Integration of component descriptions in product data management systems

M. El-Hadj Mimoune
Y. Ait-Ameur, G. Pierra and J.C. Potier
LISI/ENSMA, BP 109, Téléport 2, F-86960 Futuroscope cedex, France
e-mails: {mimoune, yamine, pierra, potier} @ensma.fr

Abstract

This paper addresses two fundamental aspects of concurrent engineering processes. It suggests the integration of component and product data. On the one hand, the Parts Library standard, denoted P-Lib, has been described for exchanging part libraries between CAD systems, which use a big amount of heterogeneous data related to components or parts. Its data model allows the capability to exchange complex data between library management systems : data schemas, queries, constraints, domains and methods are described in this P-Lib data model. On the other hand, Product Data Management systems, denoted PDM, have been defined, in the last decade, in order to manage thousands of heterogeneous data (CAD/CAM data, BoM etc.*) related to products. Technical data management permits the management of the all-informational patrimony of a product starting from its design until its discharge.*

Products are essentially composed of pre-existing components stored in component or part libraries. This paper presents the approach we developed to integrate component data and product data. This approach has been applied with the particular standardised P-Lib data model for representing components and a particular PDM system namely SmarTeam. It will allow the automatically storage of component data in a PDM and the easy referencing of these data in the products during their whole lifecycle.

Keywords: Parts Library (PLib), PDM, data integration, meta-data, tracability, EXPRESS language.

1 Introduction

The growth, the heterogeneity, the security of data motivates the creation of information processing systems to facilitate heterogeneous products data management during their whole lifecycle (CAD-CAM Files, Quality folders, BoM etc.). These systems are named PDM (Product Data Management Systems). In a number of industrial fields, products are constituted with pre-existent components stored in component or parts libraries.

The capability to refer to component data from a PDM, in order to allow the use of PDM to represent and to manage product data in parallel with component data, permits to represent both product and component data in the same design environment. As a result of this referencing capability, the development and management costs would decrease.

Moreover, the possibility to refer to and to manipulate component knowledge and data as well as the reason for choosing this component in a given product facilitate the maintenance of those products when, for example failing components are to be replaced.

Two main approaches can be used for describing several data models. The first one, called multi-modelling approach, consists in describing several data models. Integration of these data models requires a translation from a data model to another. The second one is the meta-modelling approach where each data model is a particular instance of a data meta model.

The multi-modelling approach may be used for representing product data models in PDM (it is the one followed in the STEP standard for example in the Application protocols). The meta model approach has been followed by P-Lib standard for representing components (or parts library) data model.

Therefore, the integration of both approaches is a crucial issue when addressing both product data and component data. So, the goal of this paper is to present an approach to integrate both component and product data and knowledge in a common framework.

In the context of our approach, component data models are defined by means of instances of the P-LIB data model, developed in the Parts Library ISO-13584 standard. The P-LIB model is a formal data model allowing to represent and to exchange part libraries.

Product data are represented in a particular PDM. For implementation purposes, we have chosen the *SmarTeam* PDM, developed by the Smart Solutions Company. So, the goal of our approach is the representation of P-LIB data in a PDM.

The integration of these models requires the description of a common language allowing to represent them. For this purpose, we have used the EXPRESS data

modelling language and its graphical representation. This choice was motivated by the fact that this language allows the representation of all the knowledge categories related to both components and products.

So, our approach uses the EXPRESS language and its graphical representation as specification language for representing all the data models presented in this paper.

The approach we developed is a mixed (or hybrid) approach. Indeed it does not totally use a meta modelling nor a multi modelling approach. It uses the capabilities of both approaches and their ways of representation. It consists in representing the P-Lib data components in two different representation levels: the meta level representation for component families descriptions (component classes) and the hard encoded (or the direct representation) for representing data stored in tables. The first represents component families while the second represents the components themselves. So, some P-Lib descriptors are represented by the PDM objects (representation at the meta level) and the other entities representing tables are represented by the PDM tables (direct representation). This choice allows to use the capabilities of both the LMS (Library Management System) for selecting components and PDM for managing product data. Therefore, the PDM becomes capable to refer directly to the components data.

This paper is structured as follows. Next section presents an overview of the formal data modelling language EXPRESS and its associated graphical representation : EXPRESS-G. Section 3 gives an overview of P-Lib data models and its specifications. The overview of a PDM systems and description of a particular PDM (SmarTeam) is presented in section 4. Finally, last section presents the integration approach of the P-Lib data model in the selected PDM system.

2 Overview of the EXPRESS language

Knowledge modelling and automatic data processing have a great importance in different kinds of engineering knowledge. Several formal representations are used to formalise these knowledge models among them, we can cite the NIAM method, OMT[1], UML[2] etc. These formalisms were developed to allow a real world modelling (a universe of discourse) by semantic entities.

