

From digital libraries to electronic catalogues for engineering and manufacturing

G. PIERRA, J.C. POTIER and E. SARDET

{*pierra | potier | sardet*}@ensma.fr

LABORATORY OF APPLIED COMPUTER SCIENCE (LISI)
National School of Engineers in Mechanics and Aeronautics (ENSMA)
Téléport 2 - 1, avenue Clément Ader - BP 40109
86961 FUTUROSCOPE Cedex
(to be published in IJCAT)

Introduction

In a number of engineering areas, such as electronic engineering or mechanical design, products are mainly defined from pre-existing components. Availability of digital libraries representing pre-existing components, and their exchange or remote access through networks would drastically increase the efficiency and the quality of the design process of such products. Since 1990, a normative project takes place in ISO to define a neutral format for representing families of parts, thus providing for exchanging parts libraries between heterogeneous systems. This multi-part specification, known under the P-Lib acronym (officially, ISO 13584 "Parts Library"), is intended to become a series of International Standards. Some parts have already reached the International Standard (IS) stage, some others are still in progress.

The P-Lib specification follows the object oriented paradigm. It represents the content of a parts catalogue as a class hierarchy. It uses a meta-description technique [Ait-Ameur and al. 95] for characterizing intentionally component classes, and not explicitly each component instance, through selection rules, instantiation constraints, and instances behavior specification. However, the database-oriented view of the P-Lib specification, and the lack of user-friendly tools to capture data, made it difficult, for parts suppliers, to describe their own components using the P-Lib format. This paper proposes an approach intended to bring together the database oriented view carried by the P-Lib specification and a document-oriented view as it appears in catalogues on the same parts data. Indeed, data intended to be stored in a P-Lib compliant database already exist in traditional paper catalogues. Capturing these data directly from catalogues would appear as an efficient and friendly method. It would avoid to re-enter already existing data, and it would offer the parts supplier a familiar capture interface: its own parts catalogue.

The content of this paper is as follows. In the first section, we outline the P-Lib data model. This model is described using the EXPRESS [Schenck and al. 94] data specification language defined in ISO 10303-11[ISO10303.11-94]. In a second section, we discuss how the gap between an EXPRESS-defined data model and an XML-defined document structure [Bray and al. 98] on the same content might be filled. We propose an approach allowing to merge their respective advantages: semantics control and data integrity resulting from an EXPRESS data model, and human legibility resulting from an XML Document Type Definition (DTD). The result is an XML representation of the P-Lib EXPRESS data model. The third section presents a tool developed in order to capture P-Lib data just by tagging a catalogue document. An example illustrates how this capturing process occurs. Finally, two possible uses of the captured data are presented: parts library physical files generation for exchange purpose between P-Lib databases, and parts library Web server generation for remote access to an electronic parts catalogue.

1. The P-Lib Standard

The Parts Library standardization initiative was launched at the ISO level in 1990. Its goal [Pierra 94][Pierra 97] was to develop a computer-interpretable representation of parts library data to enable a full digital information exchange between component suppliers and users.

1.1 P-Lib contents

The main points of the P-Lib approach are the following:

- an object oriented approach as the more efficient way for capturing knowledge about components;
- an information model formally specified in EXPRESS to ensure portability on different platforms and software systems (both data management systems and CAx systems);
- an exchangeable data dictionary mechanism to support the integration of multi-supplier libraries and the progressive standardization of the data element types that describe the various kind of components (e.g., a *screw*, a *door*, a *capacitor*) and the various technical properties of components (e.g., the *threaded_diameter* of a screw, the *maximum_working_temperature* of a pump, the *capacity* of an electric capacitor);

- separation of components definitions (carried by one particular class hierarchy: the general model classes hierarchy), and components representations (carried by functional model classes) to support multi-representation (e.g., geometry, schematics, or simulation models) of the various components.

1.1.1 P-Lib description methodology

A parts library is not only intended to be exchanged between computers that process data. It is also intended to be accessed by human catalogue-users who shall understand the meaning of data. In paper catalogues, a supplier normally defines in the very first pages the categories of components, and, for the various categories, the properties that are meaningful to describe some aspects of these components or some aspects of the context in which it is intended to insert these components. These definitions are often pretty long, and they consist of several pieces of information (e.g. for a property, its definition, its acceptable domain of values, its possible measurement unit, its symbolic representation in a formula). Each of these definitions are intended to define a concept, and to associate it with a symbol that consists of a word, or a group of words. Such a symbol may thereafter be used throughout the catalogue to represent instances of the concept wherever the concept applies.

When shifting from paper catalogues to electronic ones, a double requirement appears. First, each concept shall be defined as precisely (hopefully, more precisely) as in a paper document, both in a human readable and in a computer sensible ways. Second, each concept shall be associated with a computer-sensible symbol (an identifier) that will stand for the concept wherever it is intended to be used.

The capability to build such computerized concepts dictionaries, or ontologies, was provided by means of an EXPRESS information model [Sardet and al. 97] and of a methodology both developed as a joint effort with a standardization committee of the International Electrotechnical Commission (IEC/SC3D). This common EXPRESS information model has been published twice: once as the IEC 61630-2 standard and once as the ISO 13584-42 standard [ISO13584.42-98]. An important point is that IEC/SC3D has already built a content according to this structure and currently uses it. IEC 61630-4 [IEC61360.4-97] defines and enables references to most of the technical properties and families of components existing in the electrotechnical area. A number of other standard P-Lib dictionaries are currently under development [Langlois 96][Bazari 98].

The basic idea of this above approach [Pierra 94][Pierra and al. 98], called the P-Lib dictionary model, is that a technical property cannot be defined without defining, in the mean time, its field of application by means of families of components, and conversely, that a family of components cannot be defined without defining, in the mean time, the technical properties that characterize this family.

Therefore, a data dictionary conforming to ISO 13584-42/IEC 61630-2 consists of two parts:

- a classification tree where components families and technical properties are identified and connected;
- a set of templates that describe successively each components family and each technical property.

A technical 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 dimensional equation, a unit, a source document, a symbol, a formula, an ISO 31 classification, etc.

