ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE MAÎTRISE EN GÉNIE MÉCANIQUE
M.Ing.

PAR
KHALED SOUFI

GESTION DE LA PROPAGATION DES MODIFICATIONS À TRAVERS DES DOCUMENTS MÉTHODES

MONTRÉAL, LE 16 JANVIER 2006

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. Louis Rivest, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Alain Desrochers, Professeur et vice-doyen aux ressources
Université de Sherbrooke

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

LE 16 JANVIER 2006

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

GESTION DE LA PROPAGATION DES MODIFICATIONS À TRAVERS DES DOCUMENTS MÉTHODES

Khaled Soufi

SOMMAIRE

Les systèmes de gestion du cycle de vie d'un produit (Product Life Cycle Management PLM) gèrent les informations relatives à la définition d'un produit. En particulier, ils permettent de gérer les liens entre les documents méthodes définissant la géométrie, le processus d'usinage et le processus d'inspection d'un produit. Si l'un de ces documents est modifié, les systèmes PLM permettent de visualiser les documents qui pourront être impactés par cette modification. Cependant, ces systèmes ne permettent pas de localiser les modifications dans les documents méthodes suite à un changement survenu dans un modèle de Conception et Fabrication Assistées par Ordinateur (CFAO).

Ce rapport présente l'étude de la propagation des modifications d'ingénierie à travers des documents méthodes. L'étude porte sur l'amélioration de la propagation des modifications d'ingénierie au cours du cycle de vie d'un produit. Lorsqu'un document méthode est modifié, il est important de déterminer avec précision les autres documents impactés par cette modification. La définition d'un produit durant tout son cycle de vie est assurée par des applications informatiques. La maîtrise de la propagation des modifications passe nécessairement par la communication entre ces applications. D'où l'idée de créer des « entités parlantes » au sein des documents méthodes et de créer des liens entre elles. Ces entités parlantes, lorsque modifiées, assurent la notification des autres entités.

Le modèle propose d'établir un module central de gestion de la propagation des modifications. Les différentes applications participants à la définition du produit sont intégrées à ce module. Si une entité est modifiée, elle informe le module central qui, grâce à une base de données des liens, permettra de simuler l'impact de cette modification sur les autres entités contenues dans divers documents. Ce module permet aussi l'exécution de la propagation de modification. Ce modèle servira à pallier au manque de fonctionnalités offertes par les gestionnaires de cycle de vie des produits (systèmes PLM).

Une maquette a été réalisée afin de prouver la faisabilité du modèle proposé. Les résultats obtenus sont prometteurs. Des notifications de changement sont envoyées par le système de CFAO. Ces notifications sont traitées et le schéma de propagation est dressé, montrant avec précision les autres entités impactées. Le processus de propagation des modifications peut être automatisé selon des règles préétablies.

CHANGE MANAGEMENT AND PROPAGATION SYSTEM THROUGHOUT ENGINEERING DOCUMENTS

Khaled Soufi

ABSTRACT

The Product Life Cycle Management (PLM) systems manage information related to the product definition. Especially, they manage links between documents defining the geometry, the machining process and the inspection process of a product. If one of these documents is modified, PLM systems can visualize the documents which could be impacted by this modification. However, these systems can't locate exactly the changes inside the documents following a change which has occurred in a Computer Aided Design (CAD) model.

This report presents the study of the engineering change propagation throughout engineering documents. The study relates to the improvement of engineering change propagation during the product life cycle. When an engineering document is changed, it is important to determine with precision the other documents impacted by this change. The product definition during all its life cycle is ensured by data-processing applications. The change management control has to go through a better communication between these applications at very detailed level. The idea, than, is to create "smart entities" within the engineering documents and to create links between them. These smart entities, when modified, ensure the notification of the other entities.

The study proposes to establish a central change propagation management module. The various applications participating into the product definition are integrated into this module. If an entity is modified, it informs the central module which, thanks to a links database, will simulate the impact of this modification on the other entities contained in various documents. This module also allows the change propagation execution. This model will be used to mitigate the lack of functionalities of the commercial PLM systems.

A prototype was developed in order to prove the feasibility of the proposed model. The results are promising. Change notifications are sent by the CAD/CAM system. These notifications are treated and the propagation diagram established showing, with precision, the other entities impacted. The change propagation process can be automated according to pre-established rules.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont aidé à la préparation du présent mémoire. Particulièrement, mes deux professeurs M. Jean François Châtelain et M. Roland Maranzana qui m'ont encadré et guidé tout au long de ce projet de recherche.

Mes remerciements s'adressent également à tous les collègues du Laboratoire d'Ingénierie des Produits, Procédés et Systèmes (LIPPS).

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET DES SIGLES

CAO	Conception Assistée par Ordinateur
FAO	Fabrication Assistée par Ordinateur
CFAO	Conception et Fabrication Assistées par Ordinateur
FEA	Finite Element Analysis
ECR	Engineering Change Request
ECO	Engineering Change Order
ECN	Engineering Change Notification
PLM	Product Life Management
PDM	Product Document Management
cPDM	collaborative Product Document Management
ERP	Enterprise Resource Planning
EDM	Enterprise Data Management
PIM	Product Information Management
TDM	Total Document Management
OMG	Object Management Group
UML	Unified Modeling Language
DAO	Dessin Assistée par Ordinateur
GHEC	Graphe Hiérarchique d'Entités et des Contraintes
LIPPS	Laboratoire d'ingénierie des produits, procédés et systèmes
CM	Configuration Management
l'URA	Unified Representation of an Artifact
API	Application Program Interface
BOM	Bill Of Material
STEP	Standard for the Exchange of Product
IGES	Initial Graphics Exchange Specification
STL	Stereolithography format
OLE	Object Linking and Embedding

INTRODUCTION

L'adoption du concept de l'ingénierie concourante a permis aux entreprises de réduire le temps de mise en marché « time to market » et de faire face à une concurrence féroce. Les différents intervenants travaillent en collaboration afin de définir le produit. Ils utilisent divers moyens de traitement des données du produit. Au cours de l'évolution de la définition du produit, plusieurs informations s'accumulent et sont généralement sauvegardées sous différents formats de documents. Dans le cas des produits aéronautiques, la quantité de documents est très importante. Par exemple, on peut compter près de 150,000 documents associés au développement d'un avion d'affaires. Des difficultés de gestion des informations reliées au produit sont donc naturelles. Il s'agit en fait, de l'hétérogénéité des formats de stockage et des standards d'échange entre les différentes applications. La diversité des outils de travail (logiciels de CAO, FAO, FEA, etc.) est inévitable afin de définir complètement un produit. Les difficultés de communication entre les entreprises ou au sein d'une même entreprise étendue ainsi que l'utilisation de plusieurs logiciels pour la manipulation des données « produits » ont conduit à une complexité de gestion de l'ensemble des informations.

Au cours du cycle de vie d'un produit, et par définition même de l'ingénierie simultanée, il y a apparition de modifications incontournables : des changements qui affectent les composants fabriqués ou achetés, les documents, les processus de fabrication et d'approvisionnement, etc. Ces changements prolongent les délais de production et touchent de 70 à 80% du coût final du produit (Chen et al. 2002), d'où la nécessité de gérer ces modifications afin de raccourcir les délais d'industrialisation et de garantir la cohérence des données « produit ». Avec une telle gestion, on s'assure que tous les intervenants (internes ou externes à l'entreprise) utilisent les mêmes données « produit » en tout temps; en tenant à jour les données relatives à ce produit.

Dans les entreprises, des procédures de gestion des processus de modifications et des documents permettant le déclenchement, l'approbation, le suivi et l'enregistrement des différentes modifications (Engineeing Change Request ECR, Engineering Change Order ECO, Engineeing Change Notofication ECN, Workflow) ont été introduites. Des moyens formels et informels tels que le courrier électronique, le papier, le fax, le téléphone, les notes écrites, etc. sont utilisés. Cependant, l'étude de l'impact et l'étendue des modifications, l'étude des moyens humains et financiers nécessaires ainsi que l'influence sur le déroulement du projet restent nécessaires pour évaluer les délais de réponse et de prise de décision.

Des logiciels de gestion de cycle de vie des produits « Product life Management » (PLM, autrefois désignés Product Data Management PDM et parfois collaborative Product Data Management cPDM) ont été introduits pour aider à la gestion de l'ensemble des données « produit » dans le champ de la conception et de la fabrication mécanique. Ils permettent de gérer l'ensemble des documents relatifs à un produit. Ils maintiennent des liens (souvent dynamiques) entre divers fichiers. En cas de modification d'un détail dans un document, la propagation des changements se fait au niveau des fichiers liés. Ces logiciels sont incapables de capter des informations supportées par des conteneurs plus petits que le fichier (Jacobs 1998). Ils traitent la gestion des documents à un niveau élevé de granularité. Ces logiciels se limitent à la propagation des changements au niveau des fichiers impactés sans pour autant chercher les détails des modifications.

Il est nécessaire d'avoir une granularité au niveau des objets manipulés par les logiciels de gestion de cycle de vie (PLM) plus fine que le fichier (Eustache 2002) afin de garantir une propagation ciblée, efficace et complète de toutes les modifications apportées au produit. Ce travail présentera une solution aux difficultés dans les processus de gestion des changements et propagation des modifications afin de pouvoir déterminer avec précision les entités impactées et pour gérer les liens entre ces entités.

Ce travail présentera un modèle de gestion de propagation des modifications entre documents méthodes. La première section présentera la revue de littérature et résumera les travaux et modèles développés. Dans la deuxième section on présentera les définitions nécessaires pour bâtir l'architecture générale du modèle. La troisième section sera consacrée à l'étude d'un exemple, suivi d'une conclusion.

CHAPITRE 1

PROBLÉMATIQUE

1.1 Introduction

Parmi les documents méthodes, on dénombre les gammes d'usinage des pièces, les dessins de définitions, les programmes d'usinage d'une pièce sur une machine outil à contrôle numérique, les dessins de gabarits de montage, des gammes de préparation d'outils. Il est courant que des changements d'ingénierie surviennent et par conséquent affectent le processus d'usinage de la pièce. Tous les documents qui ont été auparavant préparés avant que le changement ne survienne, doivent être vérifiés. La propagation est généralement faite manuellement ce qui induit des oublis et des erreurs. Par conséquent, des délais de production s'allongent et la productivité décroît. Nous nous proposons de résoudre ce problème.

1.2 Contexte d'étude

La propagation des modifications est étudiée dans le cadre d'un processus d'ingénierie concourante. Un tel processus est caractérisé, entre autres, par une meilleure coordination entre les acteurs d'un projet de conception et de fabrication d'un produit (Chen et al. 2002). L'échange des données entre les diverses applications participants à la définition d'un produit au cours de son cycle de vie est accru. Une autre caractéristique d'un processus d'ingénierie concourante est l'accroissement rapide du nombre de versions produites dans chacun des logiciels utilisés en un temps relativement court, ce qui complique la tâche de vérification manuelle de la bonne propagation des modifications. Sans un outil PLM, il est très probable d'oublier de propager les changements que subissent certains détails (Mokhtar et al. 1998). À l'aide d'un outil PLM, il est parfois inutile de vérifier tous les documents mentionnés comme

« non à jour » puisque les modifications n'ont réellement aucun impact sur ces documents (Jacobs 98, St-Martin 2001). Aussi, en se servant d'un outil PLM, on n'a pas d'information crédible sur les détails à modifier. Cette dynamique intense des échanges de données doit être traitée efficacement par un outil d'assistance automatique de propagation ou au moins par un indicateur fiable d'entités réellement impactées par des changements. Notre modèle s'inscrit dans ce cadre et se veut un outil de l'ingénierie concourante.

À la page suivante, la figure 1 présente un cas typique de gestion des données techniques d'un produit, soit une configuration dans laquelle on retrouve :

a. fichier Part 1 : un fichier de données qui décrit la géométrie du produit;
b. fichier gamme et trajectoires qui inclut la description du processus de fabrication du produit incluant les machines utilisées, les outils et les trajectoires décrites;
c. fichier inspection : décrivant les règles de contrôle du produit;
d. fichier spécifications : un document décrivant les spécifications du produit;
e. fichier Excel : comportant des résultats d'analyses et de tests.

Ces différents fichiers sont liés logiquement. Par conséquent, une modification dans la définition géométrique du produit pourrait nécessiter des changements appropriés dans le reste des documents. Le problème est de déterminer ces changements nécessaires avec précision avant même que l'on effectue la première modification. Ceci est très important afin de mesurer la portée des modifications à apporter à un produit.

1.3 Scénario de propagation des changements

Les ingénieurs de conception utilisent les logiciels de CFAO pour modéliser la géométrie des pièces à produire, effectuer des analyses dynamiques, simuler des processus de fabrication ou d'utilisation du produit, etc.

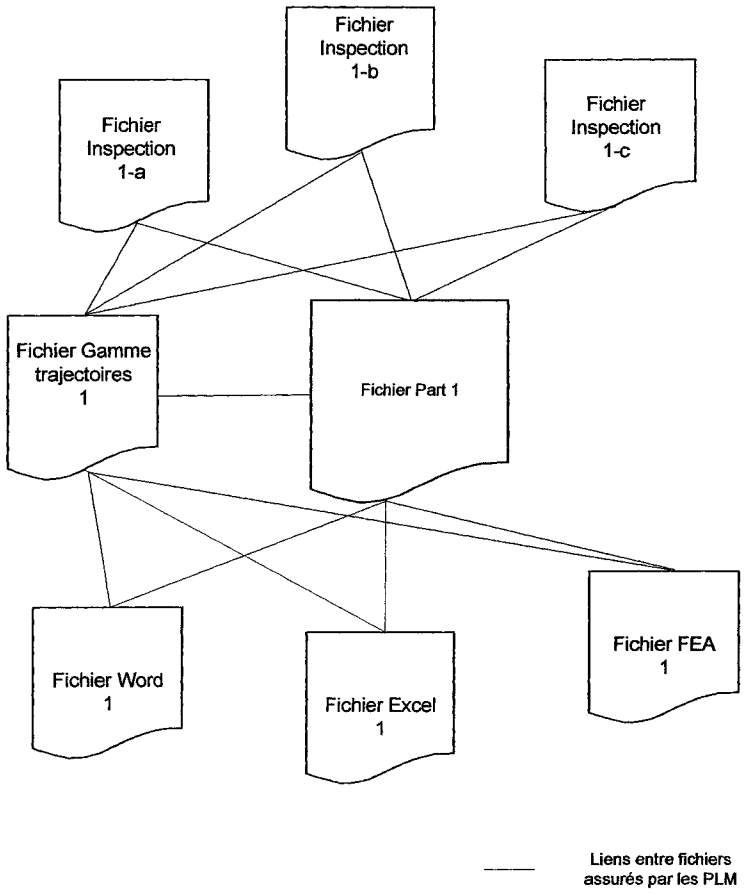

Figure 1 Les liens entre les documents

Les ingénieurs de fabrication utilisent également le système de CFAO pour produire les trajectoires d'outils, mais ils ont recours à d'autres applications pour communiquer avec les techniciens, les sous-traitants, les fournisseurs, etc. Parmi ces applications, on peut citer des logiciels de tabulation Excel, de traitement de texte Word et autres. Au cours de la conception d'un produit, il arrive souvent de modifier la géométrie de la pièce, par exemple, la dimension d'une pièce. Les systèmes de CFAO intégrés permettent de mettre à jour la géométrie des pièces en relation avec la pièce changée ainsi que les tous les processus d'analyse ou de fabrication générés par ce même système de CFAO. Cependant, les ingénieurs de production doivent vérifier manuellement les autres documents produits par les autres applications pour déceler les changements et appliquer

les mises à jour nécessaires. En résumé, un changement est survenu dans un fichier décrivant une pièce, le changement est identifié et les autres documents liés sont identifiés et mis à jour.

