ICT-2011-288048

EURECA

Enabling information re-Use by linking clinical Research and CAre

IP
Contract Nr: 288048

Deliverable: 4.4 Initial prototype of the semantic interoperability framework

Due date of deliverable: (10-01-2014)
Actual submission date: (10-27-2014)

Start date of Project: 01 February 2012 Duration: 42 months

Responsible WP: UPM

Revision: <accepted>

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Service	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (excluding the Commission Services)	

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Sergio Paraíso Medina	UPM	sparaiso@infomed.dia.fi.upm.es
Santiago Aso Lete	UPM	saso@infomed.dia.fi.upm.es
David Perez del Rey	UPM	dperez@infomed.dia.fi.upm.es
Raúl Alonso Calvo	UPM	ralonso@infomed.dia.fi.upm.es
Kristof De Schepper	Custodix	kristof.deschepper@custodix.com
Vassilina Nikoulina	Xerox	vassilina.nikoulina@xrce.xerox.com

0.2 Documents history

Document version #	Date	Change
V0.1	24.06.2014	Starting version, template
V0.2	04.07.2014	Definition of ToC
V0.3	22.09.2014	First complete draft
V0.4	22.09.2014	Integrated version (send to WP members)
V0.5	23.09.2014	Updated version (send PCP)
V0.6	24.09.2014	Updated version (send to project internal reviewers)
V0.7	20.10.2014	Reviewed version
Sign off	25.10.2014	Signed off version (for approval to PMT members)
V1.0	27.10.2014	Approved Version to be submitted to EU

0.3 Document data

Keywords	Semantic interoperability, core dataset, services
Editor Address data	Name: Sergio Paraíso Medina Partner: UPM Address: Facultad de Informática Universidad Politécnica de Madrid Campus de Montegancedo, s/n 28660 Boadilla del Monte, Madrid, Spain Phone: +34 91 336 74 45 E-mail: sparaiso@infomed.dia.fi.upm.es
Delivery date	

0.4 Distribution list

Date	Issue	E-mailer
27.10.2014	V1.0	fp7-eureca-wp4@listas.fi.upm.es

Table of Contents

Table of Figures

1 Introduction

This deliverable reports on the initial prototype of the semantic interoperability layer within the EURECA project. The main goal is to provide homogeneous access to EURECA services, enabling linkage between the patient data in the EHR and the clinical trial systems.

The main functionality of the EURECA semantic interoperability layer is to provide access to the clinical information (see chapter 2). For that purpose, a semantic interoperability approach has been proposed to provide an endpoint to store and retrieve clinical data. The semantic layer provides mechanisms and facilitates concept mapping and HL7 messages construction using a shared medical vocabulary. The importing process of clinical data into the clinical data warehouse is executed by an open source ETL tool.

In order to enable data retrieval from the platform, a set of components and services (Common Data Model, SPARQL endpoint and query templates) has been defined. The platform provides an user friendly tool for patient information management across institutions to allow researchers focus on the study of patients. The final objective is to provide a homogenous endpoint to retrieve patient information across institutions.

In the next sections, the semantic interoperability components and services are described and how it is interconnected with the EURECA security layer. Note that this deliverable contains the current state of the EURECA semantic interoperability framework and deliverable 4.6 “Final prototype of the semantic interoperability framework” will contain the final state.

2 Common Information Model

The main goal of the EURECA semantic interoperability framework is to provide homogeneous access to different data sources. For that purpose, the EURECA infrastructure is based on a Common Information Model (CIM) to represent, manage and retrieve clinical data stored in the platform [1]. The proposed model is comprised by two main components as it is showed on *Figure 1*: Core Dataset and Common Data Model (CDM).

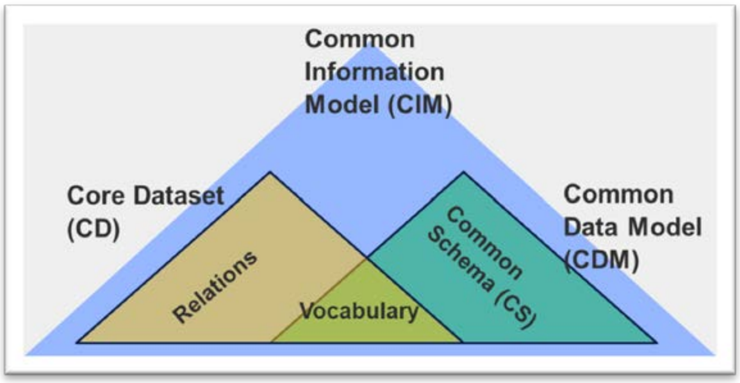


Figure 1: Common Information Model

The Core Dataset is the component that contains the medical vocabulary of the platform, relationships among concepts and tools for inferring semantic knowledge from the ontology. The CDM is a structure that contains the data of the different sources, following a common schema and the shared medical vocabulary of the platform. A set of services for storing, querying and exploiting the possibilities these components has been developed.

2.1 Common Data Model

The Common Data Model (CDM) is the schema of the EURECA semantic interoperability layer and it has to be able to homogeneously store all clinical data provided from different data sources. The HL7 Reference Information Model (RIM) has been selected to develod the EURECA CDM due to the adoption of the interoperability standard and automatic mechanisms available to link to medical vocabularies such as SNOMED CT, HGNC and LOINC. Results from the INTEGRATE project[1] regarding the CDM has been reused according to the Description of Work, and described in detail in deliverable 9.2, “Canonical models of EHR and CT systems” . In addition, a comparision among interoperability standards used in the EURECA implementation can be found in deliverable 2.1 “State of the art report on standards” [18] and deliverable 1.3 “Report on state of art on relevant knowledge and data sources and on reusable tools” [2].

2.1.1 RELATIONAL MODEL BASED ON HL7 RIM

As it was detailed on EURECA deliverable 9.2, “Canonical models of EHR and CT systems” [19] the HL7 v3 standard “*is designed to be comprehensive in the scope, complete in detail, extensible and model-based conformance testable and technology*

[1] http://www.fp7-integrate.eu/

independent. It tries to handle health-care communications in an unambiguous way" [3]. The Reference Information Model (RIM) is the core element of HL7 v3 standard; it is intended to represent almost any medical situation, and any kind of information associated with it. However, the RIM is not a definition of a data structure or database model, it is a class diagram that needs to be modeled to construct a relational database.

Since the HL7 RIM is a model for representing any health care situation and that it is not a database model, not all of its classes have to be modeled for the project environment, neither all their attributes. In this representation there are 3 main classes; *Act*, *Role* and *Entity*, they are linked using three association classes, and these are *ActRelationship*, *Participation* and *RoleLink*. Additionally, there are more specific subclasses of the main ones, e.g. HL7 RIM defines that one instance of the *Act* class (defined as "a record of an event that has happened or may happen") could be an observation, a substance administration or a procedure. In these cases, these subclasses have specific characteristics because they add specific attributes. The other two main classes (Entity and Role) also have specific subclasses.

Therefore, the selected classes to be modeled as tables in the CDM have been those that are needed to store and share clinical data. These classes are:

- *Act*, with its sub-classes *Procedure*, *Observation* and*SubstanceAdministration*.
- *Role*
- *Entity* and its sub-classes *LivingSubject* and *Person*.

The subclasses are modeled as a specification of the father class. For example, a *SubstanceAdministration* instance requires the existence of another instance with the same *id* in the *Procedure* table and also in *Act* table. Additionally, two relationship classes have been modeled, *Participation* and *ActRelationship*.

Regarding attributes, some of them are defined in the standard as they can store several values. A specific table has been created for storing these attributes. For example, the attribute *interpretationCode* of the *Observation* class could store more than one value. And the table *ActObservationInterpretationCode* stores all the Observations related to an Interpretation. Other similar examples are *methodCode* and *targetSiteCode* attributes; both attributes are located at *Observation* and *Procedure* classes, thus, a unique table for each attribute has been created similar to *interpretationCode*. This allows searching information related with a target site just in one table independently if the action is a procedure or an observation. All the modeled tables that represent attributes are:

- *ActProcedureApproachSiteCode*
- *ActMethodCode*
- *ActTargetSiteCode*
- *ActObservationInterpretationCode*
- *ActObservationValues*

The final structure of the CDM can be seen in the *Figure 2*.

