

Rapport de recherche

N° 04 001

**Context-explication
in conceptual ontologies:
PLIB ontologies and their use for
industrial data**

Guy PIERRA

Remark

Special issue of JAMS - Journal of Advanced Manufacturing Systems

Abstract

A number of computer science problems, including heterogeneous database integration, natural language processing, document intelligent retrieval would benefit from the capability to model the absolute meaning of things, independently of any particular use of these things. Such models, termed ontologies, have been heavily investigated over the last ten years, with various purposes and within various contexts. The goal of this paper is to investigate the role of ontologies for data integration and to present an ontology model that was developed to allow neutral exchange and automatic integration of industrialComponent catalogues and of technical data. We first present a taxonomy of ontologies into linguistic ontologies, based on words and usable for intelligent document processing, and concept ontologies, multilingual and usable for structured data management. We then discuss differences between ontologies and usual conceptual models. We claim that the main difference is the consensual nature of ontologies when conceptual models are specifically designed for one particular target system. Reaching consensus, in turn, needs specific models of which context sensitivity has been minimized. We identify four requirements for making ontologies less contextual than models and suitable for data integration, and we present how these requirements have been fulfilled in the PLIB ontology model developed to give meaning to technical data. Finally, we outline the use of PLIB-based ontologies in various domains including database integration, engineering component database development, electronic catalogues of industrial components generation and the semantic Web.

Keywords

Ontology, data integration, electronic catalogue, PLIB

Journal of Advanced Manufacturing Systems
© World Scientific Publishing Company

CONTEXT-EXPLICATION IN CONCEPTUAL ONTOLOGIES: PLIB ONTOLOGIES AND THEIR USE FOR INDUSTRIAL DATA

GUY PIERRA

*Laboratory of Applied Computer Science (LISI)
National Engineering School for Mechanics and Aerotechnics (ENSMA), Poitiers
86960 Futuroscope Cedex - France
pierra@ensma.fr*

Received (Day Month Year)

Revised (Day Month Year)

A number of computer science problems, including heterogeneous database integration, natural language processing, document intelligent retrieval would benefit from the capability to model the absolute meaning of things, independently of any particular use of these things. Such models, termed ontologies, have been heavily investigated over the last ten years, with various purposes and within various contexts. The goal of this paper is to investigate the role of ontologies for data integration and to present an ontology model that was developed to allow neutral exchange and automatic integration of industrial component catalogues and of technical data. We first present a taxonomy of ontologies into linguistic ontologies, based on words and usable for intelligent document processing, and concept ontologies, multilingual and usable for structured data management. We then discuss differences between ontologies and usual conceptual models. We claim that the main difference is the consensual nature of ontologies when conceptual models are specifically designed for one particular target system. Reaching consensus, in turn, needs specific models of which context sensitivity has been minimized. We identify four requirements for making ontologies less contextual than models and suitable for data integration, and we present how these requirements have been fulfilled in the PLIB ontology model developed to give meaning to technical data. Finally, we outline the use of PLIB-based ontologies in various domains including database integration, engineering component database development, electronic catalogues of industrial components generation and the semantic Web.

Keywords: Ontology, data integration, electronic catalogue, PLIB

1. Introduction

In the so-called information society, more and more information is computer-recorded. In any domain of human activity available information in so huge that computer are to be used to retrieve, to collect and to present information in a human understandable way. In the structured-data universe, information is represented as data. Indeed, a lot of research has been performed to integrate heterogeneous and autonomous data base ¹, in particular using ontologies ². Distributed architecture models have been developed, where mediators ³ provide uniform access to heteroge-

neous data sources. Mediators export integrated schemas that reconcile data both at the structural (schematic heterogeneity) and at the meaning level (semantic heterogeneity). If large progress have been made to automate schema integration at the structural level, using in particular new model management techniques⁴, the major challenge remains the automation of semantic integration of several heterogeneous schema. Such an automation would needs to make computer-interpretable:

- which data have exactly the same semantic meaning,
- which data are similar and may be converted in or compared with each other by defined process, and
- which data have no semantic commonality.

In the above list, data means either atomic data or structured data like tuple or entity instance. On the Internet, another way is also used for representing information, namely documents. Through html and xml, a generic (meta) structure was defined for gathering various documents in same semi-structured repositories. Huge progresses were achieved by search engines to retrieve over the Internet the most relevant documents with respect to a user query stated as a sentence of words. Unfortunately, if semantic of both the query and the target documents are not made computer-sensible, it is impossible to retrieve documents dealing with the query subject but without using exactly the same words (e.g., workers in place employees, size in place of length or convertible in place of car). Here again some kind of computer interpretable representation of word meaning is needed:

- in a first step to improve search engine in order to retrieve which documents are semantically relevant for a topics defined by a set of words, even when the same words are not used, and
- in a second step, to retrieve which information sources, either unstructured, semi-structured or structured provide exact answer to a user query.

Both kinds of information integration requiring explicit representation of meaning, these last ten years a lot of research has been done to develop ontology models intended to capture the a priori nature of reality, as independently as possible from any particular use of this reality. Once defined, such representations may then be used to reconcile various information sources at the meaning level.

The word ontology is now extensively used in a number of computer science domains: knowledge management, natural language processing, database, object oriented modeling, etc. If there seems to be some consensus on what an ontology structure should be - categories (classes), properties, logical relationships - the focus of the various approaches is so different that the same word seems to represent quite different realities, and that ontologies developed, e.g., for natural language processing seems to be nearly useless for e.g., database integration, and conversely.

The goal of this paper is to investigate the concept of an ontology in a structured-data perspective. It is also to show how the ontology model we have developed over the last 10 years in the PLIB standardization project (officially ISO 13584) may

be used in the various domains where the meaning of structured data need to be made computer-interpretable, like multidatabase, e-engineering, B2B electronic commerce and web services over the semantic web. The initial goal of PLIB was to allow engineering database integration and neutral exchange of electronic catalogues of industrial components.

The content of this paper is as follows. In the next section we discuss the various kinds of ontologies needed for representing semantics. We distinguish between document-oriented linguistic ontology (LO) and structured-data-oriented concept ontology (CO). In the third section we investigate the difference between ontologies and models. We claim that the major difference is the existence of a consensus that found ontologies as a shared meaning, and that consensus, in turn, needs explication of the modeling context. We introduce four mechanisms allowing to make ontology much more generic through context explication. In the fourth section we present how these mechanisms are represented in the PLIB ontology model to allow automatic integration of several structured data sources, and we present, in section 5, a formal model of PLIB ontologies, including the mapping capabilities to external ontologies. We outline in section 6 how such ontologies may be used for database integration, e-engineering and the semantic web. A discussion of related works is presented in section 7. Conclusion is presented in section 8.

