ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN TECHNOLOGIE DES SYSTÈMES
M.ING.

PAR
SERGE ST-MARTIN

MAÎTRISE DU CHANGEMENT DANS LES MODÈLES CAO

MONTRÉAL, 18 JANVIER 2001

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

- M. Jean-François Châtelain, directeur de mémoire
  Département de génie mécanique à l'École de technologie supérieure

- M. Roland Maranzana, codirecteur de mémoire
  Département de génie de la production automatisée à l'École de technologie supérieure

- M. Alain Desrochers, professeur
  Département de génie mécanique à l'Université de Sherbrooke

- M. Louis Rivest, professeur
  Département de génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE PRÉSENTATION DEVANT JURY ET UN PUBLIC

LE 14 DÉCEMBRE 2000

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

MAÎTRISE DU CHANGEMENT DANS LES MODÈLES CAO

Serge St-Martin

Sommaire

Les gestionnaires de données de produit assurent la bonne administration de l'ensemble des fichiers générés pour la description et la fabrication d'un produit. Parmi ces fichiers, on retrouve les modèles de CAO, contenant une représentation géométrique du produit, mais également une foule d'informations y étant associées. Comme le modeleur CAO permet la modification rapide des concepts établis, des familles entières de produits peuvent être créées. L'évolution en révisions des modèles CAO est si rapide que l'on peut facilement perdre de vue les différences qui distinguent ces révisions.

Ce projet d'application présente un outil de comparaison de modèles géométriques basé sur l'analyse des arbres de modélisation qui définissent les produits, sous leur représentation par historique (arbre CSG). Dans un ordre de grandeur de la minute, le comparateur extrait systématiquement toutes les nuances géométriques qui distinguent les deux modèles. C'est en analysant les paramètres qui définissent les volumes primitifs de l'arbre de modélisation, ainsi que sa structure, que le comparateur discerne les données différentes. Il en résulte un rapport de comparaison identifiant les primitives altérées ou isolées, ainsi qu'une coloration de ces primitives, autant dans la représentation solide du modèle que dans son arbre de modélisation.

Des essais ont été réalisés avec des pièces de conception simple, mais aussi à l'aide de composants de type aéronautique fournis par une multinationale (Bombardier). Ces essais ont révélé des opportunités de poursuite des travaux, en particulier sur l'optimisation temporelle des algorithmes de comparaison utilisés et l'enrichissement de la description des différences extraites dans le contexte d'une évolution de modèle.

Il n'en demeure pas moins que le comparateur de modèles décèle toutes les modifications présentes entre deux révisions d'un modèle. Il trouve alors sa raison d'être parmi les systèmes de gestion de données techniques, dans un secteur non couvert par les gestionnaires de données de produits, c'est à dire la gestion de l'évolution d'un modèle CAO par l'analyse de données contenues à l'intérieur des fichiers des différentes révisions du modèle.

MASTERING CHANGE MANAGEMENT FOR CAD FILES

Serge St-Martin

Abstract

The evolution of parts through the process of design of an enterprise produces a large amount of CAD files necessary to describe a product along his development life cycle. Product data manager are required to manage these files efficiently for the use of a user. In the context of the evolution of a part, therefore of his CAD model, it is desired to know and understand the slight differences that distinguish versions of the model between themselves.

In this project, we propose the use of a model comparator to extract and define the differences between two CAD models, therefore between the versions of a same CAD model or part. We propose a method that uses the historical representation of the model, extract the parameters of the primitives composing the CSG tree, partially analyses the structure of the tree and finally identifies the modifications applied from a versions of the model to the other.

The model comparator has been used with aeronautical parts to validate his efficiency. The results of the tests revealed that the comparator can be optimized in two specific fields : the enrichment of details mentioned in the report of differences and the optimization of the comparison algorithm's speed. Nevertheless, the comparator finds all the differences between two models, within the limits described in the document.

Hence, a model comparator that can extract and analyse information within a CAD file finds its purpose along technical data managing systems such as PDM, which is limited to the external management of files.

REMERCIEMENTS

J'aimerais remercier mes codirecteurs Jean-François Châtelain et Roland Maranzana qui m'ont proposé un projet de développement en conception assistée par ordinateur pour mes études de deuxième cycle. Ils ont, au cours de mes deux ans de formation, orienté mes travaux de recherche et m'ont conseillé sur les opportunités à saisir afin d'innover dans le domaine des systèmes de gestion de données techniques couramment utilisés chez les grandes entreprises automobiles et aéronautiques, pour ne nommer que celles-ci.

Mes remerciements s'adressent également aux professeurs Louis Rivest et Alain Desrochers dont les opinions et les jugements ont considérablement enrichi la qualité et la profondeur des réalisations et présentations émergeant des ouvrages effectués au LIPPS.

Je tiens également à remercier mes collègues d'étude qui m'ont soutenu durant mes travaux, en particulier (par ordre alphabétique) Jérôme Eustache, François Giguère, Serge Lemire, Laurent Macabies, René Maury, Omar Msaaf, Éric Taillon, au risque d'en oublier quelques-uns.

Pour terminer, je ne pourrais passer sous silence la contribution de tout le corps professoral de l'É.T.S. qui m'a encouragé à pousser plus loin dans mon cheminement de carrière.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS ET SIGLES

API	Application Program Interface	(Programme d'application)
B-Rep	Boundary Representation	(Représentation par frontières)
CAD	Computer Aided Design	(Conception assistée par ordinateur)
CAO	Conception assistée par ordinateur	
CATGEO	CATia GEOmetric interface	(Routines d'interface géométrique CATIA)
CSG	Contructive Solid Geometry	(Géométrie solide par construction)
CSP	Constraint Satisfaction Problem	(Problème à satisfaction de contraintes)
DSG	Destructive Solid Geometry	(Géométrie solide par soustraction)
ERP	Enterprise Ressource Planning	(Planificateur des ressources d l'entreprise)
FAO	Fabrication assistée par ordinateur	
IUA	Interactive User Access	(Langage interactif de programmation)
MRP	Manufacturing Ressource Planning	(Planificateur des ressources de production)
NURBS	Non Uniform Rational B-Spline	(B-spline rationnelle non uniforme)
PDM	Product Data Manager	(Gestionnaire de données de produit)
PDM II	Product Data Management system	(Systèmes de gestion de données techniques)
SIB	Système d'ingénierie de Bombardier	
VPDM	Virtual Product Data Management	(Gestionnaire de données de produit virtuel)
VPM	Virtual Product Model	(Modèle de produit virtuel)

INTRODUCTION

Les produits fabriqués par les industries manufacturières contemporaines sont plus complexes que jamais en plus d'être distribués et vendus mondialement. Les avantages concurrentiels visant la satisfaction accrue des besoins du consommateur ne sont plus essentiellement centrés sur la qualité ou le prix du produit, mais plutôt sur la réduction de la durée de mise en marché des produits [1].

En effet, la compétition croissante force les entreprises à réduire le cycle de développement de leurs produits. Ainsi, toute modification appliquée à un produit dans un marché de grande variété nécessite une gestion efficace de l'information à travers toutes les sphères de l'entreprise. Le grand défi des corporations est donc d'introduire de nouveaux produits sur le marché aussi rapidement que possible en atteignant au plus tôt la production de masse [2].

En fait, la production efficace d'une grande famille de biens ou de produits complexes requiert une très bonne coordination de toutes les données techniques utilisées. Pour des produits tels que des automobiles ou des aéronefs, on peut facilement comprendre la quantité astronomique d'informations à gérer lorsque le concept de ces derniers est modifié pour fins d'optimisation ou pour simplement offrir une configuration personnalisée du véhicule selon le lieu d'utilisation. Devant l'ampleur de la tâche, il est tout à fait évident que les entreprises ont recourt à des systèmes informatiques pour effectuer une gestion efficace et surtout rapide de l'information. C'est la raison pour laquelle l'utilisation des systèmes de gestion de données techniques est devenue un élément clé pour la survie et l'expansion des moyennes et des grandes entreprises dans ce nouveau millénaire.

Les modifications des modèles de Conception Assistée par Ordinateur (CAO) à travers leur évolution sont des données que nous tenterons de gérer dans le cadre de ce mémoire. Il ne s'agit pas de gérer les fichiers des modèles CAO, fonction déjà comblée par les gestionnaires de données de produit (PDM), mais bien des données contenues dans ces fichiers pour l'optimisation des processus de l'entreprise. Nous verrons alors si les PDM sont propices à cette gestion et qu'elle est l'alternative à envisager dans le cas contraire.

Nous poserons d'abord, au Chapitre 1, la problématique soulevée autour de la gestion des modifications des modèles CAO. Nous verrons quels sont les phénomènes qui ont suscité le besoin de gestionnaires de données techniques, plus précisément pour les modèles CAO, et ce, tout au long du cycle de développement d'un produit. Ensuite, nous décrirons au Chapitre 2 le contexte dans lequel se situe la gestion des modifications, entre autres, par rapport aux outils disponibles comme le PDM. Nous ferons une évaluation des forces et des limites de ces outils et nous justifierons la solution pratique que nous avons développée pour résoudre la problématique tel que présente chez la multinationale Bombardier Aéronautique.

Au Chapitre 3, nous décrirons un comparateur de modèles CAO qui constitue la solution développée lors de nos travaux. Nous verrons de quelle manière le comparateur a été conçu, ce qui illustrera déjà les forces et les limitations de l'application. Nous introduirons aussi dans ce chapitre la description des algorithmes de comparaison créés et les paramètres analysés. Cela précèdera l'énumération des différentes étapes à suivre pour l'utilisation du comparateur par un usager. C'est au Chapitre 4 du rapport que l'on trouvera les différents exemples de validation de la fiabilité du comparateur à partir de pièces aéronautiques.

Subséquemment, nous verrons l'interprétation des résultats obtenus pour la validation de l'application, nous passerons en revue les performances du comparateur et nous

exprimerons quelques recommandations, en plus des sujets de futurs développements. Une conclusion des recherches faites sera exposée en dernier lieu.

CHAPITRE 1

LA PROBLÉMATIQUE

Auparavant, les données complexes de fabrication d'un produit devaient être coordonnées avec les systèmes de production de masse des usines afin de fournir une certaine flexibilité aux désirs des clients. Aujourd'hui, chaque produit demande un certain degré de personnalisation, selon ses configurations disponibles. Même les systèmes manufacturiers sont conçus pour la fabrication d'une gamme de produits variés. La flexibilité d'un bien doit être planifiée et mise en considération dès la phase de conception du produit afin que les systèmes manufacturiers envisagés répondent bien au besoin de souplesse des industries [3].

En fait, l'histoire de la production manufacturière peut se diviser en trois grandes étapes [3]. Il y a eu d'abord le *fordisme* ou autrement dit la production de masse. En effectuant une standardisation majeure des quelques produits fabriqués par une usine, on pouvait réaliser des économies d'échelle. Ce système de production favorisait également la génération de larges inventaires.

Par la suite, cette stratégie de production a fait graduellement place à la personnalisation. À cette période de l'ère manufacturière, la variété des produits est devenue l'un des atouts les plus stratégiques des entreprises. Beaucoup d'efforts et de méthodes de travail ont été développés pour la réduction des coûts de production, ce qui inclut la réduction des inventaires.

Finalement, la dernière période dans laquelle nous nous trouvons actuellement est la réduction des cycles de développement pour une introduction rapide des produits sur le marché. C'est ce qu'on désigne couramment par le terme anglais « *time-to-market* » [3]. Il est devenu bien plus intéressant financièrement d'être le premier à lancer un nouveau

produit, dans le but de gagner la plus grande part de marché possible, plutôt que d'attendre que le produit ne soit optimal en terme de qualité et de rentabilité. En effet, même si un bien est appelé à être modifié et optimisé dans son cycle de développement tel qu'illustré à la Figure 1, les statistiques recueillies [4] démontrent qu'une rapide introduction du produit chez les clients favorise son acceptation. Pour compléter cela, un souci particulier s'est développé autant pour le service à la clientèle que pour sa satisfaction personnelle, afin de créer une atmosphère de confiance qui assure la fidélité de l'acquéreur.

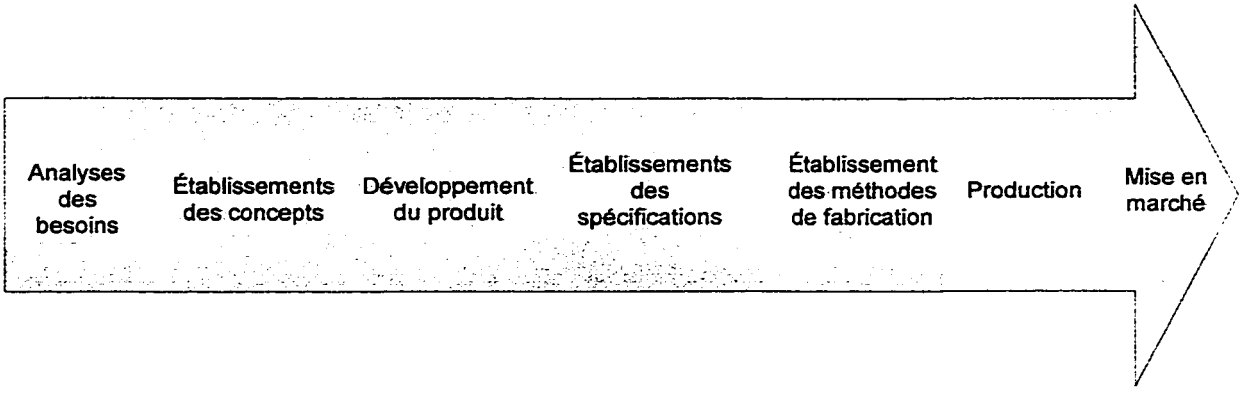


Figure 1 Cycle de développement du produit

Avec la diversification des besoins des consommateurs, l'industrie manufacturière doit s'adapter aux modifications des produits qui surviennent plus fréquemment mais surtout plus rapidement. Il est donc devenu nécessaire pour les entreprises d'effectuer de nombreux retours en arrière dans le cycle de développement de leurs produits pour tenir compte de tous ces changements. Ces retours sont effectués mais ils s'avèrent fort dispendieux.

La CAO a considérablement influencée la vitesse de développement des produits conçus. Les modifications appliquées aux représentations du produit s'effectuent maintenant beaucoup plus rapidement qu'autrefois. Le dessin assisté par ordinateur en deux dimensions était essentiellement utilisé comme médium de documentation statique

pour la description de pièces individuelles faisant partie d'un assemblage. Maintenant, le modèle géométrique en trois dimensions contient une foule de propriétés et d'informations annexées dès sa création. Ces dernières années, les professionnels en la matière ont réellement adopté la notion de modélisation 3D comme générateur de bases de données sur les produits à être conçus [5].

Comme les modèles CAO permettent la modification rapide des concepts établis, des familles entières de produits et de révisions de ces produits peuvent être créées. L'évolution des modèles CAO est si rapide qu'il favorisent la génération accrue de révisions de modèle dont on peut facilement perdre de vue les différences qui les distinguent. Il est primordial pour les usagers de modèles CAO de pouvoir spécifier quelles sont les nuances entre les révisions afin de savoir s'il est opportun d'utiliser une révision particulière du modèle pour une certaine tâche.

Imaginons un ouvrier de fabrication qui désire savoir si les pièces qu'il possède, à une certaine révision, peuvent toujours être machinées selon la gamme d'usinage standard. Il doit s'assurer que la révision des pièces est bel et bien identique à celle utilisée pour la programmation des chemins d'outils concernés, du point de vue usinage. Il suffit simplement que la position d'un mur dans la pièce ait été déplacée pour que la pièce, une fois usinée, soit inutilisable. Par contre, la position des trous qui doivent être percés dans la pièce, après le machinage, ne concerne pas l'ouvrier. Il devra comparer manuellement la révision du modèle de pièces qu'il a en main avec la révision du modèle de la pièce utilisée pour la programmation d'usinage afin d'en découvrir les différences, s'il y a lieu. On constate rapidement que cette méthode de vérification archaïque peut prendre plusieurs minutes, voire des heures, surtout si l'ouvrier possède plusieurs révisions de pièces en main. C'est une situation qui se produit souvent en industrie et à laquelle nous tenterons de fournir une solution.

Le cas particulier que nous avons étudié est celui du processus de gestion des révisions de modèles chez Bombardier Aéronautique. En bref, Bombardier est une entreprise québécoise qui est devenue une multinationale à l'échelle mondiale. Sa filiale, Bombardier Aéronautique, concentre ses activités principalement dans la fabrication d'avions d'affaires, régionaux et amphibies. Comme les aéronefs conçus par l'entreprise comprennent des milliers de pièces, il est primordial que l'avionneur puisse compter sur des gestionnaires informatiques performants, ce qui inclus un gestionnaire de produits et un gestionnaire des modifications. Actuellement, la filiale utilise le logiciel de CAO CATIA sous sa révision 4 pour la conception et la gestion de ses produits.

Ce qu'il faut savoir sur le processus de modification des fichiers CAO de l'entreprise est fort simple. Une fois que le concept d'un sous-produit ou d'une pièce est terminé, vérifié et approuvé, le modèle CAO de cette pièce est dit « publié ». En d'autres termes, le modèle est devenu public et accessible aux utilisateurs, sans toutefois pouvoir être modifié : le modèle est « gelé » ! En cours de développement de l'aéronef, donc de ses composantes, il peut arriver que le concept d'une pièce soit altéré afin d'être optimisé. Comme beaucoup de travail et d'efforts ont été déployés lors de la conception primaire des composantes, celles-ci requièrent habituellement des modifications plutôt qu'un nouveau concept.

Les ingénieurs utilisent la révision initiale ou antérieure du modèle pour en créer une nouvelle révision modifiée. Ainsi, un même modèle ou même pièce existe sous plusieurs révisions. Un protocole du Système d'Ingénierie de Bombardier (SIB) oblige les ingénieurs à identifier les modifications géométriques du modèle par rapport à sa révision précédente en y appliquant une certaine couleur prévue à cet effet. Prenons un exemple : nous avons le modèle géométrique d'un longeron sous sa quatrième révision, la révision D. Après plusieurs études, les ingénieurs de l'entreprise considèrent que les arrondis de la pièce sont sous-dimensionnés pour les charges et contraintes que peut subir la pièce en cas extrême. Ils résolvent le problème en décidant d'augmenter la

dimension des arrondis du longeron. Dans ce cas-ci, ils se serviront de la révision D du longeron pour en créer la révision E. Pour identifier l'évolution des révisions, ils donneront la couleur orangée aux arrondis modifiés, ce qui sera en contraste avec le reste de la pièce de couleur cyan. Ainsi, ceux qui visualiseront la nouvelle révision du modèle sauront à quel niveau la pièce a été modifiée et comprendront la nature du changement rapidement (exemple à la Figure 2).

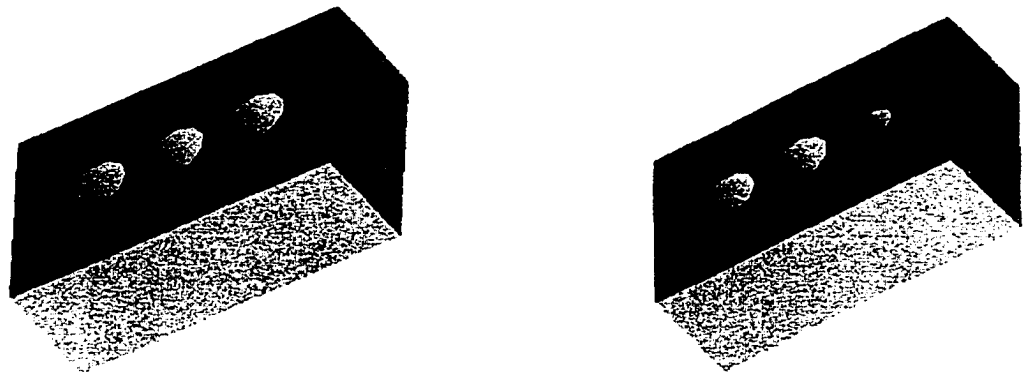


Figure 2 Identification colorée de la modification

En plus de colorer les modifications géométriques du modèle, les ingénieurs doivent également justifier ces changements dans un espace texte qui se trouve à l'intérieur du fichier CAO et qui est prévu à cet effet. Au fur et à mesure que le modèle évolue en révisons, la description des altérations s'ajoutent chronologiquement dans le petit compte rendu des modifications introduites au modèle. Ceci facilite énormément les recherches des utilisateurs qui désirent savoir si la révision d'un modèle correspond à ses besoins.

En pratique, ces deux dernières procédures, c'est à dire l'identification colorée des modifications et leurs justifications dans l'espace texte sont la plupart du temps

inexactes, incomplètes et même omises dans l'entreprise. Différents facteurs tels que les contraintes de travail et de projet favorisent ces omissions.

C'est pourquoi il s'avère fort intéressant de trouver l'outil approprié qui pourrait détecter systématiquement les différences entre deux révisions d'un modèle et les identifier par coloration. Même si un tel outil ne pourrait pas justifier automatiquement la modification, il pourrait néanmoins mettre en évidence le maximum d'informations pertinentes pour orienter la quête de l'utilisateur. La plupart du temps, avec un modèle CAO modifié et dans les meilleurs des cas, on peut avoir rapidement accès au rapport de modifications, l'espace texte du fichier qui comporte la nature ou les raisons pour lesquelles les changements ont été effectués.

Bien entendu, ce rapport de modifications expose les opérations réalisées entre des révisions consécutives d'un modèle. La lacune de cet outil est qu'il ne permet pas de saisir rapidement les différences entre deux révisions non-consécutives d'un modèle. Il peut arriver que certaines révisions du modèle ne consistent qu'en l'annulation de modifications antérieurement apportées. On perd alors ici l'essentiel de ce que l'on cherche, soit la simple différence entre deux révisions d'un modèle.

Un exemple simple est l'utilisation d'une pièce semi finie. Pour des raisons de qualité, un lot de pièces à sa révision D est inspecté. Une fois les pièces vérifiées, on veut les réintroduire dans le processus de fabrication du produit. Par contre, durant les délais d'inspection, la pièce est passée de la révision D à la révision G. Les responsables de la fabrication se demandent alors si les pièces inspectées sont conformes, modifiables ou tout simplement non utilisables pour la nouvelle révision du produit. Ils aimeraient pouvoir comparer la révision G à la révision D de la pièce pour valider son utilisation. Pour cela, ils devront nécessairement passer par les trois sections du rapport de modifications qui séparent les deux révisions, en espérant que ces rapports soient

exacts. C'est ce genre d'exercice de vérification que nous aimerions automatiser à l'aide de l'outil le plus approprié.

CHAPITRE 2

LES OUTILS DISPONIBLES

Actuellement, il y a deux principaux types d'outils informatiques qui peuvent potentiellement gérer la nature de ces modifications. Il y a bien sûr les PDM qui sont déjà conçus pour la gestion des modifications et la dépendance des documents. Il y a également les programmes informatiques d'application qui peuvent être intégrés aux systèmes de CAO utilisés et parfaitement adaptés aux données du modèle. En fait, il existe une série de raisons qui favorisent une solution plutôt qu'une autre et qui, par conséquent, orienteront le type de gestionnaire qui sera conçu dans le cadre de ce projet. L'évaluation de ces raisons fait l'objet de la discussion de ce chapitre.

2.1 Le gestionnaire de données de produit

Le gestionnaire de données de produits, communément appelé PDM, rend disponible une multitude de types d'informations reliées à la conception, la fabrication ou la vente du produit. Cet amas d'informations est exploité de manière à apparaître comme une seule base de données afin d'en favoriser son accessibilité, son contrôle et la gestion des modifications qui surviendront tout au long du développement du produit.

Le PDM est semblable à une banque qui renferme toutes les informations nécessaires permettant la reconstitution d'un produit quelconque. Il y a bien sûr les données techniques qui décrivent géométriquement et physiquement le produit par ses dimensions, son volume, ses matériaux de fabrication, sa masse, etc. Ces données peuvent se trouver sous forme de fichiers texte mais plus probablement en tant que modèles créés par l'entremise d'un logiciel de dessin ou de conception assistée par ordinateur. D'autres types d'informations plus détaillées peuvent être emmagasinées

dans cette banque, tel que la couleur de la couche de peinture finale, les analyses structurales par éléments finis du produit ou les études sur la durée de vie des prototypes. Même les mémos et les notes qui se transmettent entre différents services de la corporation peuvent être informatisés afin d'être stockés à l'intérieur du PDM. Ceci permet d'expliquer aux utilisateurs ultérieurs du gestionnaire pour quelles raisons une option particulière a été adoptée plutôt qu'une autre lors de la conception du produit.

Toutes ces informations ont pour but l'orientation plus efficace des décisions prises par les intervenants de l'entreprise. Par conséquent, cette banque n'est pas seulement disponible aux ingénieurs de la compagnie mais également aux ouvriers, aux administrateurs, aux vendeurs et à tous ceux qui participent au développement et à la mise en marché du produit. Le gestionnaire de données de produits présente donc sa base de données de façon claire et uniforme afin qu'elle soit compréhensible et utilisable par les intervenants provenant des différents secteurs de l'entreprise.

