Modélisation du contenu des catalogues de composants industriels : de la représentation implicite à la représentation explicite.

Mourad El-Hadj Mimoune, Yamine AIT-AMEUR*,**

Guy PIERRA**

*Email: ** {pierra, mimoune}@ensma.fr*
Address: LISI-ENSMA
BP 40109 – Site du Futuroscope
86961 FUTUROSCOPE Cedex France
Tel / Fax: +33 5 49 49 80 64

*Email : * yamine@supaero.fr*
Address: ENSAE-SUPAERO
10 Av E. Belin. BP 4032
31055 Toulouse Cedex 4. France
Tel / Fax : +33 5 62 17 83 45

Résumé:

La norme PLIB est une norme internationale pour la représentation formelle et l'échange de bibliothèques de composants (ISO 13584). Dans les premières versions de PLIB, les données de composants étaient représentées exclusivement en suivant une approche implicite. Cette représentation est différente de celle utilisée dans les Systèmes de Gestion de Données Techniques (SGDT) pour représenter les objets techniques (ceci rendait difficile le couplage entre PLIB et les SGDT). Les versions les plus récentes de PLIB autorisent une représentation de composants en suivant une approche explicite. L'intérêt de cette représentation est à la fois de permettre une description plus simple des contenus de catalogues et de faciliter l'implémentation de gestionnaires de données PLIB. Du point de vue des SGDT, elle permet un référencement simple des composants, via l'identification des composants dans l'univers PLIB. Le processus permettant la conversion d'une représentation implicite dans une représentation explicite facilite donc l'intégration de données PLIB dans les SGDT. Cet article présente à la fois les extensions du modèle de données PLIB permettant de supporter la représentation explicite de composants et une approche pour la conversion des représentations des catalogues d'une forme implicite à une forme explicite, adaptée à la fois aux SGDT et aux bases de données relationnelles-objets.

Mots clés : échange de données, intégration de données, représentation implicite, représentation explicite, PLIB, base de données relationnelles-objets, SGDT.

Abstract:

The Parts Library data model, denoted PLIB, is an international standard that has been described for exchanging parts libraries between CAD systems (ISO 13584). In the previous versions of PLIB, component data have been represented solely in an implicit form. This representation, different from that one used in Product Data Management systems (PDM) to represent technical objects, makes renders difficult the linkage between PLIB and PDM systems. The last version of PLIB allows an explicit representation of parts. The aim of this representation is to allow and to facilitate representation and storage of component data in object-relational databases, and facilitate standard parts referencing in PDM systems.

The conversion of implicit representation into explicit representation facilitates the integration of PLIB in PDM systems. In this paper, we present, both the PLIB data model allowing explicit representation support and an approach to convert catalogue contents from this implicit representation into the explicit representation. The result is adapted at the same time to PDM systems and to object-relational databases. Moreover, It makes it easy to implement Library Management Systems (LMS).

Keywords : data exchange, data integration, implicit representation, explicit representation, PLIB, object-relational database, PDM.

1. Introduction

La norme PLIB est une norme internationale pour la représentation formelle et l'échange de bibliothèques de composants (ISO 13584). Dans ses premières versions, les données de composants étaient représentées exclusivement d'une manière implicite. La représentation implicite des données consiste à définir des familles de composants par un ensemble de propriétés décrites dans des tables élémentaires et par des expressions et des contraintes. Les instances existent seulement à l'issue du processus de sélection. En effet, le bibliothécaire PLIB (« Library Management System » ou « LMS ») calcule les valeurs des propriétés lors de la sélection d'un composant en évaluant les expressions, en testant les contraintes et en faisant éventuellement la construction de la table globale par jointure à partir des tables élémentaires. Le but de cette représentation est de réduire à leur minimum la taille des tables représentant les familles de composants. Celles-ci peuvent en effet être très volumineuses dans certains cas. Elle permet aussi la représentation des comportements des composants en liant la valeur d'une propriété au contexte dans lequel est inséré le composant. A contrario, ce modèle présente également des inconvénients. Le premier est la complexité du modèle de données de composants. Le deuxième est la difficulté de stocker les données dans des bases de données relationnelles-objet si on voulait les utiliser en tant que bibliothécaire. Enfin, le troisième est l'impossibilité de référencer des composants à partir d'un SGDT.
Dans de nombreux cas, et en particulier dans le domaine des composants électroniques actifs sur lesquels existent actuellement plusieurs projets, les composants sont rarement décrits par des comportements et des contextes d'insertion. Une description implicite s'avère alors inutile et coûteuse. Le développement récent d'une version simplifiée permet alors de représenter les composants d'une manière complètement explicite. Cette approche consiste à décrire les composants directement par énumération de couples (propriété, valeurs). Elle présente les avantages suivants :

- plus grande simplicité de représentation des contenus des catalogues et d'implémentation de systèmes de gestion de bibliothèque de composants;
- les instances décrites peuvent être référencées directement à partir d'un SGDT;
- les données peuvent être facilement stockées dans des bases de données.