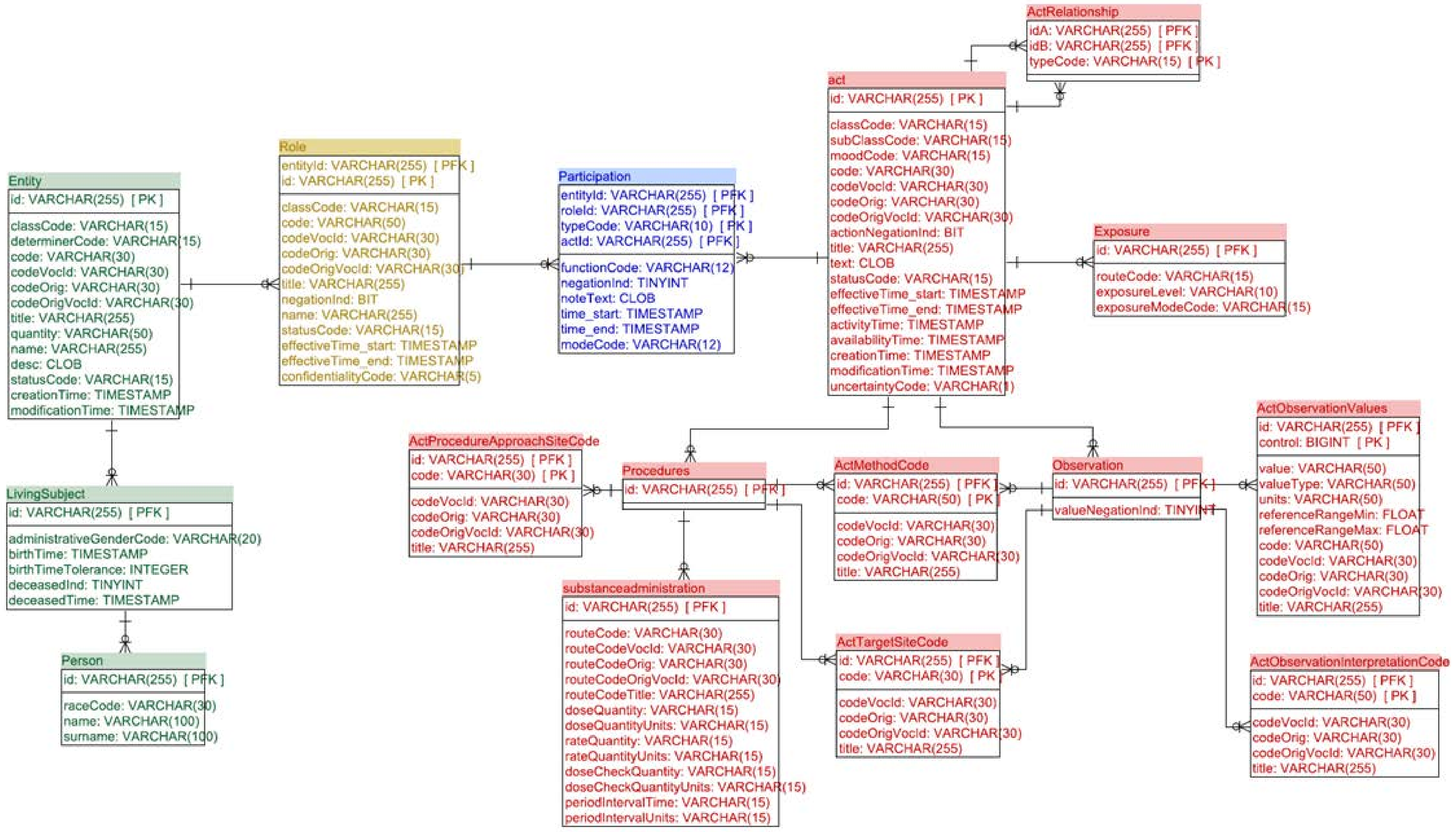


Figure 2: Common Data Model MySQL Scheme 2.6.2

2.1.2 EURECA Guidelines

This section describes the set of technical guidelines required to homogeneously import data from hospitals and research institutions into the CDM. This process is divided in two types of tasks depending on the responsible actor: (i) Data provider or (ii) EURECA platform, as it is showed in *Figure 3*:

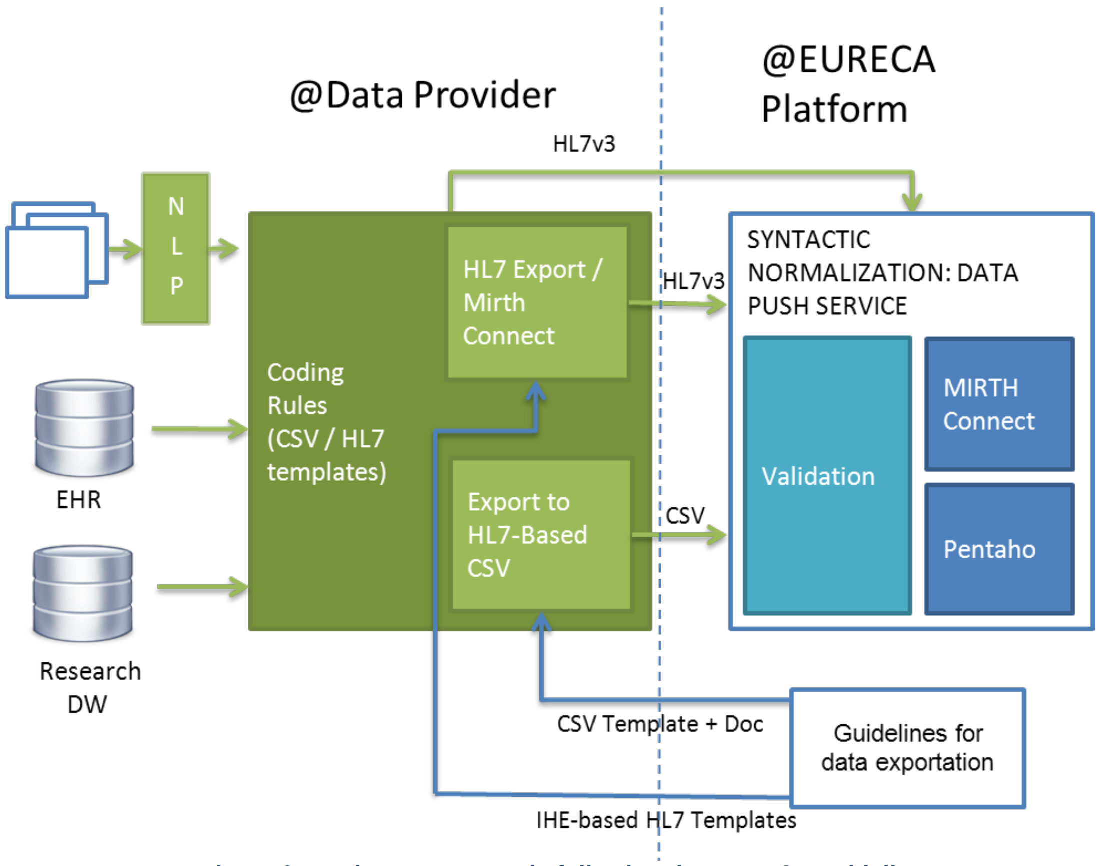


Figure 3: Deployment example following the EURECA guidelines

The first step of this process is the data annotation using standard medical vocabularies of the Core Dataset. Once data is annotated, it is necessary to generate HL7 v3 messages following a group of templates previously defined. Such templates follow IHE recommendations [20] regarding HL7 message generation.

The main goal of the guidelines is to describe how to model information extracted from their data sources, i.e. how to represent different types of information from their own HIS using the most suitable HL7 v3 template from those provided. Besides, some recommendations are included for the mapping between data sources and the corresponding HL7 v3 message fields. More information regarding the mapping formalism can be found in deliverable 4.3-9.3 “Initial proposal the mapping formalism”.

2.2 Core Dataset

The Core Dataset is the central medical knowledgebase of the platform. The main goal of this component is to facilitate the extraction and exchange of clinical concepts among data models. For that purpose, a set of standards, vocabularies and rules have been developed for the information exchange on the semantic interoperability layer.

More information is availiable in deliverable 4.2 “Initial proposal for the core datasets” [9].

In order to obtain a complete medical vocabulary, a set of clinical vocabularies has been selected to represent the datasets involved in current clinical trials on cancer [4]:

- SNOMED CT (Systematized Nomenclature Of Medicine Clinical Terms)[2]
- LOINC (Logical Observation Identifiers Names and Codes) [3]
- HGNC (HUGO Gene Nomenclature Committee) [4]

The main vocabulary, covering most of the general terms involved, is the SNOMED CT ontology. LOINC covers laboratory results and HGNC represents unique and meaningful names for every known human gene. It is also possible to translate concepts from other terminologies as the International Classification of Diseases (ICD), Medical Dictionary for Regulatory Activities (MedDRA) and others, to one of the Core Dataset selected vocabularies in the semantic interoperability services through the Terminology linking service [7].

2.2.1 Semantic Reasoning Repository

The Core Dataset contains information of these vocabularies such as codes, label, relationships, etc. including the necessary methods for inferring knowledge and managing this information within the semantic interoperability layer. For that purpose, a semantic repository is necessary to store and manage such information. The Sesame Server[5] provides a framework for querying terminologies that are stored as OWL resources.

Therefore, an OWL version of SNOMED CT has been generated based on tools facilitated by the International Health Terminology Standards Development Organisation. Once the OWL file is generated, LOINC and HGNC vocabularies have been also included in the semantic repository.

The large amount of information represented in the SNOMED ontology (311,000 active medical concepts, nearly a million descriptions and over a million relationships) implies a significant complexity that needs to be managed. First steps on this field consist of inferring implicit stated knowledge from explicit represented information; thereby eliminating inconsistencies and all types of non-necessary information on the creation process.

Finally, the OWL file with this information is stored into the semantic repository. In this context, Sesame has been tested with two different configurations: (i) storing triplets in a native storage (triplet store database) and (ii) storing triplets in main memory. We obtained the best performance with the native storage due to the size and complexities of the OWL [5].

[2] Systematized Nomenclature Of Medicine Clinical Terms (SNOMED CT), http://www.ihtsdo.org/snomed-ct/
[3] Logical Observation Identifiers Names and Codes (LOINC®) - http://www.loinc.org
[4] HUGO Gene Nomenclature Committee, http://www.genenames.org/
[5] Sesame server http://www.openrdf.org

2.2.2 Query Expansion

A method for retrieving semantically uniform information from the Common Information Model has been developed. The main objective of this method is to exploit relationship information from Core Dataset to enrich CDM queries. A data flow of the proposed method is shown in *Figure 4*, including a relational wrapper (Morph) that will be detailed on section 3.1.