The P-Lib dictionary model identifies and provides for describing three different kinds of pertinent properties:

- A *part characteristic* is an invariable property, characteristic of a part, whose value is fixed once the part is defined. It enables to characterize a part. For example, *the inner diameter* and *outer diameter* of a *bearing* are part characteristics for this *bearing*.
- A *context parameter* is a property whose value characterizes the context in which a part is intended to be inserted. It enables to characterize the usage conditions of a part and the design problem to be solved by the part. For example, the *dynamic-load* intended to be applied to a *bearing* is a context parameter for this *bearing*.
- A *context dependent characteristic* of a part is a property of a part whose value depends on some context parameter(s). For a given part, a context dependent characteristic is mathematically defined as a function whose domain is defined by some context parameter(s) defining the part environment. It enables to characterize the behavior of a part. For example, in case of a *ball-bearing*, the *life time* is a context dependent characteristic that depends on the *radial load*, the *axial load* and the *rotational speed*, and the bearing itself.

The main interest of this taxonomy is that it enables to capture relationships between these properties within a P-Lib library, and thus, to model the supplier knowledge about components behavior and selection rules.

Before describing his catalogue, a supplier shall first define its P-Lib dictionary, possibly by reference to pre-existing (e.g., standard) data dictionaries (for instance: IEC 61630-4). A classification tree of components families is to be built, and, at each level of the defined hierarchy, properties are defined. Properties are inherited

by the lower levels families. Thus, within a P-Lib library, the properties defined or inherited by each tree node may be used to query the whole sub-tree of its children families.

1.1.2 From dictionaries to libraries: modeling licit instances

A catalogue does not contain only definitions of components families and definitions of properties: it also enumerates by some means the various components that belong to each family. In the P-Lib specification, this enumeration is done intentionally, i.e., by means of predicate rules.

P-Lib contains EXPRESS information models that provide for associating properties with both their domains of allowed values, and the functions which enable to compute values of properties from values of other properties. Such a complete description of a catalogue population is called a P-Lib library. Thanks to the capability to model dynamic expressions, a P-Lib library may contain not only data about components, but also various pieces of supplier knowledge about behavior and selection rules of these components.

Figure 1 shows the information that may be defined by a P-Lib library supplier and provided to its customers:

- a *characteristics* table, whose content enumerates the various bearings that belong to the family and defines the value of their characteristic properties (*inner diameter, outer diameter, maximum speed, etc.*);
- the definition of some *context parameters*, whose values are intended to be provided by the user to describe its design problem (*axial load, radial load, intended speed*), and
- the definition of some *context dependent characteristics*, such that the bearing *life-time*, whose values are intended to be computed by the P-Lib library management system, from values selected by the user (characteristics and context parameters) and from computation rules (derivation functions) provided by the supplier.

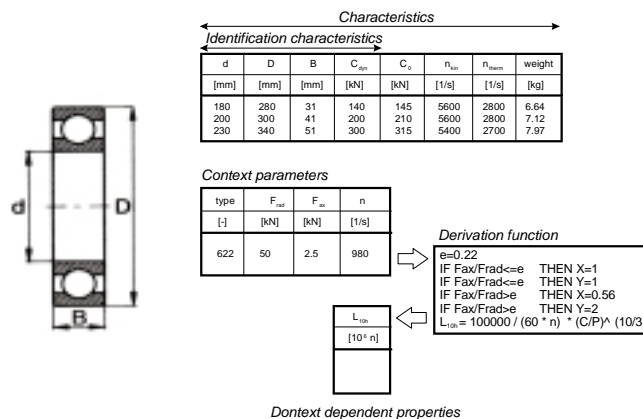


Figure 1 – Example of knowledge captured in a P-Lib library

Thanks to this information recorded in a P-Lib library, the user is able, for instance, to select all the *bearings*, with a given *internal diameter*, whose *life time* is greater than 10 000 hours in given conditions of *load* and *speed*.

1.1.3 Components representation

A major user requirement regarding parts libraries and electronic catalogues is the capability to provide representations (e.g., schematic, geometry, etc.) of a selected component. This requirement is supported by the P-Lib standard. In fact, P-Lib does not define "new" kinds of representations. It just enables to associate representations defined by any other Standards (or by a private agreement between the sender and the receiver) with each parts implicitly defined in a P-Lib library. For each particular kind of representation, a small specification called a view exchange protocol (VEP) needs to be developed. It just specifies how such a representation shall be connected with the part it corresponds to. For instance, the P-Lib VEP 102 specifies how to exchange explicit representations of each library component as a STEP AP conforming representation.

2. From libraries to catalogues

The previous section outlined the various pieces of information that may be captured in a P-Lib library. In the P-Lib standard, this information is specified by means of EXPRESS information models. Such information models ensure P-Lib data integrity and consistency through the semantic constraints that may be described in the EXPRESS language (strong typing, cardinality constraints, set theory constraints, predicates, etc.). But it appeared that such a database oriented view of a catalogue content was difficult to understand even for parts suppliers who are familiar with the catalogue document.

Indeed, a catalogue document, either in paper or in electronic form (SGML/XML/HTML), gives a user friendly presentation of the information it contains. Moreover, this information often contains most of P-Lib required pieces of information. But, a number of these pieces of information are only informally described in the textual part of a catalogue. They may only be extracted by human beings, for instance the catalogue users.

Conversely, each piece of information within the population of an EXPRESS information model is explicitly characterized. It may be easily read by computers. Moreover, each piece of information may be enforced to fulfill a number of constraints that ensure reliability of the exchanged information. But, on the other hand, information is broken in so many unordered small pieces with so many relations between them that it cannot be apprehended by human beings. Computers are always needed to give a synthetic representation.

In this section, we propose a method for integrating these two approaches to gather their respective advantages. A by-product of this integration is the capability to offer P-Lib suppliers a tool allowing to capture P-Lib data just by locating, or by "tagging", for each parts family described in a catalogue, the place where are represented (if they are effectively represented) the various P-Lib defined pieces of information.

Performing such an integration involves four issues:

- what are the general rules that could be defined for mapping an EXPRESS information model onto an XML DTD;
- how to represent the object oriented paradigm conveyed by the EXPRESS language onto XML, in particular how to represent inheritance and polymorphism [Rumbaugh and al. 91] [Meyer 88] in an XML DTD;
- in the special case of the P-Lib information model, how to apply these rules and representation technique to maximise the legibility of the resulting document;
- and finally, how to use the supplier catalogue as a user capture interface without changing its presentation when new tags are introduced.

