Génération graphique interactive de programmes de géométrie paramétrée[1]

Jean-Claude Potier, Patrick Girard, Guy Pierra, Frédéric Besnard.

Laboratoire d'Informatique Scientifique et Industrielle, ENSMA
Site du Futuroscope - B.P. 109 - 86960 FUTUROSCOPE Cedex
Tél. (33) 49-49-80-63 - E-mail {girard,pierra}@ensma.univ-poitiers.fr

RÉSUMÉ. L'échange de bibliothèques de composants entre systèmes intégrés de production suppose en particulier la possibilité d'échanges des géométries paramétrées. La norme ISO CD 13584-31 définit un tel format d'échange sous forme de programmes FORTRAN s'appuyant sur une interface normalisée. On montre dans cet article qu'un tel format peut être généré à partir d'un système graphique interactif, tout à fait analogue à un système paramétrique, ce système pouvant même supporter des constructions différentes de celles de l'interface cible.

ABSTRACT. Exchanging parts libraries between CAD-systems requires capabilities to exchange parametric geometry. The exchange format specified in ISO CD 13584-31 consists in FORTRAN programs based on a standard API. In this paper we show that such a format may be generated from a graphic system that provides similar capabilities as parametric systems, and that may support constructs which are different from the target API.

MOTS-CLÉS : Programmation sur exemple, CFAO, géométrie paramétrée., bibliothèques de composants.

KEYWORDS : Programming by example, CAD/CAM, parametric design., parts library.

[1] *paru dans* Revue d'Automatique et de Productique Appliquée (RAPA), 1995, n° 8, vol. 2-3, pp. 229-234.

Génération graphique interactive de programmes de géométrie paramétrée

Jean-Claude Potier, Patrick Girard, Guy Pierra, Frédéric Besnard.

Laboratoire d'Informatique Scientifique et Industrielle, ENSMA
Site du Futuroscope - B.P. 109 - 86960 FUTUROSCOPE Cedex
Tél. (33) 49-49-80-63 - E-mail {girard,pierra}@ensma.univ-poitiers.fr

RÉSUMÉ. L'échange de bibliothèques de composants entre systèmes intégrés de production suppose en particulier la possibilité d'échanges des géométries paramétrées. La norme ISO CD 13584-31 définit un tel format d'échange sous forme de programmes FORTRAN s'appuyant sur une interface normalisée. On montre dans cet article qu'un tel format peut être généré à partir d'un système graphique interactif, tout à fait analogue à un système paramétrique, ce système pouvant même supporter des constructions différentes de celles de l'interface cible.

ABSTRACT. Exchanging parts libraries between CAD-systems requires capabilities to exchange parametric geometry. The exchange format specified in ISO CD 13584-31 consists in FORTRAN programs based on a standard API. In this paper we show that such a format may be generated from a graphic system that provides similar capabilities as parametric systems, and that may support constructs which are different from the target API.

MOTS-CLÉS : Programmation sur exemple, CFAO, géométrie paramétrée., bibliothèques de composants.

KEYWORDS : Programming by example, CAD/CAM, parametric design., parts library.

1. Introduction

Dans de nombreux domaines de conception (électronique, thermique, mécanique...), un objet technique est défini sous la forme d'un assemblage de composants préexistants. Ainsi, en conception unitaire (montage d'usinage) ou même de petites séries (machines spéciales), les composants standard atteignent jusqu'à 90 % de la totalité des pièces d'un assemblage. Cette utilisation massive s'explique par la baisse sensible des coûts de production qu'elle entraîne. C'est la raison pour laquelle l'intégration d'objets préexistants dans les systèmes de Conception Assistée par Ordinateur (CAO) représente

l'une des préoccupations majeures pour les concepteurs.

Mais l'existence de bibliothèques complètes et certifiées est subordonnée à la possibilité de les rendre portables d'un système à l'autre. Une norme internationale (ISO CD 13584: Parts library, aussi connue sous le nom de CAD-LIB [PIE 94]) a été développée au cours de ces dernières années pour permettre l'échange, d'un système CAO à l'autre, de bibliothèques de composants multi-représentations et multi-fournisseurs. Actuellement en cours de vote au niveau international, elle définit en particulier une interface de programmation (ISO 13584-31: Programming interface) permettant d'assurer la portabilité des géométries paramétrées de familles de composants sous forme de programmes écrits en FORTRAN 90. L'objectif de cet article est de montrer que ce format (d'échange) peut en fait être généré automatiquement à partir d'un système interactif graphique, et ceci de manière analogue à un système paramétrique.

