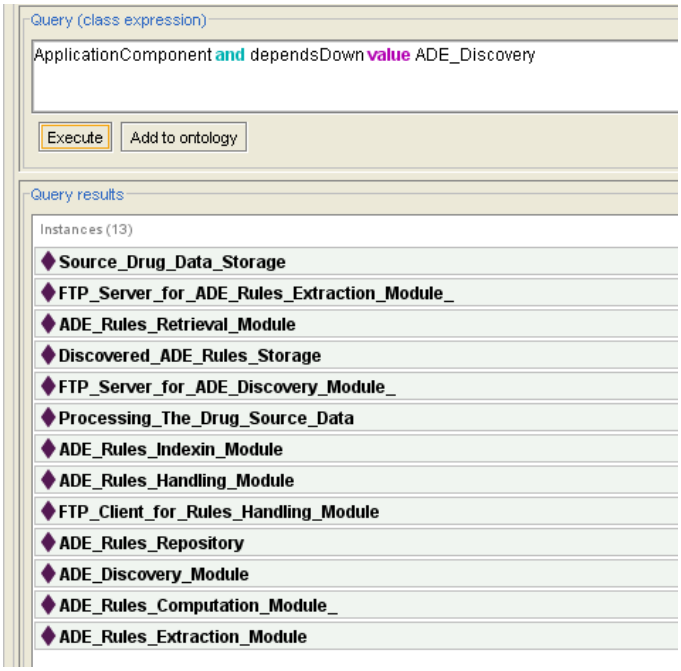

Figure 62: "What application components support business process *ADE Discovery*?" Query Results

- What are the technological entities supporting business process *ADE Discovery*?

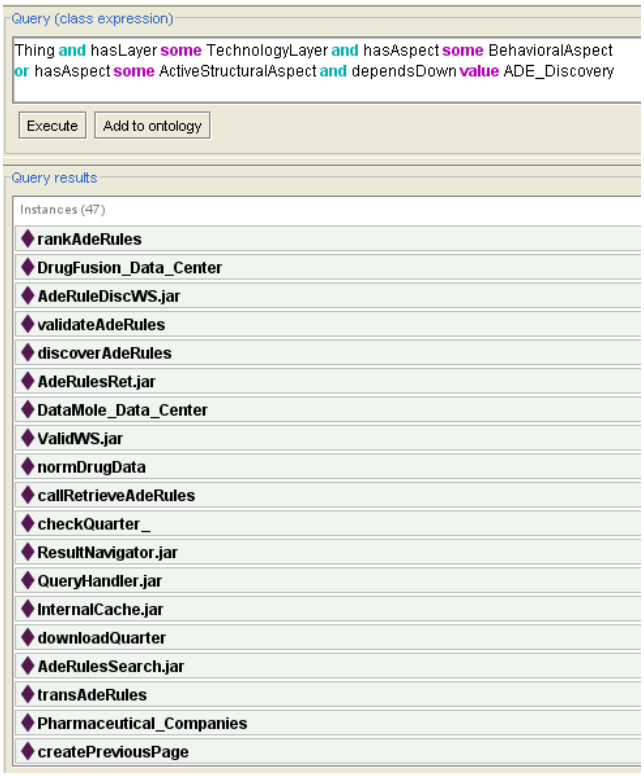

Figure 63: "What are the technological entities supporting business process *ADE Discovery*?" Query Results

- What are the application dependencies of application component *ADE Discovery*?

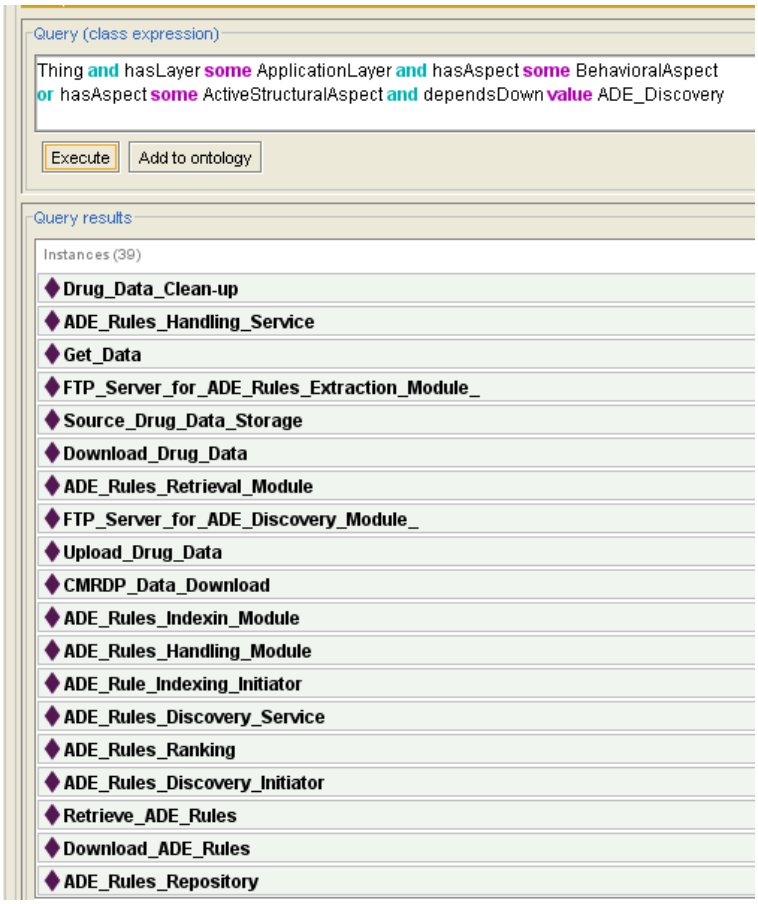

Figure 64: "What are the application dependencies of application component *ADE Discovery*?" Query Results

- What is the input data to Component *ADE Rules Repository*?

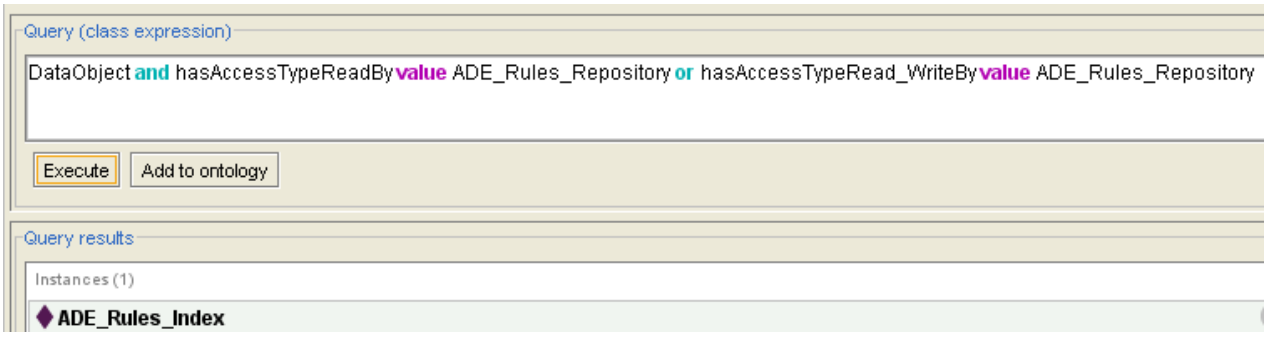

Figure 65: "What is the input data to Component *ADE Rules Repository*?" Query Results

- What is the output data to Component *ADE_Rules_Indexin_Module*?

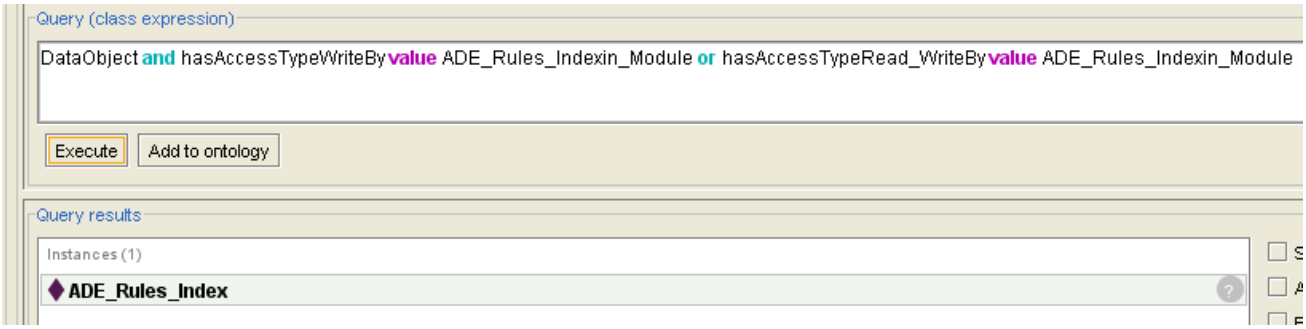

Figure 66: "What is the output data to Component *ADE_Rules_Indexin_Module*?" Query Results

9 Related Work

This section presents relevant related work concerning context and dependencies modelling that provided the background for the developments reported in this deliverable. Different areas are approached: enterprise, software, hardware, and digital preservation.

9.1 Enterprise Context and Dependencies

Enterprise context and dependencies have been a concern of enterprise architecture for a long time, as already shown with ArchiMate earlier in this deliverable. Enterprise architecture deals with ensuring the business/IT alignment, through the management of the dependencies between IT and the software/hardware stack.

One of the first references on this subject was the Zachman framework (Zachman, 1987). Zachman offers a classification matrix for categorizing the different entities existing in an organization at different abstraction levels (i.e., scope, business, system, technology, component, and instances), highlighting the fact that for achieving business/IT alignment, different aspects of the organisation should be considered: the motivational aspect, the functional aspect, the people aspect, the data aspect, the time aspect, and the location aspect. Each cell results from the crossing of an abstraction layer with an aspect, offering a holistic coverage of the organisation. Despite the fact that some suggestions are made concerning the contents of the cells, the Zachman framework does not mandate any use of particular techniques for modelling the entities that should be in the cells, and neither reinforces any intra- or inter-cell dependencies.

The Open Group Architecture Framework (TOGAF) (The Open Group, 2011) is one of the most relevant Zachman framework descendants. It offers a framework and a method, along with a meta-model specifying the relevant entities that should be captured in the framework. The meta-model defines the kinds of entities existing in an enterprise, at multiple levels, and the horizontal and vertical relationships existing between those entities, which could point to possible dependency relationships. The meta-model entities can then be instantiated in the development of concrete models of the organisation. Similarly to ArchiMate, cross-layer dependencies between the concept of process and other concepts are also possible to be observed, namely through the concept of business service, which interfaces with the logical and physical abstraction layers.

From the ontology domain, two works attempted to model the business enterprises: Enterprise Ontology and the TOVE Project. The Enterprise Ontology is a collection of terms and definitions relevant to business enterprises, developed as part of the Enterprise Project, a collaborative effort to provide a method and a computer toolset for enterprise modelling. The Enterprise Ontology is composed by a set of entities and relationships between entities. Entities can have roles in relationships. An attribute is a special kind of relationship and a state of affairs a situation which is characterised by a combination of entities in any number of relationships with one another (Uschold et al., 1996). The TOronto Virtual Enterprise (TOVE) project aimed towards the development of an ontological framework for Enterprise Integration (EI) based on

and suited for enterprise modelling. TOVE first identifies the objects in the domain of discourse that will be represented by constants and variables in TOVE's syntax (Fox et al., 1997). Subsequently the properties of these objects are identified as well as the relations that exist over these objects and these are represented by predicates in TOVE.

Context and dependencies are also present in other attempts to model more specific aspects of the enterprise. The Business Process Model and Notation (BPMN) is a specification adopted by the Object Management Group, and it is currently on its 2.0 version (OMG, 2011). BPMN creates a standardised bridge for the gap between business process design and process implementation, providing a language and notation for creating business process diagrams. BPMN captures intra-process dependencies, such as activities that might depend on other activities, on certain events, or even on determined data.

9.2 Software Context and Dependencies

Software context and dependencies can also be modelled using typical software modelling languages. Different approaches to the modelling of context and dependencies are possible: at the conceptual level, with conceptual modelling languages, and at the technical level, with representation formats specifically for capturing concrete technical dependencies.

At the conceptual level, the Unified Modelling Language (UML) is one of the best known examples. It is a standardised, general-purpose modelling language, created and managed by the Object Management Group (OMG) (OMG, 2007). It offers a diverse set of diagrams displaying both structural and behavioural aspects of software, and as such, the dependencies between those two aspects. The Service oriented architecture Modelling Language (SoaML) is an UML profile and meta-model for the specification of service oriented architectures adopted by OMG (OMG, 2009). It can be used for the modelling and specification of service oriented architecture at a conceptual level, allowing dependencies to be made explicit at that level.

At the technical level, dependency representation formats are available particularly for the Linux operating system. The Common Upgradeability Description Format (CUDF) (Treinen et al., 2008) and the Distribution Upgradeability Description Format (DUDF) are formats for describing upgrade scenarios in package-based Free and Open Source Software (FOSS) distributions. CUDF was designed to capture and express upgrade problems in a format that is independent of the distribution and allows for solvers to work on identifying possible solutions for upgrading a set of packages requested by the user. DUDF is generated on a per distribution basis and captures information about the packages that are available to the installer at the time the upgrade request is made, which is then submitted to a centralised server based on the distribution. The distribution servers collate the information and convert the DUDF files into CUDF. The CUDF representation is then submitted to a centralised repository where the upgrade problem sets can be collected.

The Virtual Resource Description Framework (VRDF) (Kadobayashi, 2010) is a framework developed to describe and analyse complex dependencies in the context of cloud computing. As such, it aims at representing dependencies of services and virtualised infrastructure (such as virtual machines or virtual networks) to the physical infrastructure. To this end, it provides an RDF schema to model connections,

devices, networks and other related entities. This model shall then allow to, for example, assess the impact of failures of physical hardware on the virtual infrastructure.

9.3 Hardware Context and Dependencies

Hardware dependency analysis is a subset of the larger area of inventory and asset management. Any large enterprise organisation or IT department can expect to be using tools available today to aid with asset management such as SAP Enterprise Management, IBM's Enterprise Asset Management, and Xasset's Asset Management. These are essentially inventory tools which will automatically scan all devices on the network to build up a map of the IT landscape. Being proprietary tools, the internal representation formats/schemas are not available.

Despite this fact, some proposals for representing devices have surfaced throughout the years. The Foundation for Intelligent Physical Agents (FIPA) device ontology (FIPA, 2002) is such an example. It has the aim of being used by agents when communicating about devices. As such, it captures different aspects of technological devices, such as hardware and software descriptions. The IT Service Management Ontology (ITSMO) is a more recent example of an ontology for describing IT services and their dependencies to IT components, claiming to be aligned with the ITIL Glossary (ITSMO Project, 2011).

9.4 Digital Preservation Context and Dependencies

The Reference Model for an Open Archival Information System (OAIS) (CCSDS, 2002) is the de-facto reference model for digital preservation. Besides providing the terminology, it also provides a reference information model for guiding the implementation of information packages for preservation. The OAIS considers that an Information Object is composed of a Data Object and the Representation Information, which adds meaning to the data object, so that it can be interpreted in the future. The Representation Information might contain Structure Information, which describes the way the data on a data object is structured, Semantic Information, which provides meaning to the structures defined by the Structure Information, and other Representation Information, such as Representation Networks, which might contain all the linkages of Data Objects and Representation Information required for interpreting a Data Object.

Different specialisations of Information Object are possible: Content Information, which represents the data object targeted for preservation and the accompanying Representation Information; Preservation Description Information, which includes information that is needed in order to adequately preserve the Content Information; Packaging Information, which binds or related the components of the package to be preserved (Content Information plus Preservation Description Information) into an identifiable entity; and Descriptive Information, which allows the search for and retrieval of the information packages.

The PREservation Metadata: Implementation Strategies (PREMIS) preservation dictionary (PREMIS Editorial Committee, 2012) provides a set of conceptual elements and the relationships between such elements. It is implementation independent as the elements define information needed for preservation regardless of how that information is stored. The semantics from PREMIS carry dependency relations for information such as

usage notes, applicability, object categories, data constraints, rationale, and the environment needed for rendering the information.

The notion of preservation network models was created with the intention of representing digital objects and relationships, depicting the dependencies existing between objects, so that these can be understood in the future and preservation objectives are met (Conway et al., 2011). These preservation networks can then be stored in registry repositories of representation information, so that knowledge can be reused. Preservation networks are represented in a similar fashion to that of class diagrams, depicting to kinds of entities: Objects, which are uniquely identified digital entities with the attributes of information, location, and physical state; and Relationships, which have the attributes of function (for depicting any necessary function to be performed on object), risks and dependencies, tolerance (if the absence of a determined function is critical or not), and quality assurance and testing (if a determined function has been subjected to testing or quality assurance). Relationships can be composed in to alternate or composite relationships, depicting respectively the fact that only one relationship needs to function or the fact that all the relationships must function in order to fulfil the objective.

10 Tools

In this section, the tools used for the creation of the DIO, DSOs, respective instances, and converters are described.

10.1 Archi

Archi[6] is a free, open-source, cross-platform tool and editor to create ArchiMate models. Its evolution as an open-source GUI is closely linked to the developments of the TOGAF standard and the emerging results from The Open Group forums and work groups active in this area. It is currently in version 2.4 and it is built upon the Eclipse 3.8.1 Rich Client Platform (RCP). It is built in a modular fashion and it can be extended by means of eclipse-based plug-ins.

