Interopérabilité entre modèles hétérogènes en conception coopérative par des approches d'Ingénierie Dirigée par les Modèles

Hanène Chettaoui

INSTITUT POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THESE

pour obtenir le grade de

DOCTEUR DE L'Institut Polytechnique de Grenoble

Spécialité : «Génie Industriel»

préparée au laboratoire G-SCOP (Sciences pour la Conception, l'Optimisation et la Production de Grenoble)
dans le cadre de **l'Ecole Doctorale «Ingénierie - Matériaux Mécanique Energétique Environnement Procédés Production»**

présentée et soutenue publiquement

par

Hanène CHETTAOUI

le 25 Novembre 2008

Interopérabilité entre modèles hétérogènes en conception coopérative par des approches d'Ingénierie Dirigée par les Modèles

Directeurs de thèse : Frédéric NOËL

JURY

M. Michel Bigand	Maître de Conférences, Ecole Centrale de Lille	Rapporteur
M. Lionel Roucoules	Professeur, ENSAM Aix-en-Provence	Rapporteur
M. Alain Bernard	Professeur, Ecole Centrale de Nantes	Examinateur
M. Aziz Bouras	Professeur, Université Lumière Lyon 2	Examinateur
M. Hervé Verjus	Maître de Conférences, Université de Savoie	Examinateur
M. Frédéric Noël	Professeur, INP Grenoble	Directeur de thèse

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse Frédéric Noël pour son suivi, sa disponibilité ainsi que les nombreuses discutions que nous avons eu autour de ce travail.

J'adresse mes plus vifs remerciements aux deux rapporteurs de ce mémoire Monsieur Michel Bigand et Monsieur Lionel Roucoules pour le temps qu'ils ont investit dans cet exercice.

Je tiens à remercier messieurs les membres du jury pour avoir accepter cette charge ainsi que l'intérêt qu'ils ont porté à mes travaux de recherche : je cite Monsieur Alain Bernard, Monsieur Aziz Bouras et Monsieur Hervé Verjus.

Au-delà de tout, je dois à ma mère beaucoup de reconnaissance. Elle m'a toujours encouragé et soutenu pour avancer. Ses conseils ont aiguillé mes choix : elle est en effet à l'origine de tout mon parcours de recherche en France. Ainsi, je lui dédie cette thèse comme expression de ma profonde reconnaissance. Je remercie également mon père pour son amour et son affection ainsi que tous les membres de ma famille.

Je remercie aussi mes amis et collègues du laboratoire G-SCOP pour tous les bons moments qu'on a passé ensemble.

Je ne saurai remercier assez mes amis en France et en Tunisie pour leur soutien et leur amitié.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

AIF Athena Interoperability Framework
AMIS Automated Methods for Integrating Systems
API Application Programming Interface
ATL Atlas Transformation Language
B-Rep Boundary Representation
CAD Computer Aided Design
CAO Conception Assistée par Ordinateur
CIM Computation Independent Model
CODEMO Co-Operative DEsign MOdeler
CPM Core Product Model
CSG Constructed Solid Geometry
DXF Drawing eXchange Format
EIF European Interoperability Framework
ERP Enterprise Resource Planning
FAO Fabrication Assistée par Ordinateur
FBS Fonction Behavior Structure
GAM Gestion d'Analyses Mécaniques
IDM Ingénierie Dirigée par les Modèles
IHM Interface Homme-Machine
IDEAS Interoperability Development for Enterprise Applications and Software
IDM Ingénierie Dirigée par les Modèles
IGES Initial Graphics Exchange Specification
KBE Knowledge Based Engineering
LISI levels of information systems interoperability
MDA Model Driven Architecture
MDD Model Driven Development

MDI Model Driven Interoperability

MOF Meta Object Facility

MOKA Methodology and tools Oriented to Knowledge-based engineering Applications

MOLA MOdel transformation LAnguage

OMG Object Management Group

OMT Object Modeling Technique

OOD Object Oriented Design

OOSE Object Oriented Software Enginnering

PDES Product Data Exchange Specification

PDM Product Data Management

PDML Product Data Markup Language

PIM Platform-Independant Models

PLM Product Lifecycle Management

PM Platform Model

PPO Produit Processus Organisation

PSM Platform-Specific Models

SCM Supply Chain Management

SET Standard d'Echange et de Transfert

SGDT Système de Gestion de Données Techniques

SOA Services Oriented Architecture

STEP STandart of Exchange of Product Model Data

STEP-NC FStep compliant Numerical Command

UML Unified Modeling Language

VDA-FS Verband der Automobilindustrie Flaechen-Schnittstelle - Grafikstandard

XMI XML Metadata Interchange

XML eXtensible Markup Language

YATL Yet Another Transformation Language

Introduction générale

La conception et l'ingénierie des systèmes deviennent de plus en plus distribuées et collaboratives (Danesi *et al.*, 2006). La conception de produits nécessite la collaboration entre différents acteurs métiers. Un produit est co-défini par des équipes d'experts géographiquement distribués utilisant un ensemble d'outils métier hétérogènes. Pour faire face à ce contexte, les éditeurs logiciels, producteurs d'outils d'assistance aux expertises métier ont souvent recours au concept d'intégration. On tente ainsi d'intégrer dans un seul outil tout les expertises interconnectées. D'autres solutions proposent des environnements collaboratifs pour supporter un ensemble d'outils métiers spécifiques mais qui restent en nombre limité. Ces environnements ne sont efficaces que si les applications expertes peuvent communiquer et interopérer avec un modèle partagé. L'aptitude des expertises métier à collaborer revient plus spécifiquement à leurs capacité à interopérer avec des environnements de collaboration.

Nos travaux de thèse étudient les différentes approches existantes pour l'interopérabilité en conception de produit ainsi que les approches émergentes pour l'interopérabilité entre outils logiciels. Nous proposons à la suite de cette étude une méthodologie de synchronisation pour l'interopérabilité entre outils de conception de produit basée sur le principe de l'Ingénierie Dirigée par les Modèles. Dans cette approche, l'information décrite par un outil métier est un modèle qui est conforme à un méta-modèle. Les concepteurs utilisant les outils métiers hétérogènes pour modéliser leurs expertises contribuent à la conception des différentes vues d'un même produit. Les correspondances entre ces vues ne sont formalisées dans aucun modèle. Nous étudions comment formaliser ce genre de connaissance, et comment cette formalisation peut être employée pour aider à la synchronisation de vues métier.

