A Method for Ontology Modeling in the Business Domain[1]

Michele Missikoff, Federica Schiappelli

[*] LEKS, IASI-CNR
Viale Manzoni, 30 – 00185 – Rome, Italy
{missikoff, schiappelli}@iasi.cnr.it

Abstract. Today ontology languages present a syntax which looks not "natural" and are lacking of built-in primitives (i.e., modeling notions) domain experts are familiar with. In this paper we present an ontology representation method , OPAL, based on Object, Process and Actor primary concept categories. OPAL offers a number of modeling notions useful in the e-Business domain, but general enough to be used in diverse business sectors (such as automotive, tourism or banking). The domain expert, during the ontology modeling process, is required to identify the relevant concepts of the domain and to classify them according to OPAL categories. Furthermore, a set of semantic relations (such as is-a, part-of, relatedness) and some domain specific relations (generated-by, updated-by, roles, skills, etc.) can be used to describe the relationships among these concepts.

1. Introduction

Today ontology languages present a syntax which looks not "natural" and are lacking of built-in primitives (i.e., modeling notions) domain experts are familiar with. For instance OWL[1], currently a standard language for ontology representation, has a strong rooting in mathematical logic (in particular, DL: Description Logics [11]) and is well suited to be processed by inference engines. But OWL is inherently domain generic, i.e., it is not conceived for modeling one specific domain.

In this paper we present an ontology representation method that offers a number of modeling notions useful in the e-Business domain, but general enough to be used in diverse business sectors (such as automotive, tourism or banking). We believe that enhancing domain specificity of the ontology modeling language will support domain experts in their challenging task.

The proposed method includes an ontology representation paradigm, built on top of OWL, enriching its constructs to enhance domain specificity. To this end, we identify a number of conceptual categories (referred to as *concept kinds*) aimed at supporting the construction of ontologies in the business domain. In the conceptualization process, the domain expert can categorize the concepts identified observing the reality

[1] This work has been partially supported by the Athena Integrated Project, IST-507849, and the INTEROP Network of Excellence, IST-508011.

according to the given concept kinds. In essence, a concept kind represents a meta-concept that becomes a new modelling notion of the language.

The first concept categories proposed for the business domain are: Business Object, Business Process, Business Actor, Business Event, Business Message, Business Goal (for sake of conciseness, we will drop the term "business" in the rest of the document). Taking the first three conceptual categories, we refer to our approach as OPAL (Object, Process, Actor modelling Language).

The indicated categories have been selected after an analysis of the most relevant business modeling languages, such as PSL[12], Rosettanet[6], OAGIS[9], ebXML[8], BPML[7], and UML[5]. Particular attention has been paid to the latter that, even if originally conceived for software development, is progressively gaining popularity in business and enterprise modelling.

With OPAL we intended to get from UML and OWL the features that are best suited to model an e-Business application ontology. In particular we intend to keep the possibility of building complex enterprise models starting from a few basic categories (e.g., Object, Process, Actor, taken from UML. At the same time, we wanted to use the advanced reasoning and query services typical of DL and OWL. In summary, with OPAL we intend to propose an ontology representation method able to leverage on the positive features of the two, minimizing the negative aspects.

The rest of the paper is organized as follows: Sections 2 and 3 introduce the OPAL ontological framework, while Section 4 describes the OPAL templates. In Section 5 we give a first formalization of OPAL, by means of an abstract syntax. Section 6 concludes the paper.

2. Business Meta-Concepts[2] in OPAL

In this section, we introduce a first set of concept categories (meta-concepts) useful in modeling a business domain that are at the basis of the proposed ontology representation method.