2. La génération interactive de programmes

Divers travaux [HAL 84], [VAN 91], [GIR 93], ont établi la possibilité de transformer les scripts (i.e. des enregistrements composés de commandes et d'opérandes) en de véritables programmes, grâce à l'introduction de la notion de **contexte dynamique** en programmation graphique sur exemple. Le principe consiste à associer à toute **création d'objet** dans le modèle du système interactif une **déclaration de variable** dans le contexte de programme. Cette déclaration s'effectue en attribuant automatiquement à la variable un nom et en lui affectant pour valeur une référence à l'objet créé dans le modèle. Lors de toute désignation, il est ensuite possible de réaliser les substitutions **valeur / nom** requises durant la phase de création de programme, et les substitutions **nom / valeur** nécessaires à la phase d'exécution. Cette association des variables aux objets permet même d'introduire dans de tels programmes les structures de contrôle.

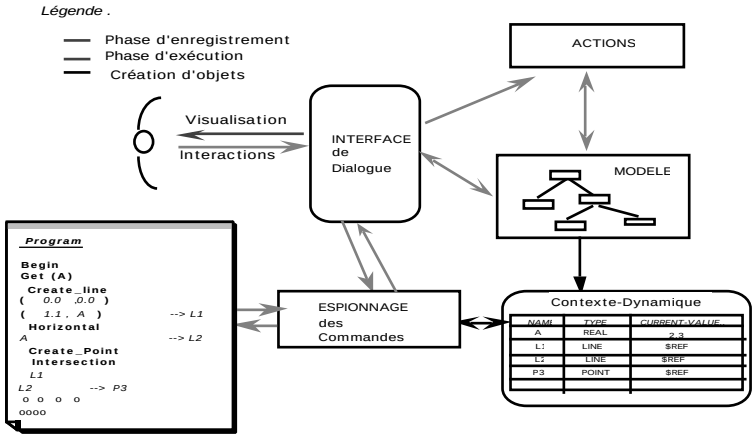

Figure 1. *Architecture du système d'espionnage LIKE*

Le système LIKE [GIR 93] ayant servi à valider cette approche (illustré sur la figure 1) présente néanmoins une limitation majeure : tout programme généré est constitué de commandes appartenant au seul système sur lequel il a été conçu, le rendant inutilisable pour un échange entre systèmes différents.

En effet, la différence de structure qui existe entre l'enchaînement de commandes enregistré dans un script et le séquencement des instructions d'un programme impératif rend toute traduction *a posteriori* illusoire. En effet, les règles d'ergonomie [COU 90], [MEI 91] appliquées lors de la conception des interfaces rendent la chronologie des commandes contextuelle. Par exemple, lorsque l'utilisateur suspend une commande en cours pour en effectuer une autre (fils d'activités multiples), un opérande peut se trouver totalement séparé de la commande à laquelle il se rapporte. De même, lorsque l'utilisateur rend une commande rémanente, celle-ci peut, de par son caractère modal, ne jamais apparaître dans le script. À l'inverse, dans un programme, le séquencement des instructions est très rigide, et chaque procédure doit être suivie des paramètres sur lesquels porte son exécution. Ces difficultés montrent que, pour pouvoir générer des programmes neutres, le niveau de capture (ou encore d'espionnage) ne doit pas correspondre aux interactions émises par l'utilisateur, mais plutôt aux appels d'actions résultant de ces interactions.

3. Adopter différents niveaux de capture des interactions

Le modèle "de Seeheim" [PFA 85] fournit un cadre universellement reconnu pour l'architecture des interfaces de dialogue ; il identifie ainsi trois composants, le composant de présentation, le composant de contrôle, et le composant servant d'interface avec le noyau fonctionnel de l'application. Deux niveaux possibles pour la capture des interactions peuvent alors être distingués :

(1) Le premier niveau se situe immédiatement après le composant de présentation. Il s'agit d'un niveau *commande-opérande* où l'enchaînement des interactions correspond à celui émis par l'utilisateur. Il permet essentiellement de remplacer les désignations d'objets (pointés) par les identificateurs des entités correspondantes.

(2) Le second niveau se situe après le contrôleur de dialogue. Il s'agit d'un niveau *requête-système* où chaque action est associée à ses paramètres et où l'on peut collecter toutes les informations nécessaires à l'élaboration d'une représentation abstraite susceptible d'être ensuite réexploitées pour générer un programme neutre.

Le système EBP ayant servi à concrétiser cette approche est illustré sur la Figure 2. Les interactions de l'utilisateur sont exploitées à ces deux niveaux. Le niveau inférieur permet, lors de toute désignation, de vérifier que chaque entité désignée fait bien partie du contexte dynamique du programme, et peut être licitement utilisée dans la construction. Quant au niveau supérieur, il permet de capturer les appels d'action pour construire l'arbre abstrait du programme.