Le chapitre 1 présente le contexte de notre travail : celui de la conception collaborative. C'est parce que les experts métier sont amenés à collaborer que des problèmes d'interopérabilité apparaissent. Nous commençons par détailler les approches d'organisation des activités de collaboration dans une entreprise pour suivre l'évolution des logiques d'organisation. Ceci nous montre à quel

point la communication entre acteurs de conception est indispensable pour la survie de l'entreprise. Nous exposons les différents niveaux de collaboration dans une entreprise pour se placer plutôt sur un niveau de coopération ou les acteurs métier travaillent simultanément sur les même objets. Enfin nous concluons par les problématiques auquelles nous tentons d'apporter des réponses.

Dans le chapitre 2, nous étudions les différents outils et approches existants pour l'interopérabilité en conception de produits manufacturiers. Nous détaillons les modèles produit qui offrent un moyen de modéliser l'information autour du produit. Les modèles produit ne sont utiles que s'ils peuvent être échangés. Nous exposons les différents standards d'échanges utilisés et spécialement le standard STEP, le plus utilisé, pour l'échange de données autour du produit. Nous exposons aussi les technologies développées pour favoriser l'échange d'information entre les Système de Gestion de Données Technique dans une optique d'interopérabilité.

Le chapitre 3 explore les outils utilisés pour nos travaux de thèse. La première partie traite de l'Ingénierie Dirigée par les Modèles (IDM) comme solution pour l'interopérabilité entre outils logiciels. Nous présentons spécialement la MDA (une mise en oeuvre de l'IDM) et ses différents composants ainsi que le concept de transformation de modèles. La deuxième partie traite l'étendue de la problématique d'interopérabilité à l'échelle de l'entreprise. Nous exposons l'apparition de l'Interopérabilité Dirigée par les Modèles pour cadrer l'interopérabilité pour les applications et logiciels d'entreprise ainsi que les différents cadres apparus pour classer les degrés d'interopérabilité. Enfin nous discutons l'interopérabilité en entreprise manufacturière et concluons par l'application possible de l'IDM dans ce contexte.

Le chapitre 4 présente l'application des concepts de l'IDM en l'occurrence de modélisation et de méta-modélisation avec des outils métiers de conception et de fabrication. Au cours de ce chapitre, nous présentons l'environnement coopératif GAM que nous utilisons pour implémenter nos modélisations. Pour disposer des modèles métier dans l'environnement GAM, un moyen d'export est nécessaire. Pour cela, nous détaillons les différents étapes d'établissement de ce module. Nous testons notre approche avec deux outils métier, de CAO : $SolidWorks^{TM}$, et de FAO : $Esprit^{TM}$. A la fin de ce chapitre, nous disposons des différents éléments à partager des modèles métier dans un environnement unique : GAM.

Disposer de ces modèles dans un environnement unique comme GAM ne permet pas la synchronisation. Dans le chapitre 5, nous détaillons notre proposition de module de synchronisation entre outils métier hétérogènes. Nous nous situons dans une approche fédérée pour l'interopérabilité. Nous expliquons l'architecture de l'environnement et détaillons les différents composants du module de synchronisation. Nous spécifions les différents modèles et méta-modèles mise en jeu : métier, de fédération et de correspondance. Les modèles métiers sont ceux des outils de CAO et de FAO précédemment choisis. Le modèle de fédération est celui du modèle produit de PPO (Produit Processus Organisation). Nous définissons un méta-modèle de correspondance qui permettra de formaliser les connexions possibles entre différent expertises. Nous présentons l'implémentation technique des modules d'export et d'import pour l'extraction des concepts de partage des modèles métier à partir des outils métier. Nous identifions le rôle des différents acteurs, utilisateurs, experts métier et développeur logiciels pour assurer la qualité de ce process. Nous définissons une méthodologie pour la synchronisation de modèles métier homogènes. Tout au long de ce chapitre nous présentons l'implémentation de nos propositions sur notre scénario d'usage. Nous terminons par un bilan de fonctionnement du module de synchronisation. Nous exposons les limites de l'application de l'IDM entre outils métier et la nécessité de prise en compte de démarche hors IDM pour l'aboutissement à un module de synchronisation générique.

Nous clôturons ce document en donnant les conclusions et les perspectives de ce travail.

Chapitre 1

Besoin de collaboration en conception

Sommaire

La conception est une activité variée et complexe qui implique de nombreux acteurs et moyens matériels (Charles, 2005). Les applications CAO usuelles sont complexes, et ne sont pas dédiées à des pratiques collaboratives. Plusieurs travaux ont essayé de donner des réponses méthodologiques et en terme d'outillages pour assister la conception de produits manufacturiers. L'ambition de ce document est de proposer une architecture collaborative favorisant la collaboration entre utilisateurs de CAO et de ce fait, à toute une gamme d'applications dont l'usage en mode collaboratif est peu évident.

1.1 Introduction

Ce chapitre présente le contexte dans lequel évoluent les travaux menés dans notre thèse ainsi que la problématique qui en est à l'origine. Dans la partie relative au contexte, nous présentons les différentes approches d'organisation dans un processus de conception. Ces approches soulignent l'évolution de l'activité de conception d'une organisation séquentielle vers une organisation simultanée et intégrée, et enfin vers une organisation collaborative. Nous définissons alors le sens de la collaboration pour une activités de conception. Nous exposons les différents niveaux de la collaboration pour spécifier notre positionnement sur un niveau de coopération, où des experts métiers travaillent de façon simultanée sur les mêmes objets, plutôt qu'un niveau de collaboration stratégique entre entreprises.

1.2 Le processus de conception : Approches d'organisation

Il est couramment admis que la conception d'un produit commence par une définition des besoins (cahier des charges). Le concepteur essaye de répondre à ces besoins par la proposition de solutions qui au début ne sont que des idées vagues. Le concepteur procède à l'étude de certaines solutions pour en retenir celle qui correspond le mieux à ses besoins. Cette solution émerge progressivement.

