Proc. of RPO-XAO, Montpellier 29-30 juin 1995, EC2 & C^ie ed., Paris, 1995, pp 165-175

Classes d'objets techniques et données de produit dans les systèmes d'I.A.O.

G. Pierra
Laboratoire d'Informatique Scientifique et Industrielle - ENSMA
Site du Futuroscope - B.P. 109 - 86960 FUTUROSCOPE Cédex
Tél. (33) 49-49-80-60 - Fax : (33) 49-49-80-64
e-mail pierra@ensma.Univ-poitiers.fr

Résumé

L'architecture logicielle usuelle des systèmes de XAO distingue le modèle de produit, les programmes d'applications, l'interface de dialogue et les bibliothèques de données. On étudie dans ce papier l'impact de l'approche objet, d'une part sur cette architecture générale et, d'autre part, sur la possibilité d'échange de bibliothèques d'objets techniques permettant de représenter la connaissance intrinsèque et préexistante sur des objets réels.

On montre qu'un processus de conception met en œuvre trois niveaux d'abstraction. Les niveaux classes et instances relèvent des nouvelles bibliothèques, orientées objets. Le niveau occurrence relève du modèle de produit. Cette séparation montre la faisabilité d'une intégration de l'approche objet dans le système de XAO sans remettre en cause l'autonomie du modèle de produit, condition nécessaire à l'échange entre systèmes hétérogènes.

La représentation souhaitable, pour un objet technique, est éminemment dépendante du point de vue de l'utilisateur. On montre alors que la technique de l'agrégat d'instances fournit une approche très ouverte pour intégrer les différentes perspectives, propre à chaque discipline, sur un objet technique. Concernant la modélisation de la connaissance intrinsèque, propre aux objets techniques préexistants, on propose trois mécanismes pour la représentation de la connaissance procédurale : les attributs dérivés, les contraintes d'intégrités prédicatives et les méthodes de génération de vue d'occurrence dans le modèle de produit. Ces trois mécanismes définissent une interface précise entre la bibliothèque orientée objet, d'une part, détentrice de la connaissance sur les composants, et le modèle de produit, d'autre part, orienté objet ou non, et détenteur de la connaissance sur le produit.

Abstract

The software architecture of CAx systems consists of a product data model, application programs, dialogue interface and libraries. We discuss the consequences of the object oriented approach, both on this architecture and on the exchange of component libraries intended to model the intrinsic component knowledge.

Three abstraction levels may be identified in a design process. The class and instance levels are modelled within the new object oriented libraries. The occurrence level corresponds to product data models. Such a structurization enables to integrate object oriented capabilities within a CAx system while insuring the feasibility of product data model exchange between heterogeneous systems.

In a design process, the representation needed by an expert of a particular discipline is heavily dependent on its discipline. We propose to model each technical object as a set of instances, each one corresponding to a different perspective on the same real world object. For modelling procedural knowledge on a component, we propose three mechanisms: attribute derivation, integrity constraints expressed through predicates, and representation creation method. These mechanisms specify the exchange protocol between object oriented components

libraries, that contain the component knowledge, and product modelling systems, object oriented or not, that contain product knowledge.

Mots clés : XAO, modèle de produit, bibliothèque de composants, multi-représentation, modélisation de connaissances techniques.

Keywords : CAD/CAM, CAx, product modelling, parts library, multi-perspective model, technical knowledge modelling.

Classes d'objets techniques et données de produit dans les systèmes d'IAO[1]

G. Pierra
LISI/ENSMA
Site du Futuroscope - B.P. 109 - 86960 FUTUROSCOPE Cédex
Tél. (33) 49-49-80-60 - Fax : (33) 49-49-80-64 - E-mail pierra@ensma.Univ-poitiers.fr

Introduction

Le processus de conception technique est le processus qui consiste à concevoir, à simuler, à fabriquer et à maintenir des produits matériels qui remplissent certaines spécifications. Ce processus est essentiellement un processus informationnel [Syvertsen 92], et des systèmes informatiques sont très largement utilisés pour le supporter. Le terme générique Ingénierie Assisté par Ordinateur (IAO) peut être utilisé pour désigner ces systèmes.

Comme beaucoup d'autres systèmes informatiques, les systèmes d'IAO se sont progressivement efforcés de séparer les données manipulées, des programmes qui les manipulent. Dans un système d'IAO, les données qui représentent le produit en cours d'étude constituent le *modèle du produit*. Plusieurs *applications* regroupent les programmes qui traitent ces données selon différentes perspectives. Enfin une *interface de dialogue* permet à l'opérateur d'activer ces programmes par l'intermédiaire de commandes, et de visualiser leurs résultats en termes d'évolution du modèle. Cette séparation permet à un utilisateur de développer ses propres applications destinées à effectuer des traitements correspondant à des besoins spécifiques. Elle permet également d'échanger des modèles de produit indépendamment des systèmes ayant permis de les concevoir.

