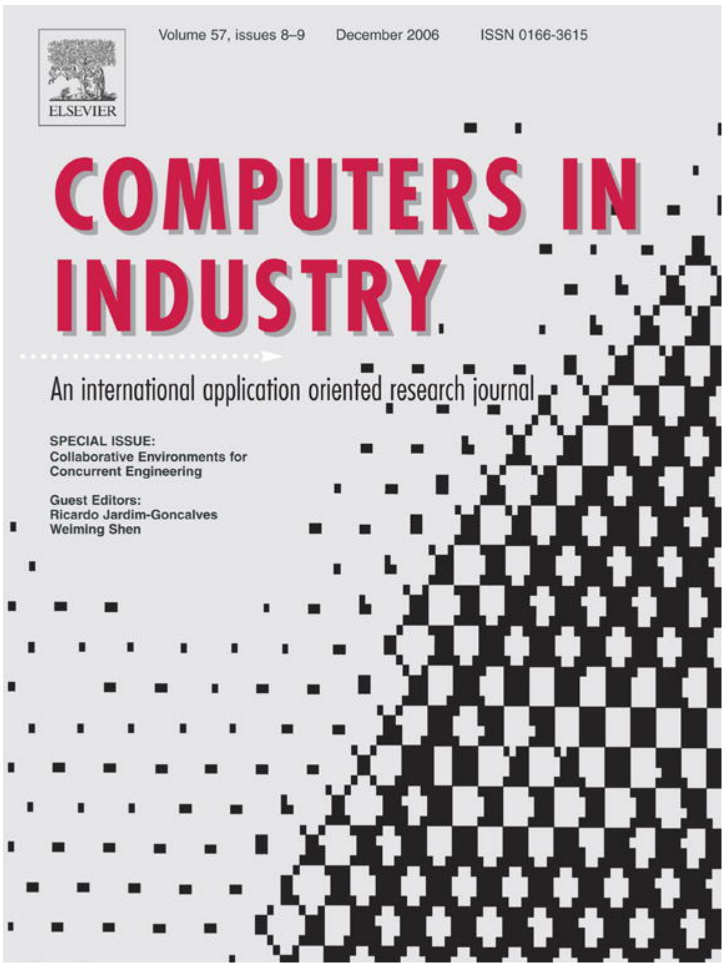

Volume 57, issues 8–9
December 2006
ISSN 0166-3615
ELSEVIER
COMPUTERS IN INDUSTRY
An international application oriented research journal
SPECIAL ISSUE: Collaborative Environments for Concurrent Engineering
Guest Editors: Ricardo Jardim-Goncalves Weiming Shen

Available online at www.sciencedirect.com
ScienceDirect
Computers in Industry 57 (2006) 711–724
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases

Ladjel Bellatreche *, Nguyen Xuan Dung, Guy Pierra, Dehainsala Hondjack

Laboratory of Applied Computer Science (LISI), National Engineering School for Mechanics and Aerotechnics (ENSMA), Poitiers 86960 Futuroscope Cedex, France

Accepted 27 April 2006
Available online 7 July 2006

Abstract

Developing intelligent systems to integrate numerous, autonomous and heterogeneous data sources in order to give end users an uniform query interface is a great challenging issue. The process of constructing a global schema of the integrated system is usually done manually. This is due to the presence of semantic and schematic heterogeneities among schemas of sources. In most cases, sources do not contain enough knowledge to help in solving these heterogeneities and then generating the global schema. In this paper, we present an ontology-driven integration approach called *a priori approach*. Its originality is that each data source participating in the integration process contains an ontology that defines the meaning of its own data. This approach ensures the automation of the integration process when *all sources reference a shared ontology, and possibly extend it by adding their own concept specializations*. We present two integration algorithms where (1) the shared ontology may be extended during the integration process, and (2) the instances of local sources are projected onto the shared ontology. Finally, we show that this theory allows to integrate automatically electronic catalogues into corporate engineering databases using the PLIB ontology model.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Database integration; Ontology-based modeling; Electronic catalogues; Engineering database; PLIB

1. Introduction

Product data are collections of all database objects and document files describing an industrial product (e.g., CAD drawings, 3D models, fluid dynamic simulations) and its engineering and manufacturing process. These data are produced by many different heterogeneous engineering applications. In order to prevent errors resulting from tedious manual or file-based data transfers between engineering systems, to ensure data consistency across different tools and to provide a global view of all project-related data, an integration of different engineering systems is required [27]. This integration will facilitate data sharing. Data sharing systems are crucial for supporting a wide range of applications, such as scientific collaborations, data management on the Web and cooperation between government agencies. Nowadays integrating heterogeneous and autonomous data sources represents a significant challenge to the database and AI communities. Availability of numerous sources increases the requirements for developing tools and techniques to integrate these sources. By autonomy is meant the ability of a database system to choose its own design with respect to any matter, including the data being managed, the representation (data model, query language) and the naming of the data elements [28]. Data integration is the process by which several autonomous, distributed and heterogeneous databases sources (where each source is associated with a local schema) are integrated into a single data source associated with a global schema. It recently received a great attention due to many data management applications: examples are peer-to-peer data [1], data warehouse [2], e-commerce [19], and web services [25].

Formally, a data integration system is a triple I: $\langle G, S, M\rangle$, where G is the global schema (over an alphabet A_G) which provides a reconciled and an integrated schema, S is a set of source schemas (over an alphabet A_S) which describes the structure of sources participating in the integration process, and M is the mapping between G and S which establishes the connection between the elements of the global schema and

* Corresponding author. Tel.: +33 5 49 49 80 72; fax: +33 5 49 49 80 64.
E-mail address: bellatreche@ensma.fr (L. Bellatreche).

0166-3615/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2006.04.015

those of the sources. Queries to a data integration system are stated in terms of relations in G, and are intended to provide the specification of which data to extract from the virtual database represented by I.

To integrate local schemas into a global one, semantic correspondences between concepts of these schemas needs to be specified. These correspondences can be elaborated using two different approaches: (1) by the use of procedural techniques (like SQL views) to map each concept from one source to another. This correspondence is expected to involve an human expert, or (2) by the use of ontologies that bring some levels of automation. In the first generation of integration systems (e.g., TSIMMIS [6]), data meaning was not explicitly represented. Thus, concept meaning and mapping meaning were manually encoded in a view definition. The major progress toward automatic integration resulted from some levels of explicit representation of data meaning through ontologies [32]. Ontologies are consensual and explicit representations of conceptualization [9]. Most ontologies used in proposed integration approaches [4] are based on words, and not on concepts. Such ontologies are called linguistic ontologies. To capture the meaning of the terms used as names of concepts and properties by sources, these approaches exploit the terminological relationships among terms represented by a thesaurus (like WordNet [18]). For example, WordNet has a set of terminological relationships like SYN (Synonymous-of), BT/NT representing Hypernym/Hyponym, Broader/Narrower. These terminological relationships are weighted. Consequently, these approaches offer approximate results (2 concepts are 90% similar). Their fundamental problem is their inability to integrate automatically at the meaning level several *heterogeneous* and *autonomous* data sources.

In a number of domains, including Web service, e-procurement, synchronization of distributed databases, the new *challenge* is to perform fully automatic integration of autonomous databases. We claim that: *if we do not want to perform human-controlled mapping at integration time, this mapping shall be done a priori during the database design.* This means that some formal shared ontologies must exist, and each local source shall embed some ontological data that references explicitly this shared ontology. Some systems are already developed based on this hypothesis: Picsel2 [25] project for integrating Web services, the COIN project for exchanging for instance financial data [8]. Their weakness is that once the shared ontology is defined, each source shall only use the common vocabulary. The shared ontology is in fact the integrated schema and each source has few schematic autonomy.

Our approach gives more schematic autonomy to various data sources participating in data integration process. To achieve this goal, three hypothesis are required:

1. Each data source participating in the integration process *shall* contain its own ontology. We call that source an ontology-based database (OBDB).
2. Each local source references a shared ontology ‘‘as much as possible’’ (see Section 6).
3. Local ontology may extend the shared ontology as much as needed.

When these assumptions hold, our integration process integrates automatically ontologies and then data of sources.

1.1. Context of our study

The context of our work is the automatic integration of industrial component data sources [24]. In this domain, each component supplier provides component catalogues and each manufacturing company represents component data in databases. Each catalogue, either on paper or electronic, has its own structure, and the used database has also a specific structure. It is to solve this integration problem that our approach has been developed since the early 1990s [22] in the context of ISO 184/SC4/WG2. The result of this work, known as PLIB (officially ISI 13584), allows to model, and to integrate automatically electronic catalogues and engineering databases designed following the PLIB model as documented in ISO 13584-25:2004. Indeed, both electronic catalogues and engineering databases include not only component data, but also their own ontology. Our approach requires that the target domain is already modelled by a shared consensual (e.g., standard) ontology. We already contributed to the development of such ontologies at the international standardization level (e.g., ISO DIS 13584-511, IEC 61360-4:1998, etc.). Note that in other domain, a number of other initiatives go to the same direction to ensure an automatic data integration process (see, for example, a travel agency ontology in [25]).

Our *a priori* data integration approach is based on *conceptual* ontologies. Contrary to linguistic ontologies, ontologies that we use represent language-independent concepts. They are formal, consensual, embedded within each data source (consequently they can be exchangeable), extensible using the subsumption relationship (each source may add whatever property or class). Like COIN [8] (where the ontology represents a contextual information of values), our ontology *also* represents the context of ontology definitions. In [3], preliminary results about our *a priori* data integration approach were presented.