De l'idée à la mise à disposition, plusieurs étapes et compétences sont nécessaires pour arriver à la conception d'un produit final. Dans la littérature, plusieurs tentatives d'organisation du processus de conception en phases sont apparues (Pahl et Beitz, 1996). Le processus de conception évolue souvent du général au spécifique (de l'abstrait au concret). La conception est vue comme la transformation d'un concept (abstrait) en un produit (réel).Dans la plupart des cas de conception, une idée générale est détaillée, étape par étape (Miaoulis, 2002).

Suite à l'expression des exigences des clients et aux contraintes du marché, des nouvelles méthodes de travail sont apparues pour faire face aux pressions de la concurrence en réduisant les délais de production. Les industries délaissent l'ingénierie séquentielle et privilégient l'ingénierie simultanée et intégrée.

1.2.1 L'approche séquentielle

L'ingénierie séquentielle est une approche traditionnelle de conception. En ingénierie séquentielle, les activités de conception sont exécutées séquentiellement avant l'élaboration des procédés de production dans des départements spécifiques. Les acteurs métier de la conception, du marketing à la vente en passant par le bureau d'études, se répartissent les tâches. Chaque expert métier se focalise sur son domaine d'expertise sans intervenir directement dans les activités d'un autre domaine de spécialité.

L'ingénierie séquentielle propose une organisation séquentielle des acteurs selon les étapes du processus de développement du produit. Le processus de développement se décompose en plusieurs phases :

- phase de spécifications : dans cette phase on définit le cahier des charges qui comporte les contraintes de développement ainsi que les exigences du client. Une fois les besoins définis, ils sont communiqués au bureau d'études,
- phase de conception générale : dans cette étape, le bureau d'études élabore les divers schémas conceptuels possibles qui permettent de satisfaire le cahier des charges défini précédemment,
- phase de conception détaillée : ici les solutions techniques sont décrites. La nomenclature du produit réel est définie. Si le produit est un objet manufacturier, sa représentation en 2D ou 3D est réalisée. Ces maquettes permettent des analyses variées du produit,
- phase de production : dans cette phase la fabrication des solutions définies par le bureau d'études est réalisée. Des tests sont réalisés pour vérifier si le résultat satisfait les besoins définis à l'origine.

Les phases se déroulent séquentiellement et en théorie sans chevauchement. Cette approche ne reflète pas le processus de conception réel d'un produit manufacturier. En effet travailler selon cette approche implique que chaque expert métier réalise sa tâche de façon indépendante et sans aucune collaboration avec les autres services. De tout temps lors de la conception d'un produit manufacturier, le phase de conception a toujours pris en compte des paramètres de fabrication. Cependant, certaines organisations cloisonnent plus ou moins les phases successives et rendent effectives les pratiques séquentielles. Dès lors, ces paramètres de fabrication décidés par un concepteur non spécialiste du domaine de la fabrication peuvent être raffinés ou modifiés par le fabricant. Des informations de modification doivent alors être transmises de la phase de fabrication à la phase de conception.

Les modifications qui surviennent en phase de fabrication sont des modifications tardives. Leur impact sur le temps de développement du produit est considérable et ne touche pas seulement le modèle de conception de la pièce mais aussi celui de la fabrication. Pour réduire le coût de développement, il faut réduire les modifications tardives. De nouvelles méthodes de travail sont apparues pour résoudre ces inconvénients avec pour objectif essentiel la réduction des délais et des coûts de développement. Ces méthodes de travail sont entre autres l'ingénierie simultanée et l'ingénierie intégrée.

1.2.2 Conception simultanée et Conception Intégrée

L'ingénierie concourante ou simultanée est apparue pour accélérer les délais de développement et ainsi répondre aux exigences accrues des clients. Son but est d'offrir au client une meilleure qualité du service et de mettre fin au gaspillage de temps dû à la redondance des données et aux problèmes de communication entre les fonctions de conception et de fabrication de produit.

La conception d'un produit est constituée d'un ensemble d'étapes qui sont réalisées de façon simultanée. Les concepts de l'ingénierie simultanée permettent un travail en synergie des différents acteurs autorisant une réduction des délais dès la conception jusqu'à la mise a disposition du produit. L'ingénierie simultanée déploie des équipes multidisciplinaires et multi-métiers travaillant en parallèle, le plus tôt possible, vers un même but. La figure 1.1 illustre le gain de temps apporté par la démarche d'ingénierie simultanée.

De son côté, l'ingénierie intégrée est une approche organisationnelle multidisciplinaire qui tend à l'intégration simultanée de la conception des produits et de l'étude des procédés de fabrication jusqu'à la prise en compte de la logistique de soutien nécessaire à l'exploitation de ces produits par les utilisateurs (Jagou, 1993). L'ingénierie intégrée est une approche systématique pour concevoir un produit en considérant tous les éléments de son cycle de vie, depuis la conception jusqu'à la mise à disposition. Il s'agit de prendre en compte toute les contraintes métiers au plus tôt dans le cycle de vie du produit.

A partir de l'identification des besoins du client, cette ingénierie intègre la définition du produit en intégrant toutes les sphères d'activités reliées à son développement : les processus de fabrication, et tous les autres processus requis dans le cycle de vie tels que la recherche, les achats, la qualité, l'expédition, le marketing, ou encore la maintenance. Il doit en résulter un produit de grande qualité qui répond aux besoins du client, qui peut être mis en production plus rapidement que par les méthodes de conception traditionnelles.

