The Role of Foundational Ontologies for Domain Ontology Engineering: An Industrial Case Study in the Domain of Oil and Gas Exploration and Production

Giancarlo Guizzardi[1], Fernanda Baião[*,2], Mauro Lopes[2], Ricardo Falbo[1]

[1]Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo, Espírito Santo, Brazil
[2]NP2Tec – Research and Practice Group in Information Technology, Federal University of the State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil

ABSTRACT

Ontologies are commonly used in computer science either as a reference model to support semantic interoperability, or as an artifact that should be efficiently represented to support tractable automated reasoning. This duality poses a tradeoff between expressivity and computational tractability that should be addressed in different phases of an ontology engineering process. The inadequate choice of a modeling language, disregarding the goal of each ontology engineering phase, can lead to serious problems in the deployment of the resulting model. This article discusses these issues by making use of an industrial case study in the domain of Oil and Gas. We make the differences between two different representations in this domain explicit, and highlight a number of concepts and ideas that were implicit in an original OWL-DL model and that became explicit by applying the methodological directives underlying an ontologically well-founded modeling language.

Keywords: Ontology; Ontology Languages; Conceptual modeling; Oil and Gas domain.

Introduction

Since the word ontology was mentioned in a computer related discipline for the first time (Mealy, 1967), ontologies have been applied in a multitude of areas in computer science. The first noticeable growth of interest in the subject in mid 1990's was motivated by the need to create principled representations of domain knowledge in the knowledge sharing and reuse community in Artificial Intelligence (AI). Nonetheless, an explosion of works related to the subject only happened in the past decade, highly motivated by the growing interest on the Semantic Web, and by the key role played by ontologies in that initiative.

There are two common trends in the traditional use of the term ontology in computer science: (i) firstly, ontologies are typically regarded as an explicit representation of a shared conceptualization, i.e., a concrete artifact representing a model of consensus within a community and a universe of discourse. Moreover, in this sense of a reference model, an ontology is primarily aimed at supporting semantic interoperability in its various forms (e.g., model integration, service interoperability, knowledge harmonization, and taxonomy alignment); (ii) secondly, the discussion regarding representation mechanisms for the

construction of domain ontologies is, typically, centered on computational issues, not truly ontological ones.

An important aspect to be highlighted is the incongruence between these two trends. In order for an ontology to be able to adequately serve as a reference model, it should be constructed using an approach that explicitly takes foundational concepts into account; this is, however, typically neglected for the sake of computational complexity.

The use of foundational concepts that take truly ontological issues seriously is becoming more and more accepted in the ontological engineering literature, i.e., in order to represent a complex domain, one should rely on engineering tools such as design patterns, computational environments, modeling languages and methodologies that are based on well-founded ontological theories in the philosophical sense (e.g., (Burek, 2006; Fielding, 2004)). Especially in a domain with complex concepts, relations and constraints, and with potentially serious risks which could be caused by interoperability problems, a supporting ontology engineering approach should be able to: (a) allow the conceptual modelers and domain experts to be explicit regarding their ontological commitments, which in turn enables them to expose subtle distinctions between models to be integrated and to minimize the chances of running into a *False Agreement Problem* (Guarino, 1998); (b) support the user in justifying their modeling choices and providing a sound design rationale for choosing how the elements in the universe of discourse should be modeled in terms of language elements.

This marks a contrast to practically all languages used in the tradition of knowledge representation and conceptual information modeling, in general, and in the semantic web, in particular (e.g., RDF, OWL, F-Logic, UML, EER). Although these languages provide the modeler with mechanisms for building conceptual structures (e.g., taxonomies or partonomies), they offer no support neither for helping the modeler on choosing a particular structure to model elements of the subject domain nor for justifying the choice of a particular structure over another. Finally, once a particular structure is represented, the ontological commitments which are made remain, in the best case, tacit in the modelers' mind; in the worst case, even the modelers and domain experts remain oblivious to these commitments.

An example of an ontologically well-founded modeling language is the version of UML 2.0 proposed in (Guizzardi, 2005) and, thereafter, dubbed *OntoUML*. This language has its real-world semantics defined in terms of a number of ontological theories, such as theory of parts, of wholes, types and instantiation, identity, dependencies, unity, etc. However, in order to be as explicit as possible regarding all the underlying subtleties of these theories (e.g., modal issues, different modes of predication, higher-order predication), this language strives for having its formal semantics defined in a logical system as expressively as possible. Now, as well understood in the field of knowledge representation, there is a clear tradeoff between logical expressivity and computational efficiency (Levesque & Brachman, 1987). In particular, any language which attempts at maximizing the explicit characterization of the aforementioned ontological issues risks sacrificing reasoning efficiency and computational tractability. In contrast, common knowledge representation and deductive database languages (e.g., some instances of Description Logics) have been specifically designed to afford efficient automated reasoning and decidability.

In summary, ontology engineering must face the following situation: on one side, we need ontologically well-founded languages supported by expressive logical theories, in order to produce sound and clear representations of complex domains; on the other side, we need lightweight ontology languages supported by efficient computational algorithms. In order to reconcile these two sets of contradicting requirements, Guizzardi & Halpin (2008) advocated that two classes of languages are required to fulfill these two sets of requirements. Moreover, as any other engineering process, an ontology engineering process lifecycle should comprise phases of conceptual modeling, design, and implementation. In the first phase, a reference

ontology is produced, aiming at representing the subject domain with truthfulness, clarity and expressivity, regardless of computational requirements. The main goal of these reference models is to help modelers to externalize their tacit knowledge about the domain, to make their ontological commitments explicit in order to support meaning negotiation, and to afford the tasks of domain communication, learning and problem solving as best as possible. The same reference ontology may further give rise to different lightweight ontologies in different languages (e.g., F-Logic, OWL-DL, RDF, Alloy, and KIF), thus satisfying different sets of non-functional requirements. Defining the most suitable language for codifying a reference ontology is then a choice to be made at the design phase, by taking both the end-application purpose and the tradeoff between expressivity and computational tractability into account.

In this article, we illustrate the issues at stake in the aforementioned tradeoff by discussing an industrial case study in the domain of Oil and Gas Exploration and Production. However, since we were dealing with a pre-existing OWL-DL codified ontology, we had to reverse the direction of model development. Instead of producing a reference model in OntoUML which would then give rise to an OWL-DL codification, we had to start with the OWL-DL domain ontology and apply a systematic reverse engineering process to it in an attempt to reconstruct the proper underlying reference model in OntoUML. By doing that, we manage to show how much of important domain knowledge had either been lost in the OWL-DL codification or remained tacit in the domain experts' minds.

The remainder of this article is organized as follows. Section 2 briefly elaborates on the research assumptions and methods used in this article. Section 3 characterizes the domain and industrial setting in which the case study took place, namely, the domain of oil and gas exploration and production and in the context of a large Petroleum Organization. Section 4 discusses the reengineering of the original lightweight ontology produced in the settings described in section 3. This reengineering step was conducted by transforming the original ontology to well-founded version represented in OntoUML. Section 5 presents the conclusions of the article.

Research Assumptions and Methods

In a previous work, Guizzardi (2005) proposed a formal framework for the evaluation and (re)design of modeling languages. The proposed framework extended the works of (Weber, 1997) and (Gurr, 1999) to offer theoretical means to systematically analyze the suitability of a language to model phenomena in a given domain in reality. This analysis is conducted by evaluating the level of homomorphism between a language (captured in a specification of the modeling primitives available in the language , i.e., a metamodel) and the domain (captured in a domain reference ontology). As (Gurr, 1999) argues at length, the stronger the match between a model and the represented domain, the easier is to reason with the former. The easiest case is when these matches are isomorphisms. The implication of this for the human agent who interprets the model is that his interpretation correlates precisely and uniquely with a portion of reality being represented. By contrast, where the correlation is not an isomorphism then there may potentially be a number of different phenomena which would match the interpretation.

The framework proposed in (Guizzardi, 2005) enumerates a number of properties that need to be reinforced in order to guarantee an isomorphism of this mapping. Two of these properties are *Completeness* and *Lucidity*. In a nutshell, a modeling language is complete if it offers modeling primitives to capture all the necessary domain distinctions. In contrast, a language is Lucid if, for each modeling primitive, there is no more than one interpretation in terms of domain distinctions. Incompleteness hinders the expressivity of the language: there are aspects of the subject domain which cannot be able to directly capture using the language. The lack of lucidity, in contrast, introduces ambiguity and hinders the clarity of the language:

people have to bring extra-model knowledge in order to attempt to disambiguate competing possible interpretations of the modeling constructs. Moreover, there is a practical connection between these properties. If a language is incomplete, the following situations may arise: (i) information will be left out of the descriptions represented by the language (thus remaining tacit and, hence, inaccessible in a stakeholder´s mind); (ii) the modeler will semantic overload modeling constructs of the language in order to represent the necessary domain distinctions (thus rendering a non-lucid representation system).