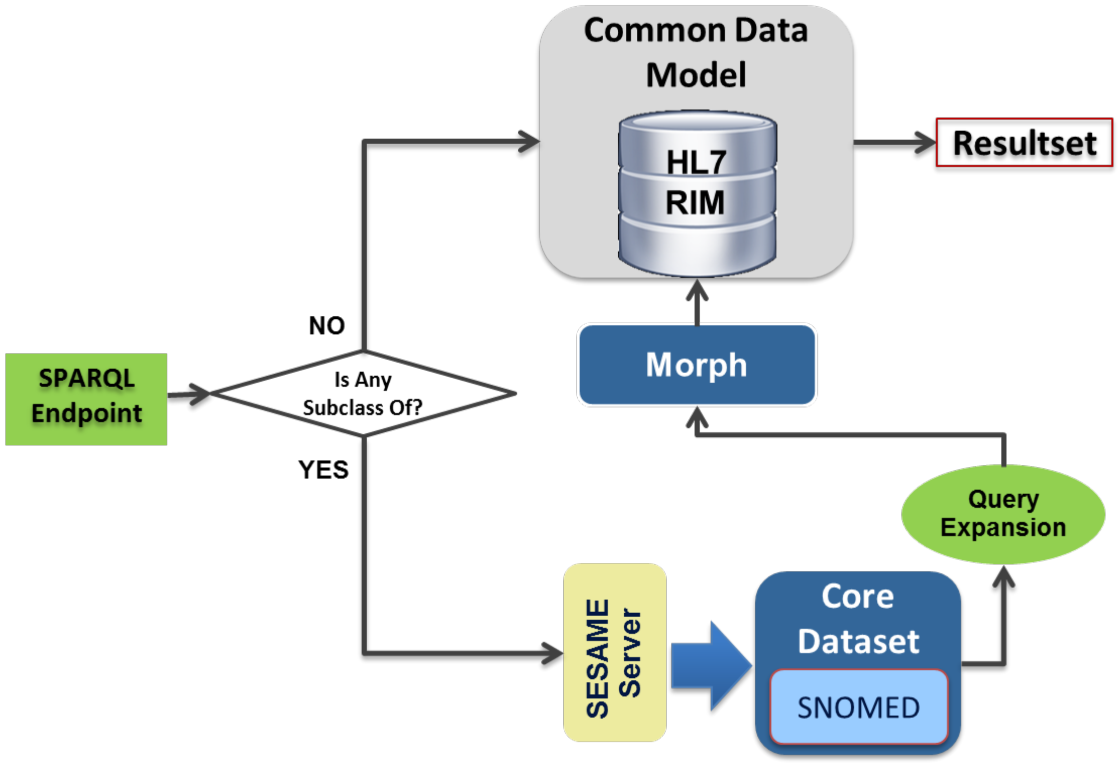


Figure 4: Query expansion data flow

For this purpose, a new functionality is defined and used by the query execution service (described in the next section), the isAnysubClassOf function. This functionality enriches the original SPARQL query with related concepts to the concepts presented on the query. If the original query does not need to be expanded, then it is directly sent to the CDM. If the query has to be expanded then the concepts will be sent to the semantic reasoning repository. Sesame repository executes SPARQL queries on the medical vocabularies for retrieving hierarchical concepts. So finally the original query is enriched with concepts related with the "is_a" relationship, as showed in *Figure 5*. The tree structure corresponds to the Conjunctive Normal Form (CNF), including every subclass concept.

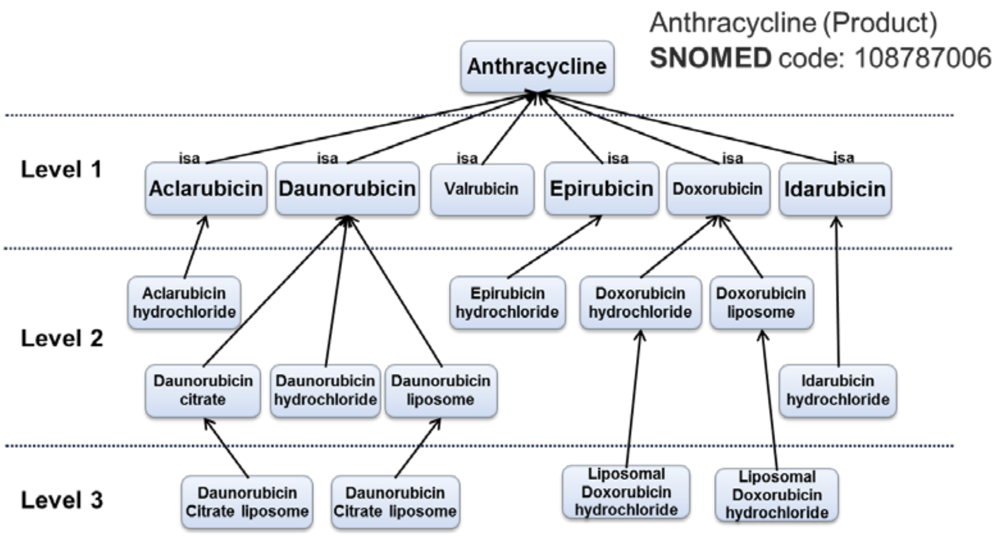


Figure 5: Tree structure example of Anthracycline product

SNOMED CT defines a method (called post-coordination) to represent medical knowledge joining different existing concepts (pre-coordinated concepts). Therefore, it is required to expand post coordinated concepts on the proposed approach. In these cases, the original query considers all the concepts that could be expanded and the relationships among them (represented by the CDM structure), similar to the pre-coordinated query expansion. So the semantic repository expands the post-coordinated concepts and builds the corresponding tree structures as shows in the example of *Figure 6*.

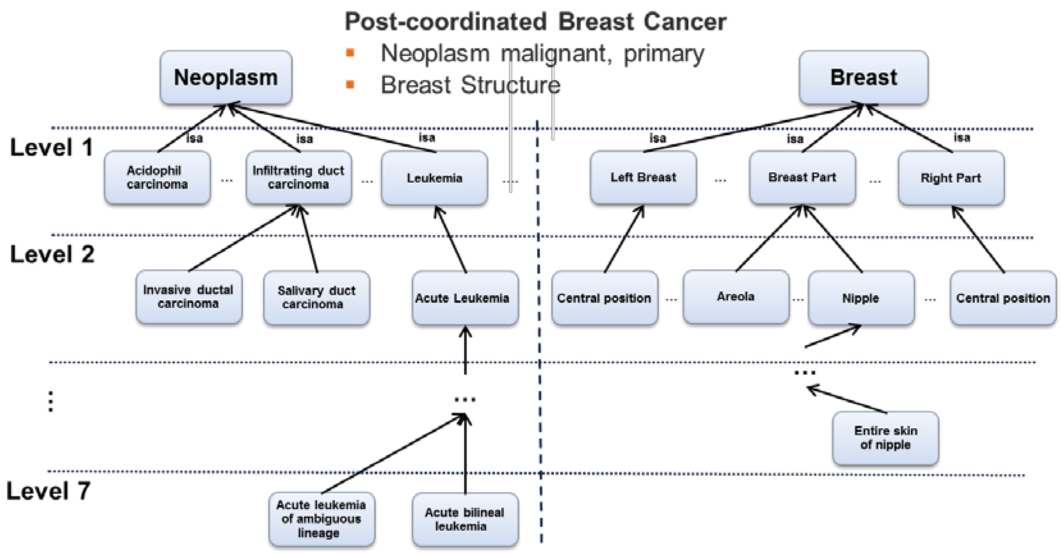


Figure 6: Tree structure example of a post-coordination concept

Finally, the expanded query is sent to the CDM, executed against the data warehouse and results are retrieved.

3 Semantic Interoperability Services

Once components related to exploit semantic knowledge from the Core Dataset, the query execution and data load/normalization services are described in this section. Some of them have been previously described in [6] and [7] with the last updates reported here.

3.1 Query Execution

Query Execution Service enables EURECA applications to retrieve information from the EURECA CDM. It is deployed as a HL7 RIM based CDM on a relational database including a wrapper to execute SPARQL. These queries can be expanded in the Core Dataset component through the Query Expansion method detailed.

In order to translate SPARQL queries into SQL queries, an SPARQL to SQL engine is used to enable this process. The engine used for this is the morph-RDB [8], an open source project based on W3C standards. Morph-RDB interacts with the Query Execution Service once it receives a SPARQL query (expanded or not), translating the query into the SQL sentence and executing it against the CDM.

The SPARQL wrapper requires a mapping that defines the relations between the relational database and the non relational approach (attached in Appendix A). Finally, a XML resulset is returned to the EURECA application.

3.2 Query Normalization

The Query Normalization Service was developed for obtaining query templates that encapsulate data model schema and query syntax for data retrieval from the CDM. For that purpose, the Query Normalization Service has two different methods:

- The uncontextualized method, returning a list of the most relevant query templates for a given Core Dataset concept
- The contextualized method, returning the corresponding query template to the HL7 context provided (i.e. Observation, Substances Administration, Entity or Procedure) and Core Dataset concepts.

These templates are included in the Query Template Library (QTL) for CDM data retrieval based on Core Dataset concepts, attached on Appendix B.

3.3 Data Push Service

The objective of the Data Push Service is to store data from different data sources into the CDM. At the initial prototype of the semantic layer this is achieved by a process divided into two steps. After the data is received by the data push service, being this data structured like a HL7v3 message, an ETL tool extracts, transforms and load the data into an initial version of the information to the CDM. Then, the normalization process takes place in the Normalization Pipeline. Such pipeline transforms the information stored from this initial format, into a standardized form that ensures the homogenization of the information.

