Proc. of Concurrent Engineering (CE'2003), Special track Data Integration in Engineering, Madeira, Portugal, 26-30/07/2003, 243-254

Context-Explication in Conceptual Ontologies: The PLIB Approach

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

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 precisely developed to allow neutral exchange and automatic integration of technical data. We first propose a taxonomy of ontologies into linguistic ontologies, based on words and usable for intelligent document processing, and concept ontologies, multilingual and usable with structured data. We then discuss differences between ontologies and usual conceptual models. We claim that the main difference is context-sensitivity, and we identify four requirements for making ontologies less contextual than models and suitable for data integration. Finally we present how these requirements have been fulfilled in the PLIB ontology model developed to give meaning to technical data, and we outline the use of PLIB-based ontologies in various domains including database integration, e-engineering and the semantic Web.

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 (Elmagarmid et al. 1999). Distributed architecture models have been developed, where mediators (Wiederhold 1992) provide uniform access to heterogeneous 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 techniques (Bernstein 2003), 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 (semantic equivalence, Kashyap & Sheth 1996)
- 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 answer elements 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 that has been 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 component libraries.

The content of this paper is as follows. In the next section we discuss the various kinds of ontology needed for representing semantics. We propose to 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 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. We discuss in section 5 how such ontologies may be used for database integration, e-engineering and the semantic web. A discussion of related works is presented in section 6. Conclusion is presented in section 7.

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 a formal explicit specification of a shared conceptualization" (Gruber 1993). 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" (Gruber 1993),
- *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 we consider as the major criteria for classifying ontologies and ontology models: whether their area of interest consists of *beings* –what there is 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. We claim that these two kinds of ontologies address quite different problems and should have quite different content.

LO (e.g., Everett et al. 2002) 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 world (Gruber 1993). 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 way (Walche et al. 2001) 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 ontology in Gruber (1995), 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 when 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. CO may also be property-oriented to reduce again the number of concepts that need to be represented. Indeed, only those classes that cannot be represented by restriction of a class trough 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.

	LO	CO
Token	Word	Concept
Token representation	Word	Model
Ontology size	Extensive	Minimal
Relation	Formal + linguistic	Identity/subsomption/ conversion
Content	Primitive items + conservative definition	Primitive items
Focus	Class-oriented	Property-oriented
Development	Semi-automatic	Manual
Ontology usage	Computer-aided	Automatic

Table 1 – Typical characteristics of LO and CO

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 a "formal explicit specification of a shared conceptualization". The usual definition is not precise enough. So, according to (Guarino & Welty 2002) "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 an ontology and a model.

An old definition from Minsky (1965) 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 also *which questions the modeler is interested* about UoD objets. 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 two modelers never or seldom wonder exactly the same questions about domain objects. Thus models are always slightly different. Enough to make model incompatible.

We claim that the major difference between ontologies and models is that the latter are context dependant, when the former should be either context-free or context-explicit.

Importance of context representation for semantic integration of heterogeneous database was already underlined by researchers in multidatabase systems.

Kashyap & Steh (1996) 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 its virtual box* where it might be packaged.

But even if a property definition is clearly understood, property value may be context dependant, such context-sensitivity was studied in particular by Sciore et al. (1992), Goh et al. (1999), Bressan et al. (1999). 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. Goh (1997) identifies 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 claims 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 attributes (Kashyap & Steh 1996) 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 *rigid* (Guarino & Welty 2002), 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 source 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.

Requirements for data-integration-oriented ontologies

Thus, 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 possible 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 late 80's the goal of the PLIB project was to develop an approach and standard models for exchanging and integrating automatically engineering component database (Pierra 1990). 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 (ISO 13584-42 1998, ISO 13584-24: 2003) and each database contains both an ontology, and component data represented according to a schema that references the ontology.

The role of a PLIB ontology is twofold. First, it provides for automatic integration. Second it is intended to support user query over the integrated 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)*

Because we cannot assume that complete shared ontologies will ever exist, each database shall have its own ontology. But to promote the use of standard ontologies, each particular ontology may reference pre-existing ontologies (e. g. standard ontologies) through a subsomption relationship (see 5.1) 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 61360-4:1998, ISO 13584-511 currently under ballot).

