Actes de la Conférence MICAD'99

Une approche pour échanger des géométries paramétrées sur le réseau Internet

J.C. POTIER, G. PIERRA
{potier | pierra}@ ensma.fr

LISI / ENSMA – BP109 – 86960 FUTUROSCOPE Cedex
☎: 05-49-49-80 {78 | 60} 🖹: 05-49-49-80-64

RESUME **:** L'échange de géométrie paramétrée entres systèmes CAO hétérogènes suppose de définir un format qui fasse l'objet d'un consensus entre les concepteurs de logiciels. En l'absence de propositions de la norme STEP, la norme ISO 13584 (P-LIB) propose un tel format d'échange sous forme de programmes FORTRAN 90 s'appuyant sur une interface normalisée. Mais, cette approche, acceptée au niveau normatif, n'est que très peu utilisée. On propose, dans cet article, une représentation de l'interface en langage JAVA ainsi qu'une extension de cette interface en direction des expressions numériques permettant de sauvegarder le caractère paramétrique d'une définition géométrique, malgré son expression sous forme de programme. On montre que de tels programmes peuvent être générés automatiquement par un système CAO, et l'on discute différents usages qui peuvent en être faits, tant dans le domaine de la CAO que dans le cadre, plus large, des applications Internet.

ABSTRACT : Parametric geometry exchange would require a consensus on a standard exchange format. No proposal on this topics have been issued by the STEP project. P-LIB, officially ISO 13584-31, proposed an approach based on parametric functions associated with a FORTRAN binding. Formally accepted by ISO member bodies, this approach is not implemented. In this paper, we propose to develop a Java binding for ISO 13584-31. We also propose to extend this interface by modeling also numeric expressions. This extension should enable a parametric program to create a parametric model within a parametric system. We show that such parametric programs might be automatically generated from a parametric CAD system, and we discuss various usage of Java parametric program, both in the CAD area, and for Internet applications.

Mots-Clés: modélisation géométrique, échange de composants, CAO.

1. Introduction

Dans de nombreux domaines de la conception (mécanique, électronique, ingénierie...), les composants issus de catalogues fournisseurs représentent jusqu'à 90% de la totalité du produit. Cette utilisation massive de composants standard montre l'intérêt de disposer, au sein du bureau d'étude de versions numérisées des catalogues fournisseurs, comportant en particulier des représentations CAO.

L'existence de tels catalogues pose néanmoins un problème mal résolu jusqu'alors: comment fournir les représentations géométriques des composants dans un format exploitable par tout système CAO, et sous une forme dont le volume ne soit pas rédhibitoire. En effet, s'il existe à l'heure actuelle des formats trés largement acceptés, tels que STEP ou DXF, pour échanger de la géométrie explicite, il n'y a, en revanche, aucune solution réellement acceptée pour échanger les géométries paramétrées qui s'avèrent indispensables dès lors qu'une famille de composants regroupe un grand nombre d'éléments. En géométrie paramétrée, les solutions propriétaires, telles Autolisp ou les modèles paramétriques Pro-Engineer, ne sont en général pas supportées en dehors de leur système d'origine. Malgré un certain nombre de propositions [PIE 96] [PIE 96b] [PRA 96] [PRA 97] [AGB 98], aucun format paramétrique n'a encore émergé du projet STEP (ISO 10303), et il ne semble pas y avoir de consensus sur l'approche même devant être suivie. Une des seules solutions non propriétaire est celle proposée par la norme P-LIB (ISO 13524) dans sa partie 31. Cette norme permet l'échange de géométries paramétrées sous forme de programmes référençant une interface de programmation géométrique normalisée. L'inconvénient majeur de cette interface, qui explique sa faible diffusion, est que la syntaxe d'appel des fonctions de création de la géométrique est définie dans un langage de programmation nécessitant compilation/édition de liens (FORTRAN). Outre la programmation de l'interface dans un langage désormais peu utilisé dans les systèmes CAO, cette interface suppose, pour obtenir la géométrie d'un composant, de compiler le programme de géométrie paramétrée, d'en faire une éditions de lien avec son système CAO avec toutes les difficultés que cela présente. De plus, lorsque le système cible est lui même paramétrique, les expressions numériques étant évaluées à la traversée de l'interface FORTRAN, l'aspect paramétrique du composant est perdue lors de l'exécution du programme.

L'objectif de cet article est de proposer une approche qui résout ces deux difficultés à travers, d'une part, l'utilisation du langage JAVA, et, d'autre part, une extension particulière de l'interface P-LIB pour sa représentation en JAVA. L'utilisation de JAVA interprété, facilite à la fois, l'exploitation et le téléchargement des programmes. Sa caractéristique de langage très liée à l'Internet, ouvre également le domaine d'application de l'approche proposée bien au-delà des catalogues pour la CAO. La représentation en JAVA des opérateurs et expressions numériques sous forme d'instances de classe, susceptibles d'être créées par des fonctions de l'interface, et non sous forme d'expressions textuelles, évaluées lors de la traversée de l'interface

permet de restaurer la spécification paramétrique échangée en JAVA pour autant que le système receveur ait un modèle de donnée paramétrique.

Le contenu de ce papier est le suivant. Dans la page suivante, numérotée 2, nous présentons brièvement l'interface ISO 13584-31 : les entités géométriques qu'elle sait manipuler, les expressions qu'elle sait évaluer, et enfin, les fonctions de construction géométrique par contraintes qu'elle offre. Dans la troisième partie, nous proposons une technique de représentation des entités géométriques et numériques en langage JAVA qui permet de préserver la définition paramétrique des composants. Dans la quatrième partie, nous montrons également comment il est possible, avec des outils EXPRESS, de générer une partie importante de la représentation de l'interface en langage JAVA. Nous présentons également une expérimentation que nous avons mené en 2D, tant pour générer automatiquement le programme paramétrique JAVA correspondant à une géométrie paramétrée construite avec un éditeur graphique, que pour exploiter cette description dans divers environnements :

- incorporation dans un document HTML pour y représenter des dessins facilement modifiable,
- génération du code d'un format de géométrie explicite (DXF, IGES, STEP...) en parcourant la population d'instances par le programme paramétrique JAVA.

Dans la dernière partie, enfin, nous présenterons brièvement quelques travaux en cours sur ce sujet.

2. Architecture de l'interface ISO 13854-31

L'interface proposée dans la norme PLIB sous la référence 13584-31 [HEI 95] est composée deux parties : une hiérarchie d'entités géométriques, qui décrit formellement en EXPRESS [ISO 94a] le modèle du système auquel l'interface est supposée se connecter, et un ensemble de fonctions géométriques, qui permet de créer ces entités.