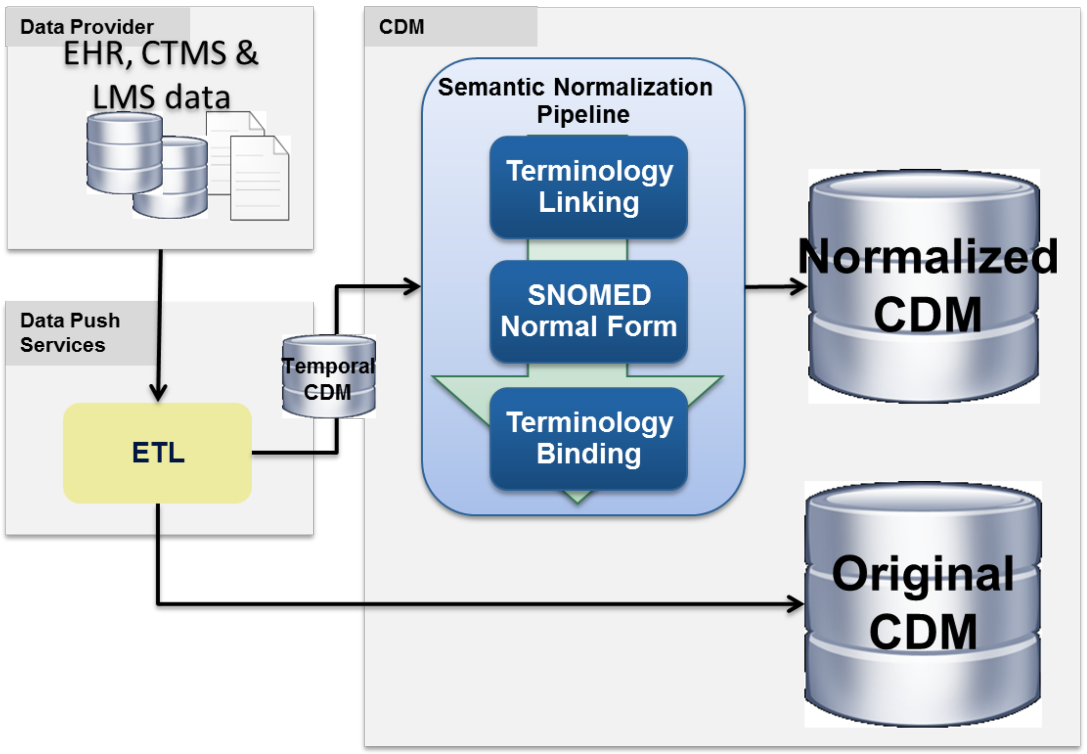


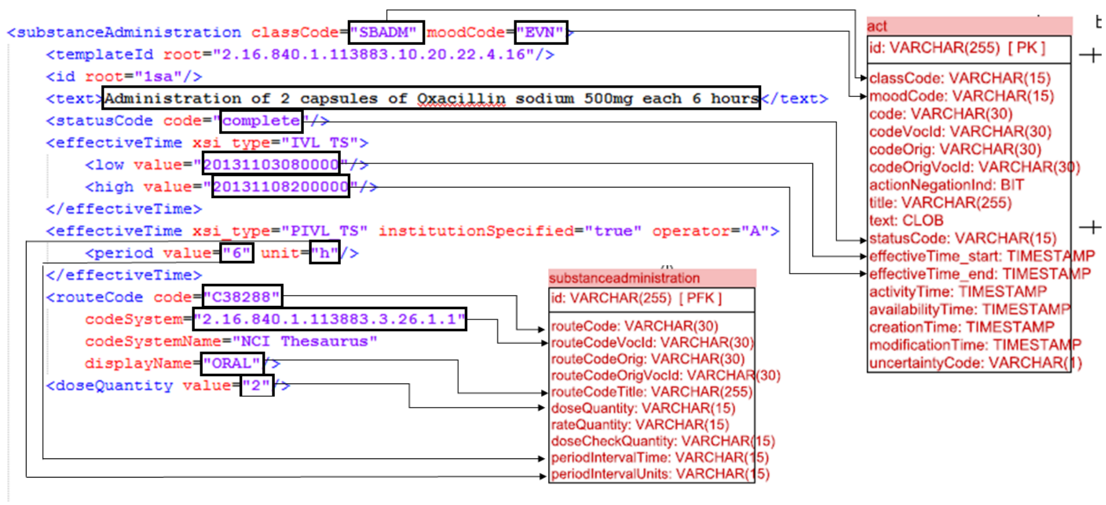
Figure 7: Data Push Service diagram

As it is showed on *Figure 7*, the Data Push Service stores the given message in different databases. In the first step, the original message is stored in a temporal database and in other database that will keep the data at its original state. The temporal database will be used for data normalization as input for the Semantic Pipeline, obtaning a normalized and standardized database. The original database contains the original messages without the transformation process.

3.3.1 ETL

An initial mapping formalism for the ETL process, which consists on to extract, transform and load information from different sources, was initially proposed in the D4.3-9.3 [7] Although in this deliverable it is proposed the use of two different tools for the ETL process, finally for the initial prototype of the semantic layer will only use the Mirth Connect tool.

The first step of the Mirth Connect tool is to parse HL7 messages. After the parsing process (extraction), Mirth Connect enables the transformation for this information, and then this data is piped to the destination channel, which target is the MySQL model of the CDM.



```
<substanceAdministration classCode="SBADM" moodCode="EVN">
    <templateId root="2.16.840.1.113883.10.20.22.4.16"/>
    <id root="1sa"/>
    <text>Administration of 2 capsules of Oxacillin sodium 500mg each 6 hours</text>
    <statusCode code="complete"/>
    <effectiveTime xsi_type="IVL_TS">
        <low value="20131103080000"/>
        <high value="20131108200000"/>
    </effectiveTime>
    <effectiveTime xsi_type="PIVL_TS" institutionSpecified="true" operator="A">
        <period value="6" unit="h"/>
    </effectiveTime>
    <routeCode code="C38288"
        codeSystem="2.16.840.1.113883.3.26.1.1"
        codeSystemName="NCI Thesaurus"
        displayName="ORAL"/>
    <doseQuantity value="2"/>
```

```
act
id: VARCHAR(255) [ PK ]
classCode: VARCHAR(15)
moodCode: VARCHAR(15)
code: VARCHAR(30)
codeVocId: VARCHAR(30)
codeOrig: VARCHAR(30)
codeOrigVocId: VARCHAR(30)
actionNegationInd: BIT
title: VARCHAR(255)
text: CLOB
statusCode: VARCHAR(15)
effectiveTime_start: TIMESTAMP
effectiveTime_end: TIMESTAMP
activityTime: TIMESTAMP
availabilityTime: TIMESTAMP
creationTime: TIMESTAMP
modificationTime: TIMESTAMP
uncertaintyCode: VARCHAR(1)
```

```
substanceadministration
id: VARCHAR(255) [ PFK ]
routeCode: VARCHAR(30)
routeCodeVocId: VARCHAR(30)
routeCodeOrig: VARCHAR(30)
routeCodeOrigVocId: VARCHAR(30)
routeCodeTitle: VARCHAR(255)
doseQuantity: VARCHAR(15)
rateQuantity: VARCHAR(15)
doseCheckQuantity: VARCHAR(15)
periodIntervalTime: VARCHAR(15)
periodIntervalUnits: VARCHAR(15)
```

Figure 8: Mirth Connect e.g. HL7v3 to CDM

The *Figure 8* shows a brief scheme example of how the information stored in a HL7 v3 message is extracted, and then which fields from the CDM will be used to store each specific piece of information. Other fields of the CDM are fulfilled according to transformations from the information sent with the HL7v3 message, that in order to simplified, is not included in the figure.

After an HL7v3 message is received by the Data Push, this service sends the message to the corresponding Mirth Connect channel. The message is then processed and stored in two different databases. The original database keeps an initial state of the information prior to the normalization process. The other database, a temporary one, will store a duplicate of the information that will be the main input of the following process, the Normalization Pipeline.

3.3.2 Normalization Pipeline

Normalization Pipeline has been thoroughly described in section 2.2.2.2 of deliverable 4.3-9.3 [7]. There are no changes in this process since the initial version.

3.4 Auto Complete

The Auto Complete service main objetive is to provide Core Dataset concepts that contain the string or label introduced by final users of applications. In order to obtain a better accuracy on finding Core Dataset concepts, it is possible to search concepts filtering by the HL7 RIM context. The possible location of the Core Dataset concept on the CDM is defined as the context in such filters (i.e. Observation, Substances Administration, Entity or Procedure). It is also possible to add secondary filters for additional precision when too many results are retrieved. These filters are defined as the root concept of Medical ontologies, in this case only available for SNOMED CT.

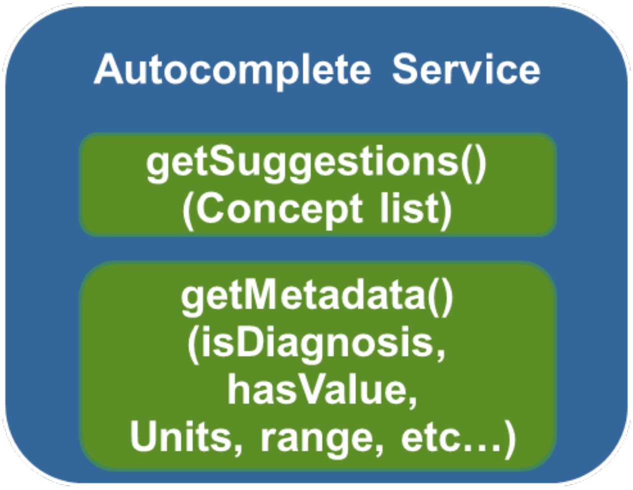


Figure 9: Methods provided by the Autocomplete Service

Once the concept is selected, the Auto Complete offers the getMetadata method to return the additional information about Core Dataset concept. Metadata of a given concept include information regarding how this concept has been stored in different instances of the CDM. Therefore, for one concept is possible to know if it has associated values, the respective units and the maximum and minimum value. It is also possible to obtain other related information about doses, diagnosis or other measurements, as showed in *Figure 10*.