Dans beaucoup de domaine de l'ingénierie, les produits techniques à concevoir sont essentiellement des assemblages d'objets techniques préexistants. C'est le cas en électronique, dans les systèmes de fluidique [Syvertsen 92], mais aussi en mécanique [Guillot 89]. Dans de tels domaines, les systèmes IAO doivent fournir des bibliothèques de représentations informatiques de ces objets techniques préexistants, de façon à éviter la nécessité de redéfinir le même objet de façon répétitive.

L'architecture classique d'un système d'IAO est ainsi constituée de quatre modules :

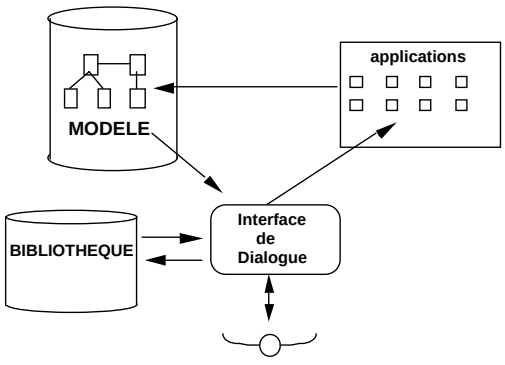

Figure 1: Architecture classique d'un système d'IAO

– Le modèle de produit contient les données qui décrivent le produit en cours d'étude,

[1] Le travail présenté dans ce papier a été pour partie financé par des aides du Ministère de l'Industrie (conventions 90.1.3134, 93.4.93.00.80). Il est actuellement partiellement financé par un projet ESPRIT (PLUS # 8984).

– La *bibliothèque contient des données susceptibles d'être insérées dans le modèle de* produit, pour représenter l'intégration d'un objet préexistant,
– Les applications contiennent les programmes susceptibles de créer modifier ou traiter les données du modèle, et, enfin
– L'interface de dialogue permet à l'opérateur de commander l'ensemble du système.

Au cours des dernières années, l'approche objet s'est progressivement étendue à tous les domaines de l'informatique. Pour modéliser les objets techniques, l'approche objet présente deux caractéristiques particulièrement intéressantes. D'une part, son concept de classe et d'instance permet de modéliser la connaissance structurelle sur un univers d'objets techniques, telle que l'existence de familles de composants [Pierra 90]. D'autre part, le regroupement de données et de programmes dans le concept d'instance permet de modéliser n'importe quelle catégorie de représentation, que celle-ci soit définie par de simples données statiques, ou qu'elle nécessite de décrire un comportement dynamique. Plusieurs systèmes d'IAO basés sur l'approche objet sont apparus récemment sur le marché [Beeker 89] [Touron 92].

Cette généralisation de l'approche objet pose le problème de l'architecture logicielle des systèmes d'IAO sous un jour nouveau. Suivi à la lettre, le principe de B. Meyer [Meyer 88] qui consiste à "mettre les programmes dans les données" reviendrait à remettre en cause la séparation, difficilement acquise au cours des années, entre modèle de produit et application de conception, et entre modèle de produit et classes d'objets prédéfinis. Le but de ce papier est de montrer qu'une telle remise en cause n'est pas inéluctable : l'approche object est compatible avec la séparation des données (actives) et des programmes qui les manipulent.

Dans la première section de ce papier, nous étudions les différents niveaux d'abstraction qui interviennent dans un processus de conception technique. Les niveaux classe et instance permettent de modéliser les objets techniques pré-existants. Le niveau occurrence permet de définir le modèle d'un nouveau produit. Dans la seconde section, nous montrons comment cette architecture à trois niveaux est compatible avec les besoins de multi-représentation nécessaire en conception technique. Dans la troisième section, nous proposons une représentation pour les classes d'objets techniques préexistants et leur différentes catégories de représentation. Enfin, dans la dernière section, nous montrons comment cette architecture est compatible avec l'échange entre systèmes.

1. Mécanismes d'abstraction dans la conception technique

1.1 Le niveau instance

Le concept d'objet technique est le premier mécanisme d'abstraction utilisé en conception technique. Un objet technique est une abstraction qui permet de représenter un ensemble (éventuellement inconnu) d'objets techniques considérés, d'un certain point de vue, comme étant interchangeables. Nous appelons une telle abstraction une *instance* d'objet technique. Les objets techniques peuvent être identifiés à différents niveaux d'abstraction, représentant de plus ou moins grands ensembles d'objets physiques. Par exemple, un roulement SKF 6200-ZNR est un objet technique qui représente un ensemble de composants fournisseurs interchangeables. Un roulement ISO 30 BC 02 XE est un objet technique plus abstrait (c'est-à-dire représentant un ensemble plus grand). Il peut être appelé un *composant abstrait*. Chaque objet technique est défini par une certaine entité (par exemple : SKF, ISO, ...). Cette entité choisit un nom, qui identifie cette abstraction, et définit un certain nombre de propriétés qui *caractérisent* tous les objets techniques qu'elle représente. Ces propriétés sont indépendantes du point de vue

adopté par l'utilisateur. Nous les qualifierons de *propriétés générales* et nous appellerons *modèle général* l'instance qui permet de les représenter.

