Translating the Observational Medical Outcomes Partnership - Common Data Model (OMOP-CDM) electronic health records to an OWL ontology

Jean-Baptiste Lamy, Abdelmalek Mouazer, Karima Sedki, Rosy Tsopra

Translating the Observational Medical Outcomes Partnership - Common Data Model (OMOP-CDM) electronic health records to an OWL ontology

Lamy Jean-Baptiste[a], Abdelmalek Mouazer[a], Karima Sedki[a], Rosy Tsopra[b,c,d]

[a] *Université Sorbonne Paris Nord, LIMICS, Sorbonne Université, INSERM, UMR 1142, F-93000, Bobigny, France*
[b] *INSERM, Université de Paris, Sorbonne Université, Centre de Recherche des Cordeliers, Information Sciences to support Personalized Medicine, F-75006 Paris, France*
[c] *Department of Medical Informatics, Hôpital Européen Georges-Pompidou, AP-HP, Paris, France*
[d] *INRIA Paris, 75012 Paris, France*

Abstract

The heterogeneity of electronic health records model is a major problem: it is necessary to gather data from various models for clinical research, but also for clinical decision support. The Observational Medical Outcomes Partnership - Common Data Model (OMOP-CDM) has emerged as a standard model for structuring health records populated from various other sources. This model is proposed as a relational database schema. However, in the field of decision support, formal ontologies are commonly used. In this paper, we propose a translation of OMOP-CDM into an ontology, and we explore the utility of the semantic web for structuring EHR in a clinical decision support perspective, and the use of the SPARQL language for querying health records. The resulting ontology is available online.

Keywords:

Medical Records, Electronic Health Records, Biological Ontologies, SPARQL.

Introduction

Electronic health records (EHR) [1] lead to a major progress in the storage, the transmission and the standardization of clinical patient data. However, today, many EHR models and formats exist, each software vendor proposing its own. This heterogeneity is a huge problem for research studies that need to collect data from many EHR, but also for clinical decision support systems that need to be interfaced with many different EHR.

In the last ten years, OMOP-CDM (Observational Medical Outcomes Partnership - Common Data Model) [2] from the OHDSI (Observational Health Data Sciences and Informatics) community emerged as a common and simple EHR model, used to structure clinical data extracted from various other EHR, in order to facilitate clinical research studies. This model is proposed as a relational database schema. A study showed that OMOP-CDM has a higher content coverage than three other similar data models [3]. More recently, OMOP-CDM has been considered for clinical decision support [4, 5].

In parallel, formal ontologies and the semantic web [6] have emerged as a standard for the formalization of medical knowledge. Ontologies permit formal reasoning but also facilitate the reuse of the data and the knowledge. In particular, the interest of ontologies for validating EHR models have been shown in the literature [7], and ontologies are commonly used in decision support systems.

In this paper, we propose an OWL translation of the OMOP-CDM model, and we explore the utility of the semantic web for structuring EHR in a clinical decision support perspective. Our objective is not to maintain a full compatibility with OMOP-CDM database, but rather to structure an EHR as a formal ontology, grounding on the experience of OMOP-CDM. Consequently, we will focus on the clinical part of OMOP-CDM, and we will not consider the vocabulary part, because ontologies offer native support for structuring hierarchical terminologies. We will also consider the use of the SPARQL language for querying health records, and compare it to the SQL language.

Material and methods

Material

We used OMOP-CDM version 6.0 [2]. In OMOP-CDM (Figure 1), patients and healthy volunteers are represented by the Person table. Each Person may have zero, one or several Visit Occurrence, *e.g.* visits to a GP or hospital stays. Each Visit may be associated with some diagnoses (Condition Occurrence), tests (Measurement), medical procedures (Procedure Occurrence), drug prescriptions (Drug Exposure), etc. A higher level of abstraction, Eras, is also provided, for facilitating epidemiological studies. An Era groups one or more similar time periods in a single entity; *e.g.* if a patient was prescribed metformine for 3 months, and then after 3 months, metformine was prescribed again, there are two Drug Exposures (one per prescription), but a single Drug Era. OMOP-CDM provides procedures for computing Eras from the Drug Exposures and Condition Occurrences. Both Eras and lower-level entities (Condition Occurrence, Drug Exposure,...) are associated with a concept from a medical terminology.

We used the Python programming language for parsing OMOP-CDM specification and generating the OWL ontology, with the Owlready ontology-oriented programming module [8, 9, 10].