Barring the XML representation of inheritance and polymorphism, extensively discussed in [Sardet 99] and which is not presented here to keep this paper in reasonable length, all the issues are discussed in the follow-up.

2.1 Data model vs. document legibility

A fundamental difference between EXPRESS and XML is the way used to represent relationships between pieces of information. In fact, such relationships may be represented according to two different approaches:

- an "external" *referencing mode*, where each object is represented as a stand-alone piece of information and references other objects by means of identifiers;
- an "internal" *referencing mode*, where objects may be embedded within the context of the instance(s) that reference(s) them.

The first approach, scattered, simplifies data management and prevents information duplication. However, it makes the information not human readable. The second approach is mainly legibility oriented.

2.1.1 "Internal" referencing mode

Let's consider the example given in Figure 2 below. The used notation is the graphical EXPRESS one (EXPRESS-G). Simple boxes represent entities. Boxes with a vertical bar on their right side represent a data type. Finally, each line represents an attribute. Note that these lines are oriented toward their rounded extremity.

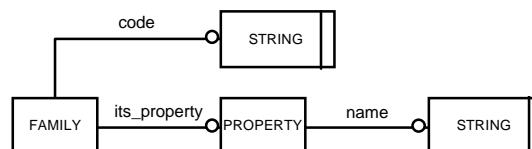


Figure 2 – An example of a simple EXPRESS data model

A *family* entity is defined by a *code* attribute, whose data type is a string, and by an association relationship, *its_property*, whose data type is the *property* entity. Finally, a *name* attribute, whose data type is a string again, characterizes the *property* entity.

An internal referencing mode applied to instances belonging to the previous data model, expressed, for example, in an exchange format similar to the STEP exchange format ISO 10303-21 [ISO10303.21-94], would be the following (Figure 3):

```
#1 = FAMILY('F1', PROPERTY('P1'));
#2 = FAMILY('F2', PROPERTY('P2'));
```

Figure 3 – Instances representation according to an internal mode

In the used notations, each instance is associated with an instance identifier (#i) that provides for referencing it, and attributes values (or references) are embedded within each instance.

Property instances characterizing *family* instances are directly represented inside *family* instances. The same referencing mode is used for representing characteristics properties of *family* and *property* entities. The main advantage of such a representation is legibility of the created instances. However, this approach raises an important issue: let's assume that the following relationship holds: "two different families have a common property". Modeling this relationship would mean creating one *property* instance for each *family* class instance. Consequently, information is duplicated, and this particular dependency is lost.

2.1.2 "External" referencing mode

In external referencing mode, each entity is represented independently and in any order. Besides, entities are unambiguously identified. Thus, identifiers may be used to model relations, and so, each created instance is completely autonomous. Figure 4 illustrates such a representation.

```
#1 = FAMILY('F1', #3);
#2 = FAMILY('F2', #4);
#3 = PROPERTY('P1');
#4 = PROPERTY('P2');
```

Figure 4 – Instances representation according to an external referencing mode

The main advantage of the external referencing mode is that information duplication is now avoided, whatever be the number of references to the same information. Within a database context, updating is easy and referential integrity is preserved. The drawback is for human beings: information legibility is completely lost when dealing with large instances populations.

2.1.3 EXPRESS and XML

The EXPRESS language uses an external referencing mode. Because of the lack of any "record" construct in EXPRESS, structured entities may not be embedded within each other. They may only be referenced by means of their identifier (called *entity_id*). Thus, the population of an EXPRESS information model consists of a huge number of small and unordered pieces of information. Conversely, XML mainly uses an internal referencing mode. A document is a collection of sections and subsections intended to appear in a given order, thus defining a tree structure intended to maximize legibility. The structural organization of information in a document is therefore deeply different from the one defined by an EXPRESS model.

In fact, XML supports also the external referencing mode where relations are modeled by references to instance identifiers (or tag identifiers): each XML block, called an element (*ELEMENT*), shall be defined as embedded within another block. However, it may also be associated with an identifier. In such a case, it may be directly referenced from any other block, wherever it is in the tree structure, by means of this identifier.

Rationale for the choice of a referencing mode (internal or external) when representing a particular EXPRESS relationship in XML is based on comparisons presented in Table 1. A data model instance referenced only once should be represented in XML as embedded within the instance that references it. Conversely, an XML representation of a data model instance intended to be referenced by more than one other instance shall be done according to an external referencing mode. In this second case, a particular XML *ELEMENT* (called "container") shall be defined within the tree structure of the XML DTD, to gather instances intended to be referenced from multiple sources.

	Unique Reference	Multiple References
<ul style="list-style-type: none"> Internal mode #1 = FAMILY('F1', PROPERTY('P1')); #2 = FAMILY('F2', PROPERTY('P1')); 	Good legibility (presentation)	Information duplication
<ul style="list-style-type: none"> External mode #1 = FAMILY('F1', #3); #2 = FAMILY('F2', #3); #3 = PROPERTY('P1'); 	<ul style="list-style-type: none"> Bad legibility Need of an <i>ELEMENT</i> gathering <i>property</i> instances 	<ul style="list-style-type: none"> Consistency Need of an <i>ELEMENT</i> gathering <i>property</i> instances

Table 1 – Internal vs. External referencing mode

2.1.4 Ordering the authoring representation of a data model

The authoring (XML) representation of the complete information represented in an EXPRESS data model requires to design a tree-structured presentation.