It is considered an easy-to-use graphical environment, offering user assistance. It provides the standard ArchiMate viewpoints, providing graphical cues which enable/disable modelling elements that should not appear in a view derived according to a viewpoint. It also enforces the ArchiMate meta-model so that the only relationships that can be established and valid are those allowed to be sketched by the tool. According to its website, Archi enjoys a large user community and tool support and its becoming the de-facto open-source modelling tool for ArchiMate, with new features added to it on a regular basis.

10.2 Protégé

Protégé[7] is a free, open source ontology editor and knowledge-base framework. It is used for manipulating ontologies, as well as testing queries and ground rules on them. It provides numerous facilities, including a number of reasoners, Consistency Checking, a DL Query interface, and a SPARQL Query engine. Protégé is based on the Java programming language, and provides an extensible environment that makes it a flexible base for ontology prototyping and development.

It contains numerous plug-ins (tab plug-ins, slot widgets, back-ends) that add new functionalities and new visualization facilities. It supports OWL as well as other ontology file formats, providing navigation facilities that can aid in the management of the ontology. It includes support for class hierarchy with multiple inheritance; template and own slots; specification of pre-defined and arbitrary facets for slots (which include allowed values, cardinality restrictions, default values, inverse slots). It also provides flexible modelling components; e.g. meta-classes and meta-class hierarchy.

10.3 ArchiMate to OWL Converters

As decided in Y1, the context model would be based on ontologies, and particularly in OWL. As the new iteration of the context model is inspired in ArchiMate, it was thus decided to take advantage of available

[6] http://archi.cetis.ac.uk/

[7] http://protege.stanford.edu/

modelling tools, which lead to the adoption of Archi as the ArchiMate modelling tool to be used for the manual capture of context information. Although Archi uses an XML-based format for storing ArchiMate models, it is still necessary to proceed to the conversion of the models into OWL in order to take advantage of the features provided by that particular technology.

The method employed for converting Archimate into OWL encompasses the three phases: (*i*) Transforming the ArchiMate Meta-model; (*ii*) Adding Axioms and Cardinalities; (*iii*) and Transforming the ArchiMate Models. Phases (i) and (ii) are one-time efforts independent from phase (iii), which is performed for each specific case being addressed.

10.3.1 Transforming the ArchiMate Meta-model

For the transformation, an XML-based representation of the ArchiMate Meta-model and official extensions provided by the Archi tool was used. First, the ArchiMate meta-model and extensions were analyzed concept by concept, including the meanings, the relationship with other concepts, and the constraints existing in such relationships. Then, transformation rules were created, which involved defining a map from the equivalent elements of the XML format to the equivalent elements of the OWL representation.

The XML elements were transformed into OWL Classes, as the definition of an element in the XML representation of the ArchiMate meta-model and a class in OWL can be considered equivalent. Relations were transformed into object properties in OWL. The excerpt of the XML-based representation of the ArchiMate meta-model used by Archi depicted in Figure 65 shows one ArchiMate source element and possible relationships that can be maintained with other target elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--ArchiMate 2.0 rules -->
<elements>
<source element="BusinessActor">
<target element="BusinessActor" relations="cfgostu" />
<target element="BusinessRole" relations="fiotu" />
<target element="BusinessCollaboration" relations="fiotu" />
<target element="Location" relations="o" />
<target element="BusinessInterface" relations="fiotu" />
<target element="BusinessProcess" relations="fiotu" />
<target element="BusinessFunction" relations="fiotu" />
<target element="BusinessInteraction" relations="fiotu" />
<target element="BusinessEvent" relations="ot" />
<target element="BusinessService" relations="ioru" />
<target element="BusinessObject" relations="ao" />
<target element="Representation" relations="o" />
<target element="Product" relations="o" />
<target element="Contract" relations="ao" />
<target element="Meaning" relations="o" />
<target element="Value" relations="o" />
```

Figure 67: Excerpt of the Archi XML ArchiMate representation

The mapping between the Archi XML representation of ArchiMate and an OWL representation of ArchiMate is shown in Table 14.

Table 14: Transformation Rules for the ArchiMate meta-model (as implemented by Archi)

XML	OWL
source element	class
target element	class
relations	object properties

A converter was then implemented in Java for executing the transformation rules. The converter parses the XML file and transforms the elements representing ArchiMate concepts in OWL Classes. Table 15 depicts a source element in the XML representation and the corresponding class in OWL.

Table 15: Archi XML representation element and respective OWL Class

XML	OWL
<source element=”BusinessActor”>	<Declaration> <Class IRI=#BusinessActor”/> </Declaration>

The converter begins by processing the source elements, and after it processes the relations to the target elements. For each relation found in the XML file, an OWL ObjectProperty is generated. Table 16 depicts such an example, where each one of the characters in the relations attribute is mapped to an ObjectProperty in the OWL ArchiMate representation, following the rules displayed in Table 17.

Table 16: Archi XML representation element relations and respective OWL ObjectProperties

XML	OWL
<source element=”BusinessActor”> <target element=”BusinessObject” relations=”ao”/> <target element=”Contract” relations=”ao” /> </source>	<Class IRI=#BusinessActor”/> <ObjectAllValuesFrom> <ObjectProperty IRI=”#accesses”/> <ObjectUnionOf> <Class IRI=”#BusinessObject”> <Class IRI=”#Contract”/> </ObjectUnionOf> </ObjectAllValuesFrom>

Table 17: Archi XML representation relations and respective OWL ObjectProperty Mappings

Archimate Identifier	OWL ObjectProperty
a	accesses
i	assignedFrom
c	composedOf
r	realizes
t	triggers
g	aggregates
o	association
f	flowTo
s	specialization
u	usedBy

10.3.2 Adding Axioms and Cardinalities

After the transformation to the OWL representation, it was checked for missing concepts, relationships and/or constraints. It was detected that the converted ontology was missing Inverse Object Properties and Super Object Properties axioms. These were added to the OWL ontology so that derived relationships could be inferred through the use of reasoners. Cardinalities were also added to the concepts to reinforce the coherence and conformance of the ontology to the ArchiMate meta-model. For instance, in Figure 19, the class called Business Function has exactly one layer which is Business layer and has exactly one aspect (structure) which is Behavioural aspect.

10.3.3 Transformation of the ArchiMate Models

Besides the transformation of the ArchiMate meta-model into an OWL representation, the ArchiMate models, i.e. instances describing an enterprise, also need to be converted to an OWL representation. The plug-in processes the instances of the concepts of the model and transforms them to an ontology containing OWL Individuals and importing the classes from the ArchiMate meta-model, which was previously converted. Table 18 describes the mappings from the ArchiMate model elements to OWL.

Table 18: Mapping between the Model XML Representation and OWL

Archi XML	OWL
Element instance	individual
Element instance relationship label	Sub- property of the object property
Element instance property key	Data property
Element instance value	Value of the Data property

Figure 68 depicts an excerpt of an ArchiMate model with three concept instances: customer, claim registration service, and customer information service. The corresponding OWL excerpt can be seen in Figure 69, with the relationships of the type “usedBy” with the label “update” reflected in there. The instance customer has the property age and the value 22, which is converted to a data property and value of data property, respectively.

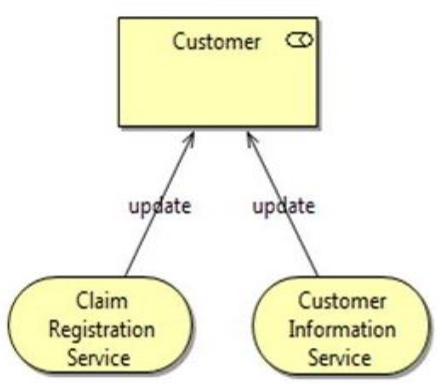

Figure 68: ArchiMate Model Excerpt