2.1 Modèle de données

La hiérarchie d'entités (figure 1) qui constitue le modèle de données provient essentiellement de la norme STEP [ISO 94b]. Afin de permettre une spécification précise des fonctions, ce modèle de données est défini en EXPRESS. Certaines entités sur lesquelles des constructions particulières sont possibles (i.e les segments de droite, les arcs de cercle,..) ont été définies par spécialisation des entités STEP («api_line », « api_circular_arc,..).

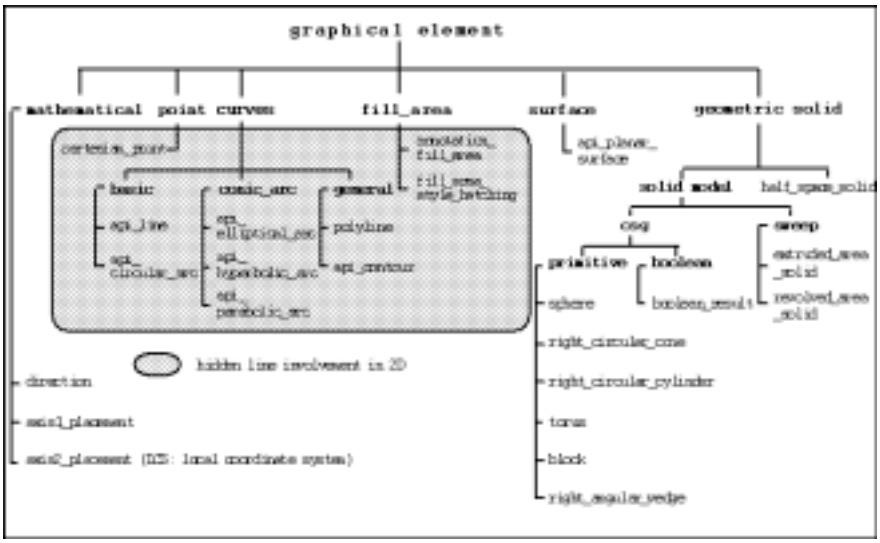


Figure 1 *– Aperçu de la Hiérarchie des entités de l'interface ISO 13584-31*

Chacune des entités de ce modèle peut être créée à deux endroits différents : soit directement dans le modèle du système CAO receveur, soit dans la base de données temporaire supposée gérée par l'interface. Cela permet, lorsque le système receveur n'est pas paramétrique, de réaliser les constructions d'esquisse dans la base de donnée temporaire, avant de transmettre au système CAO le résultat final.

2.2 Fonctions géométriques

L'interface ISO 13584-31 est basée sur le principe des API (Application Programming Interface) qui consiste à spécifier un ensemble de fonctions dont la syntaxe d'appel est décrite dans un ou plusieurs langages de programmation. Dans la norme PLib, le seul langage disponible pour cette interface est le langage FORTRAN 90. Par contre, l'interface supporte un grand nombre de fonctions de création géométriques spécifiées avec précision.

2.2.1 fonctions de création par valeurs

Ces fonctions permettent de créer des entités géométriques 2D/3D en fournissant des valeurs numériques qui les caractérisent. Ces valeurs peuvent être soit des paramètres de l'objet, soit des expressions qui référencent d'autres caractéristiques des entités déjà créées. Dans l'interface ISO 13584-31, des fonctions permettent de référencer certaines grandeurs numériques du modèle géométrique en cours (distance de ..,rayon de ....), par contre les opérateurs numériques eux-mêmes n'appartiennent pas à l'interface. L'écriture, dans le programme, d'une instruction telle que :

*Point10 = Point_Cartesian_Absolute(0.0,0.0,2*Distance_2_Pnt (Point4,Point5) , KFIX)*

entraîne l'existence de deux appels successifs de l'interface. D'abord, la fonction *Distance_2_Pnt()* est appelée et retourne la distance entre les deux points désignés. L'expression numérique est alors évaluée avant le nouvel appel de l'interface, puis l'interface appelle la fonction *Point_Cartesian_Absolute()* en transmettant comme troisième coordonnée le résultat de cette évaluation. La relation géométrique entre les points *Point4, Point5, Point 10* est donc ignorée par le système receveur, et par conséquent impossible à regénérer dans ce dernier.

2.2.2 fonctions de création par contraintes

Ces fonctions permettent de créer des entités géométriques 2D/3D en spécifiant des contraintes géométriques (tangent à, parallèle à,...). Dans la plupart des cas, ces contraintes donnent lieu à des constructions géométriques ambiguës. Par exemple, lorsque l'utilisateur souhaite construire un segment tangent à un cercle et passant par un point, il existe deux solutions distinctes (figure 2).

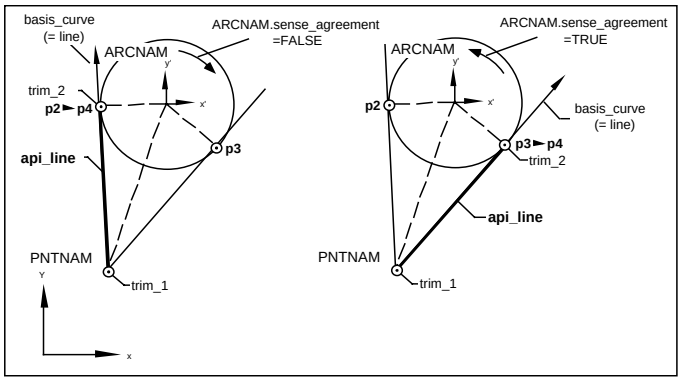


***Figure 2** – Exemple de construction par contraintes supportée par l'interface*

Dans l'interface ISO 13584-31, ces ambiguïtés sont levées grâce à des informations topologiques (orientation des entités, qualificatif extérieur /intérieur,..) qui sont explicitées dans la description de la fonction. Par exemple, dans la figure 2, le sens du cercle est utilisé pour préciser de quel coté il faut construire le segment.

2.2.3 fonctions de constructions 3D

Les opérations de construction 3D sont essentiellement des opérations de type CSG : extrusions générales et opérations booléennes.