2. Concept ontologies Versus Linguistic Ontologies

Since the term ontology was borrowed from philosophy by John Mc Carthy in the 70's and introduced in the computer science vocabulary, many definitions have been offered. The most commonly cited definition is one by T. Gruber "An ontology is an explicit specification of a conceptualization", therefore "shared ontologies" provide for "knowledge sharing" ⁵. In all the ontology models, such a conceptualization consist of three parts :

- *primitives items* of the ontology, where items are either classes or properties, are those items "for which we are not able to give a complete axiomatic definition ; we must rely on textual documentation and a background of knowledge shared with the reader" ⁵,
- *defined items* are those items for which the ontology provides a complete axiomatic definition by means of necessary and sufficient conditions, and
- *logical relationships* (or *inference rules*) provide for reasoning over ontology items, and for solving the problems for which the ontology was designed.

The agreed definition and structure description leave open what may be considered as the major criteria for classifying ontologies and ontology models: whether their area of interest consists of beings -what does exist in the world - or of word- how beings are apprehended and reflected in a particular natural language.

We call linguistic ontology (LO) those ontologies whose scope is representing the meaning of the words used in a particular Universe of Discourse (UoD) in a

particular language. We call concept ontology (CO) those ontologies whose goal is representing the categories of objects and of objects properties that are in some part of the word. These two kinds of ontologies address quite different problems and should have quite different content.

LO⁶ are document-oriented. The typical problem they address may be termed as follows:

"find all documents pertinent with respect to a query expressed as a set of words possibly connected by logical operators like AND, OR and NOT, even if these documents don't contain these words".

Since natural language contain a number of different words for reflecting identical or similar meanings, LO are large in nature. They include a number of *conservative definitions*, i.e., defined items that only introduce terminology and do not add any knowledge about the world⁵. They are language-specific and contain a number of linguistic relationships such that *synonym*, *hypernym*, *hyponym*, *overlap*, *covering*, *disjoint* to capture in a semi-formal way² meaning similarity. Such relationship being not formally grounded, inference could only provide some help to a user supposed to be involved in some computer-aided search process. Development of LO may be done through a semi-automatic process were significant words are automatically extracted from a document collection and then validated and structured by experts.

CO, for instance the measure ontology⁵, are structured-data-oriented. The typical problem they address may be termed as follows:

"decide whether two instances belong to the same beings class and whether two properties have identical meaning or may be converted in each other".

To be able to represent all the beings existing in some part of the world, CO need only to describe those primitive concepts that cannot be derived from other concepts. Like technical vocabulary where one and only one word should always be used for the same meaning, CO may be restricted to primitive concepts. Such primitive CO are compact in nature. To reduce again the number of concepts that need to be represented, CO may also be property-oriented. This means that in place of introducing a number of different concepts such that "10-HP-engine", "20-HP-engine", "25-HP-engine", they introduce only two concepts that may express the same meaning: one class (engine) and one integer-valued property (power in HP). Indeed, only those classes that cannot be represented by restriction of an existing class by means of property values need to belong to a property-oriented CO. The focus being on primitive concept, and primitive concept understanding being based partially on textual documentation and on reader background knowledge, extensive information model need to be used to describe both textually and formally each primitive concept. CO are multilingual because most concepts are language-independent. But their development is mainly manual. If relationships involved in a CO are formally defined, and if two data sources reference the same CO, semantic integration of these data sources may be done automatically.

Table 1, below, emphasize the main differences between CO and LO. Of course, real ontology model are sometimes in between, and LO may be built on top of CO, a CO defining the primitive concept of a UoD, and a LO representing the language representation of this UoD in some particular language.

Table 1. Typical characteristics of LO and CO

	LO	CO
Token	Word	Concept
Token representation	Word	Model
Ontology Size	Extensive	Minimal
Relation	Formal + linguistic	Algebraic
Content	primitive items + conservative definition	primitive items
Focus	class-oriented	Property-oriented
Development	Semi-automatic	Manual
Ontology usage	Computer-aided	Automatic

3. Concept Ontologies Versus Model

Ontology became a so buzz word that it is often used in place of model. Indeed a conceptual model, e.g., an EXPRESS schema developed in the context of some standard like ISO 10303 (STEP), is an "explicit specification of a conceptualization". The usual definition is not precise enough. So, as noticed by Guarino and Welty ⁷ "today (...ontology) is taken as nearly synonymous of knowledge engineering in AI, conceptual modeling in databases and domain modeling in OO design". But we perfectly know that conceptual modeling, for instance, failed in solving the semantic heterogeneity problem. Thus, it is crucial to clarify the difference between a shared ontology and a model.

An old definition from Minsky ⁸ would introduce the discussion: "To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A". This definition emphasizes the ternary character of a model relationship: it depends on the object (A) and of the observer (B), but it depends also on which questions the observer is interested about A. In other terms, in which context the model was built. In data engineering we are in line with this definition when we teach that a conceptual model shall be built within a precise context. The issue here is that when one designs an application system, the context of the modeling activity is defined by the target system goals and environment. Systems goals and environment are never exactly the same. Thus models are always slightly different. Enough to make model incompatible.

This shows that usual conceptual models can hardly fit several needs. If one wants to build shared ontologies, i. e., ontologies that reflect the information requirements of several application contexts, not only the conceptualization approach must be different from usual modeling activity directed toward a specific target

system, but also the conceptualization formalism must have specific capabilities to allow specification of generic models. These generic models must be either context-independent or at least context explicit to fit the needs of various application contexts.

Importance of context representation for semantic integration of heterogeneous database was already underlined by researchers in multidatabase systems. Kashyap et al.⁹ proposed an explicit representation of the modeling context at the schema definition level: what means, for instance, the "width" property when we try to use it with a "car engine" without knowing in the context of which class and with which precise meaning this property was described.

The property becomes clear when we know that it was defined in a *packaging perspective* for any *material object* as the *width of the virtual box* where it might be packaged.

But even if a property definition is clearly understood, property value may also be context dependant, such context-sensitivity was studied in particular by^{10, 11}. They proposed to represent context at the extensional level, i.e., at the level of data values and object instances: what means for instance the temperature of a particular city if we don't know *when* this temperature was measured, and in which *unit*.

In fact most of the causes of semantics conflicts result from implicit context either in schema definition or in value evaluation. They may be solved if both the *modeling context* and the *value context* are made explicit. Goh¹¹ identified three main causes for semantic heterogeneity:

- (1) *naming conflicts* occur when naming schemes of information differ significantly. A frequent phenomenon is the presence of homonyms and synonyms.

We claim that naming conflicts may be avoided both by replacing the simple word that denote a concept by a complete model that describes it by means of a set of meta attributes, and by modeling explicitly both for entity definition and for property definition the definition context in which the corresponding concept is unambiguous and meaningful. *Driving license id* is unambiguous if it is defined in the context of *French car drivers* classes, it becomes ambiguous (and may have several values) in a context of a *person*.