Les objectifs à atteindre à la fin de cette étude sont :

a. dégager une méthode capable de propager les changements entre des documents méthodes produits par différentes applications informatiques;
b. élaborer un modèle de propagation de ces changements;
c. élaborer une maquette de démonstration.

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Introduction

Plusieurs chercheurs se sont intéressés à la gestion des changements d'ingénierie. Tous ont prouvé la nécessité de tenir au courant l'ensemble des acteurs d'un projet en ce qui a trait aux modifications apportées de tout genre (Jacobs 1998, Mokhtar et al. 1998, Zhang et al. 1999). Les conséquences économique et technique d'une mauvaise gestion des changements sont importantes. Des pertes de marché, des retards de livraison, des incompatibilités de montage, des non respects des exigences fonctionnelles, des non-conformités des documents méthodes, etc., sont observés chaque fois qu'il n'y a pas eu de suivi des changements faits (Jacobs 1998, El-Bibany et al.1999). Dès la fin des années quatre-vingt, des chercheurs ont publié des travaux touchant à ce problème (Heller et al 1989). Nombre d'entre eux ont proposé des méthodologies à appliquer pour gérer efficacement les changements d'ingénierie. La majorité de ces chercheurs se sont intéressés à la gestion des versions lors d'un processus de conception (Santoyridis et al. 1997, Krishnamurthy et al. 1995, Oussalah et al. 1997). D'autres chercheurs ont démontré que la gestion des versions ne suffisait pour maintenir une cohérence entre les diverses informations de produit. Ils ont proposé des modèles de gestion de la complexité associée à la conception des produits (Jacobs 1998 et Eustache 2002). Des logiciels de gestion des données techniques sont apparus prétendant apporter le soutien nécessaire aux ingénieurs. Au début, ces logiciels ne permettaient que la gestion des versions de fichiers. Par la suite, ils se sont enrichis de nombreuses fonctionnalités telles que la gestion des bases de données liées au produit, la gestion du flux documentaire, la gestion du processus d'approbation de modification d'un document, la génération de nouvelle version, etc. Des chercheurs ont travaillé sur la gestion des changements en tant que processus parmi ceux retenus en ingénierie concourante (Chen et al. 2002).

Cependant, peu de recherches ont été réalisées concernant la propagation des modifications. Certains l'évoquent brièvement dans leurs modèles de gestion de la complexité (Jacobs 1998, Eustache 2002). D'autres ont focalisé leurs travaux sur la gestion de la propagation des modifications au niveau des fichiers au cours de la gestion des versions (Krishnamurthy et al.1995). Des travaux ont été réalisés dans le domaine de la gestion des projets en génie civil ou en conception de logiciels mais ceux-ci ne s'adaptent pas facilement au contexte de la propagation des modifications en génie mécanique (Mokhtar et al 1998, Wang Z. et al. 2001, Wand X. et al. 2001).

Dans cette revue de littérature nous présentons les principales fonctionnalités des logiciels PDM reliés à notre travail. Nous discutons les travaux de chercheurs sur la gestion des versions. Nous traitons en détail les travaux relatifs à la gestion de la complexité. Finalement, une revue des travaux qui ont proposé des concepts d'entité et de liens technologiques sera présentée.

2.2 Les processus de gestion des modifications d'ingénierie

Le processus de gestion des modifications diffère d'une entreprise à une autre. Il est souvent le résultat d'expériences accumulées au fil des ans. Chen et al. (2002) ont traité dans leur article le modèle traditionnel de référence de gestion de changement d'ingénierie. Ce modèle représente le déroulement d'un changement d'ingénierie depuis la phase de proposition, passant par l'approbation et la notification. À cette dernière phase, un document résume les différentes parties approuvées de la proposition de changement, et indique le travail à réaliser ainsi que la date du début de la révision; souvent appelée effectivité et les tâches affectées aux différents intervenants. Il est parfois nécessaire de lancer un nouveau processus de changement d'ingénierie si les processus existants ne sont pas appropriés. Le changement d'ingénierie, en lui-même, est un processus hautement itératif dans lequel les changements sont revus et approuvés. À la phase de déploiement, les différents responsables sont informés et les modifications

sont envoyées aux applications telles que « Enterprise Ressource Planning ERP ». La phase finale est l'archivage, au cours de laquelle les modifications réalisées, les raisons et les justifications, les résultats et les influences sont enregistrés. Des systèmes de gestion des données d'ingénierie et des données techniques dans les processus de production et de développement doivent être capables de définir et d'implémenter un processus de changement, relever et suivre une demande de changement d'ingénierie, circuler les notes de changement d'ingénierie et interfacer le changement à effectuer avec l'application concernée. Il est important de noter qu'un processus de changement d'ingénierie possède les caractéristiques d'un processus d'ingénierie concourante, essentiellement orienté processus, basé sur le projet et centré sur le produit.

D'autre part Maurino (1993) a présenté, dans son chapitre intitulé « la gestion des modifications », l'importance de l'analyse de l'impact d'une modification sur l'ensemble des composants d'un produit. Mais cette analyse dépend beaucoup de la connaissance du produit par les intervenants. La criticité de la modification traduit le degré de l'impact sur les fonctions, compositions ou méthodes de réalisation du produit. On retrouve également les mêmes concepts développés par Jukka et al. (2000) dans leur rapport concernant le processus de gestion de modification.

2.3 Les outils de gestion des données techniques (PLM)

Il n'est pas dans notre intention de faire un inventaire de performances, de fonctions et de capabilités des systèmes PLM. Nous ne traiterons que les articles qui touchent de près la problématique du présent projet.

Jukka et al. (2000) ont tracé l'évolution historique des systèmes PLM. Ils notent la diversité des sigles employés, soient EDM, PIM, TDM, TIM, etc., pour désigner un système de gestion des données. Les principales composantes d'un PLM sont : la voûte, le gestionnaire de flux de travail, le gestionnaire des structures des produits, la

classification des documents, le gestionnaire de modification et de notification des changements.

St-Martin (2001), présente les outils de PLM disponibles. Il signale que le rôle essentiel d'un PLM est la gestion des modifications. Pour les utilitaires de gestion des informations de produits, la gestion des modifications est restreinte à la gestion des versions et des configurations pour les assemblages complexes. L'outil PLM permet de signaler l'utilité de vérifier manuellement l'état d'un document altéré par des modifications apportées à des documents avec lesquels il est en relation grâce aux liens déjà établis dans le PLM. Les liens entre ces documents peuvent être établis manuellement ou automatiquement par le PLM lui-même. L'auteur précise que les outils PLM s'arrêtent au stade de signalement des modifications apportées aux divers documents sans pour autant être capable de dire avec précision la nature, l'étendue, les conséquences et la gravité de ces modifications.

L'organisme « Object Management Group OMG » a lancé, en 2001, un cahier de charges appelé « PDM Enablers » spécifiant les standards d'un système de gestion des données d'un produit. La modélisation a été réalisée à l'aide des diagrammes UML. Un des modules défini est le « PdmChangeManagement ». Dans les spécifications de ce module on ne trouve pas d'indications sur la granularité des informations à traiter. Notons que Flater et al.(2001) présentent des batteries de tests de conformité des PLM. La granularité des entités manipulées n'est pas considérée parmi ces tests.

2.4 La gestion des versions

L'étude de la propagation des modifications entre documents méthodes ne peut être détachée de la gestion des versions d'un produit. Miles et al (2000) signalent qu'au cours de la conception d'un produit, après la phase de conception conceptuelle et la définition des contraintes interfaciales minimales entre composantes, chaque concepteur

ou équipe de travail, au sens de l'ingénierie concourante, produit des versions de composantes suivant la maturité des solutions. D'après McClatchey et al. (1997), une version des données peut être historique ou de configuration. Une version historique est l'enregistrement des informations d'un produit à un certain temps. Quant à la version de configuration, elle représente l'enregistrement de ces informations suivant des configurations de montage, de solutions techniques adoptées, de caractéristiques techniques, etc. Santoyridis et al. (1997) présentent les différents états d'une version. Ils énumèrent l'état transitoire, l'état de travail et l'état relâché. Plusieurs articles (Miles et al 2000, Santoyridis et al. 1997) ont traité le problème de gestion des versions dans un environnement d'ingénierie concourante tout en omettant de parler de la propagation des modifications au niveau des détails et de préciser l'impact de ces changements, parfois appelés mineurs, sur la création de nouvelles versions. Oussalah et al. (1997) ont introduit un modèle de gestion des versions qui repose sur le mélange de deux aspects macroscopiques et microscopiques des composants des produits dans une base de données orientée objets. Toutefois, le problème persiste lorsqu'on introduit des informations modélisées par des systèmes de CFAO, où l'accès aux informations microscopiques est limité, sinon absent pour un gestionnaire de bases de données. Krishnamurthy et al. (1995) ont proposé un modèle de gestion des données d'un produit reposant sur la gestion des versions des entités. Ce modèle présente plusieurs outils d'ajout des versions, de changement de statut, etc. Un outil de comparaison de version a été présenté. Cependant, cette solution reste intimement liée au système de DAO utilisé comme support de maquette.

2.5 La gestion des changements dans un contexte de gestion de la complexité

Jacobs (1998) se propose de résoudre la problématique de la complexité de gestion de conception des produits. Une complexité se manifeste à travers le nombre important de pièces d'un assemblage, la définition géométrique compliquée de chaque pièce et les différentes fonctions de chaque pièce. La majorité des logiciels de CAO offrent des

outils pour la modélisation des pièces mais ne peuvent supporter les relations complexes entre les diverses composantes. Ces systèmes isolent complètement les informations de conception d'une pièce. Les modèles pour chaque pièce et les modèles pour différentes disciplines; mécanique, électrique, etc. sont développés et maintenus dans des fichiers et formats séparés. Pour gérer la synergie des relations existant entre les diverses représentations, le concepteur devra traquer lui-même ces relations. Les systèmes de PLM tentent de résoudre ce problème en créant et classifiant les liens entre les diverses représentations, mais ils ne réussissent pas à représenter l'information de synergie qui est nécessaire à ces relations. L'organisation en fichiers des bases de données des logiciels PLM et la classification manuelle forcent les concepteurs à gérer les diverses représentations à un niveau élevé de granularité. L'auteur propose une structure d'organisation et de gestion des informations de conception afin d'assister le concepteur et d'automatiser la gestion de la complexité. Il propose ainsi une structure permettant d'automatiser les tâches d'organisation et de gestion des modifications des diverses composantes. Il s'inspire des modélisations et des conceptions orientées objets pour présenter des objets d'agrégation, de relations et de version. Ces objets sont munis des méthodes nécessaires pour suivre les modifications. Ce travail reste toujours au sein même d'une seule application de conception et ne peut pas intégrer d'autres applications nécessaires au développement des produits.

Le deuxième travail qui s'est intéressé à la gestion de la complexité de la définition des données des produits a été réalisé par Eustache (Eustache 2002). Il propose un modèle de Graphe Hiérarchique d'Entités et des Contraintes (GHEC) figure 2. Il vise à résoudre le problème de la gestion de la complexité dans un processus de conception de produits mécaniques tout en maintenant la cohérence de la définition du produit. Le GHEC est une composition du produit en entités liées par des contraintes. L'entité est un « objet ou une idée qui existe en soi, en dehors de tout contexte. L'idée correspond à l'état abstrait d'une entité et l'objet à l'état concret ». L'entité est un objet au sens de l'Orienté Objet, par conséquent, elle possède des attributs et des méthodes. Parmi les attributs on trouve

l'adresse et le type du contenant (type d'application et fichier de stockage). L'objet entité peut être décomposable ou agrégé. Ainsi, on parle des sous entités. Quant aux contraintes, elles sont définies comme étant des « degrés de liberté que l'on peut exprimer sous forme de contraintes ». De la même manière que les entités, les contraintes seront représentées par des objets en association avec les entités. Une contrainte possède des attributs et des méthodes, tel qu'un identifiant unique, un type, une fonction à satisfaire, des droits d'accès, etc. Une contrainte peut être raffinée; décomposée en sous contraintes reliant des sous entités. L'auteur présente une classification des contraintes selon le lien entre les entités. Il distingue 4 types de contrainte. La contrainte de dépendance indique une relation de dépendance générale entre les entités, d'un niveau sémantique faible et dont l'automatisation lors d'une opération d'évaluation est impossible. L'opérateur aura la responsabilité de contrôler ces contraintes. Le deuxième type de contrainte est relationnel, c'est l'ajout des expressions relationnelles de type égalité, infériorité, supériorité, etc. Le troisième type des contraintes représente les contraintes booléennes. Enfin, les contraintes liées aux métiers sont appelées contraintes règles. En effet, ce graphe représente une hiérarchisation des entités et des contraintes ainsi que différents niveaux d'automatisation de validation de ces contraintes. Suite à une modification ou suppression d'une entité ou d'une contrainte, le GHEC doit être mis à jour. Au cours de cette opération, l'invalidation d'une entité ou d'une contrainte provoque une propagation d'invalidation qui peut invalider tout le GHEC. L'auteur propose alors d'inclure un état intermédiaire où les entités ne sont pas invalides mais en attente d'une vérification. Lorsqu'une entité est modifiée et si les contraintes liées à elle ne sont pas modifiées, alors il est inutile de propager la modification. Concernant le traitement des modifications, l'auteur ne propose dans son travail que le suivi de la validité des entités et des contraintes, donc il ne traite pas la façon de modifier la définition des entités et des contraintes.