Each entity within the EXPRESS model becomes an *ELEMENT* in the XML DTD. Each attribute in the EXPRESS model becomes an (embedded) *ELEMENT* in the XML DTD. Choice of internal or external

referencing mode for representing relationships between *ELEMENTs* corresponding to EXPRESS entities and the design of the tree-structured presentation, are defined by the following general guidelines:

1. Possibly, partition the information model into several documents. For instance, we might decide that a document may only describe one family of parts.
2. Identify the different categories of stand-alone concepts expressed in the data model. Stand-alone concepts means concepts for which some occurrences may be added without adding any other occurrences of any other concepts of the same level. For example, in Figure 2, stand-alone concepts are, on the one hand, *families*, and, on the other hand, *properties* belonging to these *families*.
3. Choose a presentation order between concepts occurrences belonging to a same level. Each concept occurrence becomes a documentary section. In our example (see Figure 2), a structured presentation might be to represent firstly all the *properties* defining the various *families*, and secondly, the set of *families*. Another representation might be to represent each *family* description directly followed by its own *properties* description.
4. Refine for each section defined in 3:
 - a) identify the stand-alone concepts within each document section;
 - b) if each concept occurrence is referenced only in the context of one another concept occurrence, use the internal referencing mode and embed it within its referencing node;
 - c) if each occurrence of one concept is referenced by several occurrences of one another concept, create a document section to be a "container" for the occurrences of the first concept as a subsection of the documentary section corresponding to the second concept, and use an external mode referencing mechanism for referencing them;
 - d) if multiple references came from several instances of several concepts of the same level, get up the "container" at the level of the sections corresponding to the higher level referencing concept, and use an external referencing mode representation.

Note that the above process cannot be fully automated. It is a value-added design process where the information modeled in EXPRESS is added a presentation order and structure intended to enhance its legibility. Thus, the above rules are only design guidelines.

In the next section, we outline the result of these rules when applied to the P-Lib EXPRESS model to design a P-Lib DTD.

2.2 The P-Lib XML data model

In this clause, a direct application of the mapping process of an EXPRESS data model onto XML is presented. The goal is to build a P-Lib DTD able not only to contain all pieces of information of the P-Lib EXPRESS model, but also to present them in a human readable way.

2.2.1 Stand-alone concepts

The fundamental P-Lib concepts are parts *supplier* and parts *family*. We decide that a document instance may only represent either a parts supplier or a parts family description. For the description hereafter, we discuss only documents that contain parts family descriptions. Moreover, for simplification purposes, only the P-Lib data dictionary (no explicit description of the families content) is described. The complete P-Lib DTD is defined in [Sardet 99].

A P-Lib parts *family* is defined through a small number of high level concepts. Mains of them are *class*, *property* and *table*. A *class* information model is built through a *class* identification (*class_BSU*) and description that includes a definition and that references describing *properties* and *tables*. A property is characterized by its identification (*property_BSU*), its *name*, its *domain*, etc. A *table* is mainly described through its identification (*table_BSU*), its *name* and its *columns*. It should be noticed that, in the P-Lib model, columns (and also expression variables) make indirect references to properties by way of a *property semantics* entity. This entity stands for the interpretation function that associates meaning to values. These autonomous concepts may now be structured and arranged. This process is discussed in the next subclause.

2.2.2 Ordering and structuring

Following the design guidelines, a first attempt to define the P-Lib document structure is presented in Figure 5. A class family is organized according to three main successive sections: *class*, *properties* and *tables*. The two last sections contain the data dictionary description of respectively each property and each table involved in the parts family description. A property data dictionary description, like a *table* description, is subdivided in two sub-sections: *identification* (BSU) and *description*. Note that some of the next level sections are also shown (level 3 concepts). The goal is discussed latter on the XML modeling of the *property semantic* concepts.

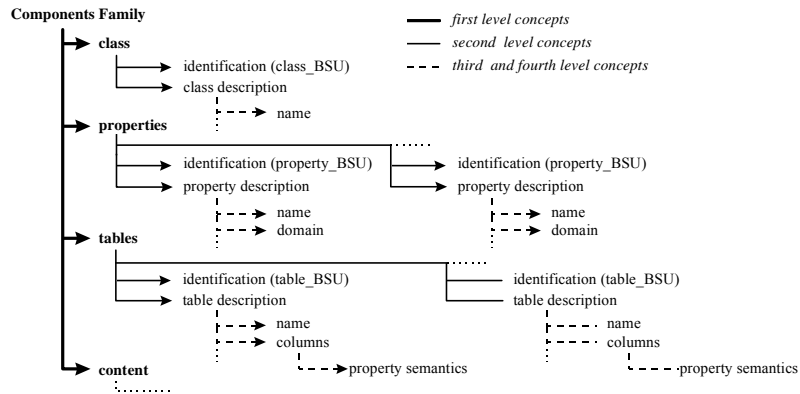


Figure 5 – A first possible document structure for P-Lib

Note that the last high level section is intended to describe the *content* of a family, omitted till then. The main interest of this cutting out is to preserve the separation defined in the P-Lib EXPRESS model between these two data representation levels, allowing to define a data dictionary (e.g., standard data dictionary such as IEC 61360-4) without any content.

This first analysis suggests a first arrangement for the concepts belonging to a same level. Now, we check the cardinality of the various relationships to validate the different choices done between external and internal referencing mode.

2.2.3 Refinement

Class, *property* or *table* are all associated with a *name* entity. But, it seems probable (even it does not appear explicitly in the EXPRESS model) to consider that a *name* cannot participate to the description of more than one concept. Thus, the relationship between a name and the characterized concept is equal to one for each role. The rule defined in 2.1.4 stipulates to use an internal reference mode for representing the *name* entity. In a same way, each *property* is characterized by a value *domain*. Conversely, each value *domain* participates to the description of only one *property*. The referencing mode to be chosen is again the internal one.

The case of the *property semantics* concept is less obvious. Indeed, this concept is used to define the means of a *column*. But, several *columns* may reference the same *property semantics*. This one-to-many relationship requires to use an external referencing mode: a *property semantics* "container" is to be created at a tree depth equal to the depth of the concepts that reference *property semantics*, therefore, at the *table description* level. But, a more detailed study of the P-Lib data model shows another use of the *property semantics* concept in the expressions that belong to the parts family *content*. We must therefore pull up the "container" at a tree depth similar to the *content*. Figure 6 summarizes the final structure resulting from the design guidelines defined in 2.1.4. External references are represented by "ref." labeled arrows pointing to a particular occurrence of a given concept.