The case study presented in this work ultimately compares general modeling languages w.r.t. their suitability for supporting the conceptual modeling phase of Ontology Engineering. Since conceptual modeling is about representing conceptualizations of reality, this work argues that the underlying conceptualization a general conceptual modeling language should commit to is that of a Foundational Ontology (Guizzardi, 2005; Weber, 1997). OntoUML has been explicitly designed to have as modeling primitives those representing ontological distinctions prescribed by a cognitive-based Foundational Ontology. Thus, the hypothesis which has been tested by this case study is the following. By using a modeling language which is known to be incomplete w.r.t. these ontological distinctions (OWL-DL) to represent a domain ontology in a large and intrinsically complex domain, one is prone to produce conceptual problems of the types (i) and (ii) above. In contrast, by using a language which is complete (and, hence, more expressive), not only these problems can be made explicit, but they also can be addressed in satisfactory manner.

Following (Mylopoulos, 1992), conceptual models are primarily meant to be used by humans in tasks of domain understanding, problem solving and communication. For this reason, as discussed in depth in (Guizzardi, 2007), a Reference Ontology for conceptual modeling languages must be one that takes human cognition and language seriously. These requirements motivate the use of an ontological theory such as the one underlying OntoUML (which is also based on a number of results from philosophy of language, cognitive psychology and linguistics) as opposed to a revisionary ontology such as the BWW model (Weber, 1997). For a detailed comparison of these two foundational ontologies with respect to their abilities to provide foundations for conceptual modeling languages, one should refer to (Guizzardi, 2005).

Characterizing the Case Study Domain and Settings

The Oil and Gas sector is marked by projects which are technically complex, significantly expansive, involve several large corporations, and, frequently, cross-cut several subareas of the involved organizations. The success of these projects is strongly affected by how well the work of the involved participants (and their supporting systems) are integrated and/or coordinated. The importance of integration, effective cooperation and coordination in this sector is becoming more and more evident due to the constant increase in the number of participants in these projects, in particular the number of small companies to which highly specialized services are outsourced. Traditionally, each of the involved organizations are equipped with their own pool of applications and databases, which are more and more encapsulated into services (Papazoglou et al., 2008). This scenario poses a challenge to application and enterprise integration. Current initiatives towards cooperation between organizations are founded on *ad hoc* infrastructures built on mappings between specific applications and databases. Moreover, there is hardly any collaboration across disciplines and process phases, since each of these rely on alternative conceptual structures and terminologies. However, due to the constant increase both in data complexity and typical numbers of required collaborators, a case-by-case mapping between databases, applications

and terminological structures cannot be accounted as a viable solution for this industry (Gulla et al., 2006).

With respect to decision making, as pointed out by (Gulla et al., 2006), the supporting processes fall short in terms of relevant and precise information. Information which is dispersed in multiple non-integrated databases, classified according to different terminologies, and represented in incompatible formats, often becomes available for decision making only too late or not at all. Finally, even when data becomes available, it is often difficult to access its real world semantics and to understand its limitations and underlying assumptions (Gulla et al., 2006).

A study conducted by the *Norwegian Oil Industry Association* and reported by (Gulla et al., 2006) makes the need for better collaboration and integration between process phases, disciplines and companies in this sector explicit. According to this study, this is not supported by existing standards and, as a result, projects are typically costly, and decision making is often made in terms of wrong or outdated information. The strong requirements of semantic interoperability make the Oil and Gas domain a potentially rich domain for applying ontologies. Ontology-based approaches for semantic interoperability and information exchange involves the use of ontologies of rich and extensive domains combined with industry patterns and controlled vocabularies, reflecting relevant concepts within this domain. Following (Chum, 2007), and in pace with the aforementioned discussion, we can summarize the motivating factors for the use of ontologies in the oil and gas industry as follows:

- The great data quantity generated each day, coming from diverse sources, involving different disciplines. Integrating different disciplines to take advantage of the real value of information has been a complex and costly task.
- The existence of data in different formats, including structured databases and semi-structured documents. To deal with the great quantity of information, as well as heterogeneous formats, a new approach is needed to handle information search and access.
- The necessity of standardization and integration of information along the frontiers of systems, disciplines and organizations, to support the decision-making with the collaborators, to the extent that better quality data will be accessible on the opportune time.

Examples of initiatives of using ontologies in the energy sector include IIP (Integrated Information Platform) (Sandsmark & Mehta, 2004), AKSIO (Active Knowledge Support in Integrated Operations) (Norheim & Fjellheim, 2006), ADI (Accelerating Development of ISO 15926) (Paap, 2006), and DPR (Daily Production Reports) (TietoEnator, 2006). An additional noteworthy effort is the O3R (Open Oilfield Ontology Repository), which aims at collecting public OWL ontologies in the domain of Oil and Gas in order to make them available to the industry as a whole.

Most of the existing ontology-based initiatives in this area rely on semantic web representation languages, such as OWL and RDF. As demonstrated in (Guizzardi, 2006), the use of non-expressive and ontologically neutral languages such as OWL and RDF cannot guarantee semantic interoperability when concurrently developed models are integrated. In fact, (Guizzardi, 2006) shows that even when relatively simple semantic web ontologies (e.g., MusicBrainz, FOAF, Mogatu BDI) are integrated, the previously discussed *False Agreement Problem* can be manifested.

The case study reported in this paper was conducted in a large Petroleum Corporation, by analyzing and redesigning a pre-existing ontology in the domain of Oil and Gas Exploration and Production, henceforth named *E&P-Reservoir Ontology*. Due to the extensiveness and complexity of this domain, only few sub domains were taken into consideration on the initial

version of this ontology, namely, the *"Reserve Assessment"* sub domain, and the *"Mechanical pump"* sub domain. The knowledge acquisition process used to create the original E&P-Reservoir Ontology was conducted by representing business process models according to (Baiao et al., 2008). The original E&P-Reservoir ontology, following the same trend of other existing projects in the area (e.g., IIP and AKSIO), was codified in OWL-DL comprising 178 classes, which together contained 55 data type properties (OWL datatypeProperties) and 96 object properties (OWL objectProperties).

In a nutshell, a *Reservoir* is composed of *Production Zones* and organized in *Fields* – geographical regions managed by a Business Unit and containing a number of *Wells*. *Reservoirs* are filled with *Reservoir Rock* – a substance composed of quantities of Oil, Gas and Water. *Production* of Oil and Gas from a Reservoir can occur via different lifting methods (e.g., natural lifting, casing's diameter, sand production, among others) involving different Wells. One of these artificial lifting methods is the *Mechanical Pump*. The simultaneous production of oil, gas and water occurs in conjunction with impurities. To remove these impurities, facilities are adopted on the fields (both off-shore and on-shore), including the transfer of hydrocarbons (Oil and Gas) via *Ducts* to refineries for proper processing. The notion of *Reserve Assessment* refers to the process of estimating, for each Exploration Project and Reservoir, the profitably recoverable quantity of hydrocarbons for that given reservoir. The *Mechanical Pump* subdomain ontology, in contrast, defines a number of concepts regarding the methods of fluid lifting, transportation, and other activities that take place in a reservoir during the Production process. For a more extensive definition of the concepts in this domain, one may refer to *The Energy Standard Resource Center* (www.energistics.org).

Reverse engineering an OntoUML version of the *E&P-Reservoir* Ontology

In this section, we discuss some of the results of producing an OntoUML version of the original E&P-Reservoir Ontology in this domain. In particular we focus at illustrating a number of important concepts in this domain which were absent in the original OWL model and remained tacit in the domain experts' minds, but which became explicit by the application of methodological directives underlying OntoUML. It is important to emphasize that this section does not aim at serving as an introduction to OntoUML neither as a complete report on the newly produced version of the E&P-Reservoir Ontology.

Making the Real-World Semantics of Relationships Explicit

Figure 1 depicts a fragment of the OWL ontology and figure 2 depicts the correspondent fragment transformed to OntoUML.

The OntoUML language, with its underlying methodological directives, makes an explicit distinction between the so-called *material* and *formal relationships*. A formal relationship can be reduced to relationships between intrinsic properties of its relata. For example, a relationship *more-dense-than* between two fluids can be reduced to the relationship between the individual densities of the involved fluids (*more-dense-than(x,y) iff the density of x is higher than of y's*). In contrast, material relationships cannot be reduced to relationships between individual properties of involved relata in this way. In order to have a material relationship established between two concepts C1 and C2, another entity must exist that makes this relationship true. For example, we can say that the Person John works for Company A (and not for company B) if an employment contract exists between John and Company A which makes this relationship true. This entity, which is the truthmaker of material relationships, is termed *relator* in OntoUML and the language determines that (for

the case of material relationships) these *relators* must be explicitly represented on the models (Guizzardi & Wagner, 2008).