Translating the database model to OWL

We translated the OMOP-CDM database model into an OWL ontology, using a automatic Python script. Each table was translated into a class, each field corresponding to an identifier

into an object property, and each non-identifier field into a data property. SQL datatypes were translated into XML Schema datatypes and assigned to the range of data properties. We also added universal class restrictions, and existential class restrictions for fields marked as required in the OMOP-CDM model. The translation may seem rather straightforward, but two difficulties were encountered.

First, contrary to SQL relational databases, OWL ontologies support inheritance. In the OMOP-CDM model (Figure 1), there are some obvious situations where inheritance could be used, *e.g.* both Person and Provider are humans, and share identical attributes, such as gender or year of birth. Similarly, many OMOP-CDM tables have attributes related to time: the date of an event (*e.g.* a Condition Occurrence) or the start and end date of a duration (*e.g.* a Drug Exposure).

We manually added superclasses, such as Base Person, Event or Duration. Then, we wrote an automatic procedure in Python that moves any attribute present in all the subclasses of a superclass to that superclass. For instance, since Base Person has two subclasses, Person and Provider, and since both Person and Provider have a gender attribute, then the gender attribute will be moved to the Base Person class.

Second, in the OMOP-CDM model, the direction of relation is often dictated by the relational database model, *e.g.* the relation between Person and Drug Era is in the Drug Era Person direction, through the "person_id" field, because each Drug Era is associated with a single Person (*-1 relation) while a Person may be associated with several Drug Era (1-* relation), and the relational model supports only *-1 relations. On the contrary, OWL ontologies support both types of relations. Here, we found more intuitive to reverse the relation, in order to start from the Person and then to obtain his Drug Era *via* the has_drug_era relation. This places the patient at the center of the model.

Moreover, in the database model, fields are defined within a given table, *e.g.* the "person_id" field in the Drug Era table is distinct from the "person_id" field in the Condition Era table, despite they share the same name. This permits to search for the Drug Era associated with a given patient, using the "person_id" field in the Drug Era table. On the contrary, in ontologies, properties are first-order entities, independent from classes. Thus, if we create a "has_person" object property (corresponding to "person_id" but named using ontology conventions), it will be shared by all Era classes. If we search for the Drug Era associated with a given patient, the "has_person" property will thus return all Era, and thus we need to select only those that are Drug Era. In a SPARQL query, this requires two RDF triple patterns:

```
SELECT ?drug_era {
    ?drug_era a DrugEra .
    ?drug_era has_person <patient_x> .
}
```

If we reverse the direction of the relation, leading to the "has_drug_era" property, we can now select specifically the Drug Era associated with a given patient, simplifying the query as follows:

```
SELECT ?drug_era {
    <patient_x> has_drug_era ?drug_era .
}
```

Therefore, we manually defined a list of relation to reverse, and then an automatic procedure in Python for reversing those relations.

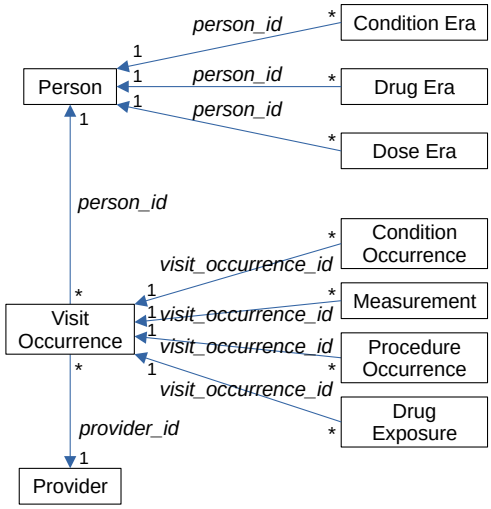

Figure 1– UML diagram showing the main tables and relations in OMOP-CDM.

Importing data

We imported the sample dataset with 1,000 patients from CMS SynPUF, proposed on the OMOP-CDM website. When importing the data, terminological concepts were mapped to UMLS. We used the UMLS import functions from Owlready2 to extract the terminologies present in OMOP from UMLS version 2020AA, and to translate them to ontologies. In these ontologies, concepts are represented by classes, in order to allow inheritance between concepts. When a concept is present in the OMOP data, the corresponding class is instantiated. For instance, if a patient has a Condition Occurrence associated with the "59621000" SNOMED CT code for Essential hypertension, we create an instance of the "59621000" class and we associate it with the Condition Occurrence. The original class can be obtained *via* the rdf:type relation between the instance and the class.