1.2. Organization and contribution of the paper

The rest of this paper is organized as follows: in Section 2, we describe the background of the integration problem in the general context of heterogeneous sources, in Section 3 we propose a classification of integration approaches that facilitates the position of our work from the previous work, in Section 4 we present an overview of the PLIB ontology model that will be used as a basic support for our integration algorithms, in Section 5, we present the concept of ontology-based database and its structure, in Section 6 integration algorithms are presented, in Section 7 we show how our theory may be used in the domain of industrial components. Both component catalogues and component databases may be structured as ontology-based data sources making applicable the presented algorithms. Section 8 concludes the paper.

The main contributions of this paper are:

1. A new classification of integration systems using three orthogonal criteria's: (1) data representation, (2) the sense of the mapping between global and local schemas, and (3) the mapping automation.
2. An *a priori* approach ensuring a fully automatic integration process and respecting the schematic autonomy of each data source.
3. A new structure for representing both component catalogues and component databases. This structure allows catalogues and databases to switch from paper to data-explicit electronic catalogues, and from conventional component databases to engineering databases, respectively.

2. Background

Any integration system should consider both integration at schema level (schema integration consists in consolidating all source schemas into a global or mediated schema that will be used as a support of user queries) and at data level (global population). Constructing a global schema from local sources is difficult because sources store different types of data, in varying formats, with different meanings, and reference them using different names. Consequently, the construction of the global schema must handle different mechanisms for reconciling both data structure (for example, a data source may represent in the same field first and last name, when another splits it into two different fields), and data meaning (for example synonym, hypernym, hyponym, overlap, covering, and disjoint).

The main task to integrate heterogeneous data sources is the identification of equivalent concepts (and properties) used by these sources. To do so, different categories of conflicts should be solved. Goh et al. [8] suggest the following taxonomy: *naming conflicts*, *scaling conflicts*, *confounding conflicts* and *representation conflicts*. These conflicts may be encountered at *schema level* and *at data level*.

- *Naming conflicts*: occur when naming schemes of concepts differ significantly. The most frequently case is the presence of synonyms and homonyms. For instance, the status of a person means her familial status or her employment status.
- *Scaling conflicts*: occur when different reference systems are used to measure a value (for example, *price* of a product can be given in dollar or in euro).
- *Confounding conflicts*: occur when concepts seem to have the same meaning, but differ in reality due to different measuring contexts. For example, the weight of a person depends on the date where it was measured. Among properties describing a data source, we can distinguish two types of properties: *context-dependent properties* (e.g., the weight of a person) and *context non-dependent properties* (gender of a person).
- *Representation conflicts*: arise when two source schemas describe the same concept in different ways. For example, in one source, student's name is represented by two elements *FirstName and LastName* and in another one it is represented by only one element *Name*.

3. A classification of integration systems

It is rather difficult to classify the existing integration systems. Most of studies classified them using only one criteria. Some studies distinguish two categories of integrated systems those using *Local as View* (*LaV*) approach [7,25,15], and those using *Global as View* (*GaV*) approach [6]. Other contributions distinguish between systems using a *single ontology*, a *multiple ontologies*, and a *shared ontology* [32]. Some other work focus on the place of data and distinguish *mediator* and *warehouse* approaches [30]. We propose below three orthogonal criteria's, which allow to characterize the various existing integration systems. These criteria's will be described in next sections.

3.1. Data representation

This criteria specifies whether data of local sources are duplicated in a warehouse remained in local sources and then accessed through a mediator.

3.2. Sense of the mapping between the global and local schemas

In GaV systems, the global schema is expressed as a view (a function) over data sources. This approach facilitates the query reformulation by reducing it to a simple execution of views in traditional databases. However, changes in information sources or adding a new information source requires a database administrator to revise the global schema and the mappings between the global schema and source schemas. Thus, GaV is *not scalable* for large applications. In the source-centric approach, each data source is expressed with one or more views over the global schema. Therefore, LaV scales better, the global schema is defined as an ontology [25], independently of source schemas. In order to evaluate a query, a rewriting in terms of the data sources is needed. Rewriting queries using views is a difficult problem in databases [14]. Thus, LaV has *low performance* when queries are complex.

3.3. Mapping automation

This criteria specifies whether the mapping between the global schema and local schemas is done in a *manual*, a *semi-automatic*, or a *fully automatic way.* Manual mappings are found in the first generation of integration systems that integrate sources represented by a schema and a population (i.e., each source S_i is defined as: $\langle \mathrm{Sch}_i, \mathrm{Pop}_i \rangle$ as in classical databases) and without explicit meaning representations.

The manual systems focus mainly on query support and processing at the global level, by providing algorithms for identifying relevant sources and decomposing (and optimizing) a global query into sub queries for the involved sources. The construction of the mediators and the wrappers used by these systems is done manually because their main objective focus on global query processing [5].

To make the data integration process (partially) automatic, explicit representation of data meaning is necessary. Thus most

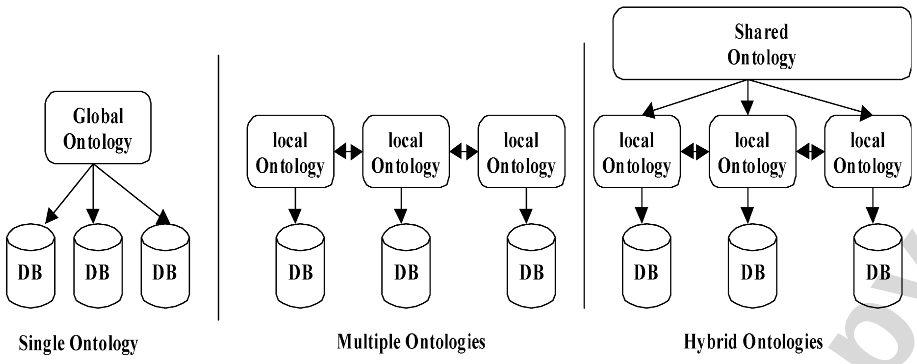


Fig. 1. Different ontology architectures.

of the recent integration approaches use ontologies [12,5,25]. Based on the way how ontologies are employed, we may distinguish three different architectures [32]: *single ontology methods*, *multiple ontologies methods*, and *hybrid methods* (see Fig. 1). In the single ontology approach, each source is related to the same global domain ontology (e.g., Lawrence and Barker [13] and Picsel [25,12]). As a result, a new source cannot bring new or specific concepts without requiring change in the global ontology. This violates the *source schematic autonomy* requirement (each source can extend its schema independently). In the multiple ontologies approach (e.g., Observer, [17]), each source has its own ontology developed without respect to other sources. Then, inter-ontology meanings need to be defined. In this case, definition of the inter-ontology mapping is very difficult as different ontologies may use different aggregation and granularity of ontology concepts [32]. The hybrid approach has been proposed to overcome the drawbacks of single and multiple ontologies approaches. In this approach, each source has its own ontology, but all ontologies are connected by some means to a common shared vocabulary (e.g., KRAFT project [31]).

In all these approaches, ontologies and ontology mappings are defined at integration time. Therefore, they always need a human supervision, and they are only partially automatic. To enable automatic integration, the semantic mapping shall be defined during the database design. This means that there shall exist a shared ontology, and moreover, each local source shall contain ontological data that refers to the shared ontology. Some systems have already been proposed on that direction such as *Picsel2* [25], and *COIN* [8]. But to remain automatic, these systems do not allow to each data source to add new concepts and properties.

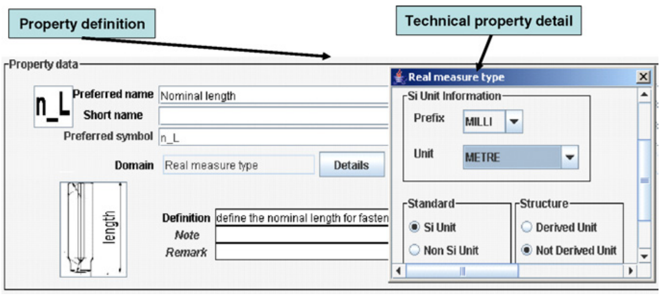
Our OBDB approach belongs to this category, but we also allow each data source to make its own extension in the shared ontology.

4. The PLIB ontology model

Several formalisms of ontologies were developed during last 10 years. One can cite for example, Ontolingua [10] for artificial intelligence applications, DAML + OIL and OWL [16] for Web applications, and PLIB for technical applications [24]. All these formalisms allow the following: (1) a representation and computer-sensible exchange of ontologies, (2) a representation and a data-processing exchange of objects defined in terms of these ontologies, and (3) a feasibility to develop consensual ontologies for more or less vast communities. In this paper, we use the PLIB ontology model because this model was specifically developed for describing technical objects and for providing integration mechanisms. Like in usual engineering practice and unlike in OWL (or in DAML), in PLIB, a technical object may be modelled by a set of different functional models, each one representing a particular discipline-specific representation (e.g., geometry procurement, simulation, etc.) [23]. The PLIB ontology model also allows to define more precisely each technical property (see Fig. 2) concerning integration mechanisms, the PLIB ontology model introduces, through a particular subsumption relationship collected *case-of* (see Section 4.3), a means to articulate formally together several ontologies. Indeed, in the technical domain, providing an approximate integration (as it might be obtained using linguistic ontologies) is worse than providing no answer at all. A number of domain ontologies based on this model already exist or are emerging (e.g., IEC, JEMIMA, CNIS, etc., see http://www.plib.ensma.fr) that covers progressively all the technical domain.