2.1.1 Justification du PDM

La raison qui suscite l'utilisation des PDM est une conséquence directe du progrès de la technologie. Les ordinateurs nous permettent d'effectuer des milliers d'opérations beaucoup plus rapidement que manuellement et ce sans erreur. Par conséquent, le papier, qui sert encore de médium pour le transfert d'information (dessins de détail, mémos, cahier des charges, contrats, caractéristiques techniques du produit) est devenu très lent par rapport aux systèmes informatisés. Il devient donc intéressant d'éliminer graduellement le papier pour le remplacer par les systèmes informatiques (même au niveau de l'archivage de documents).

2.1.2 Les rôles du PDM

Les PDM offerts par les fabricants actuels diffèrent sensiblement entre eux. En effet, le gestionnaire de données de produits contient une foule de fonctions qui s'appliquent à une certaine envergure des processus de l'entreprise, ce qu'on verra plus profondément à l'Annexe I du document. On peut néanmoins énumérer les multiples rôles de base que l'on peut s'attendre d'un PDM.

Les fonctions principales du gestionnaire se résument ainsi [6]:

- Collecter et emmagasiner les documents,
- Gérer les dépendances (relations entre les documents),
- Gérer les modifications,
- Gérer les révisions (configurations),
- Extraire et distribuer les documents.

L'impact du PDM dans les processus d'une entreprise prend toute son ampleur si l'on considère à la Figure 3 à quel moment du cycle de vie du produit le gestionnaire est le plus utilisé.

Dans cette même figure, on visualise la contribution en terme de temps d'utilisation des différents systèmes de gestion de données techniques par rapport au cycle de développement d'un produit, ce qui inclut le ERP (Planificateur des ressources de l'entreprise) et le VPDM (Gestionnaire de données de produit virtuel).

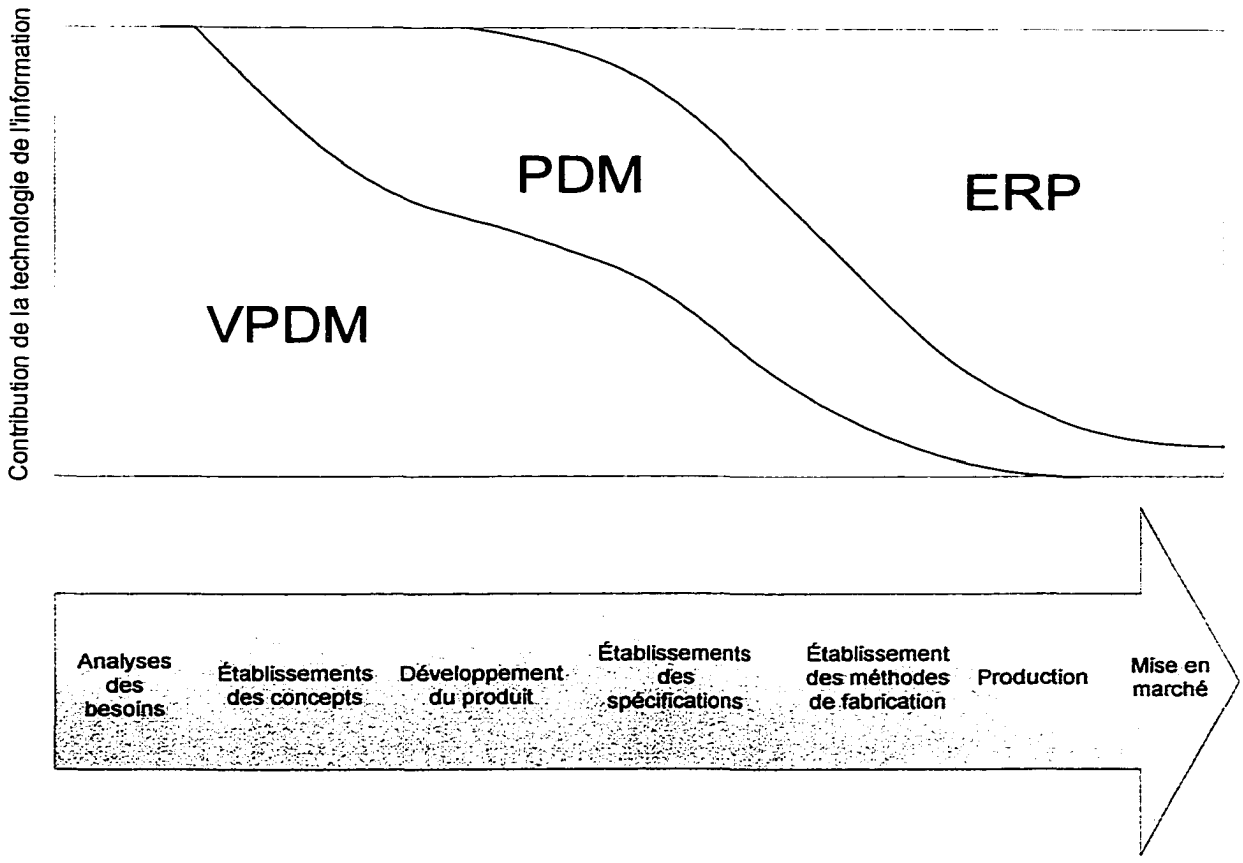


Figure 3 Contribution de la technologie de l'information

2.2 Gestion des modifications

Le point de mire de ce document concerne essentiellement la gestion des modifications des données de CAO. Pour cela, une dernière distinction reste à faire entre une modification, une révision et une configuration. En général, on parle de révision ou de révision pour une modification portée à des pièces ou des documents distincts. Une configuration est un ensemble de pièces ou de documents qui forme un assemblage à peu près semblable à l'ensemble original de pièces ou de documents tel que produit pour un marché particulier.

Par exemple, une entreprise fabrique des véhicules tout terrain. Au fur et à mesure que des essais sont réalisés, on modifie la géométrie des pièces afin qu'elles puissent supporter les contraintes imposées par l'utilisation du véhicule. Chaque modification induit donc une autre révision des pièces optimisées. Par contre, si l'on utilise des pièces différentes dans le cas où le véhicule sera utilisé dans le désert africain plutôt que dans les régions arctiques, l'ensemble modifié forme une configuration.

2.2.1 Les rôles particuliers du PDM

Ce qui est important de noter à ce stade-ci, en terme de modifications, est que la gestion des révisions et des configurations d'un produit devient une tâche très lourde à accomplir pour les assemblages complexes, d'où l'utilité d'un gestionnaire informatique tel que le PDM. Dans notre cas, la gestion des modifications est liée à celle des dépendances entre documents afin que ceux-ci soient réellement gardés à jour. C'est pourquoi un atout intéressant pour un PDM est le signalement automatique des modifications. Quand un document est altéré, le gestionnaire peut assigner un avertissement particulier à tous les documents (membres d'un assemblage ou d'une

configuration) qui sont affiliés au document modifié[7]. Le PDM devrait donc rendre possible le traitement des modifications et ses dépendances en une seule tâche.

Une possibilité intéressante des PDM pourrait être la mise à jour automatique du contenu des documents [6]. Pour cela, le PDM doit être capable d'interpréter sa base de données plus loin que juste en tant que simples fichiers ou paquets d'octets. La vaste étendue de types d'information gérées par les PDM complique énormément la conception d'un tel interpréteur de données. De plus, cette fonction intéressante coûterait chère car il faudrait structurer le format de tous les types de documents gérés de l'entreprise pour uniformiser l'entrée des données. Cependant, des modules informatiques peuvent être intégrés au gestionnaire de données ou au logiciel de CAO pour combler la fonction suggérée. C'est ce qui fera d'ailleurs le sujet du Chapitre 3 du présent document. Il est à noter que le travail qui sera présenté n'est qu'une modeste contribution à la fonction anticipée.

En somme, les fonctions particulières des PDM ayant pour but la gestion des modifications forment le cœur, en terme de performance, du gestionnaire de données. Si le système n'est pas capable de gérer toutes les interdépendances susceptibles d'être affectées par une modification, la tâche devra être effectuée manuellement par ses utilisateurs, ce que l'on essaie justement d'éviter afin de ne permettre aucune erreur humaine. C'est d'autant plus crucial lorsque la durée de développement du produit en question est constamment réduite, ne laissant ainsi qu'une très faible marge de manœuvre.

Instinctivement, nous sommes portés à choisir le PDM comme gestionnaire de nature des modifications, et pour de bonnes raisons. Il effectue déjà la gestion des dépendances, des configurations et des révisions de documents, ce qui comprend ceux reliés à la CAO [6]. La notification des modifications aux documents liés ensemble est l'un de ses modules de gestion. Cette association des documents permet une bonne prise

de conscience par les ingénieurs des changements effectués au produit [8]. Puisque ces modules du PDM s'appliquent à différents documents, ceci lui favorise l'intégration du gestionnaire de nature des modifications.

Comme le PDM est déjà présent dans la gestion des références de rapports de modifications, ceci initie dès lors cet effort de recherche en ce qui a trait aux causes des modifications appliquées au modèle géométrique du produit [9]. C'est à ce stade que s'arrête l'étendue des fonctions des outils présentement disponibles. En ce qui concerne spécifiquement le modèle géométrique, il est tout à fait possible d'adapter le PDM pour qu'il puisse déchiffrer les données propres à une révision d'un modèle de CAO et à en faire la comparaison avec une autre révision.

Notons qu'il existe quatre principales représentations CAO d'un produit [10] :

- La représentation par limite,
- la représentation par historique,
- la représentation surfacique et
- la représentation paramétrée généralement décrite par des programmes.

La représentation par les limites ou les frontières (B-Rep) comprend des données géométriques et topologiques simples pour décrire le produit. Les points, les lignes, les courbes et les surfaces sont quelques unes des entités comprises dans cette représentation. De son côté, la représentation par historique est plus souvent organisée sous forme de réseau ou d'arbre de modélisation ou de construction (CSG). Aux feuilles de cet arbre de modélisation se trouvent les volumes primitifs paramétrés, et aux nœuds, les opérations.

La représentation mathématique communément appelée modélisation surfacique est surtout utilisée pour la modélisation de courbes et de surfaces. Ces dernières sont

définies sous forme de polynômes dont l'usager a le contrôle explicite ou implicite des paramètres (splines, Bézier, B-splines, NURBS). Pour terminer, la représentation paramétrée dépend essentiellement du programme conçu pour la génération d'une famille de pièces.

2.3 Les techniques existantes

2.3.1 La représentation B-Rep

Plusieurs algorithmes ont été conçus pour différents traitements de données de modèles CAO. Dans le domaine des représentations B-Rep, les chercheurs [11] se sont heurtés à une difficulté : avec les mêmes entités géométriques, il est possible d'obtenir deux modèles différents. Ceci repose essentiellement sur les contraintes utilisées pour définir les entités et leur topologie. Par conséquent, des algorithmes ont été conçus pour normaliser les structures des données de modèles CAO sous représentation B-Rep dans le but d'en faciliter l'analyse. Dans les exemples énoncés par Leinen, Jung et Gardan [11], les données de ces modèles produits sont organisées sous forme de graphe. Suivant cette disposition, on peut comparer des révisions de modèle, via leur graphe, pour y déceler les différences. Brièvement, la méthode de normalisation consiste, en premier lieu, à l'établissement de priorités à un système de contraintes, et en deuxième lieu, à la réorganisation du graphe initial par le remplacement et l'ajout de contraintes prioritaires.

D'autres algorithmes existent également pour la comparaison des modèles produits. Certains algorithmes sont utilisés par la méthode des problèmes à satisfaction de contraintes (CSP). Cette stratégie établit d'abord un graphe CSP des révisions du modèle : il s'agit de remplacer la valeur des nœuds et des arêtes de l'un des graphes (la révision actuelle) par la plage des valeurs présentes dans le second graphe normalisé (la révision précédente). À l'aide d'un algorithme de recherche, on tente de contraindre

chaque variable du graphe CSP à une seule valeur correspondante du graphe initial. Si cet objectif est atteint, on en déduit que les graphes sont identiques. Dans le cas contraire, on peut illustrer les variables trouvées différentes et approfondir notre étude comparative pour déterminer la nature des différences entre les deux révisions du modèle produit. Les auteurs [11] affirment que l'algorithme de recherche de consistance en avant est celui qui permet d'obtenir le meilleur rapport coût/efficacité.

Malgré tout, après la localisation et l'analyse des différences entre les deux modèles, ou leur graphe normalisé, il n'en demeure pas moins qu'il faut pouvoir interpréter ces différences en opérations simples. Est-ce que les données altérées correspondent à un perçage, un chanfrein, un chemin de clé sous-dimensionné ? Il faut une fois de plus utiliser des algorithmes avancés pour interpréter les résultats obtenus en opérations compréhensibles par les usagers du PDM. Les techniques d'interprétation de ces données s'apparentent énormément à celles utilisées pour la localisation des caractéristiques de forme. Ces techniques ou algorithmes d'extraction des caractéristiques géométriques sont très recherchées pour traduire de simples données B-Rep en gammes d'opération d'usinage dans le secteur de la Fabrication Assistée par Ordinateur (FAO).

Bien que ce ne soit pas des données de type FAO que nous désirons extraire des modèles, le principe des techniques énoncées peut être adapté pour la comparaison de modèles produits. En fait, ces algorithmes peuvent être utilisés de deux manières. On peut tout d'abord les utiliser avant la comparaison des modèles produits. Cela consiste à comparer des caractéristiques de forme (chanfrein, perçage) que nous devrons définir au préalable. Sinon, on peut les utiliser après la comparaison des modèles, ce qui revient à interpréter les différences extraites en caractéristiques de forme. Aucune des deux alternatives ne garantit un résultat infaillible car chacune est plus appropriée pour un type particulier de modification entre les révisions du modèle produit. C'est la raison pour laquelle beaucoup de recherches s'effectuent encore sur le sujet. Ces techniques

d'interprétation des données B-Rep en caractéristiques de forme gèrent plusieurs données extraites du modèle produit telles que les liaisons face - contours, contours – arêtes et face - face des modèles, la concavité - convexité des arêtes et les intersections droite – droite ou droite - plan [12].

2.3.2 La représentation CSG

Si les modèles sont représentés différemment (arbre CSG, modélisation surfacique), les techniques de normalisation, d'extraction, d'analyse et de comparaison doivent être adaptées à cet effet. Pour la représentation par arbre de construction, des algorithmes particuliers sont également utilisés pour déceler la nature des modifications. Il y a deux principales causes qui suscitent le besoin de ces algorithmes de haut niveau. D'abord, les propriétés des relations de concaténation des modèles CSG font qu'une même expression d'un produit peut avoir plusieurs formulations différentes : par exemple, $(a - b) - c = a - (b \cup c)$. La deuxième cause est que le même produit peut être conçu avec des ensembles différents de primitives volumiques.

Encore dans ce cas-ci, les techniques existantes pour comparer ou analyser certaines particularités des modèles produits sous forme CSG se rapprochent énormément des études faites sur la détection des caractéristiques de forme. Nous avons retenu deux méthodes utiles pour l'extraction des différences entre deux révisions d'un modèle CSG [10]. Il y a en premier lieu la détection des caractéristiques de forme à l'aide des axes principaux. Il s'agit d'annexer à chaque primitive du modèle un certain nombre d'axes principaux selon le type de la primitive et de ses paramètres de définition géométriques. Par la suite, on tente de regrouper dans un même sous-arbre les primitives partageant les mêmes axes et semblant constituer une des caractéristiques de forme définies antérieurement par la méthode. Tout comme pour les modèles B-Rep, on constate que la méthode cherche à extraire les caractéristiques de forme contenues à l'intérieur du

modèle pour finalement les comparer (c'est à dire les sous-graphes) entre chacune des révisions du modèle produit.

La deuxième méthode consiste en la transformation de l'arbre de modélisation CSG en un arbre de soustraction appelé en termes anglais « Destructive Solid Geometry » (DSG). En résumé, on insère autant d'opérateurs de soustraction que possible dans l'arbre du modèle produit afin d'obtenir un arbre DSG, c'est à dire un arbre qui part d'une pièce brute surdimensionnée à laquelle on soustrait consécutivement des volumes primitifs jusqu'à l'obtention de la pièce finale, ce qui est une analogie très rapprochée aux procédés d'usinage. Une fois cette transformation effectuée, les deux arbres peuvent être comparés de manière pertinente pour en extraire les différences. On remarque bien que les deux méthodes énoncées sont quelques peu semblables car elles cherchent d'abord à restructurer l'arbre CSG avant d'initier le processus de comparaison.

2.4 Le choix du support

Pour réaliser un prototype de gestionnaire de la nature des modifications, il est primordial pour nous de connaître sur quel type de système il devra être conçu. Si nous devons considérer les PDM comme support du gestionnaire de la nature des modifications, il faudra considérer la gestion de toutes ces méthodes et algorithmes que nous venons de décrire, sachant qu'un PDM est idéalement utilisé pour la gestion d'informations de systèmes informatiques hétérogènes. Pour ainsi dire, dans notre cas, le PDM doit être adapté au type de structure de données établie dans le modeleur CAO (CATIA V4/V5, ProEngineer, SolidWorks, etc.). Nous sommes encore loin de pouvoir implanter techniquement un tel module, du niveau de l'intelligence artificielle dans un PDM, sachant que nous n'avons même pas encore abordé l'analyse des données du modèle produit sous ses représentations mathématique ou paramétrée. Même une

solution orientée vers la normalisation IGES ou STEP des modèles est déconseillée. La représentation normalisée de ces modèles n'est pas compatible au contexte d'évaluation, à savoir où se trouvent les modifications des modèles et comment ont-elles été introduites. D'ailleurs, la fiabilité de l'analyse des modèles sous ces deux normalisations (IGES,STEP) décroît avec l'accroissement de la complexité des modèles conçus.

Conséquemment, nous recommandons le modeleur CAO comme support du gestionnaire de la nature des modifications. Bien entendu, un programme d'application conçu pour la gestion de la nature des modifications d'un modeleur en particulier limite l'application à être utilisée à ce modeleur de CAO. Par contre son utilisation ouvre la porte à une foule d'avantages qui justifient notre recommandation.

Premièrement, si le programme d'application (API) est conçu dans le modeleur, il est adapté parfaitement à la base de données du modeleur, ce qui comprend le mode de représentation des modèles produits (B-Rep, CSG, mathématique, paramétrée) et sa structure de données. On peut subséquemment cerner quelles seront les méthodes et algorithmes nécessaires à la comparaison des révisions du modèle produit. Ensuite, il n'est plus indispensable de normaliser la représentation du modèle produit afin d'en comparer les révisions, car la procédure de modification de la révision peut, à elle seule, fournir une foule d'indications sur la nature (les raisons) des modifications. Dans le cas d'un gestionnaire de la nature des modifications implanté au PDM, ces précieuses informations étaient perdues à l'étape de normalisation de la représentation des révisions. Il s'agissait de les retrouver par analyse, si possible.

Troisièmement, dans l'éventualité où le modeleur favorise l'utilisation de caractéristiques de forme pour la création du modèle produit, il ne sera pas nécessaire de retrouver ces caractéristiques par l'analyse des données de la représentation car elles font déjà parti de la base de données [12]. Finalement, les API sont beaucoup plus

flexibles que les module de gestion des PDM [3]. Sachant que les bases de données des modèles CAO s'enrichissent de plus en plus en informations hétérogènes, il s'avère plus efficace et plus économique de léguer le gestion de la nature des modifications au modeleur CAO. Les bases de données des modeleurs sont obligatoirement développées aussi rapidement que la complexité des données qu'elles contiennent. Ainsi, l'API peut également s'adapter rapidement à la technologie croissante. On ne peut s'attendre à une vitesse de réaction aussi rapide de la part du PDM ou de l'un de ses modules car il est développé séparément du modeleur, sans oublier qu'il n'est pas fondamentalement exclusif au domaine de la CAO. On ne touche pas encore tout ce qui concerne la révision des systèmes et leur format de fichier, qui compliquent l'adaptation éventuelle du PDM à la base de données du modeleur.

Il est tentant de développer des applications à partir des fonctions du PDM sa complexité ne cesse de s'accroître afin de combler le plus de types d'entreprises et d'informations possibles. C'est d'ailleurs la raison pour laquelle les entreprises optent pour des solutions totales [13]. Cette stratégie favorise l'emploi de plusieurs logiciels simultanément pour éviter la désuétude des PDM ou de ses fonctionnalités. Ces solutions sont moins chères et plus faciles à interpréter par les utilisateurs. Du même coup, l'utilisation de l'API comme gestionnaire de la nature des modifications met en évidence le rôle de liaison des documents qui est propre et spécifique au PDM.

C'est d'après cette analyse de la technologie actuellement disponible que nous avons adopté la réalisation d'un comparateur de modèles CAO à l'intérieur d'un modeleur (CATIA V4) plutôt que d'annexer à un gestionnaire de données de produit (Enovia, Metaphase). La description de ce comparateur de modèle est décrite à la section suivante.

CHAPITRE 3

LE COMPARATEUR DE MODÈLES CAO

Pour concrètement identifier les modifications entre les révisions d'un produit, nous allons d'abord simplifier le problème en repérant simplement les différences entre deux modèles CAO, toujours du point de vue géométrique. Chaque modèle CAO représente une révision unique du produit. Avant tout, il est nécessaire de comprendre comment sont gérées les données d'un modèle CAO, dans ce cas-ci à l'intérieur du logiciel CATIA, à la révision 4. Une fois la base de données analysée, il est possible pour nous de concevoir un prototype de comparateur de modèle, tel que celui qui est décrit dans ce chapitre.

3.1 Gestion des données CAO

Lors de la conception d'un produit à l'aide d'un logiciel de CAO, plusieurs fonctions sont disponibles pour la réalisation de la géométrie du produit ou plus probablement d'une composante du produit. À l'aide de points, de lignes, de surfaces ou de paramètres connus, on peut déjà créer les différentes parties de la composante. Ce qui est commun aux logiciels de conception actuels, c'est qu'il est possible de créer une même géométrie de manières différentes. Par exemple, pour créer le cube de la Figure 4, nous pouvons d'abord créer les six plans infinis qui le contiennent pour finalement créer un volume et un solide limités par ses plans. Nous pouvons également créer six points à l'aide desquels nous créons les lignes, puis les surfaces, le volume et finalement le cube. Nous pouvons enfin utiliser une fonction CUBOÏDE, PARALLEPIPÈDE ou PRISME pour créer rapidement la géométrie en nous servant de quelques paramètres. Quoiqu'il en soit, le logiciel se sert de ces informations pour créer une ***primitive***, c'est à dire une sous-composante géométrique de la composante finale.

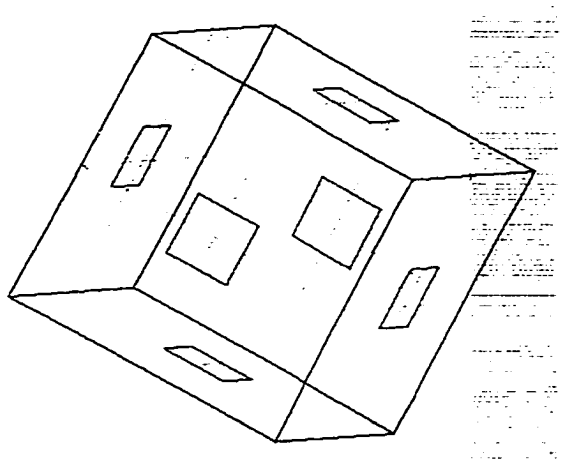


Figure 4 Exemple de création d'un cube

Ainsi, on se sert d'une série de primitives qui formeront le modèle géométrique de la composante. Des opérations sont utilisées pour modifier ou concaténer les primitives afin qu'elles deviennent un tout représentant bien la composante finale. L'addition, la soustraction et l'intersection entre deux primitives sont les opérations booléennes les plus connues et utilisées. Certaines opérations créent elles-mêmes leur primitive : c'est le cas pour un arrondi et l'exécution d'un chanfrein. En fait, le modèle CAO est une suite de primitives ayant subies des opérations de modélisation. On peut schématiser les primitives de la composante et les opérations effectuées par un graphique appelé ***arbre de modélisation,*** tel qu'illustré à la Figure 5.

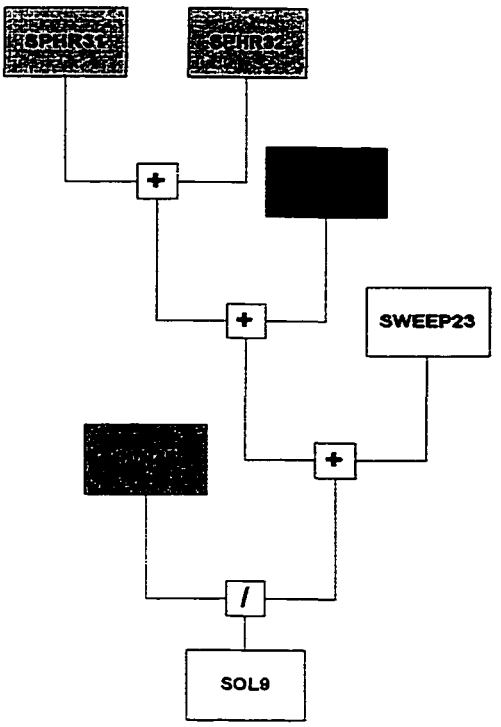


Figure 5 Arbre de modélisation

C'est une fois que l'on a toutes les composantes formées, par leur arbre de modélisation, que l'on peut les assembler en un tout pour obtenir le produit final.