La représentation explicite devrait faciliter l'utilisation de PLIB, en particulier lorsque les catalogues définis d'une façon implicite peuvent être convertis en format explicite.
Le but de cet article est de présenter le principe de la représentation explicite, de proposer une approche pour la conversion de la forme implicite à la forme explicite, et de montrer comment des catalogues, maintenant sous forme explicite, peuvent être gérés dans une base de données relationnelle ou à objets.
Cet article est structuré de la façon suivante. La section 2 présente le langage de modélisation de données EXPRESS utilisé pour définir le modèle de données PLIB. Nous décrivons dans la section 3 les principaux concepts de la norme internationale PLIB (ISO 13584). En particulier, Nous présentons, les deux méthodes de représentation des contenus des catalogues de composants : la représentation implicite et la représentation explicite. Les intérêts de chaque représentation, dans des domaines d'application spécifiques, sont également argumentés. Enfin, Dans la section 4, nous proposons une méthode de conversion de données représentées sous une forme implicite en données représentées sous une forme explicite.

2. Formalisation de modèles de données : EXPRESS

L'utilisation accrue de l'informatique dans les domaines techniques a créé un problème d'échange et de partage de données entre systèmes hétérogènes utilisés par les différents acteurs dans une entreprise étendue (ingénierie concurrente). La modélisation des données et des connaissances est très importante dans les différents types d'ingénierie. Plusieurs normes et standards nationaux et internationaux ont été développés pour pallier ce problème. Parmi tous les standards développés, on peut citer le standard STEP (STandard for the Exchange of Product data model)(ISO 10303). Ce standard, lancé au milieu des années 80, a pour objectifs, d'une part de modéliser d'une manière non ambiguë les données relatives aux produits dans toutes les étapes de leur cycle de vie et, d'autre part, de permettre leur interprétation par tout système informatique. Une nouvelle méthodologie de modélisation de données a été développée dans le contexte de STEP, pour assurer à la fois, leur indépendance de tout système et leur complétude, et augmenter leur richesse. La définition du langage EXPRESS [5] et du format d'échange et de stockage de données qui accompagne ce langage a augmenté l'efficacité de l'échange [16].
EXPRESS est un langage de modélisation et de spécification des données normalisé (ISO10303). Il a été défini principalement pour représenter les modèles de données dans les domaines techniques et est à présent largement utilisé pour la modélisation des données dans différents domaines. De plus, il peut être utilisé pour la spécification de plusieurs applications dans le domaine de développements informatiques [1].

2.1. Le langage EXPRESS

Un modèle de données EXPRESS [2] [5] est constitué d'un ensemble de modules appelés SCHEMAs. Chaque schéma décrit un ensemble d'entités qui représentent les objets de l'univers du discours. Un schéma contient

deux parties. La première est constituée d'un ensemble d'entités structurées selon une approche orientée objet avec héritage multiple. La seconde est constituée d'un ensemble de fonctions, procédures et règles globales permettant de décrire la partie procédurale du modèle de données.
Une entité est une représentation d'une catégorie d'objets. Une entité est définie par un ensemble d'attributs (qui peut être vide) décrivant ses caractéristiques. Chaque attribut possède un domaine (où il prend ses valeurs) défini par un type de données. Ce dernier peut être simple (tel qu'un entier, une chaîne de caractères …) ou bien un agrégat (telles que les listes, les ensembles …) ou encore défini par l'utilisateur en utilisant le mot clé *type* ou bien finalement une entité du modèle de données. Les attributs représentent les propriétés caractérisant les entités. Ils permettent la modélisation de la connaissance descriptive. Dans le langage EXPRESS, on distingue trois types d'attributs : explicites, dérivés, et inverses. Les attributs explicites sont utilisés pour représenter les propriétés qualifiées de fondamentales, i. g, les propriétés intelligibles et stables d'une catégorie d'objets d'un domaine particulier. Les attributs dérivés sont calculés à partir des attributs explicites ou d'autres attributs dérivés. Enfin, les attributs inverses permettent la représentation de la cardinalité de la relation réciproque ou inverse d'une relation donnée.
L'exemple de la figure suivante présente un modèle de données EXPRESS simple.

```
SCHEMA etablissement_schema;
ENTITY personne
ABSTRACT SUPERTYPE OF
       ONEOF (etudiant, enseignant);
              num_ss: INTEGER;
              nom: STRING;
END_ENTITY;

TYPE t_cours = ENUMERATION OF(
       math, info, histoire, sport);
END_TYPE;

ENTITY etudiant
       SUBTYPE OF personne;
       diplome : STRING;
       suit : LIST [3:3] of t_cours;
END_ENTITY;

ENTITY enseignant
       SUBTYPE OF personne;
       salaire : OPTIONAL INTEGER;
       enseigne : LIST [1:?] of t_cours;
END_ENTITY;
END_SCHEMA;
```