Dans l'interface ISO 13584-31, le résultat de chaque opération est considéré comme un objet unique, pour lequel, il n'existe aucune fonction d'interrogation permettant d'accéder à une représentation par les frontières (B-Rep) susceptible d'exister dans le modeleur cible. Il n'existe pas non plus de surfaces paramétriques générales, de sorte que les géométries exprimables se limitent aux géométries CSG classiques

considérées comme suffisantes pour modéliser des objets dans une perpective d'utilisation (et non de fabrication. Ce domaine restreint, mais comportant toute les constructions par contraintes usuelles, est néanmoins couvert par prés de 150 fonctions, toutes spécifiées avec beaucoup de précision et ayant demandé plusieurs années de développement.

3. Implémentation de l'interface en JAVA

Malgré le consensus international dégagé sur cette spécification en 1997, peu d'implémentation semble avoir été réalisée. C'est la raison pour laquelle une nouvelle proposition est proposée ici.

Le langage JAVA présente pour le transfert par programme de géométries paramétrées trois avantages significatifs. Le premier est qu'il est interprété et bien adapté, à travers sa représentation en byte code, au téléchargement à distance. Le deuxième est qu'il supporte les concepts de la programmation objet (abstraction, polymorphisme, héritage....) sous une forme moderne (thread, interface) en général très appréciée. Le troisième est d'être lié à l'Internet, ce qui lui donne des domaines d'applications de plus en plus divers, et un potentiel de croissant très important.

3.1 Représentation du modèle de données

Dans l'interface ISO 135484-31, le modèle de données est spécifié dans un langage, lui même orienté objet EXPRESS. Il faut donc proposer des règles de transformation pour pouvoir passer du modèle EXPRESS en classes d'objets JAVA.

3.1.1 Les types

La représentation en JAVA des types de valeur existant en EXPRESS ne présente pas de difficultés particulières.

- Les types simples tels que les entiers, les booléens, les réels, les chaînes... peuvent être représenté en utilisant les classes du package java.lang.

- Les types collection d'EXPRESS qui sont l'ensemble (SET), la liste (LIST), le tableau (ARRAY), le multi-ensemble(BAG) peuvent être tous représentés par la classe `java.util.Vector` qui fournit des méthodes d'instance pour effectuer des traitements génériques à l'aide d'un itérateur.

- Les types énumérés d'EXPRESS peut être représentée chacun sous forme d'une classe ayant comme attributs chacune des valeurs du type énuméré. Il s'agit d'une implémentation similaire à la classe *Boolean* du package `java.lang` .

- Enfin, la notion d'union de types (SELECT) non disponible directement en JAVA peut être représentées sous forme d'une classe possédant un attribut privé de type objet.

3.1.2 Les entités

Le langage EXPRESS définit la notion d'entité pour représenter une famille d'objets. Dans le langage JAVA, cette notion peut être représentée de deux manières différents. En effet, lorsque l'univers des entités peut se définir avec un héritage simple, la notion de classe JAVA est suffisante, mais s'il doit être défini avec un héritage multiple, il faut en plus introduire la notion d'interface JAVA. Dans le modèle de données de l'interface ISO 13584-31, il n'existe pas d'entités définies avec un héritage multiple, la hiérarchie des entités définies dans la norme ISO 13584-31se représente donc très naturellement sous forme d'une hiérarchie de classes JAVA.

EXPRESS	JAVA
ENTITY line SUBTYPE OF (curve) ; pnt : cartesian_point ; dir : vector ; WHERE WR1 : dir.dim = pnt.dim END_TYPE ;	public class Line extends Curve { *private Cartesian_point : pnt ;* *private Vector : dir ;* public Line () { super ();} public Line (Label aName) { super (aName);} public Line (Label aName, Vector adir Cartesian_point aPnt) throws WR1Exception {.....} ..... public void setVector (Vector adir) throws WR1Exception {....} public Vector getVector () { .....} ..... }

Toutefois, le langage EXPRESS permet d'associer à une famille d'objets des notions d'attributs dérivés (DER) d'attributs inverses (INV) (contraintes de cardinalités sur les inverses des relations d'agrégation), et des contraintes d'unicité (UNIQUE) (règles globales imposant des valeurs uniques d'attribut pour chacune des instances d'une même classe), d'intégrités (WHERE RULES) (règles de comportement des instances).

En langage JAVA, les attributs dérivés l'EXPRESS (attributs dont la valeur est calculée à partir de fonction de dérivation), peuvent être considérés comme des

variables d'instance et calculés dans le constructeur de la classe ou associé à un sélecteur qui en calcule la valeur. Il n'est pas indispensable de représenter en JAVA les nombreuses contraintes d'intégrité définissable en EXPRESS. On peut cependant les vérifier au niveau du constructeur de la classe et des méthodes portant sur les instances.

3.2 Représentation des procédures de construction géométrique

A la différence du modèle de données, les fonctions définies dans l'interface ISO 13584-31 ne possèdent qu'une spécification décrite de manière informelle et d'une interface d'appel en langage FORTRAN 90. Il est donc nécessaire d'établir des règles de traduction pour les représenter en langage JAVA .