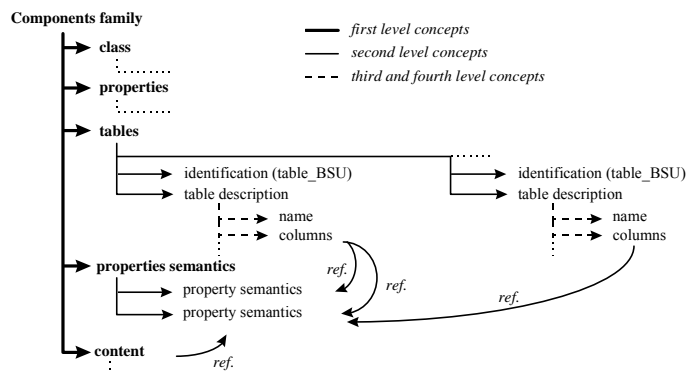


Figure 6 – Found DTD structure for representing the P-Lib data model

2.3 Using HTML for the P-Lib document presentation

The previous DTD structure would enable to record in an XML document instance all the information elements that describe a particular P-Lib library. But, our goal is to enable a library supplier to capture these information elements using a pre-existing HTML document as a capture interface. This, in turn, requires to preserve the initial document presentation while the capturing process occurs. Therefore, the same document should be used both for capturing purposes and for storing the dynamically tagged P-Lib.

To achieve this goal, we propose to use a DTD that separates the semantic content of a document from its presentation. This is done by integrating the XML P-Lib DTD, whose global structure has been presented in the previous clause, and the HTML DTD for the content. We call these DTD respectively *semantic DTD* and *presentation DTD*. Thus, the global DTD on which document instances are based consists of two parts: a semantic part, instance of the semantic DTD, and a presented part, instance of the HTML DTD.

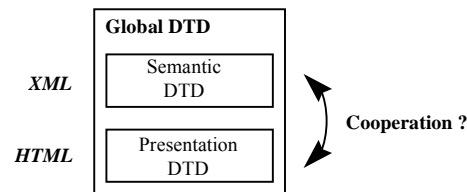


Figure 7 – A twofold XML DTD

This DTD architecture raises two kind of issues:

- at which level of the document instance shall be located data: either at the semantic part or at the presented part;
- what kind of protocol is to be used to coordinate the two worlds (presentation and semantics).

2.3.1 Data location

We want to use an existing HTML document as a user capture interface. It means that we assume that relevant data for parts family description already exist in the HTML document. Thus, these data are and remain at the same place: the presented part of the document. However, some data required, according to the EXPRESS model integrity constraints, to describe a parts family may be absent from the HTML document. Such data shall not be integrated in the presented part of the document to change the presentation during the capturing process. Thus, such data shall be recorded in the semantic part of the document.

2.3.2 Coordinating both content

Most of the data being in the presented part of the document, the semantic part of the document shall mainly contain references to these data. This is done by means of "pointers" that reference fragments of the presented part of the document, each fragment being delimited by a particular (and not styled) tag called an *anchor*, and being associated with an identifier that provides for referencing it.

The following example presents this approach on a very simple DTD. The semantic part (the part above the double line) contains only one concept: the concept of *unit*. The presented part (the part below the double line) consists of a set of *para* (paragraph).

<i>DTD</i>	<i>Document instance</i>	<i>Display</i>
<pre> <!ELEMENT unit - - def ref> <!ELEMENT def - - EMPTY> <!ATTRIBUTE def value CDATA #REQUIRED> <!ELEMENT ref - - EMPTY> <!ATTRIBUTE ref reference IDREF #REQUIRED> </pre>	<pre> <unit> </ref reference=1> </unit> ... </pre>	
<pre> <!ELEMENT anchor - - #PCDATA> <!ATTLIST anchor ident ID #REQUIRED> <!ELEMENT para - - anchor #PCDATA> </pre>	<pre> <para> d: nominal diameter in <anchor ident=1> mm </anchor> ; </para> </pre>	d: nominal diameter in mm;

Table 2 – Using a twofold DTD to decouple semantics and presentation

In the above table, the DTD description uses the XML syntax. A "tag" is defined through an *ELEMENT* declaration followed by a name. Then, after the two dashes, the allowed content (called in XML the *content model*) of the tag is defined. This description uses some meta-notations like '|' that stands for 'OR'. Some content models use XML reserved words. *EMPTY* defines an empty content model and *#PCDATA* defines a string content model. A tag may be associated with attributes (*ATTLIST*) referring to the tag. Each attribute is defined through a name, a data type and an occurrence indicator. The value of an attribute may be a string (*CDATA*). It may also be either an identifier (*ID*) or a reference to an identifier (*IDREF*) thus providing for reference between tags. Lastly, the occurrence indicator stipulates whether the attribute value is mandatory (*#REQUIRED*) or optional (*#IMPLIED*).

The semantic DTD defines a *unit* element, of which content model is either a definition (*def*) or a reference (*ref*):

- the *def* element content model is empty, but an attribute, called *value*, is associated with this element to be able to record a value when this value does not exist in the presented part of the document;
- the *ref* element is also empty, but it is associated with an attribute, called *reference*, intended to record a reference to a text area in the presented part of the document. In this case, the text area is tagged by the *anchor* tag.

Thus, each tagging action performed by the user on a document instance based on this DTD consists of parenthesizing a selected area of the presented part of the document through an anchor and to reference it from the semantic part, or of entering a value using the keyboard and to store it in the semantic part. Thus, the presentation is not changed during the capturing process, but the document is enriched semantically.

We discuss in the next section how the twofold DTD may be both associated with a capturing tool that integrates SGML/XML together with EXPRESS strength, and used to generate "active" documents.

3. Data capture process: capturing and using data

The main interest of the twofold DTD presented in the previous clause is that it gathers a semantic view on the data that reflects the content of an EXPRESS conceptual model, and a syntactical view of the data that provides for human legibility.

In this section, we present how we can make profit of this twofold structure:

- at the level of the capturing tool for checking the semantic soundness of the tagging process against the integrity constraints defined in the EXPRESS model;
- at the level of the usage of a document instance where the knowledge embedded in the semantic part of the document may be used to animate the document and to generate automatically a dynamic Web-ready catalogue providing a document oriented user interface for selecting parts.

3.1 The LITE-Cat tool

The LITE-Cat tool was developed in the context of an international cooperation project having involved Electricité de France (EDF), Toshiba Corporation (Japan), Itematic S.A. (France). The goal of this tool is not only to tag a document. It is also to check it against integrity constraints to make the result reliable enough to use it for exchange between databases or other computer applications.