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 specified in EXPRESS (Schenk et al. 1994); such a model being computer-interpretable, integrity constraints over ontology definition may be formally checked;
- modular: An ontology may reference another ontology to import class and/or 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[1] that are rigid (Guarino & Welty 2002) 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 ontologial 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 (Pierra 1993). 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*, *sex*, *current name*, *first name*, etc. A *salaried* 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 Kashyap & Sheth (1996) a property cannot be understood if we don't know in which

[1] We use the term property for an attribute whose range may be either boolean or any other data type

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 13584-42 (1998) consists of two parts:

– a classification tree where component families and technical properties are identified and connected;
– a set of meta-attributes that describe successively each component family 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, possibly a dimensional equation and, a unit, a source document, a symbol, a formula, etc. A component family 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) to every instance of this family (or of any family defined as a specialization of this family), and through a number of information elements including: definition, superclass, , etc. Single inheritance is used, but subsomption may also be expressed by another way (see 5.1).

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).

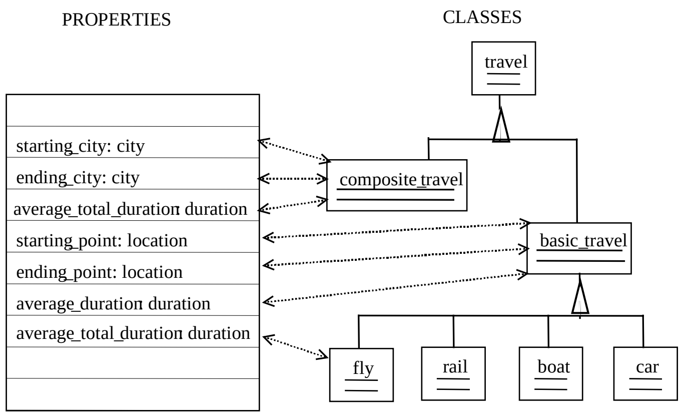


Fig. 1 – Joint definition of classes and properties

We note that at 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 to airport exit, taking into account that you need to arrive an airport before fly check-in is closed and that, after landing, you need 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 the class hierarchy, termed *identification hierarchy* (ISO13584-42:1998) is not at all a classification. Its purpose is only to define formally the domain of properties, and to describe any instance by class belonging and property values.

- PLIB is property-oriented: a class shall only be introduced in the identification hierarchy when it constitute the domain of a new property, i.e., the property is meaningless above and is meaning full in the class and all its subclass; thus definition hierarchy are in general rather flat.
- An identification hierarchy may be referenced by any number of different classification hierarchies that import properties from the identification hierarchy 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 is meaningful. (In PLIB jargon it is said to be *visible*). Class definition specify which properties are rigid, i.e., *essential* for every instance of this class (such properties are said to be *applicable*). Finally, when instances are represented by means of some DB schema, only a subset of all the applicable 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 ontology, as suggested in Sciore et al. (1992), context is represented as a set of property-value pairs. Such properties are termed *context parameters*.

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

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

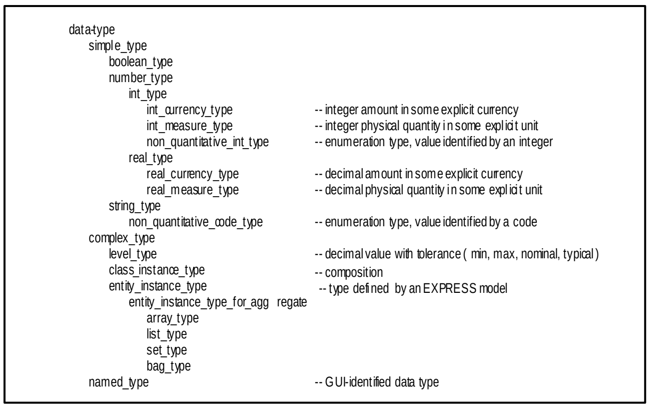
Table 2 – Representing value context

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 10P radial load and 6000 tpm, cheapest fare when 65 years old*. But, if all the evaluation context is not specified within a property definition, this property shall be represented as context-dependant property, and the context parameter of which its value depend shall be explicitly modeled at the ontology level, together with the dependency relationship.