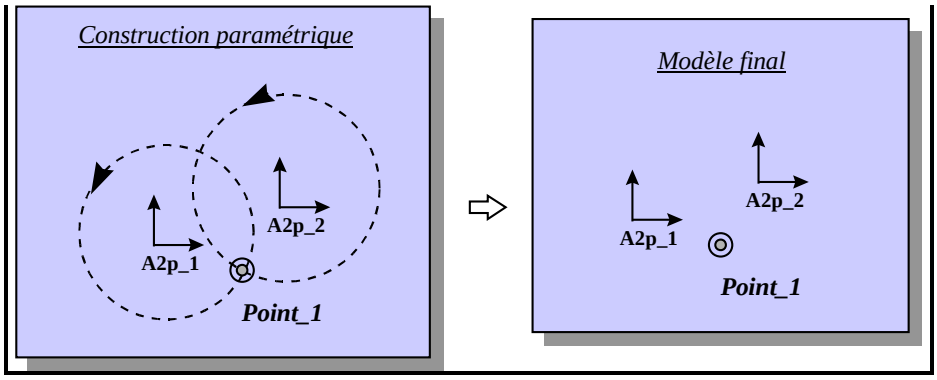
3.2.1 Notion d'identificateur d'entité

Les méthodes de création spécifiées dans l'ISO 13584-31 sont définies comme des fonctions dont le résultat est un identificateur de l'entité produite. Cet identificateur peut être donc ensuite utilisé pour référence l'entité pour des constructions ultérieures. Au cours de l'évaluation d'un programme paramétrique, des entités créées par une fonction peuvent être détruites par une autre. Il en est ainsi, en général, d'une opération booléenne lorsque celle-ci est évaluée par un modeleur géométrique effectuant une représentation par les frontières : les deux arguments de l'opération booléenne n'existent plus à l'issue de l'opération et seul existe le résultat. Il existe également une méthode, dans l'ISO 13584-31, pour spécifier qu'une entité construite par une fonction ne doit pas exister dans le modèle géométrique final. Chaque fonction de création possède un paramètre "KFIX" pouvant prendre les deux valeurs "CAO" et "TDB". Les entités créées avec une valeur "TDB" doivent exister seulement dans une base de données temporaire. A l'issue du programme, toutes les entités restant dans la base de données temporaires doivent être détruites. Ainsi, dans la figure xx où un point est créée par l'intersection de deux cercles d'esquisses qui sont ensuite détruits, les identificateurs des deux cercles (cercle_id_1 et cercle_id_2) doivent bien exister dans le modèle final pour spécifier la méthode de construction du pont, et permettre donc de le reconstruire, même si les entités géométriques cercle_1 et cercle_2 n'existe plus.

FORTRAN 90

```
! les repères A2P_1 et A2P_2 sont deux systèmes de coordonnées locaux déjà  définis

      cercle_id_1 = Circle_Rad_A2p (10.0, A2P_1, .TRUE., TDB)
      cercle_id_2 = circler_Rad_A2p (10.0, A2P_2, .TRUE., TDB)
      point_id_2 = Pnt_Intersection_2_Ent (cercle_1, cercle_2, CAD)
```



La représentation JAVA de l'interface doit donc représenter différemment les identificateurs d'entités, et les entités géométriques elles-mêmes. Les identificateurs sont représentés par une hiérarchie d'entité abstraites (non instanciables) identique à la hiérarchie de la figure 1 et qui ont un seul attribut, optionnel : l'entité géométrique qu'ils identifient, et qui est spécialisé tout au long de la hiérarchie.

EXPRESS	JAVA
ENTITY parametric_ID ABSTRACT SUPERCLASS; item : OPTIONAL geometric_representation_item; END ENTITY; ... ENTITY cartesian_point_ID ABSTRACT SUPERCLASS; SELF\parametric.ID_item : OPTIONAL cartesian_point; END ENTITY ;	public interface parametric_ID { public void setItem (Geometric_representation_item item) ; public Geometric_representation_item getItem ( ) ; public boolean existItem () ; } public interface cartesian_point_ID extends parametric_ID { ..... }

figure yy : Représentation des identificateurs d'entité géométrique

3.2.2 Représentation des focntions de création géométrique

Les fonctions de création de l'ISO 13584-31 sont donc, chacune, associée à une entité du modèle de données dont elle spécifie ainsi créée puisse, ensuite, être ré-évaluée si le système cible est paramétrique, trois conditions doivent être remplies :
- la fonction de création doit être sauvegardée,
- les paramètres de la fonction doivent être sauvegardées,
- la place de la fonction dans la composition de fonctions que représente le programme paramétrique doit être sauvegardée.

Dans l'approche proposée, ces trois conditions, sont remplies de la façon suivante. D'une part, chacune des fonctions de création représentée par une classe JAVA spécifique qui étend les classes de base d'identification (cf 3.2.1). Les paramètres d'entrée de la fonction peuvent donc être représenté comme des variables d'instance. Ceci permet de conserver pour toute instance, à la fois sa méthode de création, et l'identification des objets qui ont participés à sa création, même si dans une phase ultérieure les objets références ont cessés d'exister.

FORTRAN 90 + EXPRESS	JAVA
DEFINITION FORTRAN 90 : NAME = LIN_TANGENTIAL_2_ARC (ARCNM1, ARCNM2, KFIX) REPRESENTATION EN EXPRESS : ENTITY LIN_TANGENTIAL_2_ARC_ID SUBTYPE OF (API_LINE_ID) ; ARCMN1 : API_CIRCULAR_ARC_ID; ARCMN2 : API_CIRCULAR_ARC_ID; KFIX : INTEGER ; *DERIVE* . . . . . . *WHERE* *WR1 : (* segment length out of range [EPS,MAX]*)* END_TYPE ;	public class LIN_TANGENTIAL_2_ARC extends Api_line ID { *private Api_circular_arc_ID Arcnm1 ;* *private Api_circular_arc_ID Arcnm2 ;* *private int Kfix ;* public LIN_TANGENTIAL_2_ARC (Api_circular_arc_ID ARCNM1, Api_circular_arc_ID ARCNM2 , int KFIX) { super () ; ..... try { this.setLabel(new Label(« LINE_2_PT »)) ; this.setCurve (new Curve (.....)) ; this.setSenseAgreement (.....) ; ..... } catch (Exception e) { ..... } *public Api_circular_arc getARCNM1 () {* *return this.Arcnm1 ;* *}* *public Api_circular_arc getARCNM2 () {* *return this.Arcnm2 ;* *}* ..... }

Concernant la troisième condition, une solution possible est discutée dans la section 3.3.2.

3.2.3 Les expressions numériques

Nous avons déjà souligné que la restauration de la description paramétrique après la traversé de l'interface nécessitait la représentation explicite tant des paramétres numériques du programme que des opérateurs numériques intervenant dans les expressions. Ceci revient à considérer les paramètres et opérateurs numériques au même titre que les entités et fonctions de construction géométriques et à les représenter comme des instances de classe JAVA. Un tel modèle ayant déjà été développé en EXPRESS [AIT 95] [AIT 96], publié depuis dans une autre partie de la norme P-LIB, la partie 20 [ISO 98], la solution retenue a consisté :

- à enrichir les entités représentées dans l'interface, telles que définie par la Figure 1, par l'entité "expression_numérique" et ses sous-types tels que définis dans [ISO 98] : paramètres et variables internes, constantes littérales et opérateurs numériques.
- à représenter le type réel, pour les paramètres des fonctions de construction géométrique, par l'entité EXPRESS "entité_numérique" elle-même représenté en JAVA par la classe "expression_numérique" dont les instances représentent explicitement les expressions,
- à considérer les fonctions d'interrogation numérique, définies dans l'interface ISO 13584-31, comme des opérateurs numériques particuliers susceptibles d'intervenir dans une expression numérique ; ceci peut être fait en spécialisant l'entité "real_definied_function" définie dans ce but dans [ISO 98].

.