Inclus au logiciel CATIA, on retrouve une multitude d'outils pour concevoir des applications informatiques adaptées. Ceci comprend le langage de programmation interactif (IUA) et des routines d'interface pré établies (CATGEO). Ces outils peuvent permettent, entre autres, de concevoir une application de détection et d'identification des modifications, donc des différences entre les révisions d'un modèle produit. Notre première approche fût de comparer les propriétés physiques de deux révisions de la composante. Il est possible grâce aux routines CATGEO d'effectuer certaines analyses du modèle étudié. On peut donc comparer le volume et la surface des deux modèles CAO. Par contre, ces informations sont loin d'être satisfaisantes : en effet, elles ne servent qu'à nous indiquer si les modèles sont différents ou pas, c'est tout. On ne sait

rien quant à la nature des différences. De plus, les résultats peuvent s'avérer trompeurs. Même si le volume et la superficie de deux modèles sont identiques, ces modèles peuvent différer entre eux; on a qu'à penser à des pièces gauche et droite d'un avion, une aile par exemple.

Une autre alternative possible est d'analyser plus localement les deux modèles. On peut y arriver de deux manières. On peut tout d'abord extraire la surface mouillée ou le modèle filaire des modèles pour comparer leurs entités une à une. De cette manière, il sera possible d'identifier plus précisément la localisation des différences. Cependant, cette méthode pose des problèmes d'ordre pratique. En premier lieu, le genre de modèles que l'on désire comparer comporte généralement des centaines d'entités de type ligne, courbe (pour la représentation filaire) ou surface (pour la représentation surfacique), ce qui demandera un temps d'analyse considérable. En deuxième lieu, plus les entités sont complexes, plus il y a de paramètres pour les caractériser, donc à analyser. Si l'on prend l'exemple d'une ligne dans CATIA, il existe huit paramètres pour la décrire : les trois coordonnées de son point d'origine, les trois coordonnées du vecteur qui définit sa direction et deux paramètres définissant la ligne elle-même, ce qui inclut son point d'arrivée. Pour une courbe, le nombre de paramètres peut monter à la vingtaine et pour une surface, près de trois cents paramètres peuvent la décrire, tel que nous l'avons expérimenté.

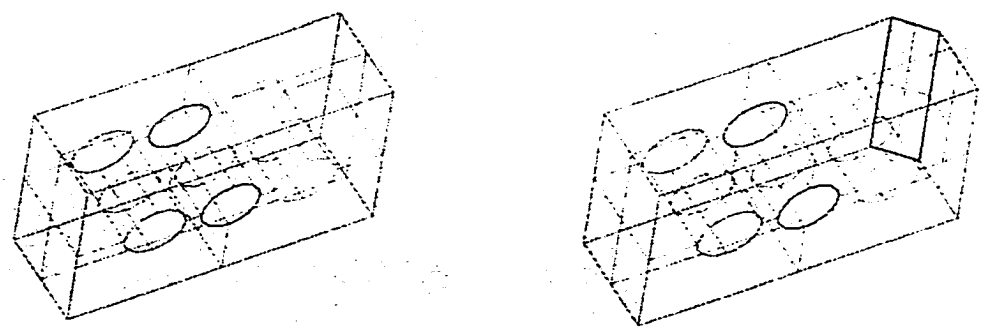


Figure 6 Comparaison de représentations B-Rep

Comme désavantage en troisième lieu, il faut pouvoir comparer les entités correspondantes entre chacun des modèles. Ces entités ne sont pas identifiées nécessairement de la même manière ; il faut obligatoirement faire une comparaison d'une entité du premier modèle à toutes les entités du deuxième pour trouver sa correspondante, si elle existe, bien entendu. Même les relations (topologie) entre les éléments doivent être analysées pour en vérifier la similitude entre les deux modèles [11]. Ce sont de nouvelles contraintes en terme de temps d'analyse et de complexité. Finalement, en dernier lieu, bien que les différences peuvent être détectées et localisées, il n'en demeure pas moins qu'on ne connaît pas la nature de la modification. Effectivement, si l'on trouve qu'une surface à angle apparaît comme différence dans notre géométrie, cela ne nous dit pas encore s'il s'agit d'un chanfrein, d'un filet à grand rayon ou de la soustraction d'une primitive de type PRISME tel qu'on l'observe aux Figures 6 et 7. Or, cette information est primordiale, ne serait-ce que pour comprendre et suivre les intentions du concepteur lors de l'émission de la nouvelle révision.

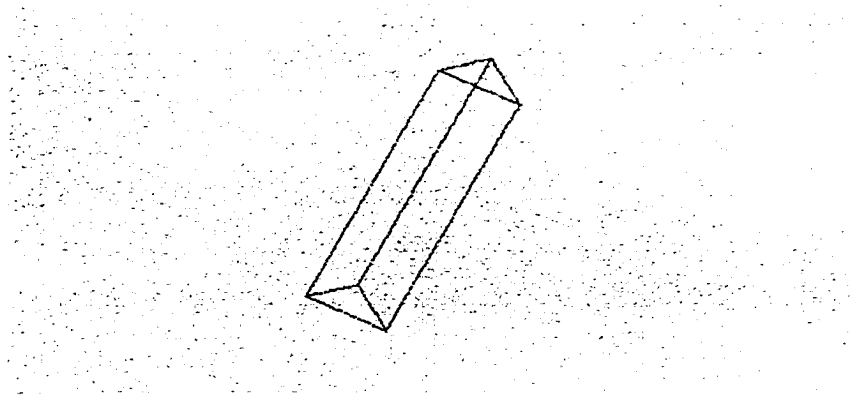


Figure 7 Résultat d'une comparaison de B-Rep

C'est un peu le même problème auquel on fait face si l'on utilise une autre alternative, soit la soustraction booléenne entre les deux modèles (exemple à la Figure 8). Certes, on identifie peut-être quelques différences et leur localisation. Par contre, la nature de la modification nous reste inconnue. À la Figure 9, on remarque bien que des morceaux de

solide isolés dans l'espace n'en disent pas très long sur le but de la modification. En plus, un effet très néfaste de cette méthode est qu'elle peut éliminer tous les indices d'une modification En effet, il suffit simplement que le deuxième modèle soit entièrement contenu dans le premier modèle pour qu'à la soustraction, aucun signe apparent de différence n'existe.

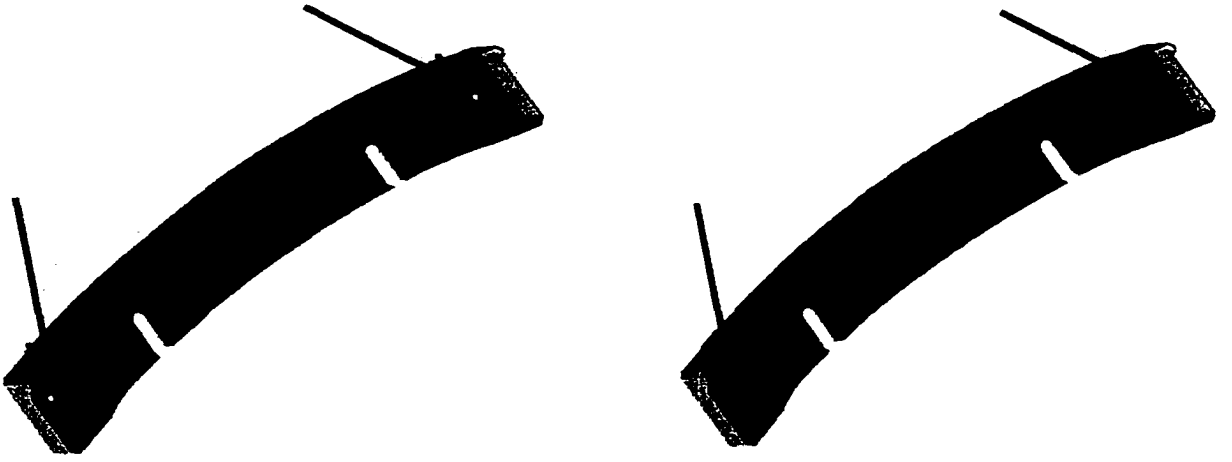


Figure 8 Deux modèles CAO sujets à une soustraction booléenne

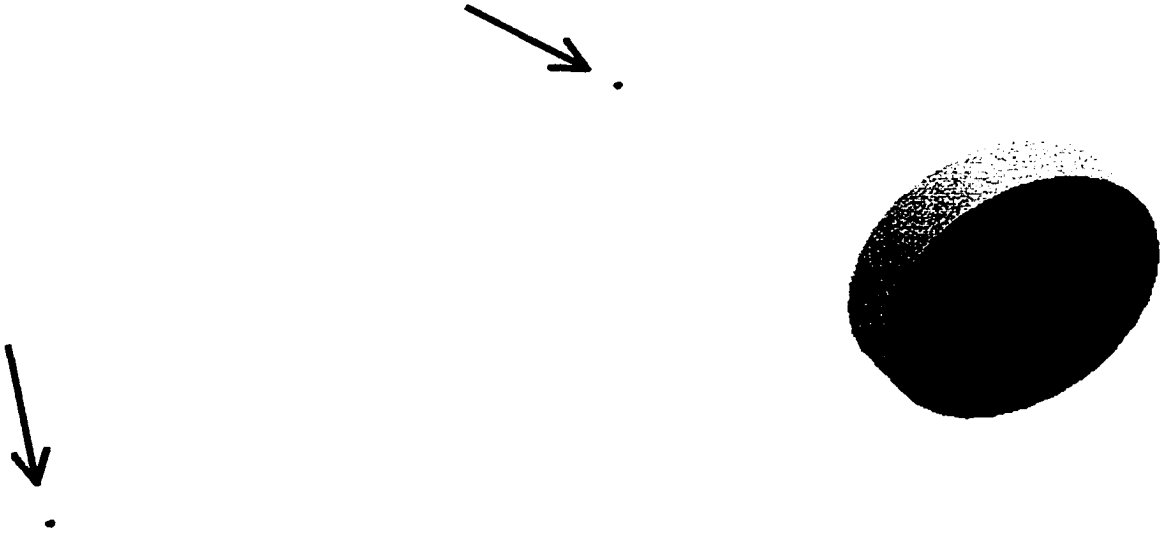


Figure 9 Résultat de la soustraction de deux modèles CAO et agrandissement

L'autre manière qui nous permet d'analyser plus localement les modèles, soit la stratégie qui a été retenue dans ce document, est la comparaison de chaque primitive des modèles. Tout comme pour les autres méthodes, celle-ci comporte également des avantages et des inconvénients. Du côté des avantages, les modèles comportent beaucoup moins de primitives que d'entités que l'on qualifie de primaires (points, lignes, courbes). Ensuite, ces primitives ne nécessitent pas l'analyse de toutes les entités qui les composent mais plutôt des quelques paramètres qui les décrivent. D'ailleurs, pour chaque type de primitive ainsi que pour les éléments géométriques, des routines CATGEO existent déjà pour en extraire les quelques paramètres inhérents. Finalement comme avantage, on dégage très précisément la nature de la modification du modèle puisqu'on peut déjà l'associer à un type de primitive.

Du côté des inconvénients, il faut reconnaître que pour les primitives également, leur identification peut différer dans chacun des modèles. Il est donc nécessaire de comparer une primitive d'un modèle avec toutes les primitives du second modèle afin de retrouver l'élément correspondant. Heureusement, cette recherche est plus orientée que celle des entités géométriques primaires car on sait en partant quel type de primitive l'on veut comparer.

Un autre désavantage émerge de la méthode. Même si toutes les primitives entre deux modèles sont identiques, les opérations utilisées et l'ordre dans lequel elles sont utilisées peuvent produire des modèles complètement différents. Ainsi, il faudra donc établir une manière de vérifier la structure des arbres de modélisation pour bien identifier les modifications présentes. On verra plus tard qu'il faudra créer des astuces de validation pour s'assurer que les résultats de comparaison soient exacts.

3.2 Application utilisée

L'application de détection des différences entre deux révisions d'un modèle est essentiellement basée sur la comparaison des primitives de chacun des modèles CAO. Dans les lignes suivantes, nous expliquons de quelle manière l'application a été conçue.

3.2.1 Gestion des données de l'application

Durant tout le processus d'organisation des données que nous manipulons (primitives, opérations, paramètres), nous utilisons des outils de gestion de piles que nous avons conçu. Il va alors nous être possible de sauvegarder des entités quelconques dans ces piles à l'aide des sous-routines EMPILER et ENFILER que nous avons programmé. Nous tenons à clarifier qu'empiler un élément consiste à ajouter cet élément au début de la pile tandis que d'ENFILER un élément correspond à ajouter un élément à la fin de la pile. Ces fonctions sont schématisées à la Figure 10.

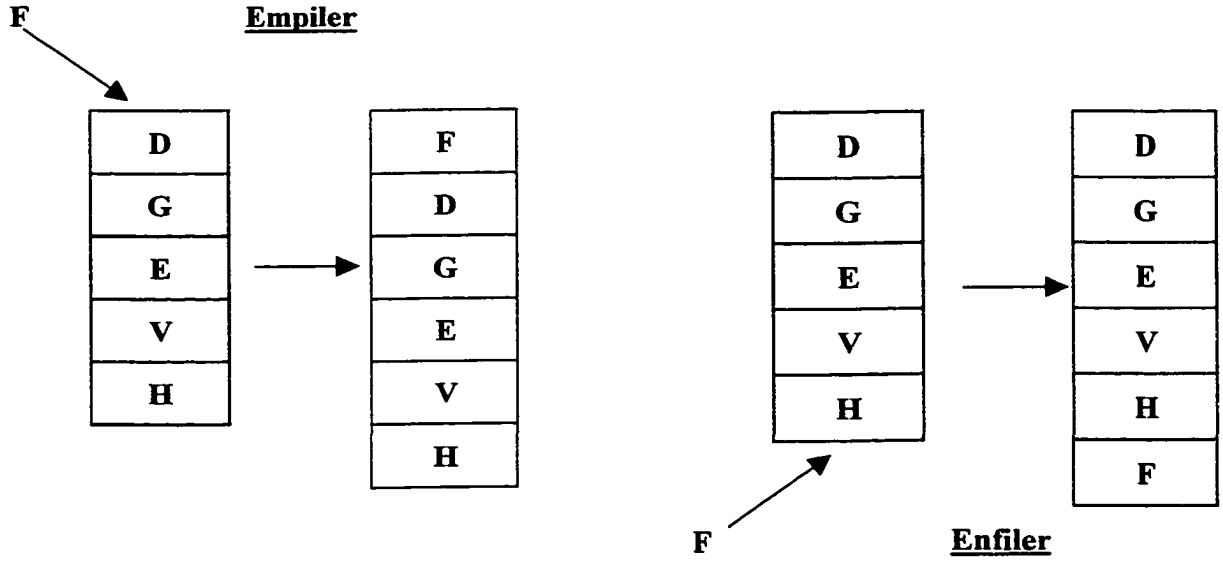


Figure 10 Fonctions de gestion des piles

Nous conservons le même principe en ce qui concerne les fonctions de gestion de piles DÉPILER ET DÉFILER. Ci-dessous est illustré un aperçu de la programmation de la fonction EMPILER.

3.2.2 Sélection des modèles

Il est important de rappeler que la création de l'application de comparaison doit s'harmoniser avec les habitudes de travail des intervenants de l'entreprise, c'est à dire les ingénieurs de Bombardier Aéronautique. Comme dans le logiciel CATIA il est possible de travailler avec plusieurs modèles dans la même session de travail, il est trivial que l'application puisse fonctionner dans cet environnement. Par conséquent, la sélection des modèles à comparer se fait par l'utilisateur, à l'aide de la souris et de quelques outils d'aide (la touche F2) après avoir ouvert les fichiers. La surintensification des modèles choisis (exemple à la Figure 11) ainsi qu'une option d'annulation en cas d'erreur permet à l'usager de vérifier sa sélection. Seuls les deux modèles à comparer doivent être présents dans la session de travail.

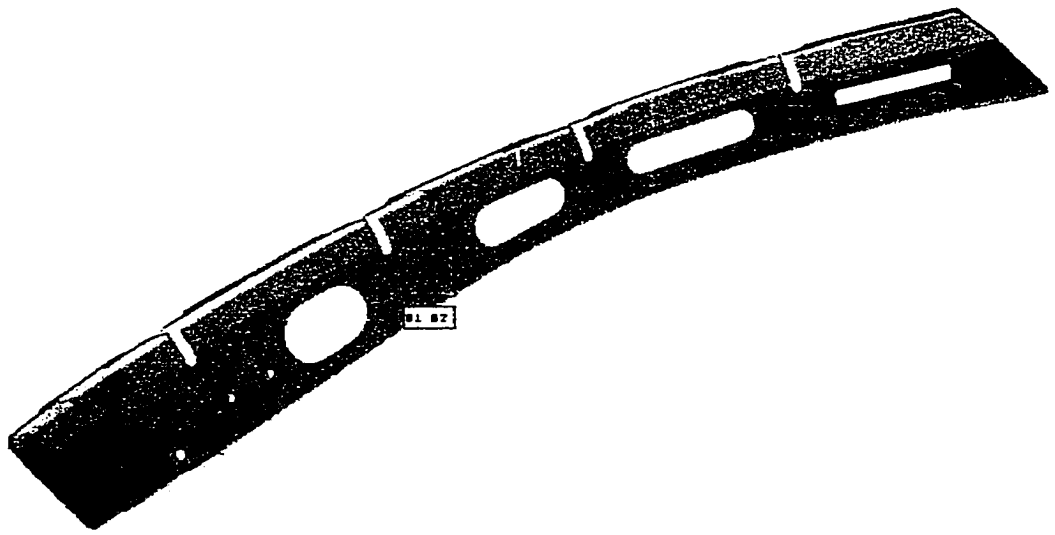


Figure 11 Sélection des modèles

3.2.3 Extraction des données

3.2.3.1 L'arbre CSG

Une fois que les modèles appropriés ont été sélectionnés, on peut extraire toutes les informations pertinentes des modèles à l'aide des routines CATGEO disponibles. Parmi ces informations, il y a les arbres de modélisation et la liste des primitives. L'arbre de modélisation est formé d'une matrice de trois colonnes : la première colonne contient les opérations de modélisation tandis que les deux dernières contiennent les éléments ayant subis cette opération. Ces éléments peuvent être une primitive ou une concaténation de primitives ayant antérieurement subies des opérations (des sous-ensembles). Chaque ligne de la matrice représente une étape de la modélisation de la pièce. On retrouve ainsi, dans l'ordre chronologique, toutes les opérations nécessaires pour créer à nouveau la composante ou le solide, de haut en bas dans la matrice, tel qu'illustré à la Figure 12.

1.	1	45123	23460
2.	1	348	2316
3.	1	-1	-2
4.	3	28213	-3
5.	14	-4	6328

Figure 12 Matrice d'un arbre CSG

À l'aide de la documentation disponible sur les routines CATGEO, on peut bien entendu traduire les données de la matrice. Tel qu'on l'observe dans la matrice de la Figure 12, dans la première colonne, chaque nombre correspond à une opération particulière. Ainsi, aux trois premières lignes de la matrice, on retrouve l'opération **1**

qui symbolise une union booléenne. L'opération **3** est une soustraction et la **14** représente une poche.

Pour ce qui est des deux dernières colonnes, il s'agit évidemment des éléments. Si le nombre est positif, l'élément est un volume primitif simple dont on peut extraire les paramètres. Le premier élément de la matrice a pour adresse 45123 ce qui représente une sphère. En se servant de routines CATGEO, nous pouvons extraire le nom et le type de la primitive. Dans ce cas-ci, la primitive est de type SPHÈRE et son identificateur alphanumérique (son nom) est $SPHR2.

Les nombres négatifs qui se retrouvent dans les deux dernières colonnes de la matrice représentent une concaténation de primitives ayant déjà subies des opérations. Le nombre, sans tenir compte du signe négatif, nous réfère simplement à la ligne de la matrice où est décrite l'ensemble de primitives concaténées. Par conséquent, si nous considérons la troisième ligne de la matrice, les éléments de l'union (il y a le chiffre **1** dans la colonne des opérations) sont symbolisés par les nombres **–1** et **–2** qui nous envoient respectivement aux deux premières lignes de la matrice. C'est ainsi qu'on identifie les primitives simples de celles déjà concaténées. En traduisant les données de notre matrice par les termes que nous venons d'énumérer, nous obtenons la matrice de modélisation de la Figure 13 :

1.	union	$SPHR2	$REVO3
2.	union	$CYLN4	$CYLN6
3.	union	-1	-2
4.	soustraction	$CYLN1	-3
5.	évidement	-4	$SHEL5

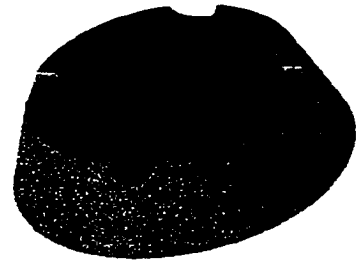


Figure 13 Matrice de modélisation d'un cendrier

Cette simple matrice de modélisation nous permet de constater de quelle manière a été construit la représentation d'un cendrier. Le concepteur a tout d'abord dessiné toutes les primitives dont il avait besoin : un cylindre plat, deux cylindres longs se croisant dans l'espace, une sphère et la révolution d'un profil. Il unit en premier lieu la sphère avec la révolution : c'est la révolution qui donnera l'aspect penché au contour du cendrier. En deuxième lieu, il effectue une autre opération d'union entre les deux cylindres (Figure 14). Ces dernières formeront les points de repos du cendrier pour les cigarettes.

Figure 14 Primitives unies

Par la suite, il unit les deux ensembles que nous venons de décrire. Ce n'est qu'en quatrième lieu qu'il soustrait du cylindre plat l'ensemble de primitives concaténées. L'opération est schématisée à la Figure 15.

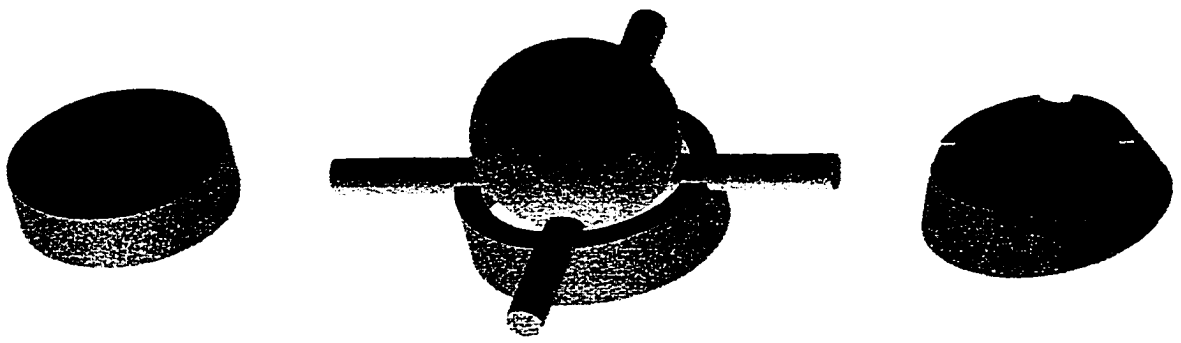


Figure 15 Soustraction booléenne

La représentation est déjà presque terminée car il ne reste plus qu'à inclure l'évidement à la surface inférieure du cendrier comme démontré à la Figure 16.

Figure 16 Création de la poche

On recueille donc dès lors énormément d'informations sur les primitives utilisées, leur type et l'historique de construction du modèle par la matrice de modélisation. Les dimensions des matrices des modèles font également parties des données extraites par les routines CATGEO.

3.2.3.2 Niveau de comparaison

Durant l'analyse des différents paramètres qui caractérisent les primitives du logiciel CATIA, nous avons remarqué qu'il peut exister plusieurs types de différences entre primitives. Nous les avons triées en cinq classes :

- *Classe A :* Les primitives sont tout à fait identiques, donc il n'y a pas de différence.
- *Classe B :* Les primitives sont tout à fait identiques, mais leurs positions dans l'arbre de modélisation ou leurs index sont différents.
- *Classe C :* Les types de primitive et leurs matrices de coordonnées dans l'espace sont identiques mais les paramètres des primitives qui les décrivent diffèrent.
- *Classe D :*Les types de primitive sont identiques mais leurs matrices de coordonnées dans l'espace diffèrent.
- *Classe X :* Les primitives sont totalement différentes ou isolées.

Les paramètres de comparaison de l'algorithme seront donc le type, la matrice de coordonnées de référence, les paramètres et l'index de position dans l'arbre des primitives d'un modèle. Nous verrons comment cette classification accélère le processus de comparaison de l'application. La matrice de coordonnées d'une primitive est du même format qu'une matrice de transformation. Cette matrice de coordonnées contient les informations nécessaires pour décrire le point de création et l'orientation, s'il y a lieu, de la primitive.