1.2 Le niveau classe

Les objets techniques sont naturellement appréhendés à travers une hiérarchie de classes. Ceci constitue une organisation naturelle du savoir pour l'esprit humain [Coad 92] [Minsky 86]. Un débat actuel dans la communauté "approche objet" porte sur l'unicité de cette hiérarchie de classes. Chaque fois que nous adoptons une perspective différente sur un ensemble d'objets, une classification différente peut apparaître naturelle. Par exemple, les composants électroniques peuvent être classifiés selon leurs caractéristiques fonctionnelles, leurs caractéristiques technologiques, leurs caractéristiques de fabrication, etc. Nous prétendons que ce problème survient surtout chaque fois que l'on essaye de représenter, dans une même hiérarchie les points de vue correspondant à différentes disciplines : tous les catalogues de composants, tous les standards, tous les livres techniques sont organisés selon une hiérarchie simple. Modéliser le point de vue de chaque discipline à travers une hiérarchie différente, comme nous le proposons à la section 2, élimine ce problème du point de vue de la hiérarchie d'héritage qui doit être choisie. Distinguer les hiérarchies d'accès de la hiérarchie d'héritage, et rendre les hiérarchies d'accès adaptables par l'utilisateur, éliminent ce problème du point de vue des méthodes d'accès.

Nous appelons *classes de modèles généraux* les classes qui décrivent l'ensemble des modèles généraux représentant les objets techniques d'une même famille. Les objets techniques étant des abstractions, ces classes décrivent essentiellement les identifications de ces abstractions, ainsi que les propriétés générales et multi-perspectives qui leur sont associées par l'entité qui définit cette abstraction.

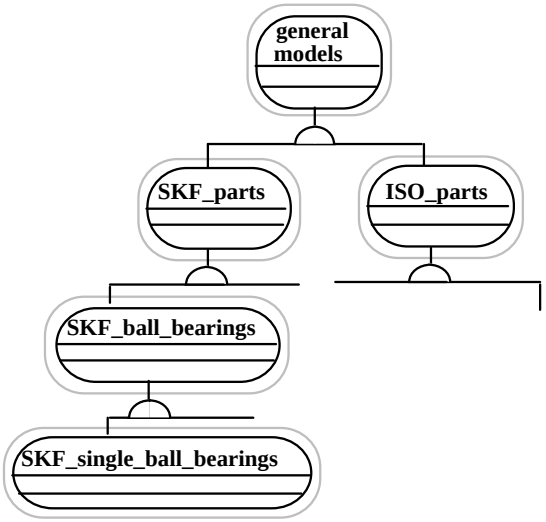

Figure 2 : Hiérarchie simple de classes de modèles généraux (formalisme de Coad et Yourdon [Coad 92])

Chaque classe définit les instances qui sont légales, c'est-à-dire celles qui correspondent à des objets matériels existants. Cette définition, en général assez complexe [Pierra 90], est effectuée sous forme de contraintes d'intégrité. Celles-ci doivent être assurées au niveau de la classe. La méthode de classe *new*, qui permet d'instancier une classe, a la charge d'assurer le respect de ces contraintes (et de guider l'utilisateur dans son choix). Une autre méthode, *change_objet*, doit permettre de modifier une instance tout en assurant le respect des contraintes d'intégrité.

1.3 Le niveau occurrence

Lorsqu'une instance est choisie, et insérée dans un modèle de produit, elle change profondément de nature. Tel roulement inséré à telle place dans tel produit n'est plus seulement le roulement de référence xxx. Nous appelons une telle entité une *occurrence*. D'une part, elle est porteuse de nouvelles informations. Certaines de ces informations, appelées *caractéristiques d'insertion*, caractérisent la position de l'occurrence dans l'espace de modélisation du produit auquel elle appartient (par exemple : un trièdre de référence, une matrice de positionnement, l'appartenance à un espace particulier ou à un niveau). D'autres informations, appelées *paramètres de contexte*, caractérisent le contexte d'insertion dans lequel l'instance est insérée (par exemple : la longueur compressée d'un ressort, la charge dynamique d'un roulement). D'autres enfin, appelés *attributs de représentation*, caractérisent la réaction de l'occurrence à ce contexte (par exemple : distance entre les spires du ressort, durée de vie du roulement). Un attribut de représentation est défini, au niveau instance, par une fonction. C'est une simple donnée au niveau de l'occurrence. D'autre part, sa représentation (informatique) elle-même est (ou peut être) profondément différente de celle de l'instance. La possibilité de faire intervenir de nouvelles classes d'objets, non prévues lors de la conception du système, suppose que toutes les occurrences aient la même structure, et donc qu'elles soient instances d'une même classe. Le fait que la même instance (exemple, dans un système d'ingénierie fluidique : un "coude" de dimensions données) donne éventuellement lieu à une multitude d'occurrences impose de ne pas dupliquer l'ensemble des données de l'instance au niveau de l'occurrence : une simple référence à l'instance suffit pour les représenter. Ce mécanisme existe d'ailleurs dans certains systèmes graphiques. Dans PHIGS, par exemple, une "invocation de structure" permet de représenter une occurrence et ses caractéristiques d'insertions.