EXPRESS	JAVA
ENTITY real_defined_function ABSTRACT SUPERTYPE SUBTYPE OF (numeric_expression) ; END_ENTITY ; ENTITY DISTANCE_2_PNT SUBTYPE OF (real_defined_function) ; PNTMN1 : CARTESIAN_POINT; PNTMN2 : CARTESIAN_POINT; END_TYPE ;	public class Real_defined_function extends Numeric_expression { ..... } public class DISTANCE_2_PNT extends Real_defined_function { *public Cartesian_point_ID Pntnm1 ;* *public Cartesian_point_ID Pntnm2 ;* public DISTANCE_2_PNT (Cartesian_point_ID PNTNM1, Cartesian_point_ID PNTNM2) { ..... } ..... }

Notons que les expressions numériques n'ayant ni à être modifiées ni à être détruites après leur évaluation, le mécanisme de référencement indirecte discuté à la section 3.2.1 n'est pas nécessaire.

3.3 Représentation des programmes

Au sens de l'interface ISO 13584-31, un programme paramétrique est défini par un ensemble de paramètres une succession d'invocations de fonctions de l'interface ou d'invocation d'autres programmes paramétriques auxquels des paramètres effectifs correspondants à des expressions numériques sont transmis. Nous proposons de représenter un tel programme en JAVA sous forme d'une classe qui importe les classes qui définissent l'interface sur le système cible et contient comme constructeur le programme paramétrique lui-même. Celui-ci commence par initialiser l'interface, puis instancier ses propres paramètres, enfin invoque les fonctions de création géométriques ou numérique de l'interface qui instancient la géométrie explicite particulière correspondant aux valeurs courantes des paramètres du programme. Cette classe est déclarée comme sous classe de la classe shape_representation qui correspond à la représentation d'une forme et est supposée définie dans l'interface.

FORTRAN 90 + EXPRESS	JAVA
FORTRAN 90 : **SUBROUTINE** screw (int_rad,out_rad) *! parameters declarations* **DOUBLE PRECISION** int_rad **DOUBLE PRECISION** out_rad *! include LIB functions* **INCLUDE** 'p31_functions_types.for' *! *** begin of constructor ******************* *! 2D point by its coordinates* pnt1C1 =PNT_CARTESIAN_ABSOLUTE(°°°) °°°°° *! fix entities into CAD system* CALL FIX_ENT **(°°°°)** *! *** end of constructor ******************* **RETURN** **END** MAPPING_EXPRESS : ENTITY SCREW SUBTYPE OF (SHAPE_REPRESENTATION) ; INT_RAD : numeric_expression ; INT_RAD : numeric_expression ; *DERIVE* *? ? ?ITEMS : LIST [1 : ?] OF ITEM_REPRESENTATION :=* *LIST_TO_ARRAY (SELF*	import ISO_13584-31.* ; public class Screw extends Shape_representation { /* attributs of the class component*/ *private Numeric_expression int_rad ;* *private Numeric_expression out_rad ;* /* constructor of the class component*/ public screw (Numeric_expression int_rad, Numeric_expression out_rad) { super () ; this.setName(«'ITS_SCREW_CHC ‘) ; ..... /*2D point by its coordinates */ Cartesian_point pnt1C1 = new PNT_CARTESIAN_ABSOLUTE(°°°); ..... /*fix entities into CAD system*/ this. items.addElements (new FIX_ENT (°°)); } /* selector of attributs component*/ public Numeric_expression Int_rad () { return this.int_rad ; } .....

REPRESENTATION.ITEMS) ; *WHERE* ..... END_TYPE ;	/* iterator of class component*/ public Enumeration Elements () { return this.Items().Elements() ; } .....

3.4 Implantation paramétique de l'interface

Plusieurs implantations de l'interface sont possibles, selon la façon dont celle-ci implante la classe "shape_representation", et toutes les classes de la figure 1 qui doivent se comporter conformément à la description définie par le modèle EXPRESS, mais ne sont pas nécessairement représentées à l'identique.

Il est, en particulier possible, lorsque l'interface est implantée sur un système paramétrique, de générer un modèle paramétrique à l'intérieur de ce système lorsqu'un programme paramétrique séquentiel est exécuté en faisant appel à l'interface. En effet, l'exécution d'un programme paramétrique séquenteil commence par instancier ses propres paramètres dans l'interface puis correspond d'identificateurs d'entités géométriques ou d'expressions numériques dont l'instance résultante (paramétric_id ou numéric_expression) conserve sa méthode de construction et l'identification de ses paramètres. Il suffit alors que l'implantation de l'interface enregistre pour chaque forme créée par un programme paramétrique, les paramètres du programme et la liste des identificateurs d'identités géométriques et des expressions numériques dans l'ordre ou elles ont été par le programme pour que la définition paramétrique puisse être réévaluée par le système par simple parcours de cette liste et sans réexécution du programme extérieur.

4. Expérimentation en 2D

L'expérimentation que nous avons mené sur la faisabilité de notre approche en JAVA a été limitée à la géométrie 2 D. Néanmoins, elle a permis de montrer qu'il était possible d'automatiser la génération d'une grande partie de la représentation JAVA de l'interface, mais également qu'il était possible de générer les programmes paramétriques en JAVAZ à partir d'un éditeur graphique tout à fait analogue à un système CAO paramétrique. Enfin, notre implantation de l'interface en dehors de tout système CAO a montré que les programmes JAVA pouvaient être exploités, dans des environnements variés liés à l'INTERNET et pas seulement pour créer une géométrie explicite ou paramétrique dans un système CAO.

4.1 Génération de la représentation JAVA de l'interface ISO 13584-31

De nombreux outils existent pour valider la cohérence d'un modèle de données décrit en langage EXPRESS. Certains permettent, en compilant un schéma, de générer la base de donnée correspondante et son interface d'accès, appelée SDAI. D'autres enfin permettent, à partir d'un sous-ensemble des entités définis dans le schéma d'en extraire un schéma minimal ne contenant que les entités référencées directement ou indirectement.