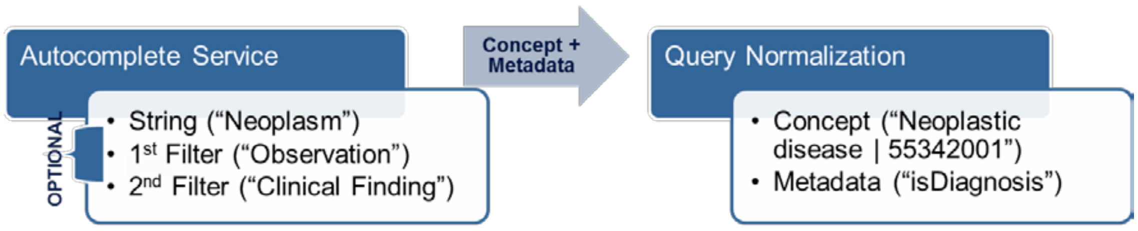

Autocomplete Service
OPTIONAL
• String ("Neoplasm")
• 1st Filter ("Observation")
• 2nd Filter ("Clinical Finding")
Concept + Metadata
Query Normalization
• Concept ("Neoplastic disease | 55342001")
• Metadata ("isDiagnosis")

Figure 10: Autocomplete Service example

The Auto Complete service, including the getMetadata method, allows efficiently searching for Core Dataset concepts and retrieving their mapping to the CDM.

3.5 Free Text

3.5.1 Context and Motivation

Medical domain is very challenging for information retrieval since it has very rich terminology. A conceptual link can exist between a query and a document even if the query terms do not occur explicitly in the document. For example, a user who wants to query for relevant trials at ClinicalTrials.gov may not necessarily be aware of different ways to express the same specific relevant disease term. In this case, standard term-based query search may either retrieve a huge amount of trials not necessarily relevant to the user (if the user query is not specific enough), or fail in retrieving any trial (if the user do not use exactly the same terms as the ones used by the medical expert designing the trial).

There is a number of previous works justifying the usage of semantic resources in medical information retrieval. In [17], authors attempt to create a MeSH counterpart for ClinicalTrials.gov for more efficient indexing and search. They do so by creating a set of eligibility criteria features mapped to UMLS concepts. Also, many participants of Medical IR tracks (such as TREC Genomics or ImageCLEF medical track) are exploiting UMLS semantic network [15][16]to improve the retrieval results. Examples of the best performing systems are [13]or [14]which are both using UMLS for indexing medical concepts in the document, and for generating term variants and/or synonyms for query expansion

The main advantage of integrating medical concept identification in a text query engine is the possibility to normalize term variants in user's queries and in indexed documents. There are multiple variation types:

1. Typographical (*Abdominal Distension vs. abdominal distension*)
2. Orthographical (*abdominal distension vs. abdominal distention*)
3. Morphological/Inflectional (*abdominal distension vs. abdominal distensions*)
4. Morphological/Derivational (*abdominal distension vs. abdomen distension*)
5. Lexical (abdominal distension vs. abdominal swelling)
6. Syntactic (abdominal distension vs. distension of the abdomen)
7. Discontinuity *("Stage I and Stage II" vs. "Stage I and II"*)

Existing medical terminologies lists only a subset of term variants, and therefore a simple term lookup method is not sufficient for term normalization. Thus, we propose to extend the EURECA free-text search engine developed by UPM with indexing based on UMLS concepts, using the Xerox concept identifier developed in WP3 [11].

3.5.2 Concept Identifier for Free text Query Search

3.5.2.1 Concept Identifier description

The concept identifier (CI) [11] is an automatic tool that identifies medical terms in free text and maps them to UMLS concepts using UMLS Unique Concept Identifiers (CUIs). The concept identifier works on the basis of a term database (DB) compiled from terminology sources provided by the user. These can be either UMLS-integrated terminologies, or user's homemade term sets.

For example, Concept C0000731 can be expressed with any of the following terms in free text:

- *Abdominal Distension*
- *abdominal distension*
- *abdominal distention*
- *Abdomen distended*
- *Abdominal swelling*

These variants are listed in different UMLS-integrated terminology sources. When the concept identifier encounters any of these terms in free text, it is capable to link them to the same CUI which allows generalizing the context of the document. On the other hand, once a specific term has been linked to its CUI, it is possible to extract all different term variations from UMLS, thus generating a list of potential synonyms.

In addition to the normalization of listed terms (i.e. with a simple dictionary lookup), the current version of the concept identifier [12] allows for the following normalizations even when the variants are not listed in the term DB:

- Typographical: *ABDOMINAL DISTENSION → Abdominal Distension*
- Orthographical: *abdominal distention → abdominal distension*
- Inflectional: *abdominal distensions → abdominal distension*
- Misspelling normalization: *abdominal disstension → abdominal distension*

Linking these different terms variations to the UMLS CUI allows for term normalization in addition to synonym detection.

Another extension of the last version of concept identifier [12] allows for embedded terms detection. For example, for the free text "*Invasive Breast Cancer*" concept identifier will not only extract the longest match term: *Invasive Breast Cancer [C0853879],* but also embedded concepts : *Breast Cancer [C0006142]* and *Invasive[C0205281]*.

Embedded concepts detection may be activated or deactivated, depending on users need.

Given all the properties of the Concept Identifier, we propose to integrate it for free text query search as the following.

3.5.2.2 Integration to the free-text query engine

The Concept Identifier can be used to preprocess both the free-text documents that are indexed by the search engine, as well as the user query used to search these documents. There are two possibilities for such preprocessing.

1. One may replace standard bag of word representation of a document and/or query with bag of concepts representation, where the concepts are equivalent to the list of CUIs provided by concept identifier. This will allow for a generalization, and a more compact index. However, this may decrease the recall of the retrieval if the query contains some keywords that are not necessarily medical domain specific, and might be omitted by concept identifier.

2. Another option would be to extend the document/query with the “bag of CUIs”. This will allow preserving generalization, without the risk of recall decreasing.

Note that if the embedded terms option of concept identifier is activated this will allow for higher recall but potentially lower precision retrieval. Given that in EURECA, the free text query search is used when no results were retrieved otherwise one might consider activating the embedded terms detection to favour higher recall.

In addition, documents indexing and query expansion with UMLS CUIs will naturally lead to free-text cross-lingual search since UMLS codes are unique across different languages. Of course, this might be achieved only for the languages that are covered by concept identifier (English and French in a current state).

FREE TEXT DOCUMENT:

Inclusion Criteria:
- Patients with a histologically or cytologically proven metastatic breast cancer.
- Patients with at least one bidimensionally measurable lesion (diameter > 1 cm), or an evaluable bone lesion that will not undergo biopsy.
…

BAG of CUIs

```
Inclusion Criteria  C1512693
metastatic breast cancer   C0278488
metastatic    C0006142
breast cancer       C0006142
breast C0006141
Patients      C0030705
Biopsy C0005558
Evaluable     C1516986
bone lesion   C0238792
lesion C0221198
```

USER QUERY : Carcinoma breast stage IV
PREPROCESSED USER QUERY : C0278488[Carcinoma breast stage IV], C0441772 [stage IV], C0007097[carcinoma], C0006141 [breast]

USER QUERY : Cancer du sein
PREPROCESSED USER QUERY : C0006142 [cancer du sein], C0006141[sein]

3.6 Terminology Linking

Terminology Linking Service has been deeply described on section 2.1.3 of deliverable 4.3-9.3 [7] and in deliverable 4.2 [9]. And there are no changes on this process since this version.

4 Security Integration

4.1 Overview

In order to ensure that the legal requirements as described in deliverable 7.3 are fulfilled by the semantic interoperability solution, several security tools (authorisation/authentication) were integrated.

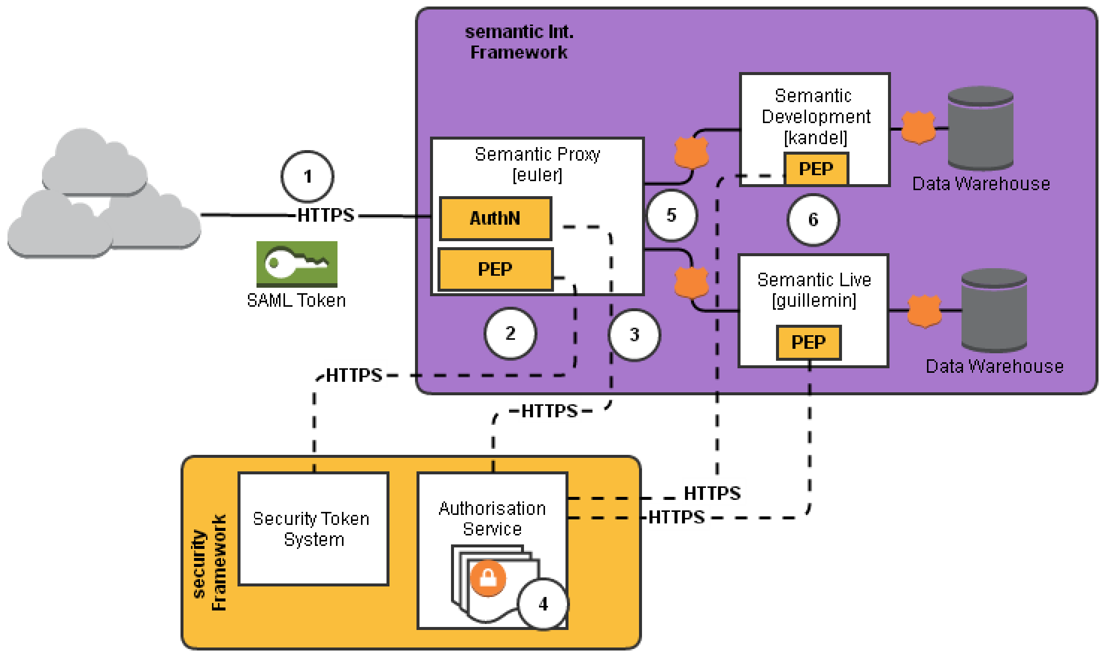


Figure 11: Semantic Interoperability Security Integration

Figure 11 gives an overview of the technical solution worked out, it contains 6 security measures taken in EURECA:

1) Each incoming data query request (send over HTTPS) should contain a signed SAML token issued by a EURECA identity provider. It contains attribute information (name, organisation, id) about the requester.
2) In the Semantic Proxy, the SAML token will be validated (signature and timestamp checking). If the SAML token is found invalid, access will be rejected to the request.
3) Next, the Policy Enforcement Point of the Semantic Proxy will generate an access request including the attribute information about the requester. This request is send to the central EURECA Authorisation service (over HTTPS).
4) The Authorisation Service will retrieve the request coming from the Semantic Proxy and will generate an access decision, based on defined policies. These policies are compliant with the contracts[6] signed by the EURECA partners.