3.1.1 Capture approach

The approach we propose for integrating document-oriented world (i.e., XML) and the database-oriented world (e.g., EXPRESS) is illustrated in Figure 8. A document, coming from a paper catalogue (in its electronic form, converted, if necessary, to HTML) is opened in an XML authoring tool based on the DTD presented in 2.2 and 2.3. The XML authoring tool has been enriched by a particular tagging control application that:

- offers a P-Lib oriented dialogue for capturing P-Lib concepts inside the document (user interaction control);
- automatically creates the tagging structure corresponding to the P-Lib concept within the twofold document instance (document structure control);
- activates an EXPRESS semantic checker, based on the P-Lib data model, to control the semantic soundness of the captured information (document semantic control).

Figure 8 shows an example of a possible user action. A document text area (*'NF-E-25-112'*) is defined as being the *code* (a P-Lib concept) of the parts family represented by the document. Then, the following process occurs:

- a *code* tagging request is sent to an EXPRESS semantic checker;
- the EXPRESS semantic checker triggers all the P-Lib data model integrity rules;
 - if some rules are violated, then the semantic checker forbids the tag to be inserted, and warns the operator to correct its selection;
 - if no rules are violated, the semantic checker authorizes the parts family *code* tagging structure to be inserted in the document instance.

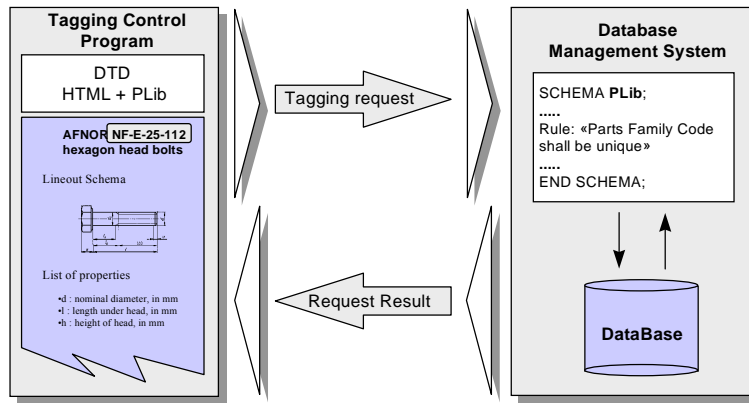


Figure 8 – Semantic control of tagging operation in a document instance

Thus, for any mandatory concepts of the P-Lib data model, a tagging process will occur if and only if it is semantically sound. We call the resulting document a *semantic document*:

- (1) it is properly structured, according to the P-Lib DTD, and
- (2) it is semantically sound, according to the P-Lib semantic model.

3.1.2 LITE-Cat user interface

Figure 9 presents a view of the user interface of the semantic capture tool, called LITE-Cat. It is an SGML/XML authoring tool, *Adept Editor* (from *Arbortext*), that has been interfaced with an EXPRESS model checker, *Ecco* (from *PDTec GmbH*), and customized to support the P-Lib capturing process.

This user interface contains:

- at the top of the editor: traditional authoring menus are available, more three other menus:
 - (1) *Dictionary*: contains the commands for capturing P-Lib data dictionary definition;
 - (2) *Content*: contains the commands for capturing P-Lib library specification;
 - (3) *Generation*: command for operating the captured catalogues (see 3.3);
- on the right side: an editing window that contains a document instance (the catalogue page) and that is the basis of all the interactions;
- on the left side: the document instance tree structure representation, based on the P-Lib DTD (this part of the window is usually hidden when the user is not familiar with SGML).

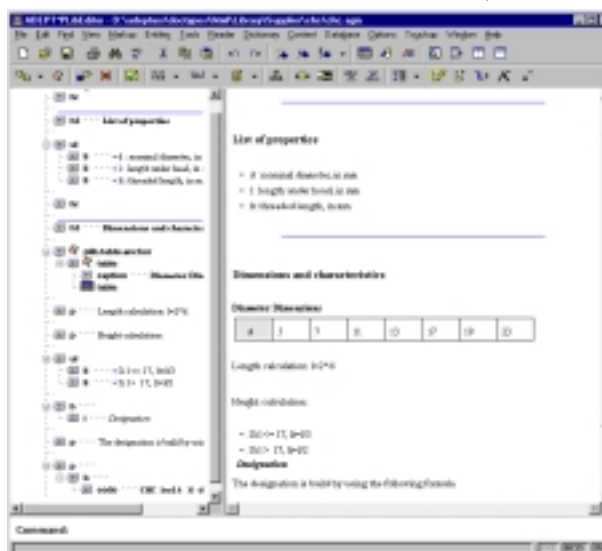


Figure 9 – Capture tool user interface

The capturing process takes place in the editing window. The user selects the capture context (e.g., a *property* description), by means of menus. Then, contextual menus enable to refine the capture context, e.g., tagging the *code* of a *property* description. Note that information may be captured either through mouse selections in the editing window when information is available in the input document, or through forms when information is not available in the input document.

3.2 Example of capturing process

Assume we would like to enrich, according to the P-Lib concepts, the catalogue page presented on the right part of Figure 9. It is a (very simplified) French standard that describes a *screw* parts family characterized by three main properties: *diameter*, *length* and *threaded length*. In this standard, some remarks, notes and a figure enrich the parts family description.

3.2.1 Document analysis

An analysis of this parts family in terms of P-Lib concepts suggests to model the various parts characteristics presented in Table 3. Note that all these properties don't appear in the document. For instance, the supported *load* is intended to facilitate the user selection. The information elements that describe the class and these properties constitute the dictionary description of the *CHC screw* parts family.

Besides, it appears in the document that the allowed values of properties are defined by means of tables or in expressions. Licit values of the *diameter* properties are defined in a table. Some (guarded) expressions define the values of the *length* and *threaded_length* properties when the *diameter* is known (there exists only one *length* and *threaded_length* for each *diameter*). Finally, some restrictions apply to the allowed *diameters* when the user specifies which *load* shall be supported.