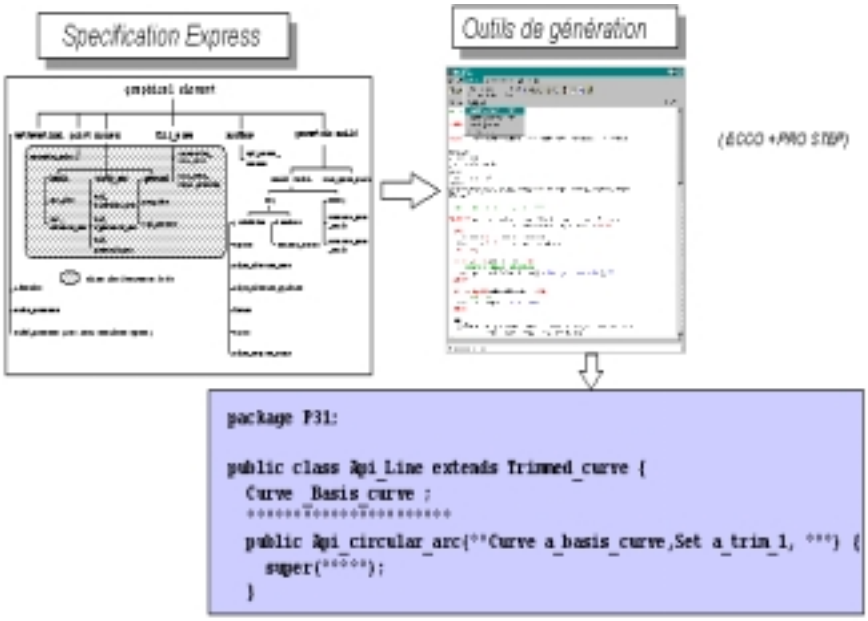


Figure 5 *– Génération automatique du binding JAVA de l'interface*

Dans notre expérimentation 2D, nous avons utiliser l'outil PRO STEP pour générer par fermetrue transitive du schéma ne contenant que les entités référencées par la géométrie 2D. A l'issue de cette phase, nous avons compilé ce schéma dans l'éditeur ECCO [STA 97] pour générer le modèle de données directement en langage JAVA (Figure 5). Les classes représentant les expressions numériques ont également été générées par ECCO de même que les identificateurs d'entités géométriques, préalablement représentées en EXPRESS. Quand aux fonctions de construction géométriques ont fait l'objet d'un développement manuel indépendamment de tout système CAO. Enfin, la classe shape_representation, destinée à être étendue par les classes qui définissent unprogramme paramétrique a été définie comme dans la norme STEP :

- un attribut d'instance, appelé "items", contient l'ensemble des entités géométriques qui appartiennent à la représentation,
- un autre attribut d'instance, "context_of_item", défini le contexte géométrique (2D, 3D,...) de ces entités.

Nous y avons ajouté une méthode d'instance pour pouvoir itérer sur la collection des entités de "items" :

```
/*iterateur sur items dans shape_representation*/
Public Enumeration Elements (){
return this.Items.Elements();
}
```

Figure xx - Iterateur sur les entités créées par un programme paramétrique

4.2 Génération des programmes JAVA

Le système que nous avons utilisé pour générer des programmes JAVA est un système CAO 2D spécialisé dans la programmation par démonstration [GIR 97]. Ce système avait été développé dans le cadre d'un projet ESPRIT pour montrer la possibilité de générer des programmes de géométrie paramétrée en langage FORTRAN 90 à partir d'un système CAO [POT 95]. Depuis, nous avons utilisé cette technique pour générer d'autres formats tels que des scripts AUTO-LISP de développer facilement pour permettant de développer facilement des formes géométriques paramètrées dans le logiciel AUTOCAD.

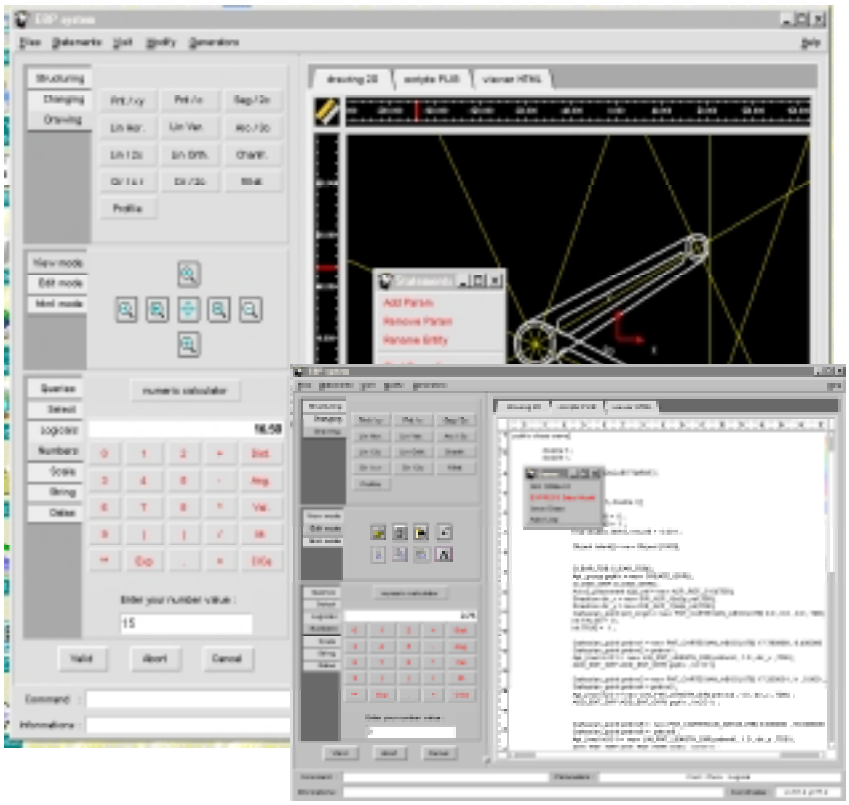


Figure 6 *– Aperçu du système EbP permettant de générer des classes JAVA*

Pour valider notre approche en JAVA, nous avons donc développé un autre post-processeur qui en analysant les programmes FORTRAN 90 conforme à l'interface ISO 13584-31 permet de générer le constructeur d'une classe paramétrique conforme à l'interface JAVA décrite dans la section 4.1. (figure 6)

4.3 Exploitation des programmes JAVA

Conçu pour développer des applications client/serveur sur le réseau INTERNET, tout programme écrit en langage JAVA offre de nombreuses possibilité d'exploitation. Dans le cadre de notre expérimentation, deux sortes d'exploitation des programmes générés ont été réalisés.

4.3.1 L'insertion dans les catalogues multimédia sur le WEB

Les catalogues multi-média sont de plus en plus présent sur le réseau INTERNET. Jusqu'à présent, les images de présentation sont essentiellement statiques (gif, tif, bmp....) et il est très difficile de montrer, lors d'un processus de sélection, l'influence du choix d'une valeur particulière d'un paramètre sur la représentation du composant.