- (2) *scaling conflicts* occur when different reference systems are used to measure the value of some properties. Examples are different currencies. Scaling conflict may be avoided, either by associating explicitly at the schema level a computer-interpretable representation of the unit that shall be used for any value of a property, or by associating explicitly with each value its own unit.
- (3) *confounding conflicts* occur when information items seem to have the same meaning, but differ in reality, e.g. due to different temporal contexts.

We claim that confounding conflicts may be avoided by investigating whether a value is an intrinsic and permanent property of some instance, or it depends on some evaluation context, and, in the latter case, by associating

this value with its context. For instance the *driving license id* of a person depends on the *country* where the license was obtained, its *weight* depends on the *date* were it was recorded, but its *birth date* is not context dependant (once the scaling conflicts is solved as above).

Moreover, most causes of schematic conflicts, and in particular schema isomorphism conflicts which means that semantically similar entities have a different number of attributes⁹ also result from context sensitivity. It is not so difficult to identify, to describe and to reach consensus on all the major properties which are rigid⁷, i.e. which are *essential*, for each instance of a class. For instance each *customer* has a *birth date*, each *mechanical component* has a *weight*, and each *town* has a (current) *number of inhabitants*. But it is impossible to agree on those rigid properties that should be represented for each class in a database. Thus, ontological description of a class shall describe all its rigid properties (at least within some very broad context common to all data sources intended to be integrated). Then, each schema that references an ontology may select, according to its design context, which ontology-defined properties are pertinent for the problem at hand and are thus represented in the database. For instance, the *weight* or *birth date* of a *person* are seldom used in a customer data base! So, when several schemas refer to a same ontology, the mapping onto this ontology allows to identify automatically which ontology-defined properties are semantically equivalent in several data source, which properties are represented in some data source without being represented in some other, and possibly, which properties if any are not defined in the common ontology.

This discussion allows to define the requirements for data-integration-oriented ontologies. To provide for automatic integration of several data sources, a conceptual ontology must explicitly represent:

- (*definition context explication*) at the schema level the modeling context in which each class or property is defined,
- (*exhaustive class description*) at the schema level for a class all its rigid properties, at least in some very broad context common to all the target data sources,
- (*value context explication*) at the value level, the local context in which each value is evaluated, and
- (*value scaling explication*) either a the schema level or at the value level, the unit of any physical quantity.

We present in the next section how these requirement are fulfilled in the PLIB ontology model.

4. PLIB: A Context-Explication Ontology For Data Integration

Initiated in the early 90's the goal of the PLIB project was to develop an approach and standard models for exchanging and integrating automatically engineering component database¹². To allow such an automatic integration, an ontology-based approach has been developed. An ontology model (known as the PLIB dictionary

model) has been defined^{13 14} and each PLIB-based data source was supposed to contain at least : (1) an *ontology*, (2) a *schema*, and (3) *component data* represented according to the schema that references the ontology. Because we cannot assume that complete shared ontologies will ever exist, each database must have its own local ontology. But, to make automatic integration feasible, each particular local ontology may also contain (4) a *mapping* onto pre-existing shared ontology(ies) (e.g. standard ontologies) through semantic relationships. In particular, subsumption relationships allow a local ontology to reference a shared ontology and to import properties (see 5.2) without needing to duplicate class or property definitions. Development of standard ontologies is encouraged. A number already exist or are in progress (e.g., IEC 61360-4:1998¹⁶).

The role of a PLIB ontology is twofold. First it is intended to support user query over integrated PLIB-based database. Such queries need to be supported at various levels of abstraction (*a screw, a machine screw, an hexagon machine screw, an ISO 1014-compliant hexagon machine screw*). Second, it provides for automatic integration. PLIB ontology are:

- conceptual: Each entry is a context-explicit concept defined by a number of facets, both formal and informal;
- multilingual: Each entry is associated with a globally unique identifier (GUI); words used in some facets may appear in any number of language;
- formal: A PLIB ontology is an instance of an ontology (meta) model (ISO 13584-25:2004)¹⁴ specified in EXPRESS¹⁵; such a model being computer-interpretable, integrity constraints over ontology definition may be formally checked;
- modular: An ontology may reference another ontology to import properties without duplicating them
- multi-point-of-view: Once defined, concepts may be associated with any number of representations; the point of view corresponding to each representation is also represented in the ontology
- consensual: Consensus on the model has been reached through an international standardization process; consensus on shared ontologies is either reached through standardization (e.g., IEC 61360-4: 1998, ISO CD 13584-511, ISO CD 13399-100, etc.) or through consortia discussion.

We discuss below the main mechanisms used to make context explicit in PLIB ontologies.

4.1. *Global structuring of the definition context and exhaustive class description*

The role of ontologies being to capture the essence of beings, PLIB propose a distinction between:

- those properties that are rigid⁷ for a class, i.e., that are *essential* for any

instance of a class (i.e., that must hold or have a value)

- those properties that may or not hold or exist according to the role in which an entity is involved.

For instance to have a birth date is an essential property for any person: such a birth date may be unknown in some context, but, if it does not exist, the person does not exist. Contrariwise, to have a *salary* is not an *essential* property. It exist only if the person is an *employee* of some organization. It is understandable only in the context of the relationship between the person and its employing organization since a same person may have several employers.

For a mechanical part, to have a mass is a rigid property, to have a price is not. The *price* only exist if the part is sold on the market, and the price depend on the market (wholesaler or retail sale, quantity of order, discounted customer, etc...)

Of course, in a database schema, a *person* may have a *salary*, and a part may have a *price* and a *supplier* but this is based on some implicit context assumption that shall be explicit at the ontological level.

A PLIB ontology consists of three categories of classes.

- *definition classes* (in PLIB jargon, general model classes) capture the *beings* of the area of interest, together with all their rigid properties.
- *representation classes* (functional model class) represent the additional properties that result from a particular role or point of view ¹⁶. A representation class exists only when associated with a definition class. Each instance of a representation class is a view of an instance of a definition class. This relationship is termed is-view-of.
- *View classes* capture the context of (i. e. the point of view corresponding to) each particular representation class: each representation class shall reference a view class as its modeling context.

For instance, the definition class of a person should contain properties such that *birth date*, *gender*, *current name*, *first name*, etc. A *employee* representation class should contain properties like: *date of-first employment*, *status*, *salary*, ... etc. An *employment status* view class allows to define the context of the representation class. It may also contains for instance the *date of recording*, and the *employer id* attribute.

The definition class of a particular subclass of *mechanical* part, e.g., *screw* should contain properties like *threaded length*, *total length*, *threaded diameter*, *material*, etc. The screw procurement representation class should contain properties such that *price*, *quantity of order*, etc. The market view class specifies the context of the screw procurement. It contain property such that *date*, *kind of market* (e.g.: wholesale, retail sale, negotiated), *supplier*, etc.