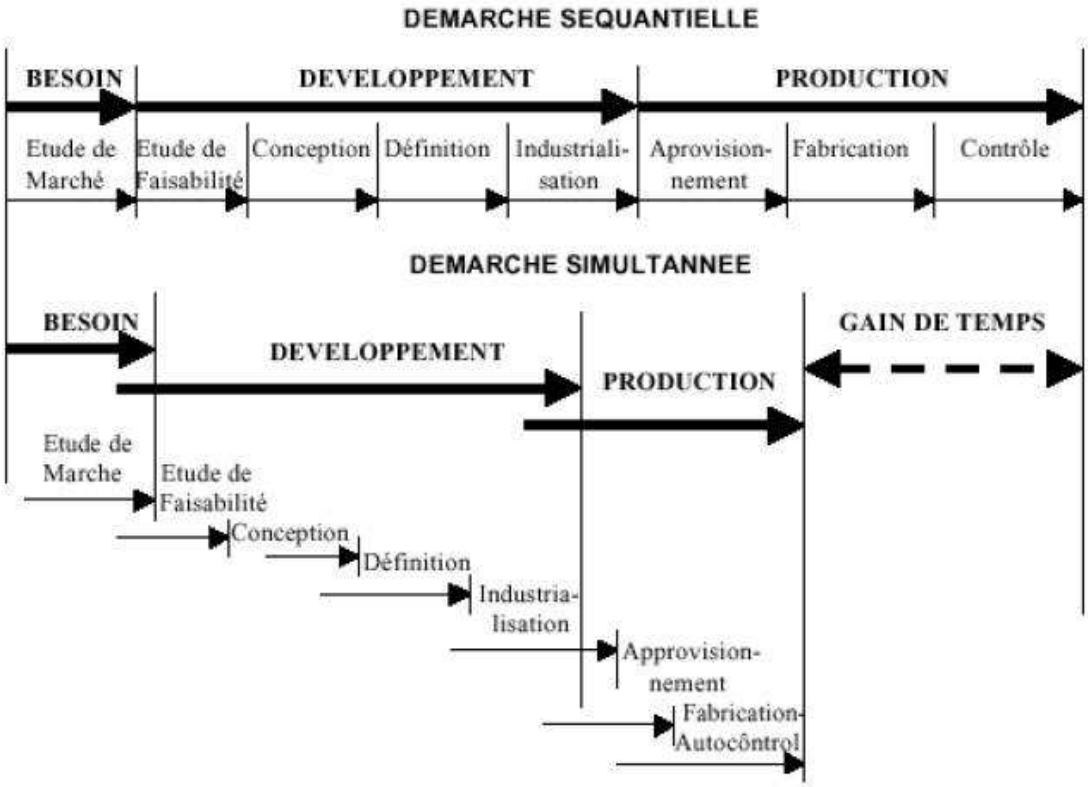

FIG. 1.1 – La démarche simultanée

La communication entre acteurs de conception est indispensable aux pratiques d'ingénierie simultanée et intégrée. Les activités des différents métiers intervenant dans le processus de développement doivent être mises en parallèles. L'incitation au travail collaboratif dès le début du développement forme un axe majeur de ces approches.

1.2.3 De l'ingénierie simultanée ou intégrée vers l'ingénierie collaborative

La conception est une activité de définition des produits manufacturiers ou des services. Cette activité fait participer un certain nombre d'acteurs qui sont des personnes, des services ou des entreprises. Un projet est subdivisé en plusieurs sous-projets où chaque corps de métiers intervient sur une tâche précise relevant de son expertise. Que l'organisation soit simultanée ou intégrée, un point clé est la collaboration de ses acteurs. En effet la conception de produit complexe nécessite l'intervention de plusieurs acteurs qui sont géographiquement distribués.

Des systèmes d'aide à la collaboration sont apparus grâce à l'évolution des technologies informatiques. L'efficacité et la complexité de ces systèmes se sont accrus considérablement. Mais, parallèlement, les exigences en support de collaboration se sont multipliées.

1.3 Conception collaborative

1.3.1 Définition

La collaboration est un processus dans lequel les acteurs, ayant des intérêts variés, travaillent ensemble pour chercher des solutions à un objectif commun. La conception est la transformation d'un concept en un produit (Marsot, 2002). La conception collaborative peut être vue comme une succession d'actions conduisant à la définition d'un produit à partir d'un besoin défini de façon formelle (cahier des charges) ou non (Charles, 2005).

Le but principal d'une collaboration efficace est de diminuer le temps de commercialisation du produit, d'améliorer le processus de conception et de piloter les informations de conception. Les groupes de collaboration doivent avoir accès à l'information dont ils ont besoin pour réussir la collaboration.

Ceci mène à définir les points clés pour une collaboration réussie. Un point clé principal est de fournir un environnement sécurisé où les acteurs peuvent exprimer des avis controversés. Un environnement où toutes les expertises nécessaires peuvent communiquer, partager de l'information et être préparées à

tous les adaptations possibles. Plusieurs approches de collaboration ont émergé pour exprimer les besoins des expertises. Le défi principal est de choisir la bonne approche pour une collaboration efficace. Nous nous intéressons aux approches qui se focalisent sur les acteurs et le système de collaboration et qui permettent leur modélisation.

Dans la section suivante, nous détaillons les différents niveaux de collaboration. Notamment les notions d'interdépendance et de coordination des activités seront présentées. Ensuite, une comparaison entre le concept de collaboration et de coopération est faite.

1.3.2 Différents niveaux de collaboration

(Touzi, 2007) s'appuie sur certains résultats issus de la théorie des systèmes multi-agents (Bouzguenda, 2006) pour caractériser des niveaux de collaboration. Nous confirmons la vison de (Touzi, 2007) sur la définition de ces différents niveaux sauf pour la coopération où nous fournissons une autre définition. Cette théorie propose les niveaux de collaboration suivants (où chaque niveau englobe le niveau précédent) :

1. le niveau de communication exprime la façon dont les personnes se comprennent et la façon dont l'information est transférée dans les organismes. Les entreprises et organisations communiquent, échangent et partagent des informations afin d'optimiser leur fonctionnement. Ce niveau de collaboration consiste en un échange simple des données des entreprises.
2. le niveau de coordination s'occupe de la synchronisation de tâches. Les entreprises et organisations réalisent des tâches dont dépendent leurs partenaires et réciproquement. Cette coordination a pour vocation l'amélioration de leurs performances individuelles. Ce niveau de collaboration, s'il semble faire apparaître un embryon de processus collaboratif (tâches partagées et synchronisées) correspond en fait à une individualisation des activités : l'absence d'objectif commun traduit l'impossibilité de concevoir un processus collaboratif en tant que tel. Ce sont uniquement leurs compétences (applications, fonctions, services) que les entreprises partagent ou mettent à disposition.
3. le niveau de coopération est associé à la réalisation d'activités interdépendantes par plusieurs entités qui opèrent simultanément sur les mêmes objets. A ce niveau opératoire, il n'est plus question de planifier des tâches : le faire alourdirait abusivement la gestion du projet.
4. le niveau d'intégration : fusion en une même entité. A ce niveau de collaboration, l'échange de données, le partage des tâches et la poursuite d'un objectif commun sont inhérents au fait que les entreprises et orga-

nisations se trouvent intégrées au sein d'une même entité (virtuelle ou concrète).