3.2.3.3 Index des primitives

Comme cité antérieurement, il est primordial de vérifier la structure des arbres de modélisation, surtout dans le cas où les primitives des deux modèles sont identiques. En effet, les opérations utilisées peuvent à elles seules produire des modèles complètement

différents. C'est pourquoi nous avons créé une nouvelle colonne pour la matrice de chaque arbre de modélisation, afin d'y ajouter un index de position pour chacune des primitives. En numérotant la position des primitives (niveau de la branche et position dans ce niveau), nous pouvons vérifier si les index correspondent bien entre deux modèles.

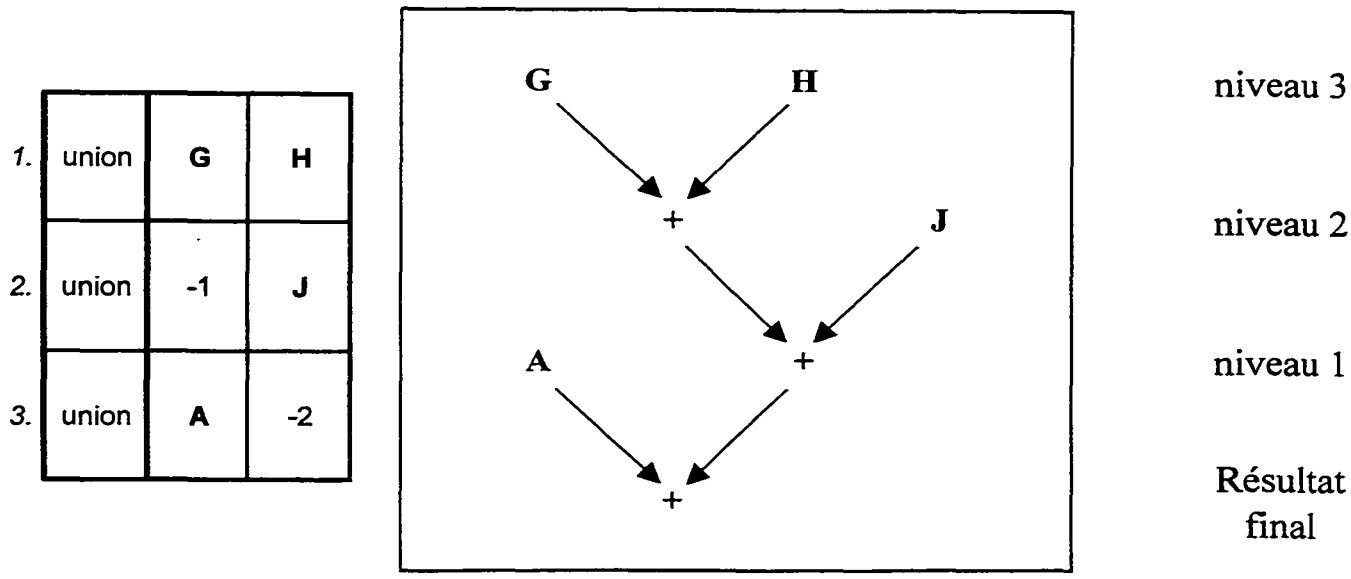


Figure 17 Primitives à indexer

Pour simplifier l'indexation des primitives, nous accumulons les primitives dans une pile, un niveau de l'arbre à la fois, en partant du bas de la matrice de modélisation. Un exemple des niveaux est illustré à la Figure 17. Cet exercice est effectué par une fonction que nous avons programmée et intitulée MARQUER. On effectue d'abord la lecture des éléments de la dernière ligne de la matrice, qui ne peuvent être qu'au nombre de deux. D'après l'exemple que nous avons ci-haut, les éléments lus de la matrice seront l'entité **A** et le renvoi **–2** (voir les Figures 17 et 18). Les primitives sont placés dans une pile de primitives tandis que les renvois sont placés dans une pile de renvois. On distingue une primitive d'un renvoi d'après le signe positif ou négatif de l'élément lu.

Figure 18 Piles de primitives et de renvois

La fonction MARQUER se déroule en deux principales étapes. Dans un premier temps, elle dépile la pile de primitives en y associant un index. La formule pour déterminer l'index de la primitive est la suivante :

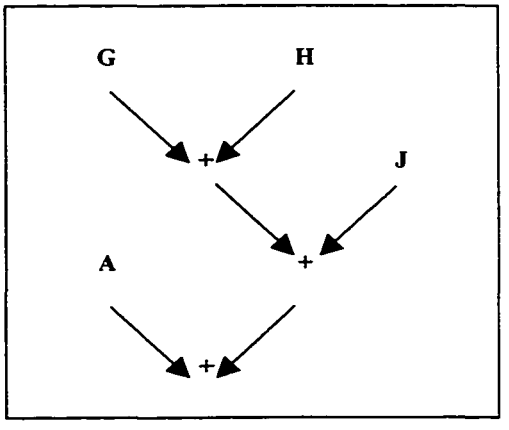
Index = Niveau * 1000 + position dans la pile

Ainsi, l'index de l'entité A sera 1 * 1000 + 1, donc 1001. Cette méthode simplifie l'indexation des primitives. Une fois toutes les primitives indexées, la fonction passe à la deuxième étape. Elle itère premièrement le niveau, qui sera maintenant le deuxième. Ensuite, elle dépile la pile des renvois et se sert de ses éléments pour connaître sur quelles lignes se trouvent les prochains éléments à empiler. Puisque c'est l'entité –2 qui se trouve dans notre pile de renvois, la fonction MARQUER se rend à la ligne 2 de la matrice pour empiler le renvoi –1 et l'entité J dans leur pile correspondante. Les primitives sont de nouveau indexées et l'on passe au niveau suivant, jusqu'à ce que toutes les primitives soient indexées. Les résultats d'indexation sont à la Figure 19 :

Entité	Index
A	1001
J	2001
G	3001
H	3002

Figure 19 Indexation de l'exemple

Le facteur de mille (1000) nous permet de distinguer dans le même nombre le niveau et la position de la primitive, tant que le nombre de primitives ne dépasse pas 999 dans l'arbre de modélisation. On peut alors aisément extraire ces deux données de l'index pour analyser la position des primitives dans l'arbre de modélisation. Bien sûr, cette méthode simpliste d'indexation a ses lacunes. On néglige tout d'abord les notions de sous-arbre : pour une paire de primitives ou d'éléments concaténés, on ne connaît pas l'élément enfant qui y est associé. On se doute seulement que cet élément parent se trouve au niveau inférieur de l'arbre. De plus, la Figure 20 démontre bien que des translations de sous-arbre sur leur niveau peuvent donner la même indexation.



Entité	Index
A	1001
J	2001
G	3001
H	3002

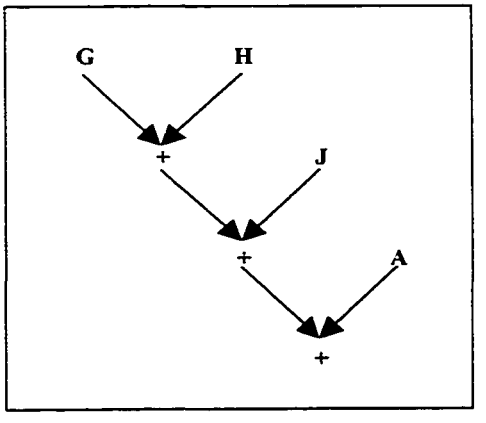


Figure 20 Deux arbres de modélisation, une même indexation

Pour combler cette dernière lacune, une comparaison des opérations, de l'ordre d'utilisation de ces opérations et des propriétés inertielles des deux modèles a été prévue pour éliminer les possibilités de ne pas trouver la différence lorsqu'elle est présente. Comme l'indexation des primitives part de la gauche sur chaque niveau, la différence de structure des arbres n'apparaît pas dans les index.

3.2.3.4 Types et coordonnées des primitives

En ce qui concerne la comparaison des types de primitive et des index, on ne fait que comparer des valeurs numériques entre elles. Comme mentionné auparavant, chaque type de primitive est associé à un nombre dans la base de données CATIA. Dans la Figure 21 sont illustrés les types de primitives qui sont considérés par le comparateur.

Type de primitive	
Élément isolé	**0**
Plan de coupe	**1**
Cuboïde	**9**
Sphère	**16**
Cylindre	**17**
Balayage	**23**
Prisme	**24**
Fillet	**26**
Tuyau	**32**
Volume complexe	**48**
Épaisseur de mur	**51**

Figure 21 Les types de primitive

Pour chaque primitive, la routine GIRSCF de CATIA nous permet d'extraire le type et la matrice de coordonnées de création de l'élément. Comme ces données sont numériques, une simple comparaison d'égalité nous permettra d'identifier les types de primitives qui ont changés d'un modèle à un autre. Pour ce qui est de la matrice de coordonnées, elle est composée de douze valeurs numériques. Chacune d'entre elles devra être comparée avec la valeur correspondante de l'autre modèle. En bref, un test élémentaire d'égalité suffit pour repérer des différences au niveau de la matrice de coordonnées des primitives.

3.2.3.5 Paramètres numériques

La majorité des modifications qui se produisent d'une révision à une autre sur les modèles CAO touche les paramètres des primitives. Le premier des deux types de paramètres de primitives est numérique. Pour bien saisir le procédé d'extraction, nous prendrons le cuboïde de la Figure 22 comme exemple.

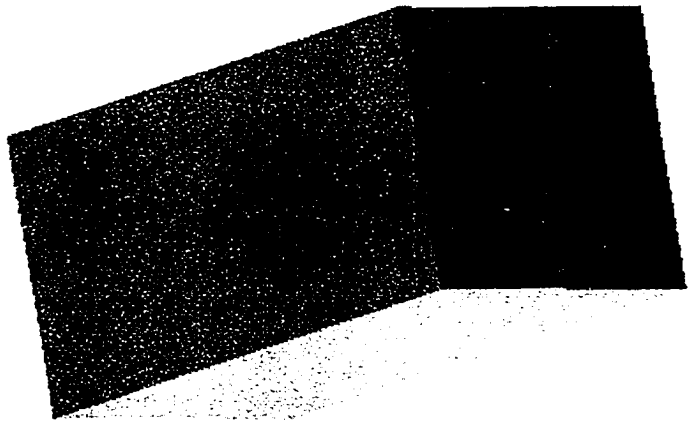


Figure 22 Image d'un cuboïde

On extrait les paramètres de définition du cuboïde à partir de la routine GIRSPL. Tel qu'illustré à la Figure 23, la routine d'extraction nécessite l'adresse de la primitive à analyser, ainsi que le solide qui le contient.

GIRSPL (MNUM, JSOL, IPRIM, PT, V1, V2, V3, XLNG, THK, IER, *LAB)

Function
Read the data of the primitive of a PARALLELEPIPED type solid.

Input

JSOL	I*4	Solid.
IPRIM	I*4	Address of the primitive.

Output

PT	R*8	(3) Coordinates of an external vertex of the parallelepiped.
V1	R*8	(3) Vector associated with the first direction.
V2	R*8	(3) Vector associated with the second direction.
V3	R*8	(3) Vector associated with the third direction.
XLNG	R*8	(3) Length of the parallelepiped along the three directions.
THK	R*8	Parallelepiped thickness. =0 for a solid parallelepiped.

Figure 23 Routine d'extraction du cuboïde

En retour, nous obtenons la coordonnée du sommet de référence, les trois vecteurs qui décrivent les directions du cuboïde, les longueurs du cuboïde selon ces directions et l'épaisseur du cuboïde, dans le cas où il est évidé. On remarque que chacune de ces données sont des valeurs réelles. Ainsi, pour comparer deux cuboïdes, il s'agira de vérifier si chacune des variables contenant les paramètres numériques de la primitive d'un modèle est bien identique à sa correspondante dans l'autre modèle.

3.2.3.6 Paramètres géométriques

Le deuxième type de paramètre est géométrique. Pour un cuboïde et la plupart des primitives simples, seules des valeurs numériques, entières ou réelles, suffisent pour les définir. La sphère, le cylindre et le filet entrent dans cette catégorie. Mais pour ce qui est d'un balayage, c'est tout à fait différent. Comme l'illustre la Figure 24, une primitive de type BALAYAGE contient, en plus de paramètres numériques, des paramètres géométriques qui requerront une analyse plus poussée.

GIRSWE (MNUM, JPRIM, JPROF, JSPIN, JCCRV, IPLAN, ICLOS, JLIM1, JLIM2, JREF, JAXIS, JANGLE, JSCALE, IER, *LAB)

Function
Read the parameters of a SWEEP primitive.

Input

JPRIM	I*4	address of the primitive.

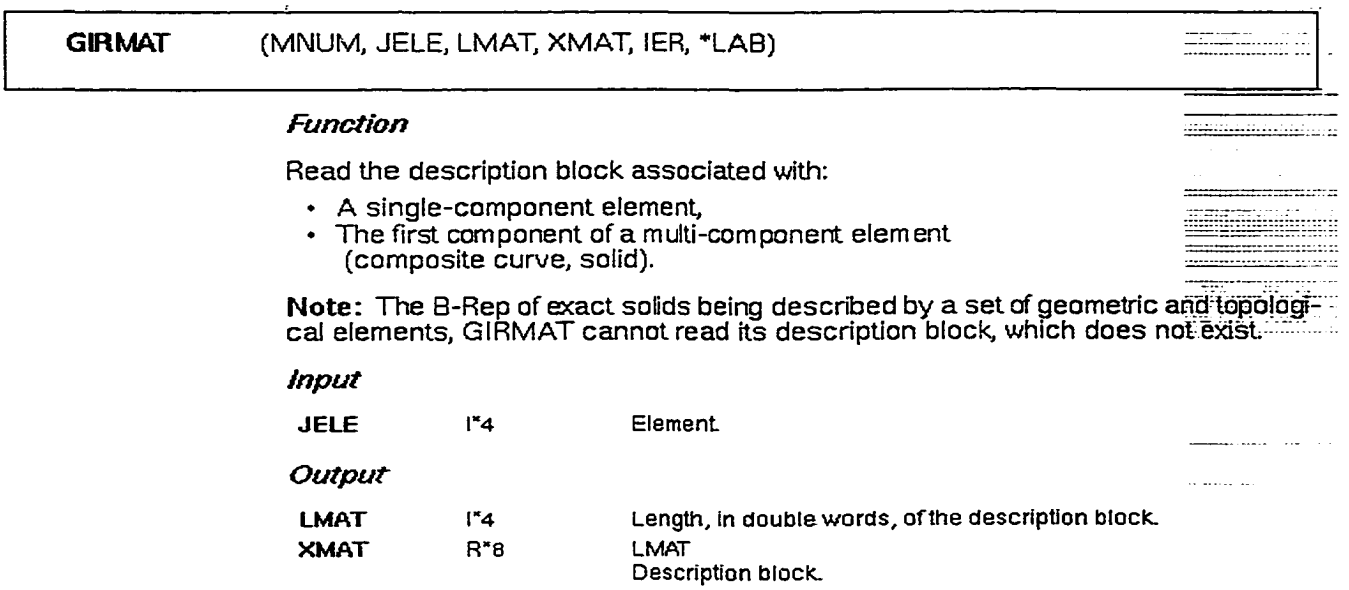
Output

JPROF	I*4	Stack of profile elements.
JSPIN	I*4	Stack of spine elements.
JCCRV	I*4	Dummy
IPLAN	I*4	(2) = 0 non planar = 1 planar For spine and center curve
ICLOS	I*4	= 0 the spine is open = 1 the spine is closed
JLIM1	I*4	Dummy
JLIM2	I*4	Dummy
JREF	I*4	Reference element (plane or surface) to define the axis along the spine
JAXIS	I*4	Axis used to position the profile on the spine
JANGLE	I*4	Dummy
JSCALE	I*4	Dummy

Figure 24 Routine d'extraction du balayage

La variable entière JPROF désigne l'adresse d'une pile d'éléments géométriques qui composent le profil à être balayé. Il faudra alors dépiler un à un chacun des éléments de cette pile pour vérifier s'il n'a pas été modifié entre les deux modèles. Il en va de même pour les éléments géométriques de la directrice du balayage, contenus dans une autre pile dont la variable de l'adresse est JSPIN.

Conséquemment, dans le cas des paramètres géométriques, nous avons besoin d'une sous-routine d'extraction pour chaque type d'éléments géométriques présent. Les paramètres de définition de ces éléments sont stockés eux-mêmes dans une pile structurée selon le type. Nous pouvons extraire et analyser les paramètres de ces éléments géométriques avec la routine GIRMAT dont on a la description dans la Figure 25.



GIRMAT (MNUM, JELE, LMAT, XMAT, IER, *LAB)

Function

Read the description block associated with:

- A single-component element,
- The first component of a multi-component element (composite curve, solid).

Note: The B-Rep of exact solids being described by a set of geometric and topological elements, GIRMAT cannot read its description block, which does not exist.

Input

JELE	I*4	Element.

Output

LMAT	I*4	Length, in double words, of the description block.
XMAT	R*8	LMAT Description block.

Figure 25 Routine d'extraction d'un élément géométrique

Tout comme pour les paramètres numériques, les index et les matrices de coordonnées de référence des primitives, une vérification d'égalité numérique validera la similitude des paramètres géométriques des primitives contenues dans les deux modèles.

3.2.4 Procédures de comparaison

Étant donné que nous avons quatre paramètres de comparaison, soit le type, la matrice de coordonnées, les paramètres et l'index de la primitive, l'analyse des deux modèles à comparer s'effectue en quatre rondes. Les primitives entre les deux modèles peuvent être identifiées différemment, c'est pourquoi notre routine compare une primitive du premier modèle avec toutes les primitives du deuxième modèle. On fait cette comparaison à travers tous les paramètres de comparaison : type de primitive, matrice de coordonnées, paramètres de la primitive et index de position dans l'arbre. Pour chaque type de différence est associé une certaine couleur (Figure 26).

Type de différences des primitives	
Aucun	
Index	Vert
Paramètres	Jaune
Coordonnées	
Type	

Figure 26 Légende colorée des différences

Pour commencer, nous formons deux piles dans lesquelles nous enfilons toutes les primitives des matrices de modélisation, une pile par modèle. Nous catégorisons ces deux piles de classe X (Figure 27) pour des raisons que nous verrons ultérieurement

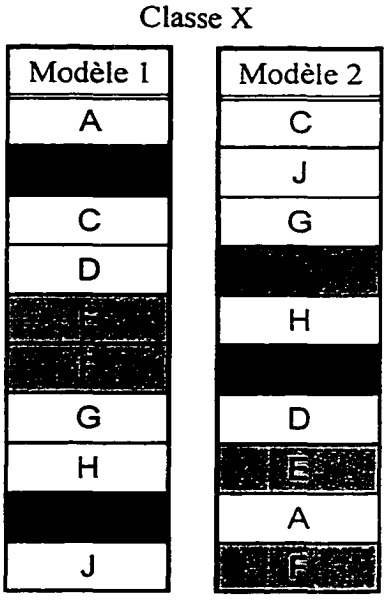


Figure 27 Piles initiales de primitives de classe X

En un premier tour, on peut repérer quelles sont les primitives qui sont parfaitement identiques. Il faudra utiliser toutes les routines et fonctions dont nous avons discuté plus tôt lors de la description de l'extraction des données. Nous allons déplacer ces primitives tout à fait identiques dans les piles de classe A de la Figure 28, toujours une pile pour chaque modèle.

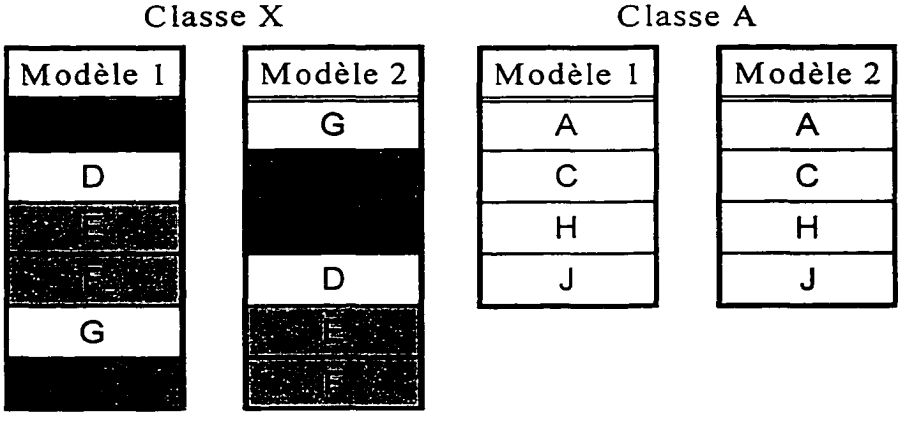


Figure 28 Création des piles de classe A

Par la suite, nous ferons un deuxième tour à partir des piles de classe X, avec les seulement les trois premiers critères de comparaison, le type, la matrice de coordonnées et les paramètres des primitives. Comme toutes les primitives de classe A ont été déplacées de la pile des primitives de classe X, cette routine de comparaison s'effectuera un peu plus rapidement. Les résultats positifs de ce tour de ronde seront déplacés dans des piles de classe B (Figure 29).

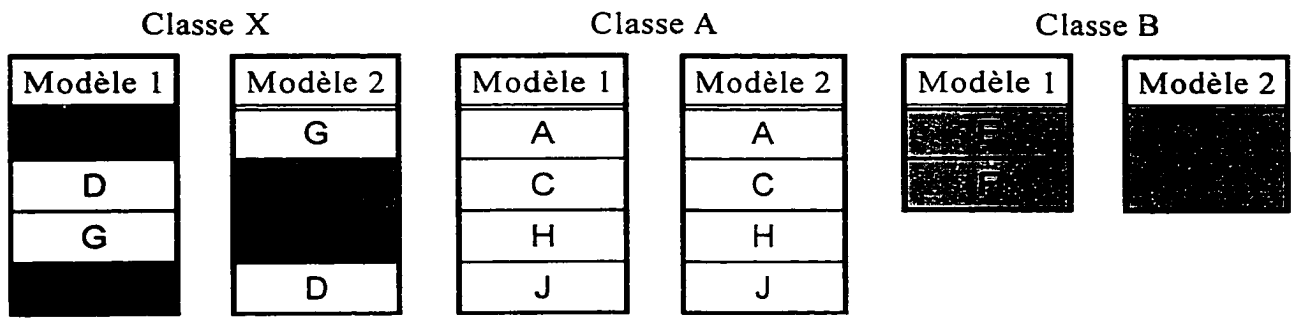


Figure 29 Création des piles de classe B

La troisième ronde de tri s'effectue aussi à partir des piles de classe X, mais seulement les primitives qui ont en commun leur type et leur matrice de coordonnées seront déplacées vers des piles de classes C. Un des avantages de procéder ainsi est d'éviter que deux primitives tout à fait semblables, donc de classe A, ne se retrouvent dans les piles de classe B,C ou D. Il est évident que si les deux primitives ont leurs quatre paramètres de comparaison identiques (classe A), ils ont les trois premiers identiques (classe B), les deux premiers identiques (classe C), etc. ce qui nous importe durant le tri, c'est le nombre de paramètres de comparaison identiques, selon la séquence de priorité établie.

On répète encore la procédure tout en omettant un critère de similarité à chaque tour. Ainsi, chaque ronde se fait avec de moins en moins d'éléments, et à la fin, on obtient dix piles distinctes : deux piles de chaque classe, c'est à dire de A à D, et deux autres piles qui sont en fait les piles principales de classe X de chaque modèle dans lesquelles on retrouve toutes les primitives qui n'ont aucune correspondance entre les modèles. Ainsi, l'apparition ou la disparition de primitives d'un modèle à un autre est détectée par des primitives se trouvant dans les piles de classe X (exemple à la Figure 30).