A PLIB ontology model has the following characteristics:

- *Conceptual*: each entity and each property are unique concepts completely defined. The terms (or words) used for describing them are only a part of their formal definitions.
- *Multilingual*: a universal identifier (UID) is assigned to each entity and property of the ontology. Textual aspects of their descriptions can be written in several languages (French,



Preferred name: Nominal length
Short name:
Preferred symbol: n_L
Domain: Real measure type
Definition: define the nominal length for fasten
Note:
Remark:
Prefix: MILLI
Unit: METRE
Standard: [✓ Si Unit] Non Si Unit
Structure: Derived Unit [✓ Not Derived Unit]

Fig. 2. A PLIB property description interface of PLIBEditor (an editor of PLIB ontology).

English, Japanese, etc.). The UID is used to identify *exactly* one concept (property or entity).

- *Modular*: an ontology can reference another one for importing entities and properties without duplicating them. Thus, providing for schematic autonomy of various sources that do reference a shared ontology.
- *Consensual*: the conceptual model of PLIB ontology is based on an international consensus and published as international standards (IEC 61630-4:1998, ISO 13584-42:1998) (for more details see [21]).
- *Unambiguous*: contrary to linguistic ontology models [21], where partially identical concepts are gathered in the same ontology-thesaurus with a similarity ratio (affinity) [5,29], each concept in PLIB has with any other concepts of the ontology well identified and explicit differences. Some of these differences are computer-interpretable and may be used for processing queries, e.g., difference of measure units, difference of evaluation context of a value.

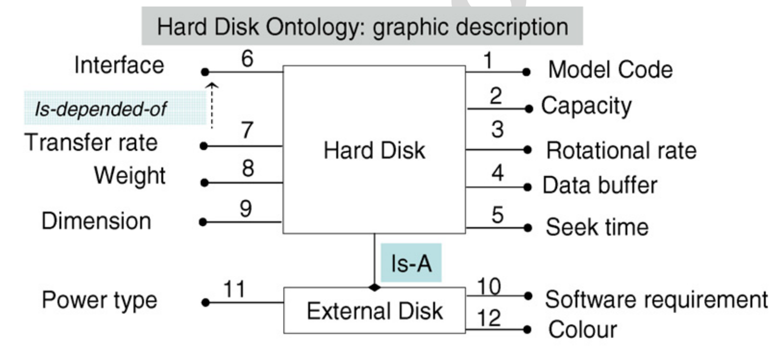
4.1. An automatic resolution of naming conflicts in PLIB

One of the utilization of UID is solving naming conflicts (due to synonyms and homonyms) as shown in the following example:

Example 1. Let S_1 be a source referencing the PLIB ontology model describing a Hard Disk in the French language (Fig. 3). This source has the schematic autonomy to use different language of its attributes (for example, it may use ‘‘Taux de transfer’’ instead of ‘‘Transfer rate’’). For the integrated system, these two attributes (properties) are similar because they have the same UID. More generally, if several sources use different names, we can identify easily whether they are different or identical using the following procedure:

1. These two properties have the same UID, for the integration system, these properties are identical (they represent the same thing, i.e., the transfer rate of a hard disk), even they have different names.
2. They have different UIDs, for the integration system, they are different, even they have the same name.

Note that unlike other integration systems based on linguistic ontologies where affinity measurements and thresholds are used to compute the similarity between concepts [5,29], orthogonality of PLIB ontologies and use of UID make resolution of naming conflicts deterministic and fully automated.



S1 : a source of external disk used for the french people

Ontology layer : instance of PLIB model (e.g., Properties part)

Property	GUI	Language Code	Property Name
...	...	...	...
Transfer rate	7	Fr	Taux de transfer
...	...	...	...

Definition	Is-depended-of	Type	...	...
...	...	...	...	...
...	6	Measure	...	...
...	...	...	...	...

Data layer: instance of PLIB instance schema

Code de modèle	Capacité	Interface	Taux de transfer
TS2OO2/128	128 (MB)	USB1.1	12 (Mb/sec)
...	...	...	...
KUSBDTI/1GB	1024 (MB)	USB2.0	480 (Mb/sec)

Fig. 3. An example of ontology exchange structure and ontology instances exchange structure.

4.2. PLIB class properties

The PLIB ontology model is represented by a tree of classes (in fact, a forest), where each class has its own properties. Two types of properties are distinguished: *characterization rigid properties* (a rigid property is a property that is essential to all instances of a class [11]) and *context-dependent properties* that are the value of which is a function depending on its measuring context. For instance, the data transfer rate of hard disk depends upon the interface used for connecting to its computer or the lifetime of bearing depends on its radical and axial load and on its rotational speed. Those variables that characterize the object context are called *context-parameters*.

4.3. Extension possibilities offered by PLIB

When a PLIB ontology is shared between various sources because these sources commit to the ontological definitions that were agreed and possibly standardized (e.g., IEC 61360, ISO 13399-100), each source remains autonomous and may extend the shared ontology using subsumption relationship. Two relationships are distinguished: specialization (is-a) and case-of.

4.4. The is-a relationship

It is the usual relationship that may be used within an ontology (either shared or specific to one source) to specialize a class. Note that both ontologies and ontology instances are modeled and exchanged together as instances of some schema. The ontology is exchanged as instances of the PLIB ontology model (see ontology layer in Fig. 3), and ontology instances are exchanged as instances of the PLIB instance schema (see data layer in Fig. 3). There are both ontologies and " data" might be stored in the same database (see Section 5). Fig. 3 considers a PLIB ontology describing an Hard Disk with nine properties. This class is specialized in order to define External Disk with three other properties.

4.5. The case-of relationship

In this case, properties are not inherited but may be explicitly (and partially) imported. Fig. 5 shows an extension of the shared ontology Hard Disk using the case-of relationship. The local ontology External Disk of S_2 imports some properties of the shared one (model code, capacity, interface, transfer rate, etc.). Note that this local ontology does not import some properties of the shared ontology like rotational rate, data buffer, etc. To satisfy its needs, it adds other properties describing its External Disk like *Read Rate*, *Write Rate*, *Marque* and *Price*. This particular relationship is a key mechanism allowing each source both to make its own extensions and to exchange information with other actors referencing the same shared ontology.

The PLIB ontology model is completely stable and several tools have already been developed to create, validate, manage or exchange ontologies (such tools can be found at PLIB home site: www.plib.ensma.fr).

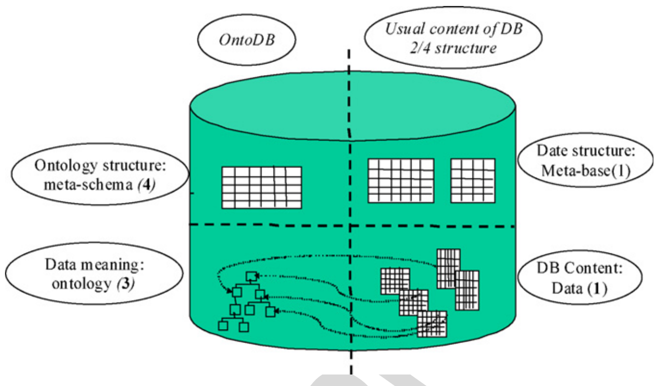

OntoDB
Usual content of DB 2/4 structure
Ontology structure: meta-schema (4)
Date structure: Meta-base(1)
Data meaning: ontology (3)
DB Content: Data (1)

Fig. 4. The ontology-based database architecture.

5. Ontology-based databases

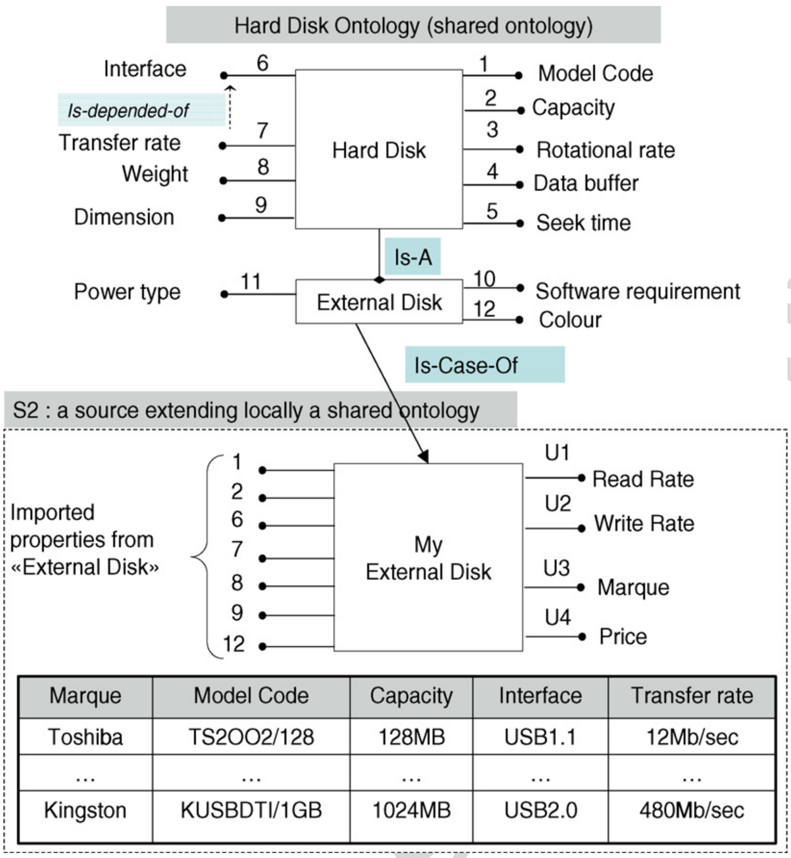
Contrary the existing database structures (that contain two parts: data according to a logical schema and a meta-base describing tables, attributes, foreign keys, etc.), an ontology-based database (OBDB) contains four parts: two parts as in the conventional databases plus the ontology definition and meta-model of that ontology. The relationship between the left and the right parts of this architecture associates to each instance in the right part its corresponding meaning defined in the left part. This architecture is validated by a prototype developed on Postgres called OntoDB. This concept of OBDB is the key concept for modeling both electronic catalogues and engineering databases (Fig. 4).