The use of CAD/CAM systems in engineering domains has begot an earnest problem, which concerns data exchange and sharing between contractors and subcontractors. This situation is due to their use of heterogeneous systems. Moreover, several formats were developed to allow this exchange. Among the different developed standards we can quote the STEP (STandard for the Exchange of Product data model) international standard which has developed the EXPRESS formal data modelling language (ISO 10303-11). The goal of this language definition is to allow data models description in order to exchange and to share data compliant with these models [3]. Moreover, it defines an exchange format for any data model specified in EXPRESS.

In the following, we will focus on the EXPRESS language constructs necessary to understand this paper. More details on the EXPRESS language can be found in [3] and [14].

2.1 General structure and EXPRESS concepts

EXPRESS is an object oriented data modelling language handling the important characteristics of object oriented language like abstraction, encapsulation, modularity and hierarchy [4].

In EXPRESS a data model is represented by a set of modules, called SCHEMAs. Each SCHEMA describes a set of entities, which represent the objects we want to model. Each entity is defined by a set of characteristics called attributes. Finally, each attribute has a data type where it takes its values. A SCHEMA may also contain a set of functions and procedures, which allow data coercion (integrity constraint) and derived attributes expressions (behavioural knowledge).

Entities represent, in the real world, objects that have physical existence (screw, bearing) or conceptual existences (address, course). In EXPRESS the entity represent a class of objects, which share common properties and behaviours. Inheritance between entities, representing structural knowledge, is allowed.

Attributes represent the properties which characterise an entity. They allow descriptive knowledge modelling. An attribute is defined by an identifier and a data type which can be simple (integer, real, character), structured (lists, sets, bags, arrays which are rigorously defined in EXPRESS), or user defined like in programming languages (C++; ADA etc.). Attributes can refer to entities (aggregation); i.e. attribute value can be an instance of this entity.

Three kinds of attributes are distinguished: Explicit attributes, Derived attributes and Inverse attributes.

Constraints: the EXPRESS language supplies the capability to describe constraint expressions. Constraints on data models are introduced by two different rules types: local rules which are applied on each entity instance separately , and global rules which are globally applied on several entity instances at the same time.

Local rules shall be checked against for each instance of a constrained entity. They are declared, at the same

time as the attributes. Local rules are applied on the value domains of set of attributes.

Inheritance: An EXPRESS data model is composed of a great number of entities, which are connected by father/son relationship (*is_a* relationship). If an entity A is a subtype of another entity B then A contains the properties of A more its own properties. A inherits all its parent properties entity.

In EXPRESS the inheritance can be simple, when an entity inherit only one entity, or multiple when an entity inherits several entities at the same time.

2.2 EXPRESS-G

EXPRESS-G is the graphical representation of textual EXPRESS data models. It allows synthetic presentation of textual EXPRESS data models, which are hard to read by human being. The graphical representation permits to make a partial representation of the full data model. Moreover, this formalism may be used to design a data model in preliminary modelling stages. EXPRESS-G allows structural and descriptive data representation with a graphical annotation. This format helps to have a global view of a data model. It should be noted that a graphical model increases readability and comprehensibility.

We have used the EXPRESS-G to model both the meta-model of P-Lib and the multi-representation model of PDM (SmarTeam) to confront the integration of their different approaches. So, the EXPRESS-G representation allows us to make a sort of interface between the two systems that will allow us to use components data (P-Lib) in PDM systems. We have chosen this formalism instead presenting program sources that we have developed.

Example

To illustrate the EXPRESS representation we will present, in the following, an example of a simple EXPRESS data model. We take the example of the geometry. Circle and point are both geometric entities. The entity Point has the attributes X, Y, Z representing co-ordinates and the inverse attribute *is_centre_of*, which allows expressing the following constraint: a point can be a centre to the maximum of two circles. The entity Circle has the following attributes: centre, radius and perimeter. The last one is a derived attribute.

The textual data model of this example can be represented as follow:

```
ENTITY Geometric_entities
SYPERTYPE OF (point, circle);
END_ENTITY;

ENTITY POINT
SUBTYPE OF (Geometric_entities);
X, Y, Z : REAL;
  INVERSE
Is_centre_of: SET [1:2] of circle for centre;
END_ENTITY;

ENTITY Circle;
SUBTYPE OF (Geometric_entities);
  Centre : Point;
  Radius : REAL;
 DERIVE
  Perimeter : PI*2.0*(SELF.Raduis)
END_ENTITY;
```

The representation of this example in EXPRESS-G is given on figure 1.