4.2. *Explication of the local definition context*

As noted in ⁹ a property cannot be understood if we don't know in which context it was defined. The same entity name may be used with quite different meaning in different context. For instance what means *average total duration* for a travel by airplane from Paris to Lyon: is it the total travel time including access from Paris to Roissy airport, check-in, fly travel from Roissy to Satolas airport and shuttle travel from Satolas to Lyon (about 3 hours) or the take off-landing fly duration (about 45 minutes)?

To represent the definition context of classes and properties, the basic ideas the PLIB ontology model, is that:

- a property cannot be defined without defining, in the mean time, its field of application by means of the class where it is meaningful; this class constitutes its definitions context;
- a class cannot be defined without defining, in the mean time, the properties that are essential for its instances; these properties constitutes the class definition context.

Therefore, a PLIB ontology conforming to ISO 13584-25 ¹⁴ consists of two parts:

- a classification tree where classes and properties are identified and connected;
- a set of meta-attributes that describe successively each class and each property.

A property is identified through a code, a version number and the identification of the class that specifies its domain. It is defined through a number of information elements, possibly translated in various languages, including a definition, a data type, a source document, and possibly a dimensional equation, a unit, a symbol, a formula, etc. A class is identified through a code, a version number and an identification of the source of its definition. It is defined through the properties that are rigid ⁷ to every instance of this class (or of any class defined as a specialization of this class), and through a number of information elements including: definition, superclass, etc. Single inheritance is used and express subsumption, but subsumption may also be expressed by another way (see 5.2).

Back to the travel duration problem. Now, if we know that *average travel duration* is defined in the *composite travel* class, we understand that some mechanism is used to compute an average of the total duration from some average midpoint of Paris to some average midpoint of Lyon. Of course, details on the mechanism shall be described in the *composite travel class* (see figure 1).

We note that it is perfectly feasible that synonymous property name exist in different classes. But the context being different, the meaning is obviously different. For instance another property named *average total duration* might also exist in *fly* class, but the meaning would be quite different from *average total duration* defined in the *composite travel* class. For instance, its definition meta attribute might precise that this duration is defined as the average needed duration from airport arrival

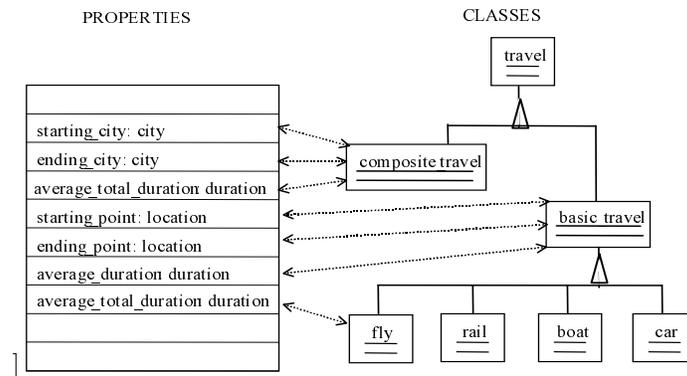


Fig. 1. Joint definition of classes and properties

to airport exit, taking into account that one needs to arrive an airport before fly check-in is closed and that, after landing, one needs to joint the air terminal.

In PLIB, the identifier of the class that constitute the definition context of a property is *part of the identifier of the property*, so the two above properties would be *different* what ever be their names in any language. Note that a standardized class hierarchy, termed *identification hierarchy*¹³ or *reference ontology* is not at all a classification. Its purpose is only to define formally the domain of properties, and to describe any instance by one class belonging and property values.

- PLIB is property-oriented: all what can be described meaningfully by properties is defined by properties. A class shall only be introduced in the identification hierarchy when it constitute the domain of a new property, i.e., the property would be meaningless for the superclass of this new class, but it is meaningful for the new class and all its subclass. Thus definition hierarchies are in general rather flat. For example, the *internal diameter* property is meaningless for a *mechanical component* whatever be its definition. It becomes meaningful if one introduces a new subclass of mechanical components that models *circular bearings*.
- A reference ontology may be referenced by any number of user-defined ontologies that import properties from the reference ontology while reflecting the particular classification used in a particular context (e.g. end-user classification).
- Properties are defined in the context of the higher class where they are meaningful, even if they don't apply to all its subclasses (in PLIB jargon they are said to be *visible*). Then, class definitions specify which properties are applicable, i.e., essential for every instance of this class and constitute its definition context (such properties are said to be *applicable*). Finally, when instances are represented by means of some database schema, only a subset of all the appli-

cable properties may be used to describe them (such properties are said to be *provided*). For any class C the following holds:

$$\text{provided}(C) \subset \text{applicable}(C) \subset \text{visible}(C)$$

This equation shows, at the property level, the difference between ontologies and schemas: several schemas may decide to provide for the same ontology class C various subsets of *applicable* (C). During an integration process, and thanks to the GUI of each ontology concept, it will be obvious which properties are the same and which are not.

4.3. *Explication of the local value context*

In a number of cases, the value of some instance property changes when its evaluation process context change. This means that the range of such properties is not a value set, it is a function set. Let C be the set of all instances of a class, P be a property whose domain includes C , D be the set of all possible values of P , $EVAL$ the set of all the contexts of a given instance where value of property P may be evaluated ;

- a *characteristic property* (characteristic for short) is a property that define a function over C :

$$P : C \rightarrow D$$

- a *context dependent property* is a property whose value in a function of the context:

$$P : C \rightarrow (EVAL \rightarrow D)$$

In PLIB ontologies¹⁴, as suggested in¹⁰, context is represented as a set of property-value pairs (another part of the PLIB standard, ISO 13584-24, provides for representing explicitly the mathematical dependency function, this is not yet introduced in the ontology model). Such properties are termed *context parameters*.

Table 2 shows various examples of characteristics and context dependent properties.

Table 2. Representing value context

Entity	Person	Ball bearing	Plane
characteristic	birth date	inner diameter	plane type
context-dependent property	hair color	life time	cheapest fare
context parameter	date	load, speed	customer age

Of course, the ontology designer may decide to freeze all the context parameter values within a property definition, like: *hair color when birth; life time for 100 Pascal radial load and 6000 t/mn; cheapest fare when 65 years old*. But, if the whole evaluation context is not specified within a property definition, this property shall be represented as a context-dependant property, and the context parameter of

which its value depends shall be explicitly modeled at the ontology level, together with the dependency relationship.

Note that representing instances is a question of schema and not of ontology. As discussed in Sciore et al ¹⁰, all the context-parameter/value pairs that characterize a context dependent property value shall be represented by some means: at the property value level, at the instance level if the same context has been used for all the instance properties, or even at the level of the whole database if properties of all instances were evaluated in the same context. Any way, it shall be available to enable integration.