5.1. Formal definition of a PLIB ontology

Formally (for a more complete model, see [21]), a PLIB ontology may be defined as the quadruplet O: $\langle C, P, \text{Sub}, \text{Applic}\rangle$, where:

- C is the set of the classes used to describe the concepts of a given domain (like travel service [25], equipment failure, etc.);
- P is the set of properties used to describe the instances of the C classes. Note that it is assumed that P defines a much greater number of properties that those usually represented in a database. Only a subset of them might be selected by any particular database.[1]
- Sub is the subsumption (is-a and case-of) function (Figs. 3 and 5) defined as $\text{Sub}: C \rightarrow 2^C$.[2] For a class c_i of the ontology it associates its direct subsumed classes.[3] Sub defines a partial order over C.
- Applic is a function defined as Applic: $C \rightarrow 2^P$. It associates to each ontology class those properties (either characterization or context-dependent properties) that are applicable (i.e., rigid) for each instance of this class. Applicable properties are inherited through is-a subsumption and partially imported through case-of subsumption.

[1] A particular database may also extend the P set.
[2] We use the symbol 2^C to denote the power set of C.
[3] C_1 subsumes C_2 iff $\forall_x \in C_2, x \in C_1$.



Marque	Model Code	Capacity	Interface	Transfer rate
Toshiba	TS2OO2/128	128MB	USB1.1	12Mb/sec
…	…	…	…	…
Kingston	KUSBDTI/1GB	1024MB	USB2.0	480Mb/sec

Fig. 5. An example of extending locally a shared ontology.

Note that as usual ontological definitions are intentional: the fact that a property is rigid for a class does not mean that value will be explicitly represented for each instance of the case. In our approach, the choice is made among applicable properties at the schema level.

Example 2. Fig. 5 gives an example of an shared ontology with two classes C = {Hard Disk and External Disk}. Let P = {Model Code, Capacity, Rotational rate, Data buffer, Seek time, Interface, Transfer rate, Weight, Dimension, Power type, Software requirement, Colour} be set of properties that characterizes these classes. Properties in P will be assigned to classes of the ontology (therefore, each class will have its own applicable properties). The subsumption function Sub defines a is-a relationship between classes (for example, the class Hard Disk subsumes the class External Disk).

5.2. *Representation of instances*

The main goal of ontologies is the representation of the semantic of objects of a given domain. This goal is reached by assigning objects to classes and by describing them using properties. According to the used ontology model, various constraints govern such descriptions. For instance, in OWL, an object may belong to any number of classes and can be described by any subset of properties. Therefore, each domain object of a given class has its own structure. In the opposite, a database schema aims to describe “similar” objects by an identical logical structure in order to optimize queries using indexing techniques. In the lack of any particular assumption on the representation of objects, the only possible common structure consists in associating each object: (1) with any subset of set of classes; and (2) with any subset of the set of properties.

The drawbacks of such structure are evident. Either it would require an important storage cost (by using a systematic representation of all properties for every instance), or it would require also a significant response time due to the need of performing a large number join operations (if only relevant properties and their relevant classes are represented for each instance).

Therefore, in the context of our proposed architecture OntoDB, we impose three restrictions referenced to as the strong typing assumptions.

- *R1*. We assume that the set of classes to which an instance belongs to are ordered by the subsumption relationship has a unique minimal class (called *instance base class*).
- *R2*. Each ontology specifies for each of its classes, those properties that are allowed for use in a class instance (applicable properties).
- *R3*. Each object can be described only by applicable properties of its instance base class.

Thanks to the strong typing assumptions, it is possible to define a relevant schema for any ontology class. This schema consists of all applicable properties that are used at least by one instance of a class. Of course this schema might contain in some case a number of null values. But it will support an efficient query processing.

5.3. A formal representation of an OBDB

For the purpose of simplicity in formulas, we assure that each non-leaf class[4] is abstract. Formally, an *OBDB* is a quadruplet $\langle O, I, \text{Sch}, \text{Pop}\rangle$, where:

- O is an ontology (O: $\langle C, P, \text{Sub}, \text{Applic}\rangle$);
- I is the set of instances of the database;
- Sch: $C \rightarrow 2^P$ associates to each ontology class c_i of C the properties, which are effectively used to describe the instances of the class c_i. Sch has two definitions based on the nature of each class (a leaf or a non-leaf class).
  - *Schema of each leaf class* c_i is explicitly defined. It shall only ensure the following:

$$\forall c_i \in C, \quad \text{Sch}(c_i) \subset \text{Applic}(c_i) \qquad (1)$$

(Only applicable properties may be used for describing class instances of c_i).

  - *Schema of a non-leaf class* c_j is computed. It is defined by the intersection between the applicable properties of c_j and the intersection of properties associated with values in all subclasses $c_{i,j}$ of c_j.

$$\text{Sch}(c_j) = \text{Applic}(c_j) \bigcap \Big(\bigcap_{c_{i,j} \in \text{Sub}(c_j)} \text{Sch}(c_{i,j}) \Big) \qquad (2)$$

An alternative definition may also be used to create the schema of a non-leaf class where instances are completed with null values:

$$\text{Sch}'(c_j) = \text{Applic}(c_j) \bigcup \Big(\bigcap_{c_{i,j} \in \text{Sub}(c_j)} \text{Sch}(c_{i,j}) \Big) \qquad (3)$$

- Pop: $C \rightarrow 2^I$ associates to each class (leaf class or not) its own instances. The instances of each leaf class c_i are explicitly implemented and the Pop(c_j) of each non-leaf class c_j is computed as the following:

$$\text{Pop}(c_j) = \bigcup_{c_{i,j} \in \text{Sub}(c_j)} \text{Pop}(c_{i,j}) \qquad (4)$$

Example 3. Let's consider the class tree in Fig. 6 where A, B are a non-leaf classes and C, D are leaf-classes. We assume that each class has it own applicable properties, and the DBA (database administrator) has chosen its schema for C, D and a formula (2) or (3) for all non-leaf classes. To find the schema of the class A using Eq. (2), we first perform the intersection operation among all properties of the schema of its subsumed classes. We obtain a set $U = \{a_1\}$, then we perform the intersection operation between U and the applicable properties of A ($\{a_0, a_1, a_2\}$). As result the schema of A contains one property a_1 (see Fig. 6).

[4] If c_i is a leaf class, this means $\text{Sub}(c_i) = \phi$.

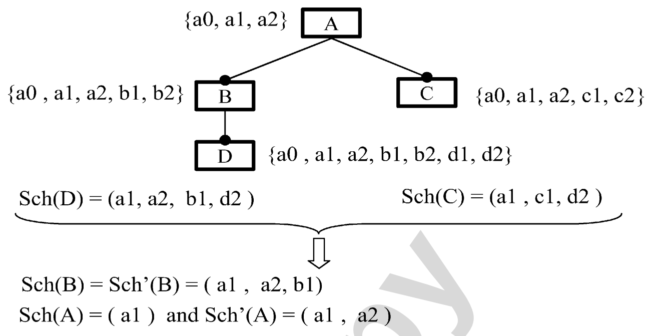

{a0, a1, a2}
A
{a0 , a1, a2, b1, b2}
B
C
{a0, a1, a2, c1, c2}
D
{a0 , a1, a2, b1, b2, d1, d2}
Sch(D) = (a1, a2, b1, d2)
Sch(C) = (a1 , c1, d2)
Sch(B) = Sch'(B) = (a1 , a2, b1)
Sch(A) = (a1) and Sch'(A) = (a1 , a2)

Fig. 6. An example of a non-leaf class schema (properties between {} are applicable properties).

By using Sch′ definition (Eq. (3)), Sch′(A) would be (a_1, a_2). The instances from C will be associated with NULL value for property a_2.

6. Algorithms for integrating ontology-based database sources

In this section, we present algorithms to integrate various ontology-based database sources that correspond to the same domain. A typical scenario is the one of databases of a particular domain like travel agencies [25]. Each supplier references the same domain ontology and adds its own extensions. Let $S = \{\text{SB}_1, \text{SB}_2, \ldots, \text{SB}_n\}$ be the set of data sources participating in the data integration process. Each data source SB_i ($1 \leq i \leq n$) is defined as follows: $\text{SB}_i = \langle O_i, I_i, \text{Sch}_i, \text{Pop}_i\rangle$. We assume that all sources have been designed referencing as much possible a common shared ontology O. As much possible means that (1) each class of a local ontology references explicitly (or implicitly through its parent class) its lowest subsuming class in the shared ontology, and (2) only properties that do not exist in the shared ontology may be defined on a local ontology, otherwise it should be imported through the case-of relationship. This requirement is called smallest subsuming class reference requirement (S2CR2). Each source is designed following three steps:

1. The DBA of each source defines her own ontology O_i: $\langle C_i, P_i, \text{Sub}_i, \text{Applic}_i\rangle$ ensuring S2CR2;
2. The DBA of each source chooses for each leaf class properties that are associated with values by defining $\text{Sch}_i: \ C_i \rightarrow 2^{P_i}$;
3. The DBA chooses an implementation of each leaf c_i class (e.g., to ensure the third normal form), and then she defines Sch(c_i) as a view over c_i implementation.

