Semantic exploitation of persistent metadata in engineering models: application to geological models

Laura Silveira Mastella*, Yamine Aït-Ameur†, Stéphane Jean†, Michel Perrin* and Jean-François Rainaud‡
*École des Mines de Paris, 60, boulevard Saint-Michel, 75272 Paris, France
{laura.mastella, michel.perrin}@ensmp.fr
†LISI/ENSMA and University of Poitiers, BP40109, 86961 Futuroscope, France
{yamine, jean}@ensma.fr
‡Institut Français du Pétrole, 1-4 Av. de Bois-Préau, 92852 Rueil-Malmaison, France
j-francois.rainaud@ifp.fr

***Abstract*—Engineering models are computer-based models that enclose technical data issued from engineering domains. Those models usually implicit many of the details required to understand and interpret the data. In this context, integrating the results of models and querying the heterogeneous information is a challenge.**

In order to address the issue of handling heterogeneous models, we propose to *annotate the engineering models* with concepts of the ontologies of the specific engineering domains. We describe here the proposal of a *semantic annotation meta-model,* which extends an ontology-based database architecture with constructs that allow to tag engineering models using ontology concepts.

This work is inspired from a petroleum engineering case study, and we validate our approach by presenting an implementation of this case.

***Index Terms*—Data integration and interoperability, Ontology-based databases, Meta-modelling, Semantic annotation, Petroleum engineering models**

Paper category: technical solution.

I. Introduction

Engineering models are computer-based models often used to run simulations or perform verification and validation over technical data. The heterogeneity of these models, their representations in logical models that are close to low-level computer representations and the verbose format in which they are documented make the integration task a challenge. Consequently, the exploitation of such engineering data models and the corresponding data becomes a difficult work to engineers that are not trained for such activities. Professionals should be able to exploit these models with their own knowledge models, instead of spending lot of effort in translating their knowledge to the current computer representation of this models.

Examples of activities that rely on various engineering models are: simulation, validation, verification, design, storage, exchange; within several engineering domains, ranging from civil engineering, aeronautics, environment, agriculture, automotive industry to social organizations. All these activities and domains are the subjects of a huge amount of heterogeneous models and data.

When dealing with complex information systems, the problem of how to provide access to heterogeneous information already appeared decades ago. The typical solution for information integration is to provide a uniform interface to a collection of heterogeneous information sources, giving users the illusion that they are a centralized and homogeneous information system. However, the models do not inter-operate, because there is no semantic added to their objects and the expression of model mappings is only hard coded.

The last decade has seen the emergence of the use of *ontologies*, in order to provide an explicit and formal definition of specific domains [1]. With the development of ontologies for specialized engineering domains, it becomes possible to access the engineering models at the semantic level, through the ontologies concepts.

Our work is conducted in the domain of petroleum reservoir engineering, in particular, the activity of *oil & gas reservoir modelling*, performed by oil & gas companies (see Section III). Considering this activity, petroleum engineers and geoscientists rely on three-dimensional representations of the earth underground (called *reservoir models* or *oilfield models*) to take important decisions about oil-reservoir operations. The reservoir model is the final result of the oilfield modelling workflow (Fig. 1).

The definition of an ideal working platform has been the major concern of petroleum exploration software vendors for long. In Chevron company's use case [2] they claim that still a large amount of heterogeneous data is generated every day from multiple sources such as seismic data, well data, drilling data, transportation data, and marketing data. In order to deal with the flood of information, as well as the heterogeneous data formats of the data, a new approach for information search and access is necessary.

The end-users of this community aim to be able to retrieve and re-use information that are created in the various areas of expertise within reservoir modelling and represented in diverse oilfields models. In summary, the petroleum exploration domain is still looking for a proper solution for the

interoperability problem.

The proposal of this work for addressing the semantic interoperability problem in engineering domains such as the petroleum exploration activity is an approach based on *semantic annotation of engineering models*. Semantic annotation is a current Web Semantic technique for adding knowledge to resources by means of semantic tags (see Section II). We envisage the use of semantic annotation for: (i) making explicit the expert knowledge enclosed in the model and (ii) interrogating raw data using semantic concepts represented by domain ontologies. This approach is explained in Section IV.

To carry out this approach, we consider to use an Ontology-Based Database (OBDB), which allows to store data and ontologies in a same database. We introduce this concept in Section II-A. In Section VII we present an implementation for the case study in oil & gas reservoir modelling. Some initial results from this domain are presented that illustrate how this approach enables to extract emergent semantics from engineering models.

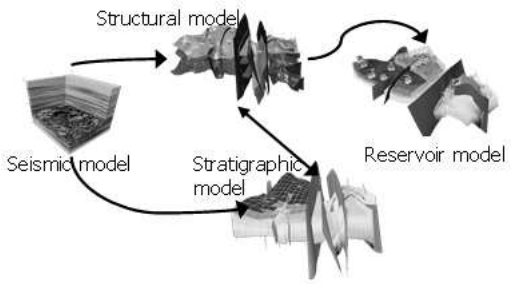


Fig. 1. Oilfield modelling workflow

II. Background

In this section we introduce the basic notions of ontology-based databases (OBDB), engineering models and semantic annotation, in order to provide the fundamental background to our approach.

A. Ontology-Based Databases (OBDBs)

Ontology-Based Databases (OBDB) are database architectures that deal with the problem of persistence of ontologies while taking advantage of the characteristics of databases (scalability, safety, capability to manage a huge amount of data, etc.) [3]. We summarize in this section a comparative analysis of OBDB systems (available in [4]) and their capabilities to solve the interoperability issues.