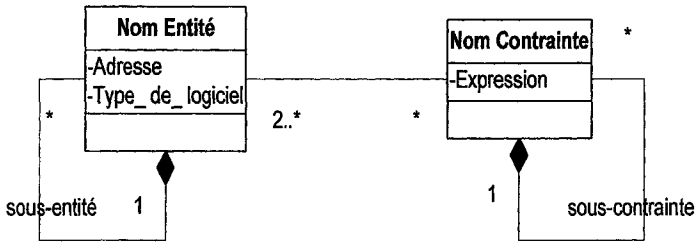

Figure 2 Graphe hiérarchique des contraintes

(Adapté d' Eustache, 2002)

2.6 Les définitions de l'entité et des liens

Un modèle d'information pour gérer l'ensemble des données d'un projet de génie civil a été développé par Mokhtar et al. (1998). Après une large revue bibliographique, les auteurs concluent que la gestion des changements de conception ne dispose pas d'une solution standard. Le problème consiste à informer les différents intervenants, de toute discipline participante au projet de développement, des modifications faites par d'autres disciplines et affectant directement leurs données. Les auteurs résument les techniques utilisées par les professionnels pour résoudre ce problème : des réunions régulières où les différents intervenants essaient de trouver dans les documents présentés des changements qui touchent à leur travail. Le concepteur qui effectue un changement en informe les autres. Un autre moyen consiste en l'utilisation d'une *check-list*. Certains utilisent des comparateurs de modèles fournis par certaines compagnies de CAO. Tous ces moyens sont jugés inefficaces, surtout pour un projet de grande envergure. Les auteurs proposent alors un modèle d'information ayant, entre autres, les caractéristiques suivantes :

a. être capable de gérer les changements de conception en aidant à propager les modifications vers les disciplines affectées uniquement;
b. suivre l'historique des changements;
c. donner la possibilité de planifier les futures modifications.

Le modèle présenté est une base de données d'un projet central composé de deux parties. La première est une base de données de toutes les composantes de conception nécessaires à tout le projet. Notons que les composantes de conception «building components » sont les données nécessaires à la description du projet (un édifice en l'occurrence). Ces données sont suffisamment détaillées pour produire les documents de détail. La deuxième partie est la base de données de gestion. Elle contient l'information nécessaire pour gérer toutes les fonctions du modèle. Le concept est d'assigner la tâche de propagation des modifications aux composantes du projet (building components) elles-mêmes. Chaque composante est responsable de trouver les disciplines affectées par la modification dans l'un de ses attributs. La composante envoie un message aux autres disciplines clarifiant la modification et la manière dont elle affecte chaque discipline. De cette façon, on s'assure d'éliminer la possibilité qu'une personne initie un changement et oublie d'en informer les autres, ou croit qu'elle n'aura pas d'influence sur les autres disciplines ou qu'une autre discipline n'en sera pas affectée.

Ces composantes doivent être équipées des « liens de connaissances : linking knowledge » qui permettent d'identifier les disciplines affectées par un changement et la façon dont elles seront affectées. Les modèles utiliseront des règles (relations, contraintes) de deux types pour capturer ces connaissances. Le premier type est le « préétabli » le deuxième type est le « dynamique ». Les règles préétablies sont dictées au moment de la configuration du modèle au début du projet. Les règles dynamiques sont créées à la volée « on the fly » au cours de la préparation des dessins de détail et lors de l'utilisation du modèle. Ces dernières sont capturées par le module de captation des règles. Ce module fait partie de la manipulation des données et est interfacé avec les concepteurs. Lorsqu'un changement survient, la composante vérifie les règles préétablies et dynamiques stockées. Les messages sont automatiquement envoyés aux autres composantes affectées. Ces messages sont sans ambiguïté et définissent clairement les attributs à modifier. Un suivi d'historique des changements est également possible. Les auteurs se sont intéressés aussi à la planification des changements et ont

remarqué qu'un changement peut donner lieu à plusieurs scénarios. Ils ont présenté un modèle dont l'idée est de bâtir un dépôt unique de toutes les composantes qui provoquent des changements de conception. Ils utilisent des données textuelles, plutôt que des dessins, comme principal moyen pour sauver et communiquer les informations de conception. Mais ce concept ne s'adapte pas aux documents méthodes dans un projet de conception mécanique.

Giguère et al. (2002) ont proposé un modèle qui vise à automatiser des tâches de modélisation répétitives. Les auteurs proposent l'utilisation d'une « caractéristique contextuelle ». Ils définissent alors un lien technologique par une relation de dépendance entre deux éléments d'information. Le lien de dérivation quant à lui est une relation unidirectionnelle entre une caractéristique cible et une caractéristique source. Le lien est utilisé pour dériver une caractéristique cible à partir d'une caractéristique source. Ce même lien jouera un rôle important pour la propagation des modifications.

Les mêmes laboratoires de recherche (LIPPS et Sherbrooke), dans une publication de Macabies et al. (2001) définissent un lien technologique comme étant des relations entre des éléments d'information technologique. Ces liens peuvent être de deux types. Le premier est le lien fondé sur la duplication de l'information. Dans ce cas, chaque modèle (fichier) est considéré indépendant des autres. Un mécanisme de propagation des changements est nécessaire pour garder le même contenu dans chaque élément. Le deuxième type est celui fondé sur l'unicité de l'information. Dans ce dernier cas, on n'a pas besoin d'un mécanisme de propagation des changements parce que l'information est contenue dans un seul contenant extérieur. La localisation d'une référence dans le premier type est nécessaire pour contrôler l'information. Cette référence peut être dans le fichier d'assemblage ou dans les fichiers modèles. On parle alors de référence à sources multiples ou à source unique. Les auteurs ont choisi de piloter les informations à partir de l'assemblage, par la suite, pour toute modification apportée (ajout, suppression, création etc.); la propagation des modifications sera pilotée à partir de ces références

dans l'assemblage. Enfin, d'après les auteurs, les entités doivent répondre à quatre critères : avoir une représentation géométrique normalisée, avoir un identifiant unique, avoir des spécifications technologiques et avoir des données de propagation. En réalité, ce travail était destiné à gérer les changements dans un assemblage modélisé avec le logiciel Catia V4.

Des développements importants ont été réalisés par le groupe de recherche du département « Sciences Informatiques et Ingénierie » de l'Institut Indien de Technologie afin de formaliser la définition des entités et l'évolution d'un graphe représentant un projet de conception. Ce travail publié par Srinath et al.(2000) affirme que la gestion de configuration (CM configuration management) fournit les outils nécessaires pour gérer les versions et notamment traquer les changements dans les projets de développement de logiciels ainsi que dans les projets utilisant des logiciels de CAO. Cependant, ces techniques conventionnelles considèrent un projet comme un ensemble d'éléments de différents types dont la configuration doit être gérée. Ils gèrent uniquement les versions des éléments individuellement, telles que les fichiers. Le projet n'est pas considéré en tant qu'entité sémantique dont le changement de version de chaque élément a un sens différent dans le cadre de ce projet. Et plus généralement, un changement dans un élément induit des changements dans d'autres éléments. Il est nécessaire de considérer la sémantique des changements lors de la gestion des versions. Les auteurs se proposent de représenter le projet par un graphe direct et d'introduire la gestion des versions, en se basant sur la sémantique, aux nœuds de ce graphe. Les auteurs utilisent le concept de l'URA (Unified Representation of an Artifact : Représentation Unifiée d'un Artefact). Un artefact est n'importe quelle entité logique d'intérêt. Cette entité peut être de différents types et de granularités diverses. Par la suite, un URA est une représentation de cet artefact. Cette URA est composée de 3 éléments essentiels : un extracteur qui extrait l'entité du système d'information dont elle fait partie. Un conteneur qui contient le contexte et d'autres informations relatives à l'artefact, un mécanisme d'authentification pour s'assurer de l'intégrité des informations extraites. En plus de ces

trois composantes, cet URA est liée à d'autres URAs reflétant ainsi les relations entre les artefacts.

Ainsi, un projet est l'ensemble des URAs liées et définissant un graphe. Un lien entre deux artefacts peut être un lien faible ou fort. Une relation forte entre deux artefacts indique que la sémantique de la cible est dépendante de la source. Les auteurs proposent un modèle générique pour la gestion des versions basée sur la sémantique. Dans ce cas un graphe des artefacts des URAs définit le projet et évolue en fonction des changements apportés. Ce graphe peut être composé de sous graphes représentant des sous systèmes. Les nœuds de ce graphe sont les artefacts URAs. Pour chaque sous-graphe il existe un unique nœud pivot. Un nœud pivot représente l'artefact correspondant au sous-système représenté par le sous-graphe. Le nœud est dit pivot dans le sens ou le changement de version du sous graphe correspond à un changement de version du nœud pivot. Il est essentiel que le nœud pivot soit relié à tous les nœuds du sous graphe. Le graphe évolue suivant la modification des attributs d'un nœud ou le changement d'état des liens d'un nœud. Les auteurs ont précisé les 12 règles qui gouvernent l'évolution du graphe. Celles-ci se basent sur les liens (forts ou faibles) pour créer des versions ou des pseudos versions. La propagation est contrôlée par un mécanisme qui met un nœud, voulant évoluer vers une nouvelle version, dans un état transitoire jusqu'à l'arrêt de la vague de propagation. Les auteurs mentionnent que l'utilisation de ce modèle n'est pas conseillée pour des changements mineurs au cours du développement d'un projet de conception mécanique. Dans un projet de conception mécanique, les attributs de version sont ceux qui gouvernent la conception de l'artefact. Par exemple, la conception d'un élément élastique sous contraint a comme attributs le module d'Young, la conductivité thermique, la ductilité, le coefficient de Poisson et la couleur. Seul le module d'Young parmi ces attributs gouverne la conception de l'élément. Chaque changement dans le module d'Young provoquera la révision de la définition de l'élément.

2.7 Conclusion

La revue bibliographique a permis de dresser l'état de la recherche en ce qui a trait à la problématique soulevée. La majorité des recherches a été orientée vers la gestion des versions. La propagation des modifications a été traitée au niveau des fichiers. La granularité des entités manipulées n'a pas été traitée convenablement surtout en ce qui concerne les documents de conception et de fabrication mécanique. Néanmoins, les concepts d'entités et de liens entre entités pourront servir de base pour bâtir un nouveau modèle de gestion de la propagation des modifications. Cependant, des précisions et des ajustements devront être apportés à ces modèles afin de répondre aux besoins énoncés.

CHAPITRE 3

MODÈLE DE GESTION DE LA PROPAGATION DES MODIFICATIONS

3.1 Introduction

La problématique nous a permis de démontrer la nécessité de gérer la propagation des modifications à un niveau de granularité de données plus fine qu'un fichier. Cela aura pour effet de mieux cibler les documents impactés par des changements dans un assemblage complexe au cours de son cycle de vie. En effet, que ce soit après une demande de changement (Engineering Change Request ECR) (figure 3) au moment de l'étude de l'impact ou au moment de propager la modification, il est nécessaire de disposer d'un outil fiable. Ce dernier indiquera avec précision les documents impactés suite à la modification d'un détail dans un document méthode. La revue de littérature nous a permis de conclure sur les limites des modèles proposés. Ces derniers ont traité la gestion du processus de propagation des modifications entre documents, au niveau du fichier, sans plus de détails.

Pour être cohérent, un modèle de gestion de la propagation des modifications doit définir en détail les entités manipulées et la manière de propager les modifications. En effet, avec leurs relations et leurs attributs, les entités constituent une composante essentielle d'un modèle conceptuel de données. Dans ce chapitre nous définissons tout d'abord la terminologie associée a l'élaboration de notre modèle. Par la suite nous présentons l'architecture du module de gestion des modifications. Nous détaillerons l'interaction entre les documents à traiter et les modules composant le modèle. Il est essentiel de noter que notre modèle servira aussi bien pour l'analyse de l'impact d'une modification que pour la propagation des changements entre documents méthodes.

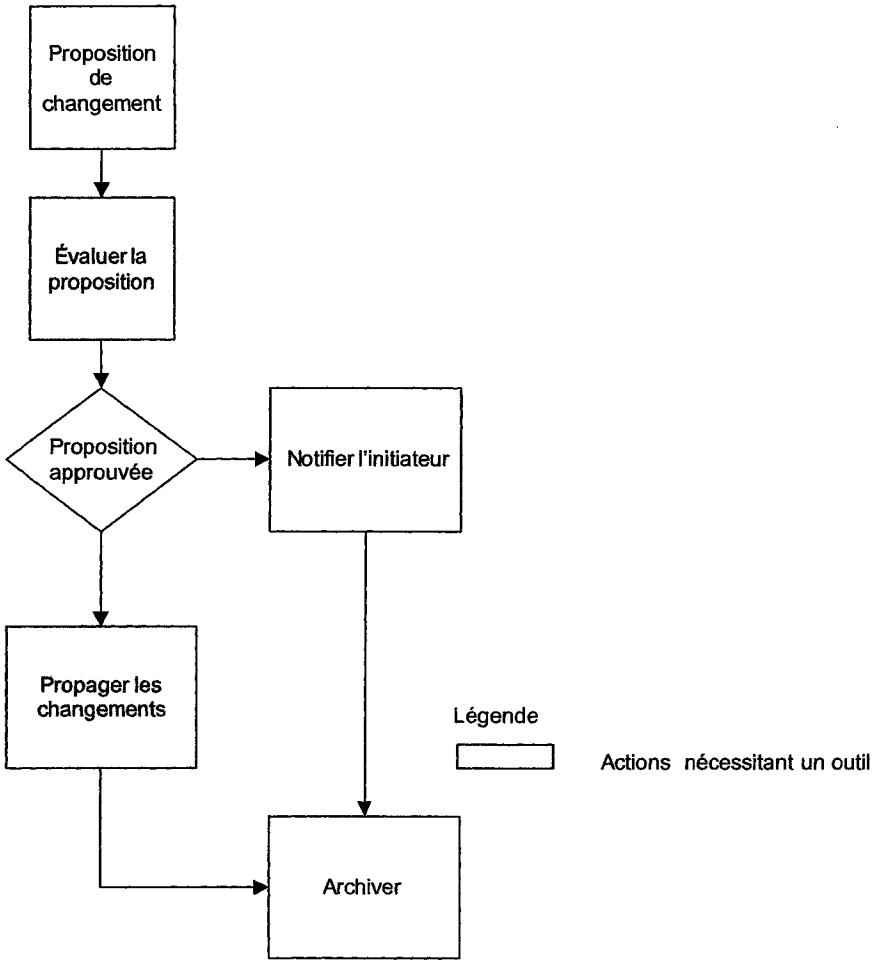

Figure 3 Besoin d'un outil fiable dans un processus de gestion de modifications

3.2 Définitions

3.2.1 Entité et granularité des données

Nous désignons par entité, d'après Le grand dictionnaire (2003), un ensemble de données géométriques et/ou technologiques stockées représentant un concept et qui peut être traité comme une unité indépendante ou un membre d'une catégorie particulière. Quant à la granularité de cette entité nous la définissons par le degré de fragmentation

(mémoire, disque, fichier, champ, etc.) en unités plus petites, dans un but de gestion. Par exemple, on dira que la granularité d'un champ est plus fine que celle d'un fichier pris globalement. Nous présentons à la section 3.3 le choix des entités à manipuler dans le présent projet.