The architecture of our integrated systems is shown in Fig. 7. Note that the structure of the integrated system is also an ontology-based database. We may distinguish several integration operations, corresponding to different assumptions on the relationships between the various ontologies involved in the integrated system. Thus defining an algebraic structure over OBDBs associated with automatic integration algorithms.

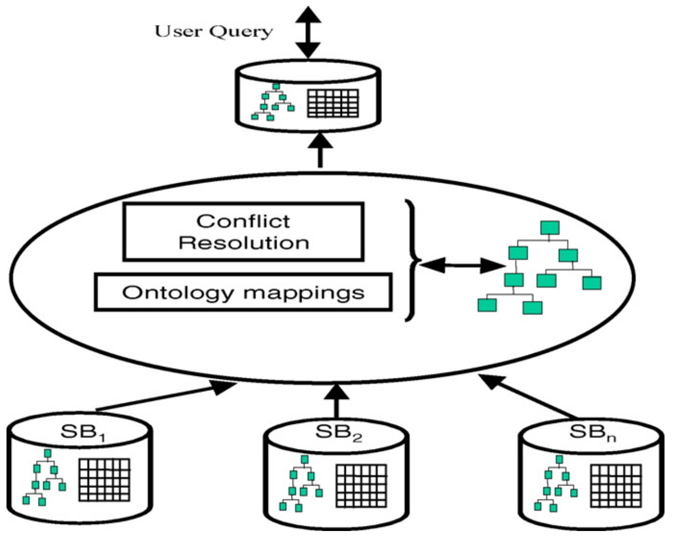


Fig. 7. Architecture of our integration system.

Two approaches seem of particular interests. In both approaches, each local ontology must be a specialization of the shared ontology O (to ensure the schematic autonomy of a local source). This specialization is done through explicit subsumption using the case-of relationship (Fig. 5) and should respect the S2CR2 requirement. Then, two approaches are:

1. *ExtendOnto*: the shared ontology is itself extended with all the local specializations of the various local sources. Then, the local instances are integrated within the integrated system without any change.
2. *ProjOnto*: the shared ontology is not modified. Each local source instance is projected onto the applicable properties of its smallest subsuming class in the shared ontology, and then it is added to the population of this class of the shared ontology.

Another approach called *FragmentOnto* that assumes the shared ontology is completed enough to cover the needs of all local sources. Such an assumption is done for instance in the Picsel2 project [25] for integrating web service (travel agency) or in COIN [8]. The source design consists in (1) selecting subset of the shared ontology (classes and properties), and (2) designing the local database schema. This approach has been presented in [3]. It leaves less schematic autonomy for the design of the local.

6.1. An integration algorithm for ExtendOnto

Note that our assumption about the relationship between the shared ontology and local source ontologies leaves a lot of schematic autonomy to each data source:

- the classification of each source may be completely different from the shared ontology;
- some classes and properties that do not exist in the shared ontology and may be added in the local ontologies.

But all the ontologies reference "as much possible" (S2CR2) a shared ontology: O: $\langle C, P, \mathrm{Sub}, \mathrm{Applic}\rangle$. Therefore, each source SB_i maps the referenced ontology O to its ontology O_i. This mapping can be defined as follows: $M_i : C \to 2^{C_i}$, where $M_i(c) = \{$greatest classes of C_i subsumed by $c\}$. Thus, with the S2CR2 assumption, each data source SB_i is now defined as a quintuple: $\mathrm{SB}_i = \langle O_i, I_i, \mathrm{Sch}_i, \mathrm{Pop}_i, M_i\rangle$. In this case, automatic integration is possible. To do so, we should find the structure of the final integrated system I^F: $\langle O^F, \mathrm{Sch}^F, \mathrm{Pop}^F\rangle$. Note that the structure of O^F is $\langle C^F, P^F, \mathrm{Sub}^F, \mathrm{Applic}^F\rangle$, where element of these structures are defined as follows:

- $C^F = C \cup_{(i|1\le i\le n)} C_i$,
- $P^F = P \cup_{(i|1\le i\le n)} P_i$,
- $\mathrm{Applic}^F(c) = \begin{cases} \mathrm{Applic}(c) & \text{if } c \in C \\ \mathrm{Applic}_i(c) & \text{if } c \in C_i \end{cases}$
- $\mathrm{Sub}^F(c) = \begin{cases} \mathrm{Sub}(c) \cup M_i(c) & \text{if } c \in C \\ \mathrm{Sub}_i(c) & \text{if } c \in C_i \end{cases}$

Then, the population Pop^F of each class (c) is computed recursively using a post-order tree search. If c_i belongs to one C_i and does not belong to C, its population is given by: $\mathrm{Pop}^F(c_i) = \mathrm{Pop}_i(c_i)$. Otherwise (i.e., c belongs to the shared ontology tree), $\mathrm{Pop}^F(c)$ is computed as follows:

$$\mathrm{Pop}^F(c_j) = \bigcup_{c_{i,j} \in \mathrm{Sub}^F(c_j)} \mathrm{Pop}^F(c_{i,j}) \tag{5}$$

Finally, the schema of each class c of the integrated system is computed following the same principle as the population of c. If c_i belongs to one C_i and does not belong to C, its schema is determined by: $\mathrm{Sch}^F(c_i) = \mathrm{Sch}_i(c_i)$. And $\forall c \in C$, $\mathrm{Sch}^F(c)$ is computed using the formula (2) (respectively, (3)). This shows that it is possible to leave a large schematic autonomy to each local source and to compute in a fully automatic, deterministic and exact way the corresponding integrated system. To the best of our knowledge, our ontology-based database approach is the first approach that reconciles these two requirements.

Example 4. Suppose we have two ontology-based databases referencing a shared ontology as in Fig. 8. Local class E in source 1 (OBDB1) references the class C_2 (by importing only a_1, a_2, a_3 properties). The local class F of source 2 references the class C_3 of the shared ontology (by importing three attributes a_1, a_5 and a_6). Moreover classes E and F add new properties e and f, respectively. To integrate these two sources, the shared ontology should be extended by considering the two added local classes (Fig. 8).

6.2. An integration algorithm for ProjOnto

This scenario differs from the previous one by the fact that the shared ontology is not modified. Thus, $O^F = O$:

- $C^F = \mathrm{C}$,
- $P^F = P$,
- $\mathrm{Applic}^F(c) = \mathrm{Applic}(c)$,
- $\mathrm{Sub}^F(c) = \mathrm{Sub}(c)$.

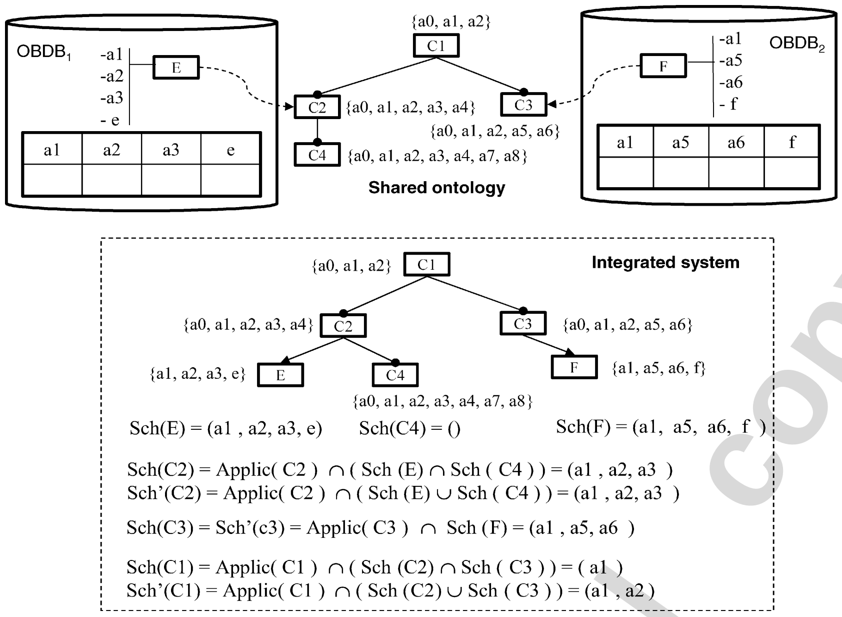

{a0, a1, a2}
C1
OBDB1
-a1
-a2
-a3
- e
E
C2 {a0, a1, a2, a3, a4}
C3
{a0, a1, a2, a5, a6}
C4 {a0, a1, a2, a3, a4, a7, a8}
a1 a2 a3 e
Shared ontology
-a1
-a5
-a6
- f
F
OBDB2
a1 a5 a6 f
Integrated system
{a0, a1, a2} C1
{a0, a1, a2, a3, a4} C2
C3 {a0, a1, a2, a5, a6}
{a1, a2, a3, e} E
C4
F {a1, a5, a6, f}
{a0, a1, a2, a3, a4, a7, a8}
Sch(E) = (a1 , a2, a3, e)
Sch(C4) = ()
Sch(F) = (a1, a5, a6, f)
Sch(C2) = Applic(C2) ∩ (Sch (E) ∩ Sch (C4)) = (a1 , a2, a3)
Sch'(C2) = Applic(C2) ∩ (Sch (E) ∪ Sch (C4)) = (a1 , a2, a3)
Sch(C3) = Sch'(c3) = Applic(C3) ∩ Sch (F) = (a1 , a5, a6)
Sch(C1) = Applic(C1) ∩ (Sch (C2) ∩ Sch (C3)) = (a1)
Sch'(C1) = Applic(C1) ∩ (Sch (C2) ∪ Sch (C3)) = (a1 , a2)

Fig. 8. ExtendOnto scenario (properties between { } are applicable properties).