- **Type 1 OBDBs**. Information is represented in a single schema composed of a unique triple table `(subject, predicate, object)`, which may be used to represent both ontology descriptions and instance data. This approach raises, however, serious performance issues when queries require many self-joins over this table. Example of type 1 OBDBases are **3Store** [5] and **Jena** [6] frameworks.
- **Type 2 OBDBs**. They store separately ontology descriptions and instance data in two different schemas. The schema for ontology descriptions depends upon the ontology model used to represent ontologies (e.g., RDFS[7], OWL[8], PLIB[9]). For instance data, different schemas have been proposed: *vertical table* can be used to store instance data as triples; or a *binary representation* can be used, where each class is represented by a unary table and each property by a binary table; also *table per class representations* have been proposed where a table having a column for each property associated with value for at least one instance of a class is associated to each class. Example of type 2 OBDBases is **Sesame** [3] framework and **ONTOMS** [10].
- **Type 3 OBDBs**. A new type of OBDB, OntoDB [11], proposes to add another layer to type 2 OBDBs. This schema called *meta-schema* records the ontology model into a reflexive meta model. For the ontology schema, the meta-schema plays the same role as the one played by the system catalog in traditional databases. The meta-schema allows: (i) generic access to the ontology, (ii) support of evolution of the used ontology model, and (iii) storage of different ontology models (OWL, DAML+OIL[12], PLIB, etc.). Indeed, OntoDB is declined on three different conceptual levels, which allows to differentiate the instances, from their structure and from their meta-model.

In the context of our work, we have two fundamental criteria for choosing an OBDB. Firstly, the OBDB must have the capability to manage a huge amount of information, since an important quantity of data is currently available in Petroleum Industry. Secondly, the support of evolution of the OBDB meta-model is important, since we need to extend this architecture to represent other data containers than the ontology meta-model (i.e., an *annotation* meta-model). As a consequence, we have chosen OntoDB architecture, which fulfills this two criteria, as we will detail further on.

B. Engineering Models

Engineering development processes involve a set of activities which manipulate data related to an engineering domain. Engineering data can be expressed in various types of models: database tables, programming units (such as classes of `Java` or `C` programming languages), mathematical expressions, and so on. These representations lead to heterogeneous models that need to be reduced. As a consequence, engineering models produce a huge quantity of instances, the raw data. When integrated, these data allow the emergence of new knowledge that is relevant for the engineers. In several areas, this integration is still made by the engineers themselves and is neither formalised nor computerised. The engineering area that is used as case study for this work is the Petroleum Industry. In this area, engineering models are depicted by three-dimensional representations of the earth underground, and the interpretation of the different petroleum engineering models demands background knowledge and context from the domains users.

C. Semantic Annotation

At the moment, there are several frameworks and tools that allow to create semantic annotations over resources (web pages, textual documents, multimedia files). From a comparative analysis of several semantic annotation projects, available in [13], we understand that most of those frameworks and tools (such as SHOE Knowledge Annotator [14]), still rely on knowledge in HTML pages, XML documents or in other textual resources. Several works were also proposed concerning the annotation of images [15] and, more recently, concerning the annotation of the Web-services[16]. Tools like Vannotea[17] also propose to add meta-data to multimedia resources such as audio and video files.

However, despite the significant number of tools and frameworks that provide ontology-based annotation, none of the annotation tools proposed so far enable the *annotation of engineering models* (or, more generally, computer-based models). Concretely, there is no technique allowing to complete those models by formal comments or explanations, or to attach more semantics to the technical data produced by the modelling tools. Indeed, much of a company's knowledge can be found in text repositories, such as projects documentation and reports. Nevertheless, we cannot deny that engineering models are the result of their builder's expertise, and this is some strategic knowledge that cannot be lost. In the next section we present the case study of petroleum engineering used to illustrate our approach.

III. A Case Study in Petroleum Engineering

In a typical workflow for oil & gas reservoir modelling, engineers and scientists from various geosciences fields provide interpretations about the data prospected from the earth underground. Those professionals are experts in a specific field within the several disciplines involved in the workflow. As a consequence, they are competent to use data and tools only from their own specific domain. Therefore, it is challenging for this community to retrieve and re-use information issued from various fields of expertise and represented differently across domains, using different modelling tools.

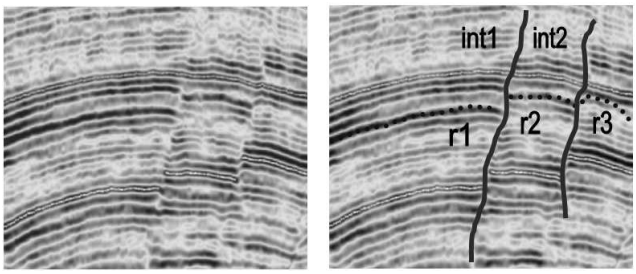
In this context, we face two main practical problems. (i) Originally, the users do not have means of preserving the interpretations made during the construction of the oilfield models. Documents in natural language that explain their interpretation are merely produced. (ii) There is no software framework that allows to making queries over the data and interpretation contained in those models. As a consequence, to formulate queries about the oilfield models, the user must know the structure of the data that constitutes the model.

The Geological Seismic Interpretation

The case study in oilfield modelling considered in this work focus an specific activity that occurs in the initial phases of the workflow. The *Geological Seismic Interpretation* aims at identifying the first geological objects from the raw data. The interpreter (a geoscientist) observes the input raw data (notably, *seismic images*, which are pictures of underground structures – Fig. 2a), and gives an interpretation about it, recognizing horizontal and vertical traces in the image and identifying them as as *Reflectors* and *Reflector Interruptions*, respectively. As illustrated in the Fig. 2b, the user has identified and pointed in the image some *reflectors* (`r1`, `r2`, `r3`) and *interruptions* (`int1`, `int2`). The spatial coordinates and other informations about the identified objects are, then, saved in cloud-of-point files (whose most common format is named *XYZ file*), which are outputted by the seismic interpretation tool. During all the task, it is not feasible for the users to make notes and comments about their interpretations, nor to establish relations between the data files generated. The informations about the geological objects are stored using the chosen data format.