Figure 1– Un exemple de modèle de données EXPRESS

La hiérarchie d'entités est la suivante : les deux entités *étudiant* et *enseignant* sont des spécialisations d'une entité non instanciable (mot clé *Abstract*) *personne*. *T_cours* est un type énuméré qui représente les cours enseignés au sein de l'établissement. L'*étudian*t prépare un *diplôme* et suit exactement 3 *cours*. L'*enseignant* a un *salaire* et donne un certain nombre de cours. Le mot clé *Optional* signifie que l'attribut peut ne pas être valué.
Un format d'échange d'instances d'un modèle de données EXPRESS a été défini dans la partie 21 du standard STEP [16]. Cette spécification définit la manière de coder, sous forme d'un fichier de caractères, une population d'instances d'entités conformes à un modèle défini en EXPRESS.
Un exemple commenté d'instances du modèle de données que nous avons décrit dans la Figure 1est présenté ci-après :

```
ISO-10303-21;

HEADER;
FILE_DESCRIPTION(('ceci est un test','le fichier contient la description de
...'),'1')
FILE_NAME('nom','2000-08T15:12:30','M.El-Hadj Mimoune',
'LISI/ENSMA','preproc_version','systeme','autoris');
FILE_SCHEMA('etablissement_schema');
END_SEC;

DATA;
#1 = ETUDIANT ('1700975121457', /* num_ss, hérité de la classe personne*/
               'Dupont',             /* nom, hérité de la classe personne */
               'Maitrise',/*diplôme préparé*/
       (#5,#6,#7)) /*les cours suivis par l'étudiant*/
#2 = ETUDIANT ('2700286054018', 'Durand', ' Maitrise', (#5,#6..));
#3 = ENSEIGNANT ('1541211100004', 'Martin', $, (#5, #6));
```

```
#4 = ENSEIGNANT ('1600366015259', 'Dupont', $, (#6)));
#5 = COURS(.MATH.);
#6 = COURS(.INFO.);
..
END_SEC;
END-ISO-10303-21;
```

Figure 2– Un exemple de fichier physique

3. Le modèle de données PLIB

La norme PLIB (Parts Library) a été définie pour échanger des bibliothèques de composants (mécanique, électronique, …) entre systèmes CAO [11]. Ces systèmes manipulent un nombre important de données hétérogènes, autonomes et distribuées. Le modèle de données PLIB est un méta modèle de données permettant la description des données, des contraintes, des sélections, des classes et des méthodes s'appliquant à des familles de composants[1]. L'Utilisation de la méta modélisation permet de représenter les catalogues de composants. Ce méta modèle est complètement formalisé en EXPRESS.
La structure de données PLIB est basée sur le dictionnaire de données qui joue le rôle de référentiel pour les descriptions de familles de composants et de leurs instances. Il permet la définition de la structure d'une bibliothèque de composants. Il définit un ensemble de concepts d'une manière non ambiguë :
a) *Supplier* : élément de données décrivant les fournisseurs de produits ou de composants préexistants.
b) *Class* : une famille de composants. Les classes sont définies par le fournisseur et sont organisées en une hiérarchie de classes avec héritage simple.
c) *Property* : chaque classe est décrite par un ensemble de propriétés qui représentent la partie descriptive d'une famille de composants. Une propriété est décrite d'une manière précise (domaine de valeurs, unité de mesure …).
On distingue trois types de propriétés : les propriétés caractéristiques (elles représentent les propriétés intrinsèques des composants), les propriétés qui caractérisent le contexte, et enfin les propriétés qui dépendent du contexte qui peuvent être calculées à partir des deux autres en utilisant des fonctions de dérivation ou à partir des tables.
d) *Table* : contenant les valeurs des propriétés. Elles décrivent la base de données associée aux composants. Les tables sont représentées par des listes de listes. Cette représentation permet de définir ces tables avant même de décrire leur contenu [3]. Elles sont représentées par un ensemble d'entités EXPRESS.
e) *Document* : permet d'associer à une famille de composants des informations stockées dans des documents.
Les contraintes sont des expressions logiques permettant par exemple d'associer à certaines propriétés des domaines de valeurs conditionnels.

3.1. Représentation implicite de contenus de catalogues

Les composants sont regroupés en familles d'objets similaires dans lesquelles chaque instance est représentée implicitement à l'aide de tables élémentaires et d'expressions (contraintes et fonctions de dérivation). Ainsi, Elle permet de représenter les familles de composants ainsi que les comportements associés. Cette représentation implicite des composants facilite leur sélection et évite une explosion du nombre de données. Une représentation similaire synthétique est d'ailleurs souvent utilisée par les fournisseurs de composants pour représenter les composants dans les catalogues papier.

3.1.1. But de cette représentation

La représentation implicite permet de factoriser l'information et de réduire au minimum la taille des tables de définition des familles de composants. Elle permet aussi de représenter les comportements des composants dans leur contexte d'insertion. En fait, cela permet de représenter les familles de composants dont les propriétés dépendantes du contexte peuvent avoir des valeurs infinies. Nous pouvons citer l'exemple de la durée de vie d'un roulement qui dépend des charges axiales et radiales exercées sur celui-ci ainsi que de sa vitesse de rotation. Dans une représentation implicite, les instances de composants n'existent qu'à l'issue du processus de sélection.

3.1.2. Exemple

Dans cet exemple, nous présentons un modèle très simplifié de la représentation implicite de vis. La Figure 3 présente une description d'une famille de vis qu'on veut échanger en utilisant le modèle de données défini ci-dessous.