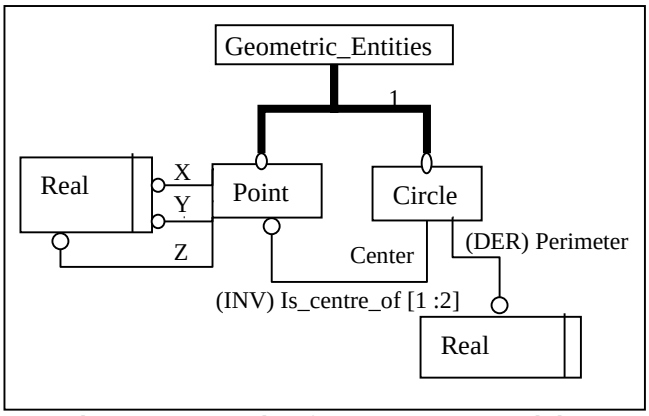
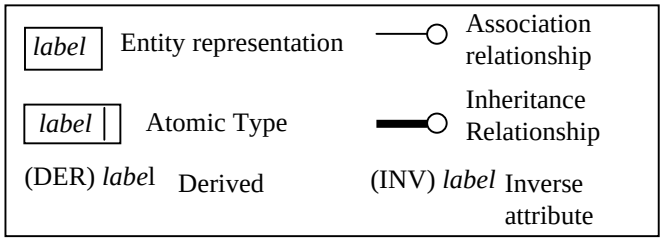


Figure 1 Example of EXPRESS-G model

Legend :



3 P-Lib : the parts libraries data model

The lifecycle of a product is a complex track. It requires different knowledge and involves several experts that work on the same product but with different perspectives and points of views.

The resulting process includes specification, design, manufacturing, and maintenance etc. [5]. Those activities are essentially informational processes and need to be handled by models, documents, computers etc.

The Computer Aided Engineering (CAE) area studies computer systems that allow to support these processes [6]. These computer systems supply efficient tools and mechanisms to represent different knowledge categories and to make it available for each expert in the suited format.

Ideally, these systems should allow each expert to work on his own perspective and should ensure co-ordination and interaction between the narrows perspectives.

Often, products to be designed are made of pre-existent technical objects [7]. It is the case for several domains of engineering such as electronic, mechanic etc. Hence, in such domain, a very significant part of design knowledge is related to components design. This knowledge corresponds to the capability to select a component, to evaluate its behaviours and to create different representations related to each discipline.

The goal of P-LIB-based digital catalogues is to allow exchange of component knowledge between suppliers and designers. These catalogues should be able to transmit both component characteristics and their representations. It also should include the various information element included in catalogues like figures, documents, tables etc.. Notice that, in practice, these catalogues are used by product designers to choose the components to be inserted in products.

In order to represent catalogues and component knowledge, the P-Lib data model has been described and standardised. The next sections give an overview of this data model. More details can be found in [8], [9], [10], [17]

data and store them in their own libraries. The data contain at the same time the component descriptions (general models) ant their representations (functional models).

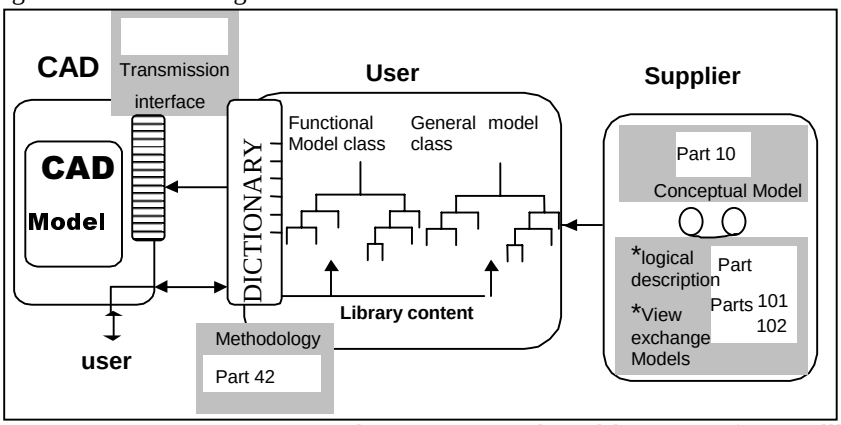


Figure 2 general Architecture of parts library

Library data are structured into classes according to the object-oriented paradigm. Three kinds of classes are considered in P-Lib:

General model classes enable library data suppliers to provide the definition of parts represented by a hierarchy of part family classes.

Functional model classes enable library data suppliers to provide various representations (e.g. Geometric, schematic, procurement data etc.) for these collections of parts.

Functional view classes enable the specification of the kind of representation provided in different functional model classes.