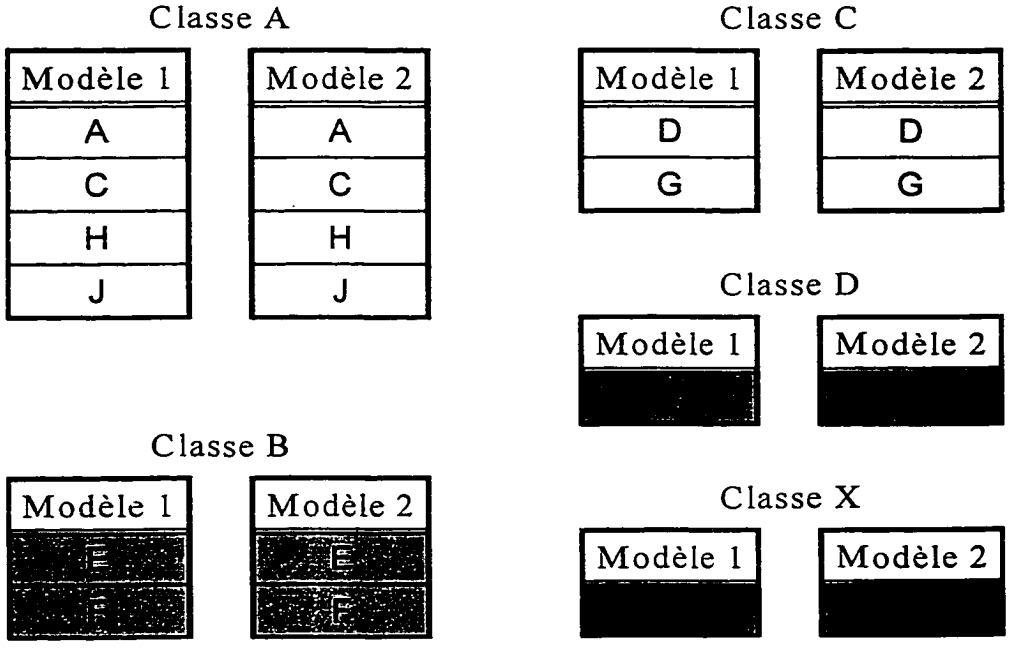


Figure 30 Création des piles des cinq classes

Puisque nous voulons comparer des révisions de modèle, on suppose logiquement que ces révisions ne diffèrent pas énormément entre elles. On s'attend par conséquent à détecter des différences mineures : changement de paramètres, disparition, apparition ou déplacement de primitives dans l'arbre, etc. Il est rare de retrouver des cas dans lesquels toute la structure de l'arbre de modélisation, ainsi que les primitives qui la composent

soient identiques entre deux révisions, et que seules les opérations de modélisation aient été modifiées. Normalement, c'est une modification des primitives qui induit une modification des opérations, raison pour laquelle on accorde pour l'instant peu d'importance à la comparaison des opérations.

Ce n'est qu'à la fin de la comparaison des primitives, une fois qu'elles ont toutes été classées et seulement si elles sont toutes identiques, que l'on vérifiera si les opérations des arbres sont les mêmes et si elles ont été utilisées dans le même ordre.

Si une différence entre des primitives a été détectée avant la comparaison des opérations, cette dernière comparaison sera négligée; la raison en est simple. Comme mentionné plus tôt, la première colonne de la matrice des arbres contient les opérations tandis que les deux autres contiennent les éléments qui subissent les opérations, soit des primitives, soit des sous-ensembles. Pour comparer si les opérations entre deux modèles sont identiques, il faudrait analyser l'opération même et ses éléments. Le problème est que la majorité des opérations ont comme éléments des sous-arbres dont on a pas de routine d'analyse et d'extraction de données. Tout ce qu'on peut obtenir est l'identification de chaque pointeur d'élément, information inutile dans notre cas. La comparaison des opérations ne sert alors qu'à vérifier si deux modèles sont tout à fait identiques.

À la Figure 31, nous présentons un organigramme simplifié du procédé de comparaison pour le tri des primitives dans les piles appropriées.

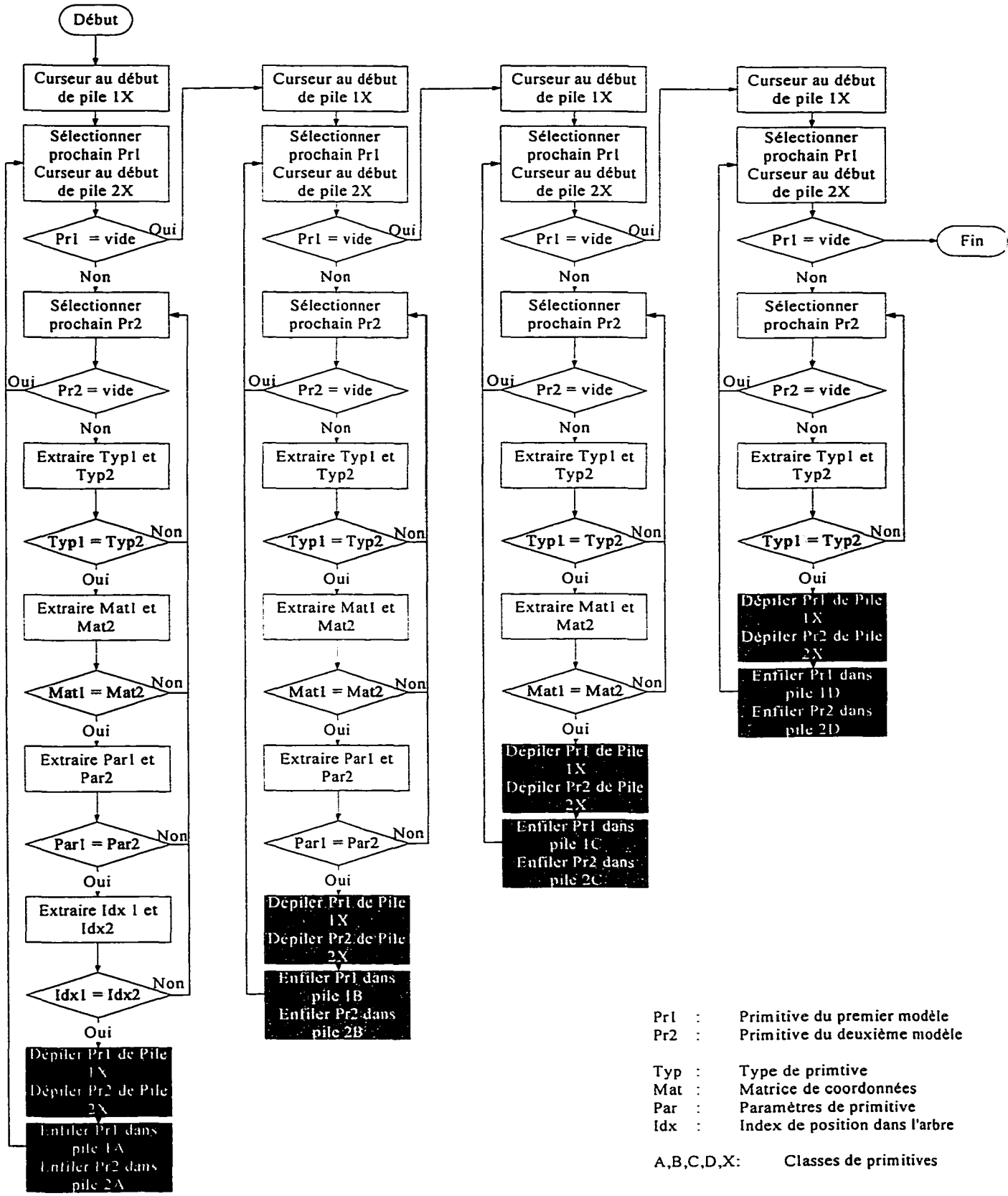


Figure 31 Organigramme de comparaison

3.2.5 Description des différences

La prochaine étape de l'application consiste en l'affichage des différences entre les deux modèles. Le rapport de comparaison sera rédigé dans un panneau CATIA déjà accessible par un fichier spécialement préparé pour cela. L'analyse des modèles par l'application se fait en deux procédures de comparaison. La première procédure de comparaison que nous venons de décrire ne sert qu'à classer les primitives selon leurs différences. C'est durant la deuxième procédure de comparaison des primitives que le rapport de comparaison est rédigé.

Comme les primitives sont déjà classées, la deuxième comparaison s'effectue directement avec le paramètre de comparaison associé à la classe. Pour les primitives de classe B, l'application rédige une notification de changement d'index pour chaque primitive. Pour les primitives de classes C, on trouve dans le rapport de comparaison quels sont les paramètres numériques et géométriques qui sont différents. Finalement, pour les primitives de classe D, on mentionne que la matrice de coordonnées a changé entre les deux modèles. Un petit exemple du rapport est illustré à la Figure 32.

Figure 32 Exemple de différences rédigées

Seules les primitives de classe A (primitives totalement identiques) et de classe X (primitives isolées) ne font pas partie de ce dernier tour de comparaison. Pour ce qui est des primitives isolées de classe X, on ne fait qu'identifier leur présence dans les deux modèles. L'affichage du rapport apparaît aussitôt les deux processus de comparaison terminés.

3.2.6 Affichage des différences

L'identification graphique des différences ou modifications est réalisée lors de la première procédure de comparaison étant donné que les primitives sont classées. Directement dans la session de travail de l'usager, on avait le choix entre surintensifier les primitives modifiées (ou isolées) et les colorer par l'application. La deuxième option est celle que nous avons privilégiée. Tout d'abord, nous n'avons pas pu surintensifier les primitives même, mais uniquement leurs surfaces fonctionnelles. Par contre, les avantages de la coloration sont bien plus intéressants. Premièrement, ce sont les primitives précisément qui sont colorées. Deuxièmement, cette coloration s'effectue également dans l'arbre de modélisation de la composante, ce qui facilite le repérage de la primitive modifiée par l'ingénieur. Finalement, la coloration des primitives est différente selon sa classe. Un exemple de coloration par classe du cendrier est illustré à la Figure 33. Cette dernière particularité oriente rapidement l'ingénieur sur la nature des modifications introduites entre deux révisions d'un modèle.

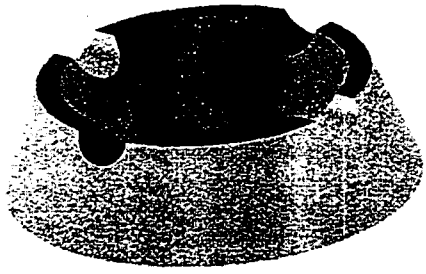


Figure 33 Image du cendrier coloré

Il s'est avéré judicieux de colorer le deuxième modèle seulement afin de comprendre les modifications présentes. Ce sera donc l'usager qui devra convenablement choisir les modèles dans le sens chronologique de leur création ou de leur utilisation. Seules les primitives isolées du premier modèle sont colorées. Cela indiquera visiblement si des éléments ont été effacés lors de la création de la deuxième révision de la composante. Par contre, la coloration du premier modèle n'est pas permanente. Dès que l'application est arrêtée, la coloration du premier modèle disparaît. Ceci constitue un problème qui n'a pas été résolu lors de la réalisation du projet. La coloration adoptée pour le comparateur est cyan pour les primitives identiques, mauve pour les modifications des matrices de coordonnées, jaune pour les changements de paramètres de primitives, vert pour les changements d'index de position dans l'arbre CSG et rouge pour les primitives isolées.

3.3 Procédure d'utilisation

L'application de comparaison que nous avons créée a été conçue, en premier lieu, pour faciliter et accélérer les tâches des ingénieurs. Nous avons donc simplifié l'utilisation de notre application à quelques étapes très concises. Nous allons décrire en détails chacune de ces étapes afin d'obtenir les résultats pertinents que nous désirons.

Tout d'abord, il faut bien entendu connaître les deux modèles du produit à analyser. A l'aide de la commande OPEN, on ouvrira les deux modèles en s'assurant que la révision la plus récente du produit soit le **modèle actif** de la session de travail.

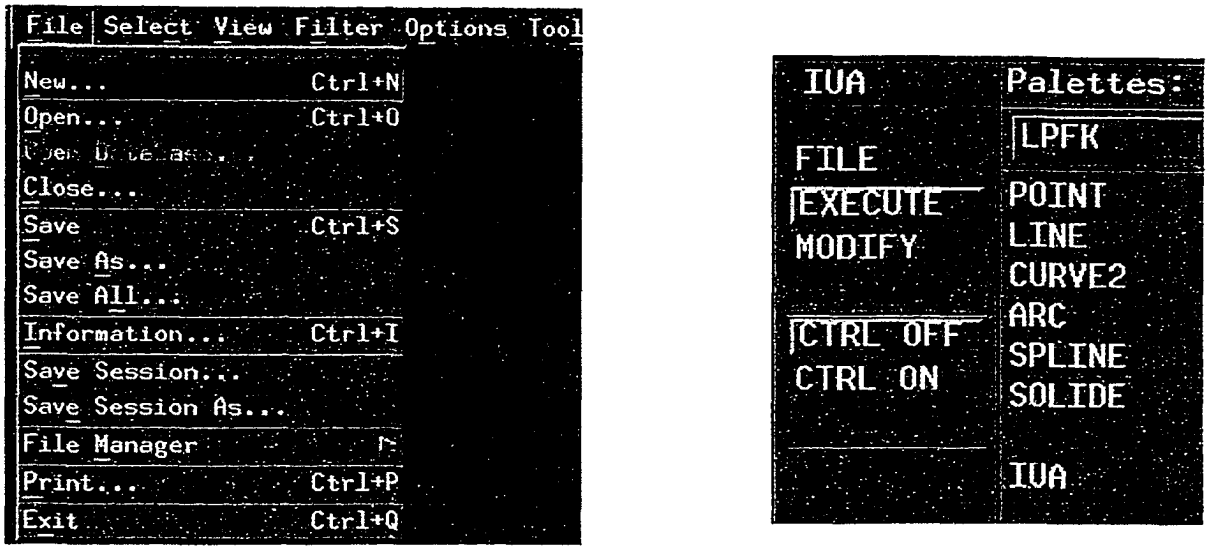


Figure 34 Démarrage de l'application

Une fois que les deux modèles se trouvent dans l'espace de travail CATIA, on doit utiliser la fonction IUA pour avoir accès à toutes les applications personnalisées disponibles à l'utilisateur (Figure 34 et 35). Dans ce cas-ci, le nom de notre application s'appelle *COMPARER* avec le commentaire suivant : *COMPARER DEUX SOLIDES.* C'est l'application à sélectionner.

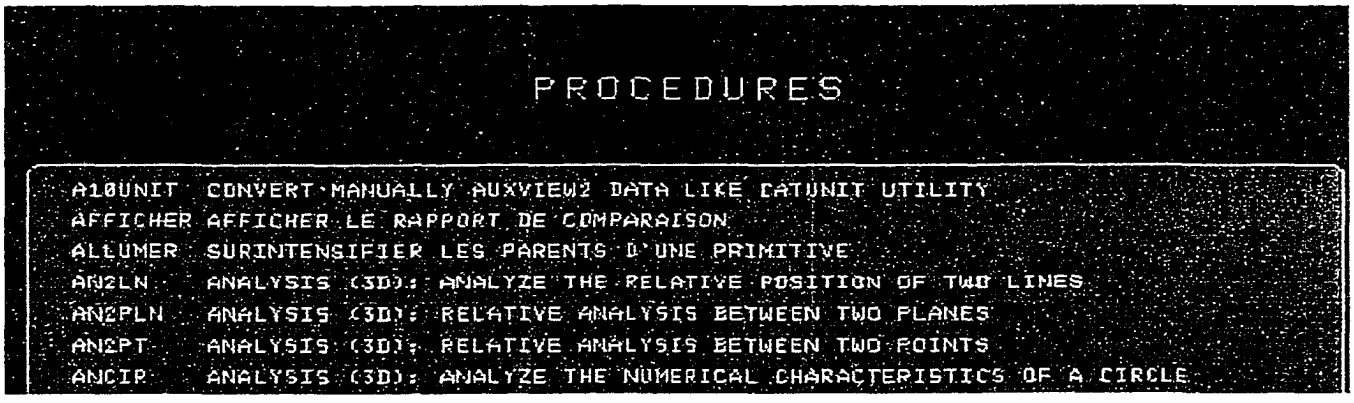


Figure 35 Sélection de l'application

Pour accélérer la recherche de l'application *COMPARER*, on peut taper les premiers caractères du nom de l'application dans la zone d'acquisition alphanumérique au bas de l'écran comme illustré à la Figure 36.

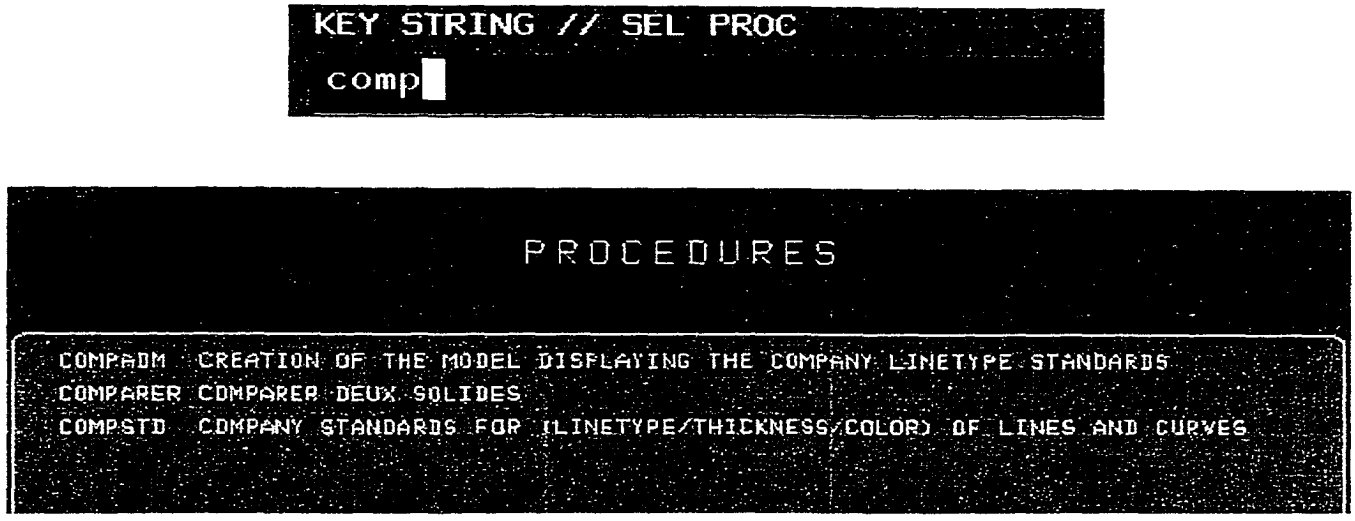


Figure 36 Astuce de sélection rapide de l'application

Après le démarrage de l'application, cette dernière demande aussitôt que l'on sélectionne le premier modèle ou solide de l'analyse, puis le second. Il est très important de choisir comme premier modèle celui qui représente la révision antécédente. C'est donc le modèle passif que l'on doit sélectionner. Dans le cas où les deux modèles sont très semblables et superposés un sur l'autre, on doit se servir de la touche F2 du clavier pour effectuer le bon choix. Au coin inférieur gauche de l'écran, le nom et l'état du solide devraient apparaître. Le terme *OVL* que l'on voit à la Figure 37 devrait précéder le nom du solide : cela signifie que l'on a sélectionné le modèle passif, donc le bon choix pour le premier solide.

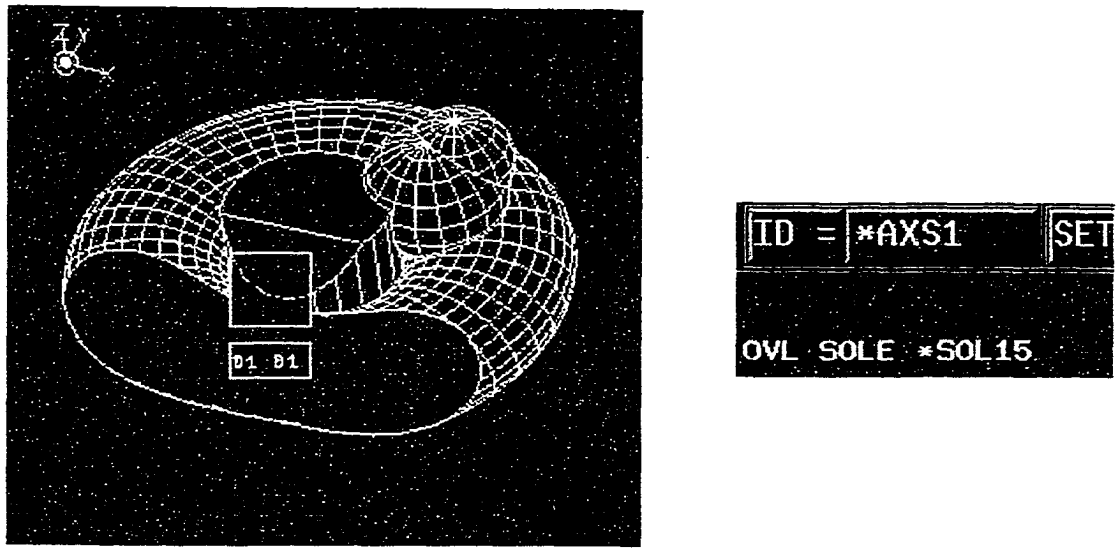


Figure 37 Aide de sélection stratégique des modèles

Le solide sélectionné est aussitôt surintensifié. Si la sélection n'est pas bonne, on peut toujours la refuser en cliquant sur le bouton NO à la question posée de la Figure 38. On refait à nouveau la sélection.

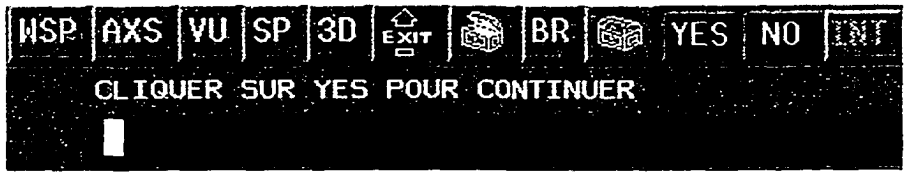


Figure 38 Confirmation de sélection de modèles

De la même manière, c'est à dire avec la touche F2 si nécessaire, on sélectionne le deuxième solide, donc le modèle actif. Dans la même zone, au bas à gauche de l'écran, le terme *DET* de la Figure 39 devrait précéder le nom du solide afin de confirmer qu'il s'agit bien du modèle actif. On peut recommencer la sélection si elle a été mal faite. Le

deuxième solide sera également surintensifié, après que la surintensification du premier solide ait été désactivée. L'application débutera ses démarches de comparaison lorsque la sélection du deuxième solide sera validée.

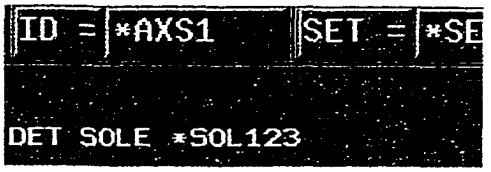


Figure 39 Identification du modèle sélectionné

Selon la taille et la complexité des modèles, l'application prendra plus ou moins de temps pour effectuer son analyse. On peut quelque peu visualiser les résultats temporaires que l'application obtient d'après les messages qui apparaissent rapidement au coin inférieur gauche de l'écran.

Lorsque la comparaison est terminée, une nouvelle fenêtre semblable à la Figure 40 apparaîtra à l'écran, faisant disparaître momentanément les modèles. Cette fenêtre contient un rapport de comparaison dans lequel sont énumérées toutes les différences que l'application a détectées, ainsi que leur nature. À ce stade-ci du processus de comparaison des modèles, il peut être utile de noter les primitives identifiées, surtout si elles sont nombreuses.

Les données exactes qui diffèrent entre les deux modèles peuvent également être affichées dans le rapport de comparaison. Cette alternative a néanmoins été mise de côté car certaines petites modifications, en particulier celles qui touchent les primitives formées d'éléments primaires comme les surfaces peuvent induire des dizaines de paramètres différents dont on ne comprendrait pas le sens sans l'explication détaillée de chacun d'eux.

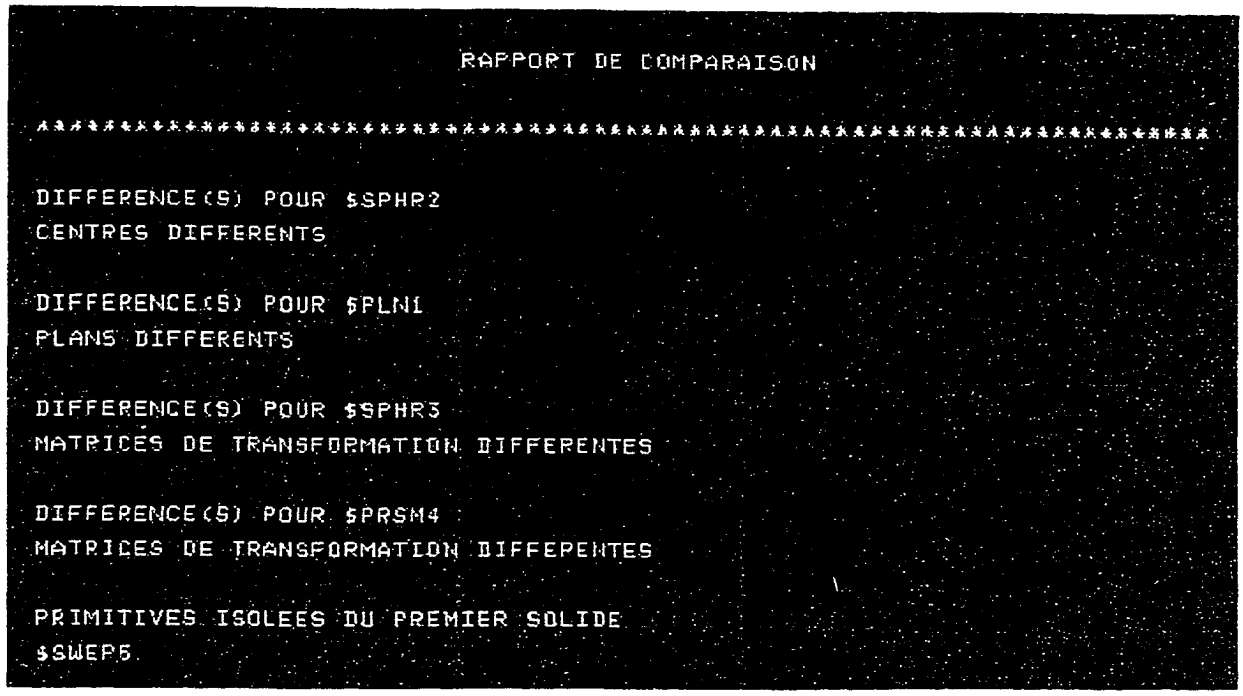


Figure 40 Exemple de rapport de comparaison

Il suffit ensuite de cliquer sur la touche YES pour voir le reste du rapport de comparaison et revenir aux modèles, dont les différences sont maintenant identifiées par coloration. Comme nous l'avons proposé précédemment, toutes les modifications sont identifiées sur le deuxième modèle, c'est à dire celui qui est actif. Seules des primitives isolées peuvent être colorées sur le premier modèle. En effet, si de la révision précédente à la révision suivante du produit, nous avons effacé des primitives, on ne peut identifier cette différence que sur le modèle dans lequel les primitives existent toujours, donc la révision précédente du produit, soit le modèle passif.