2. Multi-représentation en conception technique

Le processus de conception technique implique de nombreux experts ayant des perspectives très différentes sur le produit. Les systèmes IAO utilisés doivent fournir des mécanismes efficaces, et des outils adaptés, pour permettre à chaque expert de travailler sur sa propre perspective, tout en assurant la coordination et le partage d'information entre ces différentes perspectives.

2.1 Les besoins de multi-représentation

Dans le domaine de l'approche objet, différentes interprétations ont été données à la notion de perspective [Carre 88] [Marino 90]. Deux d'entre elles s'avèrent particulièrement pertinentes pour le processus de conception technique.

La première est l'interprétation multi-expert. Une *catégorie de représentation* du monde est la perception qu'une catégorie d'experts appartenant à une discipline particulière a de ce monde. Chaque catégorie d'expert (par exemple : l'ingénieur de calcul, le thermicien, l'expert en analyse de la valeur) est intéressée seulement par une faible partie de l'ensemble de toutes les caractéristiques du produit. Différentes catégories de représentation (par exemple : la représentation géométrique, le modèle de simulation thermique, le modèle de simulation cinématique) doivent être fournies, et exploitées par le système IAO. Un modèle réellement multi-expert doit permettre à chaque expert de travailler avec une seule catégorie de représentation : il doit permettre à ce dernier de se

concentrer simplement sur la hiérarchie qui correspond à sa propre perspective, et, au niveau de chaque instance, sur les attributs qu'il considère comme pertinents [Marino 90].

La seconde interprétation correspond à la notion de multi-niveau. Lorsqu'il traite, par exemple, de la représentation géométrique d'un produit, le concepteur mécanique a besoin de différents *niveaux de représentation* géométrique du même objet technique selon le problème précis qu'il cherche à résoudre [Minsky 86], et la phase dans laquelle se trouve son processus de conception. Un simple trait d'axe est suffisant pour représenter une vis à certaines étapes de la conception d'une nouvelle machine. Un modèle solide de cette même vis peut s'avérer nécessaire pour l'analyse des collisions. Une représentation de type dessin technique doit être représentée dans le plan final. Les méthodes d'analyse dites structurées nous fournissent un grand nombre d'exemples de représentations multi-niveaux.

La différence entre catégorie et niveau de représentation est basée sur la notion d'interchangeabilité. Bien sûr, les catégories ne sont donc pas pré-définies, et leur choix dépend du concepteur du système. Par exemple, les liaisons cinématiques entre les composants d'un objet assemblé peuvent être représentées à l'intérieur de son modèle géométrique. Elles peuvent également être considérées comme constituant une catégorie spécifique de représentation. Mais, dans le dernier cas, échanger une représentation géométrique par une représentation cinématique dans la même représentation du modèle de produit n'a aucun sens. Au contraire, échanger une représentation géométrique simplifiée, telle qu'un trait d'axe, par une représentation géométrique plus complexe, ou même échanger un modèle géométrique solide par un ensemble de vues 2D, ou encore, échanger un maillage élément fini avec un pas fin par un maillage avec un pas plus large est, à la fois, possible et utile.

2.2 La notion de modèle (technique)

Même si cette approche est utilisée dans d'autre domaine de connaissance [Minsky 86], la mise en oeuvre systématique de "modèles (techniques)" est une des principales caractéristiques de l'approche utilisée dans les domaines scientifiques et techniques. Une ancienne définition, due à MINSKY, amène directement à la représentation multi-perspective que nous proposons : "un objet M est dit le modèle d'un objet A pour un observateur O si M aide O à répondre à certaines questions qu'il se pose sur A".