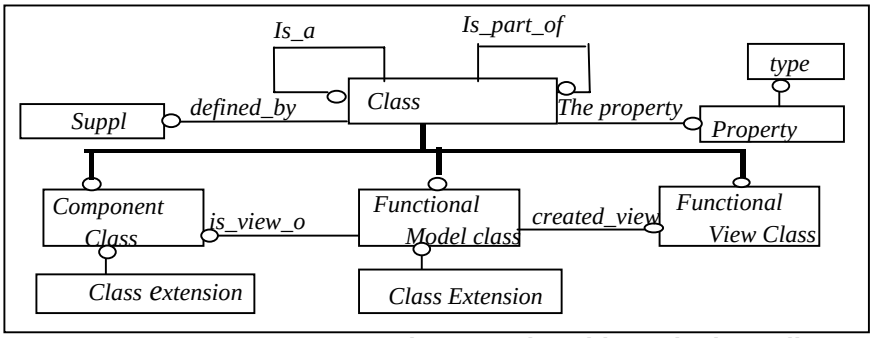


Figure 3: class hierarchy in P-Lib

3.1 P-Lib architecture

The general architecture of P-Lib is shown on figure 2. Suppliers describe components they supply within library (suppliers library) according to the formal P-Lib data model defined in EXPRESS and documented in the ISO 13584-24 and ISO13584-42 [8], [9]. Users recover these

The figure 3 summarises the description given above. It gives an overview of the whole P-Lib data model.

These classes shall be defined as a dictionary. It consists of a set of entries associated with a human-readable and computer-sensible representation of the meaning associated with each entity. This dictionary

provides a referencing mechanism between library data obtained from different suppliers and enables the user to obtain an understandable view of parts held in the library.

3.2 Fundamental principles

The international standard P-Lib is based on different principles [10]:

- It separates the representation of information held in a parts library from the implementation methods used in data exchange.
- It separates information about the structure of a parts library from the information relating to the different representations of each part or part family which belongs to the parts library.
- It uses a formal data specification language, EXPRESS to specify information about the structure of a library.
- It permits the information about the different representations of each part or family of parts within the library to be specified by different standards. The information is referenced within the information specifying the structure of the library.

3.3 P-Lib specification

The P-Lib standard was developed to allow designers to recover component data together with their different representations. A designer can, thus, choose component based on the properties supplied in P-Lib. The selection process consists of the following stages:

- the first stage is to choose a particular component family (component category), which corresponds to the desired functionality ,
- the object of the second stage is to choose, in the family, one or several accurate component instances which are adapted to the requirements expressed by the insertion context. Notice that this insertion context is usually a PDM or a CAD system.
- the end of this process is to choose a component well defined which we can precisely identify. This method of selection enables us to define a design problem to be solved. This process allows interchangeability in a maintenance context for example.

The users can access to suppliers' libraries across the interfaces, which exist between each system and P-Lib models. In this manner designers can use component data in their CAD or PDM systems.

As it was stated previously P-Lib uses the EXPRESS language to model and formalise component data. These data are stored in EXPRESS exchange format it provides and are encoded in ASCII files.

3.4 Conclusion

In conclusion we can say that the P-Lib standard has provided an approach and information models to facilitate components data exchange and their sharing between suppliers and designers systems. It allows exchanging digital component catalogues which integrate the supplier knowledge on the components he/she supplies (intelligent catalogues).

The P-Lib data model is a meta-model, which allows intentional and extensional description of component catalogues for their exchange between heterogeneous systems. This model characters both the structure and the characteristics of components (structural and descriptive knowledge), and the mathematical relations existing between different characteristics (procedural knowledge).

Use of P-Lib component data in CAD systems is possible using the interface which have been defined between P-Lib data and CAD systems. But such interfaces do not exist yet between P-Lib and PDM systems. This is the main motivation of our work.

Components are largely used in industry to design and manufacture new products. Companies use PDM systems (Product Data Management) to manage data related to the products they manufacture (CAD/CAM files, quality folders etc.). So, it is crucial to offer some capability to use component data defined in P-Lib in PDM systems.

4 PDM systems description

With the use of computers in more and more activities, a company needs to manage efficiently thousands of heterogeneous data created each year. These data are related to CAD/CAM files, Specification files, Numerical Command (NC) programs etc. Because of the big volume of information, of the various actors using this information and of the diversity of data processing tools, controlling this information generates several problems. Among these problems, we can cite:

- incoherence between various document and file versions used by different actors in development phases,
- delivery of incomplete manufacturing data for production,
- delay between product manufacturing and the delivery technical document.

To address these problems and subject to control the various technical information, involved in product development, in production, in marketing and in logistical support of industrial product, a new data management method was developed: the Product Data Management. This field is subject to an active standardisation work: STEP (STandard for the Exchange of Product model

data); CALS-CE that is an initiative of the American Department of Defence aiming at defining methods and tools to manage and exchange technical information related to weapon manufacturing.