Querying the ontology

OMOP-CDM data are usually accessed *via* SQL queries, while SPARQL is commonly used for ontologies. We compared the length and the complexity of the queries in both languages, using various queries inspired by those proposed on the OMOP website (http://cdmqueries.omop.org/), or by the recommendations of the STOPP/START guidelines [11] for detecting potentially inappropriate prescribing in the elderly.

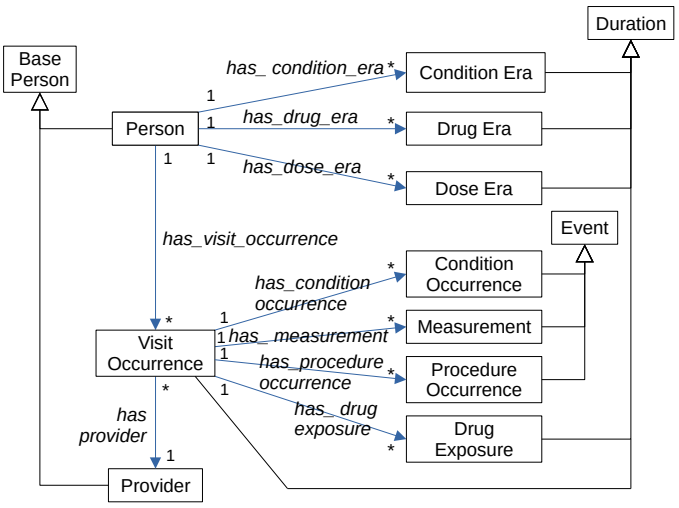

Figure 2– UML diagram showing the main classes and relations in the ontology translation of OMOP-CDM.

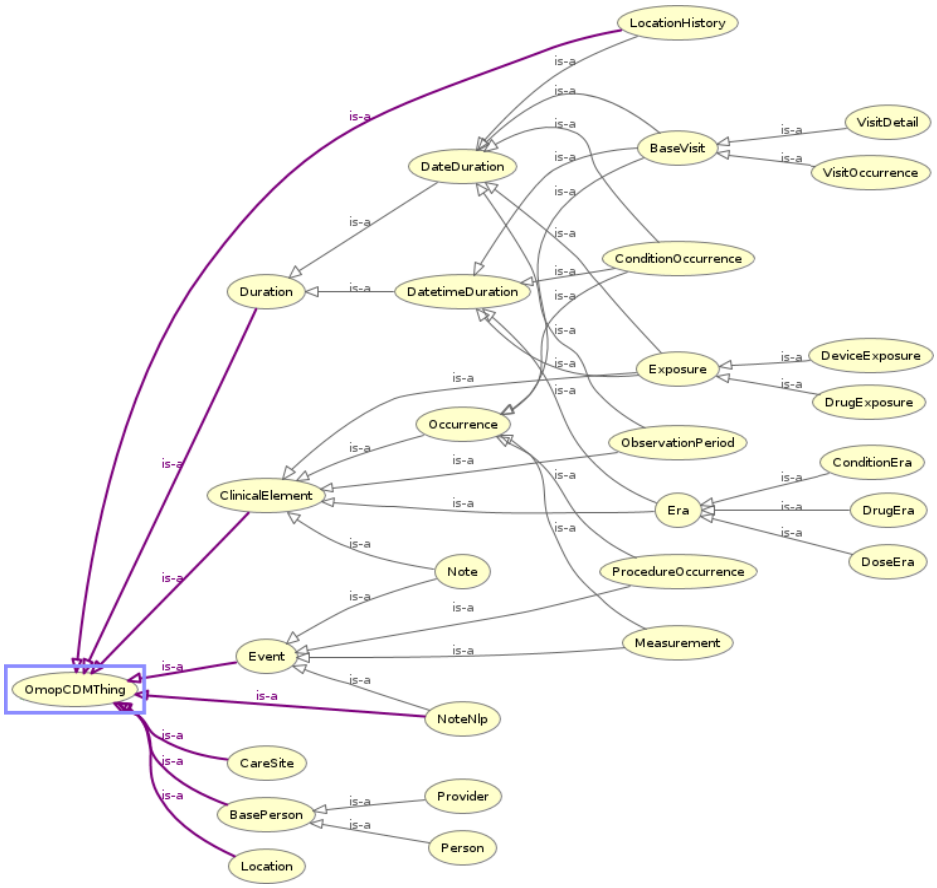

Figure 3– Class hierarchy in the ontology translation of OMOP-CDM.