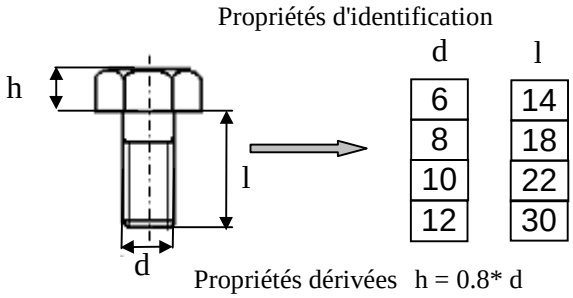

Figure 3– Description d'une famille de vis

Dans cette famille les conditions suivantes sont à remplir :

- La valeur de *d* dépend de celle de *ch* : si ch <= 12 alors *d*<= 8 et si 12.0<= *ch* < 25.0 alors *d*>8 (*ch* est un paramètre de contexte représentant la charge appliquée sur la vis).
- La valeur de *h* est calculée par l'expression suivante : *h* = 0.8 **d* .

Le modèle de données EXPRESS est représenté de la manière suivante :

```
ENTITY Vis;
        d: REAL;
        l: REAL;
     ch : REAL;-- charge appliquée sur la vis
DERIVE
        h: REAL:= 0.8*self.d;
WHERE
        WR1: SELF.d IN[6.0, 8.0, 10.0, 12.0];
        WR2: SELF.l IN [14.0,18.0,22.0,30.0];
        WR3: ((SELF.ch < 12.0) AND (SELF.d IN [6.0,  8.0])) OR ((12.0<=SELF.ch
< 25.0) AND (SELF.d IN [10.0, 12.0]);
END_ENTITY;
```

L'entité **vis** représente une famille de vis décrite par un ensemble de propriétés qui sont *d*, *l* et *h* et par un paramètre de contexte *ch* qui est la charge appliquée sur cette vis. Le mot clé *where* permet d'introduire des contraintes sur la classe. Les valeurs de *d* et *l* sont définies dans des listes et celle de *h* est calculée à partir de *d*. *Self* est un mot clé qui représente une instance de l'entité considérée. La règle WR3 permet de représenter les paramètres dépendant des contextes donnés par la condition mentionnée ci-dessus. On voit bien dans cet exemple que les instances des vis n'ont pas une existence réelle. Lors de la sélection et selon la valeur de *ch* fournie par l'utilisateur le système calcul l'ensemble des instances licites.

3.1.3. L'approche PLIB pour la représentation implicite

La représentation implicite consiste à représenter au niveau méta les différents éléments permettant de décrire les familles de composants. Un langage permettant de représenter les expressions génériques est défini pour décrire les tables, les expressions ainsi que les filtres.

- **Représentation des tables** : généralement l'ensemble des composants d'une famille est représenté dans des tables (réelles ou virtuelles). Dans ces tables, une ligne représente un composant (ou une instance) et une colonne représente une propriété. Cette table est dite "table de définition" d'une famille de composants. De telles tables peuvent être extrêmement volumineuses dans certains cas. Assez souvent, on les définit par un ensemble de tables élémentaires et un ensemble de règles de jointures permettant une construction de la table globale à partir des tables élémentaires. La Figure 4 illustre la représentation d'une table de définition d'une famille par un ensemble de tables élémentaires.
  Dans PLIB, les tables sont représentées par un ensemble de listes de valeurs. Chaque liste représente une colonne de table et est associée à une propriété d'une famille de composants. Les règles de jointure pour la construction des tables y sont définies.

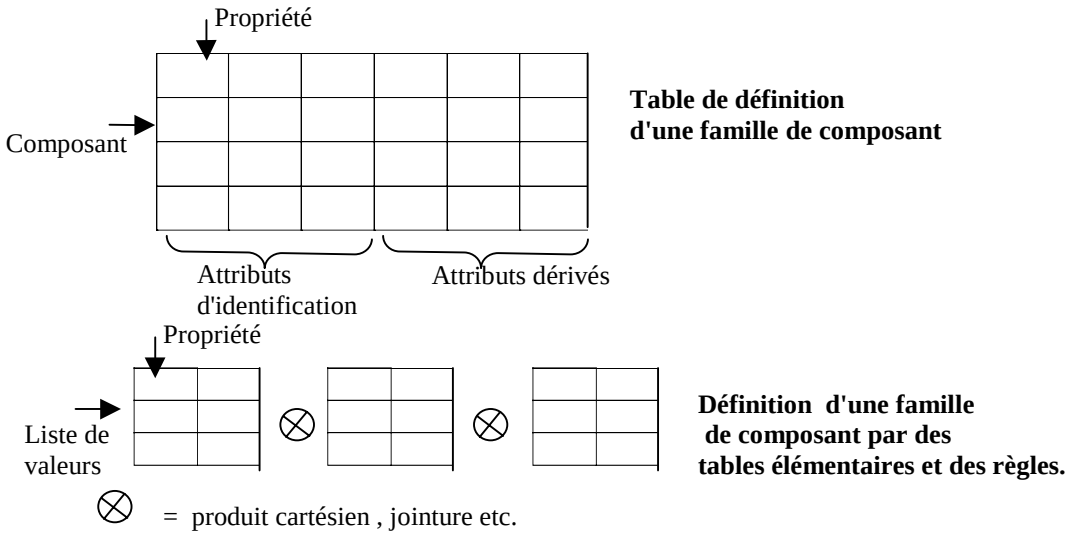

Figure 4– Représentation des tables de définition par des tables élémentaires [8].