La première conséquence de cette définition est qu'un modèle ("technique") est un objet dynamique. Fournissant des réponses à un ensemble de questions connexes, il doit réunir à la fois des données et des processus. Cette réunion est précisément celle qui est réalisée dans le concept d'instance de l'approche objet. La seconde conséquence est qu'un modèle n'existe que par rapport aux questions auxquelles il vise répondre, et par rapport à l'objet technique dont il constitue une abstraction. Un modèle peut donc être caractérisé par les objets auxquels il peut être associés, et par les questions auxquelles il peut répondre. La troisième conséquence est qu'un modèle dépend de l'observateur ou, du point de vue de l'analyse que nous menons ici, du concepteur du système d'information. Des modèles très différents *peuvent répondre au même type de questions* et, des questions différentes peuvent être supportées par le même modèle. Le choix d'une typologie des modèles dépend donc du problème en cours d'analyse et constitue un choix de l'analyste. La dernière conséquence est que les modèles d'un objet technique préexistent à toute occurrence de cet objet technique dans un quelconque modèle de produit. C'est un mécanisme d'abstraction qui est défini au niveau des instances, et dont le rôle principal est de fournir des réponses concernant les occurrences de cette instance. Un modèle technique représente ce que l'on peut qualifier de "component knowledge",

complémentaire du "product knowledge" utilisé par le concepteur pour définir le nouveau modèle d'un produit [Paasiala 93].

2.3 Niveau d'abstraction dans les différentes représentations

Les trois niveaux d'abstractions identifiés pour la modélisation des objets techniques s'appliquent à l'identique pour la modélisation de chacune de leur représentation. La représentation d'une instance est un objet dynamique. Il "répond à un ensemble de questions" sur l'objet technique. Il s'adapte, en particulier, à n'importe quelle caractéristique d'insertion. Nous appellerons une telle représentation (d'instance) un *modèle fonctionnel*. La représentation d'une occurrence est beaucoup plus statique. Elle porte les réponses de l'instance à des caractéristiques d'insertion définies (par le contexte dans lequel elle est insérée dans le modèle de produit). Nous appellerons une telle représentation (d'occurrence) une *vue fonctionnelle*.

Comme dans le cas de la représentation de l'objet technique lui-même, une vue fonctionnelle a (ou peut avoir) une représentation très profondément différente de celle d'un modèle fonctionnel. L'exemple le plus simple est celui de la représentation géométrique. Le modèle de géométrie de chaque instance d'objet technique, qu'il soit prédéfini (par exemple : un cercle ou un point), ou qu'il soit défini par une bibliothèque externe de classes (par exemple : un roulement), est essentiellement un objet dynamique (un programme ou un modèle paramétré). Les représentations géométriques d'occurrence sont créées par ce modèle et sont en général constituées d'un ensemble de données (un ensemble d'entités géométriques).

Enfin, pour chaque catégorie de représentation, les modèles fonctionnels correspondant aux objets techniques représentés par une même classe de modèles généraux sont naturellement regroupés dans une classe unique : les services qu'ils fournissent peuvent aisément être paramétrés par les attributs de l'instance auxquels ils correspondent, et donc être factorisés au niveau de la classe. Ainsi le modèle paramétré, ou le programme paramétrique, susceptible de créer les géométries des différents vérins d'une famille n'ont pas besoin d'être dupliqués pour chaque instance de vérin. Une unique description, paramétrée à la fois par les caractéristiques du vérin (instance), et le paramètre de contexte que constitue sa longueur en position (occurrence), permet de générer l'ensemble des vues fonctionnelles de l'ensemble des vérins de la famille.

3. Modélisation de la connaissance sur les objets techniques pré-existants ("component knowledge")

3.1 Propriétés générales des composants

La modélisation des familles de composants par des hiérarchies de classes avec héritage simple permet de représenter la dimension structurelle de la connaissance sur un univers d'objets techniques. Elle correspond à la façon dont cette connaissance est décrite dans les normes et les catalogues. A chaque classe sont associés des attributs qui permettent de représenter la connaissance descriptive. La principale difficulté est au niveau de la connaissance procédurale (la longueur d'un ressort dépend de la force qu'il subit) et des contraintes d'intégrité (toutes les dimensions de diamètre et de longueur ne définissent pas des vis "réelles").

Cette modélisation doit permettre non seulement l'expression de connaissance procédurale mais également définir ses mécanismes, explicite ou implicite, d'activation.

Ce problème correspond à celui adressé par les bases de données dites "actives" [Hudson 86] [Chakravarthy 89]. Pour lesquelles plusieurs approches ont été proposées que l'on peut regrouper en approche à activation explicite et approche à activation implicite.

Dans les approches à activation explicite, dont la méthode Evènement-Condition-Action (ECA), proposée par [Chkravarthy 89] est un exemple, les conditions d'activation de la connaissance procédurale sont modélisées de façon explicite. Toute connaissance procédurale est modélisée sous forme d'une règle dont la partie spécification définit la condition d'activation (l'événement) ainsi qu'une garde qui filtre les événements devant effectivement donner lieu à activation (la condition) et dont le contenu constitue l'action. Face au processus, non déterministe, de sélection d'un composant dont les valeurs d'attributs doivent respecter un certain nombre de contraintes globales d'intégrité, l'approche E-C-A nécessite d'expliciter tous les cas où une règle globale devra être activée. Ainsi, si trois attributs sont liés par une relation, trois règles différentes devront être associées à chacun d'entre eux. Chaque règle, repésentant en fait la même relation, exprimera que cette relation doit être respectée quand l'un quelconque des trois attributs est valué.

