Smart Open Services for European Patients

Open eHealth initiative for a European large scale pilot of

Patient Summary and Electronic Prescription

Work Package 3.5

Semantic Services Definition

Appendix E - Ontology Specifications

D3.5.2

February 23rd, 2010

Document Version: 0.3

WORK PACKAGE	**WP 3.5**
DOCUMENT VERSION	**0.3**
DATE	**23/02/2010**

Table of contents

Appendix E - Ontology

1. General Description

The knowledge maintained in ontologies requires some kind of representation. The classes, their attributes, the instances and all relationships can be stored in different ways: relational databases, CVS tables, simple XML files or semantic web technologies, namly Web Ontology Language (OWL) – to name a few. As long as all the necessary expressivity and level of logical coherence are covered they can be treated equally.

In the starting phase of a project, especially as long as the amount of concepts is quite limited, the use of sophisticated and established tools helps to accelerate the foundational definition of the ontology itself. Therefore, no time is wasted in developing an interface (GUI) to enter and maintain the core concepts.

The biggest advantage of OWL coded ontologies against ones represented in relational databases is that an ontology represented in a database should have a fixed structure, while OWL allows ontology development in a distributed way. We hold that this means that an OWL file will provide greater flexibility of a sort that will be a pre-requisite to the evolution of the terminology at hand. Furthermore, providing an OWL file presents a strategy to foster future use and future development of the components provided by epSOS since OWL is a W3C (Semantic Web) standard, where technologies based on “structure upfront” approaches are not seen as fit for Semantic Web computing. We stress that in developing OWL file we are working in OWL DL, which is one of the major upcoming standards in the semantic web community. OWL DL was designed to provide the maximum expressiveness possible while retaining computational completeness, decidability and the availability of practical reasoning algorithms.,

Nevertheless, we do not deny that relational databases can be useful to achieve goals within epSOS. Moreover, in Terminology Access Services we assume that for “on-the-fly” translation, transformation of ontology into relational database will be used. We stress the point that the fact that Subgroup B WP 3.5 provides an OWL coded version of the ontology should not prevent anybody from transferring the OWL file into an SQL database for query or other purposes. One strategy to convert an OWL file into an SQL database is described in the paper by Astrova et al.

The epSOS Ontology will be built using the Protégé Ontology Editor. We suggest viewing the ontology for information and review purposes using the same tool. Inference will be checked using the Pellet 2.0 Reasoner.

In order to foster harmonization and re-use a well-tested structure for ontologies for the medical arena we will use an Upper Ontology. For epSOS we will use Basic Formal Ontology (BFO). We foresee the necessity to provide a user-friendly perspective on the ontology which we will provide in time, after initial development of the OWL file is done.

The prototype will allow evaluating OWL in term of suitability (possibility to describe the required knowledge) and scalability (including performance when the terminology will increase its complexity and completeness).

1.1. The Translation Process using Ontologies

A general description of the translation process using ontologies can be seen in *Figure 1*, below