4.4. Explication of value scaling

In a PLIB ontology, it has been decided that data type and value unit have to be represented at the ontology *property definition* level (a measure data type includes a unit) and not at the *value level* (each value is associated with a unit). Figure 2 below gives an overview of the type system.

```

datatype
  simple_type
    boolean_type
    number_type
    int_type
      int_currency_type      -- integer amount in some explicit currency
      int_measure_type       -- integer physical quantity in some explicit unit
      non_quantitative_int_type -- enumeration type, value identified by an integer
    real_type
      real_currency_type     -- decimal amount in some explicit currency
      real_measure_type      -- decimal physical quantity in some explicit unit
    string_type
      non_quantitative_code_type -- enumeration type, value identified by a code
  complex_type
    level_type              -- decimal value with tolerance ( min, max, nominal, typical)
    class_instance_type     -- composition
    entity_instance_type    -- type defined by an EXPRESS model
      entity_instance_type_for_aggregate
    array_type
    list_type
    set_type
    bag_type
  named_type                -- GUI-identified data type

```

Fig. 2. Property range definition

Each *int_measure_type* and *real_measure_type* shall reference a unit modeled using an EXPRESS model borrowed from ISO 10303-41:2000. This model allow to represent both dimensional exponents for a physical quantity, and all kinds of measure unit: either SI unit (e.g., millimeter), derived (e.g., m/s), or conversion-

based unit (e.g., inch). Note that a PLIB ontology also allows to defined sharable domains of value associated with a GUI (called *named types*).

4.5. *From ontology to schema*

We call *ontology-based data base* (OBDB) a database (1) that explicitly represent an ontology (that may itself include a mapping onto other ontologies), (2) whose schema refers to the ontology for each of its represented entity and property and (3) whose each data may be interpreted in a consistent way using the meaning defined for the corresponding ontology entry. An OBDB is not required to populate either all the classes of its ontology or all the properties defined for a given class. Moreover, provided that the link from data to ontology is preserved, the schema structure is not required to preserve the ontology structure. Inheritance composition and view-of relationship may be "flattened". This means that values representing:

- properties of a definition class instance,
- properties of a part of this instance, and
- properties of a representation class instance that is view-of the definition class instance may appears in the same data base entity instance.

This shows the diversity of the various schemas that may be built just from the same ontology.

5. Formal definition of PLIB ontologies

In this section we present a formal definition of PLIB ontologies. For simplicity we restrict to ontologies that consists of definition classes (no representation class or view class). A PLIB ontology may be defined separately as a single ontology, but it may also be mapped onto one (or several) standard ontologies.

5.1. *Single PLIB ontology*

Formally, a single PLIB ontology may be defined as a 6-tuple : $O = \langle C, P, IsA, PropCont, ClassCont, ValCont \rangle$, where:

- C is the set of classes used to describe the concepts of a given domain;
- P is the set of properties used to describe the instances of C . P is partitioned into P_{val} (characteristics properties), P_{onc} (context dependent properties) and P_{cont} (context parameters). When $p \in P$ is a physical measure, its definition includes its measure unit;
- $IsA : C \rightarrow C$ is a partial function, the semantic of which is subsumption;
- $PropCont : P \rightarrow C$ associates to each property the higher class where it is meaningful;
- $ClassCont : C \rightarrow 2^P$ associates which each class all the properties that are applicable to every instances of this class (rigid properties);

- $ValCont : P_{func} \rightarrow P_{cont}$ associates to each context dependent properties the context parameters of which its value depends.

If we define recursively the visible properties as ^a):

$$visible(c) = PropCont^{-1}(c) \cup visible(IsA(c)),$$

then the following axioms shall hold :

- (1) IsA defines a single hierarchy: the class graph G is a tree. G is defined by:
 $G = (C, (c_1, c_2) | c_2 \in C \wedge c_2 \in Dom(IsA) \wedge c_1 = IsA(c_2))$
- (2) Applicable properties are inherited :
 $ClassCont(c) \supset ClassCont(IsA(c))$
- (3) Context-dependent properties and their context parameters are visible and applicable at the same time:
 $\forall c \in C, p \in P_{func}, p \in ClassCont(c) \Rightarrow ValCont(p) \subset ClassCont(c)$
 $\forall c \in C, p \in P_{func}, p \in visible(c) \Rightarrow ValCont(p) \subset visible(c)$

Moreover, for stand-alone ontologies, one more predicate applies: only meaningful properties may become applicable :

$$(4a) ClassCont(c) - ClassCont(IsA(c)) \subset visible(c)$$

Example 5.1. Figure 3 (a) presents a single ontology. Class hierarchy is represented by indentation. $P = \{mass\}$. The *mass* properties applies to *hardware* and *components*, but not to *software* and *simulation models*. *mass* is visible at the level of *resources* : $PropCont(mass) = resources$, with a definition s. t. "the mass of a resource that is a material object". It becomes applicable in *hardware* and *components*: $ClassCont(hardware) = \{mass\}$; $ClassCont(component) = \{mass\}$

5.2. Mapped PLIB Ontology

PLIB does not assume that all data sources use the same ontology. Each data source may build its local ontology without any external reference. It may also build it based upon one or several reference ontologies (i. e., standard ones). A class of a local ontology may be described as *subsumed* by one or several other class(es) defined in other ontologies. This means that each instance of the former is also instance of the latter. This relationship is named *case-of*. Though case-of relationship the subsumed class may either import properties (their GUI and definitions are preserved) or map properties (the properties are different but they are semantically equivalent) that are defined in the referenced class(es). It may also define additional properties.

A PLIB ontology O_m that includes mapping onto one (or several) other ontologies may be formally defined as a couple: $O_m = \langle O, M \rangle$, where : $O = \langle$

^aTo simplify notation, we extend all functions f by $f(\phi) = \phi$

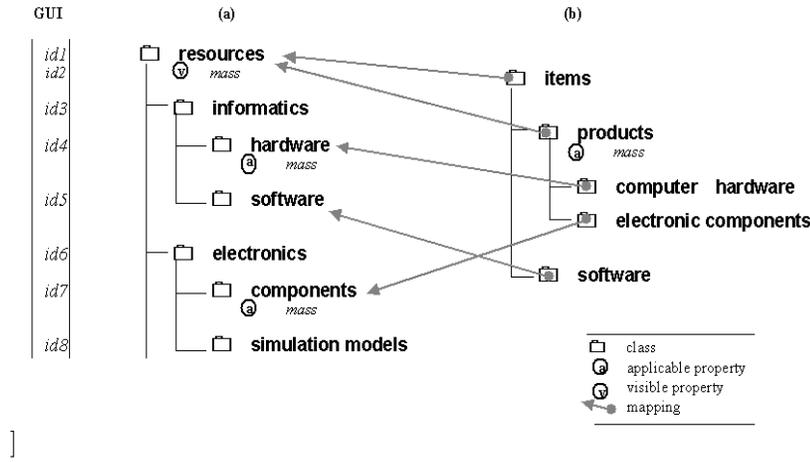


Fig. 3. An example of a reference ontology (a) and of an user defined ontology (b)

$C, P, IsA, PropCont, ClassCont, ValCont >$ is an ontology, and $M = \{m_i\}$, is a mapping defined as a set of mapping objects.