Object (O) : gathers passive entities involved in one or more business processes.
Process (P): gathers business activities. A Process aims at accomplishing one or more business goals, operating on a set of Business Objects.
Actor(Ar): an active element of a business domain (e.g. Buyer, Supplier).
Operation (Op), represents an activity that is not further decomposable.
Complex Attribute and Atomic Attribute: in modeling the properties of a concept, we distinguish between structured information, such as "address", and elementary information, such as "street name". Essentially, a (structured) *Complex Attribute (CA)* is defined as an aggregation of lower level *CA* and/or *Atomic Attributes (AA).*

[2] We experimented already a preliminary version of the proposed method in two EU Projects: Fetish and Harmonise, where an interoperability platform, based on a tourism ontology, has been built. The proposed approach has been further elaborated within IDEAS and is currently at the base of the ontology research in two 6FP European Projects: INTEROP Noe and the Athena IP.

Actors, as Objects and Processes, can be organized according to specialization and decomposition hierarchies, in order to obtain a more structured view of the business domain. Processes can be recursively decomposed into *Processes* and, finally, into *Operations,* the latter being not further decomposable.

3. Semantic Relations in OPAL

Besides the concept categories, the OPAL model includes semantic relations defined among categories. The OPAL semantic relations represent well known modeling notions common to (the meta-model of) the majority of Knowledge Representation Languages:
ISA (generalization) relation among concepts. E.g., an *Invoice* ISA *Accounting Document*;
Predication: relating attributes to a concept. E.g., *Invoice* is in Predication with *Date*, *Amount*, *Recipient*, …;
Decomposition: part-of relationship among concepts. E.g., *Department* is in Decomposition with *Enterprise*;
Relatedness: domain specific relationships (*named or unnamed*) among concepts. E.g., the invoice is related to the customer (unnamed), the customer buys the product (named).

The above relations will be further elaborated in the sequel, when OPAL templates will be described.

To concretely illustrate how the OPAL method is used in building a business ontology, we introduce an example drawn from an e-Procurement process, focusing on the Order Processing phase.

In this process we identified two actors, the Buyer and the Supplier, some business documents that can be modeled as Objects, such as the Request for Quotation (RFQ), the Purchase Order (PO), the Invoice, and some sub-processes, such as the *sending* and *processing* of the RFQ, or the *Purchase Order issuing*.

4. The OPAL templates

In OPAL, a concept is specified according to a traditional Frame-Slot-Facet modeling paradigm[2]. In particular, there is a frame structure (template) for each concept category (kind, in OPAL), such templates are used in the interface of Athos to allow the user to introduce domain concepts in the ontology.

All the templates are characterized by three sections:

1. *Identification Section*, that essentially contains traditional metadata, such as the concept label, description, and other information (such as creation date, author) with an administrative flavour;
2. *Structural Section*, that contains the slots corresponding to the attributes (simple or complex) that will be instantiated in the corresponding object; it also contains the semantic relations (such as ISA, decomposition, etc.) with other concepts;

3. *Specific Section*, that contains information and references to other entities that play a specific role (predefined) in the correct definition of the concept.

The first two sections are the same for all the kinds, while the third section is designed specifically for each kind, with the aim to represent domain specific links and dependencies.

Below we briefly describe the first two common sections; furthermore, we present the specific section of the Object, Actor, Process, Operation, Complex and Atomic Attributes templates.

Identification Section

The *Identification Section* allows descriptive information about the concept to be specified. It contains

- *Label:* the preferred term to refer the concept;
- *Identifier:* the unique identifier for the concept;
- *Kind:* the concept category the concept belongs to;
- *XMLtag:* a tag that can be used to refer the concept in an XML document;
- *Description:* a natural language description of the concept;
- *References:* documental source used for the definition of the concept;
- *Terminology:* a set of terms that can be considered synonyms of the preferred term used as label
- *Author:* the person that entered the concept specification in the ontology;
- *Defined/Updated:* the date when the concept was first defined, and then updated.

Structural Section

A *Structural Section* is built by using a set of modeling notions derived from the OWL constructors, plus the *PartOf* relation (Decomposition).

Attributes and semantic relations have been introduced earlier, here a few additional issues are reported.