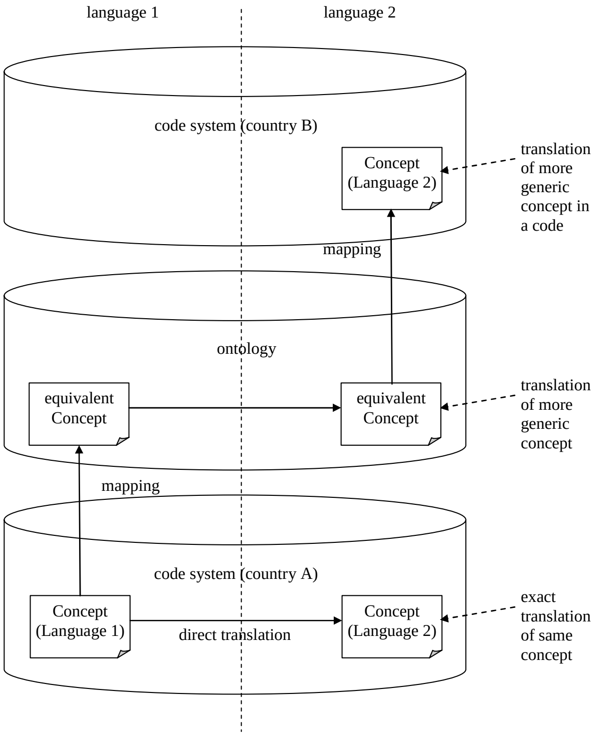


***Figure 1 –** The translation process based on ontologies.*

2. epSOS Ontology

The epSOS Ontology can be downloaded from Project Place using the following URL link: https://service.projectplace.com/pp/pp.cgi/r425028300.

3. Technical Details

The epSOS ontology is created in OWL-DL[i]. Further technical details (e.g. the number of classes and object properties) are contained in the OWL file that can be downloaded (s. Section 2)

It was created using the Protégé ontology editor[ii]. Instructions on how to download Protégé in order to view the OWL file is given in section 7 of the Appendix.
A more detailed description of the role of the epSOS Ontology and its technical details can be found in the body of this deliverable.

4. State of the Implementation

The state of implementation of the ontology is the result of all development in Work Package 3.5 until the presentation of this deliverable. However, the ontology ought not be seen as a static artifact, but as an evolving entity that will expand and alter in accordance with its basic outline and strategy as given in Section 4 of Appendix D.

Due to the state of development of the value set catalogue and licensing issues on coding schemata certain aspect are not integrated in this version of the ontology, but development will progress on this artifact.

5. Development Strategy

5.1. Upper Ontology

The epSOS ontology is based on an Upper Ontology to ensure coherence of representation and facilitate harmonization with existing ontologies. IEEE's Standard Upper Ontology Working Group provides a quick intro into the beneficial effects of using Upper Ontologies[iii].

The Upper Ontologies chosen as the foundation of the epSOS Ontology is Basic Formal Ontology (BFO). Not only does the epSOS Ontology import BFO as starting point for the representation, but the epSOS Ontology is built in accordance to the basic principles and strategies of BFO. More detail on these can be found in the BFO documentation[iv].

5.2. Translation

The translations of the relevant terms, as defined by the Master Value Set Catalogue, are given in *rdf* labels and annotated with an language code, according to ISO 639-1.

All names in the ontology (class names, object property names) are in English and given in accordance with the recommendations in the Protégé manual[v].

5.3. Transcoding

Within the epSOS Ontology classes (and their labels) will not be matched with codes using annotations, but using an object property. The ontology contains the following two object properties for this purpose:

- isCodeOf
- hasCode

The first one matches any subclass of a specific medical code system (the specific medical code) to any other class in the ontology, depending on what the code stands for. The latter matches any medically relevant class to the specified code.

All code systems and all codes are subclasses to InformationObject in the epSOS ontology.

The individual codes have to be related to a Code System Version.

5.4. Multiple Inheritance

In the epSOS Ontology multiple inheritances are forbidden for primitive classes. Primitive classes are those, which are not defined by specifying necessary conditions for class membership in the equivalent class section in Protégé.

6. Semantic Representation in the epSOS Ontology

The epSOS Ontology is dealing with a number of mostly flat lists or hierarchies of medical or medical-relevant terms. The aim is to provide one joint system of semantic reference that can bridge the gap between the different underlying semantics of the term lists and vocabularies.

Thus, one of the most important focuses is to provide formal definitions for classes. The start of this enterprise can be clearly seen in the epSOS ontology. However, in the first step the focus lay on adding all relevant classes to the taxonomy providing the basis for the ontology. A number of formal definitions have been implemented, both necessary and contingent.

In the effort of building the ontology we have tried to provide a full definition of routes, dose forms and containers using necessary condition, yet this task had to be abandoned due to a tremendous increase of reasoning time running consistency checks over the ontology.