```
<ClassAssertion>
      <NamedIndividual IRI=”#Customer”/>
</ClassAssertion>
<DataPropertyAssertion>
      <DataProperty IRI=”#age”/>
      <NamedIndividual IRI=”#Customer”/>
      <Literal>22</Literal>
</DataPropertyAssertion>
<ClassAssertion>
      <Class IRI=”#BusinessService”/>
      <NamedIndividual IRI=”#Claim_Registration_Service”/>
</ClassAssertion>
<ClassAssertion>
      <Class IRI=”#BusinessService”/>
      <NamedIndividual IRI”=#Customer_Information_Service”/>
</ClassAssertion>
<SubObjectPropertyOf>
      <ObjectProperty IRI=”#update”/>
      <ObjectProperty IRI=”#usedBy”/>
</SubObjectProperty>
<ObjectPropertyAssertion>
      <ObjectProperty IRI=”#update”/>
      <NamedIndividual IRI=”#Customer_Information_Service”/>
      <NamedIndividual IRI=”#Customer”/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
      <ObjectProperty IRI=”#update”/>
      <NamedIndividual IRI=”#Claim_Registration_Service”/>
      <NamedIndividual IRI=”#Customer”/>
</ObjectPropertyAssertion>
```

Figure 69: DIO Instance OWL Excerpt

11 Conclusion and Outlook

TIMBUS has a different scope from that of traditional digital preservation methods. It addresses business process preservation, which not only covers all aspects of traditional digital preservation systems (such as preserving digital contents), but also addresses enterprise risk analysis and business continuity planning. It covers a wider scope of digital preservation processes, which includes intelligent Enterprise Risk Management (iERM) for automatic identification and prioritization of risks within an enterprise, and the ability to minimize those risks by taking a specific set of actions including digital preservation. The TIMBUS system identifies a set of interdependent business processes, automatically detects and captures relevant context meta-data, packages the collected information, and provides facilities for long-term preservation, monitoring, and maintenance. The TIMBUS system enables the re-deployment and also re-execution of partial or complete business processes at a future time.

In previous deliverables (D4.2/4.5) we had discussed the notion of context in digital preservation, and identified relevant components of context and the dependencies and relationships between the different aspects of it. We had developed a first version of a generic model to represent the required contextual information. This deliverable has presented an approach that was built-upon the experience gathered during Y1 of the project, offering a structured and methodical means for modelling and capturing context and dependencies. This was performed according to industrial use-cases' requirements and best practices, resulting in a comprehensive and extensible model. We have also elaborated a governance method for creating augmentations to this model, and integrated three major extensions to it dealing with patents, software licenses, and sensors. Moreover, we applied the model to the Music Classification Process, WP8 Industrial Use-Case, and WP9 eHealth Use-Case.

We now have a comprehensive and integrative model developed with industrial use-cases in mind, and an extensible architecture along with a governance method for evolving it further (to be reported in D4.9 due in M36). Future work will focus on the development of the identified DSOs that were not addressed until this point and on the improvement of the already addressed industrial cases with instances of such DSOs. Additionally, the WP7 industrial case will also be addressed, which might bring new requirements into consideration.

Annex

A Stakeholder's Questions to the Context Model

Table 19: Stakeholder's Questions to the Context Model

Question	Domain / Concern	Example Output
What legal requirements R are verified by business process BP?	Legal	List of legal requirements
Does business process BP depend on legal requirement R?	Legal	Yes/No answer plus the explanation of the decisions (e.g. dependency graph)
Which legal requirement R is a license?	Legal	List of legal requirements
What business processes BP comply with legal requirement R?	Legal	List of processes
Which licenses L are open-source?	License	List of licenses
What are the licenses L required to execute software application SA?	License	List of licenses