- *Concept constructors*: allow the new concepts to be defined, by using logical operators:
- Intersection ($\cap$), Union ($\cup$), Negation ($\neg$): it is possible to introduce a concept by defining it as the intersection (union/complement) of already existing concepts;
- *Concept Axioms*: formulas asserting specific constraints on concepts. These axioms, introduced in the Structural Section of the concept template, represent global constraints, valid every time we refer to the concept. The constraints that can be specified are:
  - Disjontness,
  - Equivalence;
- *Property Axioms*: formulas asserting specific constraints or describing particular characteristics of the properties. These axioms, introduced in the Structural Section of the template, represent local constraints applied to an instance of the concept. The constraints that can be specified are:
  - Cardinality constraints: allow to specify the cardinality of attributes and relations concerning the concept;
  - Explicit constraints: axioms expressed by using the OCL language. (Please note that in this phase, OCL constraints are not yet deployed.);
  - Property characteristics: Functional, InverseFunctional, Symmetric, Transitive;
- *Relatedness additional features*: as introduced in the previous section, the Relatedness relation allows to model domain specific relationships; these

relationships can be named or unnamed. In case of named Relatedness, a property *Rel_name* can be valued with the label of the relation. Furthermore the *Inv_rel_name* can be specified as the inverse name of the relation.

Having OWL as a formal rooting of OPAL, we had to cope with a number of misalignments caused by the different nature of the two modeling methods: logic-oriented and frame-oriented, respectively.

For instance, in OPAL we assume that the definition of a relation in the Structural Section is local with respect to the concept. In the OWL perspective, a relation defined in this section of the template corresponds to a property. We must guarantee that the domain of this property is a superset of the considered concept; furthermore, in the definition of this concept, a *Restriction* on the range of the property must be added. In the next sections we will see how these constraints are expressed.

An OPAL template is completed by the Specific Section, which is different for each concept category. The Specific Section gathers a number of domain-oriented relations and axioms, aimed at capturing additional semantics.

Object Specific Section

- *GeneratedBy, UpdatedBy, UsedBy, ArchivedBy*: the Processes that create, manipulate, archive the Object;
- *ObjOwner:* the Actors that have the responsibility of the whole lifecycle of the object;
- *States*, labelled boolean expressions over the Object attributes or those of related concepts;
- *Invariants:* specific constraints that must be always satisfied by the Object instances.

The last two items are currently treated informally. Their formal representation need further analysis of OCL.

Actor Specific Section

- *Goals:* objectives that must be accomplished by the Actor, in the form of an OCL expression (e.g., *salaries should be less that 60% of department budget*);
- *Skills:* indicating the actions that the Actor is able to perform or monitor (i.e., list of processes and/or operations);
- *Responsibilities:* the processes in which the Actor is involved, in achieving a Goal (as above), with his/her/its respective role (i.e., performer, controller, stakeholder, supporter), and the Objects he/she/it can manage;
- *Collaborations:* the other actors involved in the performed activities.

Process Specific Section

- *Creates, Updates, Enquires and Archives Business Objects:* here the business objects that are directly accessed or manipulated by the Process are indicated;
- *In, Out and Fault Messages:* the incoming, outgoing messages of the Process. The *Fault messages* allow to model the exceptions handling;
- *Actors*: the actors that are requested for the accomplishment of the Process (inverse of Actor *Skills*);
- *EstimatedTime*: the estimation of the Process execution time.

Operation Specific Section
The *Specific Section* of Operation is exactly the same of the Process. What differentiates the Operation from the Process is the fact that the former is not further decomposable.
Atomic Attribute Specific Section
- *Domain*, the set of concepts for which a predication with the Attribute can be established. Please, note that, if this property is valued with a list of concepts, the domain must be interpreted as the union of the set of instances of the specified concepts;
- *Range,* the basic type of the attribute. It assumes its values in the following set of values: string, integer, real and boolean;
- *Functional*, boolean value that says if for each instance of the Domain there is at most one instance of the Attribute (if more than one, then they represent the same object);
- *InverseFunctional* boolean value that says if for each instance of the Range there is at most one instance of the Domain.

Please, note that in the Specific Section of the Atomic Attribute the range of the Attribute can be specified. This assertion has a global impact: it means that the concept indicated as range must be considered as the superset of the ranges declared elsewhere for this attribute.