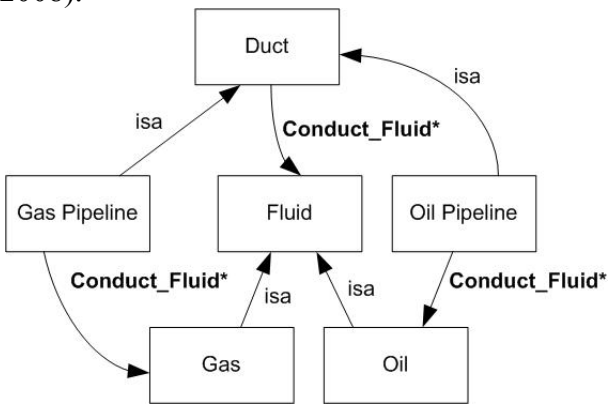

Fig. 1. Representation of Fluid transportation *(OWL)*.

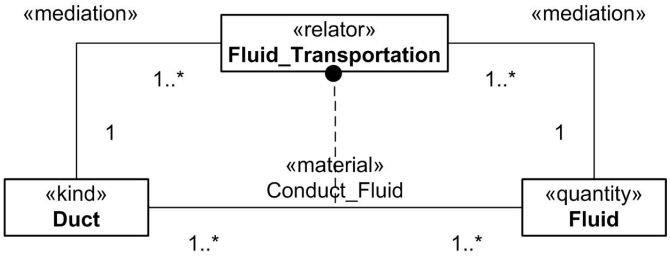

Fig. 2. Alternative Representation of Fluid transportation (OntoUML), an interpretation of Fluid transportation with unique Duct and Fluid.

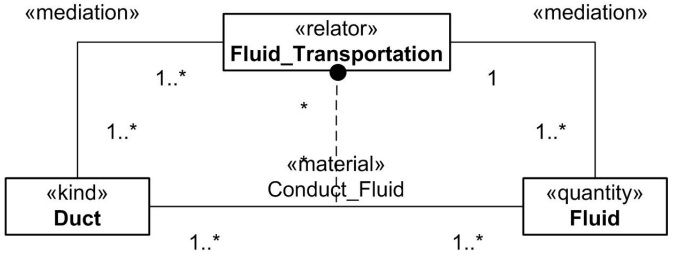

Fig. 3. Interpreting Fluid transportation with multiples Ducts and Fluids.

The *Conduct_Fluid* relationship of figure 1 is an example of a material relationship. Therefore, this relationship only takes place (i.e., the *Conduct_Fluid* relationship is only established) between a specific duct x and a specific portion of fluid y, when there is at least a fluid transportation event that involves the participation of x and y.

Besides making explicit the truthmakers of these relations, one of the major advantages of the explicit representation of *relators* is to solve an inherent ambiguity of cardinality constraints that exists in material relationships. Take for example the cardinality constraints of one-to-many represented for the relationship *Conduct_Fluid* in figure 1. There are several possible interpretations for this model which are compatible with these cardinality constraints but which are mutually incompatible among themselves. Two of these interpretations are depicted in figures 2 and 3.

On the model of figure 2, given a fluid transportation event, we have only one duct and only one portion of fluid involved; both fluid and duct can participate in several transportation events. In contrast, on the model of figure 3, given a fluid transportation event, we have possibly several ducts and portions of fluid involved; a duct can be used in several transportation events, but only one fluid can take part on a fluid transportation.

When comparing these two models in OntoUML we can see that the original OWL model collapses these two interpretations (among others) in the same representation, which have substantially different real-world semantics. This semantic overload can be a source of many interoperability problems between applications. In particular, applications that use different models and that attach distinct semantics to relationships such as discussed above can wrongly assume that they agree on the same semantics (an example of the previously mentioned *False Agreement Problem*).

Finally, in the OntoUML models in this section, the dotted line with a filled circle on one of its endings represents the *derivation* relationship between a *relator type* and the *material relationship* derived from it (Guizzardi, 2005). For example, the derivation relationship *Fluid Transportation* (*relator type*) and *Conduct_Fluid* (*material relationship*) represents that for all x, y we have that: <x,y> is an instance of *Conduct_Fluid* iff there is an instance z of *Fluid Transportation* that mediates x and y. Mediation is a specific type of existential dependence relation (e.g., a particular Fluid Transportation can only exist if that particular Duct and that particular Fluid exist). Moreover, it also demonstrated that the cardinality constraints of a material relationship R derived from a relator type U_R can be automatically derived from the corresponding *mediation* relationships between U_R and the types related by R. In summary, a relator is an entity which is existentially dependent on a number of other individuals, and via these dependency relationships it connects (mediates) these individuals. Given that a number of individuals are mediated by a relator, a material relationship can be defined between them. As this definition makes clear, relators are ontologically prior to material relationships which are mere logical/linguistic constructions derived from them. To put it in a different way, knowing that x and y are related via R tells you very little unless you know what are the conditions (state of affairs) that makes this relationship between this particular tuple true (Guizzardi, 2005; Guizzardi & Wagner, 2008).

Ontological Distinctions among Object Types

Differently from most existing conceptual modeling and ontology representation languages, OntoUML explicitly represents ontological distinctions among the category of Object Types. Common ontological criteria motivating these distinctions are the modal (e.g., temporal) aspects of the instantiation/classification relations. For instance, in the revised model of Figure 2, by representing Duct using the stereotype ⟨⟨kind⟩⟩, one is explicitly stating that this type: (i) is rigid, i.e., all instances of Duct are necessarily (in the modal sense) instances of this type; (ii) provides a principle of identity for its instances; (iii) represents essential properties for all its instances, i.e., it represents those properties that its instances must have in all possible circumstances. In an analogous form, by representing a type (e.g., Gas Pipeline) as a ⟨⟨subKind⟩⟩, we are stressing that this is a rigid type and that it has additional essential properties that must be carried by its instances, besides those inherited by the kind it specializes. However, differently from this kind that supplies a principle of identity for its instances, instances of a subkind inherit this principle from the former. The identification of rigid types such as kind and subKind (but also quantity, see section 2.3) is of fundamental importance for conceptual modeling. The taxonomic structures involving only these categories of types define the backbone of a conceptual model. By the very definition of rigidity w.r.t. modality (temporality), this taxonomy defines a stable structure, over which instances will be classified in all circumstances (or during all its lifecycle).

Figure 4 depicts a complete version of the model fragment of figure 1 illustrating the explicit representation of kinds and subkinds.

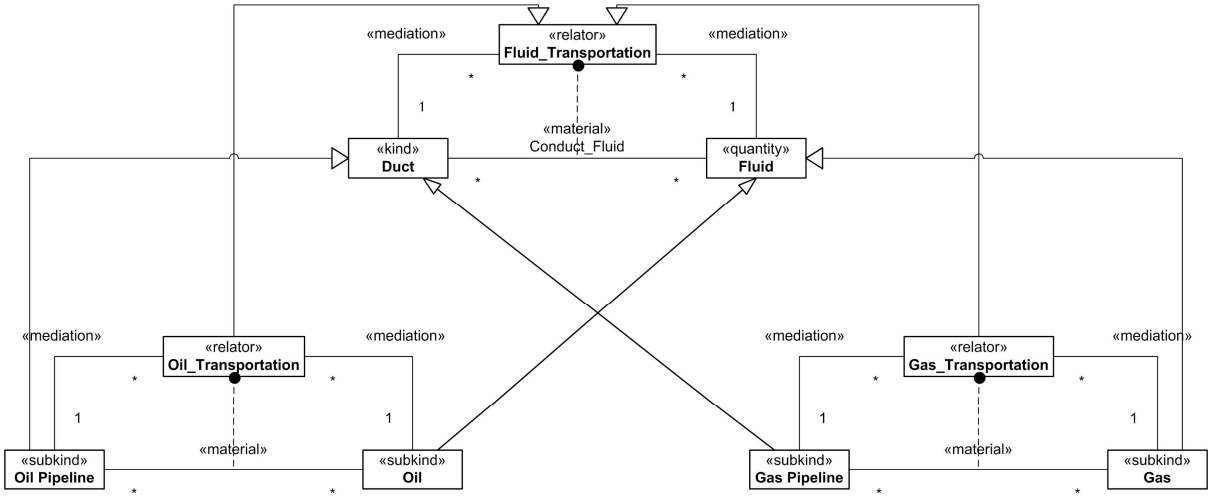

Fig. 4. Specializing Fluid and Duct Kinds and Transportation Relators.

The Ontological Status of Quantities

Figures 5 and 6 represent fragments of the domain ontology that deal with the notion of Fluid.