What restrictions on preservation actions are allowed according to license L?	License	List of actions applicable to the source code
Which patents are required for a certain component C?	Patents	List of patents
Which Service Level Agreements (SLAs) are associated with external systems?	Business	List of SLAs
What business actors are assigned to business process BP?	Business	List of business actors
What business objects are being used by business process BP?	Business	List of business objects
What business goals G are affected if component C cannot be preserved?	Business	List of (domain-specific) business goals
What business actors R support activity A of business process BP?	Organisation	List of business actors
What is the organizational structure of company C?	Organisation	Organisational chart
What are the time constraints associated with business process BP?	Organisation	List of time constraints
What application components C support business process BP?	Application	List of application components
What application components C support activity A of business process BP?	Application	List of application components
What application components C depend on requirement R?	Application	List of application components
What are the application dependencies of application component C?	Application/Software	Graph of application dependencies
What is the taxonomy of applications of domain D?	Software	Graph of (domain-specific) applications
Which data objects D use data format F?	Data Formats	List of data objects
What is the set of data D needed to preserve the current state of business process BP?	Data	Set of data
What alternative format F can be used to replace format G?	Data Formats	A (domain-specific) format
What formats F used in process P are in danger if component C cannot be preserved?	Data Formats	List of (domain-specific) formats

What alternative Interfaces E can I use to replace Interface EE?	Application	List of (domain-specific) interfaces
Which data formats used are not standardized (by an external body that would maintain them)?	Data Formats	List of data formats
What components of business process P implement the behavior?	Application	List of components that execute the business process
What components of business process P are documentation or description?	Software	List of components that document or describe the business process and its artifacts
What and How are the services used in the business process of specific component C?	Application	Used services / Used access of components
What storage nodes N support application component AC?	Hardware	List of nodes
What are the technological entities T supporting business process BP?	Technology	List of structural and behavioural technological entities
What is the taxonomy of hardware nodes at organization O?	Hardware	Graph of (domain-specific) technology
What is the minimum set of components needed to preserve business process BP?	Technology	List of components
What are the external processing components of business process BP?	Technology	List of external processing components
What are the external data sources of business process BP?	Technology	List of external data sources
What components C support business process BP?	Technology	List of components
What alternative components C can I use to replace component D?	Technology	List of components
What are the software dependencies of business process BP?	Technology	List of software dependencies
What are the hardware dependencies of business process BP?	Technology	List of hardware dependencies
Which components C are provided by external parties?	Technology	List of components
Which components C are available only in binary format?	Software	List of components
What is the input data to Component C	Data	List of data
What is the output data of Component C	Data	List of data
What is the data base of Component C	Technology/Software	Likely a list of data objects, e.g. files that store the data
What is the configuration of Component C	Technology	Likely a list of data objects, e.g. files that store the configuration
WP8 - LNEC		
What business actors BA are involved?	Business	list of LNEC business actors
What business services BS are involved?	Business	list of LNEC business services
What business processes BP are involved?	Business	list of LNEC business processes