One of the expectations of the users of this community is to be able, within any phase of the workflow, to ask questions related to the geological objects. The main issue is to determine how to formulate those queries without having to know the internal data structure of the objects. A typical question would be for example to determine *all the horizons that were interpreted within a reservoir study*. An *Horizon* is a geological concept that stands for various objects. Geoscientists can designate a *Reflector* using the term *Horizon*. So this query requires that the system knows that the term *Horizon* is a general case of the term *Reflector*, and also knows which files represent the information that constitutes the *Reflectors*. Currently it is not possible to answer this type of question, since we have no way of correlating the data stored in technical formats with their meanings in geological terms. The geological identification of an object lies in the head of the geoscientist who made the interpretation.

Another difficulty is that there is no knowledge-base shared among all the modelling tools. Such knowledge-base would indeed make it possible to (i) preserve the interpretations made by the user during the construction of the oilfield model; and, consequently, to (ii) possibly recover the previous interpretations. We explain in the next section the proposed approach for addressing those issues.



(a) Seismic Image (b) Interpreted Seismic

Fig. 2. Seismic interpretation activity.

IV. The Proposed Approach

In order to make the expert's knowledge explicit in engineering models, we propose to annotate the models using domain ontologies. This *engineering models annotation* process must be able to represent the following elements: (i) ontologies and their instances; (ii) engineering models and their data and (iii) annotations of the engineering models, which establish links between ontologies and engineering models.

(i) The knowledge about geosciences fields was acquired with experts and represented as *domain ontologies*. The geosciences ontologies are stored in an ontology-based database.

(ii) We applied, then, meta-modelling techniques to represent the engineering model's data *as instances of its meta-data*. The actual data schemas and their access information (such as file names) were represented as meta-data. As a result, it is possible to address the problem of retrieving the real data. In order to persist the access information of engineering models, it is necessary to provide a meta-model and store these information in the same database as the ontologies. But it is not desirable to represent the engineering meta-data using constructs for ontologies, since we do not expect to have, for engineering meta-data, the same features that are currently proposed to ontologies, such as, subsumption between concepts. The constructs of engineering meta-data are different from the constructs used to define ontologies (such as `owl:Class` in OWL language), because these entities have different nature. For these reasons, in order to persist engineering meta-data along with ontologies, we decided to enrich the original set of constructs for building ontologies with *Engineering Meta-model* constructs. The Engineering Meta-model is actually the minimum necessary set of the features that allows a uniform description of these models (file name, identification, main composite objects, etc.). The main added constructs for building engineering meta-data are `#DataElement` and `#DataAttribute`.

(iii) Finally, we provided a means for linking engineering meta-models to the concepts of ontologies. In this context, considering that these models keep the users' interpretation about data, and that each user may have a different opinion, it is expected that the annotations of these models can record the different interpretations supplied by different end-users. It means that, for the same dataset, there will probably be different annotations expressing each user's opinion, and that must be uniquely identified. Another requirement is that one user can annotate several data elements with one ontology concept, and vice-versa. There is need, then, for an N-to-N relationship for the annotation elements. Therefore, in this approach, the annotation becomes a top-level entity, separated from the ontological concept and from the entity being annotated. The annotation entity has also its own attributes, such as creation date, author name and version information.

It follows that we also defined a *Meta-model for Annotation*. The construct `#Annotation`, creates a link between the construct of ontology concepts (it varies depending on the ontology model) and the construct `#DataElement`, by means of the relations `#annotates` and `#isAnnotatedBy`. The meta-models are illustrated in Fig. 3 as a UML class diagram[1].

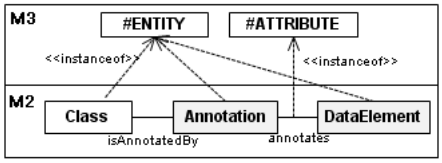


Fig. 3. Constructs of Engineering and Annotation Meta-models.

Next section introduces the OBDB architecture supporting the persistance of the described models and their instances.

V. The OntoDB ontology-based database

The OntoDB system [11] is an ODBD architecture to support evolutions of the ontology schema. OntoDB makes use of meta-modelling techniques and propose separation of modelling layers. OntoDB allows, thus, to represent the different constructors of existing ontology models (e.g, RDF, OWL, PLIB), enabling to store ontologies specified in different ontology languages, and to separate the instances, from their data structure and from their meta-model. We explain OntoDB's architecture as follows.

A. The OntoDB architecture

OntoDB is implemented on top of PostgreSQL open source database system[2], and consists of 4 parts (Fig. 4).

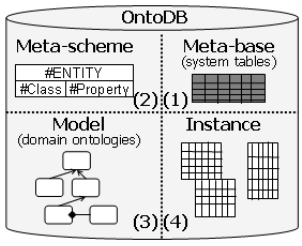


Fig. 4. OntoDB Architecture.

(1) The part shown in Fig. 4(1) is the traditional part available in all DBMSs, namely the *Meta-base*, which contains the system tables used to manage all the data contained in the database.

The other parts of the architecture of OntoDB present similarities with the OMG's Meta Object Facility, which propose four superposed layers that represent all levels of abstraction of information: meta-metamodels (M3 level), metamodels (M2 level), metadata (M1 level) and data (M0 level).