[6] Hence through the Authorisation Service the system assures that the Institution has signed the End-user Agreement and that the individual signed an Annex C to this contract (which has a function of a Non-disclosure-agreement).

All of EURECA Partners interested in access to data signed the End-user agreement.

The contracts between the End-users and CDP are included in the Annex to D7.1.

The Authorisation Service will return the access decision back to the Semantic Proxy.

5) If access was granted, the query request will be forwarded to the internal semantic services (that are only accessible by the Semantic Proxy). If access was denied, access will be rejected to the request.
6) The Semantic Services contain an extra Policy Enforcement Point which will handle fine grained access control (see next section). The flow is similar as in step 3 and 4.

4.2 Fine Grained Access Control Security

Fine-grained access control enables authorisation on the object level in the EURECA semantic solution. For data stored in the CDW, it means row-level or cell-level security. For EURECA a solution was worked out to enable fine-grained access control with trial arm granularity. The different steps are explained using the following flow diagram (see *Figure 12*):

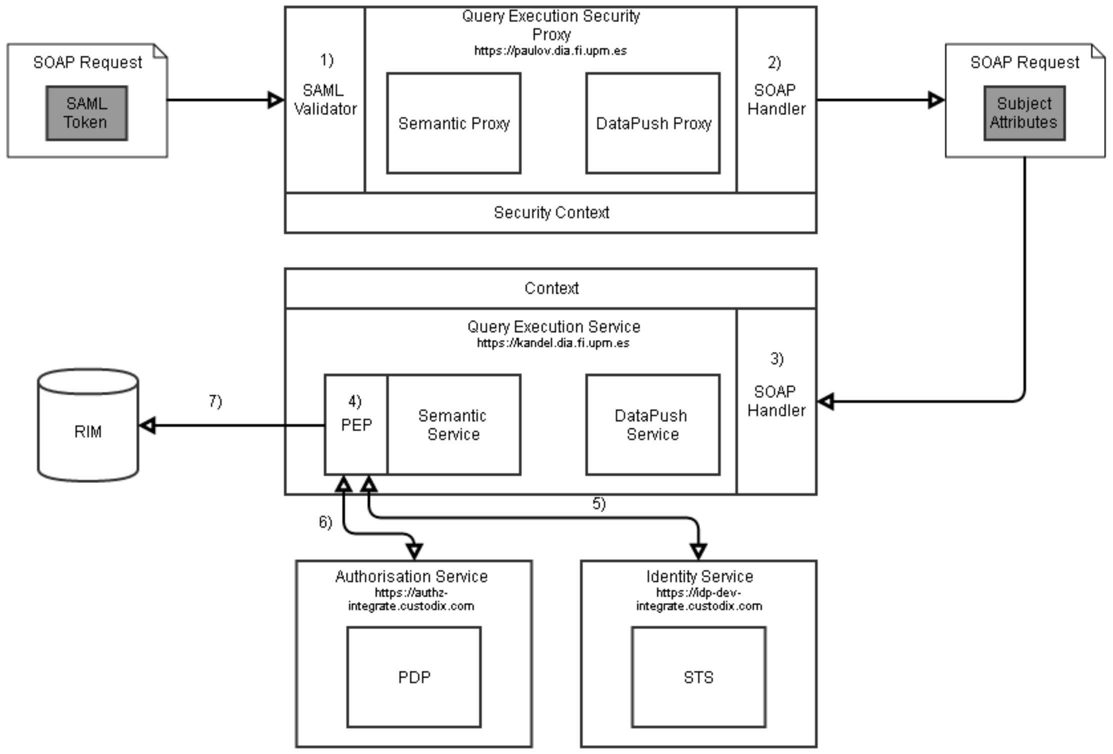


Figure 12: Fine-grained Access Control EURECA Solution

1. Incoming SOAP requests at the *query execution security proxy* contain a SAML token with SAML attributes issued by the *identity service (see previous section)*. These attributes contain information about the requester. The SAML validator will first check if the incoming SAML token is valid (if not valid, not access is given to the requestor). Next, the validator will strip the SAML attributes from the SAML token. These (subject) attributes are placed in the global security context of the service.
2. SOAP requests from the query execution security proxy to the query execution service are intercepted by a SOAP handler. This SOAP handler will add custom SOAP headers to each request containing the subject attributes available in the security context (coming from the SAML token). The SOAP headers have following structure (XSD Schema):

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://custodix.com/Security/SubjectAttributes"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="SubjectAttributes">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SubjectAttribute" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="Value" maxOccurs="unbounded"
minOccurs="1"/>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example of SOAP subject attributes headers:

```
<custodix:SubjectAttributes
xmlns:custodix="http://custodix.com/Security/SubjectAttributes">
      <custodix:SubjectAttribute name="urn:oid:0.9.2342.19200300.100.1.3">
        <custodix:Value>kristof@custodix.com</custodix:Value>
      </custodix:SubjectAttribute>
      <custodix:SubjectAttribute name="urn:oid:2.16.840.1.113730.3.1.241">
        <custodix:Value>Kristof</custodix:Value>
      </custodix:SubjectAttribute>
      <custodix:SubjectAttribute name="urn:oid:0.9.2342.19200300.100.1.1">
        <custodix:Value>4fec04d2-717d-4c0b-b456-e42c0a0eec3f</custodix:Value>
      </custodix:SubjectAttribute>
      <custodix:SubjectAttribute name="urn:oid:2.5.4.42">
        <custodix:Value>Kristof</custodix:Value>
      </custodix:SubjectAttribute>
      <custodix:SubjectAttribute name="urn:oid:2.5.4.4">
        <custodix:Value>De Schepper</custodix:Value>
      </custodix:SubjectAttribute>
</custodix:SubjectAttributes>
```

3. The Subject attributes in the headers of an incoming SOAP request in the *semantic layer* will be stripped by a SOAP handler. The functionality will be similar to 2) (inverse operation). Once stripped, the subject attributes are placed in a context accessible within the *query execution service.*
4. A PEP module is placed in the query flow of the query execution service and intercepts each incoming query.
5. When a query enters the query flow, the PEP module will first need to authenticate itself to the *identity service* before it can send an fine-grained access control request to the *authorisation service* (the authorisation service

requires SAML authentication using WS-Trust). A username and password is required (PEP needs to be registered on the *identity service*).

6. Next the PEP will generate a fine-grained access control request for each different trial arm stored in the DW (received from the governance metadata). Each such request will include:
   a. The subject attributes that were place in the context of the query execution service, for identifying the requesting user
   b. The targeting trail arm (resource attribute)

This request is send to the PDP that will generate a fine-grained access decision based on pre-defined fine-grained access control policies.

7. All the trial arms that received an access deny from the PDP cannot be queried. For this restriction, the incoming query is modified by adding an additional filter restricting the query to permitted trial arms.
8. Finally the query is executed on the common data warehouse and results are sent back to the requester.

5 CONCLUSIONS

This deliverable reports on the actual state of the EURECA Semantic Interoperability framework. It is important to note that the semantic layer must be used in almost all the defined clinical scenarios of deliverable 1.2. ”Definition of relevant user scenarios based on input from users”.

Semantic Interoperability layer first version was deployed on the first 5 months of the EURECA project and after more than 2 years it is still improving and developing new services that will be updated in deliverable 4.6 “Final prototype of the semantic interoperability platform”. Therefore this deliverable contains the actual state of the core components of the EURECA Semantic Interoperabilty layer and the stable version of the services developed to use it.

Next steps on the Semantic Interoperability Layer will be focused on:

- Improvement of performance of Query Execution service allowing query paralelization executions.
- Implementation of a Provenance Service to store the origin or state of the diferent datasets and how it is used.
- Implementation of a DataWarehouse Management service to manage the different CDM.
- Integrate developments with EURECA WP3 allowing Free Text Service for retrieving data from Natural Language Process.
- Data curation for homogeneous representations on units and values on the Data Push Servic

With such improvements, we will provide a powerful core component to homogeneously access clinical data within the EURECA platform. Current components and services were showed on the first and second EURECA annual review, supporting the following scenarios: Patient screening, Trial recruitment, Long-term follow-up, Pre-filling of CRF and AE reports and Microbiology SAE.

6 REFERENCES

[1] Paraiso-Medina S, Perez-Rey D, Alonso-Calvo R, Claerhout B, de Schepper K, Hennebert P, Lhaut J, Van Leeuwen J and Bucur A.“Semantic Interoperability Solution for Multicentric Breast CancerTrials at the Integrate EU Project” In Proceedings of HEALTHINF 2013. (1):34-41

[2] EURECA deliverable 1.3 “Report on state of art on relevant knowledge and data sources and on reusable tools”

[3] BENSON, Tim. Principles of health interoperability HL7 and SNOMED. London: Springer, 2010.