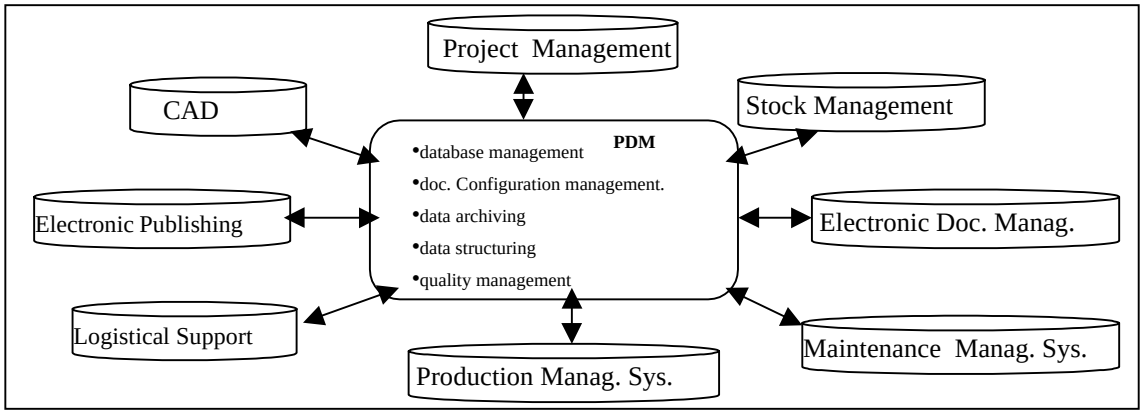


Figure 4: PDM systems interfacing with computer company systems.

4.1 Technical data

Technical data are data concerning product and process definition, which are used during the whole of product life. They include all the data that allow product description throughout its lifecycle. These data can be structured or not structured. They can have either electronic or hardcopy support. These data are necessary to identify and to describe state and configuration of a product, to manufacture a product, to control product evolution, to use and maintain the product.

These data are generated by different computer tools: CAD/CAM systems (Computer Aided Design/ manufacturing), CAE (Computer Aided Engineering), QFD (Quality Function Deployment) etc. and they are related to design, to engineering, to the manufacturing, to quality management, to the logistical support etc.

Technical data are associated with product during its whole lifecycle (starting from its design until its discharge). These data have particular characteristics. Each datum is associated to a particular computer system, which allows to interpret its content.

Lifecycle encompasses several phases. It includes the following: requirement definition, concept design, production, operation, maintenance and discharge.

4.2 PDM definition

A PDM is a system which allows to organise and to manage all the heterogeneous data described in the previous section. It also allows to give the right data to the right person and, to provide to everyone his/her own view of the project. A PDM manages technical data access, modification and sharing. It permits multiple access at the same datum and/or at the same time. It ensures interfacing between all computer systems used inside the company. The following figure shows existing interaction between PDM and various other systems.

Figure 4 shows how a PDM manages various data relating to a product. This management is done at two levels [12]. each set of product data (content) is recorded in a container associated with meta data. A PDM stores the content and manages the container.

Container management: supports the BoM (Bills of Material). It encompasses two functions: description of element components (Object ID, attributes etc.), and composition links that exist between different elements (component/composite relation). The composition link, between the component object and the composite object, is defined as another object.

Content storage: is a storage system with a great capacity: **the vault.** Vault content represents for example the drafting files, pictures digitised by a scanner, ASCII files etc. Visualisation of such object is not be possible without the engines that generated them and without PDM viewer. Without these engines we can't interpret their content. These objects are called BLOB (Binary Large Object). Objects, which are put in the vault, are controlled by the PDM. Any creation or modification is submitted to the validation by an ECO number (Engineering Change Order).

4.3 SmarTeam: description and structure

For our experimentation, we used a particular PDM system supplied by Smart solution called SmarTeam.

SmarTeam is founded on the principle of object-oriented databases.

We used this tool for study any feasibility of integration of P-Lib in a PDM. In the following, we will outline the structure of SmarTeam this will help us to

present the solution we propose to achieve this integration.

The structure of SmarTeam database is based on the concept of a project that represents the main class of the database. This global structure is constituted by a set of parallel class hierarchies where the first hierarchy represents the main classes. This structure permits to access data easily because they are generally bound to the instances of a main class and one can reach them through objects of this class (by the Browser or by the associated Application Programming Interface API).

4.3.1 Classes

In SmarTeam data are regrouped in several hierarchies of classes that represent categories of objects and then support simple inheritance. These classes are hierarchised to permit simple inheritance. In SmarTeam classes are classified in the following manner:

a) The main class: as stated above, the structure of data is organised around a hierarchy of main classes that plays a particular role in the database. It represents described product(s). Instances of the main class are displayed when SmarTeam is launched. This class has links with the other classes. Via these links, the user can reach the other objects directly.

Example: we can put the class *Projects* in the main class. Instances of this class can be motors, or gearbox for example. Others objects, such as parts of motors or gearboxes, will have a link with the instances of the class *Projects.*

b) Super-classes: they are the higher level classes in the hierarchies of which the first described is the main class. They are used to regroup classes in a specialisation hierarchy, and they constitute a separate tree. To every Super-class, it is necessary to associate indexes that allows the system to verify the uniqueness of objects in the class and that play the primary key role in a database.