- **Dérivation des propriétés** : à partir des propriétés dites d'identification, on peut déduire ou calculer d'autres propriétés dépendantes de celles-ci. Il y a deux types de dérivation : à partir de tables et à partir d'expressions.

1. dérivation à partir d'une table : assez souvent dans le domaine de la mécanique ou de l'électronique, plusieurs propriétés dépendent d'autres et sont généralement définies dans des tables. Certaines propriétés caractérisent le composant et sont les propriétés d'identification. D'autres dépendent de celles-ci mais elles sont listées dans des tables. Dans ce cas les règles de dérivation permettent de représenter cette dépendance même si chaque propriété est représentée dans des tables complètement séparées.
2. La dérivation à partir d'une expression : il existe aussi des propriétés calculées à partir d'expressions (propriété *h* dans l'exemple 4.2). Pour représenter ces propriétés, il suffit seulement de représenter la dépendance entre les propriétés ainsi que les expressions de dérivation permettant de les calculer.

- **Les filtres** : ce sont des expressions logiques permettant de restreindre l'ensemble des instances d'une famille de composants. Le regroupement de toutes les instances d'une famille de composants passe par le produit cartésien de toutes les tables élémentaires. Ce produit cartésien engendre certainement des valeurs ou des instances illicites. Les contraintes exprimées par les filtres devront être évaluées à vrai pour toutes les instances licites.

3.2. Représentation explicite de données de catalogues

Dans certains domaines, et notamment en électronique, les comportements de composants sont rarement représentés et les composants sont décrits par des valeurs explicites. Ainsi, les composants sont décrits par un ensemble de propriétés ayant un nombre de valeurs limité. De ce fait, une représentation implicite des données compliquerait la description des composants. De plus, plusieurs systèmes représentent explicitement les objets techniques (SGDT). Ce type de représentation facilitera l'intégration de données de composants dans de tels systèmes.
La représentation explicite consiste à décrire séparément chaque composant. Une famille de composants est constituée d'une liste de ses composants. En effet, chaque instance d'une famille de composants est représentée par une liste de couples (propriété, valeur).

3.2.1. But

Dans les SGDT, les données de produits sont représentées explicitement par une description en extension (énumération des composants) faisant référence à des composants préexistants pouvant être décrits au travers du modèle PLIB. Les niveaux de représentation et de modélisation dans les deux approches doivent être rapprochés. Le passage d'une représentation implicite à une représentation explicite devient une nécessité pour intégrer les deux types de représentation de données techniques (voir Figure 5). Cette représentation simplifiera le passage d'une base de données PLIB (fichiers physiques au format STEP) vers un SGDT ou des bases de données relationnelles-objets. Elle permet également d'utiliser ces dernières en tant que bibliothécaire de PLIB.

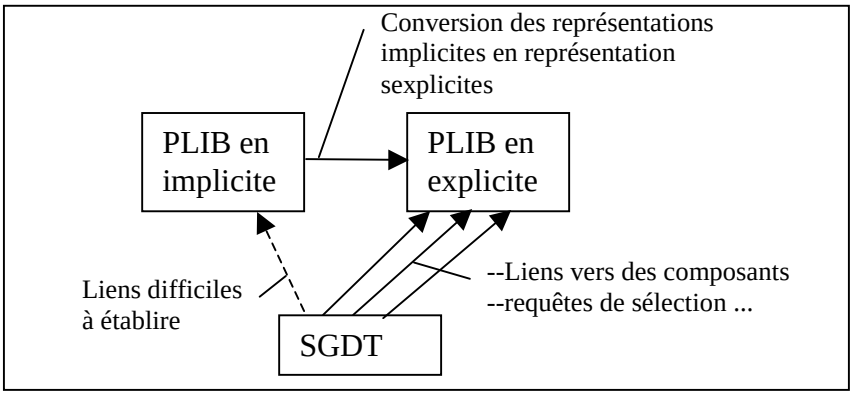

Figure 5– L'utilité de la représentation explicite

3.2.2. Exemple

Nous présentons dans ce qui suit un exemple simplifié du modèle de données explicite correspondant à la représentation implicite donnée dans l'exemple 3.1.2. Dans cet exemple, les paramètres de contexte ne sont pris en compte. Le modèle de données EXPRESS simplifié est le suivant :

```
ENTITY family;
       family_name  : STRING;
       components   : LIST [1:?] OF dic_part;
END_ENTITY;

ENTITY dic_part;
       family_ref   : family;
       properties   : LIST [1:?] OF property_value;
END_ENTITY;

ENTITY property_value;
       property_name: STRING;
       value        : REAL;
END_ENTITY;
```

Le modèle d'échange de données des vis présenté dans la Figure 3 est le suivant :

```
#1= family('vis', (#2,#5,#8));-- une famille de vis
#2= dic_part(#1, (#3,#4)); -- une instance particulière (une vis)
#3= property_value ('diametre', 6.0);
#4= property_value ('langueur', 14.0);
#5= dic_part(#1, (#6,#7));
#6= property_value ('diametre', 8.0);
#7= property_value ('langueur', 18.0);
#8= dic_part(#1, (#9,#10));
#9= property_value ('diametre', 10.0);
#10= property value ('langueur', 22.0);
```

Dans ce modèle, *family* représente une instance d'une famille de composants, en l'occurrence une vis. Elle est constituée d'un ensemble d'instances *dic_part*. Cette dernière référence la famille de composants dont elle est instance par l'attribut *family_ref*. Elle a un ensemble de *proprerty_value* qui sont des couples de types (propriété, valeur).