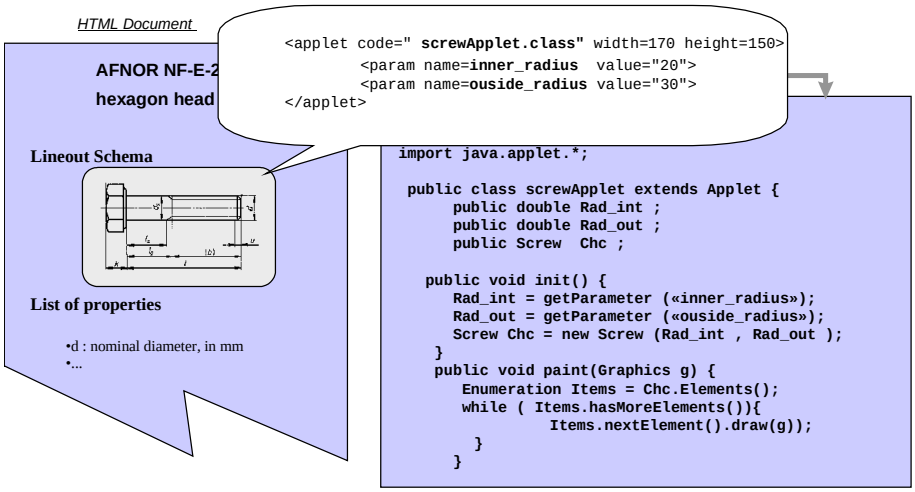


Figure 7 *– Exemple d'introduction d'une géométrie paramétrée dans un document HTML*

En substituant une image du catalogue initial par une "applet" JAVA qui permet de télécharger un programme de géométrie paramétrée et de l'exécuter le programme paramétré en local sur n'importe quel navigateur INTERNET, l'approche que nous proposons permet résoudre le problème.

4.3.2 Le génération de formats de géométrie explicite

Dores et déjà, quelques catalogue multimédia sur le WEB offrent la possibilité de récupérer la géométrie en temps réel (explicite) d'un composant sous divers formats (DXF, IGES,...). Un tel service est souvent rendu par l'intermédiaire d'un script CGI (Common Gatway Interface), situé sur le serveur du fournisseur, qui traite la requête émise par l'utilisateur pour fournir le bon fichier de géométrie. Ceci suppose que tous les fichiers contenant les différentes vues des composants dans les différentes dimensions et caractéristiques soient présents sur le serveur. Si on prend l'exemple des vis où des milliers de représentations existent compte tenu du grand nombre de dimensions existantes, cette approche apparaît impraticable.

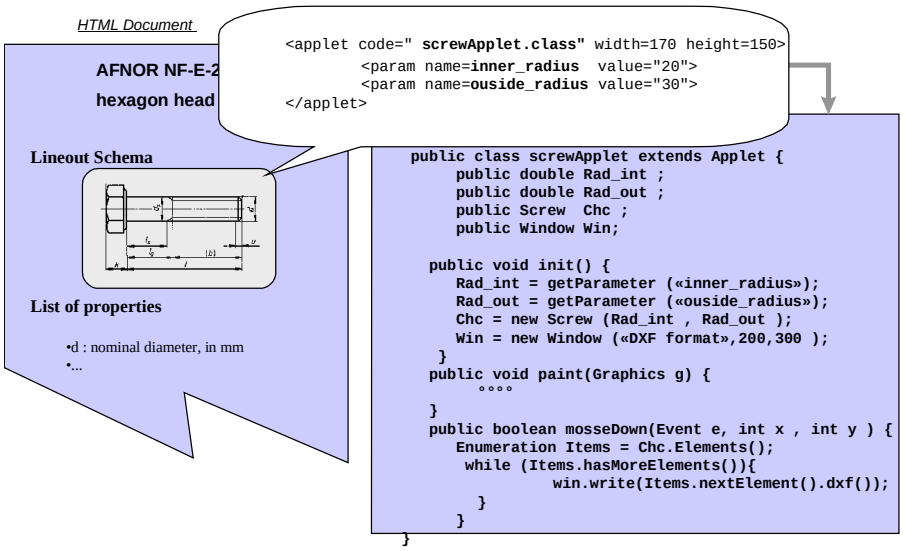


Figure 8 *– Exemple de génération d'une géométrie paramétrée au format DXF*

Avec l'approche paramétrée, une seule classe d'objets JAVA suffit pour générer la représentation explicite dans un format d'échange de tous les composants d'une famille selon une vue donnée. Au niveau programmation, il suffit d'ajouter une méthode d'instance à chaque classe d'entités permettant d'obtenir la représentation d'une instance de cette classe dans un format neutre choisi. Au niveau de l 'applet qui représente une instance de la famille d'objet, cela nécessite d'implémenter une méthode supplémentaire semblable à celle utilisée pour afficher la géométrie d'un composant dans un document HTML (figure 8). Cette méthode se limite à parcourir l'ensemble des instances d'entités du composant pour afficher leur code DXF. Ceci peut être fait dans une fenêtre d'édition implémentée au dessus de l'AWT (Abstract Widget Toolkit).

5. Autres développements et projets en cours

Parallèlement à nos développements dans le domaine du 2D, un développement est en cours dans le domaine 3D par la société LKSoftware pour implanter la version JAVA de l'interface ISO 13584-31 pour implanter la version JAVA de l'interface ISO 13584-31 sur une interface d'accès aux données de géométrie explicite 3D conforme à la norme STEP-AP 203 (ISO 10303-203). Le prototype de cette implantation, présenté lors de la réunion ISO TC184/SC4 de San Francisco (01/1999), permet d'avoir une interface de construction d'entités géométrique de beaucoup plus haut niveau (i.e avec des constructions par contraintes) que les contructeurs offerts pour créer des entités STEP.
La possibilité de transformation

Suite à une décision du groupe ISO en charge de P-LIB de recommander le développement d'une version JAVA normalisée de l'interface ISO 13584-31, des travaux ont été initiés aux USA, dans le cadre d'un projet de l'US Navy, à partir des propositions continues dans ce papier. Les premiers résultats officiels devraient apparaître au cours de l'année 1999.
Enfin, nous avons lancé deux études, à partir du système TOPCAD, pour montrer la faisabilité, d'une part, de générer des programmes paramétriques 3D en JAVA à partir d'un système CAO et, d'autre part, de restaurer dans un système CAO existant un modèle paramétrique à partir de l'exploitation d'un programme de géométrie paramétré JAVA s'appuyant sur l'interface décrite dans ce papier. L'architecture de ce dernier développement se base sur l'interface JNI qui permet d'appeler un interpréteur JABA à partir d'un système CAO pour autant que celle-soit définie en C/C++