Which business processes depend on business process BP?	Business	list of business processes
Which business actors BA are required to execute business process BP?	Business	list of business actors
Which human resources are qualified to execute business process BP?	Business	list of people that are qualified to perform the business process

What personal information is recorded by business process BP?	Data	list of data
What is the set of legal requirements, established by general law, related to business process BP?	Legal	list of legal requirements
What is the set of legal requirements, established by specific law, related to business process BP?	Legal	list of legal requirements
What is the set of legal requirements, established by contracts with third-parties, related to business process P?	Legal	list of legal requirements
What are the current legal controls to ensure the confidentiality of element E?	Legal	list of legal controls
What is the confidentiality level for element E?	Legal	description of confidentiality restrictions
What are the "rights" that apply to business process BP?	Legal	list of rights
Which business processes deal with personal information PI?	Legal	List of processes
What physical quantities can be measured by sensor type X?	Sensors	list of physical quantities
Which sensor types can measure the physical quantity Y?	Sensors	List of sensor types
Which sensor can measure the most approximate physical quantity measured by sensor X?	Sensors	sensor
What are the measurement units for sensor X?	Sensors	physical units
Which components are responsible to convert the raw data into the physical quantities?	Application	list of components
What are the properties of conversion algorithms?	Sensors	List of algorithm properties
What is the acquisition frequency for sensor X?	Sensors	frequency value
When was the last calibration?	Sensors	date
Which calibration constants and sensor properties are required to convert raw data into physical quantities for sensor type X?	Sensors	list of properties for sensor type X
What are the calibration constants and sensor properties required to convert raw data into physical quantities for sensor type X?	Sensors	list of properties for sensor type X
Does the business process BP require human intervention?	Technology	Y/N
What is the sensor location (x, y, z)?	Sensors	coordinates
What is the sensor location in the structural classification?	Sensors	structure element
Which sensors (of the same type) are located in the same structural location?	Sensors	sensor list
Does the transformation algorithm depend on other sensors? If yes, which sensors and which measurements are required to apply the calculation?	Sensors	Y/N + list of sensors
What is the observation plan for dam X?	Sensors	list of sensors and their required acquisition rate

What was the real execution of the observation plan for dam X during a specific period P?	Sensors	List of sensors and their real acquisition rate during period P
What is the logical and physical representation of information entity E?	Data	The idea here is to refer to the meta-data information, asking about the logical organization of information (e.g. an XML schema) and its physical representation in bitstream.
Which metadata information is available for architectural element E?	Data	list of metadata information
What are the physical properties of the deployed environment DE?	Technology	list of physical properties
WP9 - eHealth		
What is the minimum set of components required to re-run business process BP for discovering drug adverse-effect sequences?	Technology	The minimum set of software components include: • sequence discovery module • validations module • packaging module The minimum set of support components include: • JDK 1.6 • MySQL 7.5 Standard Edition • Apache Network/IO commons
What output format is used by the current sequence discovery algorithms and its alternative CAPRI and SPADE? How can they be utilized by other supporting software components for drug sequence discovery?	Technology	The current implementation of the sequence discovery algorithm provides a direct interaction with the database. The final result is stored within the table, with predefined schema. • The current Java implementation of the CAPRI algorithm produces output in XML format. • The current C# implementation of SPADE algorithm produces output into the plain file format, where fields are separated by “Tabular” character.