Results

OMOP-CDM ontology translation

Figures 2 and 3 show the general model of the ontology. Compared to Figure 1, notice the presence of inheritance, but also the fact that the direction of many relations was changed. The ontology consistency was verified using the Pellet reasoners. It contains 49 classes, 226 properties and 5,016 RDF triples. After importing the OMOP-CDM sample data, we obtained a total of 10,795,221 RDF triples, including 3,311,359 for terminologies.

The ontology is available online[1] (Apache License 2.0, same license as OMOP-CDM), as well as the Python scripts that generated it from the OMOP-CDM CSV specifications[2] (GNU LGPL license). The script can be customized through global variables, *e.g.* to export only some parts of OMOP-CDM.

Using SPARQL for querying health records

Figure 4 shows an example of query, in both SQL and SPARQL, inspired by the examples found in the OMOP website. The SQL query is longer and more complex, due to the presence of nested queries. The first nested query (SELECT DISTINCT condition.person_id...) is motivated by the fact that "age" appears three times in the outer query (in the SELECT, the GROUP BY and the ORDER BY clauses). But SQL does not allow creating variables and to assign value to them. The nested query is thus here to avoid duplicating three times the formula for computing the "age" value. On the contrary, SPARQL allows the creations of variable, using the BIND statement, hence removing the need for the nested query.

The second nested query (SELECT DISTINCT descendant_concept_id...) is used to select all descendants of the desired concept in the terminology. Thanks to property path expressions, SPARQL offers an easier way to select descendants, *e.g.* using the expression "rdfs:subClassOf*", where "*" means that zero, one or more subClassOf relations must exist between a descendant concept and the original concept. In addition, "/" can be used in property path expressions to chain several relations, *e.g.* "has_concept/a/rdfs:subClassOf*/rdfs:label "Fracture of bone of hip region"" means that ?condition has for concept an instance of a class that is a descendant of a class associated with the "OMOP Hip Fracture 1" label.

Consequently, with SPARQL, nested queries are no longer required here. This arguably simplifies the query.

Using Owlready, the SPARQL query can be executed on the OMOP sample dataset (1,000 patients, about 10 million RDF triples) in about 0.31 second on a recent laptop computer. In contrast, the SQL query can be executed on the same dataset in about 0.35 second using PostgresQL. This suggests that SPARQL and ontologies are as efficient as SQL and relational databases.

The semantic web also allows linking the data with other data or knowledge very easily. For instance, the sample OMOP dataset use RXNORM for coding drug prescriptions, but the ATC terminology (Anatomical, Therapeutical Chemical classification of drugs) may be more practical when dealing with therapeutical classes (*e.g.* proton pump inhibitors, PPI, instead of specific active principles such as omeprazole), especially when implementing the rules found in clinical guidelines. With ontologies, it is easy to add relations with ATC in addition to the existing relations with RXNORM: when importing the data, we instantiated the RXNORM classes; we can state that the resulting instances also belong to the corresponding ATC classes by adding new RDF triples. On the contrary, when using database, one would require to add an extra table mapping RXNORM to ATC for adding support for the ATC terminology. This would complicate queries, with additional joints between the concept table and the mapping table.

Figure 5 shows two examples of rules extracted from the STOPP/START clinical guideline [11], and their implementation in SPARQL. We used SNOMED CT codes for disorders and ATC codes for drugs. The first rule detects a simple drug-disorder interaction between digoxin and heart failure. The second rule is more complex. It involves aspirin, which has 3 ATC codes; thus, we used a UNION clause for testing the 3 codes. Moreover, the rule should not be triggered when a PPI is prescribed concomitantly. Thus, we used a FILTER NOT EXISTS clause to verify the absence of a PPI, with conditions on the start and end dates of the two Drug Eras to verify the co-occurrence of the two treatments. Both rules were implemented as SELECT queries that returns the patients and the Drug Era that should be stopped.