<i>Type</i>	<i>CHC screw</i>
<i>Parts characteristics</i>	diameter (d)
	length (l)
	threaded_length (lt)
<i>Context Parameter</i>	load (lo)

Table 3 – CHC screw parts family properties

The document analysis is now finished. The LITE-Cat tool is to be used to capture this information while preserving the original document presentation.

3.2.2 Document enrichment

Firstly, the HTML document is opened in the LITE-Cat tool. As soon it is opened, the document is associated to the P-Lib DTD. The semantic part is empty, but nevertheless, the document instance complies with the twofold DTD. Then, a semi-automatic information extraction process occurs: the tool attempts to extract pertinent information (mainly tables and properties through table headers) in order to lighten the future manual enrichment process. This semi-automatic process (a dialog occurs between the operator and the tool) involves a twofold effect:

- the connected P-Lib EXPRESS database is updated each time an information is extracted;
- if it is a valid update (according to the instances checker), the document structure is dynamically enriched by the corresponding XML tag.

For instance, a table (called *diameter dimensions*) and its column header (corresponding to the *diameter* parts characteristics) are recognized within the HTML document. These automatic extractions are interactively enriched by the operator (for instance, the system asks what is the *unit* to be assigned to the *diameter* property). The database is updated and the corresponding XML structure is dynamically created within the document.

At the end of this process, the *length* property for example has not been extracted from the document, because it does not appear in a table header. Therefore, the operator will himself define completely this property through interactions with the system. The same is to be done for *threaded length* and *load* properties.

During all this interactive process, the operator can visualize the captured data in a third kind of window called the *P-Lib browser*. It is a simple navigation interface that enables to verify all the content of the semantic part of the document. The operator can navigate using hypertext links from the dictionary description of the parts family to the dictionary description of its characteristics properties descriptions, etc., and can act consequently.

After having completely defined the dictionary description of the parts family, the operator asks the semantic checker to check the global constraints on all the captured data. If no errors occur, the next step is to define the parts family population (i.e., domains and computation rules for selecting or computing properties values). This phase is also completely controlled by the capturing tool, and the same interactive process is used.

Thus, through a minimal number of interactions, the user may capture, i.e., make explicit, all the document content: domains, filters and derivation functions. Afterward, the document contains both implicitly in its HTML part, but also explicitly in its semantic part, all the concepts contained in the catalogue page. This explicit knowledge may be used to generate various results.

3.3 Operating the resulting P-Lib XML documents

Two main exploitations of the created semantic document may be done: generation of a file for database interchange and generation of a Web-ready server for selecting components.

3.3.1 Database oriented exploitation

Once P-Lib concepts are captured using the twofold DTD, the operator is able to automatically generate P-Lib exchange files, in the format defined by ISO 10303-21. These files describe the content of the whole set of families of parts described in the catalogue. Moreover, it is possible to refine this generation process to a particular branch of the parts families hierarchy, or to a single parts family. Thus, it is possible for a customer to record this description within its corporate parts library management system. Such a corporate database would contain in an integrated form not only the description of the parts families of the various corporate suppliers, but also the knowledge on the selection processes as defined by the supplier of each parts family.

3.3.2 Web oriented exploitation

All the knowledge contained in this semantic document (structure, selection methods, etc.) may also be automatically converted under the form of a program, as, for instance, a set of Java applets or scripts intended to be embedded within the presented part (i.e., HTML part) of the document. The main interest of this approach is the possibility to generate, without any manual programming actions, dynamic electronic catalogues.

Thus, these (enriched) HTML documents may be used as a user interface for accessing the parts library. Components access and selection is realized using a simple Internet browser. All the semantics is converted into a search engine integrated in the document as a Javascript program. The user may use these documents to select its components using all the knowledge embedded by the supplier. This approach is illustrated in Figure 10 that represents the dynamic document automatically generated from a static HTML supplier catalogue semantically enriched using the LITE-Cat tool.

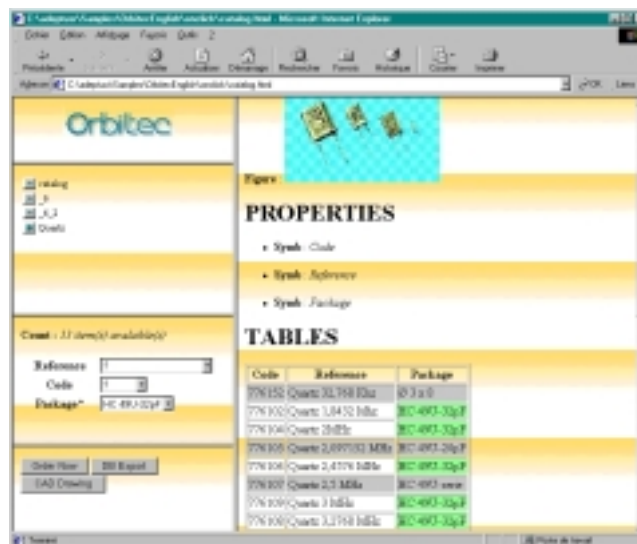


Figure 10 – An active HTML document

The active HTML document (that represents an electronic component parts family) is split by the generator into four main parts, all of them being generated. On the upper left part, the user may navigate (in client/server mode) in the parts family hierarchy. Below parts family hierarchy navigation window, an instance selection window is provided. The user may select some instances using forms: at each level of the parts hierarchy, only those properties that are available are provided to the user for selection purpose. It gives the possibility to make generic selections (at a generic level of the parts family hierarchy) or at the level of leaf family nodes. On the right part, the initial HTML document is presented, describing the parts family. But, starting from the knowledge captured in the semantic XML document, the document has been activated: properties values may be directly selected, via a mouse click, inside document tables. Then, only the available values may be further selected, etc. Note that forms selection and document selection interact mutually.

Finally, on the bottom left part, the user can directly order the part he has selected. He/she can download a database export (the physical file described in 3.3.1) of the given parts family, or even a particular representation of the parts family intended to be integrated in the user design environment, for instance in STEP or DXF format.

4. Conclusion

Over the last ten years, a twofold requirements emerged regarding parts and components catalogues. With the development of digital mock up in a number of industries, the need is to access to fully digital representations of all the parts intended to be used in new products. This is a data modeling problem that has been addressed since 1990 within the ISO committee that develops the P-Lib standard. This approach uses the EXPRESS information modeling language.