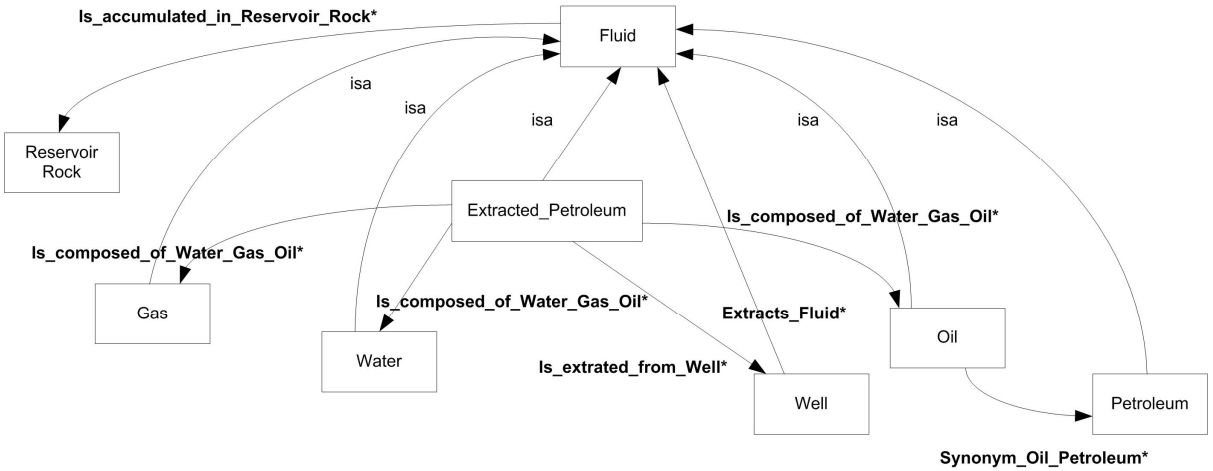

Fig. 5. The representation of Fluid *and related notions in OWL.*

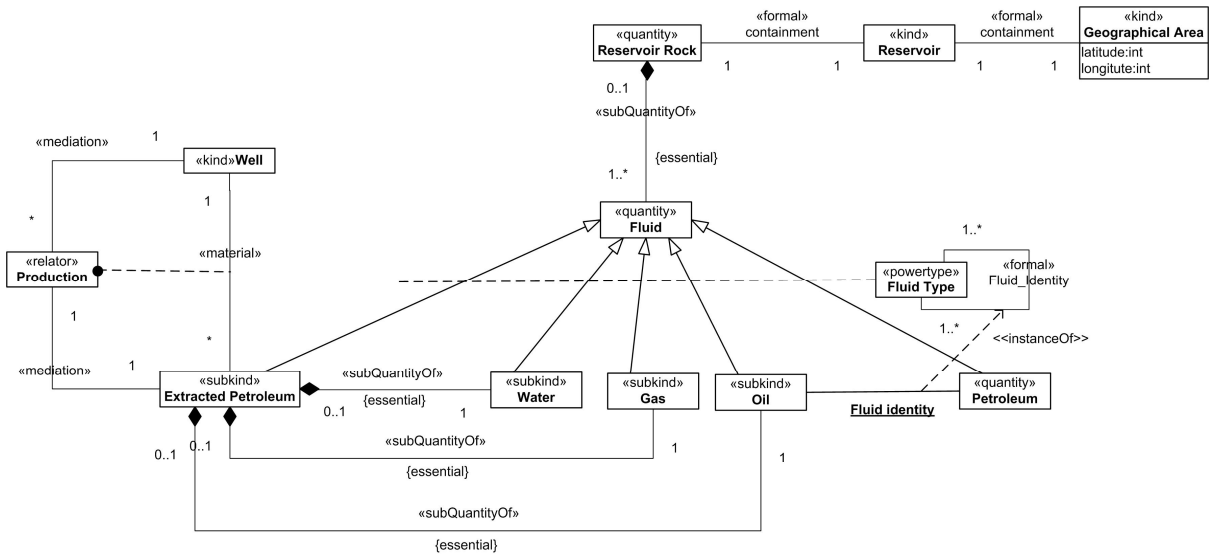

Fig. 6. The Representation of Fluid *and related notions in OntoUML*

In general, quantities or amounts of matter (e.g., water, milk, sugar, sand, oil) are entities that are homeomerous, i.e., all of their parts are the same type as the whole. Alternatively, we can say that they are infinitely divisible in subparts of the same type. Homeomerousity and Infinite divisibility causes problems both to determine the referent of expressions referring to quantities and, as a consequence, also problems to specify finite cardinality constraints of relationships involving quantity types (Guizzardi, 2005). In OntoUML, these problems are avoided by defining a modeling primitive ⟨⟨quantity⟩⟩ whose semantics are defined by invoking the ontological notion of *Quantity*. In OntoUML, a type stereotyped as ⟨⟨quantity⟩⟩ represents a type whose instances represent portions of amounts of matter which are maximal under the relation of topological self-connectness (Guizzardi, 2005).

In figure 5, the type Fluid is represented as a *quantity* in this ontological sense. A type stereotyped as ⟨⟨quantity⟩⟩ is equivalent to a kind w.r.t. the rigidity property. In ontological terms, both kinds and quantities represent the notion of *substance sortal* (Guizzardi, 2005; Guizzardi et al., 2004). Likewise, specializations of a *quantity* are also represented with the stereotype *subkind*. In figure 5, these include the specific types of Fluid: Water, Oil, Petroleum and Gas.

On the original ontology in OWL, the equivalence between the Oil and Petroleum concepts was represented by the *Oil_Petroleum_synonym* relationship defined between these concepts. This relationship is declared as being symmetric. On the original ontology, these concepts simply represent the general concepts of Oil or Petroleum and do not represent genuine types that can be instantiated. Consequently, the *Oil_Petroleum_synonym* relationship also represents a relational type that cannot be instantiated. Therefore, it does not make sense to characterize it as a symmetric relationship, since it functions as an instance and not genuinely as a type.

In the semantics adopted on the revised model, Oil and Petroleum are *quantity* types, the instances of which are specific portions of these Fluids. Therefore, in this case, there is no sense in defining an *Is_synonym_of* relationship between Oil and Petroleum. After all, defined this way, since these are genuine types that can be instantiated, this relationship would have as instances ordered pairs formed by specific portions of Oil and Petroleum, which definitely does not correspond to the intended semantics of this relationship. In fact, the relationship *Is_synonym_of* is a relationship between the Oil and Petroleum types and not between its instances. In particular, this relationship has a stronger semantics than simply symmetry, being an equivalence relationship (reflexive, symmetric, transitive).

The proper representation of an *Is_synonym_of* relationship between any two types of fluid is illustrated in figure 5. Firstly, the model makes an explicit distinction between the fluid type (instances of which are individual portions of fluid) and a type (instances of which are the concepts of Oil, Water, Gas and Petroleum themselves). Since OntoUML is an extension of standard UML, this can be represented as a *powertype*[†]. In a nutshell, a *powertype* is a type, instances of which are other types. On this specific model, the relationship between the *Fluid Type powertype* and *Fluid* defines that the subtypes of the latter (Oil, Water, Gas and Petroleum) are instances of the former. Once this distinction is made, the formal relationship of *Fluid_identity*[‡] can be defined among the instances of *Fluid Type*. This relationship can, then, be defined as an equivalence relationship which semantics is characterized by the following rule: two fluid types are identical iff they possess necessarily (i.e., at any given circumstance) the same instances. In the OntoUML language, this rule is defined outside the visual syntax of the language and as part of the axiomatization of the resulting model (ontology).

Finally, as a result of this modeling choice, particular instances of the *Fluid_identity* relationship can be defined. For example, in figure 5, the *link* (instance of a relationship) between Oil and Petroleum (instances of *Fluid Type*) is defined explicitly as an instance of *Fluid_Identity*.

In the revised model of figure 5, in the same manner as Fluid and its subtypes, *Reservoir Rock* is explicitly represented as a *quantity* type. Once more, this type represents a genuine type instances of which are particular portions of *Reservoir Rock*. The *Is_accumulated_in_Reservoir_Rock* relationship in the original model of figure 4 is, hence, replaced by a special type of part-whole relationship (*subQuantityOf*) between Reservoir Rock and Fluid. The SubQuantityOf relationship defined as a primitive in OntoUML contains a formal characterization that implies: (i) a partial order (irreflexivity, asymmetry, transitivity) relation; (ii) An existential dependency relation, i.e., in this particular example a particular portion of Reservoir Rock is defined by the aggregation of the specific particular portions of its constituent Fluids; and (iii) Non-sharing of parts, i.e., each particular portion of fluid is part of at most one portion of Reservoir Rock. It is important to emphasize that the explicit representation of the semantics of this relationship eliminates an implicit ambiguity on the original model.