3.2.3. Conclusion

PLIB propose deux méthodes différentes pour la représentation des contenus de catalogues. La première méthode, dite représentation implicite, est fondée sur la définition d'un ensemble de tables élémentaires, de contraintes et de fonctions de dérivation, permettant de caractériser d'une manière implicite les composants. Elle permet une représentation riche et complète des composants. En fait, elle permet non seulement la représentation des caractéristiques des composants, mais également leur comportement dans des contextes d'insertion particuliers. De plus, elle permet de réduire la taille des tables de définition d'une famille qui peuvent être très volumineuses dans certains cas. Néanmoins, elle reste difficile à mettre en œuvre du fait de la complexité du modèle implicite et de la différence de niveau de représentation comparé à celui des SGDT.
La seconde méthode, dite représentation explicite, est basée sur la définition explicite de chaque composant, en énumérant toutes les valeurs de ces propriétés. En effet, dans plusieurs domaines, et notamment en électronique,

les comportements des composants sont définis par un ensemble très limité de valeurs des paramètres de contexte. Dans de tels cas, une représentation explicite est bénéfique. En fait, elle présente l'avantage de simplifier le modèle de données PLIB et de faciliter l'intégration de données de composants décrites dans le modèle PLIB dans un SGDT (mise en œuvre de PLIB). De plus, cela facilite la gestion de données de catalogues de composants par des bases de données relationnelles-objets.

4. Conversion d'une représentation implicite en une représentation explicite

Comme nous l'avons vu précédemment, la représentation explicite facilite l'intégration des données de composants dans les SGDT. Cependant, la représentation implicite s'impose toujours dans les domaines où les comportements des composants sont fortement présents et doivent être représentés.
Pour passer d'une représentation implicite à une représentation explicite on peut, à partir de familles de composants représentées implicitement, extraire les données de chaque composant.
Pour regrouper toutes les instances dans une seule table, des entités EXPRESS qui supportent cette table ont été créées. La table obtenue est constituée par un ensemble de colonnes *rich_column* qui sont des couples (propriété, valeurs) de cette propriété. Cette table permet de stocker temporairement les instances d'une famille de composants pour les transformer ensuite en modèle explicite. Il suffit de correspondre chaque ligne de cette table à une *dic_part* (voir exemple 5.3).

```
ENTITY rich_column;
     column_meaning : Property;
       its_column     : column;
END_ENTITY;
```

column_meaning : référence vers l'identifiant de la propriété d'une famille de composants dont les valeurs sont stockées dans *rich_column*. Elle correspond à un nom d'un champ dans une base de données relationnelle.
its_column : référence vers l'identifiant d'une colonne d'une table *column* déclarée comme une liste de valeurs et qui contient les valeurs de la propriété référencée par *column_meaning*. Les valeurs peuvent être de type entier, réel, booléen ou bien des instances des autres classes (équivalent à des clés étrangères dans une base de données relationnelle).

```
ENTITY full_class_instances;
     instances_of : class_bsu;
     content      : SET [1:?] OF rich_column;
     ID_columns : SET [1:?] OF rich_column;
 WHERE
     W1 : SELF.ID_columns <= SELF.content;
END_ENTITY;
```

full_class_instances est l'entité qui contient la table complète d'une famille de composants. Cette entité référence la classe dont elle contient les instances, les colonnes contenant les propriétés et leurs valeurs.
instance_of : représente une référence vers l'identifiant de la famille de composants dont la table contient les instances.
Content : est l'ensemble des colonnes constituant cette table.
ID_columns : est un sous ensemble des colonnes qui représente la clé de la table.

4.1. Processus de conversion

La conversion, d'une représentation implicite des composants vers une représentation explicite, passe par les étapes suivantes :

1. Calcul de la table explicite et complète contenant toutes les instances possibles de cette famille;
2. Mise en correspondance de chaque ligne de table (une instance de la classe) à une *dic_class* représentant cette instance.

Le calcul de la table associée à une classe passe par les étapes décrites dans les sections suivantes.

4.1.1. Domaine de définition

Le domaine de définition d'une famille de composants est le produit cartésien des domaines de toutes les propriétés d'identification de cette classe. Les propriétés d'identification jouent le rôle des clés primaires dans une base de données. Le produit cartésien de ces domaines donne le domaine complet d'une famille de composants.
Le produit cartésien est calculé à partir des domaines de définition de chaque propriété d'identification. Il faut tenir compte du cas où les domaines de deux ou plusieurs propriétés d'identification sont définies dans une seule table.