La réponse en terme de support à la collaboration doit être adaptée au niveau d'organisation visé. Dans cette thèse nous viserons particulièrement le mode coopératif, où les partenaires définissent un même objet mais souvent dans des formalismes différents.

1.3.3 Collaboration VS Coopération

En analysant le travail collectif humain, il y a généralement un manque de théorie conventionnée. Deux concepts cruciaux sont confondus : la collaboration et la coopération. D'après (Nezamirad *et al.*, 2005), deux facteurs distinctifs sont principalement employés pour distinguer la collaboration et la coopération : la répartition des tâches et la coordination :

- la répartition des tâches : dans le travail coopératif une tâche est subdivisée en plusieurs activités entre les participants. D'après (Leont'ev, 1981), une activité est une structure hiérarchique à 3 niveaux interactifs de relation entre les individus et les objets. Ces trois niveaux sont : l'activité, l'action et l'opération. En coopération, les participants travaillent simultanément à un niveau action de l'activité. La collaboration implique l'engagement mutuel sur des activités. La collaboration peut impliquer la répartition des tâches qui englobent des activités.
- la coordination est un facteur de distinction de collaboration. Dans le travail coopératif, la coordination est nécessaire pour l'assemblage et la synchronisation de résultats. En coopération, le comportement de chaque coopérateur doit être prévisible.

La répartition des tâches est un attribut de la collaboration. La coopération se produit à un niveau action de l'activité. La coordination est un attribut commun à la coopération et à la collaboration. La collaboration est un état qui a un certain nombre de niveaux : un niveau est la coopération.

Dans la collaboration, les participants sont engagés dans une activité pour accomplir un objectif commun. Ils ont leurs propres buts, actions, connaissances, règles d'organisation et structure. Ils coopèrent les uns avec les autres pour atteindre l'objectif commun (Nezamirad *et al.*, 2005). La figure 1.2 montre que le modèle de collaboration devrait inclure différents objectifs communs et individuels, tâches communes et individuelles, rôles de groupe, procédés décisionnels de groupe, processus de coopération ainsi que les connaissances communes et les connaissances métier.

La coopération et la collaboration diffèrent donc en niveau d'opération. Deux personnes sont dans une situation coopérative si chacune d'entre elles tra-

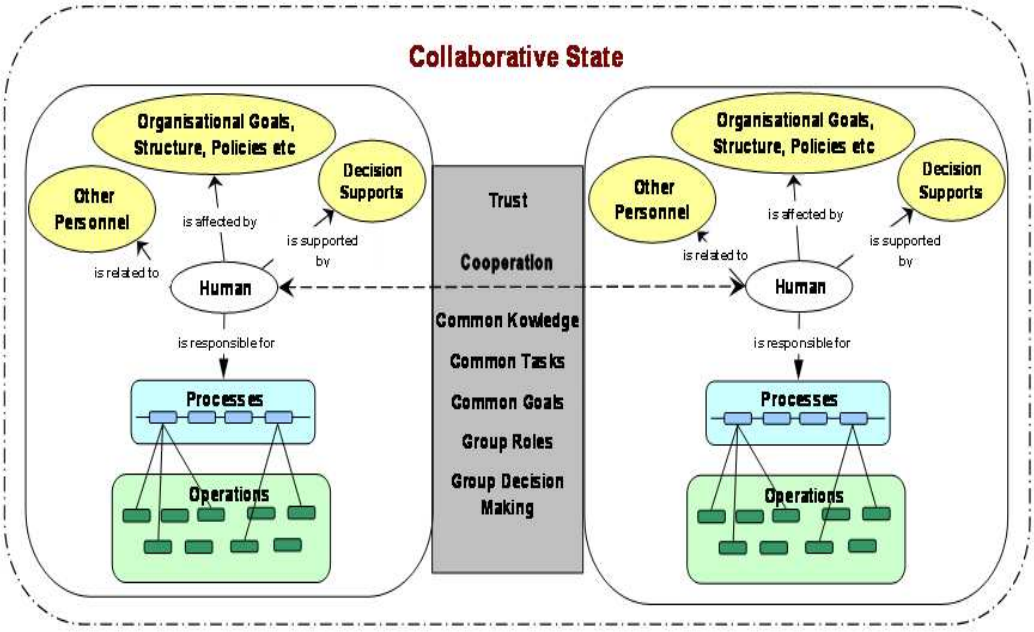

FIG. 1.2 – Le modèle de collaboration (Nezamirad *et al.*, 2005)

vaille sur ses activités dont dépend l'autre personne pour accomplir ses objectifs et peut interférer avec l'autre sur des objectifs, des ressources, des procédures, etc.. Chacun essaye de gérer l'interférence pour faciliter les différentes activités (Hoc, 2001). Le travail dit coopératif se réalise dans un contexte où plusieurs ressources distribuées (humaines ou organisationnelles) se partagent un ensemble d'activités ayant entre elles un certain nombre d'interdépendances.

(Hoc, 2001) classe les processus de coopération en trois niveaux :

- action : est le niveau bas de communication. Le niveau de la coopération dans l'action est réalisé par une équipe travaillant ensemble pour exercer des activités coopératives opérationnelles directement relatives à la gestion de but et de procédé en temps réel pendant l'exécution de tâche.
- plan : est le niveau de planification du processus coopératif. Les activités coopératives se produisent à un niveau d'abstraction plus haut que celui de l'exécution d'action. Elles peuvent faciliter la coopération dans le niveau «action».
- méta : données générales utiles aux processus de coopération des deux niveaux plus bas. Le niveau «méta» représente la méta-connaissance. À ce niveau, des données plus génériques sont produites.