Finally, representing Fluid as a *quantity* type, i.e., as a genuine type (which instances are individual portions of Fluid) and a rigid one (which instances are necessarily instances of that type) also provides information that helps to resolve cardinality constraints involving elements of the model. For example, in figure 1, a Duct is said to conduct one-to-many Fluids and a Fluid is said to be conducted by one-to-many Ducts. However, since Fluid is rigid type, a Fluid instance is classified under this type in all circumstances. It does not, however, have to participate in relations of Fluid Transportations in all these circumstances. *Mutatis Mutandis*, the same can be stated for Ducts. As consequence, the cardinality constraints in the relation of *Conduct_fluid* (and all its specializations) have been revised in figure 4 to express: a Duct conducts zero-to-many Fluids; a Fluid can be conducted by zero-to-many Ducts.

The *Containment* relation to represent the spatial inclusion among physical entities: Reservoir, Reservoir Rock and Geographic Area

The model on figure 5 also depicts the Reservoir and Geographic Area concepts and defines the formal relationship of *containment* (Smith et al., 2005) between Reservoir and Reservoir Rock and between Reservoir and Geographic Area. This relationship contains the semantic of spatial inclusion between two physical entities (with the spatial extension) that is also defined on the ontology's axiomatization, e.g., outside the visual syntax of the model.

On the original model of figure 4, there is only one relationship *Is_composed_of_Water_Gas_Oil* defined between the Extracted Petroleum and the Water, Gas and Oil concepts. On the revised ontology, this relationship is replaced by composition relationships (*subQuantityOf*). As previously discussed, the richer semantics of this relationship type makes important meta-properties of the relationship among these elements explicit in the model. As discussed in (Guizzardi, 2005; Artale & Keet, 2008; Keet & Artale, 2008), the formal characteristics of this relationship, modeled as a partially order, existential dependency relation with non-sharing of parts, have important consequences both to the design and implementation of an information system as to the automated processes of reasoning and model evaluation.

Making the *Production* Relator Explicit

As already discussed, OntoUML makes an explicit distinction between formal and material relationships. The *Extracts_Fluid* relationship between *Fluid* and *Well* in the original model is an example of the latter. In this way, following the methodological directives of the

language, the modeling process seeks to make explicit which is the appropriate relator that would substantiate that relationship. The conclusion would one come to is that the relationship *Extracts_Fluid(x,y)* is true iff there is a *Production* event involving the Well x from where the Fluid y is produced. The semantic investigation of this relationship makes explicit that the resulting fluid of this event in fact only exists after the occurrence of this event. In other words, the portion of the Extracted Petroleum only exists after it is produced from the event of production involving a well. Therefore, a mixture of water, gas and oil is considered *Extracted Petroleum* only when it is produced by an event of this kind. The *Extract_Fluid* relationship between Well and Fluid and the *Is_extracted_from_Well* relationship between Extracted Petroleum and Well on the original ontology are replaced by the material relationship *Extracts_Extracted_Petroleum* between Well and Extracted Petroleum and by the *subQuantityOf* relationships between the Extracted Petroleum portion and its sub portions of Water, Gas and Oil. This representation has the additional benefit of making clear that an event of Production has the goal of generating an Extracted Petroleum portion that is composed of particular portions of these Fluid types and not by directly extracting portions of these other types of fluid. Finally, as previously discussed, the explicit representation of the *Production* relator makes the representation of the cardinality constraints involving instances of Well and Extracted Petroleum precise, eliminating the ambiguity on the representation of the *Extract_Fluid* relationship on the original model.

Historical Dependence and Derivation Axioms

Figure 7 depicts a fragment of the original ontology representing, in very general terms, the connections between the notions of Well, Fluid and Reservoir. This model exemplifies how a plethora of important domain concepts can remain tacit in surface-level models.

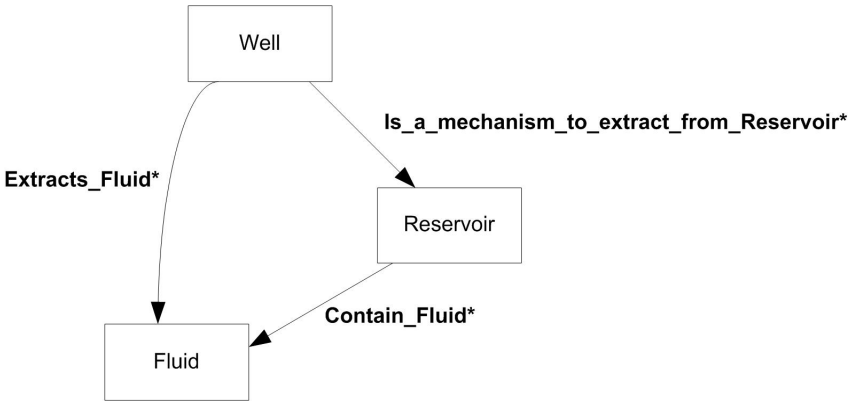

Fig. 7. Representing the notions of Well*,* Fluid*,* Reservoir *and their connections (OWL).*

One important aspect of a language such as OntoUML is, besides from the possibility to represent subtle notions explicitly, the fact that the language systematically supports the modeler to make these subtle notions explicit. In this model, we start by enquiring about the nature of *Extract_Fluid* relation. Since this is a material relation, the next question is to uncover its founding relator. As discussed previously, for an *Extract_Fluid* relation to hold between a Well x and Fluid y, in particular, there must be a *Production* event in which both x and y participate. Moreover, we have seen that the goal of this Production event is to produce a portion (quantity) of Extracted Petroleum. In other words, Production as a relator mediates Well and Extracted Petroleum (as opposed to Well and Fluids in general). This is illustrated both in figures 6 and 8.

As previously discussed, the *subquantityOf* relation defined in OntoUML to hold between portions of quantities is a type of existential dependency relation from the whole to the part.

In other words, all parts of a quantity are essential parts of it. For instance, in figure 8, we have the type Reservoir Rock stereotyped as ⟨⟨quantity⟩⟩. As a consequence, once we have the case that specific portions of water, gas and oil are extracted from a specific portion of Reservoir Rock *x* (creating a portion of Extracted Petroleum y) that specific portion *x* ceases to exists. Indeed, the resulting portion of Extracted Petroleum *y* and the Reservoir Rock *x* from which *y* originates cannot co-exist at the same circumstances. In fact, the same event that creates the former is the one that destroys the latter. However, it is important to represent the specific connection between *x* and *y*, for instance, because some characteristics from an Extracted Petroleum could result from characteristics of that Reservoir Rock. Here, this relation between *x* and *y* is modeled by the formal relation of *historical dependence* (Thomasson, 1999): in this case, since *y* is historically dependent on *x* it means that *y* could not exist without *x* having existed. As containment, historical dependence is a primitive domain independent formal relation. Once more, the formal primitive of containment is employed here to represent the relation between Reservoir Rock and Reservoir.

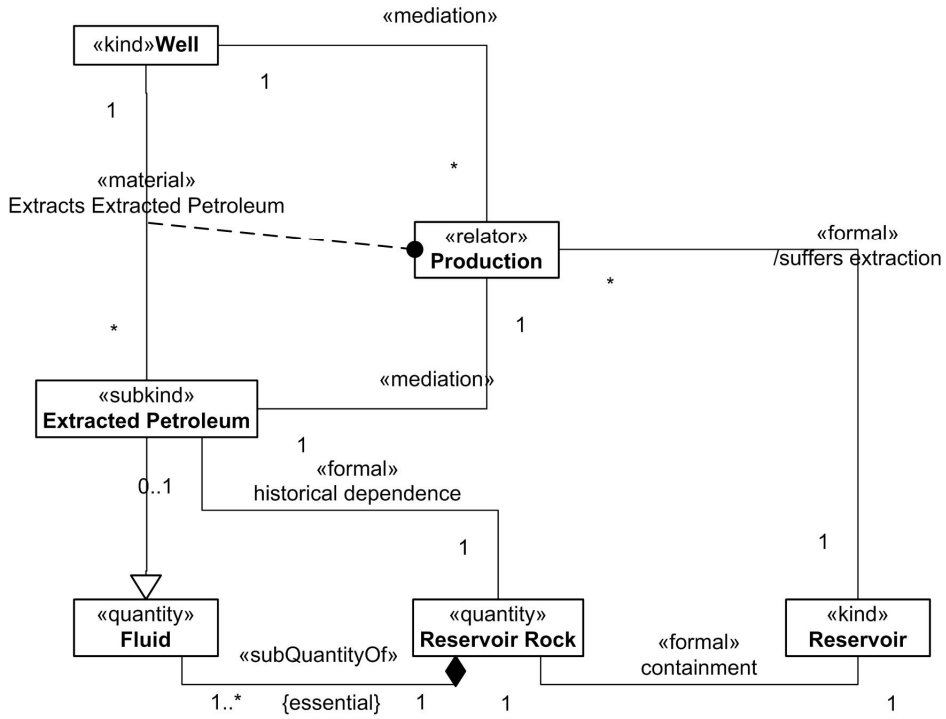

Fig. 8. Extracted Petroleum and its historical dependence to a Reservoir Rock

In the original model of Figure 7, the *Is_a_mechanism_to_extract_from_Reservoir* has a semantics which is unclear. What is represented in this case? That a Well could be used to extract something (extract what?) from a Reservoir? That a Well is in fact used to extract something (again, what?) from a Reservoir? The first possibility is eliminated by the fact that in an instance level this relation is to be established between specific wells and specific reservoirs (as opposed to Well types and Reservoir types). Moreover, the implicit relation with something which is supposed to be extracted makes clear that this should be at least a ternary relation. When the foundation of this relation (i.e., the relator) is made explicit, however, it becomes clear that a Production event involves directly only a Well and a portion of Extracted Petroleum. The relation between that Production event (and consequently between the Well it is dependent of) and the Reservoir is an indirect derived one.