Cette complexité de mise en œuvre de l'approche ECA face à un processus d'accès aux données non déterministe résulte de l'approche impérative adoptée pour décrire les conditions d'activation de la connaissance procédurale. Il est clair que les trois règles évoquées précédemment peuvent se déduire automatiquement du seul schéma de la relation. Ce point de vue est celui des approches à activation implicite, basées sur les Invariants de données [Gal 95], également adopté par le langage EXPRESS [Schenck 94]. Dans ce type d'approche, que nous avons proposé de suivre pour décrire les propriétés générales des classes de composants et qui a été retenue dans [ISO 95c], la dimension procédurale de la connaissance sur un objet technique peut se modéliser de deux façons :
– par des attributs dits "dérivés", dont les valeurs se spécifient à partir des attributs explicites et des paramètres de contexte éventuels, par l'intermédiaire d'une fonction : *la fonction de dérivation* ;
– par des prédicats, qui définissent à la fois les instances licites de la classe, et les domaines autorisés pour les paramètres de contexte : ce sont *les contraintes d'intégrité*.

La figure ci-dessous donne, en EXPRESS [Schenck 94], un exemple (très simplifié) d'une famille de roulements.

```
ENTITY single_ball_bearing
SUBTYPE OF ...;
     dynamic_load   : REAL;           -- parametre de contexte
     inner_diameter : INTEGER;        -- attribut de composant
     outer_diameter : INTEGER;        -- attribut de composant
DERIVE
     width : REAL                     -- attribut (derivé) du composant
           := compute_width (SELF.inner_diameter);
     life_time : REAL                 -- attribut de representation
           := compute_life_time (SELF.inner_diameter, SELF.outer_diameter,
                                 SELF.dynamic_load);
WHERE
     allowed_inner : (SELF.inner_diameter IN [10, 11, 12, 15, 20])
     allowed_outer : (SELF.outer_diameter IN [30, 40]);
     allowed_load : (SELF.dynamic_load > 0)
                   AND (SELF.dynamic_load < 2500);
     global_constraint : SELF.outer_diameter>=SELF.inner_diameter+10;
END_ENTITY;
```

Fig. 3 : Un exemple (très simplifié) d'une classe d'objets techniques

Ces deux constructions permettent de représenter deux types de connaissances. D'une part, la connaissance fonctionnelle sur un objet technique, à condition que celle-ci puisse se formaliser en termes d'évaluation d'une propriété descriptive de l'objet, que cette propriété soit une propriété caractéristique de l'objet (niveau instance, exemple : width) ou qu'elle soit caractéristique à la fois de l'objet et de son contexte d'insertion (niveau occurrence, exemple : life_time). D'autre part, toute connaissance assertionnelle pour autant que celle-ci s'exprime en terme des attributs explicites ou dérivés de l'objet.

3.2 Modélisation des différentes catégories de représentation

Dans un très grand nombre de cas, le processus de conception technique consiste à concevoir un nouveau produit à partir d'objets préexistants plus ou moins abstraits. Ces objets préexistants sont choisis par le concepteur, puis insérés, en tant qu'occurrence, dans le modèle du produit en cours de conception .

Lorsque l'objet technique est choisi, la principale question que se pose le concepteur qui utilise le système IAO est de créer des représentations de l'occurrence de cette instance dans le modèle du produit en cours de conception. Il en résulte que les questions auxquelles un modèle doit apporter une réponse peuvent être formalisées par le *nom de la catégorie de représentation* et éventuellement le *niveau de représentation* qu'il a la charge de fournir pour un objet technique bien défini.

La réponse à cette question consiste précisément à créer, dans le modèle de produit, la *vue* (d'occurrence). Cette vue (1) se rapporte à une occurrence d'objet technique, (2) elle est caractérisée par la question à laquelle elle apporte une réponse, et, (3) elle est souvent constituée par un ensemble de données, avec, éventuellement, une référence au modèle qui a permis de la créer.

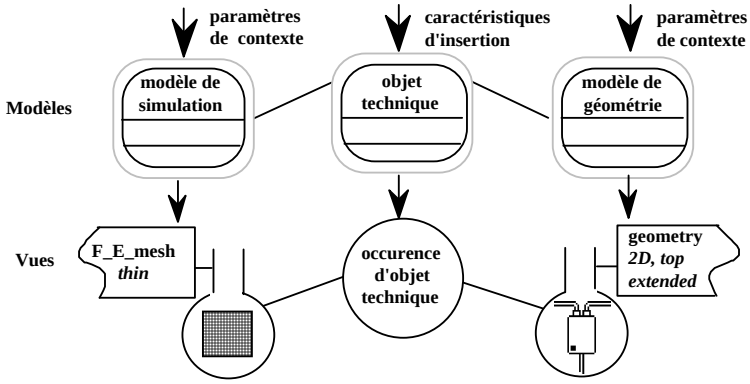

Figure 4 : Modèles et vues

Chaque catégorie de représentation peut donc se représenter, *dans la bibliothèque*, par une hiérarchie de classes, parallèle à la hiérarchie de classes de modèles généraux qui définit les composants, et reliée à celle-ci par une relation sémantique (i.e., connue du système informatique).