De leur côté, (Johansen et Charles, 1988) définissent le cadre espace-temps pour prendre en compte la localisation physique des acteurs et le moment où ils participent à la conception.

Nous nous basons sur ces deux travaux pour proposer le cadre de coopération présenté dans la figure 1.3. Ce cadre ajoute une troisième dimension au cadre de (Johansen et Charles, 1988) pour prendre en compte le niveau de coopération entre les acteurs impliqués dans processus de conception coopérative.

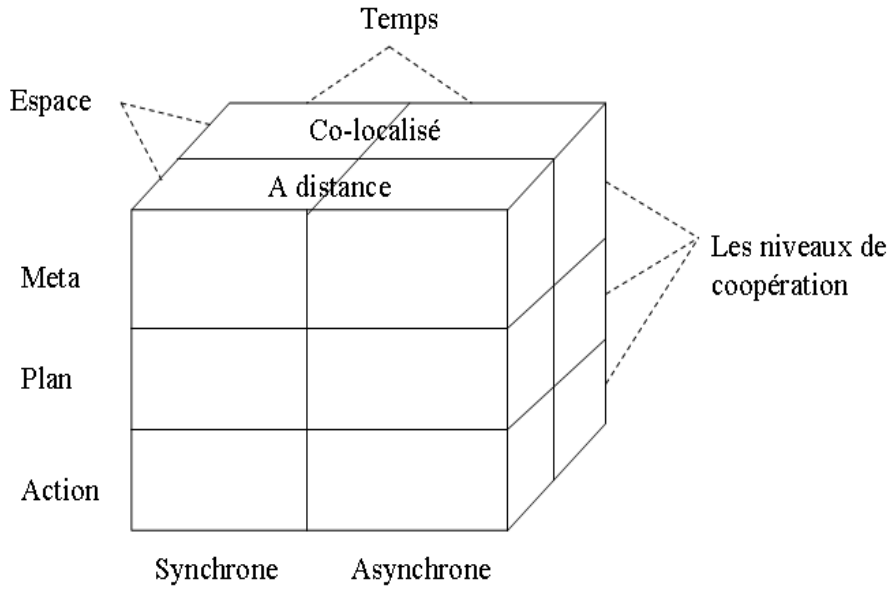

FIG. 1.3 – Le cadre de coopération

Dans ce travail, nous sommes intéressés par des acteurs métier qui opèrent simultanément sur les mêmes objets et donc en mode «synchrone». Un support à la coopération est nécessaire à des acteurs géographiquement distribués. Nous nous focalisons sur des acteurs travaillant «A distance». Nous essayons de répondre à des problématiques d'ordre opératoire. Nous nous situons donc sur le niveau «action» de la coopération.

1.3.4 La conception coopérative : structure

L'organisation institutionnelle impose nécessairement une certaine structure au collectif de travail, qui peut aller d'une forme «lâche» à une structure très hiérarchisée (Mundutéguy et Darses, 2000). La structure hiérarchisée est celle de la coopération «verticale». La coopération verticale est un rapport hiérarchique d'autorité entre les parties. Chaque partie doit coopérer avec l'autre

partie au niveau plus bas, contrôler les interférences et s'assurer également que la deuxième partie peut faciliter son objectif.

La structure hiérarchique se modifie en fonction des situations rencontrées. La coopération peut ainsi être «horizontale» (forme «lâche»). Dans cette structure, le but est de synchroniser les différents points de vue des acteurs. (Mundutéguy et Darses, 2000) précisent que cette synchronisation cognitive a pour objectif de construire un référentiel opératif commun. Nous nous situons sur une structure «horizontale» de la coopération. Nous essayons de fournir une aide à cette activité en proposant une représentation partagée des différents objets de l'activité de coopération.

1.4 Conclusion

Les nouvelles organisations industrielles imposent une intensification des pratiques collaboratives en impliquant des ingénieurs, des services, des partenaires différents dont il faut gérer les actions et coopérations. L'objectif est d'optimiser les pratiques de conception pour en réduire les coûts et les délais.

Pour développer des produits concurrentiels, les équipes de conception dans le monde entier doivent collaborer efficacement avec leurs associés internes et externes. Dans ce chapitre, nous avons présentés les différentes approches d'organisation dans un processus de conception. Après avoir définis la collaboration, nous avons signalé que notre thèse s'intéresse à des situations de coopération «synchrone» et «à distance». Nous sommes sur une structure horizontale de l'activité de coopération. Cette thèse se place dans ce cadre et a pour objectif de fournir une aide à l'activité de coopération en proposant une représentation partagée des différents objets de l'activité de coopération à travers un environnement coopératif.

Plusieurs outils et méthodes ont été développés pour répondre à des besoins de collaboration et faciliter l'échange d'information entre les acteurs de la conception tout au long du cycle de vie du produit. Parmi ces approches, nous exposerons dans le chapitre 2 de ce manuscrit :

- les modèles produit : qui sont manipulés pour décrire un produit lors de l'activité de conception,
- les standards d'échanges : qui permettent aux concepteurs d'extraire leurs besoins en informations depuis un modèle unifié pour construire leur propre modèle,
- les PDM (Product Data Management) : qui est une application de gestion des documents techniques qui permet de faciliter le travail collaboratif. Des outils pour l'interopérabilité entre les PDM ont été développés telle que les PLM services et les PDM Enablers.

Chapitre 2

Modèle et approches pour la conception collaborative

Sommaire

2.1 Modèle pour la conception collaborative

Cette section s'intéresse à la représentation des concepts dans différents modèles couramment utilisés ou proposés pour l'assistance à l'activité de conception.