In the Structural Section of a concept c_i it is also possible to specify the type of the Attribute when it is related via a Predication (or Decomposition) to c_i. In this case, that specification must be considered as a further restriction imposed over the range of the Attribute; this restriction is local to the definition of c_i.

Similarly to the Range, also the Property Characteristics (e.g. Functional) can be specified in the Specific Section of the Attribute. In this case, every time the Attribute is related to a concept, the constraint must be satisfied. Otherwise, the constraint can be specified in the Structural Section of a given c_i; in this case the constraint will be local to the definition of the c_i Predication (or Decomposition).

All the costraints specified in the Specific Section must be considered as global, while the constraints specified in the Structural Section are local.
Complex Attribute Specific Section
- *Domain*, the set of concepts for which a predication with the Attribute can be established;
- *Functional*, boolean value that says if for each instance of the Domain there is at most one instance of the Attribute (if more than one they represent the same object);
- *InverseFunctional* boolean value that says if for each instance of the Range there is at most one instance of the Domain.

As well as the Atomic Attribute, the constraints specified in the Specific Section of the Complex Attribute must be interpreted as global constraints over the Attribute.

In the remaining part of the paper we will present a first formalization of the OPAL Ontology meta-model. This formalization includes the abstract syntax of the OPAL templates.

5. Formalization of the OPAL Meta-Model

In this section we illustrate, with an algebraic approach, the formal structures that compose the definition of a concept. It can be seen as the abstract syntax of the representation language.

5.1 OPAL Abstract Syntax

Let O be an OPAL ontology , $O = (C, R)$

$C = \{c_i, i \in N\}$ is a finite set of concepts

$R = \{(c_i, c_k)\ i, k \in N\}$ is a finite set of relations established among concepts in C.

In particular we have

- *ISA* the set of Generalization relations defined in O;
- *D* the set of Decomposition relations defined O;
- *Pr* the set of Predication relations defined in O;
- *R* the set of Relatedness relations defined in O;

Furthermore, R includes also the set of relations in the *Specific Section* of each template (*Sp*).

$R = ISA \cup D \cup Pr \cup R \cup Sp$

A concept $c_i \in C$, is defined as a triple

$c_i = (IdentSect_i, StructSect_i, SpecSect_i)$

where each element is a set of functions:

IdentSect$_i$= [label(c_i), id(c_i), kind(c_i), XMLtag(c_i), descr(c_i), ref(c_i), terminology(c_i), author(c_i), def(c_i)]

- *label(c_i) $\in$ String*; the label of the concept (the label is mandatory);
- *id(c_i) $\in$ String*; the unique identifier of the concept;
- *kind(c_i) $\in$ {Ar,O,P,AA,CA,Op,M, BOD}*; the concept category the concept belongs to (the kind is mandatory. The pair (label(c_i), kind(c_i)) must be unique);
- *XMLTag(c_i) $\in$ String*; an XML tag that can be used to refer the concept in an XML document;
- *descr(c_i) $\in$ String*; a natural language description of the concept;
- *ref(c_i) $\in$ String*; sources for the concept definition;
- *terminology(c_i) $\in$ String*; a set of terms that are considered synonyms of the concept;
- *author(c_i) $\in$ String*; the name of the author;
- *def(c_i) $\in$ Date*; the date of the concept definition.

In the Purchase Order (PO) example, we will have:

IdentSect (PO) = [

label(c_i)= Purchase Order,

id(c_i) = “PO”

kind(c_i) = “O”,

XMLTag(c_i) = “<po>”

descr(c_i) = “A printed or typed document, issued by the Buyer Purchasing FU as a firm and formal request to a specific Manufacturer/Supplier to produce and supply goods/services according to Price, Terms and Conditions previously agreed and approved.”
ref(c_i) = “ebXML”
terminology(c_i) = {Order, Product Order, Service Order}
author(c_i) = Federica
def(c_i) = 02-04-04**]**.