Note that representing instances is a question of model and not of ontology. As discussed in Sciore et al. (1992), 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 data base 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 include a unit) and not at the *value level* (each value is associated with a unit). Figure below gives an overview of the type system.



```
data-type
    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_agg  regate
                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 kind of measure unit: either SI unit (e.g., millimeter), derived (e.g., m/s^2), or conversion-based unit (e.g., inch).

Note that a PLIB ontology also allows to defined sharable domain of value associated with a GUI.

4.5 *From ontology to schema*

We call *ontology-based data base* (OBDB) a database whose schema refers to an ontology for each of its represented entity and property and 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, provide 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 USING PLIB ONTOLOGIES

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

5.1 *Building local ontologies via subsomption relation*

PLIB does not assume that all data sources use the same ontology. Each data source may built its own ontology without any external reference. It may also built it based upon one or several existing ontologies, for instance standard ontologies (recommended practice).

A class 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 *is-case-of*. Though is-case-of relationship the subsumed class may import all or any of the properties that are defined in the referenced class(es). It may also define additional properties. Referenced classes and properties are identified by their GUI (see 4), thus no definition has to be duplicated. In a latter stage, when instances referencing the local ontology will be represented in some ontology-based data source, it will be possible

- to query the local source in term of the referenced ontology (mediator-wrapper approach, Wiederhold 1992)
- to represent local instances as instances of the subsuming classes if values of specific properties are not required (data warehouse approach).

This automated information technique based on PLIB ontologies is beyond the scope of this paper and it is presented in Bellatreche et alt (2003)

5.2 *Building engineering ontologies and engineering component databases*

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 (Paasilia et al. 1993).

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. Properties are defined at each level, that also apply to lower levels. In a database, each component family 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 3).

`'SCREW-ISO1014-L10-D5-GRADa'`

Figure 3 Engineering information encoding in usual item data base

PLIB ontologies allow to make explicit component engineering knowledge. Component data may then be represented:

- in OBDB, that may be automatically integrated or federated,
- in EXPRESS-based exchange format, known as “STEP physical file” (ISO 10303-21:1994),
- in XML (Sardet et al. 2001),
- as active documents for the Web (Pierra et al. 2003).

5.3 *Adding meaning to the Web: the semantic Web*

Developing the semantic web includes two kinds of tasks:

- developing smart document search engines, based on lexical ontologies, and
- developing Web services.

“Web services” means the capability to ask its own internet client questions such that:

- what is the temperature in Panama?
- Who could provide needle bearings with internal diameter of 5 mm? Characteristics and price?

If we want computers to “understand” such questions, in order for all the computers involved in a dialogue protocol to answer the same way, a lot of implicit contextual information need to be made explicit:

- which temperature is asked? Atmosphere? Water? Minimum? Maximum? Average? In which measure unit? When?
- Panama is it a town? A canal? A hat? A cocktail?

All these questions are precisely those that may be easily answered by referencing a PLIB ontology.

Whatever be the user interface provided by smart internet client of 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.

6 RELATED WORK

Importance of context for data integration was identified by several researchers in the field of multidatabase system in the 90’s. Kashyap & Shelth (1996), proposed to represent definition context at the schema level as a set of property-value pair, 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 information (Wache et al 2001). In the single ontology approach each source is related to the same global domain ontology. 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. Observer, Mena et al. 1996), each source has its own ontology developed without respect of other sources. In this case the inter-ontology mapping is very difficult to define as the different ontologies may use different aggregation and granularity of the ontology concept (Wache et al. 2001). 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. Unlike BUSTER (Stuckenschmidt 2000) we do not assume that local ontologies are only restrictions of the global ontology: each source may add whatever property or class. Like COIN (Goh et al. 1999) we represent contextual information of values but we also represent context of ontology definition and we don’t assume that a single ontology is shared.

On going research on ontologies for the semantic Web (e. g. DAML+OIL, OWL, Broekstra et al. 2002) mainly focus on class and classification. They allow to define subsomption relationships between class expressions (e. g., C1 *and* C2 *and not* C3) and use description logics for classifying concepts and individuals (Wache et al. 2001). Such ontologies are largely used for defining domain LO, but they don't provide any mechanism for context-explication which is crucial for semantic data integration and data-intensive semantic Web services. PLIB-structured ontologies are property-oriented and context-explicit. They allow to specify computer-sensible CO that reflect the structure of technical knowledge.