Le développement de produit nécessite plusieurs phases qui incluent plusieurs acteurs humains disposant d'un savoir scientifique et technologique, et interagissant avec un environnement externe (figure 2.1). De ce point de vue, trois types de préoccupations fondamentales accompagnent la phase de développement du produit :

1. la représentation du produit est un ensemble de connaissances et d'informations. La modélisation de produit est une approche qui vise à structurer les connaissances, les informations et les comportements autour du produit tout au long de son cycle de vie,
2. le processus de conception : On définit un processus comme un ensemble d'activités qui utilisent des ressources (personnel, équipement, matériels, informations) pour transformer des éléments pris en entrée en éléments avec une valeur ajoutée. Un modèle de processus de conception décrit donc les étapes qui conduisent entre autres à l'élaboration du modèle produit et correspondent ainsi à l'activité « métier » de l'organisation.
3. l'organisation désigne la façon dont l'entreprise est structurée pour réaliser son activité. On définit l'organisation de l'entreprise comme le cadre structurel du processus de décision collectif. L'organisation peut être vu comme une médiation entre le processus de production et l'environnement. Une organisation maîtrisée est nécessaire pour la réussite de la conduite du processus de conception. Ainsi, la prise en compte du contexte organisationnel permet d'améliorer le processus de conception et donc du produit le résultat de ce processus.

Nous sommes spécialement intéressés par le développement du produit en tenant compte de son cycle de vie. Dans ces phases de développement différents modèles produits sont utilisés pour décrire les informations relatives au produit. Différents standards sont également utilisés pour échanger l'information entre les experts métiers. Dans la première section, nous nous focalisons sur les modèles dédiés à la représentation de produits. Ensuite nous parcourons les différents standards utilisés pour l'échange et le partage d'information.

2.1.1 Modeleurs de conception

La représentation de la géométrie joue un rôle majeur dans les activités de conception mécanique. La CAO (Conception Assistée par Ordinateur) est apparue au milieu des années soixante-dix avec deux approches de modélisations

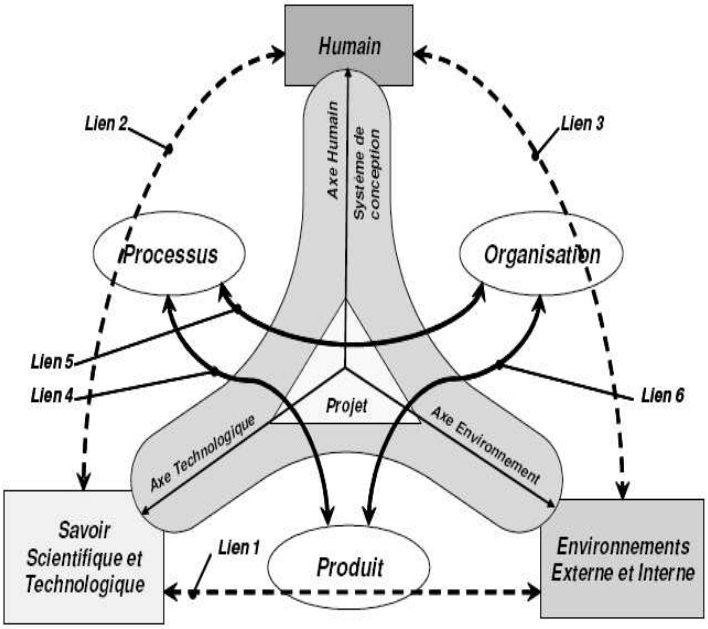

FIG. 2.1 – Modèle de description du système de conception d'aprés (Rose *et al.*, 2005)

qui ont permis de définir sans ambiguïté les modèles géométriques des parties du produit : les modeleurs B-Rep ou CSG, historiquement précurseurs de la CAO, et plus récemment les modeleurs basés sur la notion de caractéristiques. Durant la phase de conception, ces deux modélisations sont couramment utilisées dans les systèmes de CAO.

Les modeleurs géométriques CSG

Le modèle géométrique est au cœur du système de CAO. Il contient toutes les informations géométriques des entités créées. Un objet est identifiable avec son modèle volumique. La modélisation CSG (*Constructed Solid Geometry*) initiée par (Voelcker et Requicha, 1977) est une modélisation dans laquelle l'objet est défini par une suite d'opérations ensemblistes (union, intersection, soustraction, etc.) sur des solides élémentaires comme un cube, un cylindre, un cône, une sphère (Laug, 2005).

Dans un modèle CSG, le système conserve un historique de la construction de l'objet. Cet historique représente, dans la plupart des cas, les objets simples manipulés (cylindre, cube ...) et les opérations booléennes (soustraction, addition, intersection) qui leur ont été appliquées. Un arbre CSG est une repré-

sentation compacte qui peut explicitement refléter l'historique de construction d'un modèle produit. Il n'est pas univoque car il y a différentes manières de construire un modèle d'un même produit. Le modeleur CSG ne permet pas d'avoir une définition explicite de la surface de l'objet dans son abstraction initiale. Pour palier à cela, il est souvent associé à un modèle B-Rep simplifié (facettisé).

Les modeleurs géométriques par les frontières B-Rep

La modélisation B-Rep (Boundary Representation) (Weiler, 1986) (Flynn et Jain, 1991) est une modélisation par les frontières dans laquelle l'objet est représenté par la topologie et la géométrie des courbes et des surfaces qui recouvrent sa frontière. Un solide est donc représenté en décrivant les faces, arêtes, et sommets qui séparent le domaine intérieur du domaine extérieur des solides. Chaque surface élémentaire, appelée carreau, est définie par une application d'un domaine paramétrique plan vers l'espace tridimensionnel. Dans un modèle B-REP, le système connaît la peau de l'objet et l'orientation de la matière. La peau peut être approchée par des facettes planes ou par des surfaces analytiques ou paramétriques.

Les modeleurs géométriques permettent la représentation de formes très complexes en 3 dimensions et offrent au concepteur un grand degré de liberté. Ces modeleurs ne rendent compte que de la forme du produit et n'incluent pas les différents types de connaissances exigées par les concepteurs pour le construire, l'évaluer et l'utiliser. Les opérateurs de manipulation demeurent basiques et orientés géométrie. Pendant la conception, une construction longue et fastidieuse est nécessaire au concepteur pour modéliser sa pièce. Des risques d'erreurs topologiques limitent la facilité de ces modeleurs (Noel, 1992). Ils demeurent par ailleurs difficiles à modifier.