StructSect$_i$= [ISA(c_i), Pr(c_i), D(c_i), R(c_i), intersectionOf(c_i), unionOf(c_i), not(c_i)]

- $ISA(c_i) = \{c_k \mid (c_i,c_k) \in ISA , k \in \{1,\ldots,n\}\}$; it gathers the set of superconcepts of c_i;
- $Pr(c_i) = \{c_k \mid (c_i,c_k) \in Pr , k \in \{1,\ldots,n\}\}$; it gathers the set of c_i attributes;
- $D(c_i) = \{c_k \mid (c_i,c_k) \in D , k \in \{1,\ldots,n\}\}$; it gathers the components of c_i;
- $R(c_i) = \{c_k \mid (c_i,c_k) \in R , k \in \{1,\ldots,n\}\}$; it gathers the set of concepts related to c_i;
- *intersectionOf* $(c_i) = \{ c_1 ,.., c_k \}$; it indicates that the concept c_i is defined as the intersection of the concepts $c_1, .. , c_k$;
- *unionOf* $(c_i) = \{ c_1 ,.., c_k \}$; it indicates that the concept c_i is defined as the union of the concepts $c_1, .. , c_k$;
- *not* $(c_i) = \{ c_k , k \in \{1,\ldots,n\}\}$; it indicates that the concept c_i is defined as the complement of the concept c_k.

For the Purchase Order, in the Structural Section we will have:

- **ISA**(PO) = {AccountingDocument};
- **Pr**(PO)={PONumber, OrderLine, OrderDate, Charge, PaymentTerms, TransportTerms, DeliveryDate};
- **D**(PO) = {BusinessDocumentArchive};
- **R**(PO) = {DeliveryNote, Invoice, Payment, Product, Shipping List}.

These relations will be further detailed in the next section.

For what concerns the disjointness and equivalence Concept Axioms, they are represented by functions

- *disjoint*: $P(C) \rightarrow$ {‘true’, ‘false’}, where $P(C)$ is the power set of C ;
- *equivalent*: $P(C) \rightarrow$ {‘true’, ‘false’}, where $P(C)$ is the power set of C

The assertion disjoint($c_1 ,.., c_k$) means that { $c_1 ,.., c_k$ } is a set of pair wise disjoint concepts.

The assertion equivalent($c_1 ,.., c_k$) means that { $c_1 ,.., c_k$ } is a set of pairwise equivalent concepts.

Property Axioms, Cardinality constraints and Property Characteristics are expressed as binary functions that take in input a pair of concepts.

When a Predication (or Decomposition) relation is established between two OPAL concepts, the type of the attribute can be specified, by using the functions *type* and *dataRange*. These functions can be applied to a pair of concepts related by either the Predication or the Decomposition relation, depending on the kind of the second element of the pair.

If the Decomposition relation is established between a Complex Attribute and an Atomic Attribute, the type of the attribute must be specified by using the type and, possibly, the dataRange functions.

− *type*: $D \cup Pr \rightarrow$ {'string', 'int', 'real', 'boolean'}.
  NOTE: *type* is a partial function; *type*((c_i,c_k))=undefined if *kind*(c_k) $\neq$ AA
  In the Purchase Order example:
    type(PurchaseOrder,PONumber)='int'

Please note that this approach implies the locality of typing. In fact, the same attribute label may be bound, for another concept, to a different type. This is one of the main differences between the frame-oriented approach of UML, adopted in OPAL, and the logical approach of OWL.

− *dataRange*: $D \cup Pr \rightarrow$ INTERVAL $\cup$ ENUM

This function specifies a restriction on the values the attribute can assume. We indicate with INTERVAL a subset of INT $\times$ INT or REAL $\times$ REAL. We also indicate with ENUM a finite set of STRINGS or INT.

dataRange can assume the following values:

- *dataRange*((c_i,c_k)) = (x,y) $\in$ INT $\times$ INT if *type*((c_i,c_k)) = 'int';
- *dataRange*((c_i,c_k)) = (x,y) $\in$ REAL $\times$ REAL if *type*((c_i,c_k)) = 'real';
- *dataRange*((c_i,c_k)) = x $\in$ ENUM if *type*((c_i,c_k)) = 'string';

  NOTE: *dataRange* is a partial function, it is undefined if *kind*(c_k) $\neq$ AA.
  In the Purchase Order example:
    type(PO,PaymentTerms)='string'
    dataRange(PO,PaymentTerms)={cash, credit card, bank transfer} $\in$ ENUM.