[4] Bucur, A., van Leeuwen, J., Perez-Rey, D., Calvo, R. A., Claerhout, B., & de Schepper, K. (2012, November). Identifying the semantics of eligibility criteria of clinical trials based on relevant medical ontologies. In Bioinformatics & Bioengineering (BIBE), 2012 IEEE 12th International Conference on (pp. 413-421). IEEE

[5] Moratilla, J. M., Alonso-Calvo, R., Molina-Vaquero, G., Paraiso-Medina, S., Perez-Rey, D., & Maojo, V. (2012). A Data Model Based on Semantically Enhanced HL7 RIM for Sharing Patient Data of Breast Cancer Clinical Trials. Studies in health technology and informatics, 192, 971-971.

[6] EURECA Deliverable: 2.2 “Initial EURECA architecture”.

[7] EURECA Deliverable: 4.3 & 9.3 “Initial proposal for the mapping formalism and mappings to EHR and CT models”.

[8] Freddy Priyatna, Oscar Corcho, Juan Sequeda. Formalisation and Experiences of R2RML-based SPARQL to SQL query translation using Morph. World Wide Web Conference (WWW 2014).

[9] EURECA Deliverable 4.2 “Initial proposal for the core datasets”.

[10] S. Aït-Mokhtar, B. De Bruijn, C. Hagège and P. Rupi, “EURECA Deliverable 3.2: Initial prototype for relation identification between concepts,” 2013.

[11] S. Aït-Mokhtar, C. Hagège and P. Rupi, “EURECA Deliverable 3.1: Initial prototype for concept extraction out of EHR free text”, 2013.

[12] EURECA Deliverable D3.5: S. Aït-Mokhtar, B. De Bruijn, V. Nikoulina, “Intermediary-stage IE components,” 2014.

[13] Ide NC, Loane RF, Demner-Fushman D. Essie: a concept-based search engine for structured biomedical text. J Am Med Inform Assoc. 2007;14(3):253–63.

[14] Loïc Maisonnasse, Farah Harrathi, Catherine Roussey, Sylvie Calabretto: Analysis Combination and Pseudo Relevance Feedback in Conceptual Language Model - LIRIS Participation at ImageCLEFMed. CLEF (2) 2009: 203-210

[15] Henning Müller, Jayashree Kalpathy-Cramer, Ivan Eggel, Steven Bedrick, Saïd Radhouani, Brian Bakke, Charles E. Kahn Jr., William R. Hersh: Overview of the CLEF 2009 Medical Image Retrieval Track. CLEF (2) 2009: 72-84

[16] Hersh WR, Cohen A, Roberts P, Bhupatiraju RT. TREC 2006 genomics track overview. Gaithersburg, MD: National Institute of Standards and Technology (NIST); 2006. In: Proceedings of the Fifteenth Text REtrieval Conference, 2006 Nov 14–17

[17] Boland MR, Miotto R, Gao J, Weng C. Feasibility of feature-based indexing, clustering, and search of clinical trials. A case study of breast cancer trials from ClinicalTrials.gov. Methods Inf Med. 2013;52(5):382-94.

[18] EURECA deliverable 2.1 “State of the art report on standards”.

[19] EURECA deliverable 9.2, “Canonical models of EHR and CT systems”.

[20] Siegel, E. L., & Channin, D. S. (2001). Integrating the Healthcare Enterprise: A Primer: Part 1. Introduction 1. Radiographics, 21(5), 1339-1341.

A. APPENDIX

a. Mapping ttl (relational to non-relational)

The complete mapping file could be downloaded from the shared workspace portal of the EURECA platform and also on a BitBucket repository[7]. This file contains the mapping between the relational model of the developed CDM and a non relational CDM.

Attached is the mapping example for the observation table. In this example, it is defined how it is represented on RDF the table observation; primary keys, foreign keys, attributes…etc.

```
<TriplesMapObservationAct>
  a rr:TriplesMap;

  rr:logicalTable [ rr:tableName  "observation" ];

  rr:subjectMap [ rr:termType rr:IRI;
    rr:template "http://localhost:2020/resource/obsactno/{id}";
    rr:class hl7rim:observationAct;
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_act ];
    rr:objectMap    [
      rr:parentTriplesMap <TriplesMapAct>;
      rr:joinCondition [ rr:child "id" ; rr:parent "id" ; ]
    ];
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_actObservationInterpretationCode ];
    rr:objectMap    [
      rr:parentTriplesMap <TriplesMapActObservationInterpretationCode>;
      rr:joinCondition [ rr:child "id" ; rr:parent "id" ; ]
    ];
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_participation ];
    rr:objectMap    [
      rr:parentTriplesMap <TriplesMapParticipation>;
      rr:joinCondition [ rr:child "id" ; rr:parent "actId" ; ]
    ];
  ];

  rr:predicateObjectMap [
```

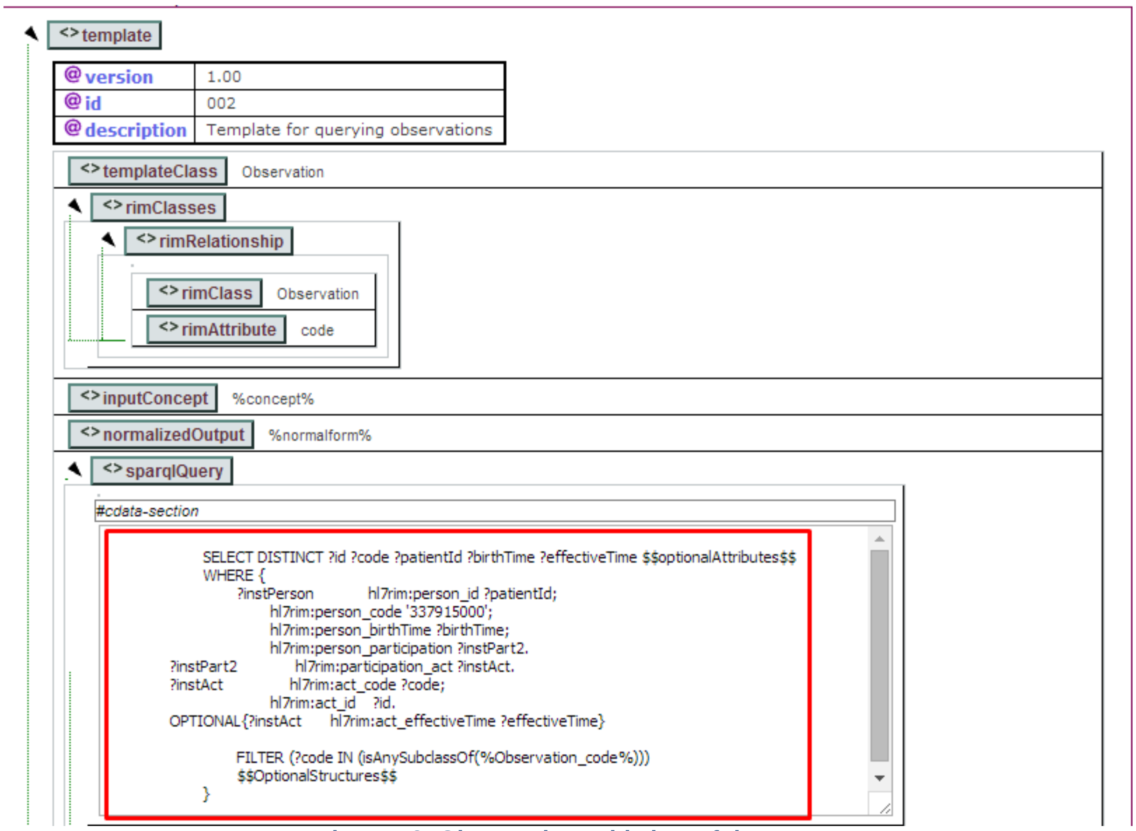
[7] https://bitbucket.org/sparaiso/semantic-normalization-and-query-abstraction-based-on-snomed/src/6f1b272a7b2a?at=master

```
    rr:predicateMap [ rr:constant hl7rim:observationAct_actObservationValues ];
    rr:objectMap    [
      rr:parentTriplesMap <TriplesMapActObservationValues>;
      rr:joinCondition [ rr:child "id" ; rr:parent "id" ; ]
    ];
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_actMethodCode ];
    rr:objectMap    [
      rr:parentTriplesMap <TriplesMapActMethodCode>;
      rr:joinCondition [ rr:child "id" ; rr:parent "id" ; ]
    ];
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_actTargetSiteCode ];
    rr:objectMap    [
      rr:parentTriplesMap <TriplesMapActTargetSiteCode>;
      rr:joinCondition [ rr:child "id" ; rr:parent "id" ; ]
    ];
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_id ];
    rr:objectMap    [ rr:termType rr:Literal; rr:column "id"; ];
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant hl7rim:observationAct_valueNegationInd ];
    rr:objectMap    [ rr:termType rr:Literal; rr:column "valueNegationInd"; ];
  ];


.
```

b. Query template file

All query templates could be downloaded from the shared workspace portal of the EURECA platform and also on a BitBucket repository[8].

Following figures show a graphical grid view of a query template. For example, in *Figure 13* is showed how is represented some structure information as the SPARQL query, concept normal form…et.c of the *Observation* template.