4.1.1.1 Filtrage du domaine de définition

Après avoir calculé le produit cartésien des domaines de définition des propriétés d'identification de la classe, le filtrage de la table résultante est effectué. Cette étape nous permet d'éliminer les valeurs illicites qui ne représentent aucune instance de la classe en question. Tout d'abord, les gardes (dans l'exemple 3.1.2, la garde est "si *ch*<= 12") puis les contraintes («*d*<= 8 » dans le même exemple) sont évaluées. Si les contraintes sont satisfaites, la ligne est conservée. Sinon, elle est supprimée. L'évaluation des contraintes nécessite l'évaluation d'expressions booléennes.

4.1.1.2 Jonction des propriétés dérivées

Le filtrage de données qui résultent du produit cartésien donne les valeurs que peut prendre une propriété appartenant à cette table. A partir de ces propriétés, les propriétés dérivées peuvent être calculées. Il existe deux types de propriétés dérivées : la dérivation à partir d'une table et la dérivation à partir d'une expression.

4.1.2. Evaluation des expressions

Une partie très importante de connaissance sur les composants concerne son comportement dans un contexte d'utilisation particulier. EXPRESS permet d'écrire des fonctions sur le modèle de données, mais ne permet pas d'échanger la description de ces fonctions comme des données. Pour pouvoir échanger des expressions comme des données une approche basée sur la méta-programmation a été développée dans PLIB [9] (ISO-13854-20).

4.1.2.1 Les expressions

D'un point de vue structurel, l'expression est modélisée par un graphe direct acyclique où les nœuds sont des opérateurs et les feuilles sont des variables ou des littéraux. Cette représentation est connue dans la théorie de la compilation sous le nom de "arbre abstrait de syntaxe". Il permet de modéliser n'importe quelle expression et de l'échanger dans un modèle d'échange de données conforme au format décrit par EXPRESS. la Figure 6 illustre un exemple simple d'une expression représentée en suivant cette approche.

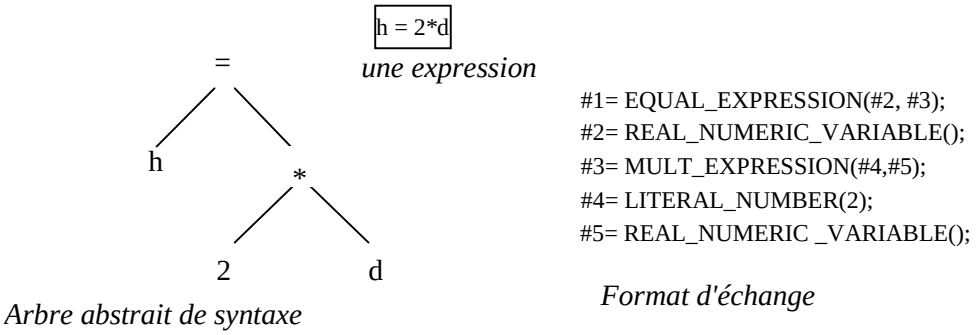

Figure 6 la modélisation des expressions par méta-programmation.

L'évaluation des expressions peut se faire par deux méthodes différentes :

4.1.2.2 Fonction d'évaluation

Cette méthode consiste à parcourir l'arbre de l'expression en évaluant les arguments puis l'opérateur de ces arguments. La fonction d'évaluation permet d'évaluer récursivement l'expression donnée en paramètre. Elle reçoit également comme paramètres une liste de variables et leurs valeurs. Elle est définie de la manière suivante :

```
Function evaluate_expression (ex : expression; variables: set of
property_value) : NUMBER ou BOLEAN;
```

Property_value est une entité qui fait correspondre à chaque propriété une valeur. Elle permet l'évaluation de variables.

4.1.2.3 Utilisation des attributs dérivés du langage EXPRESS

L'évaluation des expressions par attributs dérivés consiste à utiliser les attributs dérivés du langage EXPRESS pour évaluer les expressions [7]. Cette solution nécessite l'ajout des attributs dérivés aux expressions du modèle PLIB. Ce changement n'affecte pas le modèle d'échange car les attributs dérivés n'y apparaissent pas. De plus, cela offre un mécanisme d'évaluation simple et dynamique. Nous donnons ci-après un exemple d'entités du modèle PLIB modifiées par l'ajout d'attributs dérivés.

```
ENTITY expression
       ABSTRACT SUPERTYPE OF (ONEOF (numeric_expression,
boolean_expression));
END_ENTITY;
ENTITY numeric_expression
ABSTRACT SUPERTYPE OF (ONEOF (plus_expression,
div_expression, variable_or_constant))
       SUBTYPE OF(expression);
        operands : LIST[2:2] OF numeric_expression;
END_ENTITY;
ENTITY div_expression
       SUBTYPE OF (numeric_expression);
DERIVE
    the_value : NUMBER := SELF.operands[1].the_value DIV
SELF.operands[2].the_value;
END_ENTITY;
ENTITY variable_or_constant
       Abstract supertype of (vaiable, constant);
       SUBTYPE OF (numeric_expression);
END_ENTITY;
ENTITY constant
       Subtype of (numeric_expression);
       the_value : NUMBER;
END_ENTITY;
```

Dans ce modèle, les expressions sont organisées dans une hiérarchie de classes. Les entités *div_expression*, *plus_expression* et *variable_or_constant* sont des sous classes de *numeric_expression* et donc héritent de tous ses attributs. Une *numeric_expression* possède deux opérandes qui sont elles-mêmes des *numeric_expression*. Le résultat d'une expression peut être calculé directement par le système en évaluant l'attribut dérivé *the_value*. Un opérande peut être une autre expression ou bien une variable (par exemple *d* dans la figure 6) ou encore une constante. Le système évalue récursivement les expressions et rend une valeur.