1 http://www.lesfleursdunormal.fr/static/_downloads/omop_cdm_v6.owl

2 https://bitbucket.org/jibalamy/owlready2/src/master/pymedtermino2/omop_cdm/

Database with SQL:

```
SELECT gender, age, count(*) num_patients FROM
    ( SELECT DISTINCT condition.person_id, gender.concept_name As GENDER,
                  EXTRACT( YEAR FROM CONDITION_ERA_START_DATE ) - year_of_birth AS age
      FROM condition_era condition
      JOIN ( SELECT DISTINCT descendant_concept_id
             FROM vocabulary.relationship
             JOIN vocabulary.concept_relationship rel USING( relationship_id )
             JOIN vocabulary.concept concept1 ON concept1.concept_id = concept_id_1
             JOIN vocabulary.concept_ancestor ON ancestor_concept_id = concept_id_2
            WHERE relationship_name = 'HOI contains SNOMED (OMOP)'
              AND concept1.concept_name = 'Fracture of bone of hip region'
            ) ON descendant_concept_id = condition_concept_id
      JOIN person ON person.person_id = condition.person_id
      JOIN vocabulary.concept gender ON gender.concept_id = gender_concept_id
    )
GROUP BY gender, age ORDER BY gender, age
```

Ontology with SPARQL:

```
SELECT ?gender ?age (COUNT(DISTINCT ?patient) as ?num_patients) {
    ?patient omop_cdm:has_condition_era ?condition .
    ?condition omop_cdm:has_concept/a/rdfs:subClassOf*/rdfs:label
                                                  "Fracture of bone of hip region" .
    ?patient omop_cdm:has_gender/a/rdfs:label ?gender .
    ?patient omop_cdm:year_of_birth ?birth_year .
    ?condition omop_cdm:start_date ?start .
    BIND(YEAR(?start) - ?birth_year AS ?age) .
}
GROUP BY ?gender ?age ORDER BY ?gender ?age
```

Figure 4– A query for listing genders and ages of the patients having hip fracture, in SQL (top) and SPARQL (bottom).

When implementing STOPP/START rules, we found that SPARQL lacks the IN SQL keyword. This keyword allows testing whether a value is one of a set of given values. It is a shorthand for multiple OR conditions. IN is commonly used when several terms correspond to the desired concept, *e.g.* for testing the 3 codes for aspirin in the ATC terminology, one may use "aspirin IN ("BO1AC06", "AO1AD05", "NO1BA01")". On the contrary, SPARQL has no such keyword. Thus, in the same situation, we used UNION as seen above, which is much longer and less practical.

Finally, notice that the precision of the queries remains the same with SQL or SPARQL, both queries being semantically equivalent and returning the same results.

Discussion

OMOP-CDM was initially developed for gathering in a single model clinical data from heterogeneous sources, such as EHR from different vendors, in order to facilitate clinical research. However, clinical decision support is another situation where one may need to merge clinical data from different models, *e.g.* for a given patient, clinical data may be found in the EHR of the GP, but also in the EHR of the hospital and even in the pharmacy. In that situation, the use of OMOP-CDM seems a relevant option. Moreover, the high-level abstractions proposed in OMOP-CDM with Era are useful for clinical studies, but also for decision support. For instance, Drug Eras allow computing drug treatment durations, which are sometimes required for supporting decision (*e.g.* for rule START E2 in STOPP/START).

The use of ontologies for structuring EHR may not be relevant for clinical research, because of the huge volume of data involved, and the relational database format is well established today in that community. On the contrary, for decision support, the volume of data is often lower (after the eventual learning phase for machine learning-based systems), because decision support deals with a single patient at a time, and ontologies are frequently used.

In the literature, a previous tentative exists for translating the OMOP-CDM model to an ontology [12]. However, it does not focus on the latest version of the model (6.0), and the translation was limited to a raw conversion from database to OWL, without adding inheritance and restrictions as we did, nor reversing relations. Finally, since our translation is almost entirely automatic, and performed by a Python script, it will be easy to update the ontology for future versions of OMOP-CDM.

In the near future, we plan to use the proposed ontology for structuring heterogeneous clinical data in a decision support system for medication reviews. We also plan to implement tools for importing into the proposed ontology clinical data in standard formats such as HL7 and FHIR.

Conclusions

In this paper, we proposed an OWL translation of the OMOP-CDM relational database model for electronic health records. We successfully used the resulting ontology for importing the OMOP sample dataset. We also compared the use of the SQL and SPARQL language for querying EHR data. We found that SPARQL often permitted simpler queries, thanks to its ability to deal with recursion and to define variables, and thanks to the ease with which ontologies can be enriched and connected to other resources in the semantic web.