7. Consistency Check

The consistency of the epSOS ontology was checked using the Pellet reasoner[vi].

8. Viewing the epSOS Ontology

If you want to view the ontology, getting the taxonomy displayed and see all annotations and definitions in an easy to grasp way, we recommend you download the Protégé ontology editor.
Do that as follows:

1. Go to: http://protege.stanford.edu/download/download.html
2. If you are a registered user proceed to the download page, if not please register.
3. Once you registered go to the download page and follow the instructions. Make sure to download the newest version of the program.

The file can also be found on Project Place at the following link:
http://protege.stanford.edu/download/download.html.

Open the above OWL File (*epSOSonto1.9.owl* - **this file is an evolving entity and it is subject to change as the experience from testing and implementation will bring forth adjustments**) containing the epSOS Ontology using the downloaded program. Make sure to be connected to the internet, when you open the file, since the epSOS Ontology uses BFO as a direct import from the web.

8.1. Database Representation

The following figure shows a database representation and conversion.

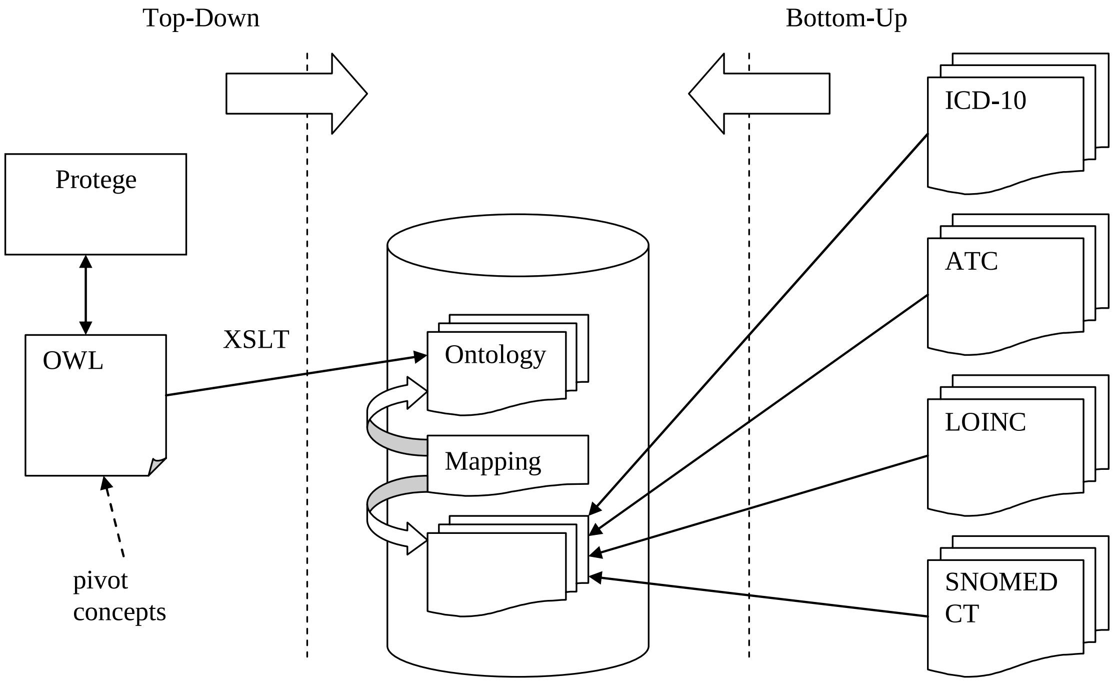


***Figure 2 –** Database conversion.*

9. Demonstrator

9.1. Aim

The aim of the demonstrator is to proof that medical information can be transmitted from one language/coding schema to another in a medical useful way (human and machine understandable). The demonstrator offers a strategy for seamless communication of medical content over borders.

.