What is the set of software and hardware platforms available for the migration of the drug sequence discovery algorithm A (e.g. Linux, Windows, IBM AIX and etc…)?	Technology	The current adverse-effect discovery algorithm can be run on Windows Server 2010 and Ubuntu 12.10 Server under JDK 1.6. The following platforms can be used for alternatives: • The CAPRI implementation can be run on the same platforms Windows Server 2010 and Ubuntu 12.10 Server. • The SPADE implementation can be run on Windows Server 2010 under .NET Visual Studio 5.
What set of expertise is required for maintaining the current drug sequence discovery algorithm?	Technology	The maintenance of the current drug sequence discovery algorithm requires the following expertise: • A general knowledge of AI and deep understanding of pattern discovery algorithms • Deep knowledge of OOP • Expert knowledge in Java, XML, and SQL • Knowledge of Windows and Linux operating systems • General knowledge of drug interactions
How can modifications of the drug adverse-effect discovery algorithm be verified?	Technology	The verification of the drug sequence discovery algorithm is performed via pre-defined subset of control sequences. Control sequences are prepared by members of medical authorities and describe well-identified drug interactions and adverse–effects. All modifications of the sequence discovery algorithms must pass all predefined test with competence of no less than 90%.
What effort of integration is required for alternative solutions, and what are the associated legal constraints R?	Tech / Legal	The alternative sequence discovery algorithms have the following integration effort (in PM) and legal constraints: • The integration of CAPRI requires approximately 5PM. The algorithm is patented. The patent is active for the next 4 years. • The integration of SPADE requires approximately 10PM. The algorithm is patented. The patent is active for the next 8 years.

What application components within the drug adverse-effect sequences discovery have the legal constraints R (such as patent restrictions)?	Legal/Patents	The software component responsible for the sequence discovery uses a patented algorithm.
How long the discovered sequences and applied algorithms need to be maintained to address any legal claims from the system users?	Legal	The discovered sequences and applied algorithms need to be maintained for 5 years to address any claims related to the quality of generated results.
Does the medical data-set for drug sequence discovery contain personal data of patients?	Data	Yes, the medical data-set used for drug sequence discovery contains personal data of patients. However, during the download of this data-set, all records containing the personal information are encrypted.
Do SemanTech and DataMole allow to store personal data of patients even in encrypted format?	Legal	No, SemanTech and DataMole are not allowed to store any personal data, even in an encrypted format.
Does DataMole have permission to apply any other data mining techniques on obtained medical data?	Legal	No, DataMole is not allowed to use any other data-mining techniques for processing the obtained quarter of medical data.
What patents are used by DataMole during sequence discovery process?	Patents	The DataMole uses the patent “Advance sequence discovery algorithm for drug related data”. Patent was granted on 14 April 2006.
How long is the patent valid for?	Patents	The patent used by DataMole for sequence discovery is valid for 10 years. The date of expiry is 14 April 2016.
Who is the owner of the patent P?	Patents	The patent used in sequence discovery has shared ownership between DataMole and DataFusion.
Who has access to the medical data?	Legal	In DataFusion, the following employees have access to the medical data: • System Administrator • Search/Discovery Support engineers • Support Medical Personnel In SemanTech, the following employees have access to the medical data: • Database Administrator In DataMole, the following employees have access to the medical data: • System Administrator • Software Engineers