Outre les attributs dérivés et les contraintes d'intégrité, la description d'une classe de modèles fonctionnels nécessite l'introduction d'un troisième mécanisme de modélisation de la connaissance procédurale : la notion de *méthode de création de vue*. Celle-ci permet, pour un objet technique donné, pour un contexte d'insertion donné, et pour une catégorie de représentation donnée, de créer la vue fonctionnelle de l'occurrence correspondante, dans le modèle de produit.

Bien que définie au niveau des classes, la relation qui existe entre un *modèle général* (c'est-à-dire une instance de classe de modèles généraux) qui représente un objet technique, et un *modèle fonctionnel* (c'est-à-dire une instance de classe de modèles fonctionnels) qui représente un "modèle technique" de cet objet, est une relation d'instance à instance [Coad 92]. Cette relation est appelée relation *is_view_of*. Quand une classe est reliée à une autre classe par la relation *is_view_of*, cela signifie qu'il existe une instance légale de celle-là qui correspond à toute instance légale de celle-ci, et que celle-ci possède une méthode susceptible de créer, dans le modèle de produit, une vue fonctionnelle de celle-là.

4. Architecture proposée et échange entre systèmes

L'architecture proposée définie une claire séparation entre la *bibliothèque*, qui contient les classes d'objets techniques préexistants (niveau classe et instance, tant des objets techniques que des modèles fonctionnels) et le *modèle de produit* qui contient des ensembles d'occurrences (d'objets techniques et de vues fonctionnelles).

La bibliothèque modélise la connaissance sur les objets techniques préexistants, le "component knowledge" [Paasiala 93]. Le modèle de produit modélise la connaissance qui a été utilisée pour assembler ces objets et créer, à partir d'eux, un nouveau produit.

Enfin la notion *d'application,* définie comme les ensembles de programmes qui créent, ou traitent, le modèle de produit, conserve sa nécessité. A chaque catégorie de représentation est associée une application qui permet de sélectionner (dans la bibliothèque) un modèle fonctionnel, de lui transmettre les paramètres de contexte nécessaires à la création d'une vue fonctionnelle, et enfin de traiter l'ensemble des vues fonctionnelles de même catégorie qui apparaissent dans le modèle de produit.

Notons que, selon une approche qui semble de plus en plus fréquente [Carré 88] [Marino 90] [Perrot 92] [Sylvertsen 92], la multi-représentation est traitée par la technique de *l'agrégat d'instances*. Une instance particulière, le modèle général, porte essentiellement l'identité de l'objet réel. Chaque perspective est représentée par instance spécifique (modèle fonctionnel/vue fonctionnelle), reliée par une connexion d'instance [Coad 92] à l'instance distinguée.

Le modèle présenté ici a déjà été validé dans différents environnements. En particulier, un format d'échange a été défini pour permettre l'échange de hiérarchie de classes décrivant des objets techniques préexistants. Ce format, en cours de normalisation internationale [Pierra 94] [ISO 95], utilise la technique de la meta-programmation [Ait-Ameur 95] pour échanger la connaissance procédurale représentée au niveau des classes. Les processus dynamiques, contraintes fonctions et méthodes, sont spécifiés par des données qui définissent leur structure. L'échange des modèles de produit peut s'effectuer soit sous forme d'occurrences et de vues fonctionnelles explicites, il consiste alors en de simples données, soit en échangeant également le niveau instance (par exemple : modèle paramétré). L'aspect procédural est alors également représenté par meta-programmation.

Conclusion

Dans ce papier nous avons analysé l'impact de l'approche objet sur les systèmes de XAO, en nous intéressant surtout aux cas où le processus de conception consiste à assembler des objets techniques préexistants pour définir de nouveaux produits. Nous avons montré que les systèmes de XAO font intervenir trois niveaux d'abstraction. Les niveaux classe et instance permettent de représenter la connaissance intrinsèque sur les

composants. Ils peuvent être échangés entre systèmes et sont destinés à être stockés à l'extérieur du modèle de produit. Le niveau occurrence est le seul *nécessairement* représenté dans le modèle de produit. Il peut être échangé isolément. Si les deux systèmes faisant l'objet d'échange sont tous deux orientés objet, il peut également être échangé conjointement avec des instances auxquelles les occurrences présentées dans le modèle de produit font référence.