Each mapping object has four attributes : $m = \langle domain, range, import, map \rangle$

- $domain \in C$ defines the class that is mapped onto an external class by a *case-of* relationship;
- $range \in GUI \subset \{string\}$ is the globally unique identifier of the external class onto which the $m.domain$ class is mapped;
- $import \in 2^P$ is a set of properties visible or applicable in the $m.range$ class that are imported in $ClassCont(m.domain)$;
- $map \subset \{(p, id) \mid p \in P \wedge id \in GUI \subset \{string\}\}$ defines the mapping of properties defined in the $m.domain$ class with equivalent properties visible or applicable in the $m.range$ class. The latter are identified by their globally unique identifiers.

Note that each mapping objet defines a subsumption relationship between the $m.domain$ and $m.range$ classes. Note also that when properties are imported, they belong to P .

Example 5.2. Figure 3 (b) present a (user-defined) ontology mapped on a reference ontology (a). $C = \{items, products, computer hardware, electronic components, software\}$ and $P = \{mass\}$. $M = m_1, m_2, m_3, m_4$ with $m_1 = (item, id1, (), ())$; $m_2 = (products, id1, (id2), ())$; $m_3 = (computer hardware, id4, (), ())$; $m_4 = (electronic components, id7, (), ())$. We note that no properties are mapped, they are all imported.

All the axioms for single ontologies hold. The specific property (4a) becomes

(4b and 4c) that state that imported properties belongs to the set of applicable properties of the importing class (and of its subclasses), and that the other new applicable properties of the importing class shall belongs to its visible properties.

(4b) $\forall m \in M, ClassCont(m.domain) \supset m.import,$

(4c) $\forall m \in M,$

$ClassCont(m.domain) - ClassCont(IsA(m.domain)) - m.import \subset visible(c)$

Moreover, mapped properties shall be visible or applicable to the *m.domain* class:

(6) $(\forall m \in M, \forall p \in P, \exists id \in GUI \text{ s. t. } (p, id) \in m.map)$

$\Rightarrow p \in ClassCont(m.domain) \cup visible(m.domain).$

As shown by example 5.2, the structure of a (user) ontology may be quite different from the one of a standard ontology she references. Nevertheless, a system storing the user ontology $\langle O, M \rangle$ may automatically answer queries against the standard ontology(ies) on which *O* is mapped.

6. Using PLIB ontologies

We outline in this section some uses of PLIB ontologies, either for building reference or local ontologies, or for integrating data in various context.

6.1. Building reference ontologies

A number of reference ontologies are already standardized¹⁷ or are in the standardization process either within IEC or within ISO. Several ontology editors have been developed (see: <http://www.plib.ensma.fr/>) for that purpose. Figure 4 shows the ontology editor developed by CNIS, China, for supporting the development process of ISO DIS 13584-511.

Recently the Open and interoperable domain dictionaries initiative (OIDDI) has been launched (<http://www.oiddi.org/>). The goal of this initiative is to promote the emergence of compatible and complementary ontologies that would progressively cover the whole technical and business domain. It is also to ensure that any technical ontology should be usable for any business process. To achieve this goal, all sponsoring organizations commit to use the PLIB ontology model for their technical exchanges. A number of key players of B2B e-commerce, including RosettaNet, ECALS, ECCMA, decided to sponsor this initiative. As a result, all the ontologies and technical dictionaries developed by these organizations should become available as PLIB-ontologies in the next future.

6.2. Integrating heterogeneous and autonomous data sources through ontology articulation

A mapping between a local ontology and another ontology (see 5.2) defines an articulation between these two ontologies that may be used for integration purpose. Such a mapping being part of each local source, it allows an automatic integration of ontology-based data sources in the following sense:

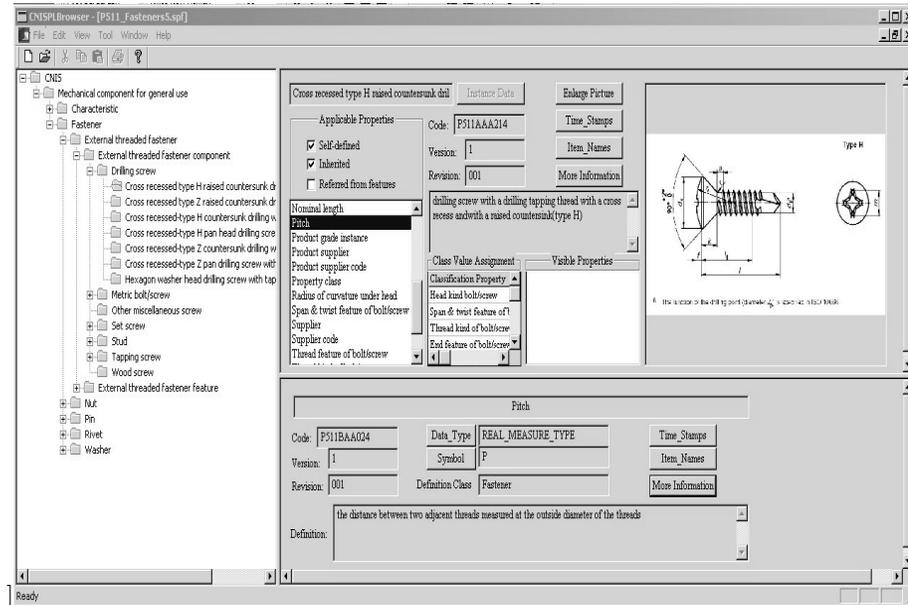


Fig. 4. A PLIB-ontology editor (courtesy of CNIS, China)

- Let's assume that there exist some reference ontology O ;
- Let's also assume that each local source S_i is associated with a local ontology O_i and that each class C_{ij} of S_i that is in the domain modeled by O is mapped by the case-of subsumption relationship, either directly or indirectly (through inheritance within O_i), onto its smallest subsuming class C_j in O (*smallest subsuming class reference requirement: SSCRR*)¹⁸
- Then, each local source, whatever be its local ontology, may answer to queries stated in terms of O .

Note that this automated integration technique leaves a lot of autonomy to each data source. It assumes that each database administrator (DBA) wants to make its data available in terms of a standard domain ontology. Thus, each DBA is required to describe *a priori* an articulation between its own local ontology and the standard ontology by means of subsumption *case-of* relationship (ensuring the SSCRR assumption), and property mapping. This *a priori* approach, different from most existing approaches where ontology mapping is done at integration time¹⁹, seems quite well suit the needs of a number of Web applications, including in particular B2B e-commerce. This approach is discussed in more details in¹⁸.