(2) The *Meta-schema* part (Fig. 4(2)) corresponds to the MOF levels M3 and M2. The meta-schema stores the upper level constructs, named `#ENTITY` and `#ATTRIBUTE`, which, from the database point of view, correspond to two tables.

[1]The proposed meta-models are represented here as the M2 layer in the OMG's Meta Object Facility (MOF) four-layer architecture (http://www.omg.org/mof/).

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

It also contains the traditional layer available in all OBDBs: the meta-model for building ontologies. OntoDB provides the ontological constructs `#Class`, `#Property`, `#DataType` and others, which corresponds to instances of `#ENTITY`. If the Meta-schema of OntoDB is extended with new constructs, they are added as new lines in `#ENTITY` table, as illustrated in the Fig. 5.

#ENTITY		
ID	Name	SuperEntity
1	Resource	
2	Class	1
3	Property	1
4	DataType	
5	owl:Class	

#ATTRIBUTE			
ID	Name	Domain	Range
1	name	1	4
2	domain	3	2
…	…	…	…

Fig. 5. Tables in OntoDB's Meta-Schema

The ontological constructs of OntoDB allows to represent the main constructors of existing ontology models (e.g, OWL, PLIB). Thus, in OntoDB it is possible to store ontologies specified in different ontology-languages. The construct `#Class` stands for ontological entities that represent categories of objects (such as `owl:Class` from the OWL language or `PLIB:Class` from the PLIB model). The construct `#Property` stands for the characteristics of ontological entities and the relations they have among them (such as `rdf:Property` from the RDF language, or `owl:DataProperty` from the OWL language or `PLIB:property_det` from the PLIB model).

(3) The third part of OntoDB, named *Model*, corresponds to the MOF level M1, where we describe the structure of the domain ontologies, that is, the concepts and relations from some domain (Fig. 4(3)).

(4) The instances are in the MOF level M0, the *Instance* part (Fig. 4(4)). Ontology-based data (instances) are represented in OntoDB using an horizontal approach; one table is created for each ontological class; its columns consists of a subset of the class applicable properties (i.e., that include the class in their domain), namely those that are used by at least one instance of the class. This representation scales well when numerous properties per instances are used.

B. The OntoQL Language

In order to exploit the OntoDB OBDB, the *OntoQL* language has been proposed by Jean et al. [18]. The OntoQL language has a syntax similar to SQL, and provides Data Definition, Manipulation and Query Languages at the three layers of OntoDB, from the logical level to the meta-schema level. Consequently, it is possible to extend the meta-schema (M2) level of OntoDB by using the OntoQL Ontology Definition Language (see the extension of OntoDB in Section VI).

The concepts of an ontology are created in OntoQL using the object-oriented constructors defined in the M2 level of OntoDB (`#Class` and `#Property`).

OntoQL provides also a *Query Language* that can be used to interrogate both the *meta-schema* and the *ontology contents*. To query the meta-schema, we use some special constructors to directly address the meta-elements, such as the '#' symbol in the query Q1, which selects the names of all classes of the database.

```
Q1: SELECT #name FROM #Class
```

```
Q2: SELECT age FROM Person
```

Querying the content is similar to a classical SQL query and the properties will not be prefixed by the '#' symbol (example: query Q2). OntoQL enables also to query both meta-schema and contents in the same query, which offers the capability of uniform manipulation of ontology and instances within the database.

In the following section we illustrate how we extended the meta-schema of OntoDB in order to implement the case study in oilfield modelling.

VI. Extending OntoDB with Annotations and Engineering Model

The implementation considered for our approach must be able to represent the following elements: (i) engineering models and their data; (ii) ontologies and their instances and (iii) annotations of the engineering models. We present here how to use OntoQL to extend the core-model of OntoDB to include constructs of the **Engineering Meta-Model** and of the **Annotation Meta-Model** proposed in the previous section. They are created by instantiating the `#ENTITY` construct.

The first part of our proposal is an *Engineering Meta-model*, which defines the entities that are the building blocks that allow to represent any data artifact used in engineering models. The Fig. 6 shows a UML representation of the *Engineering Meta-model*.

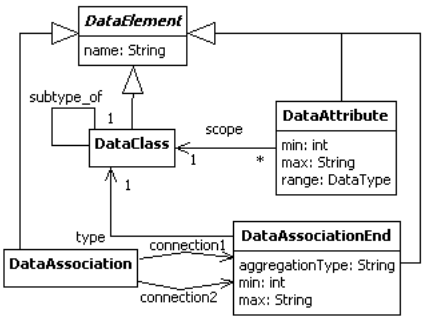


Fig. 6. Meta-model constructs for engineering models

The entity `#DataElement` is the abstract super-entity. The entity `#DataClass` can define hierarchies and have `#DataAttributes`. We can also have `#DataAssociations`, which relate two `DataClasses`. Those entities are created in OntoDB using OntoQL expressions, such as statements Q3 and Q4.

```
Q3: CREATE ENTITY #DataElement
(#name STR)
```

```
Q4: CREATE ENTITY #DataClass UNDER
#DataElement (#subtype_of REF (#DataClass))
```

The next step was to define constructs for building Semantic Annotations, which correspond to links between the *Ontology meta-model* and the *Engineering meta-model*. The Fig. 7 shows a UML representation of the Annotation entities created in OntoDB.