Super-classes permit the construction of link classes between all classes that are in the hierarchy. The link between two classes is an object of the link class that binds their super-classes.

c) Leaf classes: Leaf Classes are classes of the lower level in all parallel hierarchies. In SmarTeam objects are only instances of leaf classes. The other classes are not instanciable (super-classes and intermediate classes).

d) Intermediate classes: intermediate classes are those classes that are between super-classes and leaf classes in the parallel class hierarchies. They permit common attribute factorisation. Both intermediate classes and super-classes are abstract classes.

e) Internal classes: In SmarTeam, predefined classes are used by the system to manage the database. For example the identification of users (their names, passwords etc.) is recorded in the internal class USERS. The other classes are used for the management of documents (identification of applications that are applied to each type of file to permit its visualisation for example).

f) Link classes: we distinguish two types of links in SmarTeam: aggregation link and association link.

Aggregation (or composition) link classes: aggregation is an association between a parent object and one or several son objects. This type of link is used typically to bind an object with its components. For example a wheel is composed of a tire, of an air room and of a rim. We can associate to the wheel these components through the use of the composition link. The composition link classes are created automatically for every super-class. These classes permit to stock aggregation links between the different objects (the link composite-component between two objects is an object of the composition link class).

Association (or logical) link classes: SmarTeam permits to bind between two classes (an instance of these classes) without taking into account their place in the hierarchy. We can make this link by the association link classes. For example, we can bind a screw CHc10 that is an instance of *screws* class with a document describing this screw.

g) Lookup table classes: they contain a list of strings and they are used only to define an enumerated type of data for an attribute that can only take its values in this list.

The general structure of the described classes is illustrated by the EXPRESS-G diagram of figure 5.

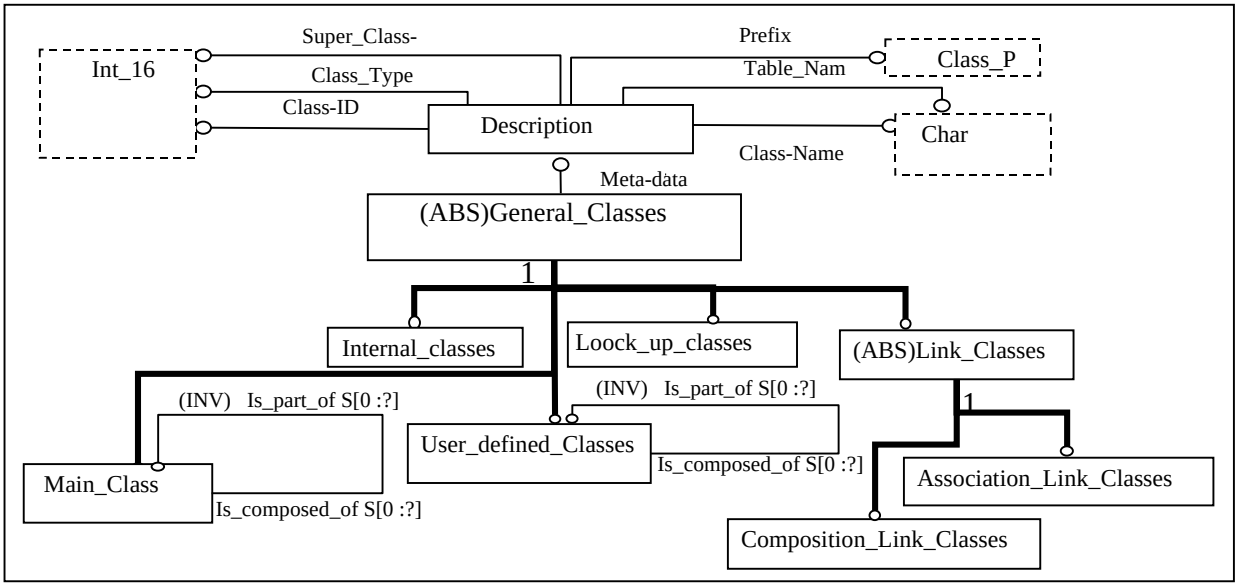


Figure 5 : general structure of classes in SmarTeam

All classes have some predefined common attributes called class attribute. They allow the definition and the characterisation of classes (class_name, class_ID etc.). They take the same values for all class instances.

4.3.2 attributes

In object oriented paradigm, a class has a number of attributes that permit to distinguish objects in one class. In SmarTeam, these attributes, themselves, have a certain number of properties that distinguishes them from some others attributes. Figure 6 illustrates properties of attributes in SmarTeam.