6.3. *Building engineering component databases and electronic catalogues*

In most engineering fields, products to be designed are essentially assemblies of pre-existing technical objects. In such fields, an important part of the engineering knowledge is the component knowledge. It corresponds to an expertise on the criteria to be used to select a component, on the condition of component usage, on the behavior of components and on the pertinent component representation for each specific discipline ²⁰.

Component knowledge is highly structured. Components are defined at various levels of abstraction (e.g., fasteners/screws/machine screws/ hexagon machine screw; bearing/circular bearing/double ball bearing) where component retrieval process may take place. Engineering properties are defined at each level, that also apply to lower levels. In a database, each component class should be described by its own table with some kind of table inheritance. It is why the relational model is so poorly adapted for managing components. Most conventional so-called article database, based on the relational technology, only contain a fixed number of properties for describing any component. These properties include a long string, (often called "designation"), where engineering properties are all encoded (see figure 5).

'SCREW-ISO1014-L10-D5-GRADa'

Fig. 5. Engineering information encoding in usual item data base

PLIB ontologies allow to make explicit the component engineering knowledge. This knowledge may then be represented within an OBDB *together with component data*. In such databases, all the technical data about components may be explicitly represented. Moreover the data meaning described in the data base as a PLIB ontology allows to provide user friendly interfaces, allowing to display together data and data meaning ²¹. Figure 6 shows an example of a graphic interface that may be automatically generated.

Moreover, an OBDB containing all the component knowledge as it is usually represented in paper catalogues, component catalogues may be automatically generated from OBDB in various formats ²², ²³. Figure 6 is an example of such an electronic catalogue generated in DHTML as an active documents for the Web. This catalogue may be browsed through Internet, it may also be loaded on the user site for use by the company designers without needing to access to Internet, it may also be used to select a component and either to issue an ordering form to the supplier, or to get the data description of the component for insertion within a current design. Note that in this catalogue, each property value that correspond to a context-dependent property is associated with the values of the context parameter of which it depends.

This contextual information is precisely the one that PLIB ontology model requires to explicate. Whatever be the user interface provided by smart internet client in the future, the computer-to-computer protocol used over the semantic Web will need to be structured-data oriented, and based on a context-explicit conceptual ontology like PLIB. Most B2B e-commerce protocol already base their product ontologies on PLIB or PLIB-like ontology models, and a Web service for this application is currently under development.

7. Related Work

Importance of context explication for data integration was identified by several researchers in the field of multidatabase system in the 90's. Kashyap and al.⁹ proposed to represent definition context at the schema level as a set of property-value pairs, but value where only informally defined. Sciore and al. (1992) proposed to represent value context at the value level. A PLIB ontology represents formally both levels. Moreover it offers mechanisms for structuring globally the definition context and for representing units.

Various approaches have been developed for ontology-based integration of information². In the single ontology approach each source is related to the same global domain ontology¹¹. As a result, a new source cannot bring any new or specific concept without requiring change in the global ontology. In the multiple ontologies approach (e.g. Observer²⁴), each source has its own ontology developed without respect of other sources. In this case the inter-ontology mapping is very difficult to define because the different ontologies may use different aggregation and granularity of the ontology concept². To overcome the drawback of single or multiple ontology approaches, several researches have proposed an hybrid approach where each source has its own ontology, but where all ontologies are connected by some means to a common shared vocabulary. PLIB-based integration follows the hybrid approach and propose a formal model for ontologies and ontology mappings. Unlike BUSTER²⁵ PLIB ontology-based approach does not assume that local ontologies are only restrictions of the global ontology: each source may add whatever property or class. To give autonomy to the local source, we use the same kind of ontology articulation as ONION¹⁹, but, unlike ONION, we suppose that articulation between local and shared ontology is done *a priori* by the local DBA administrator. As a result our approach scales to any number of data sources.

A lot of research is taking place currently on ontologies for the semantic Web (e.g. DAML+OIL, OWL,²⁶). The main focus of these ontology languages is semantic annotations of Web resources using terms. As a result, they use a great variety of mechanisms for defining terms relationships, whether the terms represent classes, properties or string values: equivalence, hyponym and subsomption, class expression, class restrictions, etc. Thus they are able to process very large vocabularies and to make inference for retrieving the various annotations that correspond to the same query. On the other hand, their capabilities for modeling numeric properties seems

rather limited. No capability to associate a numeric property with dimensional exponent (it is a length and not a mass) and a unit. No capability to associate a property value with context parameter values that define its value context. No reference to any numeric operator, neither in the Web ontology languages, nor in their draft query languages: OWL-QL²⁷ or DQL²⁸. Contrariwise, PLIB-structured ontologies are numeric property-oriented and context-explicit. Classes and properties are both defined by a number of information elements, including in particular, for measure properties, their unit. PLIB-based databases contain a number of numeric property values, and the draft query language for OBDB, CQL²⁹ largely uses numeric operators and numeric queries. Moreover, PLIB ontology model is the first model we know that explicitly represents ontology mapping within a local ontology as a first class citizen as suggested by model management vision.⁴

8. Conclusion

The concept of ontology was mainly studied in computer science since early 90's. Its intent is to capture the essential nature of things through class structures and properties. In a number of computer disciplines, it appears like some kind of philosopher's stone and a lot of understandings, models and approaches were developed. Not surprisingly, differences in approaches reflect differences in the addressed problems, and it is currently not clear which approach may be used for a particular problem.

In this paper, we have investigated the use of ontology in a structured data integration perspective. First we have proposed a taxonomy of ontologies into linguistic ontologies (LO) and conceptual ontology (CO). LO represent words and words relationships. They are document-oriented. They provide for intelligent structuring, modeling and querying set of documents, and in particular those available on the Web. CO represent concepts, as they are manipulated in the structured data universe like data base or engineering, and concept properties. They provide for integrating automatically data by means of shared models of concept meanings. COs sometimes appear as some kind of conceptual models. We have shown that COs shall be context-explicit when conceptual model are in general highly contextual. We have defined four requirements to ensure that the definitions within an ontology are not context-sensitive and may thus be used to support data integration:

- *definition context* explication for all classes and properties;
- *exhaustive class description* in terms of applicable properties;
- *value context explication* for each property value;
- *value scaling explication* for each physical measure value.

Then we have described how the above mechanisms are represented in as PLIB ontologies:

- definition context explication is done by associating with each properties, the class where it is meaningful and with each class the properties applicable to

each class instance

- *exhaustive class description* is done by ensuring that applicable properties of a class consist of all *those properties* that are essential (rigid) for its instances,
- *value context explication* is done by associating with each property value its evaluation context represented as a set of property-value pairs, and
- *value scaling explication* is done at the schema level by associating each quantitative property type both with a dimensional equation and with a unit.

Moreover the PLIB ontology model provides two mechanisms for modularity allowing (1) to separate concept definitions and context-specific concept representations (is-view-of) and (2) to explicitly map local ontologies onto shared ontologies to allow the use of model management techniques, and to provide for automatic integration of heterogeneous and autonomous data sources.