With the development of electronic documents and with the Internet, the need is to develop electronic versions of paper catalogues and to distribute them over the Internet. This appears as a document structuring problem for which XML seems to be the right approach.

In fact each approach has its strengths and weaknesses. The EXPRESS-based approach ensures the soundness of the modeled information, but it does not provide any synthetic means for browsing, presenting and understanding this information. As a result, it is difficult for parts suppliers to describe their own components using P-Lib. XML focuses on human legibility and on information presentation. Unfortunately, it is unable to assert any constraints on information elements. This makes the XML-structured information not enough reliable to be used by computer programs that should exploit the meaning of the structured information.

In this paper, we have developed a general approach that enables to integrate two different worlds: the data modeling world, through the EXPRESS language, and the structured documents world, through the XML language. The result is the definition of a XML DTD representing structurally all the pieces of information conveyed by the P-Lib EXPRESS information model. Then, using this DTD, we have presented how to use a components catalogue as a user interface for capturing P-Lib information: through an interactive tagging process, the document is semantically enriched whereas its presentation does not change. This is achieved by integrating in the P-Lib DTD a presentation part based on the HTML DTD.

Then, we have presented a tool architecture for capturing data in such a way. This architecture consists in interfacing an XML authoring tool based on the P-Lib DTD with an EXPRESS model checker based on the P-Lib data model. Thus, each information element contained in a catalogue is firstly associated to a P-Lib concept. Secondly, it is checked through the EXPRESS semantic checker. Then, if and only if no error occurs, the corresponding XML tagging structure is built inside the document. We have presented the LITE-Cat tool that uses this approach for capturing P-Lib data directly from parts suppliers catalogues.

The document resulting from this process is called a semantic document. It is both semantically checked and structurally organized. It may be used in different directions. The first one is the generation of P-Lib compliant physical files that provide for exchanging between databases based on the P-Lib model. The second one, Web oriented, consists in converting all the explicit P-Lib knowledge contained in the semantic document (e.g., domains of values, selection rules, etc.) under the form of a program (i.e., Java applets and/or scripts) that "operationalizes" this knowledge. The automatically generated result is a Web-ready (intelligent) catalogue that supports a user selection process. The result of this document-oriented selection may, in turn, be delivered by means of a P-Lib, or STEP, physical file intended to be recorded in the library management system, or in the product model, of the Web catalogue user.

This paper focuses on what XML may add to EXPRESS: data presentation and human legibility. We just outlined what EXPRESS may bring to XML: data integrity and data modeling capability. This concept of semantic document will be further discussed in a forthcoming paper.

References

- [Ait-Ameur and al. 95] Y. Ait-Ameur, G. Pierra, E. Sardet, *Using the EXPRESS language for metaprogramming*, Proceedings. of the 3rd International Conference. of EXPRESS User Group EUG'95, Grenoble 21-22 Oct. 1995.
- [Bazari 98] Z. Bazari Editor, *AP226 Ship Mechanical Systems*, Working draft of ISO 10303-226, WG3 N730, June 1998.
<http://www.nist.gov/sc4/step/parts/part226/wd2/>
- [Bray and al. 98a] T. Bray, et al, *Extensible Markup Language (XML) 1.0*, WWW Consortium 1998/02/10.
<Http://www.w3.org/TR/REC-xml>
- [IEC61360.4-97] IEC 61360-4: *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*, 1997.

- [ISO10303.11-94] ISO 10303-11, *Industrial automation systems: and integration — Product data representation and exchange – Part 11: Description Methods: The EXPRESS language reference manual*, 1994.
- [ISO10303.21-94] ISO 10303-21, *Industrial automation systems and integration — Product data representation and exchange – Part 21: Implementation methods: Clear Text Encoding of the Exchange Structure (Physical File)*, 1994.
- [ISO13584.42-98] ISO 13584-42, *Industrial automation systems and integration – Parts Library – Methodology for Structuring Parts Families*, ISO, Geneva, 1998.
- [Langlois 96] D. Langlois, *Functional data exchange conforming to STEP AP 221: Definition Classification for AP221 Objects*, ESPRIT Project # 20506 “PIPPIN”, Document 20506/ING/TNR/006/V02/22November1996, 1996.
[Http://www.lisi.ensma.fr/ftp/pub/PLUS/dictionary_ex/AP221_dict_V_01.b.doc](http://www.lisi.ensma.fr/ftp/pub/PLUS/dictionary_ex/AP221_dict_V_01.b.doc)
- [Meyer 88] B. Meyer, *Object oriented software construction*, Prentice Hall, ISBN:0-13-629049-3, 1988.
- [Pierra 94] G. Pierra, *Modelling classes of pre-existing components in a CIM perspective: the ISO13584/ENV 400014 Approach*, Revue internationale de CFAO et d’Infographie, vol. 9, n°3, 1994 , pp. 435-454, 1994.
- [Pierra 97] G. Pierra, *Intelligent electronic component catalogues for engineering and manufacturing*, Global Network Engineering 97, pp331 à 352, Antwerp, 1997.
- [Pierra and al. 98] G. Pierra, E. Sardet, J.C. Potier, G. Battier, J.C. Derouet, N. Willmann, A. Mahir, *Exchange of component data: the P-Lib (ISO 13584) model, standards and tools*, CALS Europe’98 ILCE’98, 9th International Conference and Exhibition on Enterprise Integration and CALS Europe, Paris 16-18 Sept. 1998.
- [Rumbaugh and al. 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, *Object Oriented Modelling and Design*, Prentice-Hall International Editions, ISBN 0-13-630054-5, 1991.
- [Sardet 99] E. Sardet, *Intégration des approches modélisation conceptuelle et structuration documentaire pour la saisie, la représentation, l’échange et l’exploitation d’informations. Application aux catalogues de composants industriels*, PhD thesis, October 1999.
- [Sardet et al. 97] E. Sardet, G. Pierra, *Formal specification, modeling and exchange of classes of components according to PLib. A case study*, Global Network Engineering 97, pp179 à 201, Antwerp, 1997.
- [Schenck and al. 94] D. Schenck, P. Wilson, *Information Modelling The EXPRESS Way*, Oxford University Press, 1994.