3.2.2 État d'un document vs version

Dans cette recherche, nous désignons par « état » d'un document la configuration des données telles qu'elles sont enregistrées sur le support magnétique désigné ou la base de données. On fera abstraction de la nature de la version (temporaire, release, etc.). La solution technique retenue, dans un document à un certain état, pourra ne pas être parfaite ou finale. Dans ce cas, l'analyse d'impact d'une modification est recherchée. L'état d'un document est donc intimement lié à son enregistrement sur un support magnétique ou vers une base de données. Tant que les modifications apportées à un document n'ont pas été enregistrées, elles ne seront pas propagées.

3.2.3 Liens et associations

D'après Giguère et al.(2002), les liens entre les documents traités résultent des liens technologiques entre les différentes entités composant ces documents. Ces liens résultent aussi des relations logiques entre documents relatifs à un produit. Ces liens sont de nature à maintenir une associativité entre diverses entités (Macabies et al 2001). Contrairement aux autres travaux de recherche de Srinath et al. (2000), qui ont classifié les liens en faibles et forts, nous n'envisageons qu'un seul type de lien. Si deux entités sont liées, lorsqu'une d'entre elles est modifiée, l'autre devra l'être nécessairement. C'est une relation d'association bidirectionnelle. La construction des liens sera discutée plus loin dans le paragraphe sur les mécanismes de gestion des liens. Nous notons par $L(E_i, E_j)$ un lien entre les entités E_i et E_j. Si deux entités sont liées, leurs conteneurs le

sont nécessairement. Nous notons dans ce cas $L(D_i, D_j)$ le lien entre les deux documents. En effet, l'ensemble des liens entre entités est un sous-ensemble des liens entre documents. Ainsi, les liens entre documents gérés par les systèmes PLM traditionnels seront utilisés par notre modèle.

3.2.4 Modification

Maurino (1993) a définit la modification par « une évolution apportée à un document ou à un dossier valide ». Dans cette recherche, une modification est définie comme étant un changement apporté à l'un des attributs d'une entité. Cette modification aura pour effet de changer ses caractéristiques géométriques et/ou technologiques. Certainement que le la modification est liée à l'état du document. Une modification des attributs ne sera effective que lorsqu'elle est accompagnée d'un changement d'état du document. Un changement de position d'attachement de l'entité est considéré, par exemple, comme une modification des attributs de l'entité. Par contre, le changement de la couleur d'une entité ne sera pas considéré. En effet, la modification d'un tel attribut n'aura aucun effet sur les entités associées. Notons aussi que la suppression ou la création d'une entité sont des modifications apportées au document qui seront prises en compte. Le mécanisme de gestion (section 3.5) des liens se chargera de traiter adéquatement la propagation de telles modifications.

3.2.5 Notification et propagation des modifications

Lorsqu'une entité est modifiée, celle-ci est responsable de notifier ce changement aux autres entités associées. La notification est l'expression par laquelle une entité exprime aux autres entités un ou plusieurs changements dans la valeur de ses attributs. Une entité qui reçoit une notification de changement d'un attribut d'une entité associée devra changer la valeur de ses attributs en conséquence suivant des règles préétablies qui

seront discutées plus loin. Ces dernières représentent des contraintes de liaison (Eustache et al. 2001).

3.3 Choix des entités à manipuler

Les systèmes de CFAO d'aujourd'hui reposent sur la modélisation par caractéristiques (Giguère 2002). Un modèle produit est perçu comme la composition d'un certain nombre de caractéristiques agencées ensemble. La définition et le type des caractéristiques ne cessent de s'enrichir (Shah et al. 1994, Msaaf 2002, Giguère 2002.). Malgré la diversité de ces définitions nous adoptons les caractéristiques comme « entités » à suivre pendant le processus de propagation des modifications entre documents méthodes.

Pour les documents à traiter, autres que les fichiers d'un système de CFAO, la définition des entités suivra celle présentée dans un système de CFAO. En d'autres termes, nous considérons que les entités (*features*) définies dans un système de CFAO imposent la définition et la granularité des autres entités associées et appartenant à des documents qui ne sont pas générés par un système de CFAO. Nous commencerons par définir les entités dans un modèle de produit dans un logiciel de CFAO. Par la suite, la granularité des entités associées en sera fonction. Cette granularité est donc variable et est source de flexibilité. Le fait de choisir les caractéristiques d'un modèle de CFAO comme référence de granularité pour les autres entités associées découle du fait que ce document pilote l'ensemble des documents. Nous rappelons que notre sujet traite de la propagation des changements entre documents méthodes. Ces derniers font référence principalement aux documents définissant la géométrie du produit.

3.4 Modèle proposé

3.4.1 Description de l'architecture générale

Les processus de gestion des modifications (Chen et al. 2002, Jukka et al. 2000.) s'étalent de la demande d'un changement ECR jusqu'à l'approbation (Engineering Change Order ECO), en passant par une phase d'analyse d'impact. Nous proposons des outils capables, en même temps, d'analyser l'impact d'une modification et de propager cette modification si approuvée. Notre modèle sera perçu comme un module d'un système de PLM. Ce dernier sera capable de bien gérer le flux d'information entre intervenants (Workflow), de garder un historique des changements et d'offrir ainsi un cadre général de gestion des informations d'un produit. La gestion de la propagation des modifications entre entités, telles que définies plus haut, sera une des fonctions d'un système de gestion des données techniques. Ainsi, l'intégration dans un tel système sera assurée. Nous utiliserons les fonctionnalités offertes par un logiciel de PLM telles que la construction des liens entre les conteneurs, ainsi que la gestion des versions et des historiques des modifications réalisées.

En se référant à des travaux effectués par d'autres chercheurs (Mokhtar et al. 1998, Chen et al.2002, Soh et al. 2000) et à l'émergence des techniques de la programmation orientée objet dans les divers logiciels mis en concours lors d'un développement de produit (Favre et al. 2001, Sanlaville et al. 2001), la meilleure façon de détecter les modifications dans un document est de rendre les entités « parlantes ». Une fois modifiée et validée, l'entité doit informer les autres applications des changements qu'elle a subit. Ensuite, chaque entité a l'obligation de changer ses attributs en fonction des changements apportés aux autres entités avec lesquelles elle est en relation directe (fig. 4). Il est par conséquent nécessaire de tenir une liste de liens entre les différentes entités associées. Cette liste de liens est établie par les différents intervenants du projet

de développement du produit. La différence entre notre approche et celle de Mokhtar et al. est qu'en génie civil, un projet est gouverné par un certain nombre de paramètres globaux contenus dans la même base de données (Mokhtar et al. 1998), alors que la structure des données dans les systèmes de CFAO interdit un tel stockage.

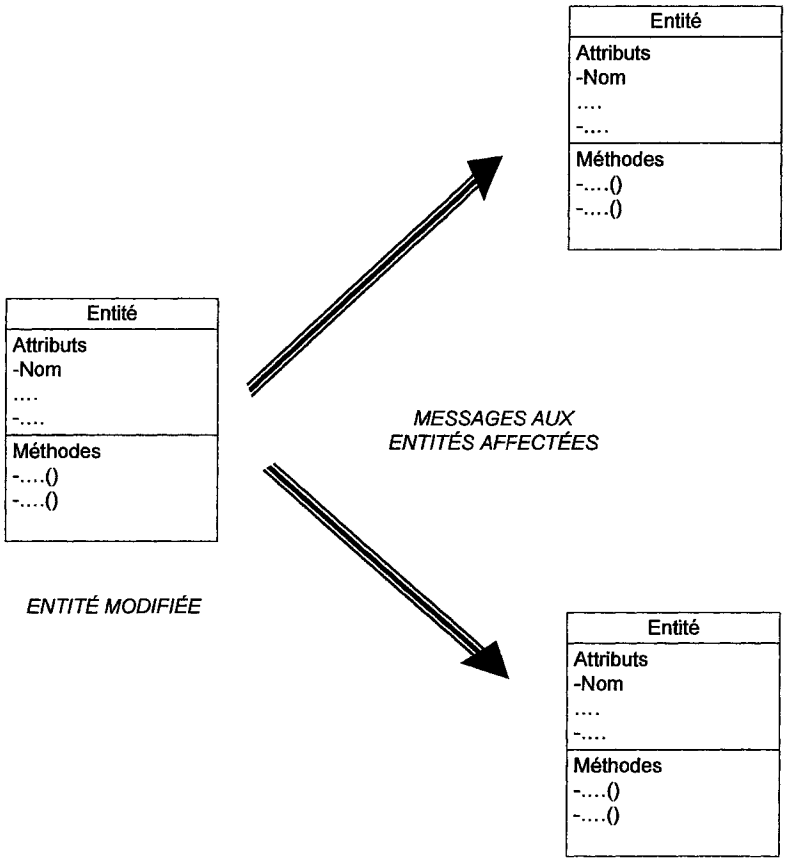

Figure 4 Entité parlante

La propagation des modifications entre documents repose essentiellement sur les liens établis entre les entités. Un système de PLM du marché utilise les liens entre les documents pour marquer les documents impactés par un changement. Les systèmes de CFAO utilisent eux aussi des liens entre les divers fichiers d'un produit (Catia V5, Solidworks, MasterCam et autres). Si un de ces fichiers est modifié, le système exige la mise à jour des autres fichiers référencés. Nous pouvons aussi citer l'exemple du

système d'exploitation Windows qui garde les liens entre les fichiers en utilisant les liens OLE (Object Linking and Embedding).

La figure 5 présente l'architecture générale du modèle. Il est composé d'un module central appelé « Gestion Modifications » qui gère l'ensemble des liens entre les entités. Il est à noter, dès le départ, que ce module est indépendant de tout logiciel pour le développement du produit tels que Catia, Solidworks, Word, Excel, etc. Notre concept repose sur l'accessibilité des informations, contenues dans les fichiers produits par divers logiciels, tout en restant à l'extérieur de ces applications. Le module central « Gestion Modifications » pourra s'intégrer à un système de PLM tel que présenté à la figure 5. Les diverses applications informatiques utilisées au cours du cycle de vie du produit sont connectées au module central. Celui-ci reçoit les notifications de changement provenant des entités modifiées. Il renvoi les ordres de changement aux entités impactées.

L'architecture du modèle est composée d'un module central (GestionModifications), d'un module lié au système de CFAO (ModuleDocumentCFAO) et un autre module (ModuleDocument) lié à un document générique. Les prochains paragraphes détailleront les différents modules et la prochaine section présentera l'implantation des divers modules sur un cas réel.