7 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 or knowledge models. We claim that the major difference between CO and conceptual models is contextually. COs shall be context-explicit when conceptual model are always 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 descript*ion 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 provide two mechanisms for modularity allowing (1) to separate concept definitions and context-specific concept representations (is-view-of) and (2) to reference an ontology from another one with class subsomption and property importation.

Finally we have outlined how such ontologies may be used for database integration, for e-engineering and for the semantic Web. These capabilities are more deeply discussed in another paper.

8 BIBLIOGRAPHY

Berbers-Lee, T. & Hendler J. & Lassila O. 2001. The semantic Web. *Scientific American*. Mai 2001: 36-43

Bellatreche L. & Pierra G. & Nguyen Xuan D. & Dehainsala H. 2003. An Automated Information Integration Technique using an Ontology-based Database Approach. *(This Conference).*

Bernstein, P.A. 2003. Applying model management to classical meta data problems, *Proceedings of the 2003 CIDR Conference*, .

Broekstra, J. & Klein, M. & Decker, S & Fensel, D. & Van Harmelen, F. & Horrocks, I. 2002. Enabling knowledge representation on the Web by extending RDF Schema, *Computer networks*, Elsevier, 39: 609-634.

Goh, C.H. & Bressan, S. & Madnick, S. & Siegel, M. 1999. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems* 17(3): 270-293.

Guarino, N. & Welty, C. Evaluating ontological decision with Ontoclean, *Comm. ACM*, 45(2): 61-65.

Gruber, T. 1993. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2): 199-220.

Gruber, T. 1995. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *Formal Ontology in Conceptual Analysis and Knowledge Representation, Guarino N. and Poli R., Eds*, Kluwer Academic Publishers.

Everett, J. O. & Bobrow, D. G. & Stolle, R. & Crouch, R. & De Paiva, V. & Condoravdi, C. & van den Berg, M. & Polanyi L. , Making Ontologies Work for Resolving Redundancies Across Documents, *Comm. ACM, 45(2)*: *55-60.*

Elmagarmid, A. & Rusinkiewicz, M. 1999. *Heterogeneous Autonomous Database Systems*. Morgan Kaufmann Publishers, Inc. San Francisco, California.

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

ISO 10303-41. 2000 *Industrial automation systems and integration — Product data representation and exchange – Part 41: Integrated generic resource: Fundamentals of product description and support*

ISO 13584-42. 1998. *Industrial Automation Systems and Intregration – Parts Library – Part 42: Description methodology: Methodology for Structuring Parts Families*, ISO, Geneva.

Kashyap, V. & Sheth, A. 1996. Semantic and schematic similarities between database objects: a context-based approach,*The VLDB Journal*, Springer-Verlag, 5: 276-304.

Mena E. & Kashyap V. & Illarramendi A. & Sheth A. 1996. Managing multiple information sources through ontologies: relationship between vocabulary heterogeneity and loss of information. In *Proceedings of the workshop on Knowledge Representation meets Databases in conjunction with European Conference on Artificial Intelligence*.

Minsky, M. 1965. Matter, Mind and Models P*roceedings of the International Federation of Information Processing Congress 1965*, vol. 1, pp. 45-49

Paasilia, P. & Aatonen, A. & Riitahuta, A. 1993. Automatic Component Selection. *In Kooij, C., MacConaill, P.A. and Bastos J. (eds)*: IOS Press.

Pierra, G. & Potier, J.C. & Sardet, E. 2003 From digital libraries to electronic catalogues for engineering and manufacturing, *International Journal of Computer Applications in Technology (IJCAT)*, vol. 18, pp. 27-42, 2003

Pierra, G. 1994. Modelling classes of pre-existing components in a CIM perspective: the ISO13584/ENV 400014 Approach, *Revue internationale de CFAO et d'Infographie* 9(3): 435-454.

Pierra, G. 1993. A Multiple Perspective Object Oriented Model for Engineering Design , *in: New Advances in Computer Aided Design & Computer Graphics, X. Zhang, Ed.*, International Academic Publishers, Beijing, China: 368-373.

Pierra, G. 1990. An object oriented approach to ensure portability of cad standard parts libraries. *Eurographics '90*. Elsevier: 205-214.

Sardet, E. & Pierra, G. & Murayama H. & Oodake, Y. & Ait-Ameur, Y. 2001 Simplified Representation of Parts Library: Model Practice and Implementation,In *proceedings of. PDT Days 2001*, QMS edition, Brussels

Schenck, D. & Wilson, P. 1994. *Information Modelling The EXPRESS Way*, Oxford University Press,.

Sciore, E. & Siegel, M. & Androsenthal, A. 1992. Context interchange using meta-attributes. *Proceedings of the 1st International Conference on Information and Knowledge Management* (CIKM-92, Baltimore, MD, Nov.), Y.Yesha, Ed.: 377–386.

Sciore, E. & Siegel, M. & Rosenthal, A. 1994. Using Semantic values to Facilitate Interoperability Among Heterogeneous Information Systems, *ACM Transactions on Database Systems*, 19(2): 254-290.

Stuckenschmidt, H. & Vögele, T. & Visser, U., & Meyer, R. 2001. Intelligent Brokering of Environmental Information with the BUSTER system. In: *Proceedings of the 5th International Conference 'Wirtschaftsinformatik'*, Ulm, Germany, pp.15-20

Wache, H & Vögele, T & Visser, U & Stuckenschmidt, H. 2001. Ontology-Based Integration of Information: A Survey of Existing Approaches, *Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing*, Seattle, WA, pp.108-117

Wiederhold, G. 1992. Mediators in the architecture of future information systems. *IEEE Comput.* 25(3): 38–49.