What limitation exists on the access to the medical data?	Data	The medical data must be used only for sequence adverse-effect discovery. Usage of this data for any other purposes is strictly prohibited.
Do the business processes use personal data?	Data	No, the business processes in WP9 use-case do not use personal data (downloaded drug data are already anonymised). If the downloaded drug data contained the personal information, the reasoning presented in table 2 would need to be addressed.
What the time period for how long the data need to be stored before deletion?	Legal	In the use case WP9 the obtained data need to be store during 50 year.
What is the reason for preserving the personal data?	Data / Legal	It could be developing personalized systems.
Is the preservation of personal data necessary to achieve this purpose?	Data / Legal	The preservation of the personal data is only necessary in case of delivering analytic results back to the source.
Did the data subject give their consent to preserve data stored in the database for providing personalization of future prescriptions?	Data / Legal	Yes, such consent must be obtained.
Was the subject informed that he/she has the right to revoke his consent to the use of their personal data?	Data / Legal	Yes, the consent must be given.
Does the database contain a possi-	Data / Legal	Yes, the database must have the possibility to delete

bility to delete the personal data of a patient who has revoked his consent to the use his personal data?		any records associated with any patient.
What happens in case of death of a patient with his personal data?	Data / Legal	The inheritors of the patient should have the possibility to request the deletion of his personal data upon presenting his death record.
How do you inform the patients in case of a change of purpose in their personal data?	Data / Legal	It is necessary that they are immediately informed about any changes and that there is a possibility that they can give their new consent of the use of their personal data.
Where do you want to store the preserved personal data?	Data / Legal	We need the exact location of the data storage only in case of having to deal with the personal data.
When the data has to be transferred to another server, where would the server be? In the same country, in another EU-Country or outside the EU?	Data / Legal	If the data transfer has to take place, it is important to know the new location.
Did your patients give their consent to the transfer of their personal data to another server?	Data / Legal	Yes, the data subject needs to give the consent for transferring data to a new server.

B ArchiMate Metamodel

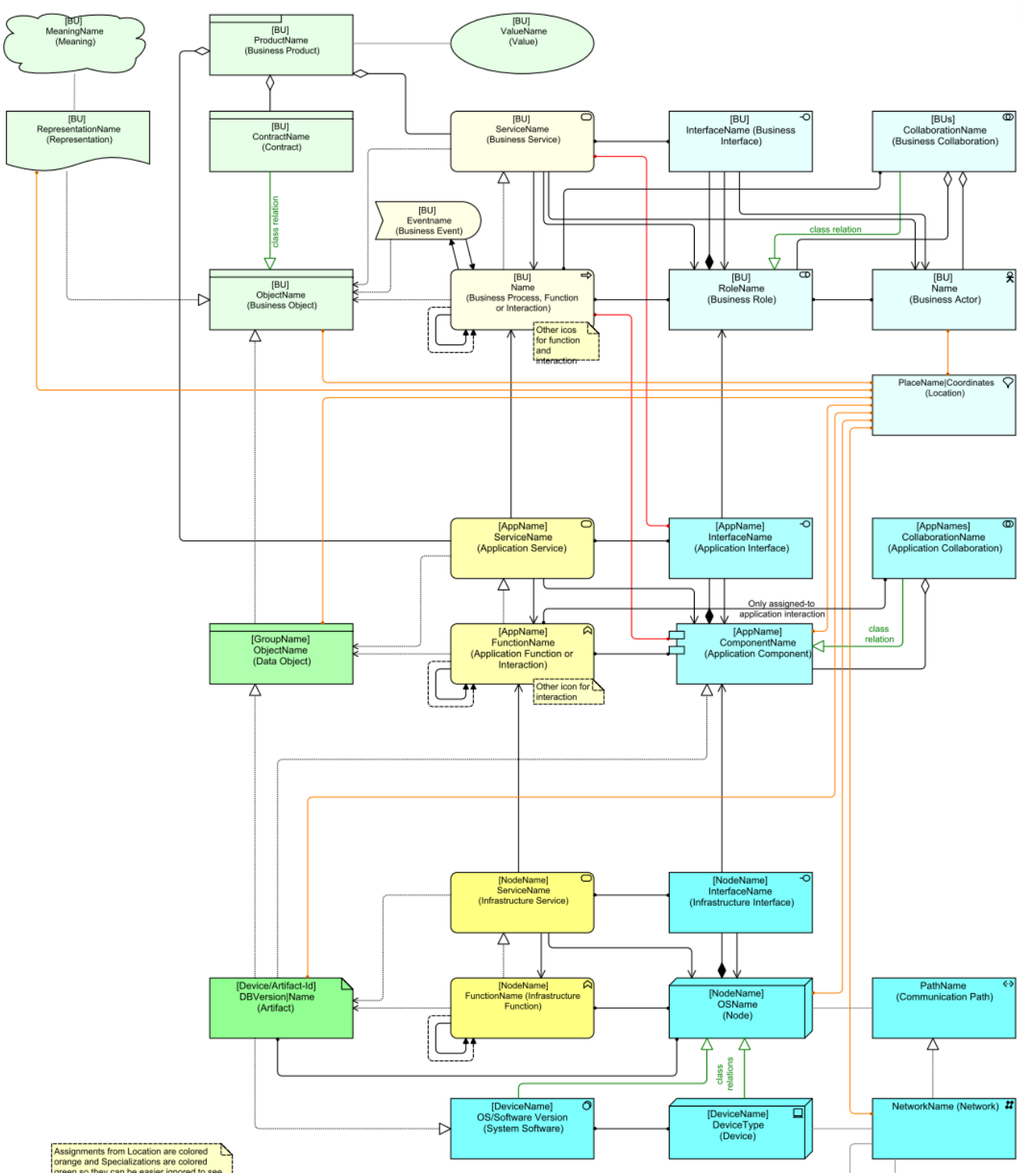

Figure 70: ArchiMate 2.0 Metamodel (Wierda, 2012)

C Patent Metadata Ontology

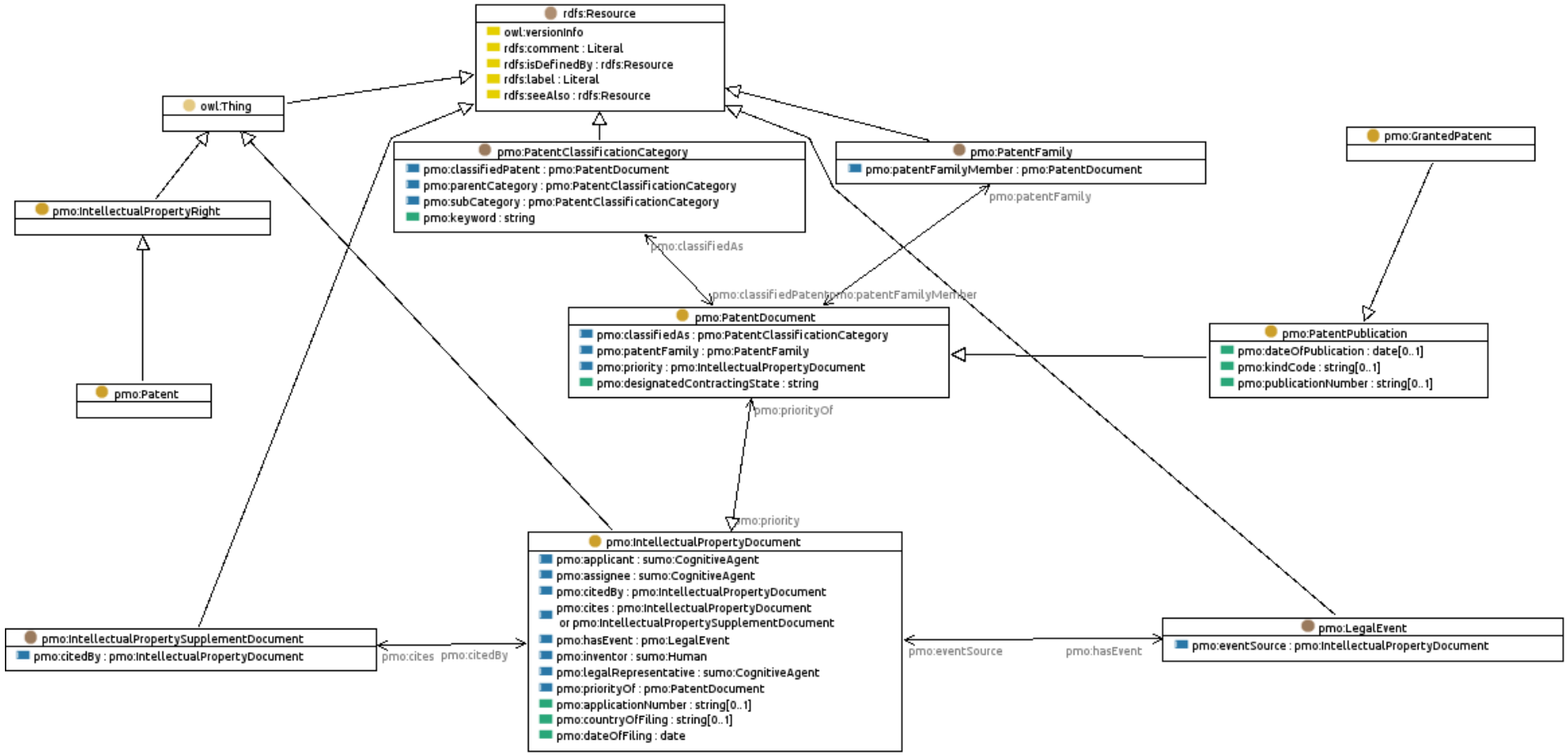

Figure 71: Classes, object properties and data properties in the Patent Metadata Ontology

D Software Ontology License Component

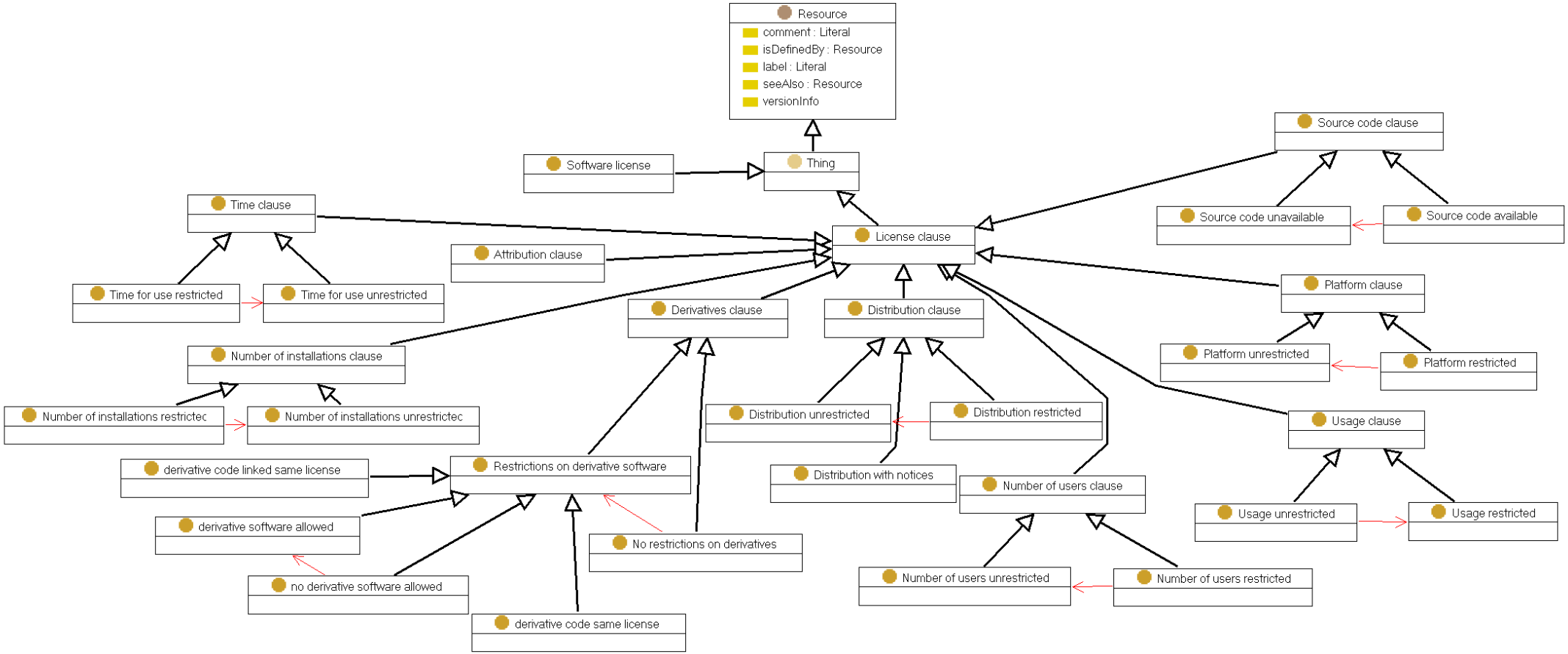

Figure 72: Structure of the "Software Ontology" license component

E Sensors Ontology

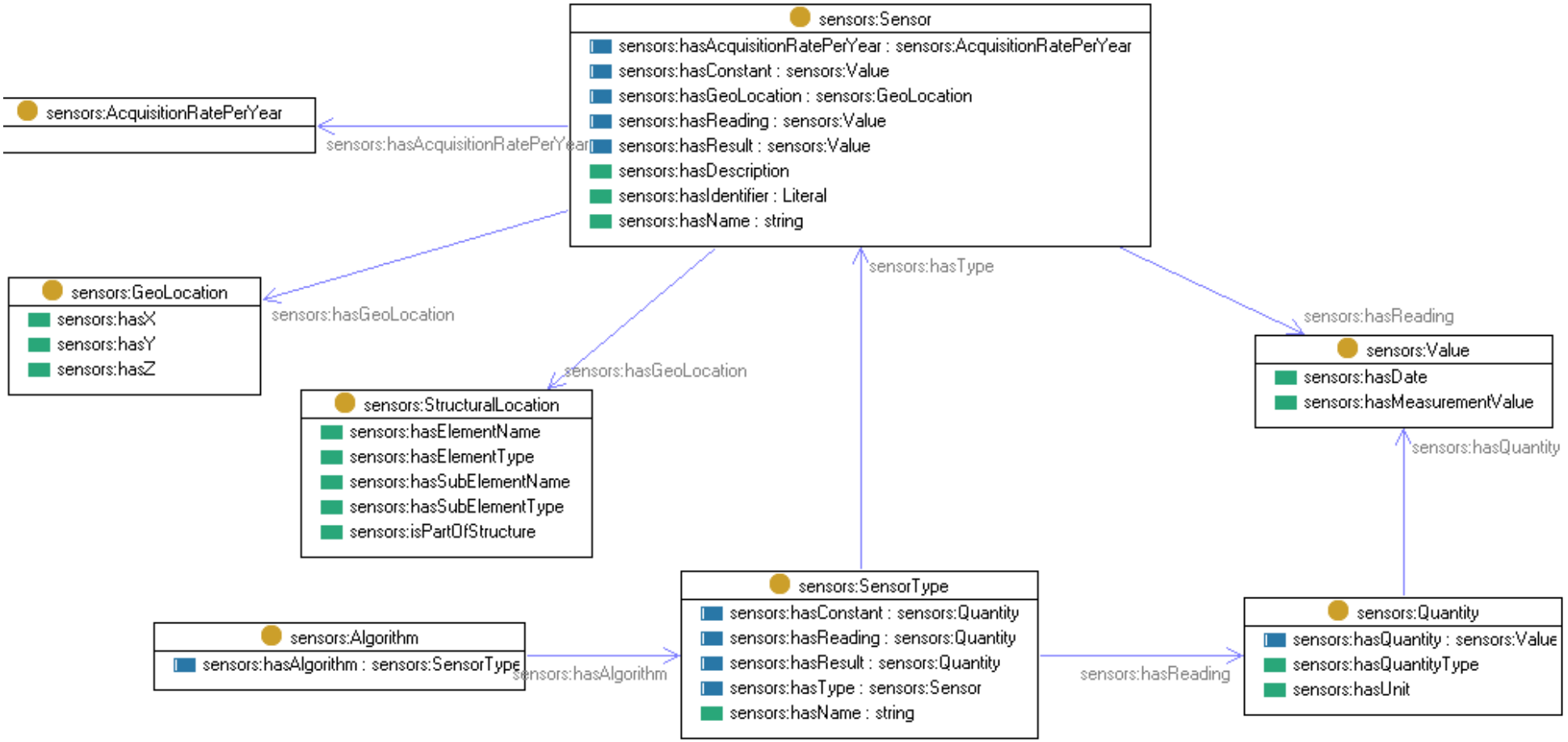

Figure 73: Structure of the Sensors Ontology

F Music Classification ArchiMate Model

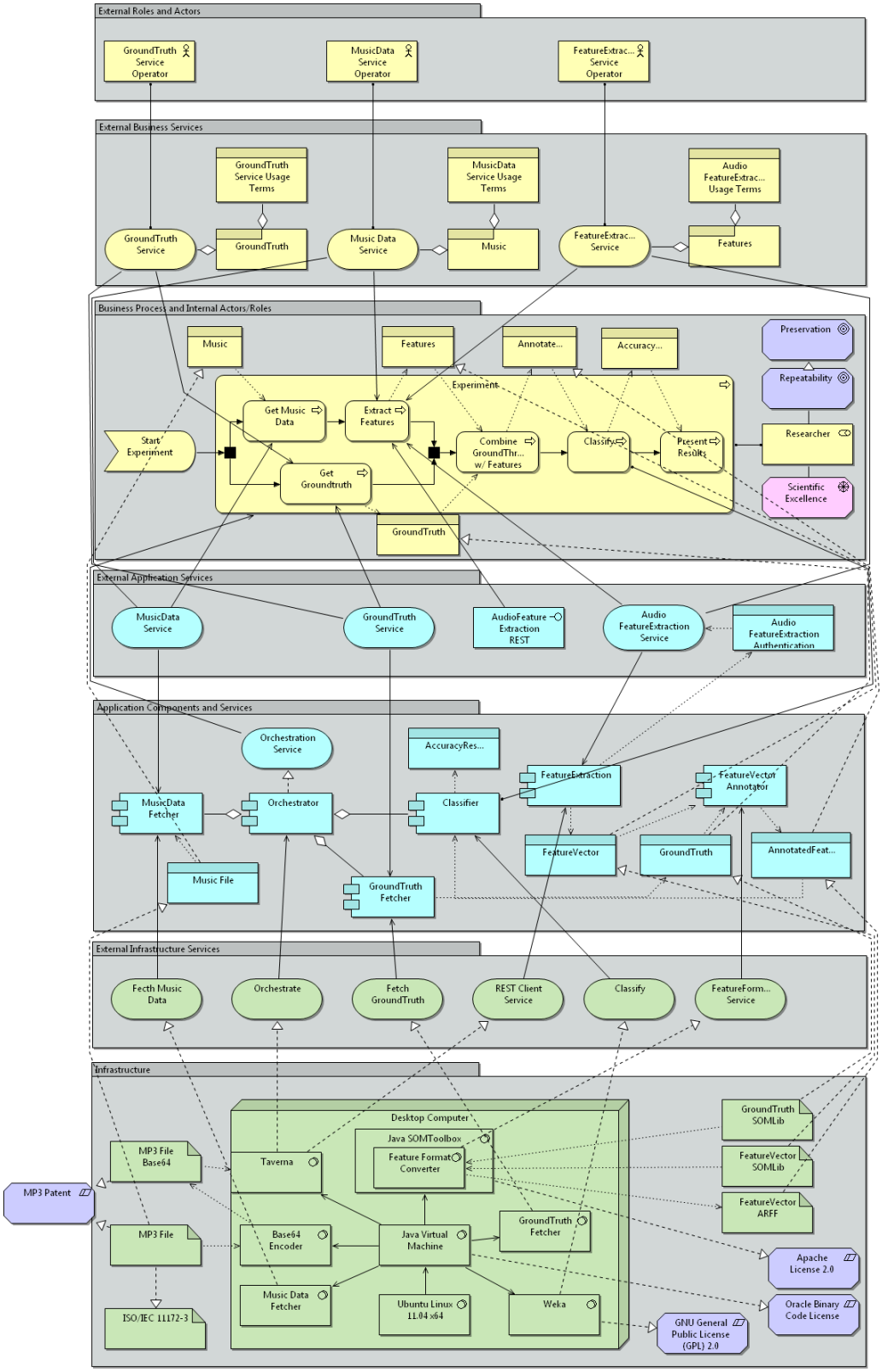

Figure 74: Music Classification Process Layered View

G WP8 ArchiMate Model

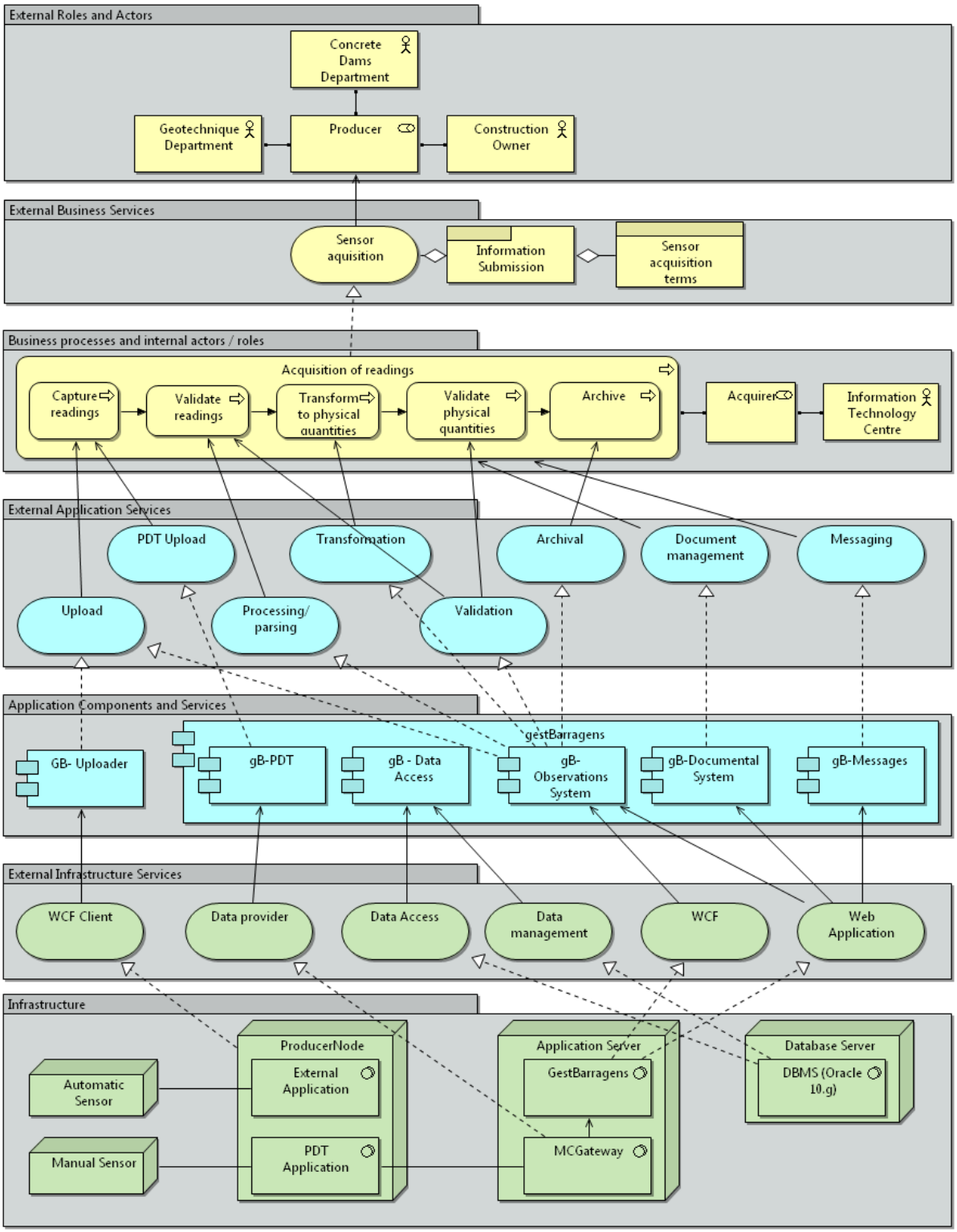

Figure 75: Layered View

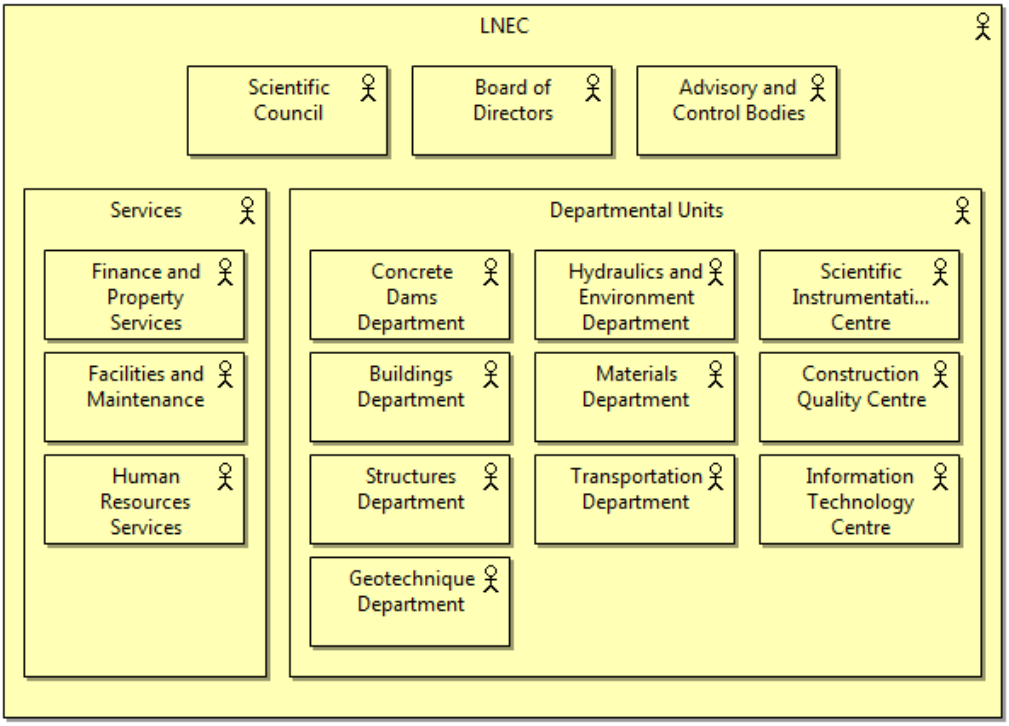

Figure 76: Organisation Structure View

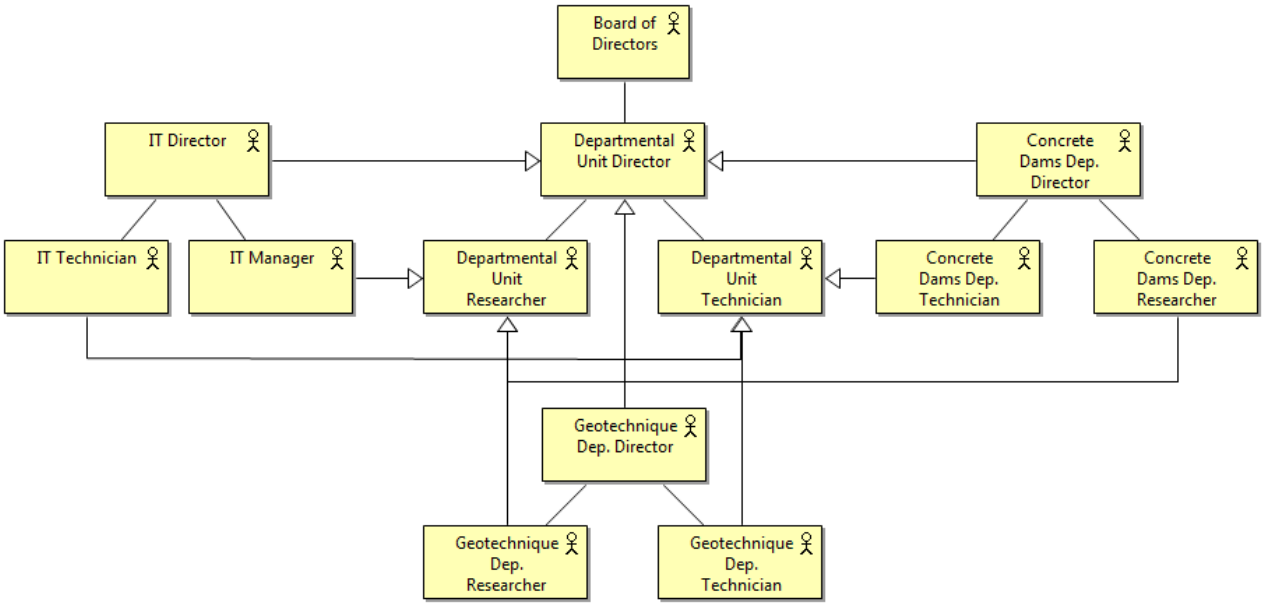

Figure 77: Organisation Tree View

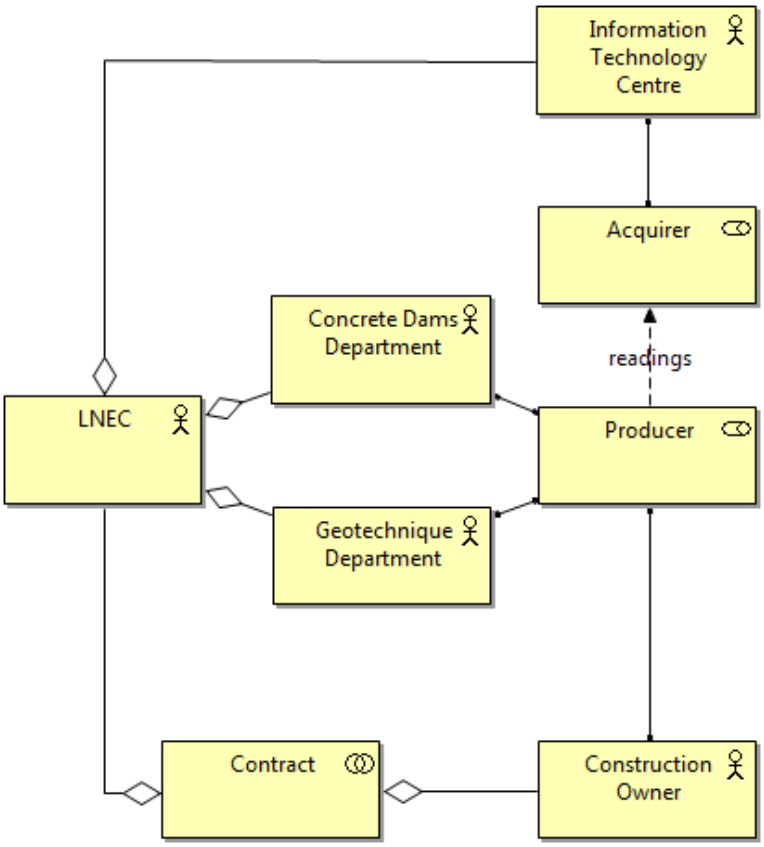

Figure 78: Actor Co-operation View

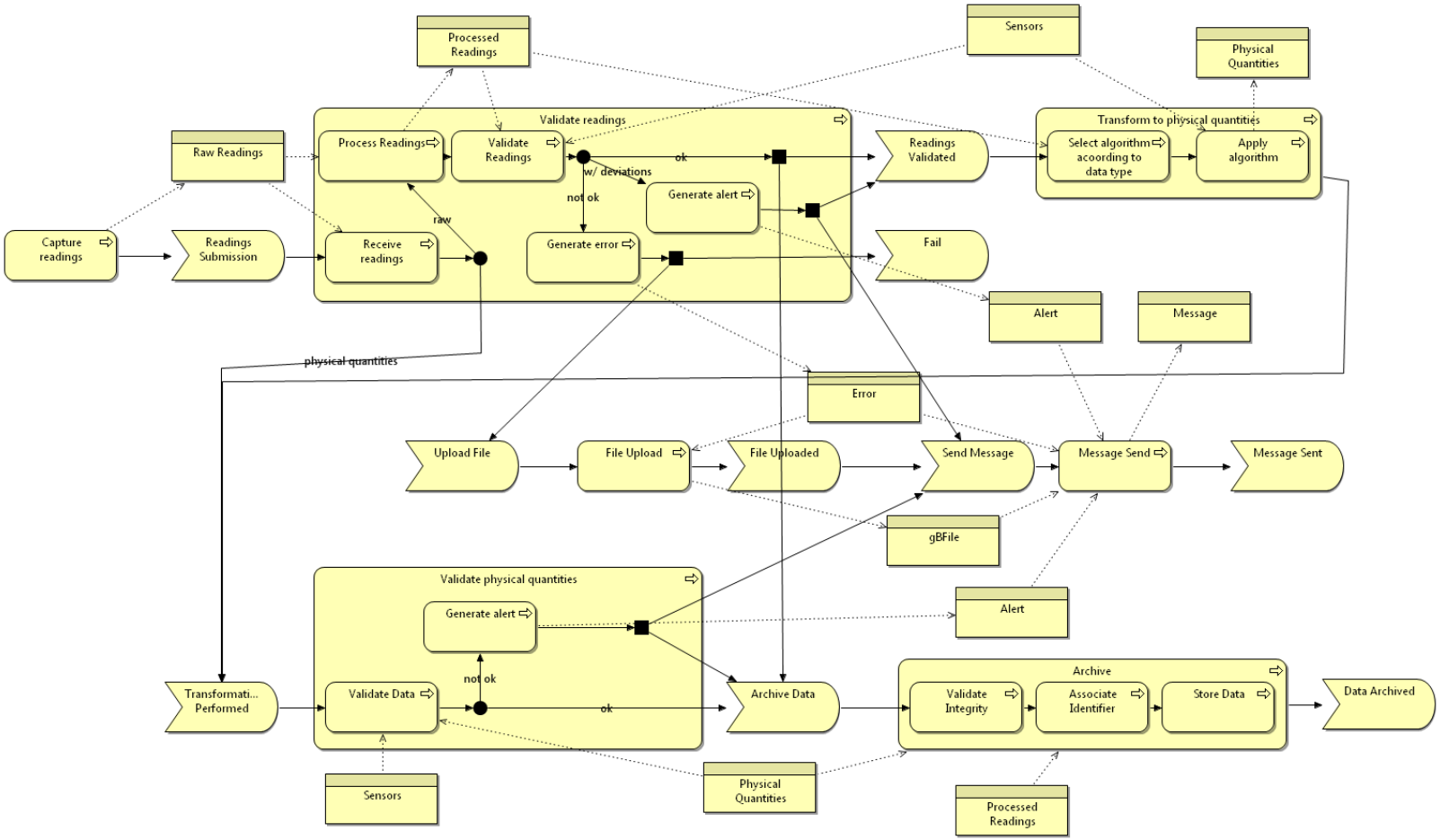

Figure 79: Business Process View

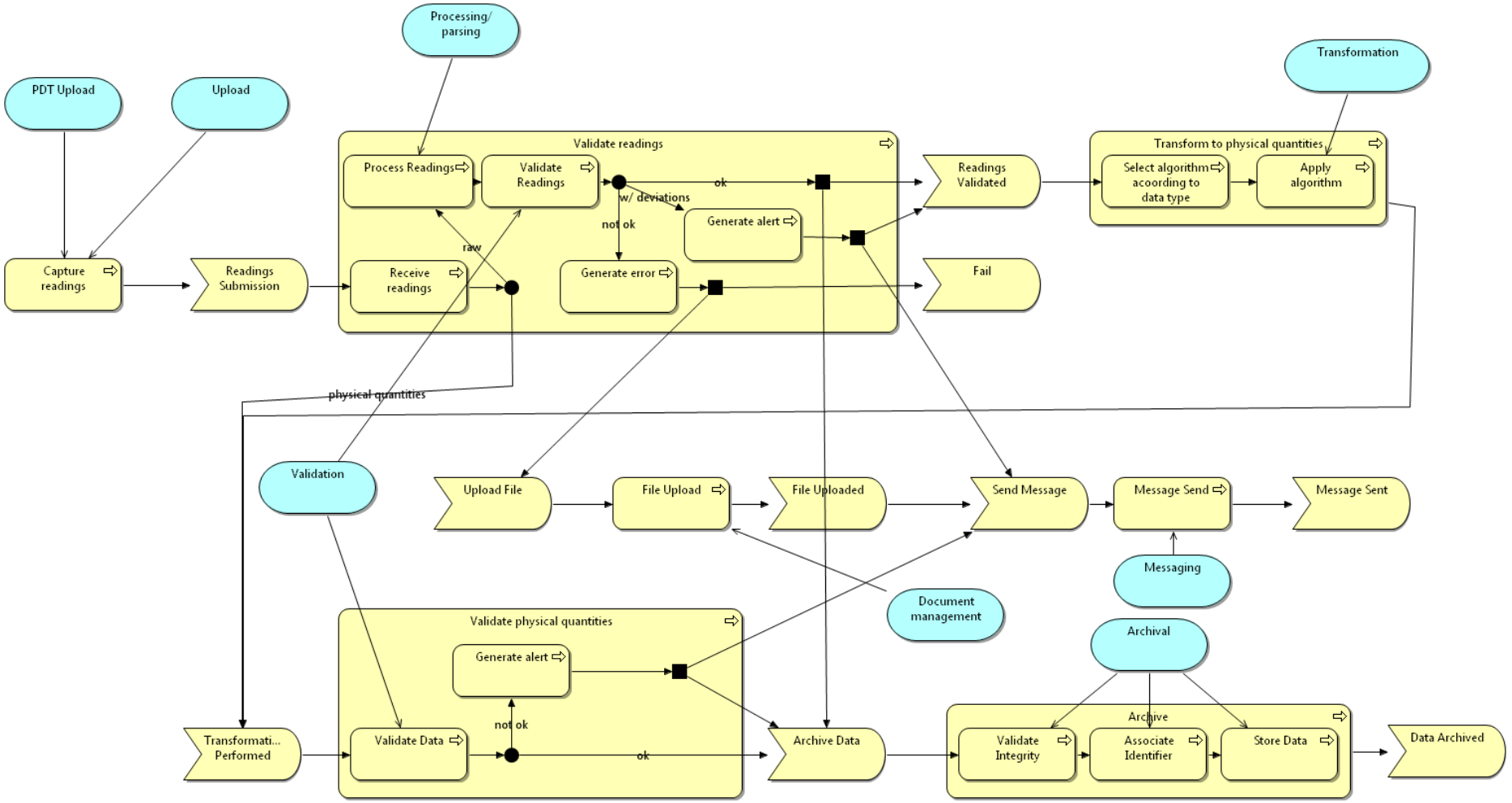

Figure 80: Application Usage View

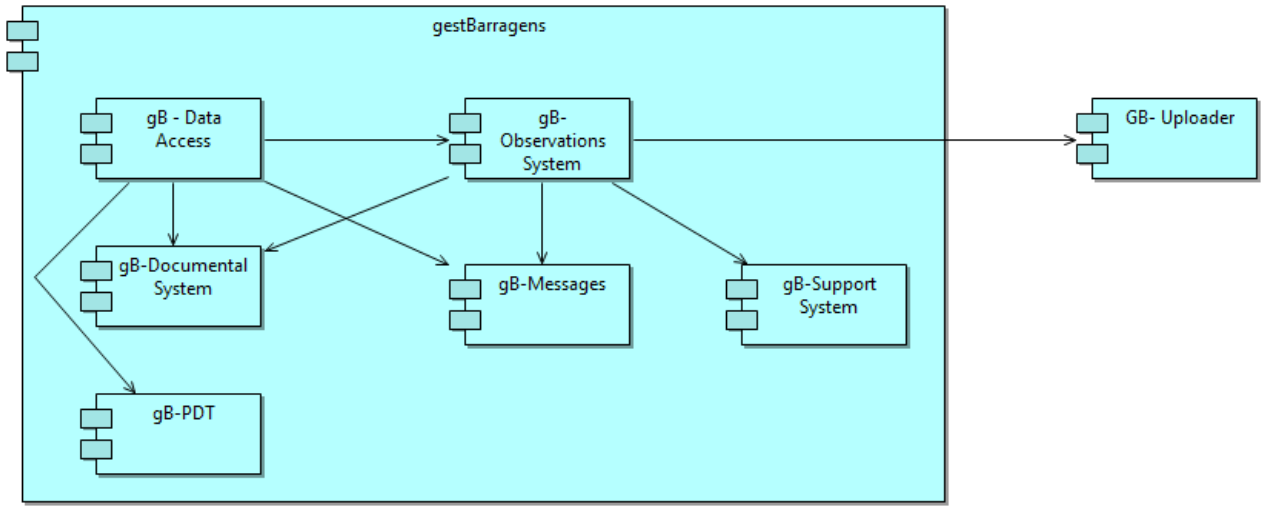

Figure 81: Application Cooperation View

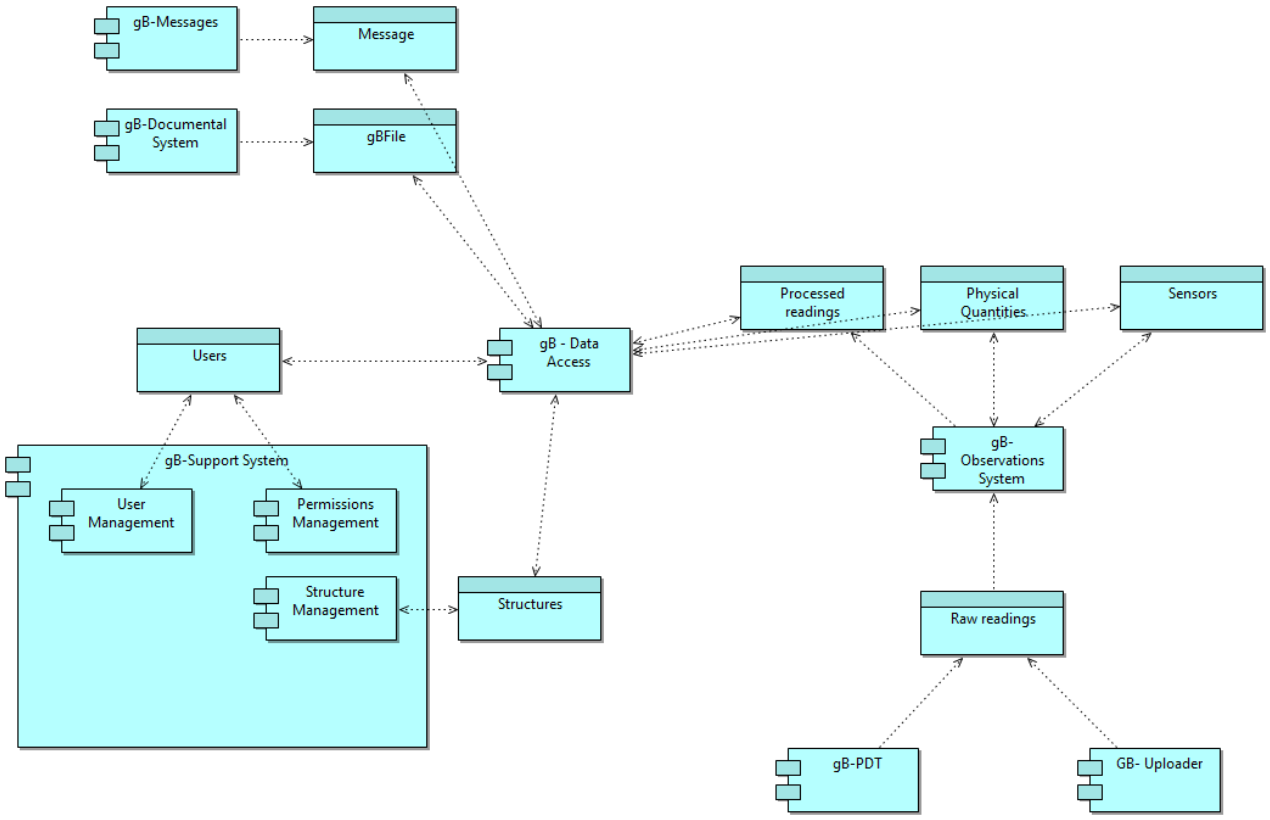

Figure 82: Application Structure View

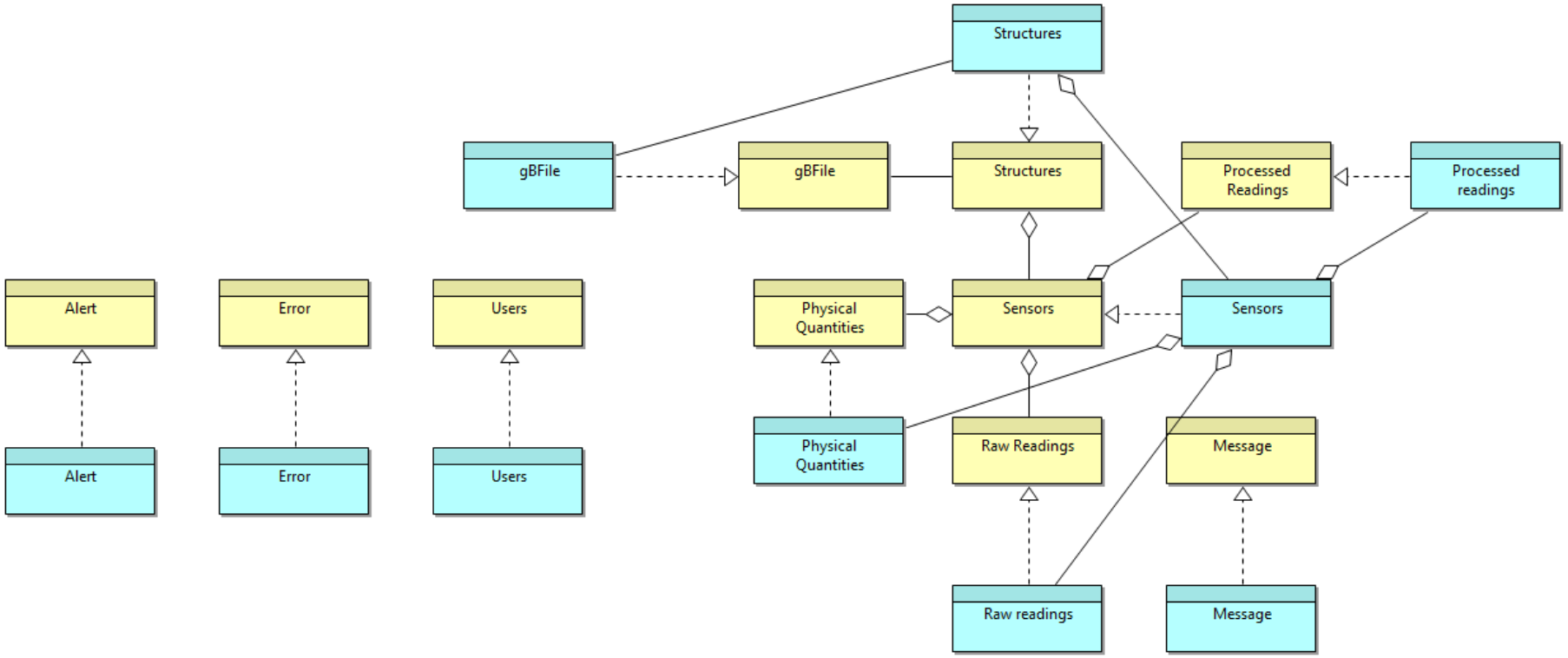

Figure 83: Information Structure View

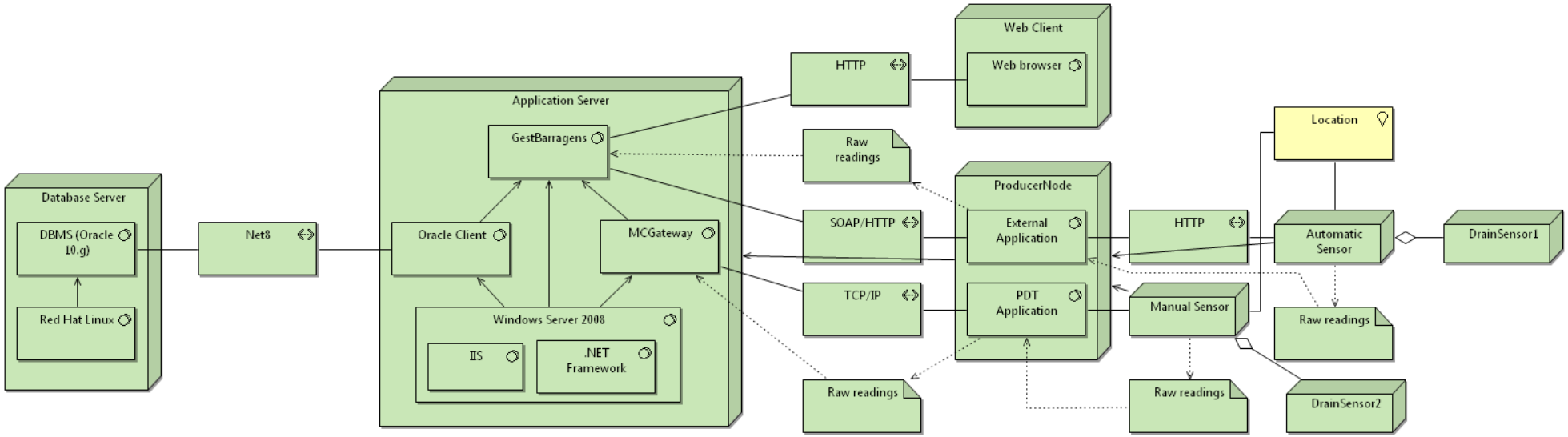

Figure 84: Infrastructure View

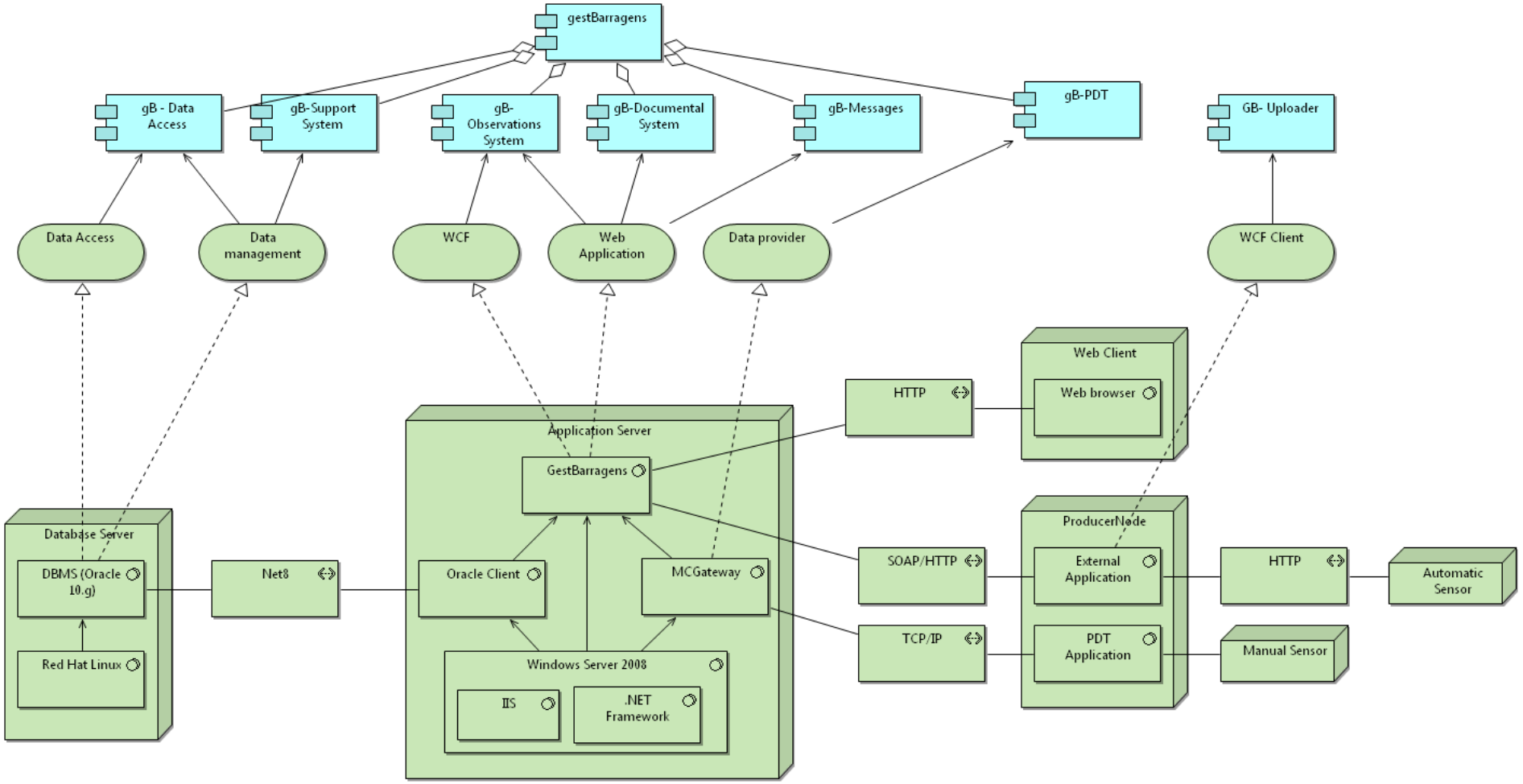

Figure 85: Infrastructure Usage View

H WP9 ArchiMate Model

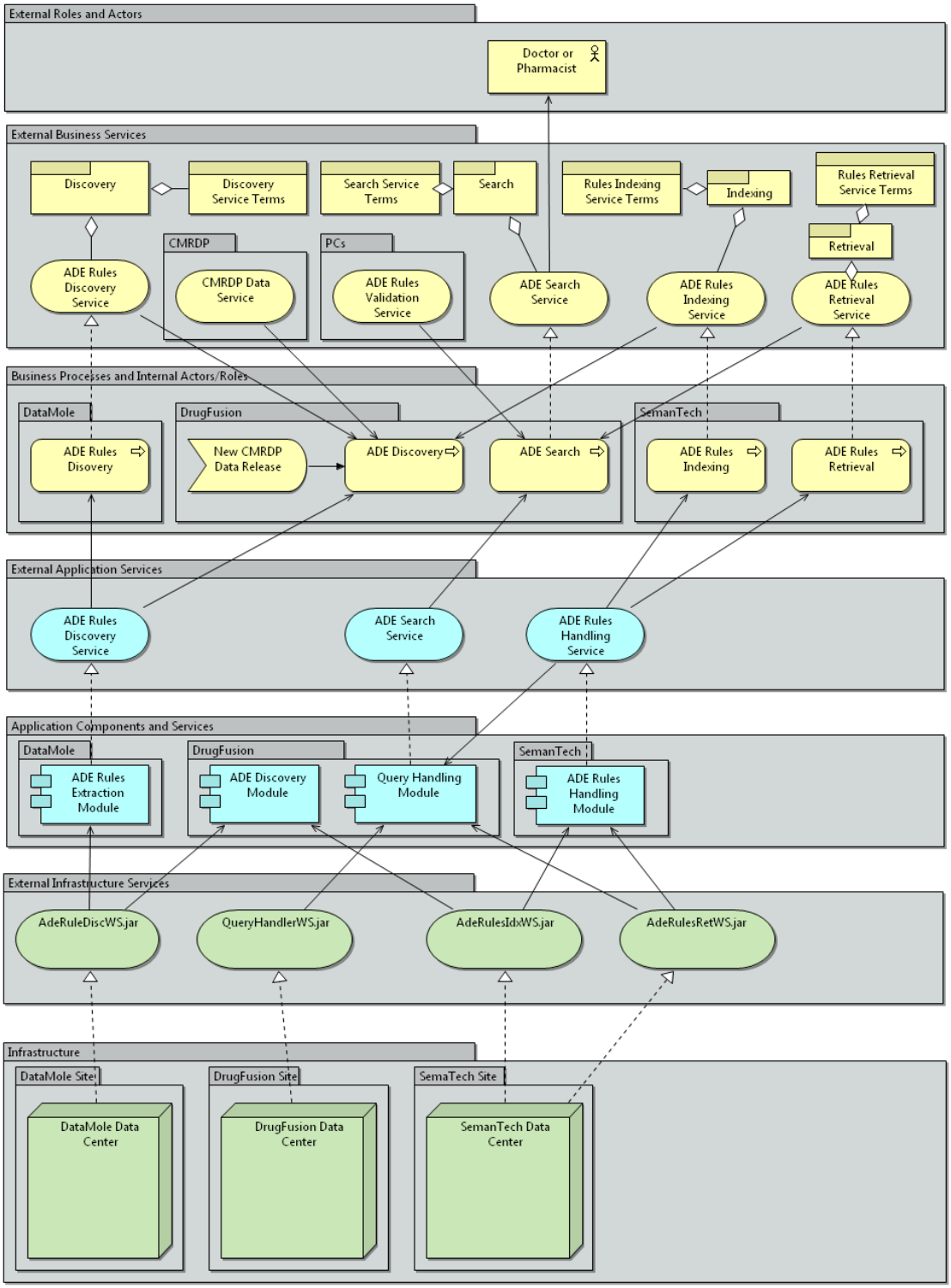

Figure 86: WP9 Layered View