In this scenario, we consider that each instance of a local source is projected onto the applicable properties of its smallest subsuming class in the shared ontology. So this class becomes instance base class (note that in the previous scenario, no class of shared ontology is instance base class). Let $\mathrm{Pop}^*(c)$ be the projected instances population on a class of the shared ontology c. It is computed by the union of all local classes referencing directly the class c.[5] $\mathrm{Pop}^*(c)$ is given the following equation:

$$\mathrm{Pop}^*(c) = \bigcup_{i \in [1:n]} \Big(\bigcup_{c_j \in M_i(c)} \mathrm{Pop}_i(c_j) \Big) \quad (6)$$

The schema of projected instances of a class c is given as follows:

$$\mathrm{Sch}^*(c) = \bigcup_{i \in [1:n]} \Big(\bigcup_{c_j \in M_i(c)} \mathrm{Sch}_i(c_j) \Big) \quad (7)$$

or

$$\mathrm{Sch}^*(c) = \bigcap_{i \in [1:n]} \Big(\bigcup_{c_j \in M_i(c)} \mathrm{Sch}_i(c_j) \Big) \quad (8)$$

The populations and the schemas of the leaf classes of the shared ontology are computed by the formulas (6) and (7), respectively (or (6) and (8)). For a non-leaf classes, formulas (6)–(8) must be completed:

$$\mathrm{Sch}^F(c) = \mathrm{Applic}(c) = \bigcap \Big(\mathrm{Sch}^*(c) \bigcup \Big(\bigcup_{c_k \in \mathrm{Sub}^F(c)} \mathrm{Sch}^F(c_k) \Big) \Big) \quad (9)$$

or

$$\mathrm{Sch}^F(c) = \mathrm{Applic}(c) = \bigcap \Big(\mathrm{Sch}^*(c) \bigcap \Big(\bigcap_{c_k \in \mathrm{Sub}^F(c)} \mathrm{Sch}^F(c_k) \Big) \Big) \quad (10)$$

and in both cases:

$$\mathrm{Pop}^F(c) = \Big(\bigcup_{c_k \in \mathrm{Sub}^F(c)} \mathrm{Pop}^F(c_k) \Big) \bigcup \mathrm{Pop}^*(c) \quad (11)$$

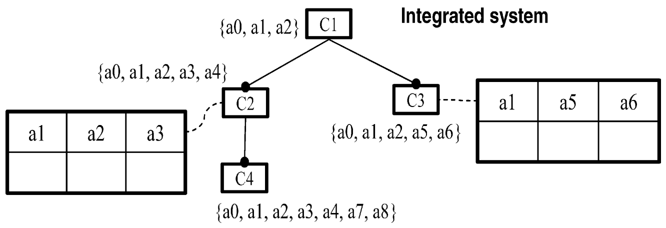
Example 5. We consider the same sources and the shared ontology as in the previous example. The structure of the integrated system is described in Fig. 9. The class C_2 is a non-leaf class and instance base class.

It is important to notice that when all data sources use independent ontologies without referencing a shared ontology (1) the same algorithm allows to record in the integrated system, the content of each data source together with its own ontology, or (2) the task of mapping these ontologies (i.e., defining M_i, $1 \leq i \leq n$) onto a receiver ontology may be done manually, by the DBA of the receiving system. In this latter case, the integration process will be performed automatically either or as in the ProjOnto case as in the ExtendOnto case.

7. Integrating electronic catalogues of industrial component within engineering database

In most engineering fields, products to be designed are essentially assembled of pre-existing components. 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 behaviour of components and on the relevant component representation for each specific discipline [20].

[5] Local classes of a source SB_i references directly a class c of shared ontology are the classes in $M_i(c)$.



Sch*(C4) = () and Sch(C4) = ();

Sch*(C2) = (a1, a2 , a3) $\Rightarrow$
Sch(C2) = Applic(C2) $\cap$ (Sch(C4) $\cup$ Sch*(C2)) = (a1 , a2, a3);
<u>or</u>
Sch(C2) = Applic(C2) $\cap$ (Sch(C4) $\cap$ Sch*(C2)) = (a1 , a2, a3);

Sch*(C3) = Sch(C3) = (a1 , a5 , a6);

Sch*(C1) = () $\Rightarrow$
Sch(C1) = Applic(C1) $\cap$ ((Sch(C2) $\cap$ Sch(C3)) $\cap$ Sch*(C1)) = (a1)
<u>or</u>
Sch(C1) = Applic(C1) $\cap$ ((Sch(C2) $\cup$ Sch(C3)) $\cup$ Sch*(C1)) = (a1 , a2)

Fig. 9. ProjOnto scenario (properties between { } are applicable properties).

The 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 the 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 Fig. 10).

'SCREW-ISO1014-L10-D5-GRADa'

Fig. 10. Engineering information encoding in an usual component database.

7.1. *From conventional component database to engineering database*

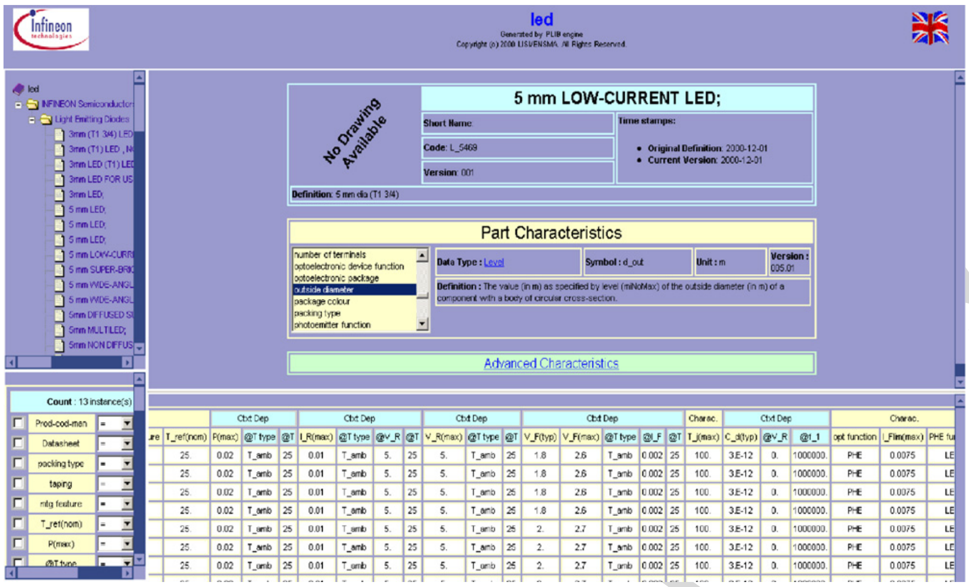
PLIB ontologies allow to make explicit the component engineering knowledge. This knowledge may then be represented within an OBDB together with component data. In such databases, that we call engineering database, all the technical data about components may be explicitly represented. Moreover the data meaning described in the database as a PLIB ontology allows to provide user friendly interfaces, allowing to display together data and data meaning [26,24]. Fig. 12 shows an example of a graphic interface that may be automatically generated.

7.2. *From paper-like catalogues to explicit electronic catalogues*

The information content of a component catalogue is exactly the same as in engineering database. This means that it contains an explicit local ontology that possibly references a standard ontology, and description of components by means of the values of their technical properties, and possibly, associated textual documentation. All these pieces of information may be modelled by the PLIB model. Moreover, the PLIB model being developed in the EXPRESS language, it is possible to exchange all these pieces of information as an electronic file (see Fig. 11) that represents in fact the exact content of an OBDB. Thus, integrating such files into an engineering database turns to the problem of automatic integration of OBDB discussed in Section 6. Note that if a customer has no engineering database, such electronic catalogues may be generated and exchanged as electronic document. Fig. 12 is an example of such an electronic catalogue generated in DHTML

```
#6=GLOBAL_LANGUAGE_ASSIGNMENT('en');

#1=SUPPLIER_BSU('112/2///61360_4_1', *);
#2=SUPPLIER_ELEMENT(#1, #3, '01', #4, #5);
#3=DATES('2000-03-02', '2000-03-02', $);
#4=ORGANIZATION('IEC 61360-4', 'IEC', 'IEC 61360-4');
#5=ADDRESS($, $, $, $, $, $, $, 'SWITZERLAND', $, $, $, $);

#10=CLASS_BSU('AAA000', '001', #1);
#11=ITEM_CLASS(#10, #12, '01', #13, TEXT('Root class providing a name scope for the entire IEC 61360 reference collection'), $, (#20), (), $, (#20), (), $);
#12=DATES('2000-03-02', '2000-03-02', $);
#13=ITEM_NAMES(LABEL('IEC reference collection'), (), LABEL('IEC'), $, $);

#20=PROPERTY_BSU('AAE000', '001', #10);
#21=NON_DEPENDENT_P_DET(#20, #22, '01', #23, TEXT('Main class in the classification tree'), $, $, $, $, (), $, 'A52', #24, $);
#22=DATES('2000-03-02', '2000-03-02', $);
#23=ITEM_NAMES(LABEL('Main class'), (), LABEL('main class'), $, $);
#24=NON_QUANTITATIVE_CODE_TYPE('X..8', #25);
#25=VALUE_DOMAIN((#26, #28, #30), $, $, ());
#26=DIC_VALUE(VALUE_CODE_TYPE('CO'), #27, $);
#27=ITEM_NAMES(LABEL('Component'), (), LABEL('CO'), $, $);
#28=DIC_VALUE(VALUE_CODE_TYPE('FEA'), #29, $);
#29=ITEM_NAMES(LABEL('Geometry'), (), LABEL('FEA'), $, $);
#30=DIC_VALUE(VALUE_CODE_TYPE('MA'), #31, $);
#31=ITEM_NAMES(LABEL('Material'), (), LABEL('MA'), $, $);

#40=CLASS_BSU('AAA001', '001', #1);
```

Fig. 11. An example of an electronic catalogue exchanged as instances of the EXPRESS language.



led
Generated by PLIB engine
Copyright (c) 2000 LISI/ENSMA. All Rights Reserved.