Finally we have outlined how such ontologies may be used for database integration, development of engineering component databases, generation of electronic catalogues of industrial components and for the semantic Web.

References

1. A. Elmagarmid and M. Rusinkiewicz, *Heterogeneous Autonomous Database Systems*, (Morgan Kaufmann Publishers, Inc. San Francisco, California, 1999).
2. H. Wache, T. Vgele, U. Visser, H. Stuckenschmidt, Ontology-Based Integration of Information: A Survey of Existing Approaches, in *Proc. of the IJCAI-01 Workshop: Ontologies and Information Sharing*, Seattle, WA, (2001) pp. 108-117
3. G. Wiederhold, Mediators in the architecture of future information systems. *IEEE Comput.* 25(3) (1992) pp. 38-49.
4. P. A. Bernstein, A. Y. Havely and R. A. Pottinger, A vision of management of complex models, *SIGMOD Record* 29(4) (2000) pp. 55-63.
5. T. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in *Conceptual Analysis and Knowledge Representation*, eds. N. Guarino and R. Poli, (Kluwer Academic Publishers, 1993).
6. J. O. Everett, D. G. Bobrow, R. Stolle, R. Crouch, V. De Paiva, C. Condoravdi, M. van den Berg and L. Polanyi, Making Ontologies Work for Resolving Redundancies Across Documents, *Comm. ACM*, 45(2) (2002) pp. 55-60.
7. N. Guarino, C. Welty, Evaluating ontological decision with Ontoclean, *Comm. ACM*, 45(2) (2002) pp. 61-65.
8. M. Minsky, Matter, Mind and Models, in *Proc. of the International Federation of Information Processing Congress*, vol. 1, (1965) pp. 45-49.
9. V. Kashyap and A. Sheth, Semantic and schematic similarities between database objects: a context-based approach, *The VLDB Journal*, 5 (1996) pp. 276-304.
10. E. Sciore, M. Siegel, A. Rosenthal, Using Semantic values to Facilitate Interoperability Among Heterogeneous Information Systems, *ACM Transactions on Database Systems*, 19(2) (1994) pp. 254-290.
11. C.H. Goh, S. Bressan, S. Madnick, M. Siegel, Context Interchange: New Features and Formalisms for the Intelligent Integration of Information, *ACM Transactions on Information Systems* 17(3) (1999) pp. 270-293.
12. G. Pierra, An object oriented approach to ensure portability of cad standard parts

- libraries. in *Proc. of Eurographics '90 conference*, (Elsevier, North-Holland, 1990) pp. 205-214.
13. ISO 13584-42. 1998. *Industrial Automation Systems and Intregation - Parts Library - Part 42: Description methodology: Methodology for Structuring Parts Families*, (ISO, Geneva, 1998)
 14. ISO 13584-25. 2004. *Industrial Automation Systems and Intregation - Parts Library - Part 25: Logical model of supplier library with aggregate values and explicit content*, (ISO, Geneva, 2004)
 15. D. Schenck and P. Wilson, *Information Modelling: The EXPRESS Way* (Oxford University Press, Oxford 1994)
 16. G. Pierra, A Multiple Perspective Object Oriented Model for Engineering Design, in *New Advances in Computer Aided Design Computer Graphics*, ed. X. Zhang, (International Academic Publishers, Beijing, 1993) pp. 368-373.
 17. IEC 61360-4. 1998. *Standard data element types with associated classification scheme for electric components - Part 4: IEC reference collection of standard data element types, component classes and terms*, (IEC Geneva 1998)
 18. L. Bellatreche, G. Pierra, D. Nguyen Xuan and H. Dehainsala, An Automated Information Integration Technique using an Ontology-based Database Approach. in *Proc. of 10th ISPE International Conf. on Concurrent Engineering: Research and Applications (CE'03) : Special Track on Data Integration in Engineering*, eds. R. Jardim-Gonsalves, J. Cha and A. Steimer-Garo (Balkena, Lisse, 2003)
 19. P. Mitra, G. Wiederhold and M. Kersten, A Graph-Oriented Model for Articulation of Ontology Interdependencies, in *Proc. of the Conference on Extending Database Technology (EDBT)*, (2000)
 20. P. Paasilia, A. Aatonen and A. Riitahuta, Automatic Component Selection, in *Proc. of the CIME conference 1993*, eds. C. Kooi, P.A. MacConaill and J. Bastos (IOS Press 1993)
 21. Dehainsala, H. Base de Donnes Base Ontologique, (to appear in *Proc. of INFOR-SID'2004, Forum Jeunes Chercheurs*)
 22. E. Sardet, G. Pierra, H. Murayama, Y. Oodake and Y. Ait-Ameur, Simplified Representation of Parts Library: Model Practice and Implementation, in *Proc. of PDT Days 2001* (QMS edition, Brussels 2001)
 23. G. Pierra, J.C. Potier and E. Sardet, From digital libraries to electronic catalogues for engineering and manufacturing, *International Journal of Computer Applications in Technology (IJCAT)*, vol. 18, (2003) pp. 27-42
 24. E. Mena, V. Kashyap, A. Illarramendi and A. Sheth, Managing multiple information sources through ontologies: relationship between vocabulary heterogeneity and loss of information. in *Proc of the workshop on Knowledge Representation meets Databases in conjunction with European Conference on Artificial Intelligence* (1996.)
 25. H. Stuckenschmidt, T. Vgele, U. Visser, and R. Meyer, Intelligent Brokering of Environmental Information with the BUSTER system. in *Proc. of the 5th International Conference 'Wirtschaftsinformatik'*, Ulm, Germany, (2001) pp.15-20
 26. J. Broekstra, M. Klein, S. Decker, D. Fensel, F. Van Harmelen, I. Horrocks, Enabling knowledge representation on the Web by extending RDF Schema, *Computer networks*, 39 (2002) pp. 609-634.
 27. R. Fikes, P. Hayes and I. Horrocks, OWL-QL - A Language for Deductive Query Answering on the Semantic Web, *Stanford University KSL Technical Report* 03-14 (2003).
 28. R. Fikes, P. Hayes and I. Horrocks (editors); DAML Query Language (DQL) Abstract Specification, *Joint United States/European Union ad hoc Agent Markup Lan-*

guage Committee; DARPA Agent Markup Language (DAML) Program; April 1, 2003; <http://www.daml.org/2003/04/dql/>.

29. H.Murayama , O.Lemarchand, Y.Mizoguchi. Modeling Product Ontology with CQL, in *Proc. of 10th ISPE International Conf. on Concurrent Engineering: Research and Applications (CE'03) : Special Track on Data Integration in Engineering*, eds. R. Jardim-Gonsalves, J. Cha and A. Steimer-Garçao (Balkena, Lisse, 2003)