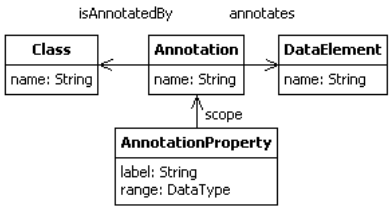


Fig. 7. Meta-model constructs for Annotation

The entity `#Annotation` defines a link between the entity `#Class` (the OntoDB construct for ontology classes) and the entity `#DataElement`, by means of the relations `#annotates` and `#isAnnotatedBy`. Those entities are created in OntoDB using OntoQL expressions, such as the statement Q5.

```
Q5: CREATE ENTITY #Annotation
(#annotates REF(#DataElement), #isAnnotatedBy
REF(#Class))
```

In Fig. 8 we show a zoom of the meta-schema part of OntoDB (from Fig. 4(2)) after its extension with the new constructs.

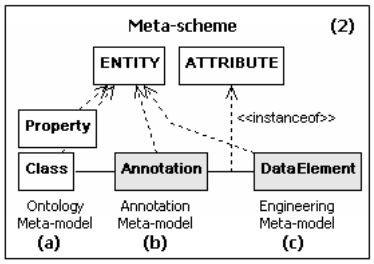


Fig. 8. Zoom of extended OntoDB's meta-schema

In the following section we illustrate how the extension of the meta-schema of OntoDB was used to implement the case study in oilfield modelling.

VII. Application to the Geo-Modelling Case Study

In the Section III we presented a case study in petroleum engineering in which we should be able to make queries about data issued from oilfield models using concepts of the knowledge-level.

After having extended the M2 level of OntoDB with the two new meta-models, we need to represent: (i) how data is identified using geosciences concepts, (ii) how data is actually structured by the oilfield modelling applications, (iii) the annotation elements over the oilfield data.

A. Representing Geosciences Ontologies

Significant efforts have been developed by various geological surveys for issuing ontology-based formalizations of the geological knowledge currently represented on geological maps [19]. However, the needs for reservoir studies are not the same as those of geological map editors. Consequently, we decided to complete the already existing geological ontologies by defining specific ontologies for describing additional geological concepts related to the particular field of oil reservoir modelling. This meets the traditional definition of Gruber, which stipulates that ontologies allow describing static knowledge attached to a field, by specifying what are the objects that compose the domain and how they are organized [1]. We thus created a common vocabulary which can then be refined to describe semantic characteristics in OWL-DL language [8]. This vocabulary was tentatively classified resulting in a set of domain ontologies adapted to our needs. We present in Fig. 9 some extracts of the geosciences ontologies, depicted as UML diagrams.

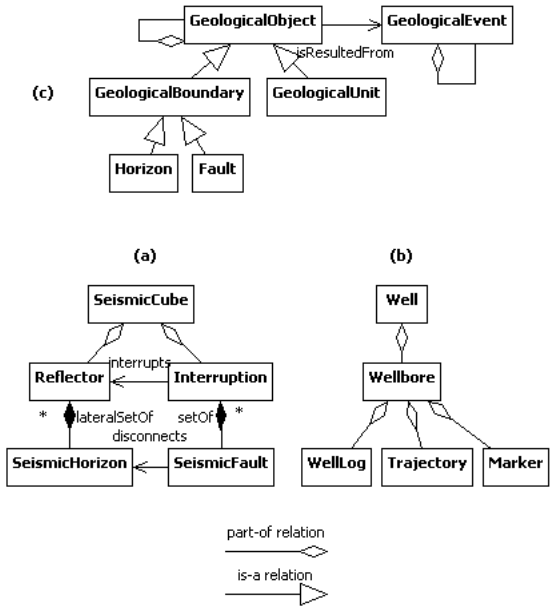


Fig. 9. Extracts of ontologies for geological modelling.

The set of domain ontologies is composed by:

- *local ontologies* for representing the concepts used by the professionals in specific fields of expertise within the oilfield modelling workflow, such as the *Seismic Interpretation ontology* (Fig. 9(a)), and the *Well ontology* (Fig. 9(b));
- an ontology for describing the concepts of *common Geology*: geological units, geological boundaries, geological processes, lithology and reservoir. This ontology is called the *Basic Geology* ontology (Fig. 9(c)).

The Basic Geology ontology (whose most recent version was presented in a W3C workshop [20]) identifies geological concepts that are applied through the whole oilfield modelling chain. Those concepts refer to objects which are identified at

the beginning of the chain, but receive a different characterization within the various geological models. The geological interpretation consists, then, in binding these multiple characterizations to the same shared ontological concept.

Although we represent the geosciences ontologies in OWL language, we choose OntoDB database to do the persistence of the ontologies and data, because of the advantages described in Section V. We use a mapping algorithm that translates OWL ontologies to OntoQL statements which use the constructs defined for building ontologies, and then store them in OntoDB. The way how the ontologies are represented in OntoDB is shown in Fig. 10: the ontology constructs in part (2), the structure of the domain ontologies in part (3) and the ontologies instances in part (4).

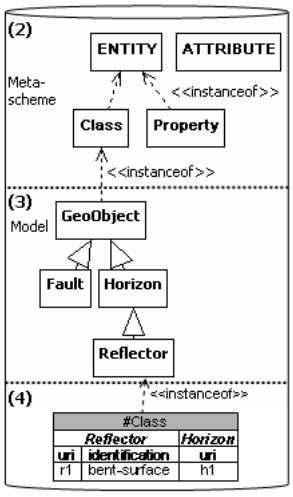


Fig. 10. Ontologies in OntoDB

Statements Q6 and Q7 exemplify the OntoQL statements that creates the ontology classes that represent the geological concept `Horizon` and the geology seismic concept `Reflector` (from the Seismic ontology) in the OntoDB database using the ontology meta-model constructs.

```
Q6: CREATE #Class Horizon

Q7: CREATE #Class Reflector PROPERTIES (URI
STR, identification STR)
```

In the example presented as the case study, the geologist identifies *Reflectors* from horizontal seismic traces. We represent this by creating an instance of the concept `Refletor`, which is identified as `bent-surface` by the geologist, as the statement Q8:

```
Q8: INSERT INTO Reflector (URI,
identification)
VALUES (r1, bent-surface)
```

B. Representing Oilfield Data

As explained in the case study section, each oilfield modelling activity corresponds to the utilization of one specific tool, which characterises data in a different structure: points, surfaces, meshes.