Furthermore, cardinality constraints can be specified for an attribute or a relation, by using the following functions:

− *minCard*(c_i,c_k) $\in$ INT
  NOTE: if *minCard* is not specified, 0 is assumed.
− *maxCard*(c_i,c_k) $\in$ INT
  NOTE: if *maxCard* is not specified, it is unbounded.

Other Property Axioms can be expressed by using the following functions:

− *Functional*: $R \cup Pr \cup D \rightarrow$ {'true','false','not-spec'};
− *InverseFunctional*: $R \cup Pr \cup D \rightarrow$ {'true','false','not-spec'};
− *Symmetric*: $R \cup Pr \cup D \rightarrow$ {'true','false','not-spec'};
− *Transitive*: $R \cup Pr \cup D \rightarrow$ {'true','false','not-spec'}.

Finally, when a Relatedness is established and the relation is named, the label of the relation can be specified by using the *Rel_name* and the *Inv_rel_name* functions.

− *Rel_name*: $R \rightarrow$ STRING
− *Inv_rel_name*: $R \rightarrow$ STRING
  In the Purchase Order example:
    Rel_name(PurchaseOrder, Invoice)='originate'
    Inv_rel_name(PurchaseOrder, Invoice)='is_originated_by'

In the Specific Sections we can find the relations specific for each concept category; we give the specification of the Actor Specific Section. The Specific Sections of the other concept catergories are defined in a similar way.

ActorSpecSect$_i$ = [Goals(a_i), Skills(a_i), Responsabilities(a_i), Cooperations(a_i)]

- Goals(a_i) = $\{(l, d), l \in$ String, $d \in$ String$\}$;
- Skills(a_i) = $\{p \mid (a_i, p), p \in P\}$;
- Responsabilities(a_i) = $\{(p, r, o) \mid (a_i, p, r, o), a_i \in Ar, p \in P, r \in$ {"performer", "controller", "stakeholder", "supporter"}, $o \in O\}$;
- Cooperations(a_i) = $\{a_k \mid (a_i, a_k)\ a_k, a_i \in Ar, a_k \in Ar\}$.

5.2 OPAL Relations and inherent constraints

In the following, the OPAL relations are formally described and, for each of them, the inherent constraints of the OPAL model are presented. The examples illustrating the constraints are taken from the business domain.

Generalisation (ISA)

Generalisation (inverse: *Specialisation*) expresses the relation between a narrower and a broader concept. In our example the concept *AccountingDocument* is specialized into *PurchaseOrder*.

The first constraint is that *ISA* relation must be defined among concepts belonging to the same conceptual category (e.g., *Objects, Actors, Processes*); accordingly, *AccountingDocument* and *PurchaseOrder* are both Objects.

Formally, let c_i (i=1,…,n) be concepts, let *kind(c_i)* be the OPAL category of the concept c_i, the following constraint must be verified:

C1) $(c_i,c_h) \in ISA \Rightarrow kind(c_i) = kind(c_h)$

Furthermore the Specialisation relation is transitive and must be acyclic.

C2) $(c_i,c_h), (c_h,c_k) \in ISA \Rightarrow (c_i,c_k) \in ISA$

C3) there are no sequences $c_1..c_n$ such that $(c_1,c_2), \ldots (c_{n-1},c_n) \in ISA$ and $c_1=c_n$

Predication (Pr)

Predication expresses the relation among a concept and its attributes. For example the *PurchaseOrder* concept can have as attributes the concepts *PONumber* and *OrderLine*.