Il est important de remarquer que la coloration des modèles n'est permanente que pour le modèle actif. En effet, si l'on quitte l'application ou que l'on change de modèle actif, la coloration du modèle passif sera perdue. C'est la raison pour laquelle il est primordial

que le modèle actif, lors du démarrage de l'application, soit le deuxième modèle, soit la révision la plus récente. C'est sur cette révision que l'on veut logiquement observer l'évolution du produit. Alors, une mauvaise sélection des modèles induira des erreurs quant à la représentation de l'évolution temporelle du produit.

Une fois l'exécution terminée, il est possible de passer à la fonction SOLIDE du logiciel pour mieux saisir les primitives qui ont été modifiées puisque la coloration est permanente même dans l'arbre de modélisation du deuxième modèle, bien sûr. La Figure 41 illustre un arbre de modélisation typique du modeleur CATIA sous sa révision 4.

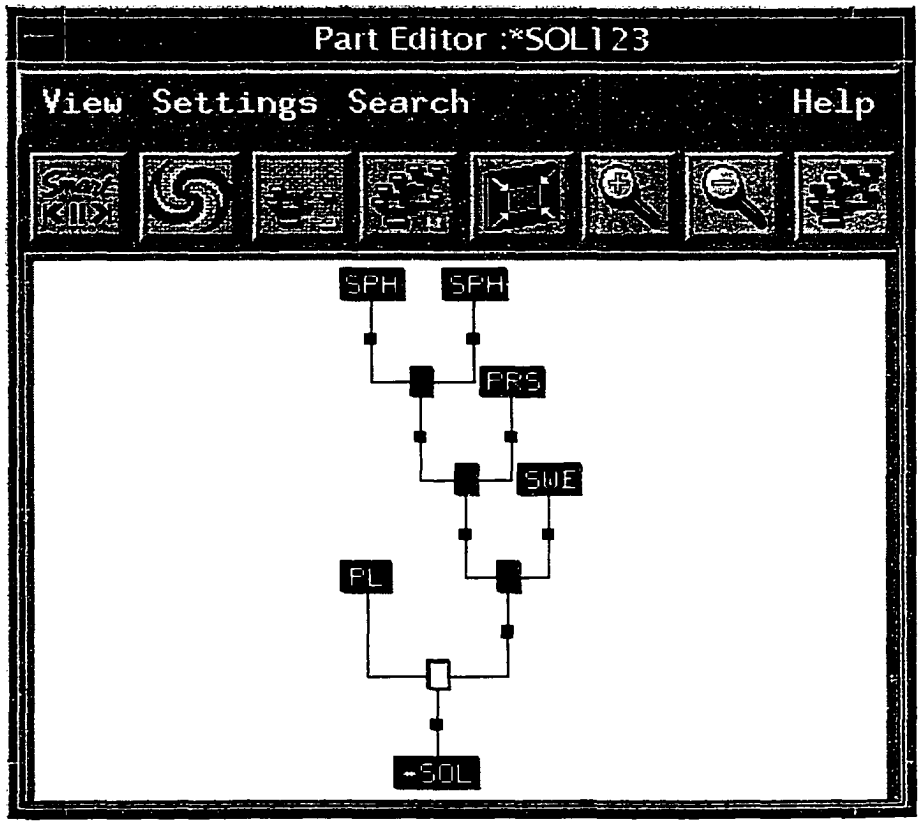


Figure 41 Exemple d'arbre CSG

CHAPITRE 4

LES EXEMPLES DE VALIDATION

L'outil de comparaison que nous avons conçu s'est révélé fort efficace durant les premiers essais que nous avons effectués. De simples géométries ont été créées puis modifiées pour vérifier le comportement et l'efficacité de l'application. Par contre, pour vérifier cette dernière dans le milieu industriel, nous avons utilisé des modèles CAO de pièces mécaniques utilisées au sein de l'entreprise Bombardier Aéronautique. En ayant des exemples concrets d'évolution de modèles, nous avons pu remarquer les limites de notre application, ainsi que les opportunités à saisir dans une suite éventuelle de nos travaux.

Nous avons inclus dans ce rapport neuf modèles dont nous avons tenté d'extraire les différences entre les révisions. Les deux premiers modèles simples permettent de comprendre et de visualiser rapidement certains types de modifications repérées par le comparateur. Les deux modèles suivants sont aéronautiques et permettent de confronter le comparateur avec des modifications un peu plus complexes. Finalement, les quatre derniers modèles sont aussi aéronautiques. Ceux-ci nous permettront de constater les performances du comparateur de modèles.

4.1 Description des modifications

Avant d'aborder les essais effectués à l'aide du comparateur, nous allons illustrer les diverses modifications qui sont typiques de l'industrie. Cette démonstration nous évitera d'illustrer toutes les modifications effectuées pour chacun des modèles utilisés.

La présentation des modifications possibles sera a été faite à l'aide du modèle de Bombardier Aéronautique que nous appellerons BOMB3, illustré à la Figure 42. Ce modèle comportait une quarantaine de primitives.

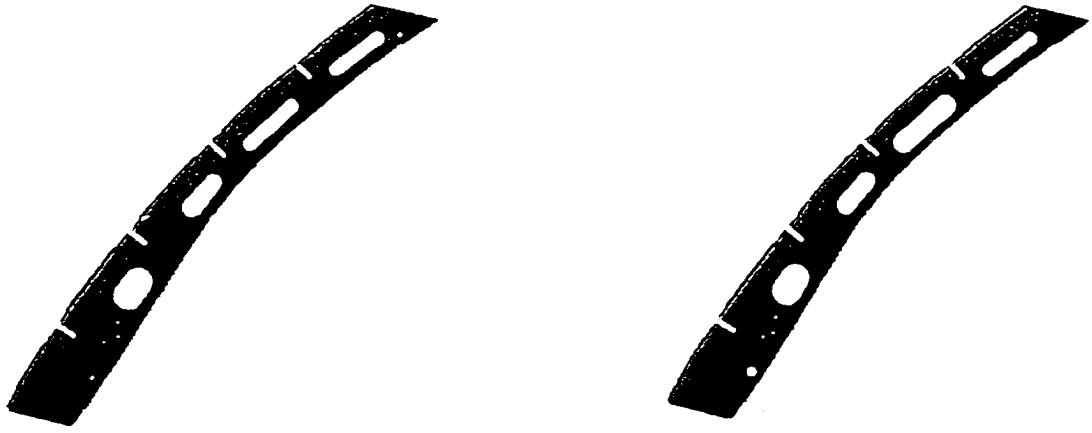


Figure 42 Révisions du modèle CAO BOMB3 à comparer

Un total de sept modifications ont été introduites au modèle BOMB3 pour en créer sa révision B. Chacune des modifications préparée sollicite au moins un des paramètres de comparaison que nous avons utilisé dans notre algorithme de comparaison des primitives. Nous allons décrire chacune de ces modifications.

4.1.1 Modification d'un congé

La première modification introduite au modèle est le changement de rayon de filet de la Figure 43. Il s'agit d'un type de modification qui survient fréquemment dans l'évolution des révisions d'un modèle. Nous avons fait passé la dimension du rayon de 0.12po à 0.2po. C'est un changement qui se décèle difficilement l'œil nu par un ingénieur travaillant sur de tels modèles CAO. Cela représente donc le changement simple d'un paramètre de définition d'une primitive de type FILET.

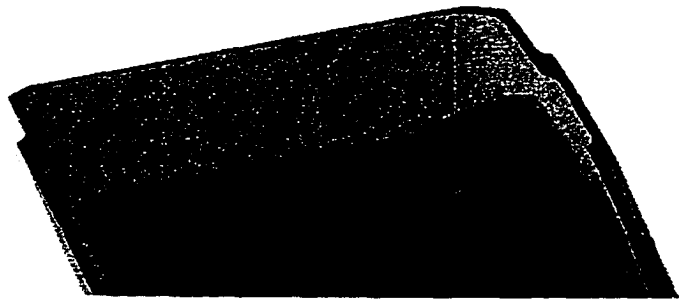


Figure 43 Modification d'un filet

4.1.2 Modification d'un mur

La deuxième modification introduite est le changement d'épaisseur d'un mur de la pièce BOMB3. Ici, on s'attaque à un autre type de primitive, un OFFSET. Cette modification est également appliquée à un paramètre de définition. L'épaisseur d'un mur est contrôlée par la valeur de deux décalages mesurés par rapport à une surface de référence. Dans notre cas, nous allons changer la valeur du deuxième décalage de 0.0077po à 0.01po. Cette modification est presque invisible à l'œil nu. C'est dans ce genre de situation qu'il devient intéressant d'utiliser un comparateur automatique de modèles. La modification est illustrée à la Figure 44.

Figure 44 Modification d'un mur (offset)

4.1.3 Modifications de trous

Toujours avec un type de primitive simple, c'est à dire de type CYLINDER, nous allons effectuer quelques modifications particulières (Figure 45). Tout d'abord, nous allons créer un trou. Il s'agit ici d'observer si l'application peut déceler l'apparition d'un élément, plutôt qu'une modification.

Ensuite, nous allons réaliser deux changements sur un même élément. En premier lieu, nous allons modifier un paramètre de définition d'un trou toujours présent dans le modèle. Nous ferons passer le rayon de cet élément de 0.125po à 0.128po. De plus, nous allons modifier la position de ce trou dans le modèle en y appliquant une translation de 0.001po selon l'axe des abscisses global du modèle. Non seulement l'application devra déceler une différence imperceptible par l'usager mais elle devra également faire part des deux types de modification que nous avons introduites dans le même élément.

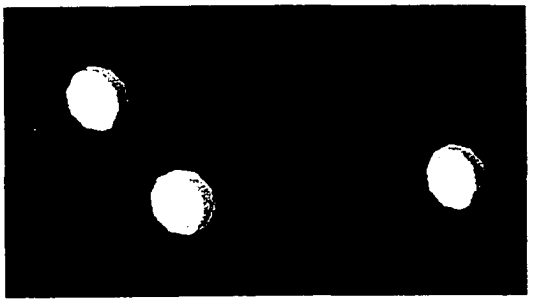


Figure 45 Modification d'un trou

4.1.4 Modification de prismes

Finalement, les dernières modifications que nous introduirons au modèle BOMB3 seront réalisées sur des primitives de type PRISM. Plutôt que de changer leurs

paramètres numériques de définition (les paramètres de position par exemple), nous allons modifier un de leurs paramètres géométrique, c'est à dire le contour. Dans un premier cas, nous avons créé le nouveau contour de la Figure 46. Celui-ci remplacera le contour présentement actif dans la primitive.

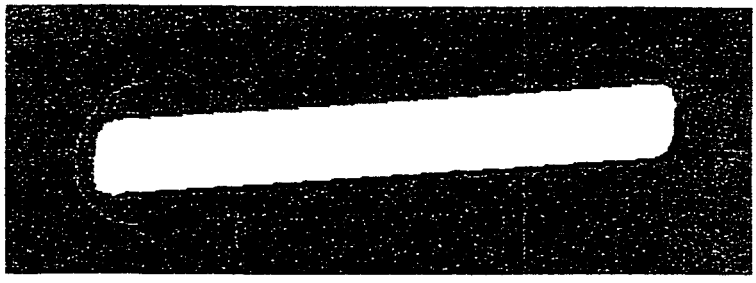


Figure 46 Modification d'un contour par remplacement

Dans un deuxième temps, comme démontré à la Figure 47, nous modifierons le contour d'une autre primitive de type PRISM, mais en utilisant cette fois-ci l'outil SKETCHER de CATIA. Cet outil permet de rapidement modifier les contours qui sont créés dans le modèle CAO. L'outil facilite la création et la modification d'entités simples sous une représentation en deux dimensions, comme sur une planche à dessin. Cet exemple permet d'illustrer qu'une modification est décelée, quelque soit la méthode utilisée. Dans notre cas et tel qu'illustré ci-dessous, nous allons simplement changer la distance séparant les segments parallèles, soit de 0.685po à 1po.

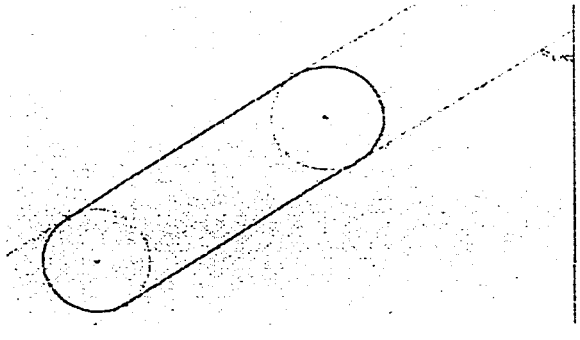


Figure 47 Modification d'un contour à l'aide de l'outil d'esquisse

Pour ces deux dernières modifications, l'application devra comparer les éléments géométriques contenus dans les paramètres de définition de la primitive.

4.2 Modification de paramètres numériques

Le premier modèle comparé lors de la période de validation du comparateur est une concaténation de trois volumes primitifs simples tel qu'illustré à la Figure 48. Les modifications apportées à la deuxième révision du modèle sont un changement de diamètre de la sphère et un changement de position de la sphère par l'entremise de la matrice de coordonnées.

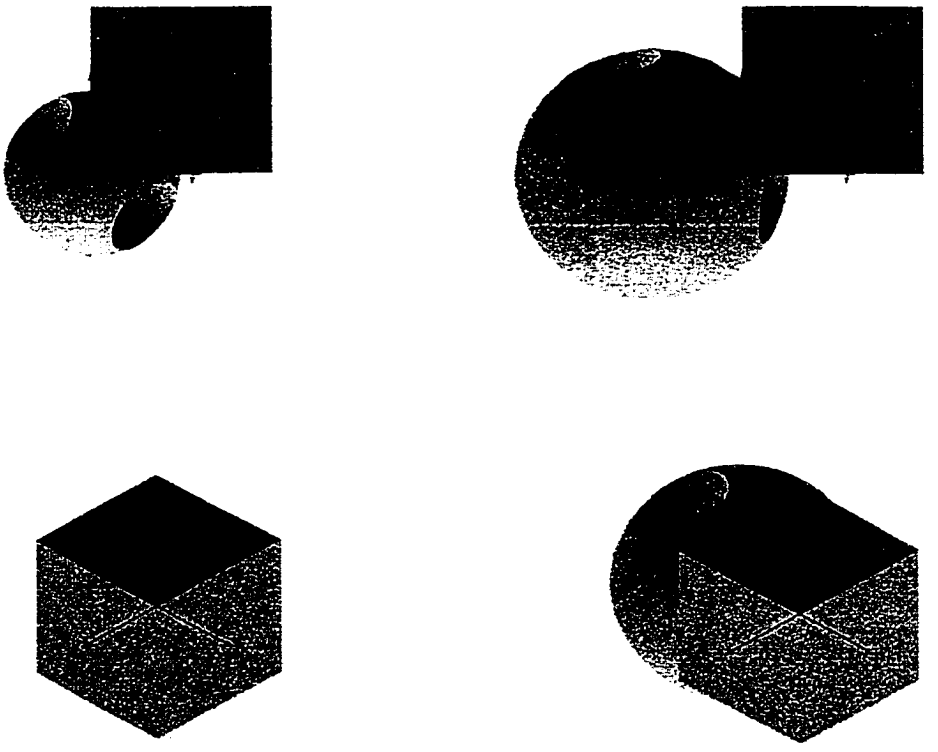


Figure 48 Essai de comparaison no1

Les résultats de la comparaison sont les suivantes (Figure 49):

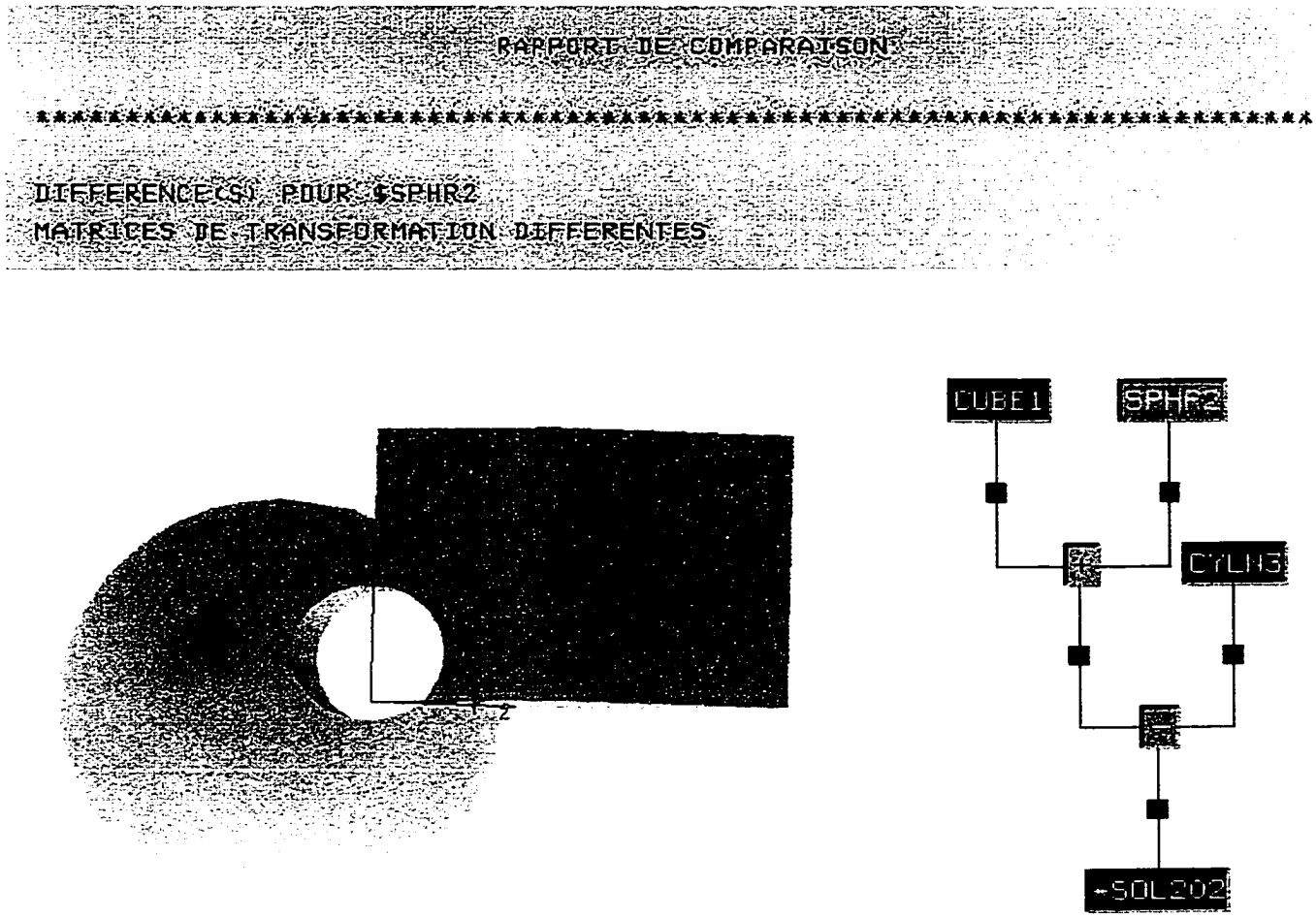


Figure 49 Résultats de l'essai no1

Modification (rév.1) :	Matrice de coordonnées et paramètres numériques de la sphère
Nombre de primitive :	3
Nombre de niveau :	2
Délai de comparaison :	0.5 seconde

4.3 Modification de la structure de l'arbre CSG

L'essai no 2 est réalisé à partir d'un modèle de cendrier. Dans ce cas, nous ne faisons aucune modification de primitives. Pour la première révision du modèle, nous permutons la position de deux primitives dans l'arbre de modélisation, tandis que pour la deuxième révision, nous changeons une des opérations booléennes de l'arbre par rapport au modèle initial. Ces modifications permettent de vérifier le comparateur lors de changements dans la structure des arbres CSG, tel qu'illustré à la Figure 50 et 51.

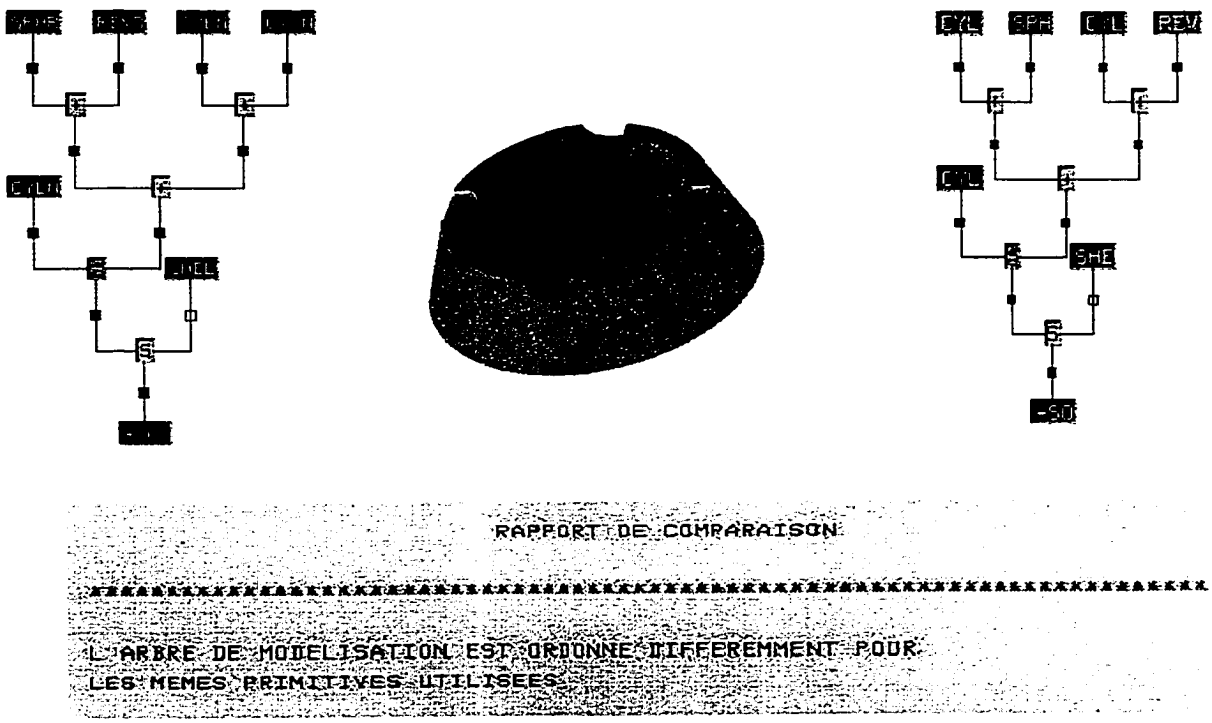
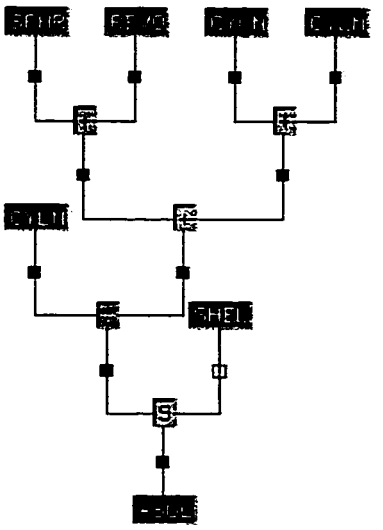
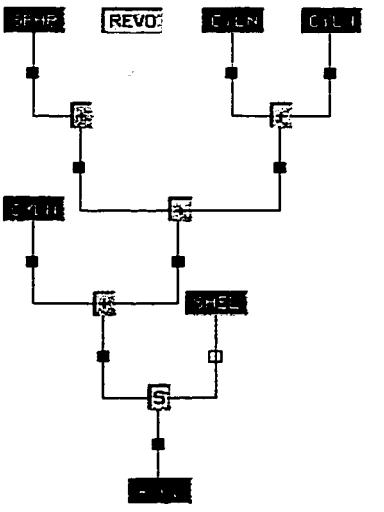


Figure 50 Essai no 2a

Modification (rév.1) :	Index de primitive
Nombre de primitive :	6
Nombre de niveau :	4
Délai de comparaison :	2 secondes






```
                    RAPPORT DE COMPARAISON
*****************************************************************

          ARBRE DE MODELISATION DU PREMIER SOLIDE

1.   UNION          ->   $SPHR2      $REVO3      N. 5001
2.   UNION          ->   $CYLN4      $CYLN6      N. 5002
3.   UNION          ->     -1           -2       N. 4001
4.   SOUSTRACTION   ->   $CYLN1        -3       N. 3001
5.   POCHE (SHELL)  ->     -4         $SHEL5     N. 2001

          ARBRE DE MODELISATION DU DEUXIEME SOLIDE

1.   UNION          ->   $SPHR3      $REVO4      N. 5001
2.   UNION          ->   $CYLN5      $CYLN6      N. 5002
3.   UNION          ->     -1           -2       N. 4001
4.   UNION          ->   $CYLN1        -3       N. 3001
5.   POCHE (SHELL)  ->     -4         $SHEL2     N. 2001
```

Figure 51 Essai no 2b

Modification (rév.2) : Opération booléenne

Nombre de primitive : 6

Nombre de niveau : 4

Délai de comparaison : 2 secondes

4.4 Modification de paramètres liés

L'essai no 3 de la Figure 52 est tiré d'une pièce aéronautique. Les modifications de ce modèle sont appliquées à des paramètres numériques. La distance de dégagement d'un mur (OFFSET) ainsi que la dimension du rayon d'un congé de raccordement ont été modifiés. Le rapport de comparaison de cet essai est illustré à la Figure 53.