STOPP B1: Stop digoxin for heart failure with normal systolic ventricular function (no clear evidence of benefit).

```
SELECT ?patient ?drug_era { # SPARQL query for rule STOPP B1
    ?patient omop_cdm:has_drug_era ?drug_era .
    ?drug_era omop_cdm:has_concept/a/rdfs:subClassOf* atc:C01AA05 .
    ?patient omop_cdm:has_condition_era/omop_cdm:has_concept/a/rdfs:subClassOf*
snomed:84114007.
}
```

STOPP C2: Stop aspirin with a past history of peptic ulcer disease without concomitant PPI (proton pump inhibitor).

```
SELECT ?patient ?drug_era1 { # SPARQL query for rule STOPP C2
    ?patient omop_cdm:has_drug_era ?drug_era1 .
    ?drug_era1 omop_cdm:has_concept/a ?aspirin .
          { ?aspirin rdfs:subClassOf* atc:B01AC06 . }
    UNION { ?aspirin rdfs:subClassOf* atc:A01AD05 . }
    UNION { ?aspirin rdfs:subClassOf* atc:N01BA01 . }
    ?patient omop_cdm:has_condition_era/omop_cdm:has_concept/a/rdfs:subClassOf*
                                                                  snomed:13200003.
    FILTER NOT EXISTS {
        ?patient omop_cdm:has_drug_era ?drug_era2 .
        ?drug_era2 omop_cdm:has_concept/a/rdfs:subClassOf* atc:A02BC . # PPI
        ?drug_era1 omop_cdm:start_date ?start1 .
        ?drug_era1 omop_cdm:end_date ?end1 .
        ?drug_era2 omop_cdm:start_date ?start2 .
        ?drug_era2 omop_cdm:end_date ?end2 .
        FILTER(?start1 < ?end2 && ?start2 < ?end1) .
    }
}
```

Figure 5– Two rules extracted from the STOPP/START clinical guideline, and their translation into SPARQL.

Acknowledgements

This work was funded by the French Research Agency (ANR) through the ABiMed project [grant number ANR-20-CE19-0017-02].

References

[1] Kataria S, Ravindran V. Electronic health records: a critical appraisal of strengths and limitations. *The journal of the Royal College of Physicians of Edinburgh*. 2020;50(3):262–268.

[2] Reich C, Ryan P, Belenkaya R, Natarajan K, Blacketer C. OMOP Common Data Model Specifications; 2018.

[3] Garza M, Del Fiol G, Tenenbaum J, Walden A, Zozus MN. Evaluating common data models for use with a longitudinal community registry. *J Biomed Inform.* 2016;64:333–341.

[4] Unberath P, Prokosch HU, Gründner J, Erpenbeck M, Maier C, Christoph J. EHR-Independent Predictive Decision Support Architecture Based on OMOP. *Applied clinical informatics.* 2020;11(3):399–404.

[5] Gruendner J, Schwachhofer T, Sippl P, Wolf N, Erpenbeck M, Gulden C, et al. KETOS: Clinical decision support and machine learning as a service - A training and deployment platform based on Docker, OMOP-CDM, and FHIR Web Services. *PloS one.* 2019;14(10):e0223010.

[6] Schulz S, Jansen L. Formal ontologies in biomedical knowledge representation. *Yearb Med Inform.* 2013;8:132–46.

[7] Martínez-Costa C, Schulz S. Validating EHR clinical models using ontology patterns. *J Biomed Inform.* 2017;76:124–137.

[8] Lamy JB. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. Artif Intell Med. 2017;80:11–28.

[9] Lamy JB. Ontologies with Python. Apress; 2021.

[10] Lamy JB. Ontology-Oriented Programming for Biomedical Informatics. *Studies in health technology and informatics (STC)*. 2016;221:64–68.

[11] O'Mahony D, O'Sullivan D, Byrne S, O'Connor MN, Ryan C, Gallagher P. STOPP/START criteria for potentially inappropriate prescribing in older people: version 2. *Age Ageing.* 2015;44(2):213–8.

[12] Pacaci A, Gonul S, Sinaci AA, Yuksel M, Laleci Erturkmen GB. A Semantic Transformation Methodology for the Secondary Use of Observational Healthcare Data in Postmarketing Safety Studies. *Frontiers in pharmacology.* 2018;9:435.

Address for correspondence

Jean-Baptiste Lamy <jean-baptiste.lamy@univ-paris13.fr>, Bureau 149, UFR SMBH, 74 rue Marcel Cachin, 93017 Bobigny cedex, France