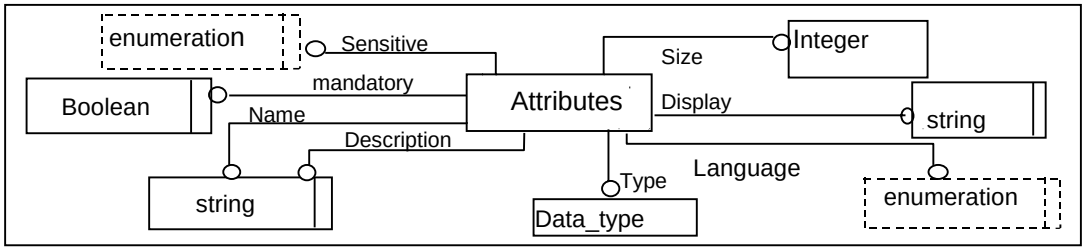


Figure 6: attribute properties

SmarTeam distinguishes four types of attributes

- class Attributes : are attributes that allows the definition of classes (CLASS_NAME, ID etc.) and they take the same values for every class instances.
- obligatory attributes: are predefined attributes assigned with each class and their values are managed by the system.
- mechanism Attributes : there are two mechanisms in SmarTeam: file management and revision management. The first one allows documentation management (in a file-managed class, each object is associated with a document and has file management attributes), the last one allows revision management and verification (register, checkin, checkout, approval etc.)
- user defined Attributes.

4.3.3 objects

Objects are instances of classes. They differ from each other by their object identification id (OID) and by their attribute values. For example, for classes Screw and Drawing, a Screw CHc10 will be an object of class Screw, its representation would be an object "Draw_screw", instance of the Drawing class, and link between these two objects would be an object of the corresponding link class.

Objects have attributes of the class to which they belong (predefined attributes and attributes defined by the user). Values of predefined attributes are assigned directly by the system. Object_ID value is assigned by the system and it takes an unique value used by the system to identify the object in the database.

4.4 PDM specifications

PDM systems are used to manage heterogeneous data, which describe enterprise products (CAD files, Production

Management System (PMS) data, BoM etc.). These data represent the static characteristics of products. Bill of Material (BoM) is the decomposition of a product in standard components purchased or in components manufactured in the enterprise from raw materials. It is well-known in mechanics and it is the point of departure of the PDM. In the PDM, at the physical components, the maps, the technical draws, the quality folders which allow the better description of the products, were appended. This decomposition is called METADATA. In the PDM the link between product and its components is an object of composition link class. So, both the OID of parent object (composite) and the child object (component) are stored in the link class. The same approach is used for the association links, which are used typically to link the physical components with their technical draws.

4.5 Conclusion

PDM systems are software developed in the last years to allow management of heterogeneous data, which describe products. These systems support products lifecycles including: requirement description, concept design, production, operation, logistical support, maintenance and discharge. They also permit quality management. This is very important to have ISO 9000 certificate to support concurrent engineering process. It is well known that using pre-existing components to design and to manufacture a new product is very useful. It allows to considerably decreasing prices of products. Pre-existing component data are recorded in a P-Lib's models manager. So, it is useful to reference directly from PDM component data. We will present the approach we developed to integrate P-Lib in a PDM system in the rest clause.

5 Integration approaches

5.1 Convergence and difference between PLib and PDMs

P-Lib describes component families by meta models. So, the data models, which hold the component (Physical files) are instances of the meta models. It has been defined so as to be easily extensible and to be very general. PDM systems describe products and manage BoM and documents which are related to these products. The modelling approach used in PDM is called the multi-modelling approach, because a specific model is developed for each BoM. Each software editor chooses a model which fits with its needs. In STEP a generic PDM data model has been defined from which everyone can derive its own specific data. So, the main difference between both approaches is due to the representation levels. Meanwhile, both approaches are intended to describe technical components and products. This is the reason which lead us to study integration of P-Lib in a PDM.

5.2 Aims of integration

The aim of integration of P-Lib in PDMs is specially to allow designers manufacturers and users to benefit from the suppliers' knowledge on components, as it is embedded in electronic catalogues. This knowledge, described by suppliers on components, must be available for use by designers, manufacturers and users and workable by constructors. So, this integration should allow storage of P-Lib component data in a PDM and should provide referencing those data in product data, therefore an efficient maintenance of products.

5.3 Integration approaches

a) Direct representation: in the PDM, data are stored in tables where each line represents an object of the class represented by this table and each column represents values of an attribute. One needs model not only table content but also table schema. Instead, in P-Lib another approach has been developed to represent tables because this approach requires that the table schema are defined before population such table. This approach consists in representing them by a list of lists representing each attribute in a column of a table. So, tables are represented by a number of EXPRESS entities. Moreover, in P-lib models, the classes representing families of components and their properties are also represented separately (meta-representation).