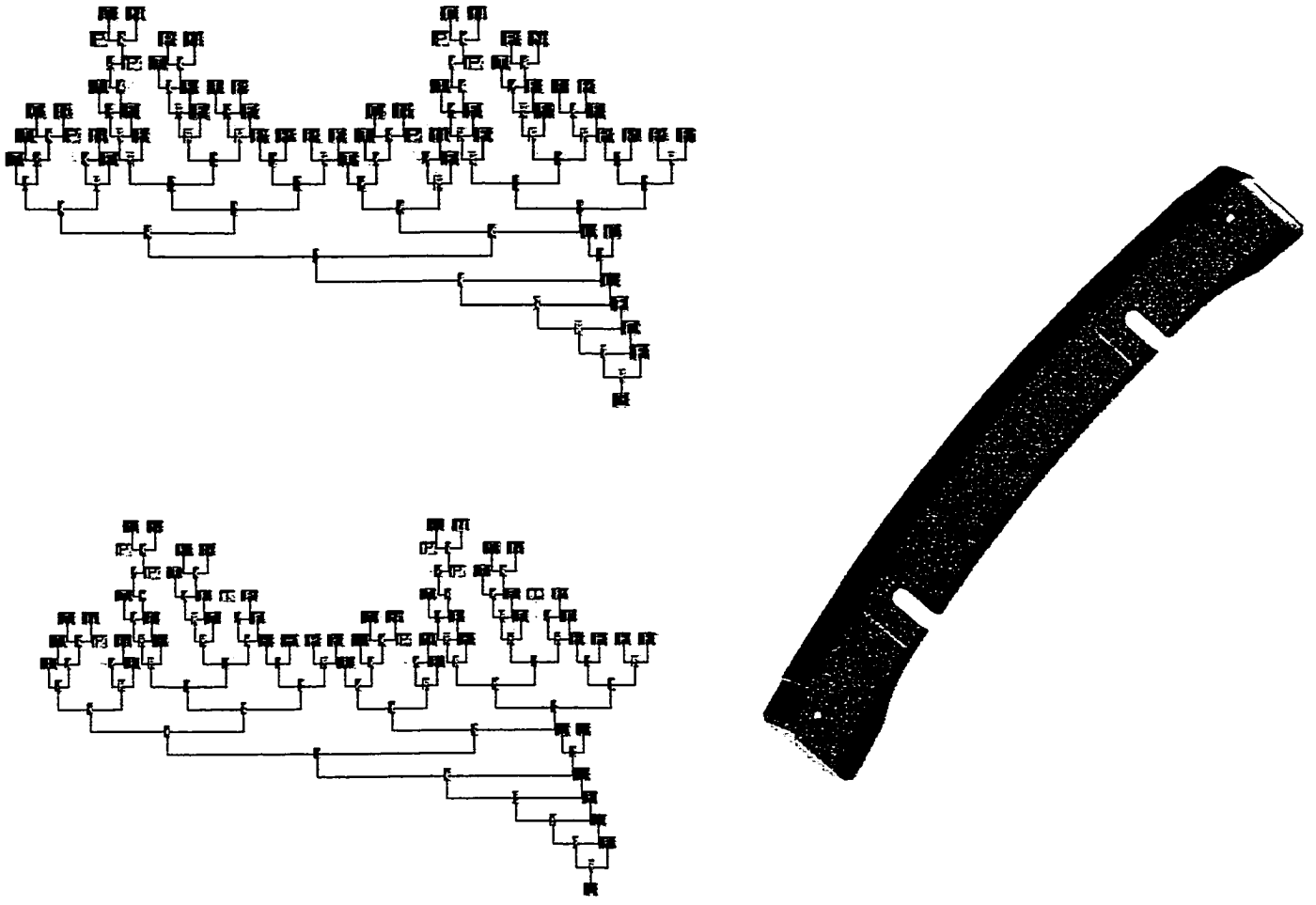
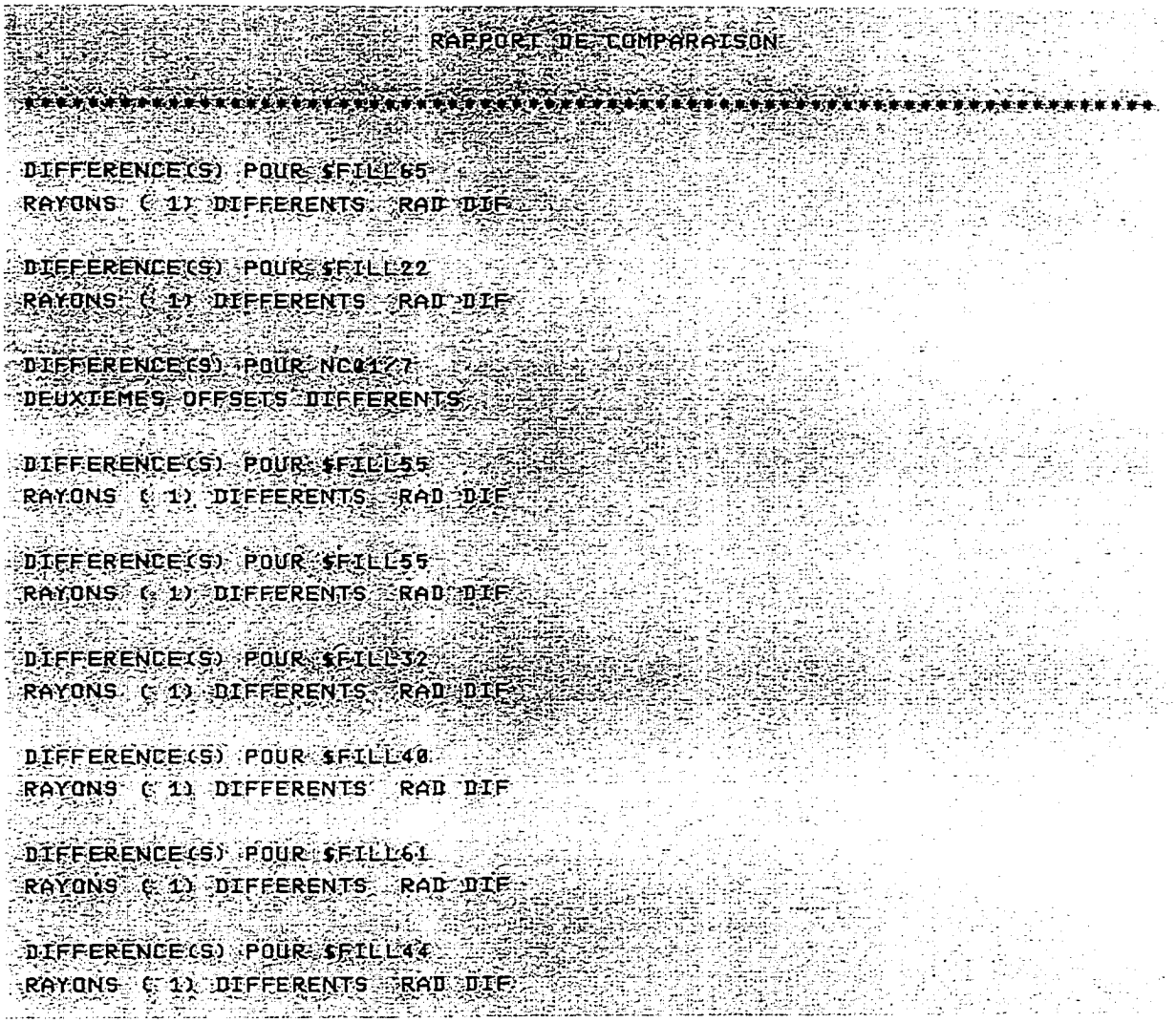


Figure 52 Essai no 3 – Arbres CSG



```
                        RAPPORT DE COMPARAISON

*****************************************************************

DIFFERENCE(S) POUR $FILL65
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR $FILL22
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR NC017
DEUXIEMES OFFSETS DIFFERENTS

DIFFERENCE(S) POUR $FILL55
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR $FILL55
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR $FILL32
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR $FILL40
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR $FILL61
RAYONS ( 1) DIFFERENTS  RAD DIF

DIFFERENCE(S) POUR $FILL44
RAYONS ( 1) DIFFERENTS  RAD DIF
```

Figure 53 Essai no 3 – Rapport de comparaison

Modification (rév.1) : Paramètres de primitive

Nombre de primitive : 59

Nombre de niveau : 15

Délai de comparaison : 11 secondes

4.5 Ajout et retrait de primitives

L'essai no 4 représenté à la Figure 54 est tiré d'une autre pièce aéronautique dont on obtient la première révision en faisant la translation d'une de ses ouvertures et en changeant le contour de sa deuxième ouverture.

Pour la deuxième révision du modèle, on fera également la suppression d'une primitive. La troisième révision sera, à l'opposé de la deuxième, un ajout de primitive par rapport au modèle initial. Finalement, la quatrième révision du modèle sera identique à la quatrième, sauf que la primitive ajoutée sera déplacée plus tôt dans l'arbre de modélisation. Cela revient à placer cette primitive plus haut dans la matrice de modélisation.

Ces modifications illustrent l'influence de la suppression et de l'ajout d'une primitive jà une position précise de l'arbre CSG du modèle. Les rapports de comparaison de l'essai 4 apparaissent aux Figure 55, 56 et 57.

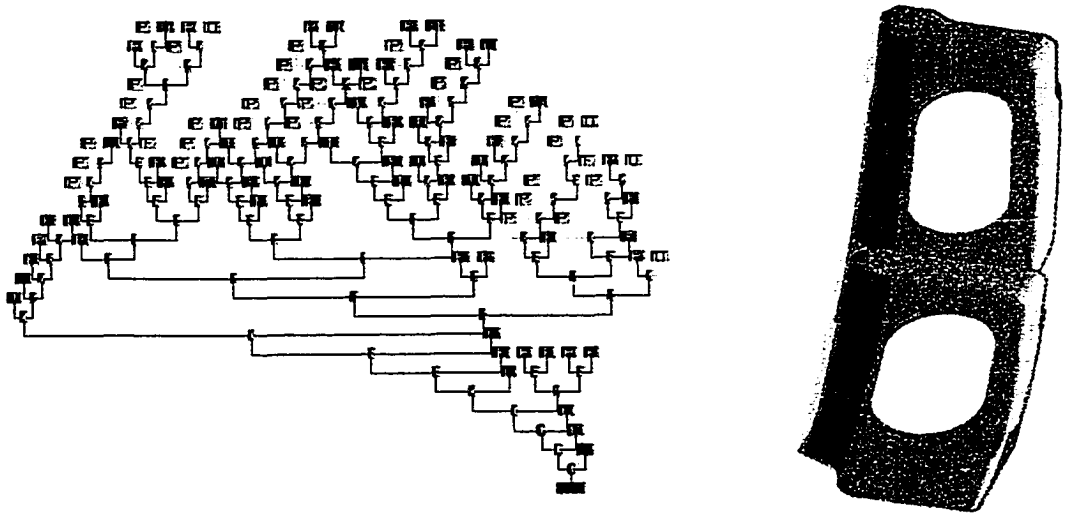


Figure 54 Essai no 4 – Arbre CSG et représentation solide

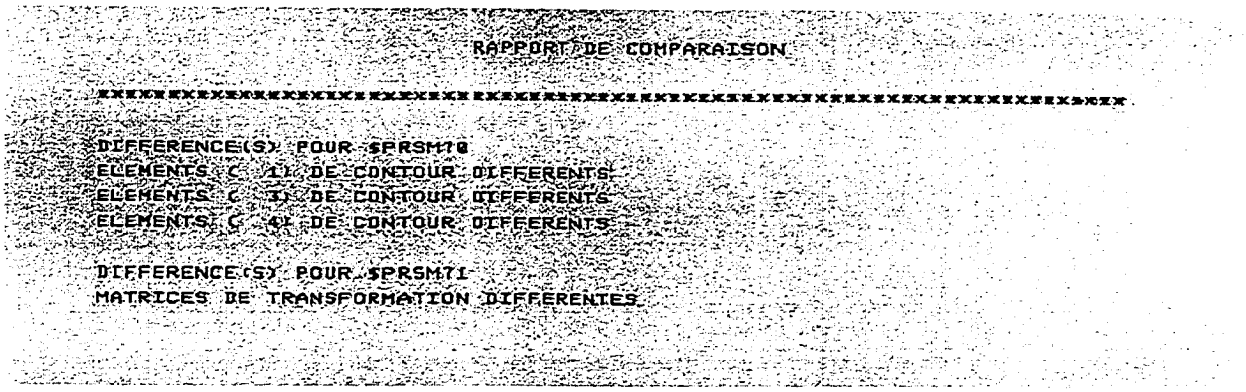


Figure 55 Essai no 4a – Translation et changement de contour des ouvertures

Modification (rév.1) : Matrice de coordonnées et paramètres de primitives

Nombre de primitive : 102

Nombre de niveau : 23

Délai de comparaison : 53 secondes

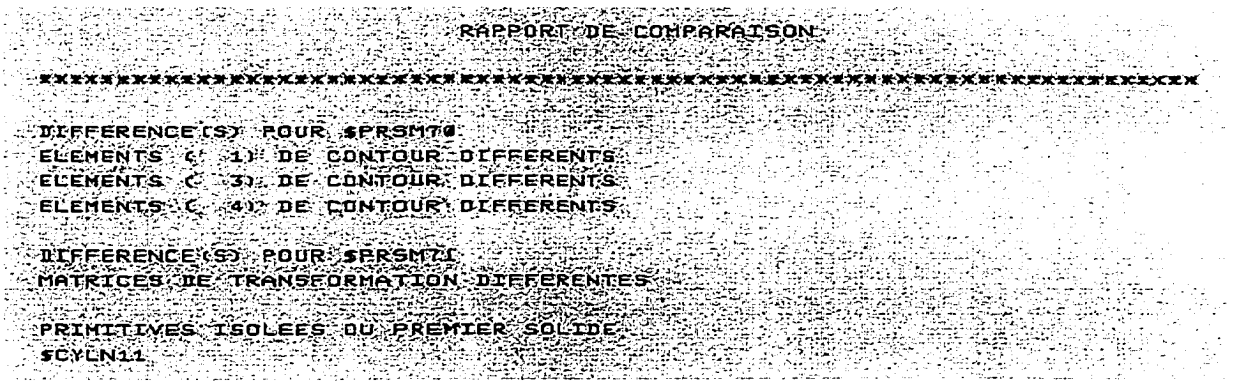


Figure 56 Essai no 4b – Même que 4a avec suppression d'une primitive

Modification (rév.2) : Matrice de coord. et paramètres de primitives, suppression

Nombre de primitive : 102

Nombre de niveau : 23

Délai de comparaison : 422 secondes

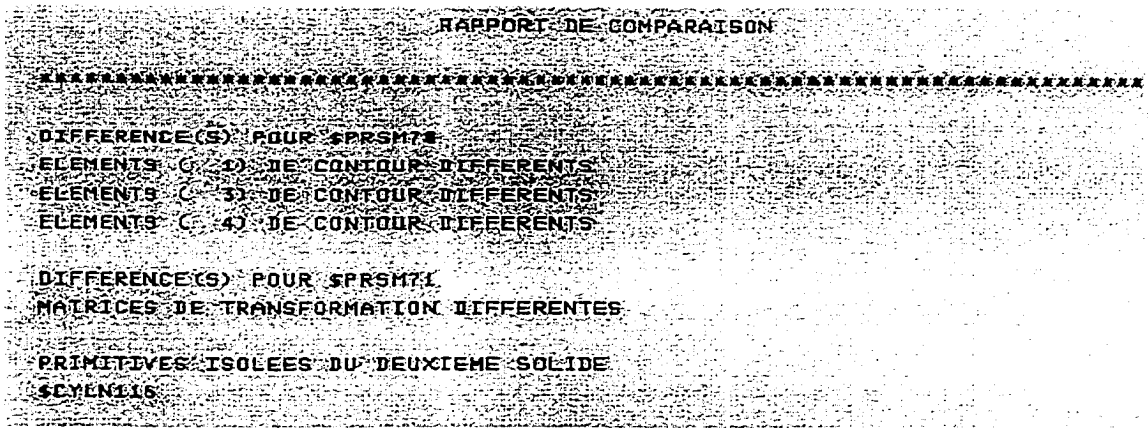


Figure 57 Essai no 4cd – Même que 4a avec ajout d'une primitive

Modification (rév.3) :	Matrice de coordonnées et paramètres de primitives, Ajout simple d'une primitive
Nombre de primitive :	102
Nombre de niveau :	23
Délai de comparaison :	280 secondes

Modification (rév.4) :	Matrice de coordonnées et paramètres de primitives, Ajout d'une primitive et déplacement dans l'arbre CSG
Nombre de primitive :	102
Nombre de niveau :	23
Délai de comparaison :	64 secondes

4.6 Modifications supplémentaires

Ces quatre derniers essais permettent de remarquer rapidement les performances du comparateur en terme de temps pour des modèles avec différents nombre de primitives. Des modèles de pièces en métal en feuille se retrouvent aussi parmi ces essais. Les pièces sont illustrés aux Figures 58, 59, 60 et 61.

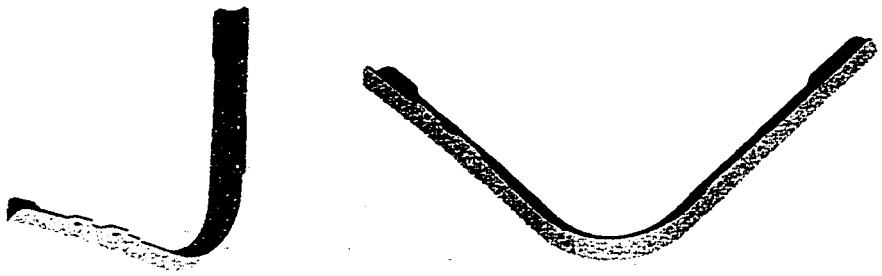


Figure 58 Essai no 5

Modification (rév.1) : Matrice de coord. et paramètre de primitive, suppression
Nombre de primitive : 24
Nombre de niveau : 15
Délai de comparaison : 23 secondes

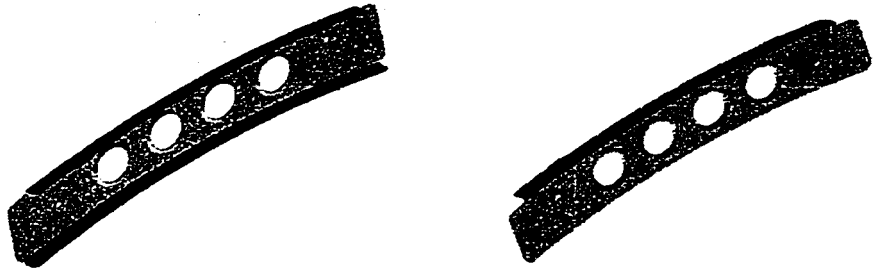


Figure 59 Essai no 6

Modification (rév.1) : Matrice de coordonnées et paramètre de primitive, ajout
Nombre de primitive : 16
Nombre de niveau : 10
Délai de comparaison : 15 secondes

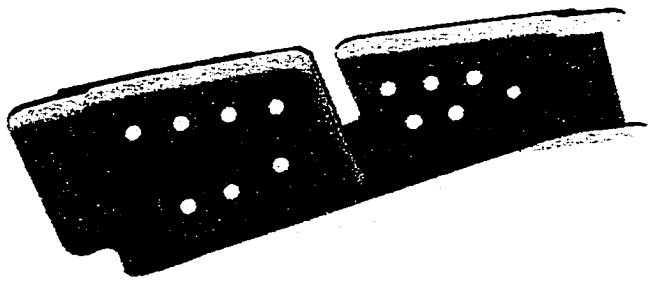


Figure 60 Essai no 7

Modification (rév.1) :	Matrice de coord. et paramètre de primitive, suppression
Nombre de primitive :	79
Nombre de niveau :	22
Délai de comparaison :	26 secondes

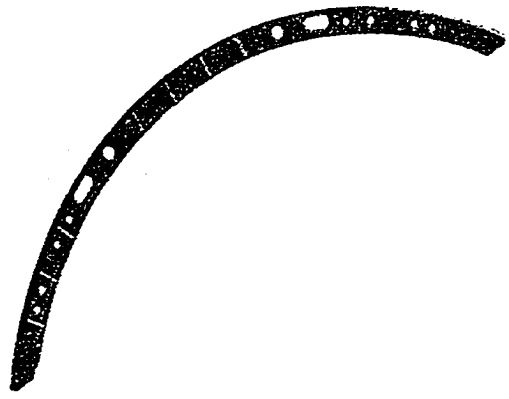


Figure 61 Essai no 8

Modification (rév.1) :	Matrice de coordonnées et paramètre de primitive, ajout
Nombre de primitive :	91
Nombre de niveau :	11
Délai de comparaison :	382 secondes

Les Figures 62 et 63 ci-dessous illustrent une récapitulation des essais de validation du comparateur.

Essai		Modification(s)	Nombre de primitives	Nombre de niveaux	Délai de comparaison (seconde)
	Démo	Matrice de coordonnées et paramètres de primitive	136	14	55
	1	Matrice de coordonnées et paramètres de primitive	3	2	0,5
	2a	Index de primitive	6	4	2
	2b	Opération booléenne	6	4	2
	3	Paramètres de primitive	59	15	11
	4a	Matrice de coordonnées et paramètres de primitive	102	23	53
	4b	Matrice de coordonnées et paramètres de primitive, suppression de primitive	102	23	422

Figure 62 Résumé des essais – première partie

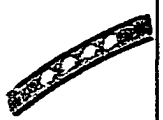
Essai		Modification(s)	Nombre de primitives	Nombre de niveaux	Délai de comparaison (seconde)
	4c	Matrice de coordonnées et paramètres de primitive, ajout de primitive	102	23	280
	4d	Matrice de coordonnées et paramètres de primitive, ajout stratégique de primitve	102	23	64
	5	Matrice de coordonnées et paramètres de primitive, suppression de primitive	24	15	23
	6	Matrice de coordonnées et paramètres de primitive, ajout de primitive	16	10	15
	7	Matrice de coordonnées et paramètres de primitive, suppression de primitive	79	22	26
	8	Matrice de coordonnées et paramètres de primitive, ajout de primitive	91	11	382

Figure 63 Résumé des essais – deuxième partie

DISCUSSION ET INTERPRÉTATION DES RÉSULTATS

La constatation importante que nous avons faite suite aux essais réalisés par le comparateur de modèles est l'identification systématique de toutes les modifications introduites aux modèles, autant dans leurs représentations solide que dans leur arbre CSG. Quelle que soit la modification géométrique introduite au modèle, celle-ci est décelée par le comparateur. Les seules modifications qui ne peuvent pas être retracées sont celles qui concernent le changement d'un paramètre numérique ou géométrique d'une primitive dont on a pas la routine d'extraction des données telles que les primitive de type CÔNE, TORE, VASE, SKIN, DRAFT et IMPORT. Les routines en question, sur CATIA à sa quatrième révision, n'étaient pas encore disponibles lors de la conception du comparateur.

À partir des modifications effectuées à la pièce industrielle BOMB3, nous réalisons à quel point il est avantageux de se servir d'un comparateur pour vérifier la similitude ou les différences entre deux révisions d'un modèle produit. Pour un total de sept modifications simples, le comparateur nous identifient les modifications réalisées en 55 secondes. C'est un excellent résultat lorsque l'on considère que cette pièce de 136 primitives prendrait plusieurs minutes, sinon des heures, pour en vérifier manuellement la similitude entre les révisions, surtout que certaines des modifications introduites sont difficilement perceptibles à l'œil.