Usually, the type of derivation axiom needed to define a relation such as *suffers_extraction* does not have a corresponding visual syntax in the language and, as

consequence, must be represented as part of the axiomatization of the ontology at hand. A recent version of OntoUML, OntoUML-R (das Graças, 2008), allows the visual representation of domain axioms (rules). This is achieved by combining the original OntoUML with the relevant portion of the URML (UML-Based Rule Modeling Language) (Lukichev & Wagner, 2006) metamodels. The latter has been conceived as a version of standard UML which includes a syntax extension for the visual representation of derivation, integrity and reactive rules. Figure 9 below illustrates the visual representation of the Derivation Rule that defines the *suffers_extraction* relation. In this example, the ellipse termed DR: *suffers_extraction* represents this rule with the following semantics: let pro, pet, roc and res be instances respectively of Production, Extracted Petroleum, Reservoir Rock and Reservoir. If (pet is mediated by pro , pet is a subQuantity of roc and roc is contained in res), then (res *suffers_extraction* of pro).

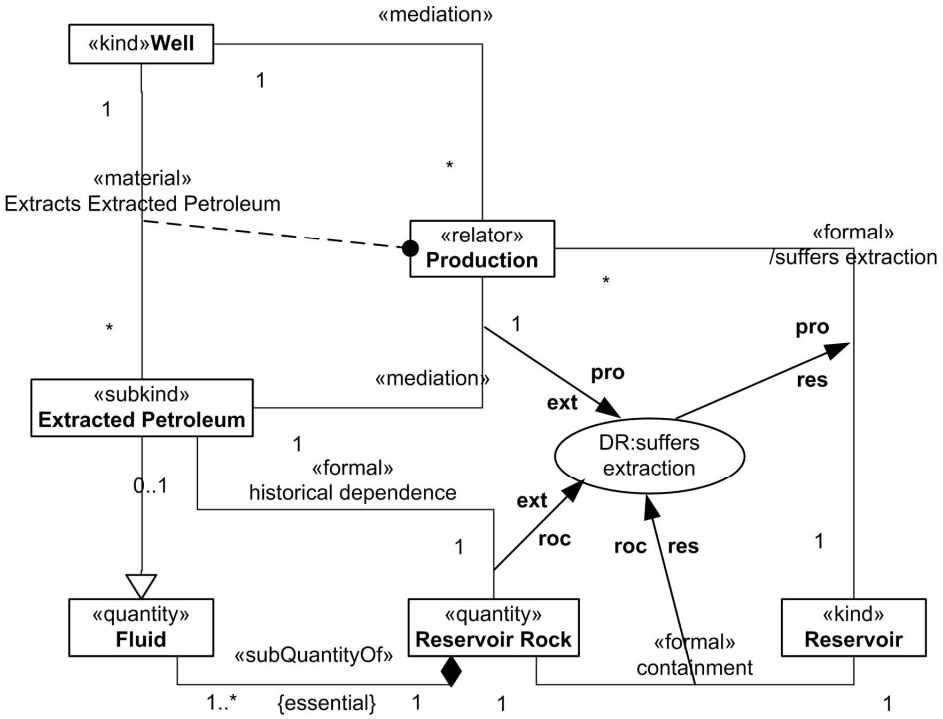

Fig. 9. Visual Representation of a Derivation Axiom and Derived Relation in OntoUML-R

Reserves as *Modes*

Figure 10 represents the notions of a Reservoir's Reserve, a Production Zone and a Reserve Class. A Reserve represents a quantity of Fluid which is economically viable to be extracted from a Reservoir. Each Reservoir is composed of several Production Zones and each of these are related to a number of Reserve Classes, namely, possible reserve, probable reserve, developed producing reserve, non developed producing reserve and developed (but) non producing proved reserve.

There are several questionable decisions in this model. In this original representation, reservoir is a number (value), which is both associated to Reservoir and Fluid. The idea behind this is that if x is a Reservoir, y is a Fluid contained in x and z is the Reservoir then z represents the "recoverable" portion of Extracted Petroleum from x which is supposed to be the Fluid y contained in x. However, there is no constraints in the model that guarantee this, i.e., that the instance of Fluid which is related to z is the same as the one contained in x. In principle, a solution would be to simply have the relation between Reserve and Fluid derived from the relations between Reserve and Reservoir and Reservoir and Fluid as discussed in the

previous section. The problem is that Fluid in the original ontology does not have specific portions of Fluid as its instances. Now, there are two important observations at this point. Firstly, reserve is the economically viable recoverable quantity of Petroleum, not of any Fluid. Thus, even if the relationship between Reserve and Fluid would make sense, it should in principle be defined between Reserve and (Extracted) Petroleum. However, since for the original ontology, there is only one instance of Petroleum (the fluid type) and reserves are always related to Petroleum, this relationship is simply superfluous.

Another problematic modeling decision in Figure 10 is to relate Reserve with a Reservoir but to classify as a Reserve Class the Production Zones. In this model, one can only know the total Reserve of a Reservoir and then know in which Reserve Classes each Production Zone is classified. During the interviews conducted with domain experts for the process of evolving this ontology via its transformation to OntoUML, it became clear that the intended conceptualization differed radically from this model at this point. Firstly, a Reserve is not a single value but a composition of values for each Reserve Class, i.e., one should have a predication of the amount petroleum in each of these classes (the possible amount, the probable amount, the developed producing amount, etc.). Secondly, one should have this information for each Production Zone. Thirdly, a Reserve can in principle change (it can change one of the estimations for a specific Reserve Class) and still be considered the same overall estimation (same Reserve) and Production Zone can have different Reserves associated with it in different periods of time.

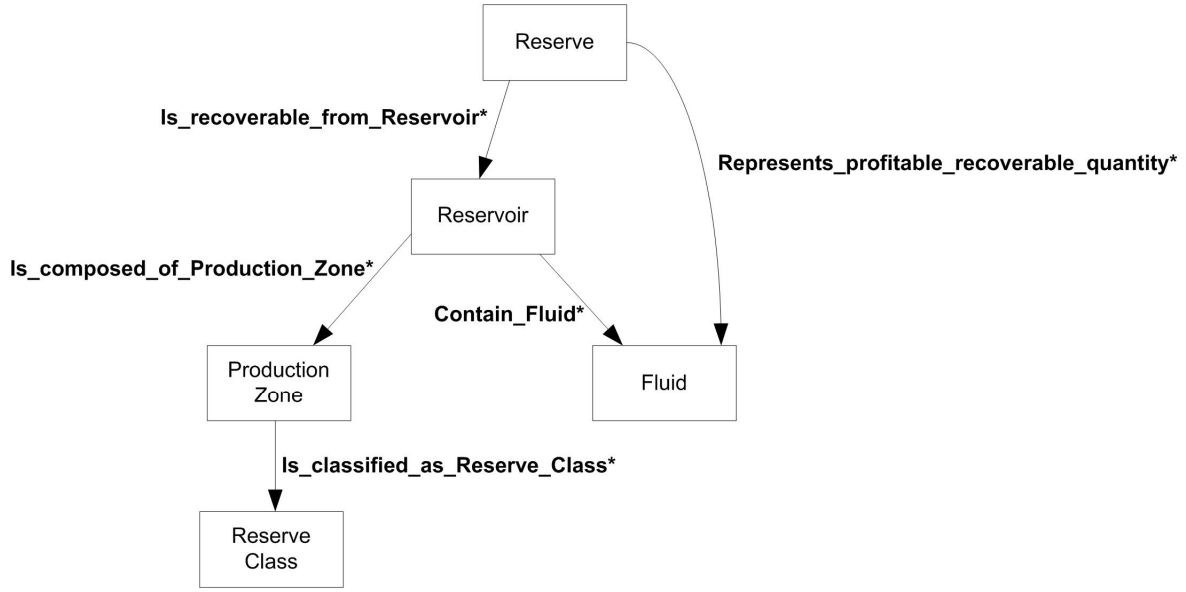

Fig. 10. Representation of Reservoir, Reservoir Classes and Production Zones (OWL)