Since this relation typically connects a primary concept with *Complex Attributes* (the *OrderLine*, in our example) or *Atomic Attributes* (the *PONumber*), it must be defined between concepts whose kinds are defined as follows:

C4) $(c_i,c_h) \in Pr \Rightarrow kind(c_i) = O \mid Ar \mid P, kind(c_h) = CA \mid AA)$

When a Predication relation is established between two OPAL concepts, the kinds of the concepts are constrained as indicated.

Decomposition (D)

Decomposition expresses the relation between a composite concept and a concept representing one of its components. For example an *Enterprise* can be decomposed into *Departments* or an *Address* can be decomposed into *City*, *Street*, *Number* and *ZipCode*.

The involved concepts must satisfy one of the following constraints, depending on their kinds:

C5) $(c_i,c_h) \in D$ and $kind(c_i) = Ar \mid O \Rightarrow kind(c_i) = kind(c_h)$

C6) $(c_i,c_h) \in D$ and $kind(c_i) = CA \Rightarrow kind(c_h) = AA \mid CA$

C7) $(c_i,c_h) \in D$ and $kind(c_i) = P \Rightarrow kind(c_h) = P \mid Op$

When a Decomposition relation is established between two OPAL concepts, the cardinality constraints can be specified by using the *minCard* and *maxCard* functions previously introduced.

Relatedness (R)

Relatedness expresses the fact that, between two concepts, a domain relation exists. Such domain relation can be labeled. For example an *Enterprise* can be related to a *Manager* with a relation named *manages*.

Furthermore, since the related concepts may be of different kinds one of the following conditions holds:

C8) $(c_i,c_h) \in R$ and $kind(c_i) = Ar \mid O \mid P \Rightarrow kind(c_h) = Ar \mid O \mid P$

C9) $(c_i,c_h) \in R$ and $kind(c_i) = CA \mid AA \mid Op \Rightarrow kind(c_i) = kind(c_j)$

When a Relatedness relation is established between two OPAL concepts, the cardinality constraints can be specified by using the *minCard* and *maxCard* functions previously introduced. A name can be specified for the relationship, as well as the name of the inverse relations. It can be done by using the functions

relName$(c_i,c_k) \in$ STRING

invRelName $(c_i,c_k) \in$ STRING

Specific Sections Constraints

Each relation in the Specific Section of an OPAL template must satisfy the following domain constraints:

Let S be a relation in the Object Specific Section:	**C10)** $(c_i,c_h) \in S \Rightarrow kind(c_i) = O$
Let S be a relation in the Process Specific Section:	**C11)** $(c_i,c_h) \in S \Rightarrow kind(c_i) = P$
Let S be a relation in the Actor Specific Section:	**C12)** $(c_i,c_h) \in S \Rightarrow kind(c_i) = A$
Let S be a relation in the CA Specific Sections:	**C13)** $(c_i,c_h)\in S \Rightarrow kind(c_i) = CA$
Let S be a relation in the AA Specific Section:	**C14)** $(c_i,c_h)\in S \Rightarrow kind(c_i) = AA$
Let S be a relation in the Operation Specific Section:	**C15)** $(c_i,c_h)\in S \Rightarrow kind(c_i) = Op$

Also the range of the relations in the Specific Sections is constrained.

C16) $(c_i,c_h) \in$ GeneratedBy $\Rightarrow kind(c_h) = P$

Similarly, the ranges of the other specific relations can be desumed by the description of the *Specific Section* given in the abstract syntax description.

In some Specific Section properties relate an object to a tuple. For instance, in the Actor Specific Section the Responsabilities relation is definied as follow:

Responsabilities$(a_i) = \{(p, r, o) \mid (a_i, p, r, o), a_i \in Ar , p \in P, r \in \{$"performer", "controller", "stakeholder", "supporter"$\}, o \in O \}$;

The range of this relation is constrained to assume values in the set of tuples [Process, role, Object]. This set is given by the cartesian product of the sets in which processes, roles and objects can assume values.