9.2. Functionality

The demonstrator will translate a medical term/code from one language/coding scheme to another one. For example, if two languages are used such as German and Spanish, the English language will be used as "pivot language" in order to provide a correspondence between the two.

9.3. Elements

- **An owl implementation of the terminology**

  The owl file is used to populate the mapping table/database.

  The owl file is a means to guarantee future evolution of the terminology in size and the possibility to enable further use of the data beyond the epSOS scope.

- **<u>Mapping table/database</u>**

  This is the terminological basis that is read by the service and will provide the information wanted by the user. The mapping table is provided in a way that it is <u>populated in automatic fashion from the owl file.</u>

- **<u>Application</u>**

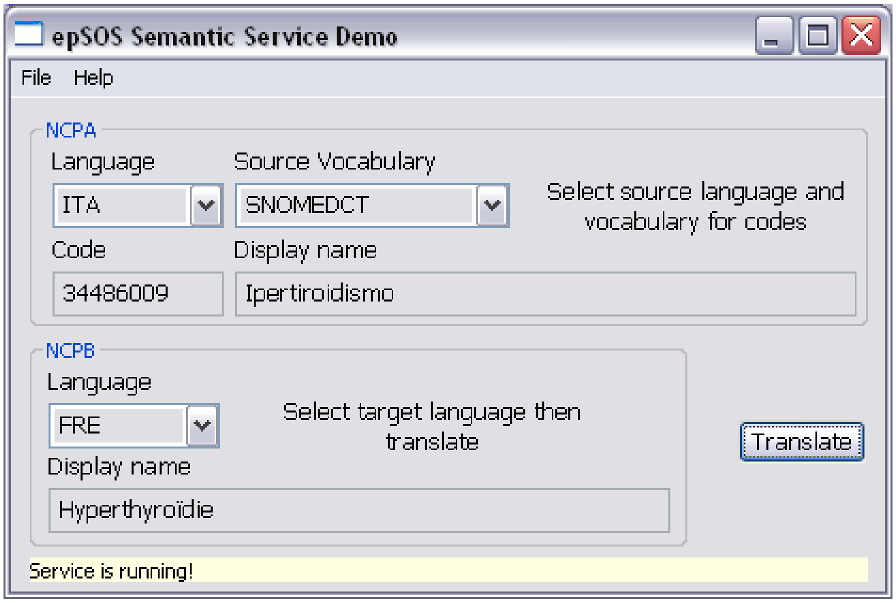
  An application will be provided that checks the information of equivalent terms/coding according to the query from the Mapping table/database.

- **<u>User interface</u>**

  The initial user interface will consist of four elements:

  - Input box which enables the user to enter a medical code
  - Output box which will display the translation result
  - Dropdown menu for source language
  - Dropdown menu for target language
  - Dropdown menu for the used code system

The following screenshot shows all described components:



NCPA

Language: ITA

Source Vocabulary: SNOMEDCT

Select source language and vocabulary for codes

Code: 34486009

Display name: Ipertiroidismo

NCPB

Language: FRE

Select target language then translate

Display name: Hyperthyroïdie

Service is running!

***Figure 2** – The user interface of the demonstrator*

- **<u>Use case description:</u>**

The user starts the application. At initialization time, the application connects to the translation service. If the connection is successful, then a message is displayed in the status bar confirming that the service is available.

The user then selects a coding system and a source language and enters a valid code of that coding system. When changing the focus of the code text box the display name of the code is filled in. By selecting a target language and clicking the “Translate” button the user receives the translation of the display name in the textbox.

Exceptions: if the user does not select the elements needed for translation or does not enter a code or the code is invalid for the selected source vocabulary then error messages are displayed in the text-boxes.

[1] http://www.w3.org/TR/owl-semantics

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

[3] http://suo.ieee.org

[4] http://www.ifomis.org/bfo

[v5]http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial-p4.0.pdf

[6] http://clarkparsia.com/pellet