Nous avons également observé qu'un seul type de modification est mentionné pour chaque primitive. C'est pourquoi, au premier essai, alors que nous avons opéré des changements à la matrice de coordonnées et à un paramètre numérique de la sphère, seule la modification de la matrice est mentionnée. En effet, l'algorithme du comparateur donne des priorités aux types de modification dans l'ordre dans lequel nous les avons classés, soit : type de primitive, matrice de coordonnées, paramètres de primitive et index dans l'arbre CSG. Un seul type de modification suffit pour que l'on

sache que la primitive a changé. De plus, on ne peut affecter qu'une seule couleur à une primitive. Par contre, une pluralité de modifications pourraient être signalées dans le rapport de comparaison. Cette option pourrait aisément être ajoutée parmi les fonctions du comparateur.

À partir de l'essai no 3, nous avons remarqué que plusieurs primitives étaient de couleur jaune, ce qui indique un changement de paramètre. Effectivement, lorsque nous avons modifié un des paramètres du congé de raccordement, toutes les primitives qui avaient le même paramètre par liaison ont également été modifiées. Il est alors tout à fait évident d'observer plusieurs primitives de type FILLET de couleur jaune, indiquant que leurs rayons de congé étaient liés ensemble dans le modèle.

Parmi les inconvénients du comparateur, il y a l'identification colorée des changements d'index. Pour la permutation de primitives, tel que démontrée à l'essai no 2, la modification est bien identifiée. Par contre, lorsqu'il y a ajout ou suppression de primitive, il est possible que la majorité des primitives de l'arbre CSG changent de niveau, donc d'index, surtout si la modification s'effectue vers la fin de la concaténation du modèle. Même si cette coloration répandue nous permet de situer localement une modification dans l'arbre, elle ne symbolise pas le degré d'importance du changement. Il peut alors être opportun de trouver une manière plus appropriée d'identifier les changements d'index selon les résultats que nous voudrions obtenir.

Toujours en ce qui concerne les suppressions et les ajouts de primitive, ces types de modification font varier la structure des piles de primitives du comparateur de modèles. L'algorithme de recherche et de correspondance des primitives n'a pas été optimisé pour des changements dans le nombre des primitives et la structure des piles les contenant. C'est la raison pour laquelle les modèles qui contiennent un grand nombre de primitives et qui subissent des ajouts ou des suppressions de primitives requièrent des délais de comparaison plus élevés que la normale, sachant qu'une comparaison dite

normale détient un délai de calcul inférieur à la minute (60 secondes). Mais puisque l'accent du projet n'est pas principalement axé sur les performances temporelles du comparateur, cet aspect peut faire parti de développements futurs concernant l'optimisation du comparateur.

Un autre cas de performance temporelle concerne les primitives à éléments géométriques. En effet, certaines primitives requièrent, en plus de paramètres numériques, des éléments géométriques pour être complètement définies. Si nous prenons l'exemple d'un balayage (SWEEP), ce dernier nécessite des éléments géométriques pour former son profil, d'autres éléments géométriques pour former la directrice, l'axe et la surface de référence de la directrice. Tous ces éléments géométriques doivent être vérifiés par d'autres routines de comparaison adaptées. Ce type de primitive accroît énormément la durée de l'analyse.

Pour l'instant, un examen extensif des opérations n'a pas encore été introduit dans l'application. Cet examen demanderait des algorithmes supplémentaires tirés du domaine de la théorie des graphes. Pour palier à ce problème, nous nous sommes plutôt tournés vers la comparaison des propriétés physiques et inertielles des modèles pour vérifier les résultats de comparaison. Généralement, ce sont des modifications aux primitives qui induisent des changements aux opérations utilisées dans l'arbre, et non le contraire. Ainsi, il n'est pas impératif d'effectuer une analyse systématique des opérations pour trouver les différences entre deux modèles. Nous n'avons besoin que d'une confirmation de la similitude géométrique parfaite entre les deux modèles. Ceci est accompli, en comparant leur surface ou leur volume, pour ne citer que celles-ci comme propriétés physiques des modèles.

On constate aussi comme inconvénient, la grande précision des paramètres géométriques de certaines primitives comme le balayage, le prisme et le volume complexe. Si nous prenons pour exemple un balayage, la surface de référence de la

directrice peut être un plan qui devra être analysé par une routine appropriée. Un plan est définit par neuf paramètres numériques qui peuvent bien contenir une quinzaine de chiffres avant comme après la virgule. Ces paramètres peuvent quelques fois être légèrement différents entre les deux modèles, surtout si c'est le logiciel et non l'ingénieur qui les crée (en effet, l'ingénieur peut s'être servi des autres éléments de sa session de travail pour créer le plan, sans utiliser de données numériques). Il faudra alors arrondir quelque peu ces paramètres pour en vérifier intelligemment la correspondance. Ce problème d'arrondissement survient généralement lorsque des entités primaires doivent être analysées (points, lignes, plans).

Finalement, nous avons observé qu'il serait possible d'améliorer le rapport de comparaison fourni à l'usager. Pour certaines modifications, il est pertinent d'être plus spécifique dans la description de leur nature. Par exemple, au lieu de dire que le rayon d'une primitive de type CYLINDER a changé, on pourrait indiquer que le rayon du trou $CYL001 a été réduit à 12.5mm. Un processus d'analyse pourrait être développé ultérieurement à cette fin.

Malgré tous ces points et dans les limites mentionnées ci-haut, l'application décèle toutes les différences géométriques entre deux modèles. Un effort de conception supplémentaire reste à être fourni quant à la description enrichie de la nature des modifications, d'une révision à une autre d'un modèle.

CONCLUSION

En somme, les PDM constituent une solution intéressante pour maintenir un contrôle adéquat sur tous les fichiers que les entreprises utilisent pour concevoir et fabriquer des produits. La gestion des modifications fait parti d'une multitude de facettes des systèmes de gestion des données techniques (SGDT) qui permettent de suivre et de contrôler un produit tout au long de son cycle de développement. Par contre, nous avons démontré qu'un outil plus convivial et mieux adapté était nécessaire pour la gestion du contenu des fichiers de modèles CAO. Nous avons souligné que l'étendue d'utilisation des PDM s'arrête à la gestion de fichiers. Par conséquent, d'autres outils sont essentiels pour l'extraction et l'interprétation des données à l'intérieur de ces fichiers.

Le comparateur de modèle solide CAO que nous avons conçu réussit, à l'intérieur de certaines limites, à identifier les différences entre les révisions d'un même modèle. Il est à noter que la pertinence de cet outil s'inscrit précisément dans l'évolution d'un modèle CAO. Il n'est pas approprié pour la comparaison de deux modèles quelconques, malgré que des résultats seront exprimés néanmoins.

L'application qui a été conçue dans le cadre de cette recherche constitue un prototype d'avant-garde dans le désir de combler les besoins pressants suscités dans le secteur de l'ingénierie assistée par ordinateur. Compte tenu des bons résultats obtenus, il est tout à fait propice d'envisager la poursuite des travaux accomplis, en particulier en ce qui a trait au contrôle de la structure des arbres de modélisation et de l'extraction de la nature des modifications des révisions de modèles.

ANNEXE 1

LES SYSTÈMES DE TYPE PDM

LES SYSTÈMES DE TYPE PDM

Il existe présentement une dizaine de fournisseurs de PDM sur lesquels on peut facilement s'informer dans le marché de la technologie de l'information. Parmi les PDM disponibles, certains ont été conçus spécifiquement pour les industries financières tandis que d'autres ont pour but la gestion de données dans les entreprises manufacturières. Il est utile de se familiariser avec quelques-uns des fournisseurs présents ainsi que les particularités de leur produit.

Les principaux fournisseurs

Les PDM sont légèrement différents entre eux concernant les modules offerts. Dans l'éventualité où l'on veut faire l'acquisition d'un PDM, on suggère d'abord d'évaluer les besoins de l'entreprise et de les comparer avec les options présentées par les fournisseurs suivants :

- Agile Software Corp.
- CMstat Corporation
- Enovia Corporation
- IFS Industrial & Financial Systems
- McLaughlin Research Corporation
- NovaSoft System, Inc.
- Parametric Technologies Corporation
- Parametric Technology Corporation
- SDRC
- Sherpa Corporation
- Workgroup Technology, Inc.

Les modules de gestion des modifications

La plupart des PDM contiennent un module de gestion des modifications. En général, c'est le type de gestion ainsi que le type de modification qui diffère entre les multiples PDM sur le marché (il faut se rappeler ici des distinctions décrites à la Section 7.3).

Voici les types de modules de gestion des modifications que l'on retrouve parmi les PDM des fournisseurs répertoriés à la Section précédente :

- Contrôle des révisions de fichiers
- Archivage de l'historique des fichiers comprenant l'auteur, la révision originale, les modifications apportées, les révisions, etc.
- Gestion automatisée des données de produits ainsi que des ordres de changement
- Gestion des modifications et contrôle de l'effectivité des pièces
- Gestion des changements : requêtes de changement, ordres de changement, modifications des tâches, changement d'effectivité.
- Journal de bord informatique répertoriant le statut de toutes les modifications
- Gestion des configurations

Dans le cas où l'on veut se procurer un PDM pour administrer les modifications du produit à fabriquer, il est faut préalablement connaître le type de modifications que l'on veut contrôler, à l'instar des types de modules de gestion que l'on vient d'énumérer plus tôt.

ANNEXE 2

CARACTÉRISTIQUES DES PDM / VPDM

CARACTÉRISTIQUES DES PDM / VPDM

Dans la Section suivante, nous décrivons brièvement les habiletés principales des systèmes de type PDM 2 que nous avons retrouvé sur le marché. Il est à noter que l'information n'est parfois que conceptuelle puisque la majorité des données se trouvent à l'intérieur d'annonces ou d'articles promotionnels.

Enovia VPM

$ENOVIA_{VPM}$ est un outil qui vise le marché des VPDM (gestionnaires de développement de produits virtuels). Ces outils initient et encouragent surtout l'innovation des acteurs par rapport à la conception du produit. $ENOVIA_{VPM}$ couvre les aspects techniques suivants :

- Une gestion flexible des configurations capable de tenir compte de la définition de plusieurs produits en parallèle ou successifs afin d'être évalués et optimisés.
- Des niveaux multiples de visibilité afin de gérer les modifications et les caractériser globales ou locales avant la distribution générale de l'information.
- Un contrôle sur la maturité des modifications pour rendre l'information disponible aux bons acteurs et au bon moment.
- Un mécanisme pro actif qui permet aux usagers de s'inscrire à des événements particulier afin que le système les informe immédiatement de toute modification.
- Une organisation basée sur le principe du flux des activités qui s'harmonise avec les tâches d'ingénierie lors de l'optimisation du concept du produit, autant pour les concepts reliés au produit que pour ceux reliés aux services connexes.
- Une gestion du produit structurée graphiquement dans laquelle sont inclus la configuration des pièces et leurs mécanismes de mise à jour.

- Une méthode de stockage et de distribution des données du produit qui se trouvent normalement dans les fichiers d'outils de CAO.
- Une indépendance au modeleur CAO.
- Une structure des services de sécurité et d'autorisation qui rejoint les particularités de l'entreprise.
- Des supports pour les maquettes virtuelles numérisées, la conception de l'usine et la numérisation des applications de production.
- Une habileté à effectuer la configuration de la documentation technique afin qu'elle reste synchronisée avec les modifications apportées au produit
- L'utilisation de standards appropriés tels que STEP, CORBA, OLE/DCOM et encore.

Enovia $_{PM}$

ENOVIA$_{PM}$ est le module de gestion des données du produit offert par ENOVIA Corporation. C'est un des systèmes PDM de premier plan dans l'industrie. Quelques-unes de ses capacités distinctives sont énumérées à travers les points suivants :

- Une stratégie incorporant les meilleures pratiques industrielles permettant de réduire considérablement le temps requis pour implanter et optimiser le système.
- Une ouverture et une flexibilité qui permet d'adapter ENOVIA$_{PM}$ aux spécifications des usagers par un module graphique puissant de personnalisation.
- Une gestion de sites multiples couplée au partage des données et au service de distribution sélective à travers toute l'entreprise.
- Un module très développé de solutions d'intégration concernant d'autres systèmes tels que les outils de CAO, les ERP, l'automatisation des processus et l'inventaire des pièces.
- Des supports pour des environnements informatiques de grande échelle tels des stations de travail UNIX, des PC, des réseaux et des systèmes d'archivage.

AgileWorks

AgileWorks est un puissant gestionnaire de fichiers s'apparentant à un PDM mais exclusivement dans l'environnement du système de CAO SolidWorks. Il atteint certains fonctions propres à la gestion de fichiers où les méthodes d'ingénierie simultanée sont utilisées. Voici quelques caractéristiques techniques du système AgileWorks :

- L'utilisation d'une voûte centrale pour la gestion des fichiers et le contrôle des révisions des fichiers.
- Une coordination accrue de la gestion des fichiers pour les acteurs travaillant en parallèle sur les produits ou sous-produits.
- La conservation de l'historique des fichiers comprenant l'auteur, la révision originale, les modifications apportées, les révisons, etc.
- La gestion des fichiers à l'aide d'attributs pour fin de recherches.
- Une architecture client/serveur jumelée à l'interface graphique Windows NT
- La gestion automatisée des données du produits ainsi que des ordres de changement.

OPTEGRA$_{TM}$ Configuration Master

OPTEGRA$_{TM}$ fournit à ses utilisateurs des capacités à emmagasiner et à gérer toutes les données reliées à la configuration du produit le long de son cycle de vie. Chaque caractéristique du produit est acquise dans le système, incluant les modifications survenues depuis la définition du concept jusqu'au retrait du produit sur le marché. OPTEGRA$_{TM}$ possède les caractéristiques suivantes :

- Une architecture client/serveur capable de gérer un accès à l'information par des usagers travaillant en parallèle.
- Une voûte centrale pour les données recueillies dans son système de gestion des données d'ingénierie mais nécessitant Oracle RDBMS
- Une interface graphique pourvue d'icônes, de menus déroulants, de panneaux, de liens hypertextes et de formats de visualisation personnalisée.
- Un fureteur de recherche par catégories.
- Une gestion de la liste des composantes.
- Une gestion des modifications et un contrôle de l'effectivité des pièces.
- Une gestion des références extérieures (exemple : dessins de CAO)
- Une possibilité d'intégration avec des système de gestion de la production.
- Une utilisation du système sur des plates-formes de travail Sun Solaris ou Hewlett-Packard HP-UX.

PDM Windchill™

Le gestionnaire de données de produit Windchill fait partie d'une suite d'applications offert par la compagnie Parametric Technology Corporation. Le PDM fournit un support complet pour la gestion et la transmission des informations concernant la structure du produit et les modifications qui y sont apportées tout au long de son développement et de sa production. Voici les quelques capacités du système :

- la création simplifiée d'une variété de représentation de la liste des composantes,
- la gestion des bons de commande en matériaux par plusieurs intervenants,
- la production automatique de rapports concernant les bons de commande,
- la gestion de l'effectivité des pièces selon les modifications apportées au concept du produit et sa révision courante,
- la possibilité de relier des références extérieures aux données du produit tels que des dessins de CAO, des fichiers multimédias ou des sites WEB,
- une architecture totalement centrée sur les applications WEB,
- une gestion des changements : requêtes de changement, ordres de changement, modifications des tâches, changements d'effectivité,
- une intégration avec le système CAO Pro/ENGINEER en cours de réalisation.

IFS / PDM-Configuration™

IFS (Industrials & Financial Systems) offre une application de type PDM jumelée à un organisateur des ressources de l'entreprise (ERP). Pour le développement d'un produit ou pour une modification du concept présent, IFS / PDM-Configuration permet de spécifier les pièces ou les assemblages à travailler tout en créant les documents associés à ces tâches. Conçu pour être couplé à un système de CAO, le PDM suit l'évolution du produit durant son cycle de vie. Nous voyons ci-dessous, une énumérations de certaines habiletés de l'application offerte par IFS :

- Un registre commun des pièces associées à des données techniques comme référence des départements de production et de distribution.
- Les manufacturiers des pièces peuvent être branchés au système d'IFS pour une réaction rapide à la demande et aux changements.
- Des coûts peuvent être obtenus par le département de distribution pour l'optimisation du calcul d'estimation des coûts d'ingénierie.
- Un sommaire des coûts des ouvrages et des pièces peut être extrait.
- La base de données contient un fureteur de recherche pour trouver la meilleure solution parmi les nouveaux designs.
- Des niveaux simples ou multiples sont disponibles pour le représentation graphique de la structure du produit.
- Un journal de bord automatique répertorie le statut de toutes les modifications.
- Un fichier de CAO peut être directement attaché aux données du produit à travers le module IFS / Document Management.
- Des attributs librement définis peuvent être appliqués au format principal de la base de données de façon à ce que le tout s'harmonise au type d'organisation de l'entreprise.

Metaphase Enterprise 3.0 Product Suite

Cette suite de produits de type PDM a été conçue spécifiquement pour les entreprises internationales qui veulent optimiser les procédés de leur organisation associés à la conception, la production, l'entretien et la livraison des produits fabriqués. La suite offerte par la compagnie SDRC remplit les fonctions suivantes :

- la gestion de données de produit,
- la gestion et le contrôle des documents,
- la gestion des configurations,
- le flux et l'assignation des tâches,
- la gestion des cycles de révision et d'approbation,
- la réingénierie des procédés de l'entreprise,
- l'analyse de l'impact des modifications,
- la gestion du changement,
- et l'ingénierie simultanée.

Voici quelques modules de la suite Metaphase Enterprise 3.0 :

- l'interface d'accès à l'information ***e!Vista,***
- l'organisation de la gestion des items,
- le gestionnaire du cycle de vie du produit,
- le gestionnaire du contrôle des modifications,
- le gestionnaire de la structure du produit,
- la configuration avancée du produit,
- le gestionnaire des familles de pièces,
- l'interface d'accès commercial ***MetaWeb***
- et la trousse d'intégration des applications connexes.

SherpaWorks Enterprisewide Product Data Management

Inso Corporation est le leader en développement de PDM et gestionnaires d'informations sur le produit. C'est un système de type PDM que la compagnie offre aux entreprises prêtes à aborder les applications étendues de l'organisation. Le système comprend les applications suivantes :

- un gestionnaire des documents,
- un gestionnaire des modifications,
- un gestionnaire de la définition du produit,
- un gestionnaire des procédés de fabrication,
- un administrateur des opérations de l'entreprise,
- un administrateur des systèmes,
- un service de distribution et d'archivage des fichiers
- et un système de sécurité pour la transaction d l'information.

SherpaWorks contient également un récent gestionnaire de développement de produits virtuels (VPDM) qui s'appuie essentiellement sur la disponibilité de maquettes virtuelles créées par des systèmes de CAO et distribuées à l'ensemble de l'entreprise. SherpaWorks est compatible avec les applications suivantes :

Conception mécanique assistée par ordinateur (MCAD)

- AutoCAD
- Autodesk Mechanical Desktop
- CATIA
- I-DEAS Master Series
- Pro / INTRALINK
- Unigraphics

Conception électrique assistée par ordinateur (ECAD)

- Cadence
- Mentor Graphics
- ViewLogic

Organisation des ressources de l'entreprise (ERP)

- BAAN
- CINCOM
- Oracle Manufacturing
- QAD
- SAP R/3

Le système peut être utilisé sur les plates-formes suivantes :

- Intel Windows NT
- Hewlett Packard HP-UX
- Sun Solaris

BIBLIOGRAPHIE

[1] Davis M., O'Sullivan D. (1999), *Systems Design Framework for the Extended Enterprise*, Production Planning and Control, 10(1), pp.3-18

[2] Sanderson S.W. (Dec 1991), *Cost models for evaluating virtual design strategies in multicycle product families*, Journal of Engineering and Technology Management, 8(3-4), pp.339-358

[3] Bryan M.G., Sackett P.J. (Août 1997), *The point of PDM*, Manufacturing Engineer, IEE Stevenage Engl, 76(4), pp.161-164

[4] Ngô A.D., Youssef A.Y. (Mar 1995), *Ingénierie simultanée – Principes, organisations et outils*, Notes de cours, École de Technologie Supérieure (É.T.S.), p.9

[5] Wiebe E.N.(1998), *Impact of Product Data Management (PDM) trends on Engineering Graphics Instruction*, Engineering Education : Contributing to U.S. Competitiveness ; Annual Conference & Exposition 1998, Seattle, Washington, 28 juin-1er juillet

[6] Peltonen H., Pitkänen O., Sulonen R. (1996), *Process-based view of product data management*, Computers in Industry, 31, pp.195-203

[7] Joseph J., Shadowens M., Chen J., Thompson C. (Oct 1991), *Strawman reference model for change management of objects*, Computer Standards & Interfaces, 3(1-3), pp.249-269

[8] Dong A., Agagino A.M. (1998), *Managing design information in enterprise-wide CAD using 'smart drawing'*, Computer-Aided Design, 30(6), pp.425-435

[9] Aoyama M. (Août 1996), *Sharing the Design information ia a distributed Concurrent Development of Large-Scale Software Systems*, Proceedings-IEEE Computer Society's International Computer Software & Applications Conference Proceedings of the 1996 IEEE 20th Annual International Computer Software & Application Conference, COMPSAC'96, Seoul,,, S Korea, Sponsored by IEEE IEEE Los Alamitos CA USA pp.168-175:0730-6512 CODEN : PSICD2

[10] Gardan Y., Minich C. (1992), *La modélisation géométrique et l'extraction de caractéristiques de forme*, Revue de CFAO et d'informatique graphique, 7(3), pp.311-333

[11] Leinen S., Jung J.-P., Gardan Y. (1997), *Comparaison de modèles de CAO par normalisation de graphes*, Revue de CFAO et d'informatique graphique, 12(1-2), pp.153-167

[12] Gardan Y., Minich C. (Nov 1993), *Featured-based models for CAD/CAM and their limits*, Computer in Industry, 23(1-2), pp.3-13

[13] Miller E. (Oct 1998), *PDM moves to the mainstream*, Mechanical Engineering, 120(10), pp.74-79

[14] Chen Y.-M., Liao C.-C. (Sep 1998), *A Systematic Approach of Virtual Enterprising Through Knowledge Management Techniques*, Concurrent Engineering, 6(3), pp.225-244

[15] Browne J., Sackett P.J., Wortmann J.C. (1995), *Future manufacturing systems – Towards the extended enterprise*, Computers in Industry, 25, pp.235-254

[16] Lee G.-H., Kim Y.-G. (Août 1998), *Implementing a Client/Server System in Korean Organizations :Interrelated IT Innovation Perspective*, IEEE Transactions on engineering management, 45(3), pp.287-295

[17] Hameri A.-P., Nihtilä J. (1998) , *Product data management – exploratory study on state-of-the-art in one-of-a-kind industry*, Computers in Industry, 35, pp.195-206

[18] Dwyer J. (Mars 1997), *PDM solutions – but what are the problems ?*, Manufacturing Computer Solutions, pp.40-43

[19] Engineering (Juin 1997), *Tougher demands, new solution*, Engineering, Project Management, pp.27-29

[20] Krishnamurthy K., Law K.H. (1995), *Révision management in a CAD paradigm*, Proceedings of the Computers in Engineering Conference and the Engineering Database Symposium, ASME, pp.1133-1144

[21] Krishnamurthy K., Law K.H. (Juin 1995), *Change management for Collaborative Engineering*, Computing in Civil Engineering (New York), pp.1110-1117

[22] Jansen M.H., Vermeer B.H.P.J., Jagdev H.S. (1997), *Towards a topology of electronic product information distribution*, Computers in Industry, 33, pp.395-409

[23] Song U., Nagi R. (1997), *Design and implementation of a virtual information system or agile manufacturing*, IEEE Transaction, 29, pp.839-857

[24] Waddell N., Sinclair M. (1994), *Extended Enterprise Patnerships*, Husat Research Institute ; IEE, Savoy Place, London WC 2R OBL, UK

ADRESSES INTERNET

http://www.docmanage.com/Docs/WhitePaper/ibm.htm [25]

http://www.sap.com

http://www.enovia.com

http://www.pdmic.com/articles/ssomgapx.shtml

http://www.gsia.cmu.edu/bb26/70-456/projects/sap/intro.html

http://www.sapfans.com/sapfans/saphist.htm

http://www.clearlake.ibm.com/MFG/solutions/

http://www.choosesmart.com

http://www.agilesoft.com

http://www.konfig.com

http://www.auto-trol.com

http://www.pdmic.com/cv/

http://www.ptc.com/products/windchill/modules/pdm/htm

http://www.sme.org/News/get/1999/05/04

http://www.ptc.com/news/inthenews/articles/1999_03_72.htm

http://www.ptc.com/news/inthenews/articles/1999_10_52.htm

http://www.ptc.com/products/cadds/optegra.htm

http://www.novasoft.com/web/products/document_management.htm

http://www.metaphasetech.com/products.html

http://www.inso.com/sherpaworks/sherpatb.htm

http://www.wokrgroup.com/ProductCenter.html