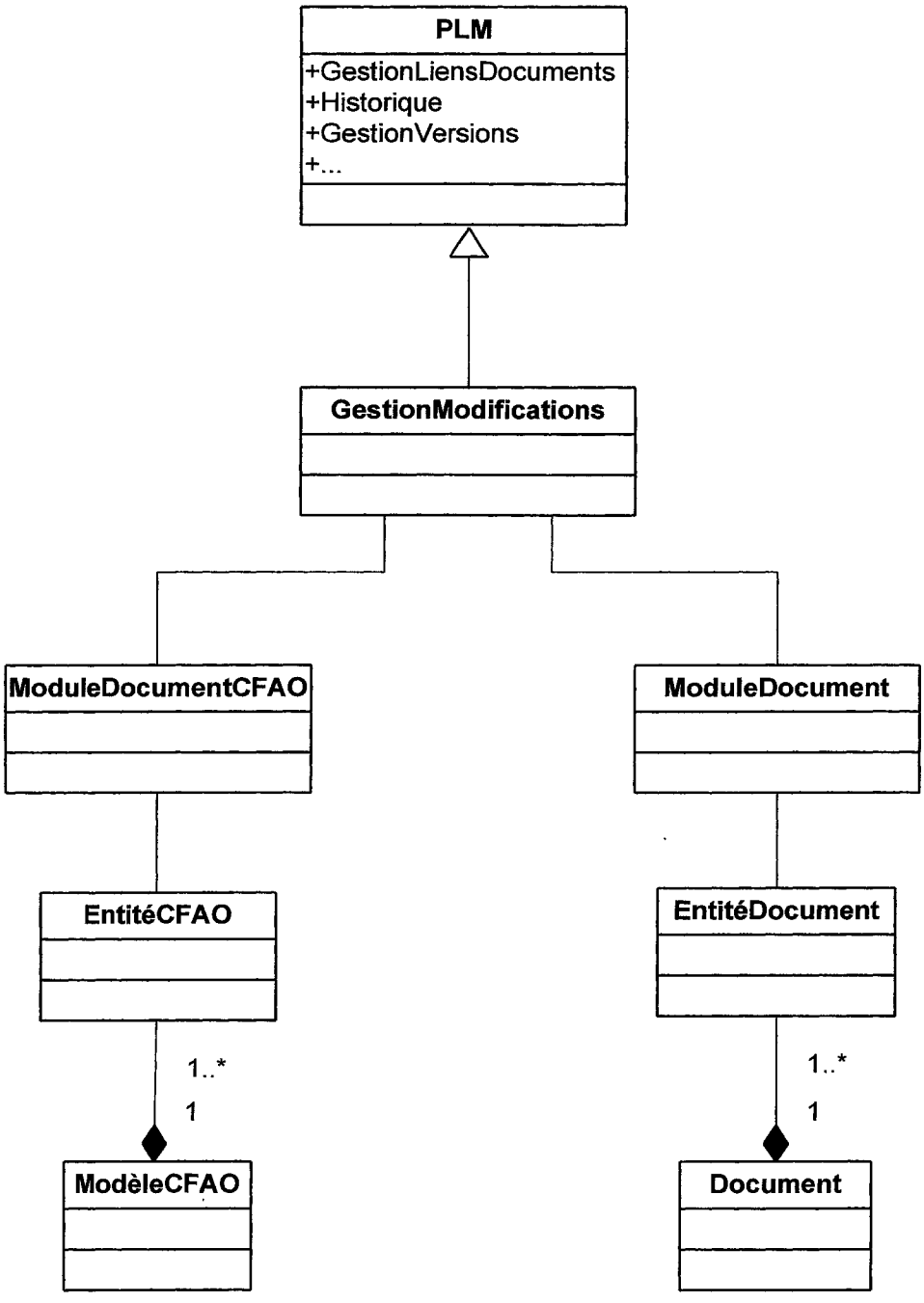

Figure 5 Architecture générale du modèle

3.4.2 Description des modules

3.4.2.1 Le module central

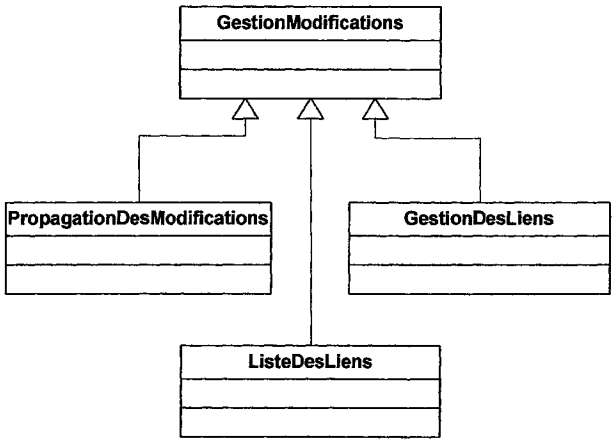

Figure 6 Module central

Le module central (figure 6) : tel que présenté plus haut, un lien entre entités est une relation d'association $L(E_1, E_2)$. C'est grâce à cette relation que nous pourrons propager efficacement les modifications. Afin de construire et de tenir à jour une liste de ces liens, le module central offre trois services. Le premier est la gestion des liens; il s'agit d'un ensemble de fonctionnalités telles que l'ajout, la suppression et la modification des liens. Le deuxième service est la liste des liens. Le module central offre un troisième service. Il s'agit de la propagation des modifications entre les documents. Nous détaillerons plus loin les méthodes utilisées.

3.4.2.2 Les modules documents

Chaque document à traiter est relié au module central via un module document. Les changements que subit une entité sont envoyés au module central qui les traite en

fonction des liens. Il est utile de rappeler que chaque document à inclure dans le présent travail doit être généré par un logiciel permettant l'accès aux entités. Malheureusement, plusieurs logiciels ne permettent pas encore l'accès à de telles informations ou le permettent mais avec un niveau de granularité grossier.

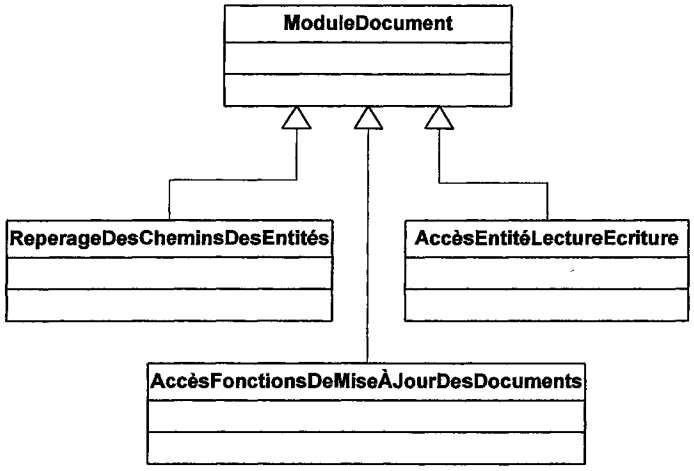

Figure 7 Module document

La figure 7 présente la composition de chaque module. Ceux-ci seront intégré à chacune des applications informatiques participantes. Les différentes fonctionnalités offertes permettent l'accès aux attributs des entités composant un document. Cette accessibilité est de type écriture/lecture. Cela permettra une meilleure interopérabilité entre les diverses applications mises en jeu. Finalement, il est essentiel de pouvoir accéder aux fonctions traditionnelles de mise à jour, de sauvegarde et d'ouverture d'autres documents.

3.5 Mécanisme de gestion des liens

Les entités contenues dans les documents à manipuler peuvent déjà être créées ou seront créées ultérieurement. La difficulté est de trouver un moyen pour mettre en place ces liens entre les entités des documents à traiter. D'après la revue de bibliographie,

Mokhtar et al ont proposé deux manières de construire ces liens (Mokhtar et al. 1998). Les liens préétablis et les liens établis en cours de développement. Nous reprenons cette manière de bâtir les liens. C'est ainsi que la construction des liens pourra se faire de deux manières. Lorsque les entités à associer existent déjà, les liens entre elles peuvent être construits statiquement. Lorsque les entités ou une partie d'entre elles à associer sont à créer en cours de développement du produit, les liens peuvent être créés dynamiquement. La liste des liens sera mise à jour en fonction de la création des liens entre les entités. Ainsi, ce module central présente la flexibilité nécessaire à tout processus d'ingénierie concourante. Le mécanisme de gestion des liens est un gestionnaire de base de données relationnelle de liens.

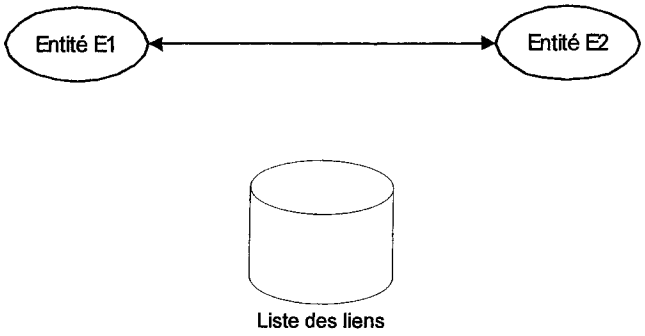

Figure 8 Liens et listes de liens

Un lien $L(E_i, E_j)$ figure 8 sera constitué par les chemins absolus de l'emplacement de stockage de chaque entité. Ces chemins permettront de repérer les entités dans leurs documents respectifs.

3.6 Propagation des modifications

La propagation des modifications et la détermination précise des documents impactés rapidement et efficacement sont le cœur même de notre travail. Grâce aux liens qui ont été construits, il est possible de repérer les entités impactées par un changement. En

effet, dès qu'une entité est modifiée, au sens de la modification déjà définie dans le présent chapitre, elle en informe le module central. Celui-ci, par l'intermédiaire des liens établis, déterminera les entités impactées et par la suite, les documents impactés. Ces derniers sont marqués « non à jour ».

Concernant l'automaticité de l'exécution des changements, certains travaux (Eustache et al. 2001) ont cité trois niveaux : manuel, semi-automatique et automatique. Pour être en harmonie avec la gestion des modifications proposée dans les autres concepts de PLM (Jukka et al. 2000, Maurino 1993) nous souscrirons à la même philosophie. Nous ajoutons un troisième argument au lien L(E_i, E_j, Automaticité) qui indiquera le degré d'automaticité choisi. Dans le cas où la propagation est automatique i.e. le système l'effectuera sans l'intervention de l'utilisateur, des règles d'interaction préétablies entre les attributs des entités associées seront utilisées. Ainsi on ajoutera un quatrième argument au lien mentionnant les règles à utiliser au besoin L(E_i, E_j, Automaticité, Règles)

3.7 Conclusion

Pour une meilleure propagation des modifications, notre approche repose sur la mise en place de liens entre entités qui s'apparentent aux caractéristiques d'un système de CFAO. Une exploitation judicieuse de ces liens permettra une analyse d'impact et une propagation contrôlée et ciblée des modifications. Ainsi, seuls les documents réellement touchés par la modification d'une partie d'un document associé, seront traités. Nous pallions ainsi à une des limitations des systèmes PLM.

La prochaine section sera consacrée à la validation de ce nouveau modèle. La mise en œuvre des différents modules, la construction des liens et la propagation des modifications seront traitées. Des illustrations seront également présentées.

CHAPITRE 4

LE PROTOTYPE DE VALIDATION

4.1 Introduction

Dans le but de montrer la faisabilité du concept proposé, nous présentons dans ce chapitre la description d'un prototype de validation. Cette maquette est loin d'être un produit commercial fini. Il s'agit de montrer la possibilité de surcharger des caractéristiques d'un système commercial de CFAO et de propager la modification de certains attributs vers des entités en dehors de ce système. L'accent sera mis sur l'accès à l'information utile à la propagation des modifications entre plusieurs applications informatiques tout en restant indépendant de ces systèmes. Nous exploiterons les différentes interfaces de programmation (API) des applications informatiques participants au développement du prototype.

Nous présenterons une traduction du modèle présenté à la figure 5 en diagrammes UML. Ces derniers montreront l'interaction entre les divers documents. Le choix de ces derniers et de leurs générateurs sera également discuté. Des cas d'application seront présentés.

4.2 Traduction de l'architecture générale du modèle en diagramme UML

4.2.1 Module central

La figure 9 ci-dessous détaille les objets du module central représentés par des boîtes. Les attributs et les méthodes de chaque objet sont énumérésg. Les liens entre les rectangles représentent des liens au sens de la programmation orientée objet.

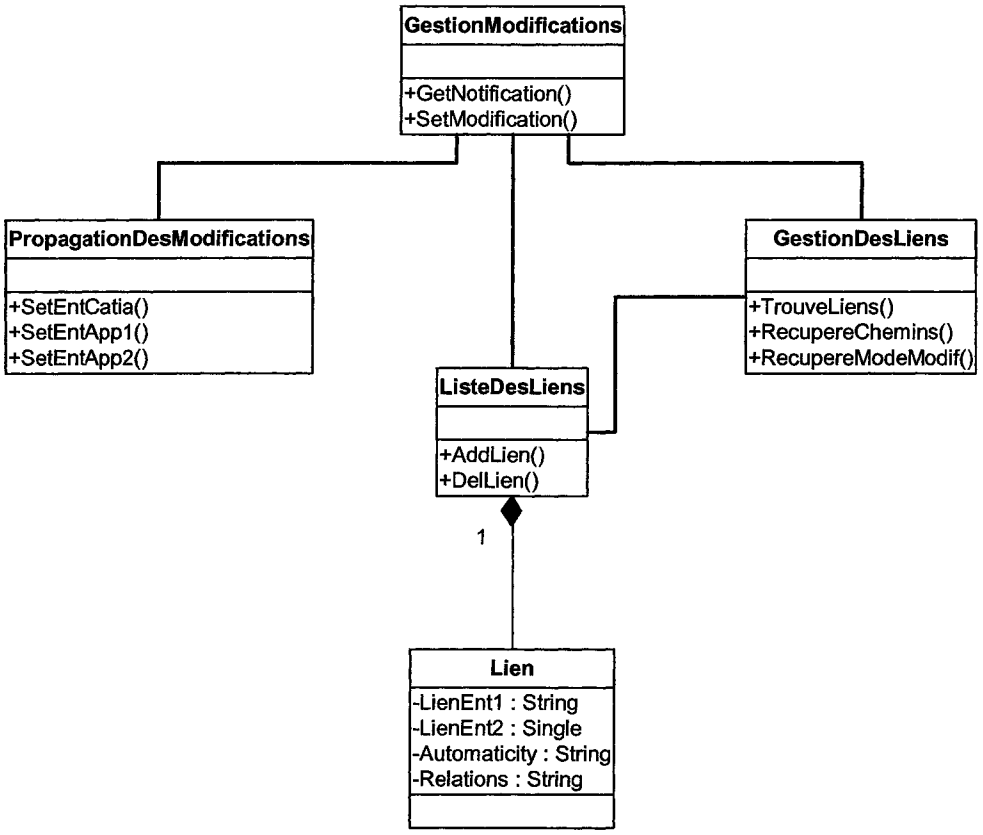

Figure 9 Modèle UML du module central

Lorsque la notification de changement est reçue par le module central, celui-ci demande au module « GestionDesLiens » de déterminer les entités associées en récupérant leurs liens. Ces derniers sont stockés dans une base de données « ListeDesLiens ». Un lien renferme les informations nécessaires concernant les entités liées ainsi que l'automaticité du mode de changement (automatique, manuel, semi-automatique). Notons que cette base contient également les règles établies entre les entités. La propagation des modifications est réalisée par les méthodes de la classe « PropagationDesModifications » après l'autorisation émise par le module central via la méthode « SetModification ». Ces méthodes sont destinées à modifier les entités

impactées par un changement. Les informations nécessaires (application, document, entité, attribut et sa valeur) ont déjà été collectées.

4.2.2 Module document

Le module document figure 10 renferme des méthodes spécifiques à chaque type d'application intervenant (CATIA, Word, Excel, etc.). L'entité à modifier doit être contenue dans un document généré par une application dont l'interface de programmation (API) est disponible. Les fonctions recherchées dans l'interface de programmation de l'application sont :

a. ouverture du document en écriture/lecture;
b. repérage précis de l'entité;
c. modification et mise à jour du document.

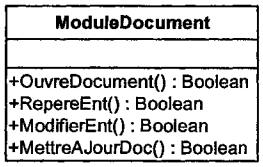

Figure 10 Modèle UML du module document

Nous présentons dans les paragraphes suivants les applications qui ont été développées avec les logiciels CATIA V5, Word et Excel.