We observed the available metadata of oilfield models (types, attributes, header) and reduced this metadata to the minimum necessary structure that allows a uniform description of those models (file name, identificator, main composite objects, etc.). The objective in capturing and formalizing this metadata is to enable querying data using concepts of the knowledge-level and, for future works, to allow data transformation between models. For this, the oilfield data artefacts should be represented as *instances of their metadata*.

For the activity observed in the case study (the Geological seismic interpretation) the folowing metadata was abstracted: seismic traces are represented as clouds of 3D points in an ASCII file, a format known as *XYZ file*. Thus, we represent data used in this activity as instances of the *XYZ file* metadata. The Fig. 11 illustrates a simplification of the XYZ format metadata.

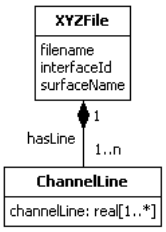


Fig. 11. The XYZ metadata

The statement Q9 exemplifies the encoding of the `XYZ format` metadata in OntoDB using using the just-added `#DataClass` constructor. It creates a `DataClass` entity named `XYZFile`, with a DataAttribute named `surfaceName`, of type `String`, and multiplicity exactly 1.

```
Q9: CREATE #DataClass XYZFile
(PROPERTIES (surfaceName STR 1 1)
```

The statement Q10 create the `ChannelLine` dataclass with attribute `channelLine`, and statement Q11 alters the dataclass `XYZFile` by adding an association `hasLine` which connects `XYZFile` to multiple `ChannelLine`.

```
Q10: CREATE #DataClass ChannelLines
(PROPERTIES (channelLine ARRAY 1 '1', )

Q11: ALTER #DataClass XYZFile
(ADD PROPERTIES (hasLine REF(ChannelLines) 1
'*')
```

Then, we represent seismic data as instance of the metadata created above. This is exemplified in the OntoQL statements Q12 to Q15 below, Here we create an instance of `XYZFile`, whose attribute filename is `reflect3D-0047.xyz`, which has two `channelLines` (each line is composed of 3 points, that represent the 3D coordinate).

```
Q12: INSERT INTO XYZFile
(URI, filename, surfaceName)
VALUES (r_47, 'reflect3D_0047.xyz',
'reflect47')
```

```
Q13: id_1 = INSERT INTO ChannelLines
(channelLine) VALUES ([0.19 1.31 0.24])

Q14: id_2 = INSERT INTO ChannelLines
(channelLine) VALUES ([0.91 2.31 3.04])

...

Q15: UPDATE XYZFile WHERE URI = r_47 SET
hasLine = [id_1, id_2]
Q16:      INSERT INTO hasLine (end1, end2)
VALUES (id_5, id_7)
```

The advantage of representing the technical data as instances in OntoDB is that we will have stored, in the same place, both data and ontologies, and this will make it possible to create the link between the two.

C. Defining typed-annotations

Finally, since we have all metadata and the local ontologies represented in OntoDB metamodel (level M1), we can now define the link between those models By means of the construct `#Annotation`. In the present case study we know that a very common annotation will be performed by geologists during the interpretation of the reflectors from a seismic image. Thus, in statement Q17 we create an annotation of the type *ReflectorAnnotation*, which annotates entities of type `XYZFile` with concepts of type `Reflector`.

```
Q17: CREATE #Annotation ReflectorAnnotation
(XYZFileURI REF(XYZFile), ReflectorURI
REF(Reflector))
```

The final step is to create instances of the typed annotations in order to annotate real data. In statement Q18 we create an instance of the typed-annotation *ReflectorAnnotation*, which makes reference to an instance of the metadata `XYZFile` and an instance of the ontology concept `Reflector`, created previously.

```
Q18: INSERT INTO ReflectorAnnotation (author,
date, ReflectorURI, XYZFileURI) VALUES
(Geologist_1, 10/10/2008, r_47, r1)
```

The advantage of the typed-annotation is that we group in a same annotation table files that could have been created by any type of seismic application. The annotation table stands for all the annotations performed during one same interpretation task.

Fig. 12 extends Fig. 10 and illustrates the implementation of the case study in the three levels of OntoDB. In the part (2) (M2 level) we have original constructs of ontologies, augmented of Annotation and Engineering Models constructs; in the part (3) (M1 level) we have the geosciences ontologies, the oilfield metadata, and the typed-annotations; finally, in the part (4) (M0 level), we depict an extract of the tables in OntoDB: in the table `#Class` we have all the geosciences concepts and their instances, the table `#DataElement` contains all the metadata for Oilfield Models and the reference to the real data. The links between the data and their ontological meaning is expressed in the table `#Annotation`. In this example, the XYZ file named `ref3D-47.xyz` is annotated by an instance $r1$ of the concept `Reflector`. The instance $r1$ is the result

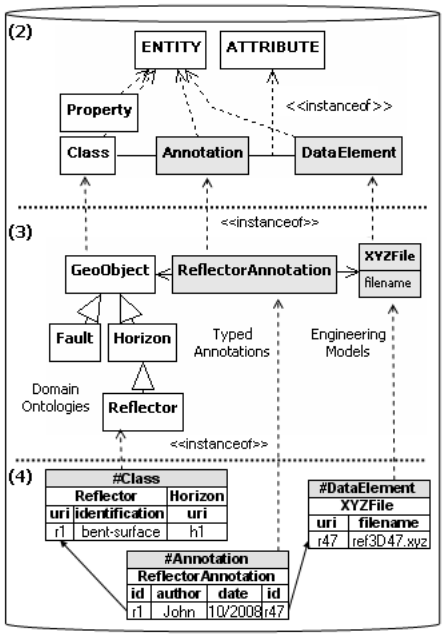


Fig. 12. Implementation of the case study using OntoDB extension.

of the interpretation of some geoscientist, who identifies this reflector as 'bent-surface'.