- led
  - INFINEON Semiconductor
    - Light Emitting Diodes
      - 3mm (T1 3/4) LED
      - 3mm (T1) LED , N
      - 3mm LED (T1) LED
      - 3mm LED FOR US
      - 3mm LED;
      - 5 mm LED;
      - 5 mm LED;
      - 5 mm LED;
      - 5 mm LOW-CURR
      - 5 mm SUPER-BRI
      - 5 mm WIDE-ANGL
      - 5 mm WIDE-ANGL
      - 5mm DIFFUSED S
      - 5mm MULTILED;
      - 5mm NON DIFFUS

No Drawing Available

5 mm LOW-CURRENT LED;

Short Name:
Code: L_5469
Version: 001
Time stamps:
- **Original Definition:** 2000-12-01
- **Current Version:** 2000-12-01

Definition: 5 mm dia (T1 3/4)

Part Characteristics

number of terminals
optoelectronic device function
optoelectronic package
outside diameter
package colour
packing type
photoemitter function

Data Type : Level | **Symbol :** d_out | **Unit :** m | **Version :** 005.01

Definition : The value (in m) as specified by level (miNoMax) of the outside diameter (in m) of a component with a body of circular cross-section.

Advanced Characteristics

Count : 13 instance(s)

☐	Prod-cod-man	=
☐	Datasheet	=
☐	packing type	=
☐	taping	=
☐	mfg feature	=
☐	T_ref(nom)	=
☐	P(max)	=
☐	@T type	=

		Ctxt Dep			Ctxt Dep				Ctxt Dep			Ctxt Dep					Charac.	Ctxt Dep			Charac.		
re	T_ref(nom)	P(max)	@T type	@T	I_R(max)	@T type	@V_R	@T	V_R(max)	@T type	@T	V_F(typ)	V_F(max)	@T type	@I_F	@T	T_j(max)	C_d(typ)	@V_R	@t_1	opt function	I_Flm(max)	PHE fu
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	1.8	2.6	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	1.8	2.6	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	1.8	2.6	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	1.8	2.6	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	2.	2.7	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	2.	2.7	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE
	25.	0.02	T_amb	25	0.01	T_amb	5.	25	5.	T_amb	25	2.	2.7	T_amb	0.002	25	100.	3.E-12	0.	1000000.	PHE	0.0075	LE

Fig. 12. An electronic catalogue automatically generated for the Web (each Ctx Dep set of columns gathers context-dependent properties, with their context parameters values).

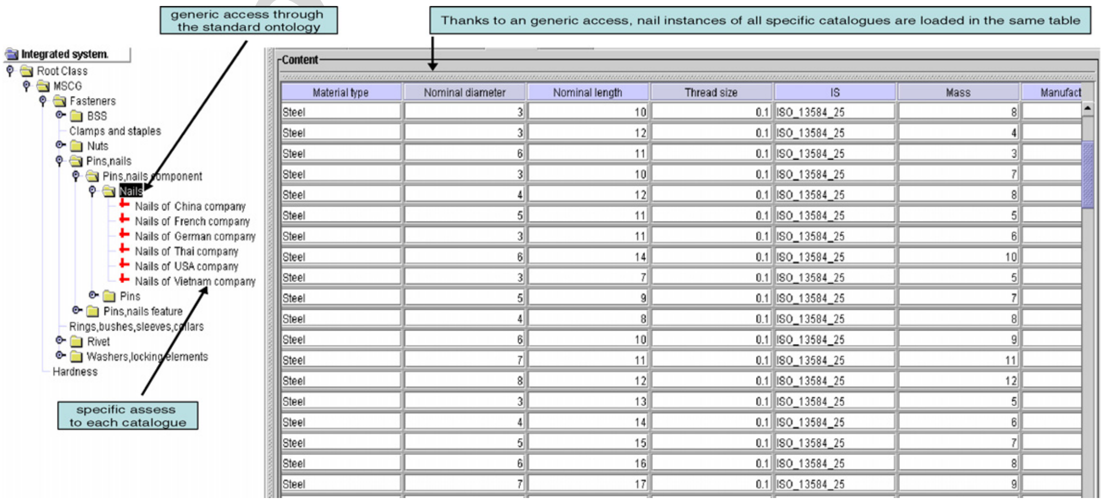
as an active documents for the Web. This catalogue may be browsed through Internet. It may be loaded on the user site for use by the company designers without needing to access to Internet, it may also be used to select a component and either to issue an ordering form to the supplier, or to get the data description of the component for insertion within a current design. Note that, in this catalogue, each property value that corresponds to a context-dependent property is associated with the values of the context parameter of which it depends.

7.3. Interpretation of OBDB integration scenario for industrial components

We have already noticed that standard PLIB ontologies were emerging covering more and more industrial sectors. We may quote for instance IEC 61360-4 for electronic components, ISO 13399 for manufacturing cutting tools, ISO 13584-511 for mechanical fasteners, ISO 13584-501 for measuring instrument, etc. (see http://www.plib.ensma.fr or www.oiddi.org). In those domains where such ontologies exist, we make the following assumptions:

- Component manufacturers will reference as much as possible standard ontologies for describing their own component catalogues.
- Component users having an engineering database will record in their engineering database the relevant standard ontologies (associated e.g., with SQL view to define their own ontologies).

With the above assumptions, the two automatic integration algorithms presented in Section 6 permit the following:


generic access through the standard ontology
Thanks to an generic access, nail instances of all specific catalogues are loaded in the same table
specific assess to each catalogue

Integrated system.
- Root Class
  - MSCG
    - Fasteners
      - BSS
      - Clamps and staples
      - Nuts
      - Pins,nails
        - Pins,nails component
          - Nails
            - Nails of China company
            - Nails of French company
            - Nails of German company
            - Nails of Thai company
            - Nails of USA company
            - Nails of Vietnam company
          - Pins
        - Pins,nails feature
      - Rings,bushes,sleeves,collars
      - Rivet
      - Washers,locking elements
    - Hardness

Content

Material type	Nominal diameter	Nominal length	Thread size	IS	Mass	Manufact
Steel	3	10	0.1	ISO_13584_25	8	
Steel	3	12	0.1	ISO_13584_25	4	
Steel	6	11	0.1	ISO_13584_25	3	
Steel	3	10	0.1	ISO_13584_25	7	
Steel	4	12	0.1	ISO_13584_25	8	
Steel	5	11	0.1	ISO_13584_25	5	
Steel	3	11	0.1	ISO_13584_25	6	
Steel	6	14	0.1	ISO_13584_25	10	
Steel	3	7	0.1	ISO_13584_25	5	
Steel	5	9	0.1	ISO_13584_25	7	
Steel	4	8	0.1	ISO_13584_25	8	
Steel	6	10	0.1	ISO_13584_25	9	
Steel	7	11	0.1	ISO_13584_25	11	
Steel	8	12	0.1	ISO_13584_25	12	
Steel	3	13	0.1	ISO_13584_25	5	
Steel	4	14	0.1	ISO_13584_25	6	
Steel	5	15	0.1	ISO_13584_25	7	
Steel	6	16	0.1	ISO_13584_25	8	
Steel	7	17	0.1	ISO_13584_25	9	

Fig. 13. An example of nail integrated system.

1. For ProjOnto, to integrate automatically all components of all manufacturer electronic catalogues as instances of the standard ontology;
2. For ExtendOnto, to store in the user database the complete content (ontology + component data) of all manufacturer electronic catalogues with two kinds of possible accesses: (1) a generic access through the standard ontology (if any) and (2) a specific assess to each catalogue by going down the integrated ontology hierarchy (see Fig. 13).