4.3 Choix des systèmes et des documents

Pour la mise en œuvre de ces digrammes UML, nous avons retenu, en concertation avec nos partenaires, les applications suivantes :

a. CATIA V5 : ce logiciel est installé au LIPPS ainsi que son API (CAA) et l'environnement de développement (RADE). Notons que CATIA V5 est un logiciel de CFAO intégré; il permet de concevoir des pièces, de mener des analyses dynamiques et de préparer les processus de fabrication et de simulation;
b. Microsoft Word et Excel : ces deux applications sont d'une utilisation très répandue pour préparer les documents méthodes utilisés par les ingénieurs de production. Notamment pour générer et maintenir la nomenclature des pièces (Bill Of Material BOM).

Le travail d'ingénieur dans les bureaux d'études ou de méthodes consiste à modéliser des mécanismes et à préparer les gammes d'usinage appropriées. Par conséquent, pour illustrer notre propos, nous avons choisi comme document à traiter une pièce modélisée avec CATIA V5 et comportant un certain nombre d'usinages à réaliser. Quelques trajectoires d'usinage ont été réalisées. Les documents Word et Excel présentent les documents méthodes. Soit l'ensemble des documents suivant :

a. CATPart : fichier modèle CATIA de la pièce;
b. CATProcess : fichier de processus d'usinage de la pièce;
c. fichier Excel de la gamme d'usinage;
d. rapport d'inspection;
e. fichier de la gamme d'inspection.

4.4 Développements réalisés

4.4.1 Développement du module central

L'environnement de développement « .NET » a été utilisé pour la programmation du module central. Ceci qui correspond aux développements suivants figure 11 :

a. la visualisation de chaque document et de ses entités dans un arbre;
b. le traitement des messages de modification émis par les applications (CATIA V5, Word et Excel);
c. l'analyse de l'impact d'un changement survenu dans un document sur les autres documents;
d. la simulation de la propagation des changements;
e. l'envoi des ordres de modifications;
f. l'historique des réceptions et des notifications.

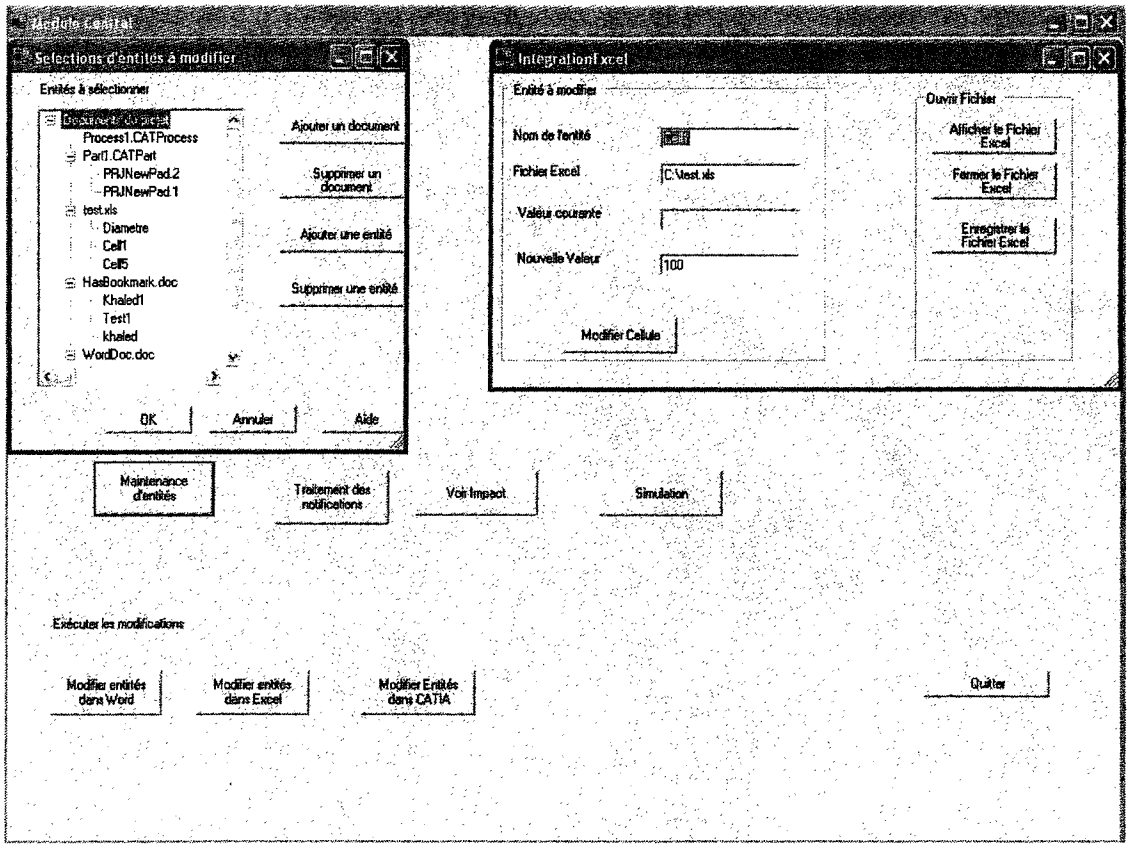

Figure 11 Interface de la maquette

4.4.2 Surcharge des entités dans CATIA

Nous avons créé une nouvelle caractéristique dans CATIA V5 nommée « PRJNewPad » figure 12. Cette caracteristique est dérivé d'un catalogue fourni par Dassault Systèmes, ce qui confère à ce nouveau feature des comportements originaux s'intégrant facilement dans le logiciel CATIA V5.

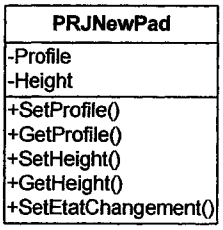

Figure 12 Feature PRJNewPad

Ce feature permet l'extrusion simple d'un profil (sketch) dans une direction perpendiculaire au plan du profil à une hauteur donnée. La démarche de création de ce feature est présentée en annexe. Notons que cette création nécessite beaucoup de développement et des connaissances approfondies de l'API de CATIA V5.

La méthode qui a été ajoutée « SetEtatChangement » permet d'envoyer les messages de notifications au module central qui est en dehors du système de CFAO. Ces notifications peuvent être des notifications de création ou de modification. Ce feature peut être modifié suite à une requête du module central.

4.4.3 Module document (autre que CATIA V5)

Nous avons développé deux autres modules pour les applications Word et Excel. Ces modules, communément appelés « intégrations » dans les systèmes PLM, permettent

d'effectuer des changements dans les documents de type Word ou Excel. Les changements sont effectués sans nécessiter l'ouverture du document et de chercher l'entité à modifier. Ceci permet de minimiser le temps d'exécution de la propagation des modifications.

Une base de données « Microsoft Access » a été utilisée pour stocker les documents, les entités contenues dans chaque document et les liens entre ces entités. La maquette réalisée peut s'intégrer à des systèmes commerciaux PLM ou PDM pour mieux exploiter les fonctions de gestion d'historique des changements et de gestion des versions.

4.5 Exemple d'application

4.5.1 Scénario

Nous nous plaçons à l'étape où l'ordre de modification a été autorisé (fig. 3). Cet ordre peut porter sur n'importe quelle entité stockée dans la base de données. En effet, les liens construits entre les entités sont bidirectionnels; par conséquent notre modèle est en mesure de propager la modification à partir de n'importe quel document du produit traité. Cependant, la formulation de l'ordre de changement doit respecter certaines contraintes. Ces dernières sont d'ordre syntaxique et de clarté de l'entité visée par cet ordre. Le module central reçoit cet ordre et localise l'entité à modifier en premier lieu. Cette entité sera considérée comme l'entité qui déclenche le processus des modifications. Ainsi, le document d'où la modification sera lancée est marqué. La figure 13 présente l'arbre des documents associés à un produit. Par souci de clarté, seules les entités contenues dans le document DI ont été montrées. La figure 14 montre la propagation des modifications à partir de l'entité à l'origine du changement. En effet l'entité EI-1, contenue dans le document DI, étant liées aux entités EII-1 et EII-1, contenues dans le document DII ainsi que les entités EIII-1 et EIII-2, contenues dans le

document DIII. Donc lorsque l'entité EI-1 est modifiée, les entités EII-1, EII-2, EIII-1 et EIII-2 seront impactées par cette modification.

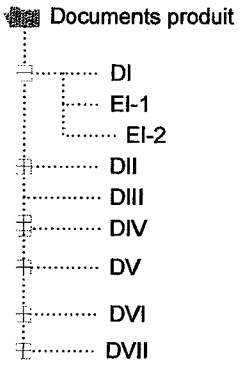

Figure 13 Arbre des documents et des entités

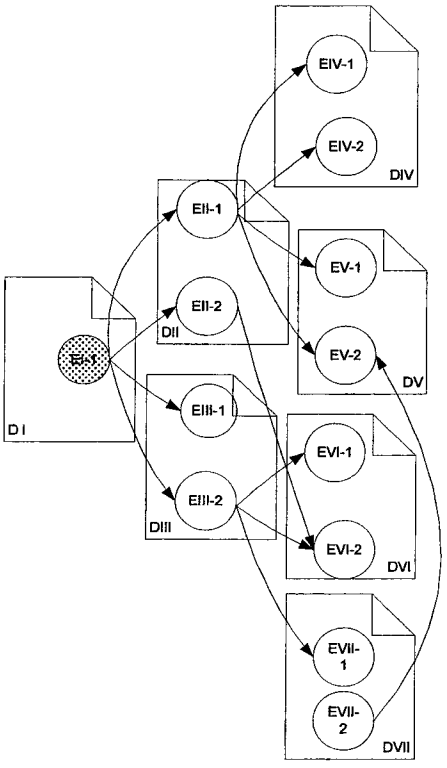

Figure 14 Propagation des changements

4.5.2 Étude de l'impact de la modification

Le module central consulte la liste des liens et détermine toutes les entités associées (figure 15). À partir de la liste des entités associées impactées par la modification de la première entité, le module central établit pour chacune d'elle la liste des entités impactées et ainsi de suite.

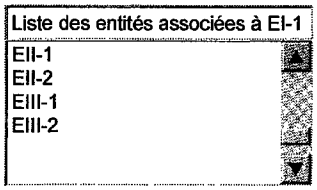

Figure 15 Liste des entités associées

Les liens établis entre l'entité EI-1 et les entités touchées contiennent l'information sur l'automaticité de la modification des ces dernières. D'autre part, le lien contient l'information relative aux règles établies entre les deux entités.

Le module central est capable de construire la liste des liens pour chacune des entités associées à l'entité qui a déclenché les modifications. Ainsi, on aura le résultat de la figure 16.

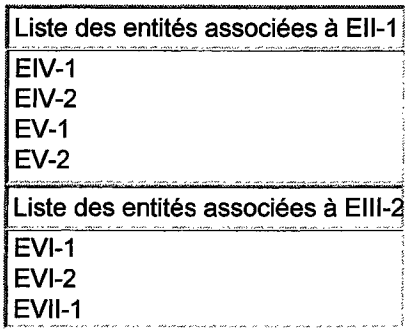

Figure 16 Entités impactées

4.6 Conclusion

La maquette réalisée a prouvé l'efficacité du modèle mis en place. En effet, on a pu démontrer qu'il est possible d'extraire des informations extrêmement importantes pour la gestion des changements encapsulés dans des systèmes de CFAO réputés d'être très fermés et inaccessibles. La propagation des modifications est faite au travers des différentes applications utilisées en se basant sur des concepts forts simples.

CHAPITRE 5

DISCUSSION

5.1 Limitations du modèle

Le modèle de gestion de propagation des modifications proposé dans ce travail de recherche nécessite l'accès à toutes les informations d'une entité encapsulée dans le modèle du produit. Pour des besoins d'échange entre systèmes de CFAO, des formats de stockage standard sont utilisés tels que STEP, IGES, STL. Une fois importé dans un système de CFAO ces modèles ne permettent plus l'accès à des entites granulaires mais plutôt à une seule entité qui répresente en fait la totalité du modèle. Notre modèle repose plutôt sur la décomposisition d'un document en entités, de petites tailles, afin de maîtriser la propagation des modifications, ce qui represente une limitation à ce modèle.

5.2 Efficacité du modèle vis-à-vis de la propagation des modifications

L'efficacité du modèle est d'autant meilleure que les liens ne représentent pas des boucles. Une boucle est, par exemple, lorsqu'une première entité qui subit une modification impacte une deuxième entité (un lien existe entre elles) et que cette deuxième impacte une troisième. Cette dernière est liée à la première. De telles situations doivent être séparées et analysées avant de propager la modification. La maquette a l'avantage de montrer ces boucles lorsqu'elles existent.

5.3 Création des liens

La principale difficulté est la création des liens entre les entités. C'est une opération qui reste manuelle et qui dépend beaucoup de ce que les ingénieurs veulent suivre et analyser au cours d'un processus de gestion des modifications. Cette tâche ne pourra pas

être automatisée étant donné l'utilisation d'applications indépendantes « déconnectées ». À priori, il n'existe aucun lien entre un document Word et un fichier CATIA sauf si on établit un lien d'une manière explicite.

5.4 Traitement d'un nombre élevé de liens

Certes, le modèle proposé ajoute un degré supérieur de complexité aux systèmes PLM du marché. En fait, dans un projet de construction d'un avion d'affaires, le nombre de documents est estimé à 150,000. Établir des liens entre tous ces documents est en lui même un travail fastidieux. Que dire d'ajouter des liens entre des entités contenues dans ces documents ? C'est une complexité de plus que l'on franchit mais en même temps c'est une nouvelle étape importante de la maîtrise de la gestion des propagations des modifications.

CONCLUSION

Nous avons présenté un modèle de gestion de la propagation des modifications. Ce modèle est original en ce qu'il exploite des idées utilisées dans d'autres champs de l'ingénierie des systèmes informatiques et l'applique à des systèmes de CFAO réputés être rigides et clos. En effet, la principale problématique des applications informatiques, utilisées pour générer les documents méthodes, est que l'information reste encapsulée dans ces systèmes. Tous les événements de modifications restent ignorés par les autres applications. Cette notification de changement est la clé pour étudier l'impact qu'elle aura sur l'ensemble des documents participant au cycle de vie d'un produit.