4.2. Conclusion

La représentation implicite de catalogues de composants est nécessaire dans les cas où les comportements de composants sont fortement présents. De plus, plusieurs catalogues sont jusqu'alors représentés d'une manière implicite. La conversion des contenus de catalogues définis implicitement en contenus définis explicitement permet de bénéficier des avantages fournis par la représentation explicite, à savoir : le stockage des contenus de catalogues de composants dans des bases de données relationnelles-objets et le référencement à partir d'un SGDT des composants préexistants décrites dans PLIB.
Le stockage des données de composants dans des bases de données relationnelles-objets facilite la mise en œuvre de la norme PLIB. En effet, cela simplifie la création d'un système de gestion de bibliothèques basé sur des systèmes de gestion de bases de données relationnelles-objets préexistants, ce qui permet d'utiliser les fonctionnalités de ceux-ci pour gérer les données de composants. De plus, la sélection des composants devient plus simple ce qui permet de diminuer considérablement le temps de conception et facilite la maintenance des produits en cas de changement de pièces défaillantes par exemple.

5. Conclusion

Dans cet article les deux formes de représentation susceptibles d'être utilisées pour décrire un catalogue de composants ont été présentées. L'approche implicite est intéressante dans le cas où les comportements sont fortement représentés (dans le domaine de la mécanique par exemple). Par contre la mise en œuvre de cette représentation s'avère inutilement coûteuse lorsque, comme c'est le cas dans le domaine des composants actifs en électronique, les composants appartenant à chaque famille sont peu nombreux et sont décrits par des valeurs explicites de propriétés. L'intégration de PLIB dans des SGDT et des bases de données relationnelles-objets devient très difficile du fait de la différence des niveaux de représentation dans les deux modèles (PLIB et SGDT).
Les bibliothèques au format PLIB existant souvent sous format implicite, et étant appelées à continuer à être distribuées dans un certain nombre de cas sous cette forme, nous avons proposé, dans cet article, une approche permettant la conversion de la forme implicite à la forme explicite. Nous avons également montré que le résultat d'une telle transformation était bien adapté à l'intégration des composants décrits par PLIB dans l'univers des SGDT et au stockage des données dans des bases de données relationnelles-objets. Il est donc possible d'utiliser des systèmes préexistants en tant que gestionnaire de bibliothèques de composants au format PLIB. Le référencement des composants, décrits sous forme explicite, à partir d'un SGDT, devient également plus simple. Un tel référencement permet alors d'éviter la duplication des données qui sont déjà décrites dans les catalogues

de composants en se limitant à un lien ou une référence vers la bibliothèque PLIB à partir du SGDT. En outre, il facilite la maintenance des produits quand par exemple un échange de pièces défectueuses est envisagé.
L'adoption de cette nouvelle forme et la conversion des contenus de catalogues de composants d'une forme implicite à une forme explicite facilitera considérablement la mise en œuvre de PLIB en diminuant, le coût des systèmes de gestion. Cela devrait promouvoir un rapide développement de la mise en œuvre de cette norme, d'autant que cette forme simplifiée devrait s'avérer particulièrement adaptée aux besoins d'échange d'information techniques dans le cadre du commerce électronique.

6. Références

[1] Y. Aït-Ameur, G. Pierra, E. Sardet " An object approach to represent behavioural knowledge in heterogeneous information systems" International Conference on Object-Oriented Information Systems, London, pp. 315-339, 2000.

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

[3] M. El-Hadj Mimoune, Y. Ait-Ameur, G. Pierra et J.C. Potier, " Integration of component descriptions in product data management systems" ISPE International Conference on Concurrent Engineering "CE2000", LYON, pp. 370-379, 2000.

[4] K. R. Dittrich and A. Geppert, "Object-Oriented DBMS and Beyond", Conference on Current Trends in Theory and Practice of Informatics, pp. 275-294, 1997.

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

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

[7] A. Plantec, " Utilisation de la norme STEP pour la spécification et la mise en œuvre de générateurs de code", thèse, 1999.

[8] 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.

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

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

[11] G. Pierra, Y. Ait-Ameur and E. Sardet, "Parts library: Logical resource: Logical model of supplier library", ISO document: ISO/IS 13584-24,1999.

[12] G. Pierra, "Représentation et Echange de données techniques", Conférence aux Entretiens POLYMECA, Valenciennes, pp35-54,1999.

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

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

[15] T. Schreuber, B. Wielinga, J. Breuker, KADS: "A Principled Approach to Knowledge-based System Development", Academic Press, London, Forthcoming, 1992.

[16] 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.

[17] M. Bouzeghoub, G. Gardarin, P. Valduriez, "Les objets", Editions Eyrolles 1997.

[18] E. Sardet, G. Pierra, H. Murayama, Y. Oodake, Y. Ait-Ameur, "Simplified Representation of Parts Library : Model, Practice and Implementation", PDT Days 2001, Brussels. To be published (ISBN 1 901782 05 0).