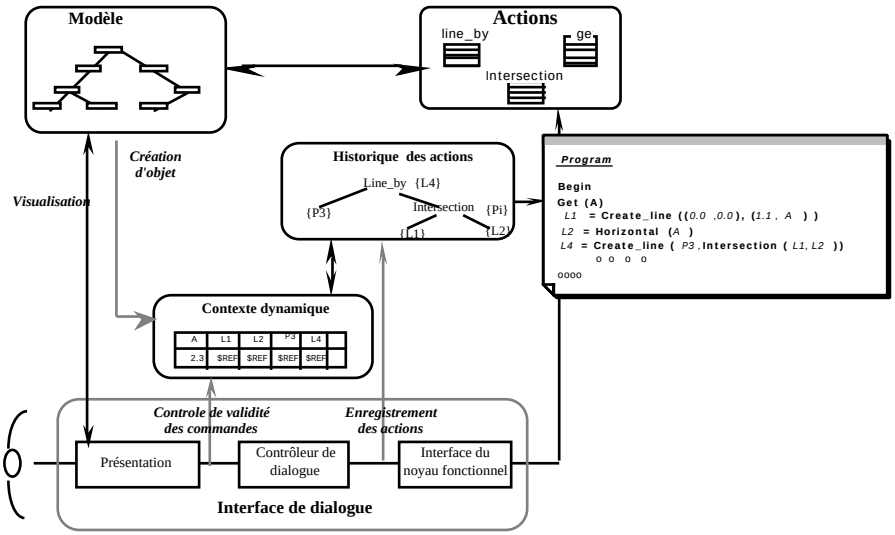

Figure 2. *Architecture du système d'espionnage EBP*

L'analyse effectuée permet de reconstituer la signature de l'action concernée, puis de remplacer les valeurs, qui figurent dans les paramètres effectifs, par des références aux variables qui doivent être stockées dans le programme, selon trois cas de figure :

(1) la valeur du paramètre appartient au contexte du programme : il s'agit donc d'une *variable* dont le nom doit être substitué à la valeur dans le fragment d'arbre abstrait de l'action traitée.

(2) la valeur du paramètre n'appartient pas au contexte du programme : il s'agit d'une *constante* dont la valeur doit être enregistrée dans le fragment d'arbre abstrait de l'action traitée.

(3) enfin, la valeur du paramètre est le résultat de l'évaluation d'une expression grapho-numérique ou géométrique [GAR 86] (ex : la **projection** du **centre** d'un cercle sur une droite donnée) ; l'arbre abstrait de l'expression doit être substitué à la valeur dans le fragment d'arbre abstrait de l'action traitée.

À ces substitutions s'ajoutent une substitution particulière, lorsqu'il s'agit d'actions géométriquement ambiguës (ex : droite tangente à deux cercles). Elle consiste à enrichir la signature des actions par une information topologique additionnelle (un qualificatif représentant une disposition *intérieur / extérieur* pour un cercle, ou un sens trigonométrique *plus / moins* pour une entité orientée). Cette dernière information permet non seulement, d'une réexécution à l'autre, de respecter une certaine configuration lors de l'élaboration des objets, mais également de faire abstraction des pointés de positionnement utilisés dans l'exemple pour lever les ambiguïtés qui en résultent, et qui perdent tout sens dans un programme paramétré.

4. La réécriture des actions en langage neutre

La génération du programme neutre nécessite d'abord d'établir, pour chacune des actions de l'arbre abstrait, un squelette de traduction dans le langage hôte (i.e. des blocs de code écrit en FORTRAN 90 faisant appel aux procédures de l'interface normalisée). Dans la mesure où l'interface offerte à l'utilisateur est beaucoup plus riche que l'interface normalisée, la plupart des actions abstraites ne peuvent avoir une correspondance immédiate avec les primitives offertes dans l'interface normalisée. L'action est alors décomposée en une succession de primitives de cette même interface. Des blocs d'instructions sont générés pour effectuer le codage en langage neutre. Ces blocs contiennent des identifiants formels (en italique gras dans la Figure 3) à substituer lors de l'intégration dans le programme généré. Pour les actions géométriquement ambiguës, la correspondance est établie en utilisant des règles d'interprétation entre les conventions propres au système support et celles disponibles dans l'interface normalisée (essentiellement des changements d'orientation d'entité [ISO 95]). Ces règles sont appliquées aux informations topologiques mémorisées dans l'arbre abstrait, pour permettre d'insérer dans le squelette de l'action des instructions supplémentaires nécessaires.