Ce modèle a l'avantage d'apporter une aide précieuse à l'ingénieur afin de mieux mesurer l'impact d'une modification sur chaque document avec précision et rapidité. Cela qui permettra de gagner du temps et d'éviter des erreurs. La maquette réalisée en prouve la faisabilité malgré des difficultés de programmation reliées à la disponibilité et à l'ouverture des fonctions nécessaires dans les API des systèmes de CFAO.

L'inconvénient majeur, pour réaliser commercialement ce modèle, est la nécessité de surcharger les diverses caractéristiques d'un système de CFAO et de créer des modules pour chaque application participante utilisée (couramment appelé intégration). Trouver un mécanisme totalement intégré et natif à chaque système de CFAO comblera cet inconvénient. Un tel mécanisme consistera en une méthode de notification de changement dans la classe parent de toutes les classes servant à l'instanciation des objets géométriques. En effet ce mécanisme existe déjà dans les systèmes de CFAO, ce qui permet d'ailleurs, par exemple, la mise à jour automatique d'un assemblage lorsqu'une pièce est modifiée.

ANNEXE 1

Surcharge des caractéristiques dans CATIA

Telle que présenté au chapitre 3, nous avons identifié une entité à une caractéristique dans un système de CFAO. Cette caractéristique doit être capable de communiquer avec les autres entités externes. Les features standards du système de CFAO CATIA V5 n'offrent pas cette possibilité. Nous étions amenés à « surcharger » ces caractéristiques afin d'ajouter une nouvelle méthode. Cette méthode permettra d'annoncer les modifications apportées à un feature. Nous présentons dans cette section les étapes utiles pour réaliser une telle surcharge.

L'utilisation de l'environnement de développement « Visual Studio 6» configuré avec les ajouts « add-ins » de Dassault Systèmes RADE « Rapid Development Environment » a permis de développer notre application. L'exploitation des classes et méthodes de l'API de CATIA V5 a conduit à développer le projet. Notons que CAA (Component Architecture Application) utilise la technologie de programmation COM (Component Object Model de Microsoft). La principale caractéristique de cette manière de programmer est la distinction entre interfaces et implémentation. Le client utilise toujours les interfaces pour manipuler les objets. La figure 17 présente l'architecture du « framework » développé pour ajouter le « SmartFeature » dans CATIA. Il s'agit d'une simple extrusion « Pad » d'un profil (sketch) dans une direction perpendiculaire au plan de ce profil et à une hauteur donnée. Nous employons les conventions d'appellation fixée par Dassault Systèmes, ainsi PRJ est l'abréviation de PROJET.

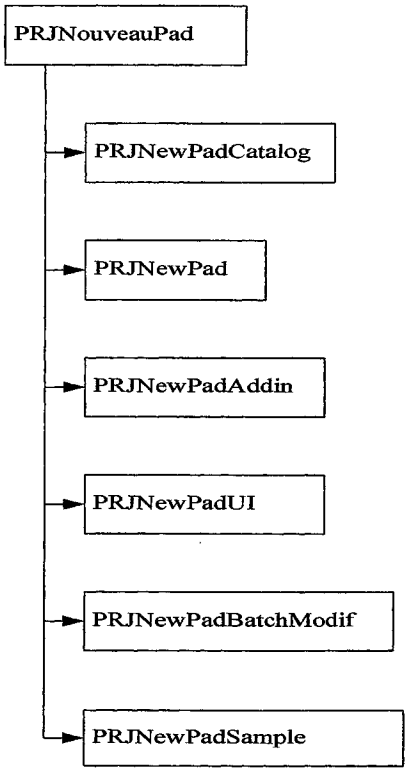

Figure 17 Le « framework » et ses modules

La création d'un nouveau feature débute par la création d'un catalogue. Celui-ci indiquera les attributs de la « StartUp », père de la future caractéristique. Le catalogue renseigne également sur les pères de cette StartUp, il s'agit dans notre cas d'une dérivation à partir d'un catalogue fourni par Dassault Systèmes appelé « MechMod ». Ce catalogue contient la définition du feature « MechanicalFormFeature » que nous choisissons comme père de notre feature. Ce dernier est couramment appelé « late type » et portera le nom de « PRJNewPad ». La StartUp « PRJNewPadStartUP » est créee dans un catalogue utilisateur nommé « PRJNewPadCatalog.CATfct » avec les attributs qui serviront à sa création. Les attributs de cette StartUp sont le profil et la hauteur (figure 18).

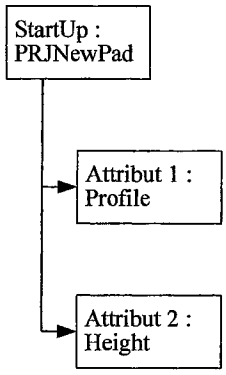

Figure 18 La StartUp

Une fois la StartUp crée dans un catalogue, on pourra l'utiliser pour instancier des features. Pour pouvoir s'insérer dans l'arbre de spécification, notre caractéristique doit être agrégée à un feature valide. Dans le présent cas il s'agit du feature « MechanicalPart ». Le module « PRJNewPad » du « framework » est composé d'interfaces et d'extensions d'objets ayant pour but l'instanciation et la construction du nouveau feature. On remarquera la méthode « SetEtatChangement » ajouté à l'implémentation de la caractéristique. Cette fonction sera responsable d'envoyer les informations de modification à l'extérieur de CATIA. Cette méthode n'est pas publique, elle est accessible uniquement via les autres méthodes de l'interface « PRJINewPad ». Lorsqu'il y a un changement qui est apporté à la caractéristique par l'intermédiaire de la méthode « SetAttribut », cette dernière fait appel à « SetEtatChangement ». Rappelons que ce processus est totalement transparent à l'utilisateur. La figure 19 détaille la composition du module « PRJNewPad ». Elle montre les différentes étapes pour instancier et construire le nouveau Pad. Seules les interfaces sont montrées sur cette figure.

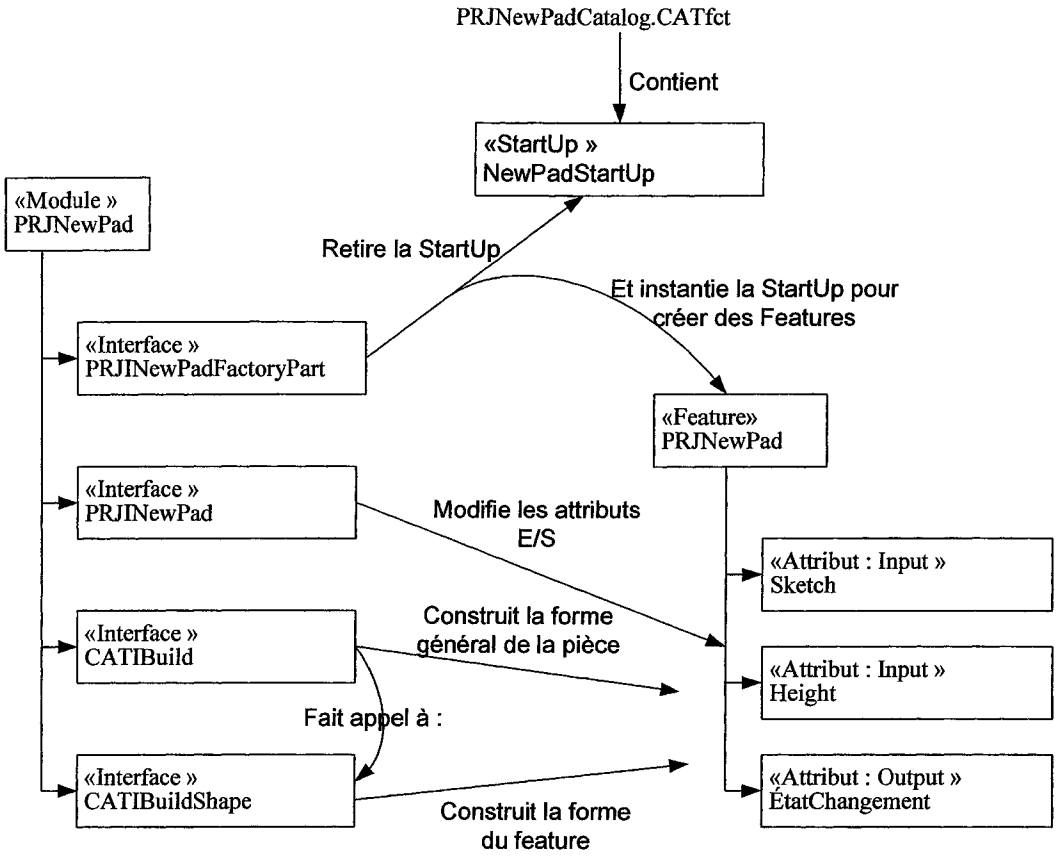

Figure 19 Module PRJNewPad

L'implémentation de ces interfaces par des objets est montrée à la figure 20. Elle détaille l'emplacement de la future feature et les extensions utilisées pour sa construction. Nous établissons la correspondance avec l'interface utilisateur dans CATIA V5 (figure 21).

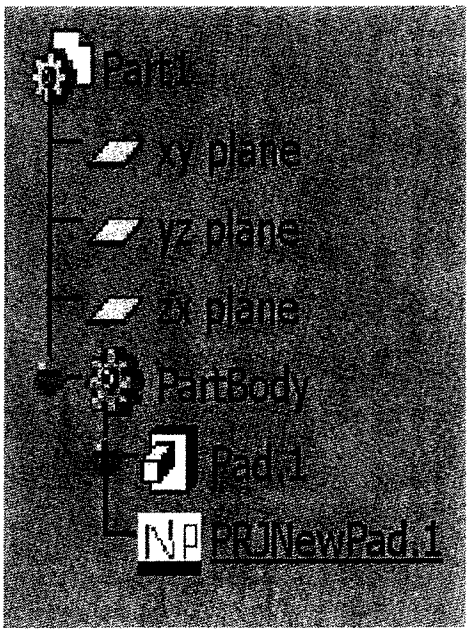

Figure 20 Nouveau feature

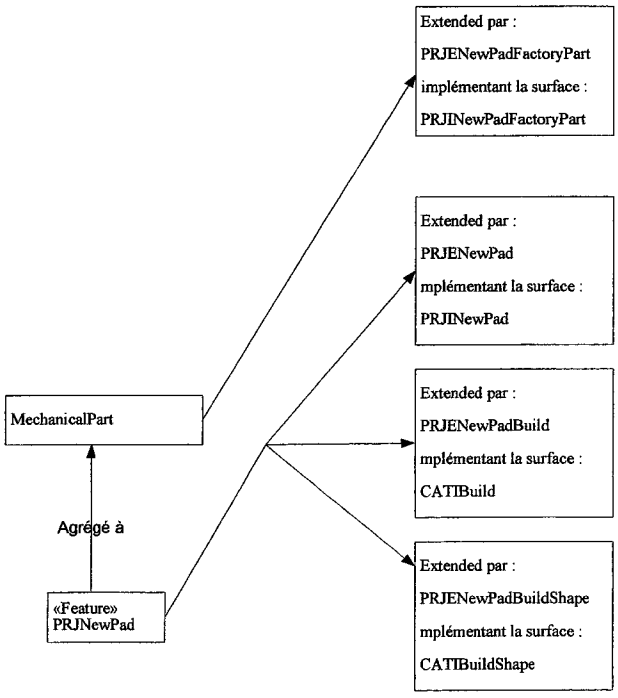

Figure 21 Implémentation des interfaces du module PRJNewPad

Afin de pouvoir visualiser le nouveau feature dans l'arbre de création, nous lui ajoutons une icôn par l'intermédiaire de l'interface « CATIIcon ». L'édition du feature est possible grâce à l'interface « CATIEdit ». La figure 22 montre l'intégration de ces deux interfaces et les résultats obtenus. Notons que lorsque l'utilisateur change une valeur des entrées, soit le profil ou la hauteur et qu'il valide par l'activation de la touche « OK », une notification de modification est envoyée au module central, à l'extérieur de CATIA V5.

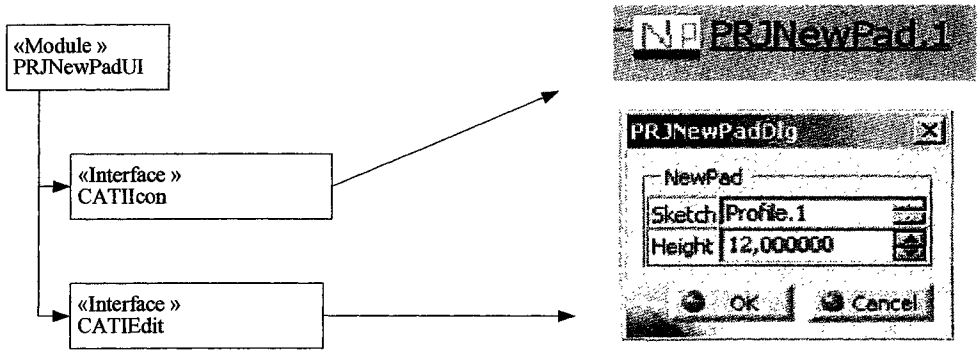

Figure 22 Interface utilisateur

Lorsqu'il s'agit de la création d'un nouveau feature on utilise une commande sur une barre d'outils. Cette dernière est intégré dans le « workbench » de « Part » dans CATIA grâce à l'implémentation de l'interface « CATIPrtCfgAddin ». La barre d'outil « NewPad » créee contient les icônes nécessaires à la création du nouveau feature (figure 23).

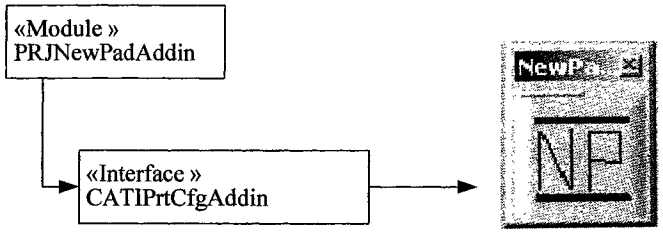

Figure 23 Barres d'outils de création d'un nouveau feature

La modification de ce feature par l'extérieur est réalisée à l'aide du module « PRJNewPadBatchModif ». L'ordre de changement est reçu du module central. Celui-ci fournit le chemin complet de l'entité à modifier avec la nouvelle valeur de l'attribut. L'opération de modification est faite en batch et le document est mise à jour (figure 24, 25).