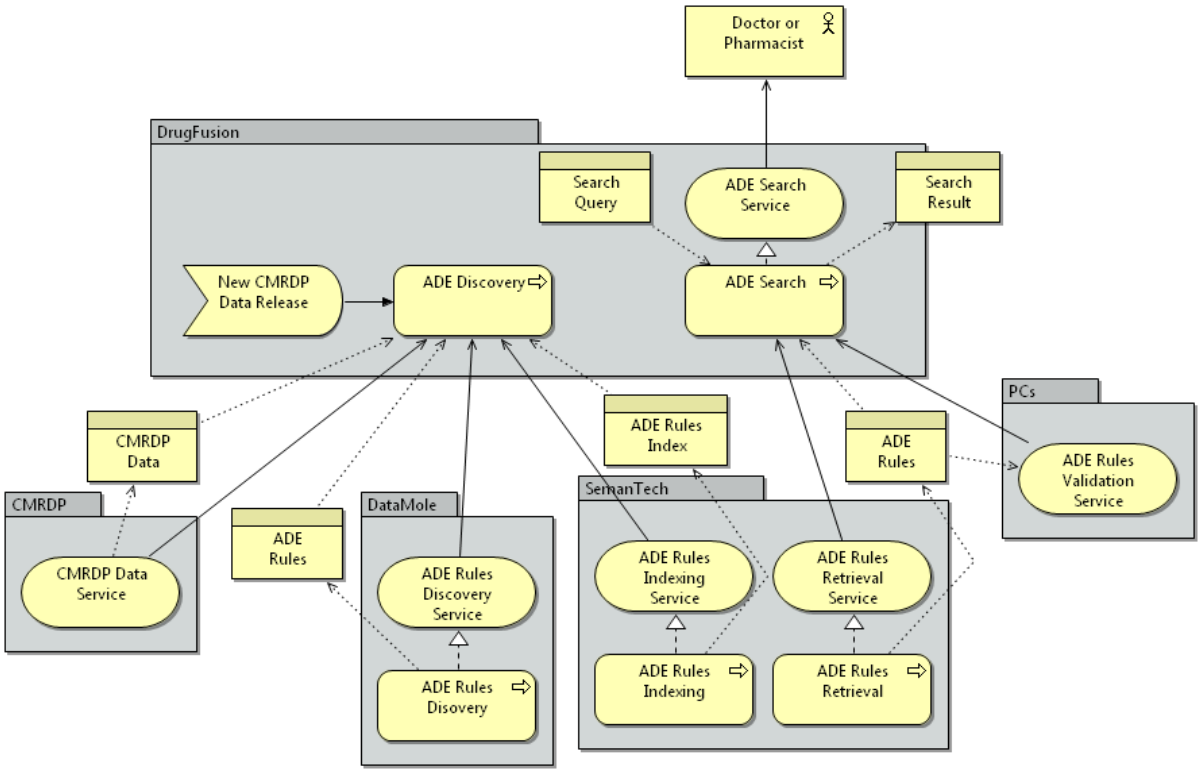

Figure 87: Business Process Co-operation View

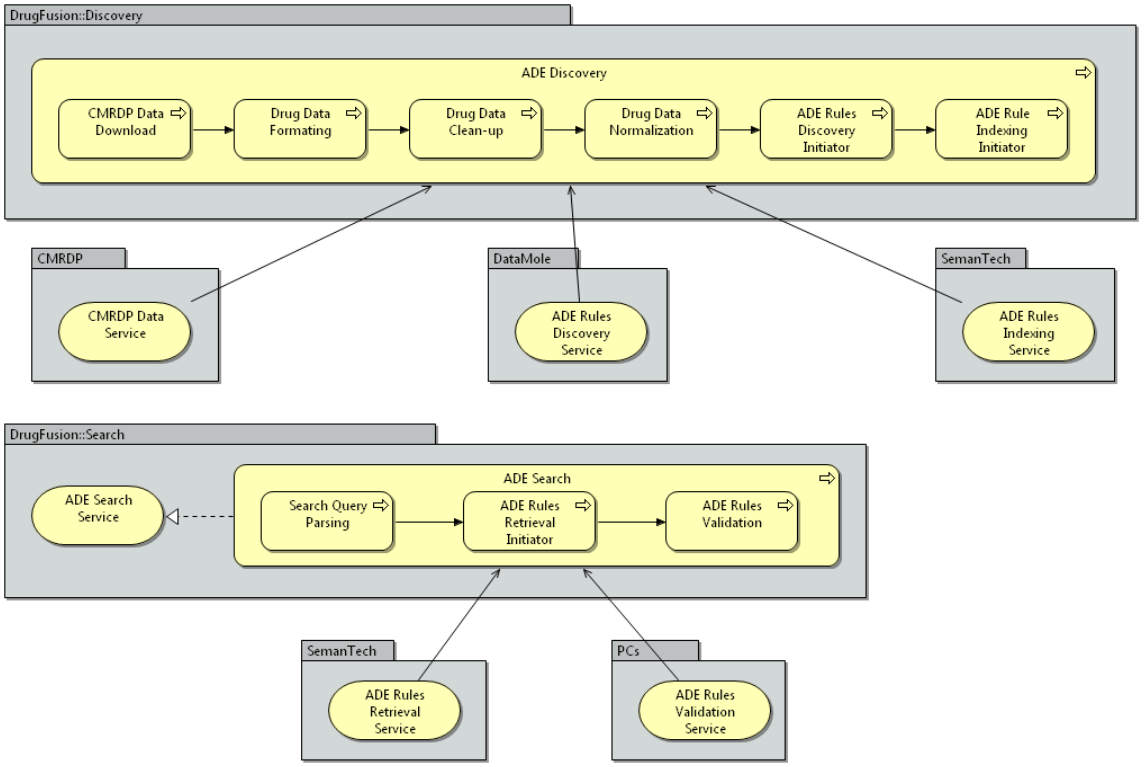

Figure 88: Business Process View (DrugFusion)

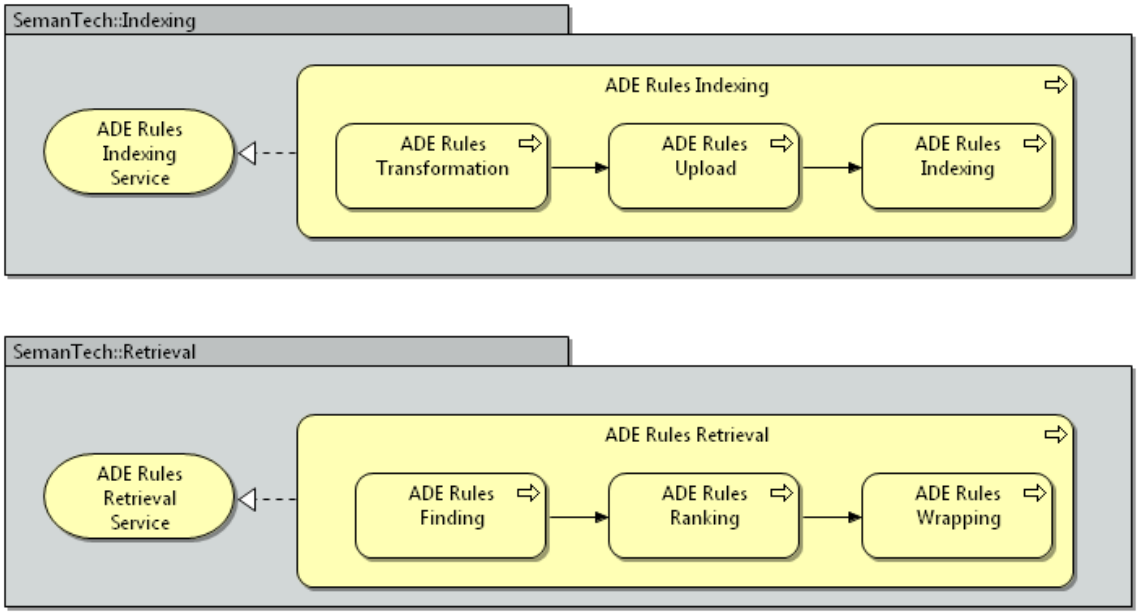

Figure 89: Business Process View (SemanTech)

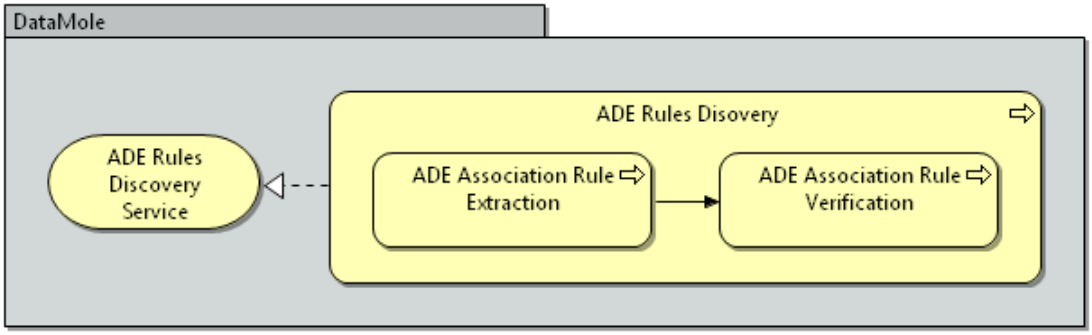

Figure 90: Business Process View (DataMole)

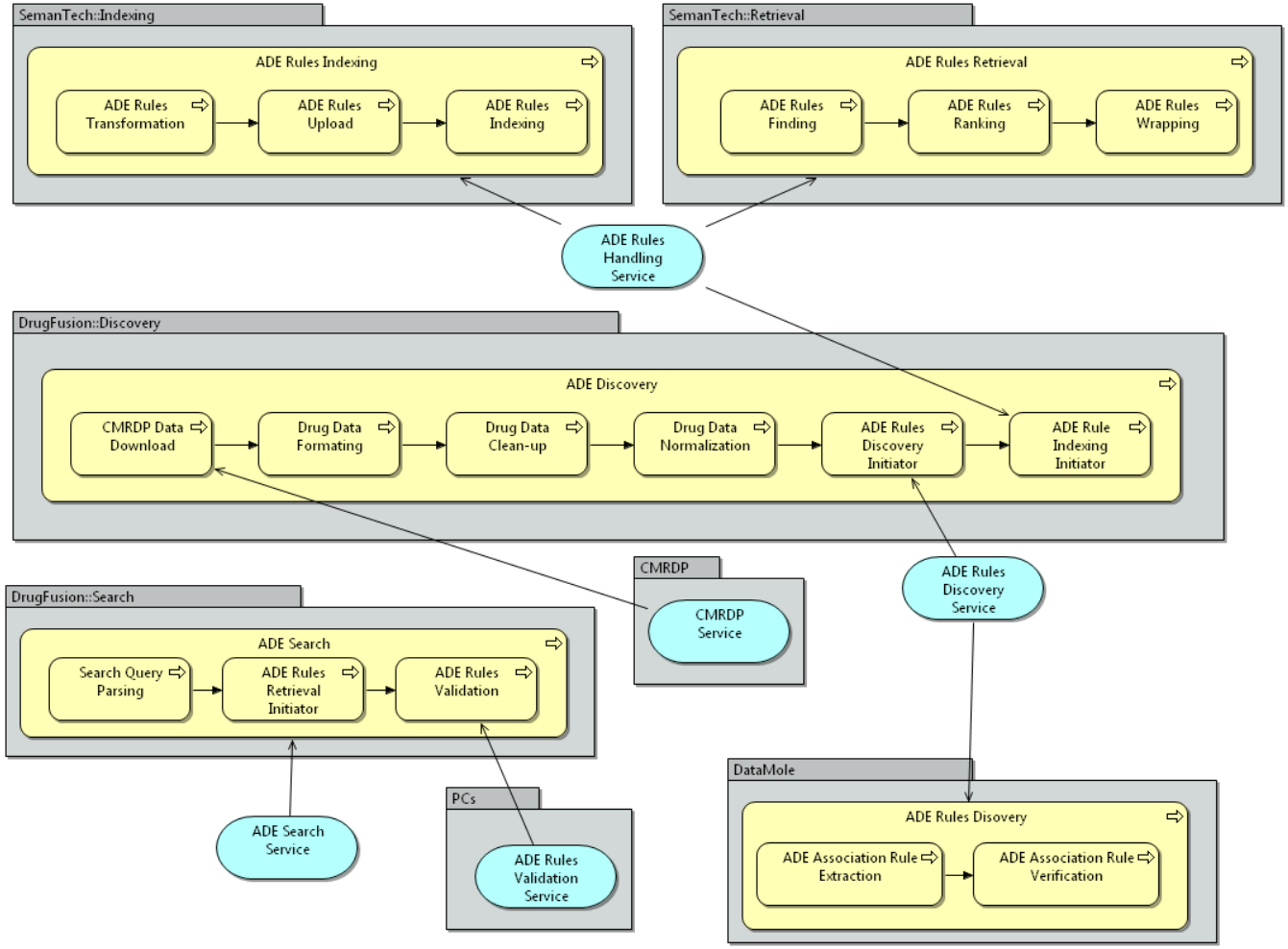

Figure 91: Application Usage View

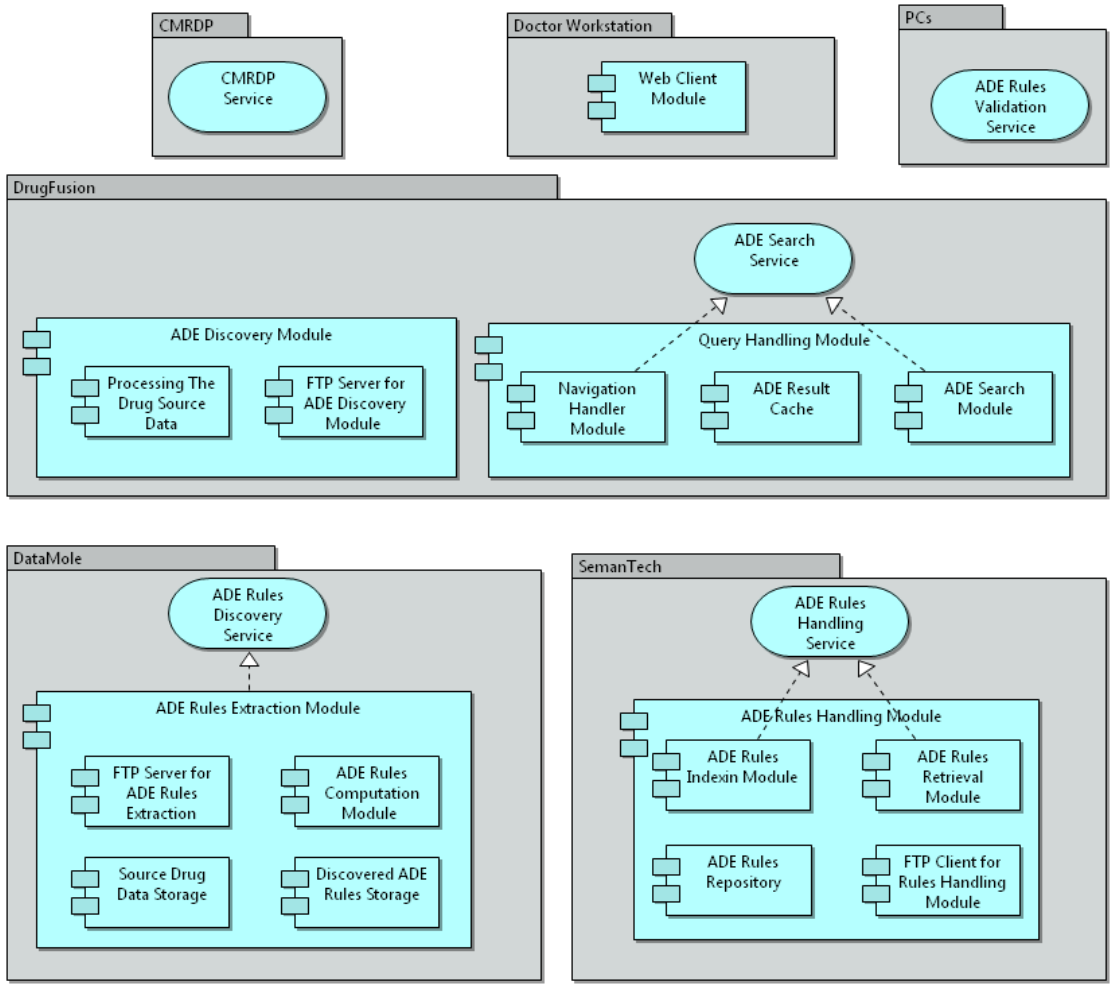

Figure 92: Application Cooperation View

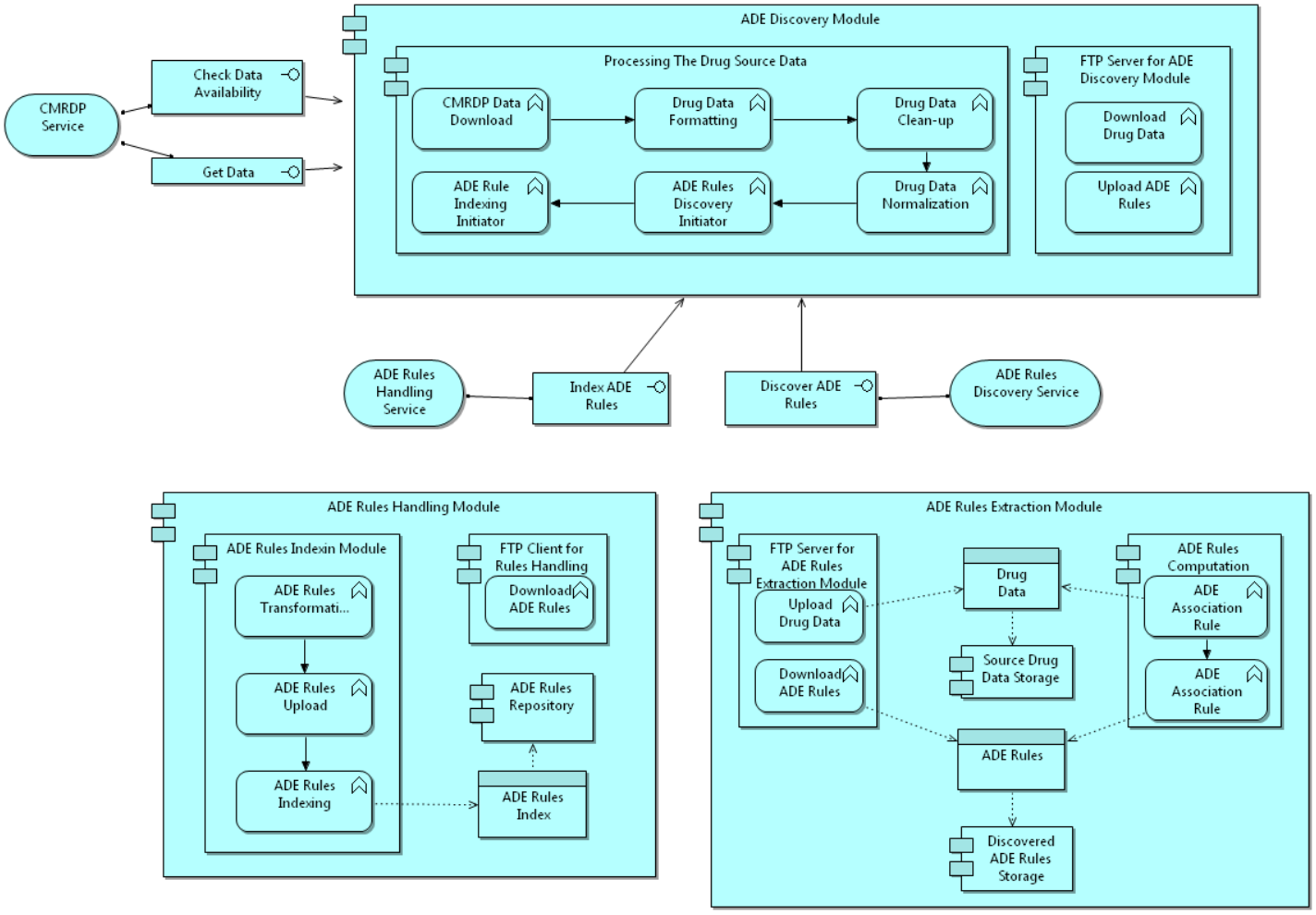

Figure 93: Application Behaviour View (Discovery)

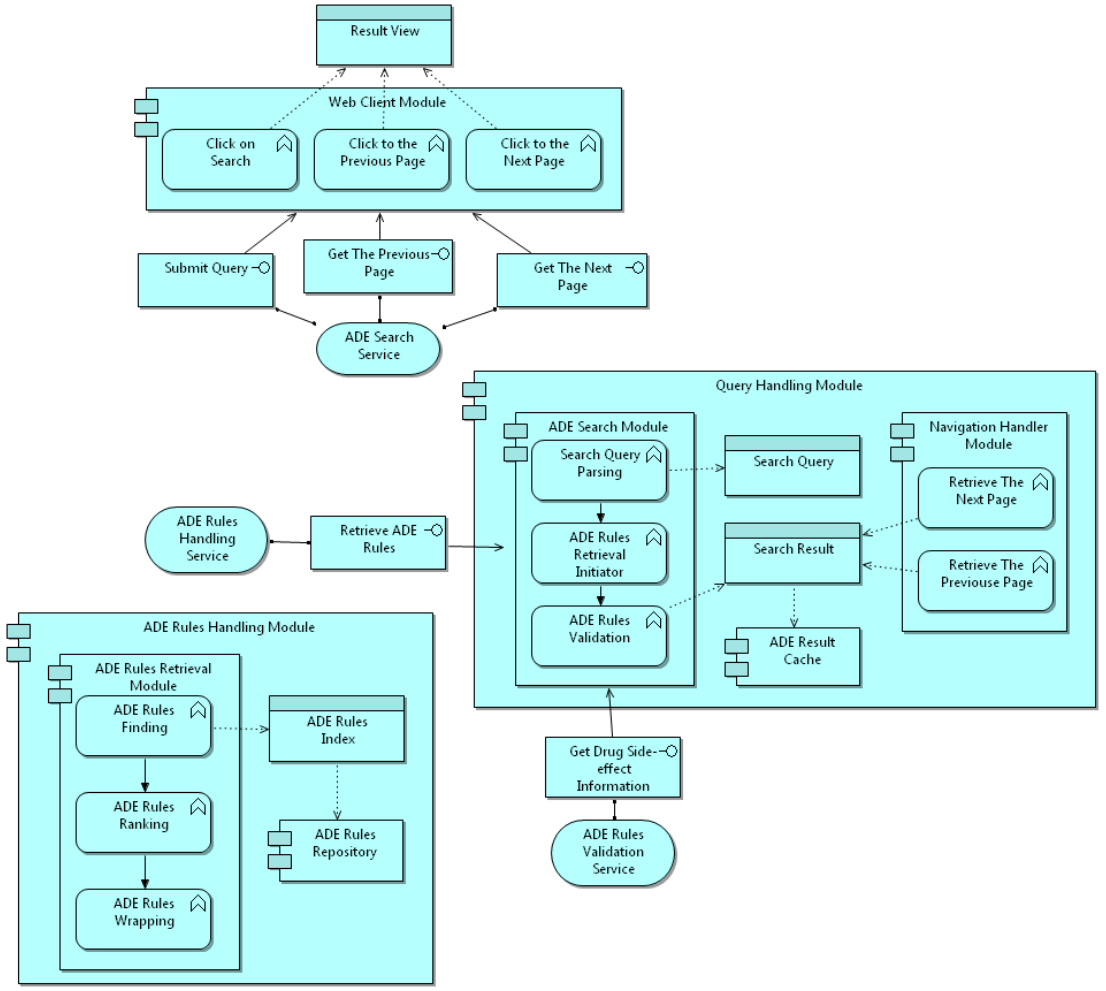

Figure 94: Application Behaviour View (Search)

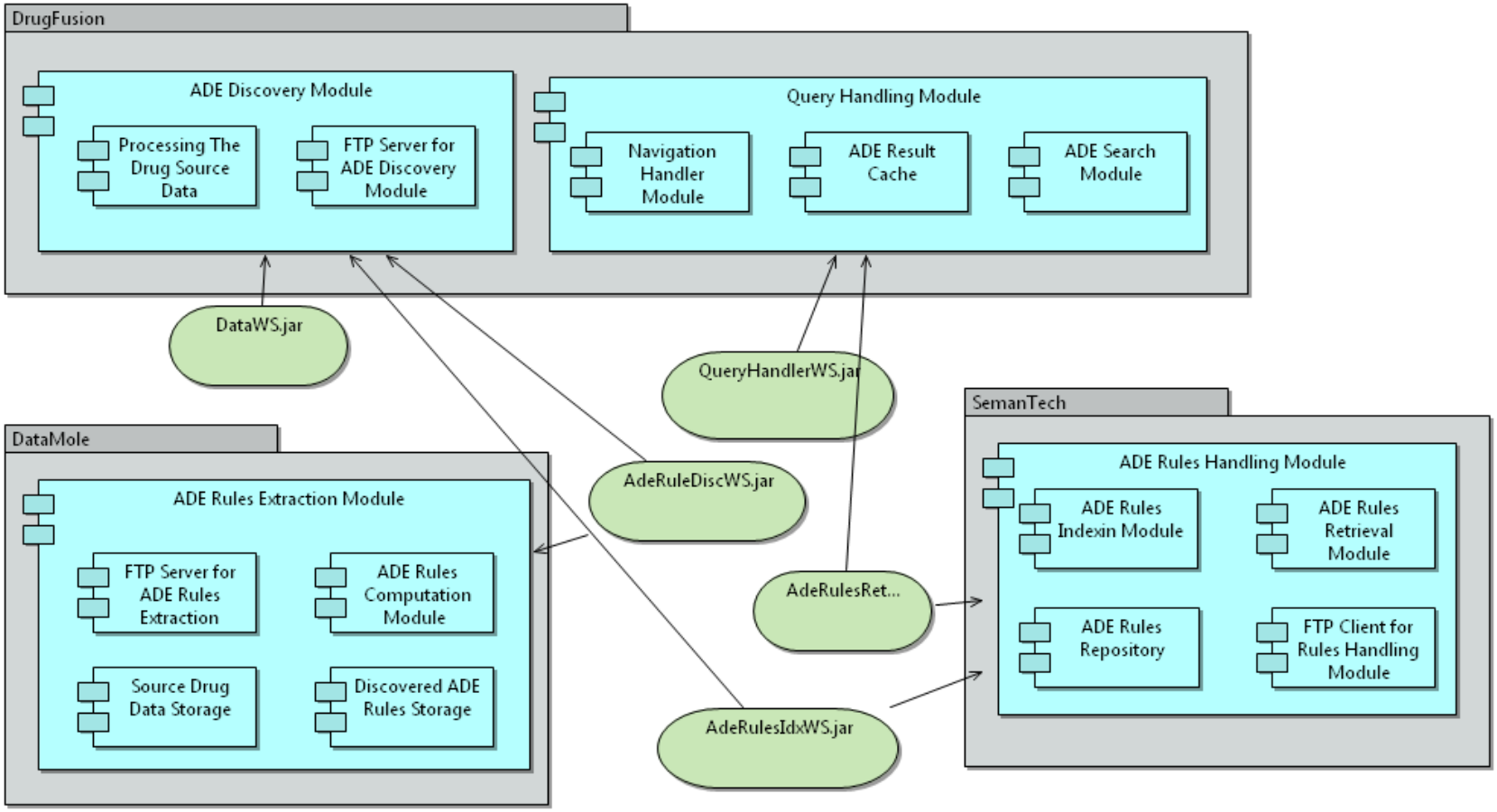

Figure 95: Infrastructure Usage View

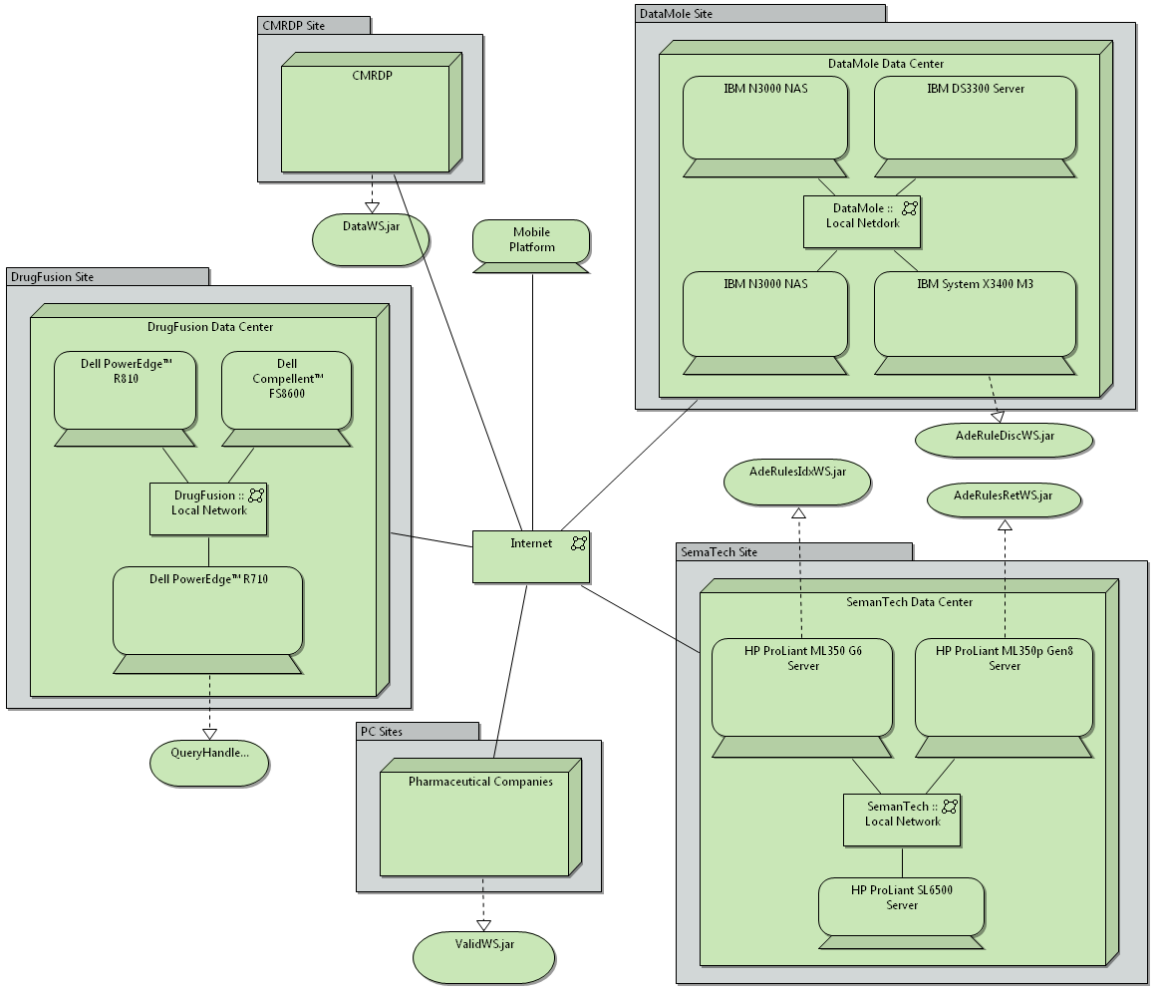

Figure 96: Infrastructure View (General)

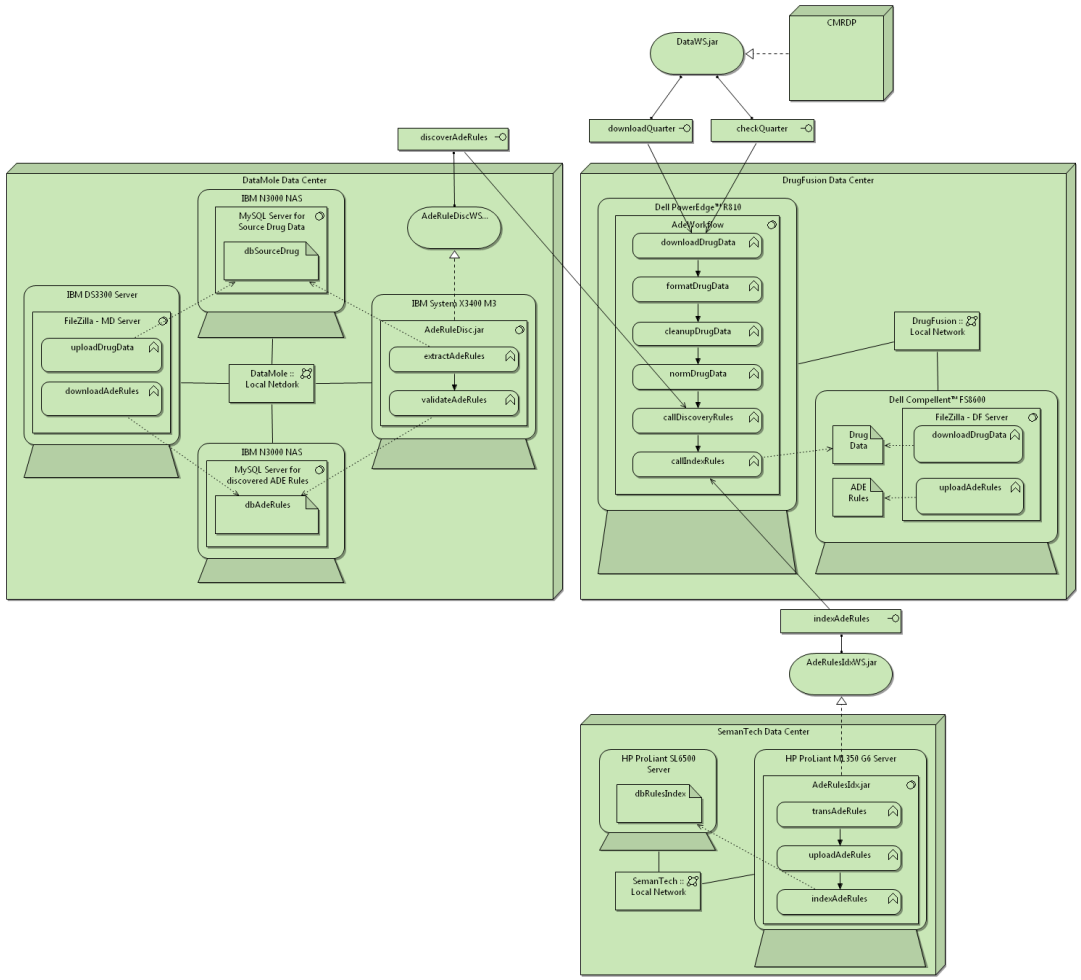

Figure 97: Infrastructure View (Discovery)

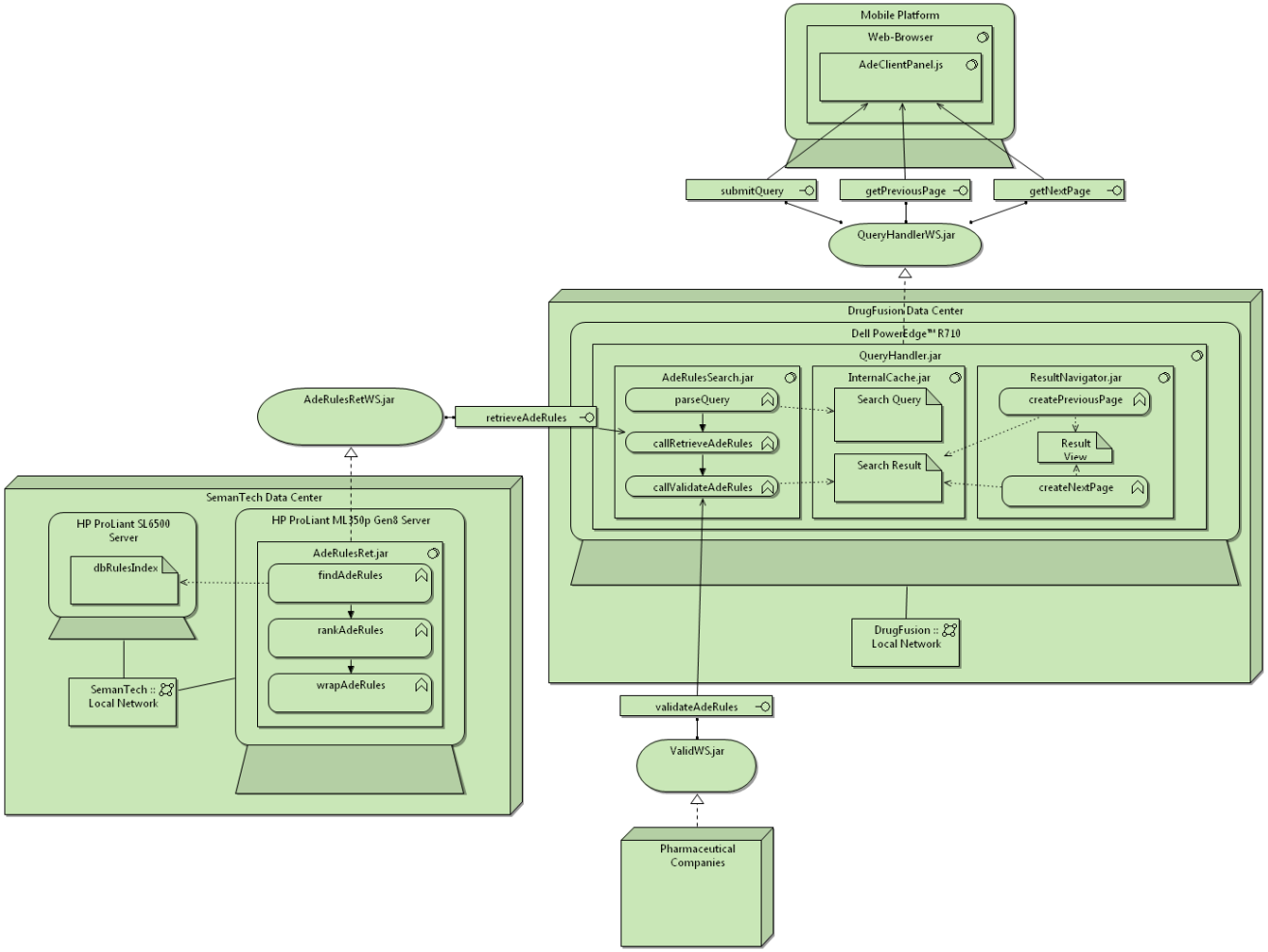

Figure 98: Infrastructure View (Search)

References

1. Bunge, M. A. (1977) Treatise on Basic Philosophy Volume 3: Ontology I — The Furniture of the World, Dordrecht, The Netherlands, Kluwer Academic Publishers.
2. CCSDS (2002) Reference Model for an Open Archival Information System (OAIS), Blue Book.
3. Conway, E., Mattheus, B., Giaretta, D., Lambert, S., Draper, N. and Wilson, M. (2011) Managing Risks in the Preservation of Research Data with Preservation Networks, In the 7th International Digital Curation Conference, 2011.
4. FIPA (2002) FIPA Device Ontology Specification, Standard SC00091E, 2002/12/03.
5. Fox M., Barbuceanu M., Gruninger M., and Lin, J. (1997) An Organisation Ontology for Enterprise Modeling. University of Toronto. Canada.
6. Guizzardi, G. (2005) Ontological Foundations of Information Systems, Enschede, The Netherlands, Telematica Institut.
7. Henderson-Sellers, B., and Ralyté, J. (2010) Situational Method Engineering: State-of-the-Art Review, Journal of Universal Computer Science, Vol. 16, no. 3, pp. 424 – 478.
8. Horridge, M. (2011) A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3, The University of Manchester.
9. ISO, IEC and IEEE (2011) ISO/IEC/IEEE 42010:2011 – Systems and Software Engineering – Architecture Description.
10. ITSMO Project (2011) ITSMO IT Service Management Ontology Language specification, http://ontology.it/itsmo/v1/itsmo.html.
11. Kadobayashi, Y. 2010. Toward Measurement and Analysis of Virtualized Infrastructure: Scaffolding from an Ontological Perspective, http://www.caida.org/workshops/wide-casfi/1004/slides/wide-casfi1004_ykadobayashi.pdf
12. OMG (2007) OMG Unified Model Language (OMGUML) Superstructure, Version 2.4.1.
13. OMG (2009) Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS), OMG Adopted Specification, Finalisation Task Force Beta 2 document (FTF Beta 2).
14. OMG (2011) Business Process Model and Notation (BPMN) Version 2.0, OMG Standard, formal/2011-01-03, 2011.
15. PREMIS Editorial Committee (2012) PREMIS Data Dictionary for Preservation Metadata, version 2.2, July 2012.

16. Rosemann, M., Green, P., and Indulska, M. (2004) A Reference Methodology for Conducting Ontological Analyses, Proceedings of the 23rd International Conference on Conceptual Modelling (ER 2004), Shanghai, China.
17. The Open Group (2011) TOGAF Version 9.1, Van Haren Publishing, Netherlands.
18. The Open Group (2012) ArchiMate 2.0 Specification, Van Haren Publishing, Netherlands.
19. Treinen, R., and Zacchiroli, S. (2008) Description of the CUDF Format. CoRR Journal. Vol, abs/0811.3621. EE, http://arxiv.org/abs/0811.3621.
20. Uschold M., King M., Moralee S., and Zorgios Y. (1996), The Enterprise Ontology. AIAI, The University of Edinburgh. United Kingdom
21. Uschold, M. and Gruninger M. (2006) Ontologies: Principles, Methods and Applications, Knowledge Engineering Review, Vol. 11, pp 93 – 136.
22. Weber, R. (1997) Ontological Foundations of Information Systems, Coopers and Lybrand and the Accounting Association of Australia and New Zealand, Melbourne, Australia.
23. Wierda, G. (2012) Mastering ArchiMate, R&A.
24. W3C (2012) OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation 11 December 2012.
25. Zachman, J. (1987) A Framework for Information Systems Architecture, IBM Systems Journal, vol. 12, pp. 276 – 292.