OntoUML makes a fundamental distinction between two sorts of enduring entities, namely, ***Objects*** (e.g., a person, a car, a restaurant, a crowd, the moon, a forest, a portion of extracted petroleum) and a ***Moment*** (also known as ***Trope*** or ***Objectified Property***). The difference between the two categories is made using the ontological relation of existential dependence. Whilst an Object can exist by itself, a Moment can only exist in another individual in the way, for example, that a charge in an electric conductor existentially depends on that conductor, that a hole in a piece of cheese depends on the latter, or that a headache existentially depends on a patient. An example of Moment is the category of Relator previous discussed: a Marriage between John and Mary can only exist if John and Mary exist. A Mode is another category of moment. However, whilst a Relator characterizes a relational property (or a set of relational properties) of individuals and hence, is existentially dependent of this

multitude of individuals, a Mode characterizes an intrinsic property (or set of intrinsic properties) of one single individuals. Again, suppose that both John and Peter have headaches x and y, respectively. Even these two headaches instances are qualitatively indistinguishable, we have that x is existentially dependent on John whilst y is existentially dependent on Peter. Notice that x and y can have their own properties (e.g., duration, graveness) which can vary independently, can have different relations with other symptoms (e.g., causality), and so on.

In figure 11 below, we use the notion of Modes in the model. Each reserve is existentially dependent and characterizes a single Production Zone during a period of time. A Production Zone can be characterized by several Reserves in different periods and a Reserve has its own properties which refer both an attribute value space of time (date) and a numeric value space in which a value is assigned corresponding to each Reserve Class.

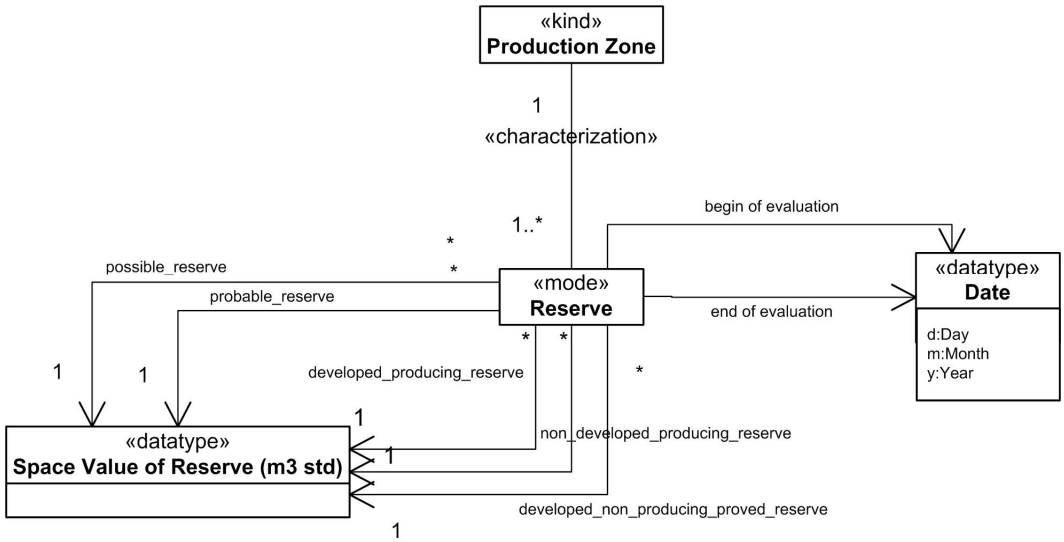

Fig. 11. Representation of Reserves as Modes as Reserve Classes as Attribute Functions assigning values in a Attribute Value Space (OntoUML)

Conclusions

An ontology engineering process is composed of phases, among them are conceptual modeling and implementation. During the whole process, the ontology being built must be made explicit by a representation language. The diverse ontology representation languages available in the literature contain different expressivity and different ontological commitments, reflecting on the specific set of available constructs in each one of them. Therefore, different ontology representation languages, with different characteristics, are suitable to be used in different phases of the ontology engineering process so as to address the different set of requirements which characterize each phase. In particular, conceptual ontology modeling languages aim primarily at improving understanding, learning, communication and problem solving among people in a particular domain. Therefore, these languages have being designed to maximize expressivity, clarity and truthfulness to the domain being represented. In contrast, ontology codification languages are focused on aspects such as computational efficiency and tractability and can be used to produce computationally amenable versions of an ontologically-well founded reference conceptual model. The inadequate use of a representation language, disregarding the goal of each ontology engineering phase, can lead to serious problems to database design and integration, to domain and systems requirements analysis within the software development processes, to knowledge representation and automated reasoning, and so on.

This article presents an illustration of these issues by using an industrial case study in the domain of Oil and Gas Exploration and Production. The case study consists in the generation of a Conceptual Ontological Model for this domain from an existing domain ontology in the organization where the case study took place.

The ontology representation language used to produce the redesigned model is OntoUML. OntoUML is a theoretically sound and highly expressive language, explicitly designed to have as modeling primitives, representations of the ontological distinctions prescribed by a Foundation Ontology. The hypothesis tested in this work is that a modeling language that is less expressive than OntoUML in the ontological sense (i.e., that lacks representation of these ontological distinctions) is prone to produce conceptual domain ontologies which are incomplete, ambiguous and less truthful to the portion of reality being represented. This hypothesis has been confirmed by our case study. The choice of OntoUML highlights a number of explicit concepts and ideas (tacit domain knowledge) that were implicit in the original model coded in OWL-DL. Moreover, the case study shows that a well-founded language such as OntoUML is capable of highlighting this tacit knowledge not only because it is able to express the underlying subtle ontological distinctions. It also demonstrates that the rich choice of modeling primitives offered by the language (representing these distinctions), as well as the methodological support of the formal meta-properties used define them (e.g., rigidity, existential dependence), forces the modeler to make the assumed ontological commitments explicit.

In particular, the aspects of ontological incompletes of OWL-DL which are made explicit by this case study are summarized as follows: (i) The distinction between formal and material relations; (ii) the representation of the foundation (truth-makers) of material relations which, in turn, allows for eliminating the intrinsic ambiguity which arises when material relations are represented as simple set-theoretical associations; (iii) the distinction between rigid object types and contingent object types. This, in turn, allows for the identification of the model's stable backbone structure, i.e., those essential types which are instantiated by their instances in all possible situations (in a modal sense); (iv) an ontologically-sound treatment of masses (quantities) and the consequent implications for conceptual modeling (e.g., the elimination of the intrinsic ambiguity of cardinality constraints in relationships involving quantities); (v) distinctions among primitive formal relations such as different kinds of parthood relations, containment, historical dependence, mediation, characterization. Incorporating these relations as primitives in the language also enables the definition of their formal meta-properties (e.g., reflexivity, transitivity, shareability, existential dependence) which, in turn, have notorious important consequences both to the design and implementation of an information system as well as to the automated processes of reasoning and model evaluation; (iv) the distinction between existentially independent enduring entities (Objects) and existentially dependent ones (Moments). This distinction has also important consequences for the design and implementation of information systems, which must reinforce that the lifecycle of existentially dependent entities are included in the lifecycle of their bearers.

To cite just one example of the impact of such incompletess, in the original representation of Conduct_Fluid relationship, it is possible to define that a duct can conduct several fluids and a fluid can be conducted by several different ducts. However, the lack of the Fluid Transportation concept (a relator uncovered by the methodological directives of OntoUML) hides important information about the domain. For instance, it is not explicit in this case how many different fluids can be transported at the same time or even if a duct can have more than a fluid transportation at a time. By making these concepts explicit as well as defining a precise real-world semantics for the notions represented, the newly E&P-Reservoir ontology produced in OntoUML prevents a number of ambiguity and interoperability problems which

would likely be carried out to subsequent activities (e.g., database design) based on this model.

It is well known in many engineering disciplines (e.g., software and data engineering) that the benefits of using complex modeling techniques become more evident once we deal with complex domains and/or complex tasks. However, many of the examples currently used in the literature to promote the use lightweight modeling languages such as the Semantic Web languages are either conducted in simple domains (e.g., the pizza ontology, the travel agency ontology) and/or are cases worked with a focus on structuring the subject domain using a limited subset of possible relations (e.g., taxonomic structures). Here, the intrinsic complexity of the chosen domain as well as of the industrial settings poses a clear demand to a plethora of ontological distinctions to be used in order to make a number of subtle domain notions explicit.