D. Annotation Production Process

The annotation elements described below will be produced within the different activities of the oilfield modelling workflow. According to the type of task and the computer-based tool used, we distinguish three scenarios for producing instances of annotations:

- **white box annotation**: the annotation system is integrated to the modelling tool. The annotation system knows the associated local ontology, so, when the user produces an interpretation about the oilfield model, the annotation system automatically creates instances of annotation referring to the URI of the ontology. For the Seismic interpretation task, it is being developed a tool for automatic seismic interpretation (by Verney et al. [21]) that generates: (i) the *instances of the Seismic Ontology*, which corresponds to the seismic interpretation, as well as (ii) the *annotations* that link these ontology instances to the technical data used by the tool.
- **black box annotation**: when using a proprietary oilfield modelling tool, it is not possible to integrate the annotation system into it. The annotation must then be carried out in an interactive way by the user: while interpreting the oilfield model he/she also produces annotations over the data files, like an experience report. This corresponds to the practice of generating text documents that explicate the interpretations, the difference is that documents are in natural language, and cannot be processed later on by automatic tools, while annotations produced with an ontological tool are explicit and formal.

- **intrusive annotation**: when the oilfield model is already interpreted, the annotation system examines the data files associated and discovers, by using heuristic rules, which objects must be annotated and then associate the correspondent ontology instances to these objects, producing then the instances of annotations. An practical example is the set of files used to represent Well information, which are described using an XML-based standard called WITSML (Wellsite Information Transfer Standard Markup Language) [22]. This standard defines XML tags that are specific to the well domain. A well data file created using WITSML is able to be processed by a parser, and its information transformed in ontological instances in the knowledge base. The link to the original data source is maintained by the annotation.

VIII. Exploitation of the extended OntoDB architecture

As explained in previous sections, in the oilfield modelling workflow the users must know how the data is structured in internal formats so as to formulate queries about the oilfield models. We believe that annotating those models with meaningful concepts (concepts from the ontologies of the specific domains within oilfield modelling) will allow the professionals to formulate queries using their domain vocabulary, instead of using the data structure.

We present here how to interrogate the oilfield database using the constructs included in the OntoDB's meta-schema. Some simplified queries that use the meta-model constructs are expressed as follows.

- Find the name of the `TypedAnnotation` that annotates the `Reflector` classes:

```
Q19: SELECT #name FROM #Annotation
WHERE #isAnnotatedBy.#name = 'Reflector'

R: #name = 'ReflectorAnnotation'
```

- Find the name of the `DataClass` that is annotated by `Reflector`

```
Q20: SELECT #Annotation.#annotates.#name
FROM #Annotation
WHERE #isAnnotatedBy.#name = 'Reflector'

R: #name = 'XYZFile'
```

- Find the instance of `XYZFile` that is annotated by the given instance of `Reflector`

```
Q21: SELECT filename from XYZFile
 JOIN ReflectorAnnotation
 ON XYZFile.oid =
ReflectorAnnotation.annotates.oid
WHERE
ReflectorAnnotation.isAnnotatedBy.oid =
 (select Reflector.oid from Reflector
  where Reflector.URI = 'r1')

R: filename = 'reflect3D_0047.xyz'
```

Thanks to the new parts which we propose, OntoDB will be able to store within a single data base, the engineering models data and their ontology-based annotations. So the semantic concerning the objects, which is usually just in the head of the users, can be added inside the database. This approach enables to formulate queries that use the vocabulary that is significant for the domain professionals, instead of obliging them to understand how data are organised within the database.

Moreover, having ontologies stored with the data allows to use integration techniques, such as, in the case of OntoDB, the *subsumption* relations. This mechanism leaves an important autonomy to the several databases whose local ontologies refer the same shared ontology for the definition of their data schema. We will discuss about this possibility as a future work in the conclusions.

IX. Conclusion

This paper has presented an extension of a database architecture in order to handle both data elements issued from engineering models and annotations that allow to linking the engineering models data elements to their semantic definition represented by ontology concepts (classes, properties, instances or property values). As a consequence, we have obtained a homogeneous representation of the whole data and knowledge manipulated by engineers. Moreover, the availability of an exploitation language capable of manipulating both meta-models, ontologies, instances and annotations was useful since, we are capable, with OntoQL, to address semantic queries that are free from the implementation details available in the different annotated engineering models. A case study illustrating this approach has been shown. It is issued from the application domain we are working on: petroleum engineering models and geological models.

The proposed OntoDB ontology based database extension was possible thanks to the availability of:

- an explicit representation of the ontology in the database. As a consequence, we have been able to attach the annotation to the classes and to the properties of the ontology and not to the columns of the logical model of the database where instances or data issued from the engineering data models are stored;
- the possibility to access and to manipulate the ontology model through the access and manipulation to the meta-model;
- an exploitation language allowing to manipulating both the instances, their classes and the meta-model in the case of ontologies.