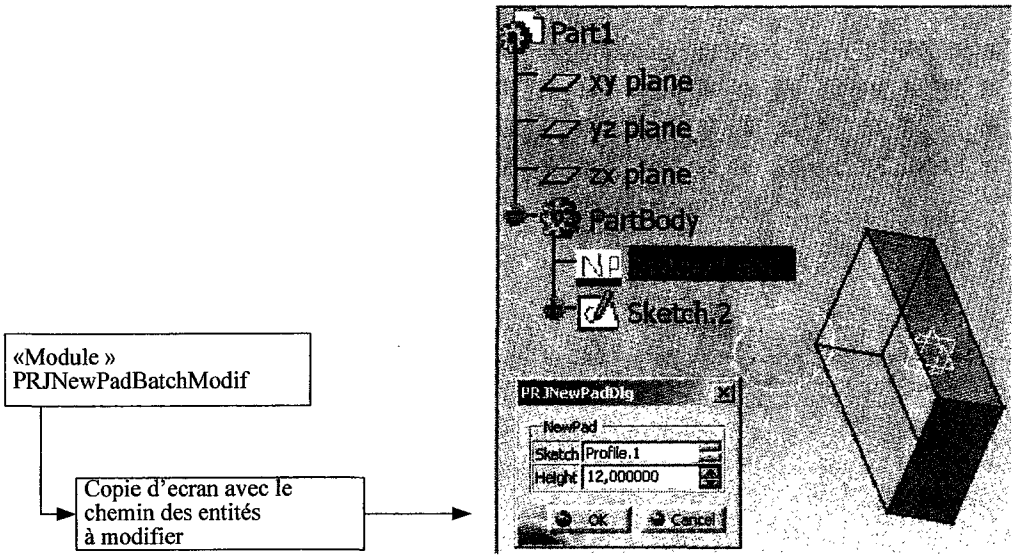

Figure 24 Avant modification

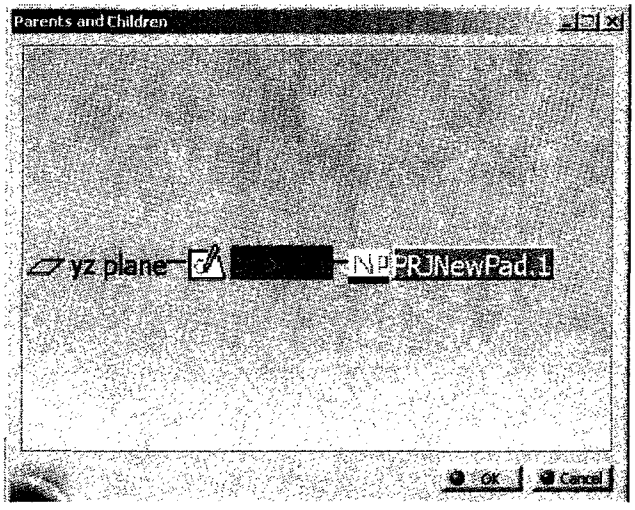

Figure 25 Relations parents/enfants avant modification

Les figures 26 et 27 montrent le changement du profil pour le feature créee. Auparavant il était un rectangle et été changé par un cercle.

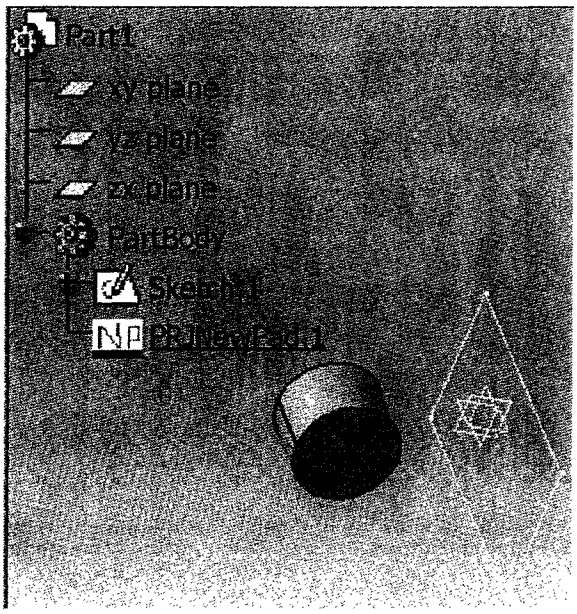

Figure 26 Après modification

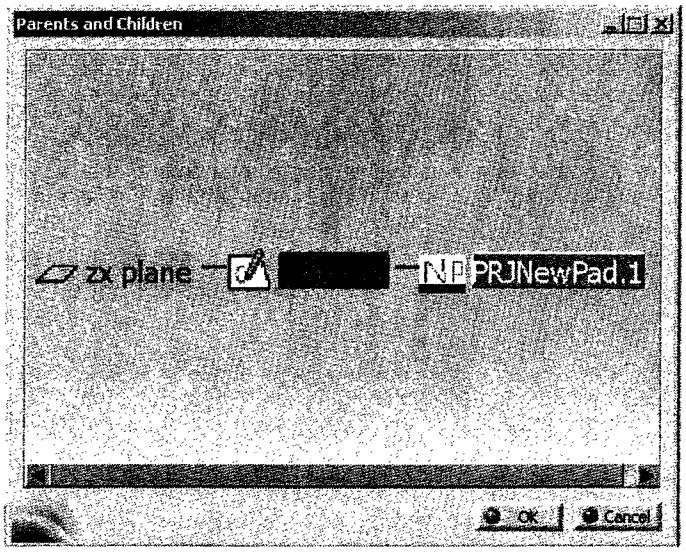

Figure 27 Relations parents/enfants après modification

Finalement, la figure 28 présente l'ensemble des modules nécessaires au développement du feature avec une liste des principaux fichiers.

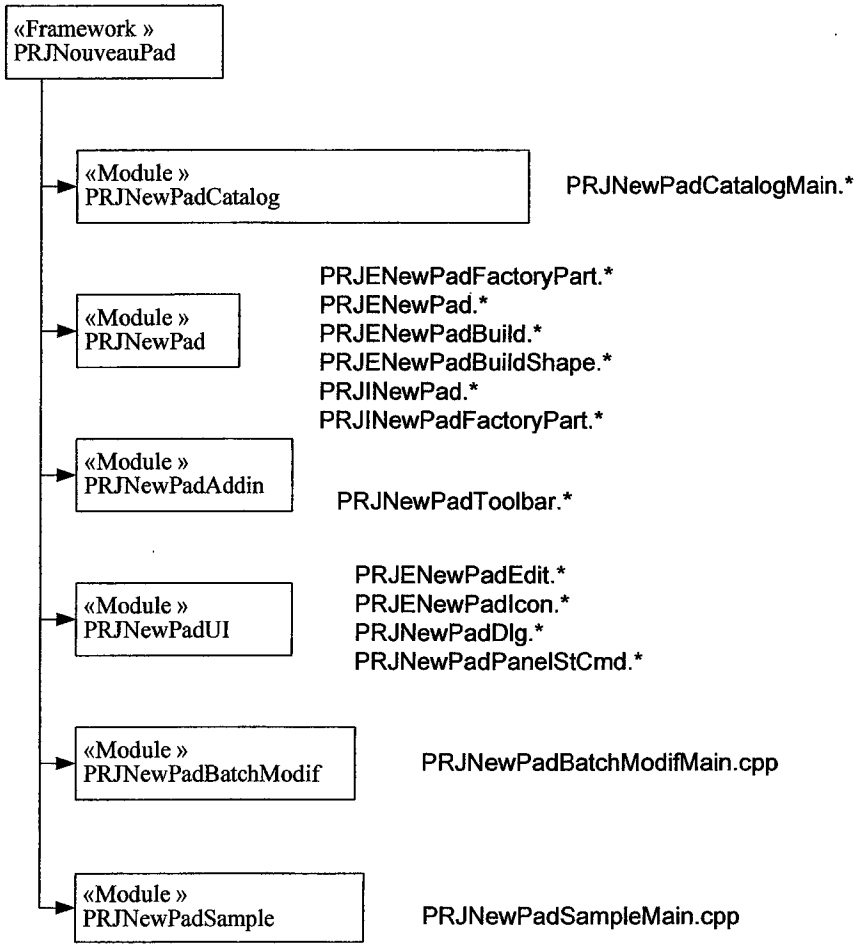

Figure 28 Liste de fichiers développés

BIBLIOGRAPHIE

Chen, Y., Shir, W.; Shen, C. (2002), Distributed engineering change management for allied concurrent engineering. *International Journal of Computer Integrated Manufacturing*, 15(2), pp 127-151.

Jacobs, T.M., (1998), *An automated framework for managing design complexity*, PhD., The University of Utah, 140 pages.

Eustache, J., (2002), *Contribution à une meilleure gestion de la cohérence des données du produit au cours de son cycle de vie*, Ph D., Université de Reims Champagne Ardenne, p 156.

Favre, J.M., Duclos, F., Estublier, J., Sanlaville, R., Jean-Jacques Auffret, J.J. (2001) Reverse Engineering a Large Component-based Software Product, Software Maintenance and Reengineering. Fifth European Conference on 14-16 March 2001, pp 95 – 104.

Sanlaville, R., Favre, J.M., Ledru, Y. (2001), Helping Various Stakeholders to Understand a Very Large Component-Based Software, Euromicro Conference, 2001. Proceedings. 27th 4-6 Sept. 2001, pp 104 – 111.

Eustache, J., Maranzana, R., Lanuel, Y., Gardan, Y., (2001), Managing complexity in a CAD environment. *IEEE International Engineering Management Conference, Change Management and the New Industrial Revolution* (IEMC'01), Oct 7-9 2001, Albany, NY, pp 104-109.

Mokhtar, A., Bedard, C., Fazio, P., (1998), Information model for managing design changes in a collaborative environment. *Journal of Computing in Civil Engineering*, 12(2), pp 82-92.

St-Martin, S., (2001), *Maîtrise du changement dans les modèles CAO*, mémoire de maîtrise, École de Technologie Supérieure, 102 pages.

Zhang, J., Chen, J., El-Mounayri, H., (1999), Generic template for collaborative product development, *American Society of Mechanical Engineers, Material Handling Division,* Industrial Virtual Reality: Manufacturing and Design Tool for the next Millennium, The ASME International Mechanical Engineering Congress and Exposition, Nov 1-Nov 2 1999, Nashville, TN, USA, pp 163-170.

El-Bibany, H.E., Paulson, B.C., (1999), Parametrics architecture for design, management, and coordination in a collaborative AEC environment. *Computer-Aided Civil and Infrastructure Engineering*, 14(1), pp 1-13.

Heller, E.B., Martiny, V., Vanderslice, J. (1989), A system for developing engineering changes, Professional Communication Conference, IPCC '89. 'Communicating to the World.', International , 1989.

Santoyridis, I., Carnduff, T. W., Gray, W. A., Miles, J. C. (1997), An Object Versioning System to Support Collaborative Design within a Concurrent Engineering Context. *Computer Science*, 1271.

Krishnamurthy K., Law K.H. (1995), Revision management in a CAD paradigm, Proceedings of the Computers in Engineering Conference and the Engineering Database Syposium, ASME, pp.1133-1144.

Oussalah, C., Urtado, C., (1997) Complex object versioning. Lecture Notes in Computer Science, v1250, Proceedings of the 1997 9th International Conference on Advanced Information Systems Engineering, CAiSE, Jun 16-20 1997, Barcelona, Spain, p 259.

Wang, Z., Chee, K. S., (2001). Managing design changes for multiviews models. Journal of Computing in Civil Engineering, 15(2), pp. 102-111.

Wang, X.,Yu, T., Zhu, C., Hu, Q., Fang, M.(2001), Research on DFRP theory and application based on PDM, *High Technology Letters*, 7(3), pp. 46-48.

Maurino, Michel (1993). *La gestion des données techniques: technologie du concurrent engineering.* Paris : Masson.

Jukka, K., Pekka, S., Jorma, T., Kari, L., (2000), *Product data management*, Technical research centre of Finland.

Flater, D., Morris, K.C.Harmonized conformance testing for product data managers.

Miles, J.C., Gray, W.A., Carnduff, T.W., Santoyridis, I., Faulconbridge, A., (2000), Versioning and configuration management in design using CAD and complex wrapped objects. *Artificial Intelligence in Engineering*, 14(3), pp. 249-260.

McClatchey, R., Baker, N., Harris, W., Le Goff, J.M., Kovacs, Z., Estrella, F., Bazan, A., Le Flour, T., (1997), Version management in a distributed workflow application. *Database and Expert Systems Applications*, 1997. Proceedings., Eighth International Workshop on , 1-2 Sep, pp. 10 -15.

Giguère, F., Rivest, L., Desrochers A., Maranzana, R., (2002), Les caractéristiques contextuelles : une solution pour accroître la productivité en CAO. *Revue internationale de CFAO et d'informatique graphique.*

Macabies, L., Desrochers, A., Rivest, L., Maranzana, R., (2001). Liens multi - modèles en CAO Application au rivetage en aéronautique, IDMME International Engineering Management Conference, Change Management and the New Industrial Revolution (IEMC'01), Oct 7-9 2001, Albany, NY, p 104-109.

Srinath, S., Ramakrishnan, R., Ram, D. J., (2000), Generic model for semantics-based versioning in projects. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 30(2), pp 108-123.

Office québécois de la langue française. Le grand dictionnaire terminologique, [En ligne], http://www.grandictionnaire.com/btml/fra/r_motclef/index800_1.asp (Consulté le 31 mars 2003).

Giguère, F., (2002), Application des liens multi-modèles à la conception mécanique. Mémoire de maîtrise, université de Sherbrooke avril 2002, 94 pages.

Shah, J. J., Urban, S. D., Raghupathy, S. P., Rogers, M. T., (1994), Product data integration framework for synergetic CAD systems. IFIP Transactions B: Computer Applications in Technology, B(17), 1994, pp 275-297.

Msaaf, O., (2002), Validation des caractéristiques d'usinage par des grammaires d'usinage attribués: Une contribution à la prise en compte des contraintes d'usinage en cours de conception, Ph.D., École de Technologie Supérieure (Canada), 262 pages.

Soh, C., Wang, Z., (2000), Parametric coordinator for engineering design. *Journal of Computing in Civil Engineering*, 14(4), pp 233-240.