So, the direct representation consists in extracting component families from the meta model and in representing them by PDM objects. As stated before, component class properties are represented by separated entities in P-Lib. These entities contain a set of attributes that are common to all the instances. In SmarTeam class attributes are predefined, so to avoid the redundancy of data we have opted to represent the common attributes which describe a class by a description classes and to link the two classes. For example, for a Screw_family class we associate directly the object attributes: diameter, length etc. and then we create a class Screw_family_description that is associated with the Screw_family class. In SmarTeam dynamic creation of classes is not allowed, so this approach does not enable us to make an automated integration.

b) Meta levels representation approach: this approach consists in representing each P-Lib entity, such as it is defined in the EXPRESS model and represented in an exchange file, by PDM objects. Every P-Lib entity will be represented as a class in the PDM. For example, each of the entities *supplier, class, component class, functionnal view class,* etc. will be represented by PDM objects. The weakness of this approach is the impossibility to represent, in SmarTeam, aggregate data types (used to encode P-Lib tables). It is also the complexity to browse and to query such a meta model.

c) Hybrid or mixed representation approach: this approach consists in using both approaches in parallel. It allows us to benefit of the advantages of both approaches. We separate the representation of tables and the representation of classes. On the one hand, we represent tables that contain component attribute values by relational tables provided by the PDM (direct representation). On the other hand, classes such as *supplier, class, component class, functionnal view class,* and so on are represented by PDM objects (meta-representation). In the meantime link classes will be created to link objects with tables that contain their attribute values.

This approach has been implemented on the Smarteam PDM system and several examples have been processed.

6 Summary and Conclusions

In this paper we presented the approaches we developed to integrate P-Lib defined component cataloguesin in to PDM system. We have discussed the possible approaches and we have proposed to merge two approaches: the direct representation and the meta level representation. The resulting approach is named hybrid or mixed approach. It allows us to represent in the same model P-Lib descriptors, which characterise component families and suppliers, and P-Lib entities representing tables by PDM relational tables. The advantage of this approach is the possibility to make a completely automated integration of P-Lib defined catalogue and to keep the specificity of P-Lib data model, which contains a complete description of component families. Notice that no changes on the PDM system are required. As a result, this integration will allow the use of a PDM systems to manage at the same time component and product data. It allows the easy recovery of any component data as well as the knowledge on these components from its reference. It will also allow the integration of LMS's (Library Management System) and PDM's functionalities. This integration leads to increasing the productivity and to simplification of the component selection and their referencing within products. It makes also easier products maintenance by making possible the automatic exchange of component referenced when some components become e.g. obsoletes.

7 References

[1] J.Runbaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorensen, Object oriented modelling and design, Prentice-Hall International edition, 1991.

[2] I. Jacobson, G. Booch and J.Runbaugh, The unified Software development process, Addision-Wesley Eds, 1999.

[3] M. Bouazza, Le langage EXPRESS, Editions Hermès, 1995.

[4] G. Booch, Object Oriented Design, Redwood City, Calif: Bejamin/Cummings, 1991.

[5] G. Pierra, Intelligent electronic component catalogues for engineering and manufacturing, International symposium on global engineering networking Antwerp, Belgium, pp. 331-352, 1997.

[6] G. Pierra, Modelling classes of prexisting components in a CIM perspective: The ISO 13584/ENV 400014 approach, revue internationale de CFAO et d'Infographie, vol 9, pp. 435-454, 1994.

[7] J.M Moranne, conception assistée par ordinateur d'ensembles mécanique avec recherche d'une bonne solution: le logiciel SICAM, Proceedings of MICAD'86, Paris, Hermès, pp. 41-71, 1986.

[8] G. Pierra, Y. AIT-Ameur and E. Saedet, Parts library: Logical resource: Logical model of supplier library, ISO document: ISO/IS 13584-24,1999.

[9] G. Pierra, H. U. Wiedmer, description: methodology for structuring parts families, ISO document, ISO/IS 13584-42, 1997.

[10] P. Harrow, M. West, Parts library: overview and fundamental principles, ISO document, ISO/DIS 13584-1, 1997.

[11] T. Schreuber, B. Wielinga, J. Breuker, KADS: A Principled Approach to Knowledge-based System Development, Acadimic Press, London, Forthocoming, 1992.

[12] J.M Randoing, Les SGDT, Edition Hermès, 1995.

[13] M Maurino, La gestion des données techniques, Edition Masson, 1993.

[14] ISO 10303-11, Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual, 1994.

[15] F. Feru, C. Viel, Echanger avec le protocole d'application 203 de STEP: Echange et partage de données CAO et GDT, Aerospatiale & Goset, 1998.

[16] CIMdata, Product Data Management, http://CIMdata.com, December 11, 1998.

[17] E. Sardet, G. Pierra, Y. Ait-Ameur, Formal Specification, Modelling and Exchange of components according to P-Lib, A case study, International symposium on global engineering networking Antwerp, Belgium, pp. 179-200, 1997.