C17) $(a_i, p, r, o) \in$ Responsabilities $\Rightarrow (p, r, o) \in P \times Roles \times O$,

where Roles = {"performer", "controller", "stakeholder", "supporter"}

Similarly, the range of the other specific relations can be desumed by the description of the Specific Sections given in the abstract syntax description.

6. Conclusions to the Athos Representational Specifications

In this paper we presented an ontology modeling method aimed at supporting the domain experts, with limited ontology engineering knowledge, in the ontology building activities. Our work starts from the assumption that ontology languages, such as OWL, are not easy to use and adding domain specific modeling notions provides a helpful solution. Therefore we presented OPAL, an ontology framework that includes a few basic notions drawn from business modeling domain and from key constructs in UML.

In particular, the primary concept categories identified are: Object, Process, and Actor. The domain expert, during the ontology modeling process, is required to identify the relevant concepts of the domain and to classify them according to OPAL categories.

Furthermore, a set of semantic relations (such as ISA, Decomposition, Relatedness) and some domain specific relations (generated-by, updated-by, roles, skills, etc.) can be used to describe the relationships among these concepts.

We presented a first axiomatization of OPAL, aimed at a better understanding of its semantics. The axioms are represented by a set of constraints defined over the semantic relations. On a more practical ground, the axioms are used to supporting the verification, for an enhanced quality of the produced ontology.

In OPAL the structure of concepts is based on a Frame-Slot-Facet paradigm. In particular, the proposed frame structure (template) is organised in three sections: Identification, Structural, Specific Section. These sections have been presented with an algebraic approach that can be seen as the abstract syntax of the representation language.

The OPAL ontology framework has been used in building the Athos ontology management system. Athos, an open source system implemented on top of the Zope platform, has been developed within the European Integrated Project Athena and is currently under experimentation. The first results indicate that the proposed method is well accepted by business people and its supporting features have been recognised.

References

1. F.van Harmelen, I.Horrocks, P.F.Patel-Schneider, *OWL Web Ontology Language Semantics and Abstract Syntax*, W3C Candidate Recommendation 18 August 2003
2. M. Minsky, *Frame-system: Theory in Thinking*, University Press, London, 1977.
3. Fernandéz, Mariano; Gómez-Pérez, Asunción; and Juristo, Natalia. 1997.
4. METHONTOLOGY: From Ontological Art to Ontological Engineering. Workshop, on Ontological Engineering. Spring Symposium Series. AAAI97 Stanford.
5. On-To-Knowledge Methodology - Baseline Version, Hans-Peter Schnurr, York Sure, Rudi Studer (University of Karlsruhe), Hans Akkermans (Vrije Universiteit Amsterdam), On-To-Knowledge EU-IST-1999-10132 Project Deliverable D15, 2000.

6. OMG Group. *Unified Modeling Language (UML), version 1.5*. Available on-line: http://www.omg.org/technology/documents/formal/uml.htm.
7. http://www.rosettanet.org
8. Business Process Modeling Language (BPML). Alameda (CA): Business Process Management Initiative, 2001. Working Draft 0.4
9. ebXML Business Process Specification Schema. Version 1.01. OASIS and UN/CEFACT, 2001. Accessed 2002-04-11 2002. Available from http://www.ebxml.org/specs/ebBPSS.pdf
10. OAGIS: A 'Canonical' Business Language Providing both Vertical and Horizontal Requirements." By Michael Rowell (Chief Architect, Open Applications Group). Version 1.0. White Paper, 2002.
11. Unified Modeling Language (UML), version 1.5, 2003. Available at: http://www.omg.org/technology/documents
12. D. Nardi, R. J. Brachman. An Introduction to Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44.
13. Gruninger, M., Sriram, R.D., Cheng, J., Law, K., "Process Specification Language for Project Information Exchange," International Journal of IT in Architecture, Engineering & Construction, 2003
14. Eric S. K.Yu and John Mylopoulos. From E-R to "A-R" – modelling strategic actor relationships for business process reengineering. In Proc. of the 13th International Conference on the Entity-Relationship Approach – ER'94, Manchester (UK), December 13-16, 1994.