L'exemple suivant illustre la traduction correspondant à la création d'un cercle de rayon donné tangent à une droite et passant par le milieu d'une autre entité (cercle,droite,arc...). l'action abstraite, intitulée *circle_pt_ln_num* contient quatre paramètre : un point (associé à une expression qui le définit comme le milieu), une droite, un réel (constante) et une information topologique additionnelle. Les noeuds sont réduits de façon ascendante avec substitution des identifiants formels.

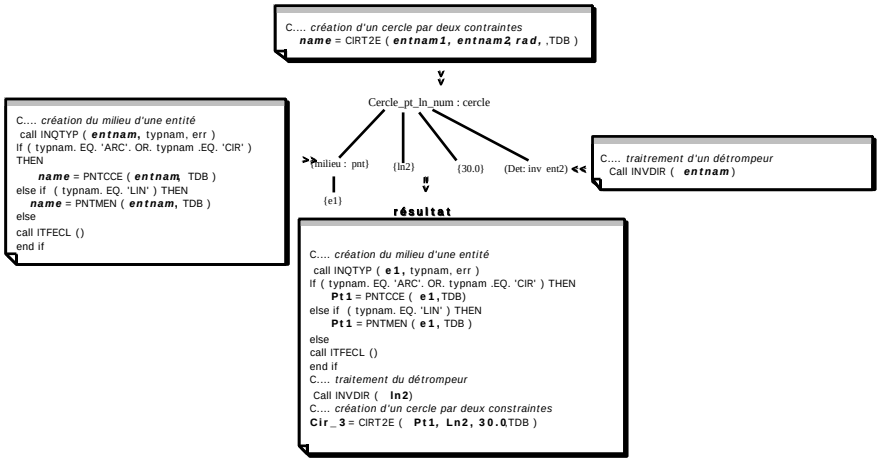

Figure 3. *Exemple de convention de réécriture*

Les blocs d'instructions ont l'avantage d'être adaptables et réutilisables à volonté chaque fois qu'un noeud de l'arbre est traité. Lorsqu'ils contiennent des

identifiants formels, ces derniers sont d'abord substitués par des variables du programme abstrait d'après l'ordre qu'ils occupent dans le fragment d'arbre correspondant au noeud. Ce processus de traduction est répété jusqu'à ce que le fragment d'arbre soit réduit à un noeud unique. Une fois cette étape réalisée, la séquence d'instructions obtenue constitue le résultat de la traduction, le programme neutre.

5. Conclusion

Par rapport aux nombreux systèmes présents sur le marché, que ce soit en conception paramétrique ou en géométrie variationnelle, le système EBP présente l'intérêt de pouvoir :

1- générer interactivement un véritable programme contenant les structures de contrôle usuelles (alternative, répétition et sous-programmes)

2- assurer la portabilité des programmes générés en s'appuyant sur un langage neutre standardisé.

3- permettre une mise au point interactive des programmes générés, en permettant à l'utilisateur de faire dérouler le programme jusqu'à l'élaboration de l'objet qu'il désigne à l'écran pour modifier la manière dont il a été conçu.

4- enrichir et personnaliser l'interface de dialogue du système interactif qui l'héberge, et ce en permettant à l'utilisateur d'associer l'exécution d'un programme construit interactivement à une commande du système.

Ces quatre grandes fonctionnalités font de EBP un environnement de programmation puissant réellement adapté à un utilisateur novice en informatique.

6. Bibliographie

[COU 90] COUTAZ J., *Interfaces homme-ordinateur, Conception et réalisation*, Dunod-Informatique, 1990.

[GAR 86]] GARDAN Y., *La CFAO, Introduction, techniques et mise en oeuvre*, Hermès, 1986.

[GIR 93] GIRARD P et *al.*, « Command Recording versus Parametric and Variational Systems, and old/new third way of parametrizing CAD models by End Users », *Proc. of COMPEURO'93, Paris (Mai 1993)*, Ed. IEEE Comp. Society Press, pp. 194-200.

[HAL 84] HALBERT D., Programming by example, PhD. Thesis, Berkeley Univ., California, 1984.

[ISO 95] *CAD Parts-library : Programming interface*, ISO CD 13584-31, Genève,1995.

[MEI 91] MEINADIER J.P., *L'interface utilisateur, pour une informatique conviviale*, Dunod, 1991.

[PFA 85] PFAFF J., « User interface Management Systems », *Proc. of the workshop on User Interface Manangement System* (1985), Springer-Verlag.

[PIE 94] PIERRA G., « Modelling classes of pre-existing components in a CIM perspective: the ISO 13584/ENV 400014 Approach », *Revue internationale de CFAO*

et d'Infographie, 9, n°3, 1994, Hermès, pp. 435-454.

[VAN 91] VAN EMMERIK M., Interactive design of parametrized 3D models by direct manipulation, PhD Thesis, Delft University, Netherland, 1991.