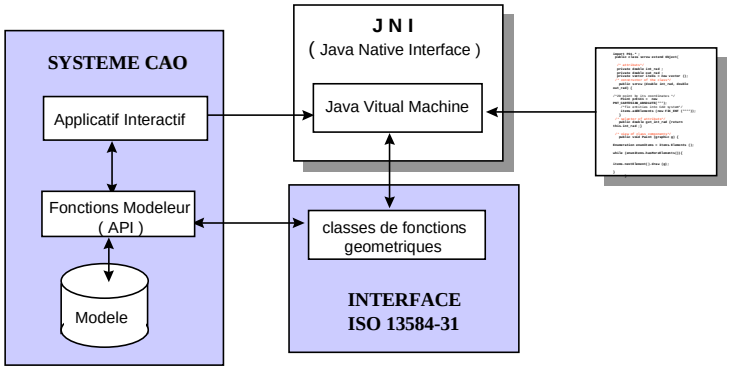


Figure 9 – Exemple d'implémentation de l'interface géométrique sur l'API Java Native Methode de Java

6. Conclusion

Même s'il existe à l'heure actuelle des formats universellement reconnus, tels que STEP, SET, IGES ou DXF, pour échanger la géométrie explicite, il n'y a en revanche aucune solution satisfaisant et faisant l'objet d'un consensus pour échanger la géométrie paramètrée. Les propositions élaborées dans le cadre du projet STEP n'avancent que très lentement et ne devraient pas être opérationnelle avant plusieurs années. L'approche par programmes paramétrés écrits en FORTRAN et s'appuyant sur l'interface ISO 13584-31, même si elle a fait l'objet de consensus au niveau normatif, ne semble pas acceptée par le marché.
Dans ce papier, nous avons proposé de réutiliser la spécification des fonctions dde géométrie paramétrique définie dans l'ISO 13584-31 mais d'en donner une représentation en JAVA. Nous avons également proposé d'étendre cette interface

pour lui permettre de représenter explicitement les expressions numériques fréquentes dans les programmes paramétriques.
Par rapport au développement d'un modèle de donnée de géométrie paramétrique permettant l'échange entre système paramétrique hétérogène, l'approche par programme paramétrés possède a priori trois inconvénients : la complexité d'écriture d'un programme, le fait que l'exécution d'un programme ne restaure pas, en général, un modèle de données paramétriques, l'incomplétude enfin des géométries couvertes. Concernant le premier aspect, nous avons montré que le programme pourrait être généré. Concernant le deuxième aspect, nos propositions sur le contenu de l'interface permettent dès implantations qui sauvegardent le caractère paramétrique. Concernant la complétude enfin, s'il est claire que tant la natrue des entités disponibles -en particulier l'absence de surfaces paramétriques- que la variété des fonctions paramétriques spécifiées apparaissent insuffisantes pour l'échange de modèles quelconques entre système CAO, d'une part elles semblent suffisantes dans la plupart des cas pour l'application la plus urgente : l'échange de modèles géométriques paramétrés de composants standard, d'autre part l'implantation proposée, avec son mécanisme de référencement indirect, devrait permettre d'étendre de façon continu les géométries couvertes, y compris l'accès interne à des représentations de type frontière. Enfin, un format d'échange de géométrie paramétrée en Java devrait contribuer à de nouveaux développements pour l'Internet, ... et bénéficier de la croissance de celui-ci pour se développer.

7. Références

[HEI 95] HEINE, L, PIERRA, G., « ISO DIS 13584-31 : Industrial Automation Systems and Integration, Parts Library, Geometric Programming Interface » , 1995

[GIR 97] GIRARD P., PATRY G., PIERRA G., POTIER J.-C., "Deux exemples d'utilisation de la programmation par démonstration en conception assistée par ordinateur", Revue Internationale de CFAO et d'Infographie, Vol 12, N° 1-2, 1997, pp. 169-188.

[ISO 94a] ISO 10303-11, Industrial Automation Systems and Integration, Product Data Representation and Exchange, "The EXPRESS language reference manual", ISO, Geneva, 1994

[ISO 94b] ISO 10303-42, Industrial Automation Systems and Integration, Product Data Representation and Exchange, "Integrated generic resources: geometric and topological representation", ISO, Geneva, 1994

[ISO 98] ISO 13584-20: Industrial Automation Systems and Integration, Parts Library, Logical Resources, "Logical Model of Expressions" ISO, Geneva, 1998

[ISO 96] ISO DIS-13584-31: Industrial Automation Systems and Integration, Parts Library, Implementation Resources, "Geometric Programming interface", ISO, Geneve, 1996

[PIE 96] PIERRA, G., POTIER, J.C., GIRARD, P., "The EBP system : Example Based Programming for Parametric Design", in Modelling and Graphics in Science and Technology, Réd. J. Teixeira et J. Rix, ISBN 3-540-60244-5, Ed. Springer, 1996, pp. 124-140.

[PIE 96] PIERRA, G., AIT-AMEUR, Y., BESNARD, F., GIRARD, P., POTIER, J.C., "A general framework for parametric product model within STEP and Parts Library", Proc. of European PDT Days, London, 18-19 Avril 1996, PDTAG-AM, pp 69-104.

[POT 95] POTIER, J.C., GIRARD, P., PIERRA, G., BESNARD F., « Génération graphique interactive de programmes de géométrie paramétrée », Revue d'Automatique et de Productique Appliquée (RAPA), 1995, n° 8, vol. 2-3, pp. 229-234.

[KLE 98] ISO ISO 10303-22: KLEIN L, STONIS, A., « STEP - Java SDAI Standard Data Access Interface », 1998

[STA 97] STAUB G., MAYER M., « User Manuel Référence : an enviromnent for evaluation of EXPRESS model», 1997

[ISO 94a] ISO 10303-11, Industrial Automation Systems and Integration, Product Data Representation and Exchange, "The EXPRESS language reference manual", ISO, Geneva, 1994

[ISO 94b] ISO 10303-42, Industrial Automation Systems and Integration, Product Data Representation and Exchange, "Integrated generic resources: geometric and topological representation", ISO, Geneva, 1994

[ISO 98] ISO 13584-20: Industrial Automation Systems and Integration, Parts Library, Logical Resources, "Logical Model of Expressions" ISO, Geneva, 1998

[ISO 96] ISO DIS-13584-31: Industrial Automation Systems and Integration, Parts Library, Implementation Resources, "Geometric Programming interface", ISO, Geneve, 1996