These characteristics are offered by the OntoDB ontology based database and by the OntoQL exploitation language. The extension of the ontology model with the annotation and engineering models model permitted to attach various types of annotations to classes and/or properties of the ontology. As a consequence, we have been able to describe semantic queries that exploit the engineering models at the semantic level, and thus abstracting from the logical model.

We believe that the possibility to access the meta-model level well adapted to define model extensions that preserve upward compatibility with the extended model. This work has opened several new directions and perspectives. Indeed, such

extensions are possible for other different engineering domains provided that an engineering and an annotation models are set up at the meta-model level. Moreover, we expect to describe other relationships at the ontology level so as to be able to a posteriori attach ontology concepts and the associated annotations to pre-existing ontologies.

ACKNOWLEDGMENT

This work is sponsored by CAPES - Brasil (process no. 4232/05-4).

REFERENCES

[1] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *Int. Journal of Human and Computer Studies*, vol. 43, no. 5/6, pp. 907–928, 1995.

[2] F. Chum, "Use case: Ontology-driven information integration and delivery: A survey of semantic web technology in the oil and gas industry," 2007. [Online]. Available: http://www.w3.org/2001/sw/sweo/public/UseCases/Chevron/

[3] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," in *Lecture Notes in Computer Science - In The Semantic Web - ISWC 2002*, vol. 2342. Springer, 2002, pp. 54–68.

[4] C. Fankam, S. Jean, and G. Pierra, "Numeric reasoning in the semantic web," in *ESWC First International Workshop on Semantic Metadata Management and Applications (SeMMA)*, vol. 346. CEUR Workshop Proceedings, 2008, pp. 84–103.

[5] S. Harris and N. Gibbins, "3store: Efficient Bulk RDF Storage," in *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03)*, 2003, pp. 1–15.

[6] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds, "Efficient RDF Storage and Retrieval in Jena2," in *Proceedings of the 1st International Workshop on Semantic Web and Databases (SWDB'03)*, 2003, pp. 131–150.

[7] D. Brickley, R. V. Guha, and B. McBride, "RDF Vocabulary Description Language 1.0: RDF Schema," 10 February 2004 2004. [Online]. Available: http://www.w3.org/TR/rdf-schema/

[8] D. L. McGuinness and F. v. Harmelen, "OWL Web Ontology Language Overview," 10 February 2004 2004. [Online]. Available: http://www.w3.org/TR/owl-features/

[9] G. Pierra, "The PLIB Ontology-based approach to data integration," in *Proceedings of the 18th IFIP World Computer Congress (WCC2004)*. Toulouse, France: Springer, August 2004 2004.

[10] M. Park, J. Lee, C. Lee, J. Lin, O. Serres, and C. Chung, "An efficient and scalable management of ontology," in *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07) - LNCS*, vol. 4443. Springer, 2007, pp. 975–980.

[11] H. Dehainsala, G. Pierra, and L. Bellatreche, "OntoDB: An Ontology-Based Database for Data Intensive Applications," in *Proceedings of Database Systems for Advanced Applications, 12th International Conference (DASFAA07)*. Springer, 2007.

[12] D. Connolly, F. v. Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "DAML+OIL Reference Description," March 2001 2001. [Online]. Available: http://www.w3.org/TR/daml+oil-reference

[13] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a survey of the state of the art," in *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, 2006.

[14] J. Heflin and J. Hendler, "Searching the web with SHOE," in *AAAI Workshop on Artificial Intelligence for Web Search (WS-00-01)*. AAAI Press, Menlo Park, CA, 2000, pp. 35–40.

[15] N. Maillot, M. Thonnat, and A. Boucher, "Towards ontology based cognitive vision," in *Third International Conference on Computer Vision Systems, ICVS 2003*. Graz, Austria: Springer, 2003.

[16] J. Farell and H. Lausen, "Semantic Annotations for WSDL - W3C Working Draft," 2006. [Online]. Available: http://www.w3.org/2002/ws/sawsdl/

[17] R. Schroeter, J. Hunter, and D. Kosovic, "Vannotea - A Collaborative Video Indexing, Annotation and Discussion System For Broadband Networks," in *Knowledge Markup and Semantic Annotation Workshop, K-CAP 2003*, Sanibel, Florida, 2003.

[18] S. Jean, A.-A. Yamine, and G. Pierra, "Querying ontology based database using ontoql (an ontology query language)," in *Proceedings of On the Move to Meaningful Internet Systems (ODBASE)*, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and e. al., Eds., vol. 4275. Springer, 2006, pp. 704–721.

[19] S. M. Richard, J. Matti, and D. R. Soller, "Geoscience terminology development for the national geologic map database," in *Digital Mapping Techniques Workshop*, D. R. Soller, Ed., vol. U. S. Geological Survey Open-File Report 03-471, 2003, pp. 157–167. [Online]. Available: http://pubs.usgs.gov/of/2003/of03-471/richard1/index.html

[20] J. Rainaud, L. Mastella, P. Durville, Y. Ait-Ameur, M. Perrin, S. Grataloup, and O. Morel, "Two use cases involving semantic web earth science ontologies for reservoir modeling and characterization," in *W3C Workshop on Semantic Web in Oil & Gas Industry*, 2008. [Online]. Available: http://www.w3.org/2008/11/ogws-papers/rainaud.pdf

[21] P. Verney, J.-F. Rainaud, M. Perrin, and M. Thonnat, "A knowledge-based approach of seismic interpretation : horizon and dip-fault detection by means of cognitive vision." in *Proceedings of SEG Las Vegas 2008 Annual Meeting*, 2008, pp. 874–878.

[22] Energistics, "WITSML - the Wellsite Information Transfer Standard Markup Language," 2007. [Online]. Available: http://www.witsml.org