Nous avons également analysé les mécanismes de représentation de la connaissance qui apparaissent nécessaires pour représenter la connaissance intrinsèque propre aux objets réels préexistants. Des hiérarchies de classes, chacune correspondant à une catégorie de représentation, permet de représenter la connaissance structurelle. La description d'attributs explicites permet de représenter la connaissance descriptive. Concernant la connaissance procédurale nous avons proposé trois mécanismes permettant de la représenter : les attributs dérivés, les contraintes d'intégrité et les méthodes de création de vue.

L'ensemble de ces mécanismes permet de faire évoluer vers l'approche objet les systèmes XAO existant, tout en conservant les avantages que représentent, selon nous, l'architecture modulaire qui est actuellement la leur : modèle de produit, bibliothèque, programmes d'application et interface de dialogue.

L'ensemble des propositions présentées ici a été développé sous forme d'une spécification formelle, écrite en EXPRESS, et utilisant l'approche meta-programmation. Cette spécification a fait l'objet d'un consensus au niveau de la normalisation internationale et est actuellement en cours de vote sous le nom ISO CD 13584 "Parts Library".

Références

[Ait-Ameur 95] Y. Ait-Ameur, F. Besnard, P. Girard, G. Pierra, J.C. Potier, *"Specification and Metaprogramming in the EXPRESS Language"*, In Intern. Conference on Software Engineering and Knowledge Engineering SEKE'95 (IEEE - ACM Sigsoft), Rockville, USA, June 1995 *(à paraître)*.

[Beeker 89] E. Beeker, *"Application de la programmation orientée objet à la CAO"*, Proc. of MICAD'89, Hermes, Paris, (1989), 121-132.

[Carre 88] B. Carre, G. Comyn, *"On multiple classification, point of view and object evolution"*, In: Artificial Intelligence and Cognitive Science, J. Demongeot, T. Herve, V. Rialle, C. Roche, Eds., Manchester University Press, Manchester, UK, (1988), 49-62.

[Carre 90] B. Carre, J.M. Geib, *"The point of view notion for multiple inheritance",* Proc. of ECOOP/OOPSLA'90, October 21-25, (1990), 312-321.

[Chakravrthy 89] U;S; Chakravarthy, *"Rule Management and Evaluation: An Active DBMS Perspective"*, SIGMOD Record, Vol. 18, n° 3, Sept. 1989, pp. 20-28.

[Coad 92] P. Coad, E. Yourdon, *"Object-Oriented Analysis"*, Prentice-Hall (1992).

[Gal 95] A. Gal, O. Etzion, *Maintaining Data-Driven Rules in Databases,* Computers, IEEE, pp. 28-38, Juanuary 1995.

[Guillot 89] J. Guillot, J.Y. Rousselot, J.C. Vignat, *"Conception assistée par ordinateur d'ensembles mécaniques avec recherche d'une bonne solution : le logiciel : SICAM",* Proc. of MICAD'89, Hermès, Paris, (1989), 197-215.

[ISO 95] ISO CD 13584-24: Industrial Automation Systems and Integration, Parts Library, Logical model of supplier library, G. Pierra, Y. Ait-Ameur, Eds, ISO, Geneve, 1995.

[Marino 90] O. Marino, F. Rechenmann, P. Uvietta, *"Multiple Perspectives and Classification Mechanism in Object-Oriented Representation"*, Proc. of ECAI Conf. Stockolm, (July 1990), 425-430.

[Meyer 88] B. Meyer, *"Object-Oriented Software Construction"*, Prentice Hall, (1988).

[Minsky 86] M. Minsky, *"The Society of Mind"*, Simon & Schuster Inc., New York, (1986).

[Paasiala 93] P. Paasiala, A. Aaltonen, "*Automatic Component Selection"*, Realising CIM'sIndustrial Potential, C. Kooiji, MacConaill, J. Bastos, Eds., IOS Press, 1993.

[Perrot 92] J.F. Perrot, F. Wolinski, "*Modélisation par objets en robotique*", Technique et Science Informatique, 11, (1), (1992), 97-115.

[Pierra 90] G. Pierra, "*An object oriented approach to ensure portability of CAD standard parts libraries*", Proc. of Eurographics'90, Montreux, (Sept. 1990), 205-214.

[Pierra 94] G. Pierra, "*Modelling classes of pre-existing components in a CIM perspective: the ISO 13584/ENV 400014 Approach*", Revue internationale de CFAO et d'Infographie, Vol.9 - n°3, 1994 , pp. 435-454.

[Schenck 94] D. Schenck, P. Wilson *"Information modelling: the EXPRESS Way",* Oxford University Press (1994).

[Syvertsen 92] G. Syvertsen, F. Lillehagen, M. Lovstad, *"A generic object model for engineering design",* Proc. of TOOLS EUROPE'92, Dortmund, (March 30-April 2, 1992).

[Touron 92] P. Touron, D. Brunier-Coulin, T. Nguyen, *"CAS. CADE, un environnement logiciel pour bâtir des applications CFAO"*, Proc. of MICAD'92, Hermes, Paris, (1992), 379-391.