8. Conclusion

In this paper, we have presented a new classification of integration systems based on three criteria's: (1) data representation, (2) sense of the mapping between global and local schemas, and (3) level of automation of the mapping (manual, semi automatic and automatic). We also proposed a fully automated technique for integrating heterogeneous sources called the *a priori* approach to data integration. This approach assumes the existence of a shared ontology and guarantees the schematic autonomy of each source that can extend the shared ontology to define its local ontology. This extension is done by adding new concepts and properties. The ontologies used by our approach are modelled according to a formal, multilingual, extensible, and standardized (ISO 13584) ontology model known as PLIB that was developed for modelling industrial components. The fact that the ontology is embedded within each data source helps in capturing both the domain knowledge and the knowledge about data, schema, and properties. Therefore it allows a complete automation of the integration process, contrary to the existing techniques. Two integration algorithms are presented (1) when sources extend the shared ontology by specific classes and properties and that these extensions are represented in the integrated system and (2) when sources extend the shared ontology but instance data are projected onto the shared ontology before being exchanged. This model of a data source that contains its own ontology is called ontology base data base (OBDB). We have shown that OBDB could be the structure of both user component database (called engineering database) and of electronic catalogues. In this case, our algorithms allow automatic integration of supplier catalogues into user engineering database.

In addition to its capability for automating the integration process of heterogeneous databases (several prototypes of ontology-based databases are currently in progress in our laboratory), there are many other tasks currently in progress. They include the following: (1) development of an XML schema for exchanging OBDBs and electronic catalogues, this one, called OntoML is currently under development within ISO TC184/SC4/WG2, (2) development of a query language for OBDB, (3) design of web services for component information exchange and for B2B e-engineering.

References

[1] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, R. Weber, Active xml: peer-to-peer data and web services integration, in: Proceedings of the International Conference on Very Large Databases, 2002, p. 1090.

[2] L. Bellatreche, K. Karlapalem, M. Mohania, Some issues in design of data warehousing systems, in: S.A. Becker (Ed.), Developing Quality Complex Data Bases Systems: Practices, Techniques, and Technologies, Idea Group Publishing, 2001.

[3] L. Bellatreche, G. Pierra, D. Nguyen Xuan, H. Dehainsala, Y. Ait Ameur, An a priori approach for automatic integration of heterogeneous and autonomous databases, in: International Conference on Database and Expert Systems Applications (DEXA'04), 2004, 475–485.

[4] S. Castano, V. Antonellis, Semantic dictionary design for database interoperability, in: Proceedings of the International Conference on Data Engineering (ICDE), 1997, pp. 43–54.

[5] S. Castano, V. Antonellis, S.D.C. Vimercati, Global viewing of heterogeneous data sources, IEEE Transactions on Knowledge and Data Engineering 13 (2) (2001) 277–297.

[6] S.S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J.D. Ullman, J. Widom, The tsimmis project: integration of heterogeneous information sources, in: Proceedings of the 10th Meeting of the Information Processing Society of Japan, 1994, pp. 7–18.

[7] F. Franois Goasdoué, V. Lattès, M.C. Rousset, The use of carin language and algorithms for information integration: the picsel system, International Journal of Cooperative Information Systems 9 (4) (2000) 383–401.

[8] C.H. Goh, S. Bressan, E. Madnick, M.D. Siegel, Context interchange: new features and formalisms for the intelligent integration of information, ACM Transactions on Information Systems 17 (3) (1999) 270–293.

[9] T. Gruber, A translation approach to portable ontology specification, Knowledge Acquisition 5 (2) (1995) 199–220.

[10] T.R. Gruber, Ontolingua: a mechanism to support portable ontologies, Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, Revision, 1992.

[11] N. Guarino, C.A. Welty, Ontological analysis of taxonomic relationships, in: Proceedings of 19th International Conference on Conceptual Modeling (ER'00), 2000, pp. 210–224.

[12] F. Hakimpour, A. Geppert, Global schema generation using formal ontologies, in: Proceedings of 21st International Conference on Conceptual Modeling (ER'02), 2002, pp. 307–321.

[13] R. Lawrence, K. Barker, Integrating relational database schemas using a standardized dictionary, in: Proceedings of the ACM Symposium on Applied Computing (SAC), 2001, pp. 225–230.

[14] A. Levy, Answering queries using views: a survey, VLDB Journal 10 (4) (2001) 270–294.

[15] A.Y. Levy, A. Rajaraman, J.J. Ordille, The worldwide web as a collection of views: query processing in the information manifold, in: Proceedings of the International Workshop on Materialized Views: Techniques and Applications (VIEW'1996), 1996, pp. 43–55.

[16] D.L. McGuinness, F. Harmelen, Owl web ontology language overview, W3C Recommendation (2004) 10.

[17] E. Mena, V. Vipul Kashyap, A. Illarramendi, A.P. Sheth, Managing multiple information sources through ontologies: relationship between vocabulary heterogeneity and loss of information, in: Proceedings of Third Workshop on Knowledge Representation Meets Databases, 1996.

[18] A.G. Miller, Wordnet: a lexical database for English, Communications of the ACM 38 (11) (1995) 39–41.

[19] B. Omelayenko, D. Fensel, A two-layered integration approach for product information in b2b e-commerce, in: Proceedings of the Second International Conference on Electronic Commerce and Web Technologies, 2001, pp. 226–239.

[20] P. Paasilia, A. Aatonen, A. Riitahuta, Automatic component selection, in: C. Kooi, P.A. MacConaill, J. Bastos (Eds.), Proceedings of the CIME Conference, IOS Press, 1993.

[21] G. Pierra, Context-explication in conceptual ontologies: the PLIB approach, in: R. Jardim-Gonçalves, J. Cha, A. Steiger-Garçao (Eds.), Proceedings of Concurrent Engineering CE'2003, Special track on Data Integration in Engineering, Madeira, Portugal, UNINOVA, A.A. Balkema, July, 2003, pp. 243–254.

[22] G. Pierra, An object oriented approach to ensure portability of cad standard parts libraries, in: Proceedings of the European Computer

Graphics Conference and Exhibition (Eurographics'90), 1990, pp. 205–214.

[23] G. Pierra, A multiple perspective object oriented model for engineering design, New Advances in Computer Aided Design and Computer Graphics (1993) 368–373.

[24] G. Pierra, J.C. Potier, E. Sardet, From digital libraries to electronic catalogues for engineering and manufacturing, International Journal of Computer Applications in Technology 18 (2003) 27–42.

[25] C. Reynaud, G. Giraldo, An application of the mediator approach to services over the web, in: Special Track Data Integration in Engineering, Concurrent Engineering (CE'2003), 2003, 209–216.

[26] E. Sarder, G. Pierra, H. Murayama, Y. Oodake, Y. Ait-Ameur, Simplified representation of parts library: model practice and implementation, in: Proceedings of PDT Days, 18, QMS Edition, Brussels, (2001), pp. 27–42.

[27] M. Schonhoff, M. Strassler, K.R. Dittrich, Version propagation in federated database systems, in: International Database Engineering and Applications Symposium (IDEAS '01), 2001, 189–198.

[28] A. Sheth, J.A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Computing Surveys 22 (3) (1990) 183–236.

[29] G. Terracina, D. Ursino, A uniform methodology for extracting type conflicts and & subscheme similarities from heterogeneous databases, Information Systems 25 (8) (2000) 527–552.

[30] J.D. Ullman, Information integration using logical views, Proceedings of the International Conference on Database Theory (ICDT), Lecture Notes in Computer Science, vol. 1186, January 1997, pp. 19–40.

[31] P.R.S. Visser, M. Beer, T. Bench-Capon, B.M. Diaz, M.J.R. Shave, Resolving ontological heterogeneity in the kraft project, in: Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA '99), 1999, pp. 668–677.

[32] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hiibner, Ontology-based integration of information—a survey of existing approaches, in: Proceedings of the International Workshop on Ontologies and Information Sharing, 2001, pp. 108–117.

Ladjel Bellatreche received his PhD in Computer Science from the Clermont Ferrand University, France in 2000. He is an Assistant Professor at Poitiers University, France since 2002. Before joining Poitiers University, he has been a Visiting Researcher at Hong Kong University of Science and Technology (HKUST) from 1997 to 1999 and also been a Visiting Researcher in Computer Science Department at Purdue University, USA during summer 2001. He has worked extensively in the areas: distributed databases, data warehousing, heterogeneous data integration using formal ontologies, designing databases using ontologies, schema and ontologies versioning. He has published more than 35 research papers in these areas in leading international journal and conferences. Ladjel has been associated with many conferences as program committee member.

Nguyen Xuan Dung received his BSc in Computer Science from Hanoi University of Technology, Vietnam in June 2002, and the master degree in Computer Science from the University of Poitiers in 2003. He is currently a PhD student in Laboratory of Applied Computer Science at National Engineering School for Mechanics and Aerotechnics (LISI/ENSMA), Poitiers. His current research interests include ontology-based integration of information sources, schema and ontology versioning, and databases.

Guy Pierra is Professor of Computer Science at ENSMA, Poitiers, and Director of LISI. His main interests are data engineering, software engineering, geometric modeling and human–computer interaction. He published one book and hundreds of papers on these topics. Since the late 80s he is working on a database-oriented ontology model, known as PLIB that has been standardized in ISO as the ISO 13584 standard series. The data engineering team of LISI is involved in a number of research projects regarding the use of ontologies in database theory including: ontology-based data integration, development of new architectures for ontology-based database, query languages, application programming interfaces and exchange formats for ontology-driven data, with a special focus on management of industrial data, on e-engineering and on the semantic Web.

Dehainsala Hondjack received his BSc in Computer Science from African Institute of Computer Science, Gabon in September 2002, and the master degree in Computer Science from the University of Poitiers in 2003. He is currently a PhD student in Laboratory of Applied Computer Science at National Engineering School for Mechanics and Aerotechnics (LISI/ENSMA), Poitiers. His current research interests include ontology-based integration of information sources, schema and ontology versioning, and ontology-based database design.