Despite becoming more evident in complex cases, the interoperability problems which can arise from the lack of expressivity of modeling languages are not limited to these cases. As previously mentioned, in (Guizzardi, 2006) a case study is carried out to demonstrate that a number of these problems occur even if simple lightweight Semantic Web ontologies are integrated. These case studies are in line with the Harvard Business Review report of October 2001 which claims that *"one of the main reasons that so many online market makers have foundered [is that] the transactions they had viewed as simple and routine actually involved many subtle distinctions in terminology and meaning"*. Thus, we defend that the relative high-complexity of the techniques employed here are compensated by the improved quality of the resulting models w.r.t. their capacity to support the tasks of domain understanding, meaning negotiation and semantic interoperability. Nonetheless, we have been working on a number of design patterns (Guizzardi et al., 2004; Guizzardi, 2009) as well as automated tool support (Benevides & Guizzardi, 2009) in order to, as much as possible, shield the user of OntoUML from the complexity of the language.

Finally, following the methodological principles behind OntoUML, once a conceptual model (ontology) is produced for a given domain, a number of specifications in different implementation environments can be derived from it, satisfying different sets of design (non-functional) requirements and application scenarios. We have defined an automatic transformation from OntoUML models to the Alloy language (Jackson, 2006), so that OntoUML models can be simulated by using a visualization extension built on top of the Alloy Analyzer (Benevides et al., 2009). We are currently also working on defining an additional automatic transformation from OntoUML-R to OWL/SWRL, so as to automatically generate an enhanced version of the OWL-DL/SWRL codified E&P-Reservoir Ontology. This enhanced lightweight model, in turn, shall contemplate the domain concepts uncovered by the process described in this article and, due to the combination of OWL-DL and SWRL, afford a number of more sophisticated reasoning tasks.

References

Artale, A., & Keet, M. (2008). Essential and Mandatory Part-Whole Relations in Conceptual Data Models. *21st International Workshop on Description Logics*, Dresden.

Baiao, F., Santoro, F., Iendrike, H., Cappelli, C., Lopes, M., Nunes, V. T., & Dumont, A. P. (2008). Towards a Data Integration Approach based on Business Process Models and Domain Ontologies. *10th International Conference on Enterprise Information Systems (ICEIS 2008)*, Barcelona, pp. 338-342.

Benevides, A. , & Guizzardi, G. (2009) A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML. *11th International Conference on Enterprise Information Systems (ICEIS)*, Milan. Lecture Notes in Business Information Processing, Springer-Verlag. Available at http://code.google.com/p/ontouml/

Benevides, A., Guizzardi, G., Braga, B., & Almeida, J. (2009). Assessing Modal Aspects of OntoUML Conceptual Models in Alloy. International Workshop on Evolving Theories of Conceptual Modeling (ETheCoM 2009), ER 2009, Gramado, Brazil.

Burek, P. et al. (2006). A top-level ontology of functions and its application in the Open Biomedical Ontologies. *Bioinformatics 22*(14), pp. e66-e73.

Chum, F. (2007). *Use Case: Ontology-Driven Information Integration and Delivery - A Survey of Semantic Web Technology in the Oil and Gas Industry*. W3C. Available at http://www.w3.org/2001/sw/sweo/public/UseCases/Chevron/. Accessed in dec 2007

das Graças, A. (2008). *Extending a Model-Based Tool for Ontologically Well-Founded Conceptual Modeling with Rule Visualization Support*. Computer Engineering Monograph, Ontology and Conceptual Modeling Research Group (NEMO), Federal University of Espirito Santo, Brazil.

Fielding, J. et al. (2004). Ontological Theory for Ontology Engineering. In: *International Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, Whistler.

Guarino, N. (1998). Formal Ontology and Information Systems. In: *1st International Conference on Formal Ontologies in Information Systems*, 3-15, Trento.

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*,Telematica Instituut Fundamental Research Series 15, ISBN 90-75176-81-3, Universal Press, The Netherlands.

Guizzardi, G. (2006). The Role of Foundational Ontologies for Conceptual Modeling and Domain Ontology Engineering, . Keynote Companion Paper at the *7th International Baltic Conference on Databases and Information Systems*, Vilnius.

Guizzardi, G. (2007), On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, Olegas Vasilecas, Johan Edler, Albertas Caplinskas (Editors), ISBN 978-1-58603-640-8, IOS Press, Amsterdam.

Guizzardi, G. (2009), The Problem of Transitivity of Part-Whole Relations in Conceptual Modeling Revisited, 21st International Conference on Advanced Information Systems Engineering (CAISE 2009), Amsterdam.

Guizzardi, G., &; Halpin, T. (2008). Ontological Foundations for Conceptual Modeling, *Applied Ontology 3(1-2)*, ISSN 1570-5838.

Guizzardi, G., &; Wagner, G. (2008). What´s in a Relationship: An Ontological Analysis, *27th International Conference on Conceptual Modeling (ER 2008)*, Barcelona.

Guizzardi, G., Wagner, G., Guarino, N., &, van Sinderen, M. (2004). An Ontologically Well-Founded Profile for UML Conceptual Models. *16th International Conference on Advances in Information Systems Engineering (CAiSE)*, Latvia.

Gulla, J.A., Strasunskas, D., & Tomassen, S.L. (2006). Semantic Interoperability in Multi-Disciplinary Domain – Applications in Petroleum Industry. Proceedings of the *2nd International Workshop on Contexts and Ontologies: Theory, Practice and Applications*. Riva del Garda, Italy.

Gulla, J.A., Tomassen, S.L., & Strasunskas, D. (2006). Semantic Interoperability in the Norwegian Petroleum Industry. *5th International Conference on Information Systems Technology and its Applications - ISTA´2006*, 81-93, Klagenfurt.

Gurr, C.A. (1999), Effective Diagrammatic Communication: Syntatic, Semantic and Pragmatic Issues, Journal of Visual Languages and Computing, 10, 317-342.

Jackson, D. (2006). *Software Abstractions: Logic, Language, and Analysis*. MIT Press.

Keet, M., & Artale, A. (2008). Representing and Reasoning over a Taxonomy of Part-Whole Relations. In Guizzardi, G. and Halpin, T. (Editors), *Special Issue on Ontological Foundations for Conceptual Modeling, Applied Ontology 3 (1-2)*, 91-110, ISSN 1570-5838.

Levesque, H., & Brachman, R. (1987). Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence, 3(1)*, 78-93.

Lukichev S., & Wagner, G. (2006). UML-Based Rule Modeling with Fujaba. Proceedings of the *4th International Fujaba Days 2006*, University of Bayreuth, Germany, 31–35.

Mealy, G. H. (1967). Another Look at Data. *AFIPS Conference Proceedings 31*, Washington, DC. Thompson Books, London. Academic Press, 525–534.

Mylopoulos, J. (1992), Conceptual modeling and Telos, In P. Loucopoulos and R. Zicari, editors, Conceptual modeling, databases, and CASE, chapter 2, pages 49--68. Wiley, 1992.

Norheim, D., & Fjellheim, R. (2006) "AKSIO – Active Knowledge Management in the Petroleum Industry", 3rd European Semantic Web Conference – ESWC´2006, Budova, Montenegro, 2006.

Paap, O. (2006). Accelerating Deployment of ISO 15926 (ADI). *FIATECH Member Meeting*.

Papazoglou, M., Traverso, P., Dustdar, S., & Leymann, F. (2008). *Service-Oriented Computing: a Research Roadmap, Int. J. Cooperative Inf. Syst. 17(2)*. 223-255

Sandsmark, N., & Mehta, S. (2004). Integrated Information Platform for Reservoir and Subsea Production Systems. Proceedings of the *13th Product Data Technology Europe Symposium, PDT´2004*, Stockholm

Smith, B. et al. (2005). Relations in biomedical ontologies, *Genome Biology 6(5)*.

Thomasson, A. L. (1999). *Fiction and Metaphysics*. Cambridge University Press, ISBN-13: 9780521065214.

TietoEnator (2006). Production Data Reporting— Standardisation. *POSC Integrated Operations (IntOPS)* SIG Regional Meeting, Houston.

Weber, R. (1997), *Ontological Foundations of Information Systems*, Coopers & Lybrand, Melbourne, Australia.

---

* Corresponding author (fernanda.baiao@uniriotec.br)

† We are aware of the fact that the stereotype powertype has been abandoned in the UML 2.0 specification (the concept of a higher-order type is, nonetheless, maintained). We maintain here the simpler original representation to avoid the need to discuss issues which are not germane to the purposes of this article. Examples include the very ontological nature of higher-order types and higher-order instantiation relations

‡ The preference for the term *Fluid_identity* instead of *Is_synonym_of* is motivated by the fact that the former refers to an identity relation among types while the latter refers merely to an identity relation among terms.