@version	1.00
@id	002
@description	Template for querying observations

<> template

<> templateClass Observation

<> rimClasses

<> rimRelationship

<> rimClass Observation

<> rimAttribute code

<> inputConcept %concept%

<> normalizedOutput %normalform%

<> sparqlQuery

#cdata-section

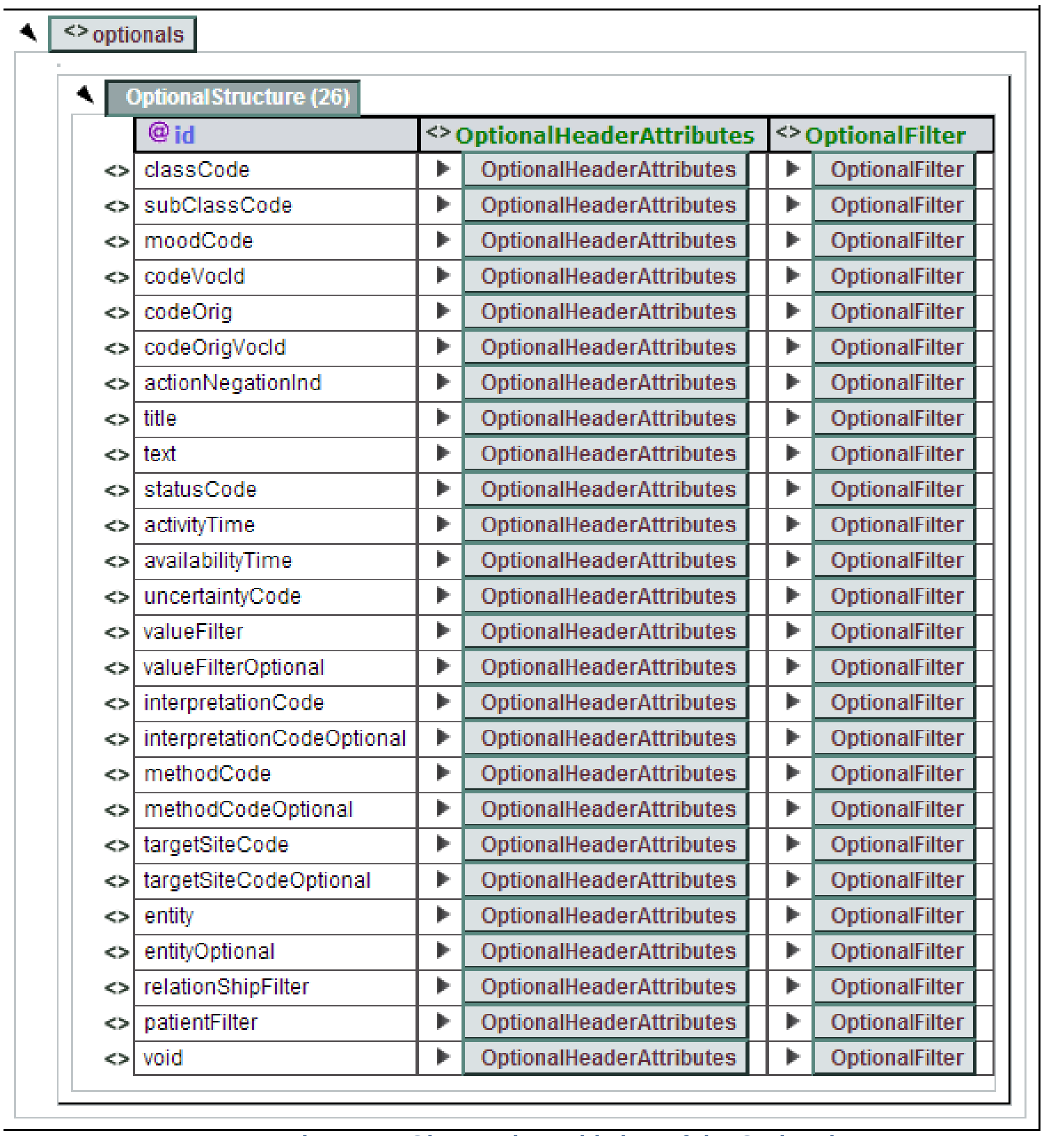
```
            SELECT DISTINCT ?id ?code ?patientId ?birthTime ?effectiveTime $$optionalAttributes$$
            WHERE {
                  ?instPerson          hl7rim:person_id ?patientId;
                        hl7rim:person_code '337915000';
                        hl7rim:person_birthTime ?birthTime;
                        hl7rim:person_participation ?instPart2.
      ?instPart2           hl7rim:participation_act ?instAct.
      ?instAct             hl7rim:act_code ?code;
                        hl7rim:act_id   ?id.
      OPTIONAL{?instAct      hl7rim:act_effectiveTime ?effectiveTime}

                  FILTER (?code IN (isAnySubclassOf(%Observation_code%)))
                  $$OptionalStructures$$
            }
```

Figure 13: Observation grid view of the XML

In *Figure 14* is detailed all the optional structure that could be added on the SPARQL query. These optional structures allow increasing the specificity of the query adding more restrictions for retriving data on the CDM.

In *Figure 15* is detailed how is represented the entity optional structure. In this example, it is possible to add headers to the original query and filters in the SPARQL query.

[8] https://bitbucket.org/sparaiso/semantic-normalization-and-query-abstraction-based-on-snomed/src/6f1b272a7b2a067570678f41c5deb6c5fc3b53b5/Query%20templates/?at=master

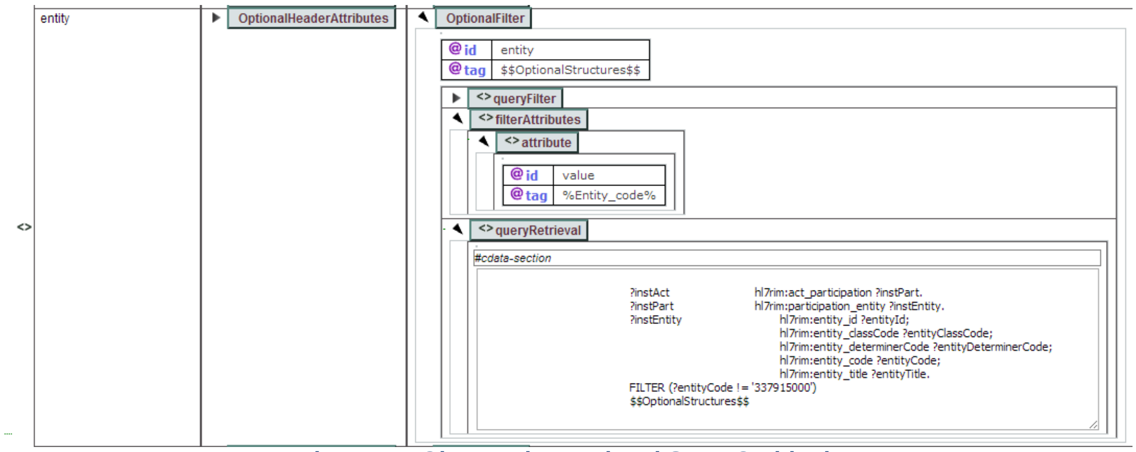


<> optionals

OptionalStructure (26)

	@ id	<> OptionalHeaderAttributes	<> OptionalFilter
<>	classCode	OptionalHeaderAttributes	OptionalFilter
<>	subClassCode	OptionalHeaderAttributes	OptionalFilter
<>	moodCode	OptionalHeaderAttributes	OptionalFilter
<>	codeVocId	OptionalHeaderAttributes	OptionalFilter
<>	codeOrig	OptionalHeaderAttributes	OptionalFilter
<>	codeOrigVocId	OptionalHeaderAttributes	OptionalFilter
<>	actionNegationInd	OptionalHeaderAttributes	OptionalFilter
<>	title	OptionalHeaderAttributes	OptionalFilter
<>	text	OptionalHeaderAttributes	OptionalFilter
<>	statusCode	OptionalHeaderAttributes	OptionalFilter
<>	activityTime	OptionalHeaderAttributes	OptionalFilter
<>	availabilityTime	OptionalHeaderAttributes	OptionalFilter
<>	uncertaintyCode	OptionalHeaderAttributes	OptionalFilter
<>	valueFilter	OptionalHeaderAttributes	OptionalFilter
<>	valueFilterOptional	OptionalHeaderAttributes	OptionalFilter
<>	interpretationCode	OptionalHeaderAttributes	OptionalFilter
<>	interpretationCodeOptional	OptionalHeaderAttributes	OptionalFilter
<>	methodCode	OptionalHeaderAttributes	OptionalFilter
<>	methodCodeOptional	OptionalHeaderAttributes	OptionalFilter
<>	targetSiteCode	OptionalHeaderAttributes	OptionalFilter
<>	targetSiteCodeOptional	OptionalHeaderAttributes	OptionalFilter
<>	entity	OptionalHeaderAttributes	OptionalFilter
<>	entityOptional	OptionalHeaderAttributes	OptionalFilter
<>	relationShipFilter	OptionalHeaderAttributes	OptionalFilter
<>	patientFilter	OptionalHeaderAttributes	OptionalFilter
<>	void	OptionalHeaderAttributes	OptionalFilter

Figure 14: Observation grid view of the Optionals



<> entity — OptionalHeaderAttributes — OptionalFilter

@ id	entity
@ tag	$$OptionalStructures$$

<> queryFilter

<> filterAttributes

<> attribute

@ id	value
@ tag	%Entity_code%

<> queryRetrieval

#cdata-section

```
?instAct                hl7rim:act_participation ?instPart.
?instPart               hl7rim:participation_entity ?instEntity.
?instEntity                  hl7rim:entity_id ?entityId;
                             hl7rim:entity_classCode ?entityClassCode;
                             hl7rim:entity_determinerCode ?entityDeterminerCode;
                             hl7rim:entity_code ?entityCode;
                             hl7rim:entity_title ?entityTitle.
FILTER (?entityCode != '337915000')
$$OptionalStructures$$
```

Figure 15: Observation optional SPARQL block