Les modeleurs géométriques basés sur les caractéristiques

Les modèles BREP permettent de représenter des formes complexes avec un bas niveau sémantique (d'un point de vue de conception). Lors de la conception de pièces mécaniques, le concepteur se base sur les significations fonctionnelles des différentes formes géométriques qui constituent la pièce (ajout, enlèvement de matière, etc.). L'avantage des modeleurs géométriques basés sur les caractéristiques (Shah et Rogers, 1988) est leur capacité d'associer des informations fonctionnelles et techniques à la description géométrique du produit.

Le concept de caractéristique a une sémantique associée à une activité particulière du cycle de vie afin de faciliter l'utilisation de modèles CAO dans

différents domaines d'ingénierie. Cette sémantique intègre les intentions de conception ou le sens de la géométrie d'une pièce. Le concept de caractéristique associe des propriétés à un ensemble d'entités topologiques-géométriques. Ces propriétés sont des informations liées à la conception, à l'usinage, à la maintenance, à la simulation numérique, etc. Le concept de caractéristique associe la connaissance sous formes d'attributs utilisés dans différentes phases du développement à la forme géométrique.

La modélisation par caractéristiques est la méthode la plus couramment utilisée pour la conception de pièces dans les logiciels CAO. Les logiciels CAO offrent des opérations prédéfinies permettant de construire un modèle solide. Ainsi on peut facilement modéliser des perçages, des bossages, des dépouilles, etc., mais ces caractéristiques demeurent souvent limitées à une interprétation géométrique simple.

(Dragonas, 2006) identifie les avantages des modeleurs à base de caractéristiques :

- ils facilitent la modélisation de formes 3 dimensionnelles de complexité moyenne,
- ils permettent au concepteur d'apporter des changements simples au projet, et réintroduisent une notion d'historique de construction,
- ils offrent un niveau d'abstraction proche de celui de l'expert métier.

Toutefois ces modeleurs ont aussi leurs limites : la modélisation par caractéristiques peut avoir différentes descriptions en fonction du contexte de l'application, ce qui montre que la seule approche de conception par caractéristiques est insuffisante (Foucault, 2007). Par exemple, une opération CAO de découpage par révolution peut représenter le passage d'une vis pour la conception, et un perçage pour la fabrication.

2.1.2 Modèle FBS

L'origine des concepts de fonction, de comportement, et de structure découle des années 70 où Rodenacker propose une méthodologie de conception pour les concepteurs débutants (Gorti *et al.*, 1998). Le modèle FBS (Fonction Structure Comportement) permet la représentation de fonctions du produit, de sa structure et de ses comportements internes (figure 2.2). Dans cette méthodologie, un concepteur détermine d'abord la fonction principale d'une entité à partir des spécifications du cahier de charges. La fonction est divisée en sous-fonctions, les sous-fonctions sont divisées en sous-sous-fonctions, et ainsi

récursivement, jusqu'au niveau où des solutions physiques connues ou technologiques réalisent de telles sous-fonctions.

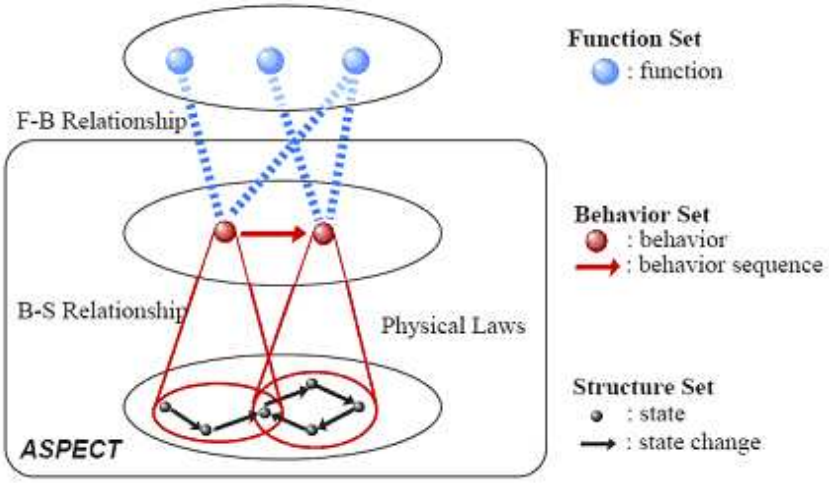

FIG. 2.2 – Relation entre Fonction, Structure et Comportement d'après (Takeda *et al.*)

Selon (Gero, 1990) (Gero et Kannengiesser, 2003) qui s'intéresse à l'analyse cognitive des activités des concepteurs, le schéma FBS représente la connaissance de conception (l'objet de conception) sous forme de trois notions abstraites : la fonction (f), le comportement (b) et la structure (s). Ces trois notions sont définies comme suit :

- la fonction (f) d'un objet de conception définit sa téléologie (finalité, but),
- le comportement (b) d'un objet de conception est défini comme l'attribut dérivé ou prévu de sa structure,
- la structure (s) d'un objet de conception est définie en tant qu'entité et les relations entre ces entités.

Bien qu'il existe plusieurs approches de représentation des fonctions, un problème commun persiste dans toutes ces approches : la fonction et le comportement sont généralement confondus. Le comportement peut être déduit directement de la structure et de l'environnement de l'objet alors que la fonction est reliée non seulement à la structure et l'environnement mais aussi à la perception de l'objet par les concepteurs. (Takeda *et al.*) définit la fonction comme «une description du comportement soustraite à travers la reconnaissance du comportement pour l'utilisation».

Pour une fonction donnée il peut y avoir plusieurs comportements mais peu d'entre eux sont significatifs pour les concepteurs lorsqu'ils conçoivent un produit.

2.1.3 CODEMO

Issus des travaux de (Chapa, 1997), (Roucoules, 1999) propose le système CODEMO (Co-Operative DEsign MOdeler) qui est un modeleur de conception intégrée. Cet outil permet d'assister les concepteurs dans la création du modèle produit lié à une étude de conception bien précise. Il permet aussi une multi-représentation des données dans chacune des vues métiers du produit. La figure 2.3 montre l'architecture du modeleur CODEMO.

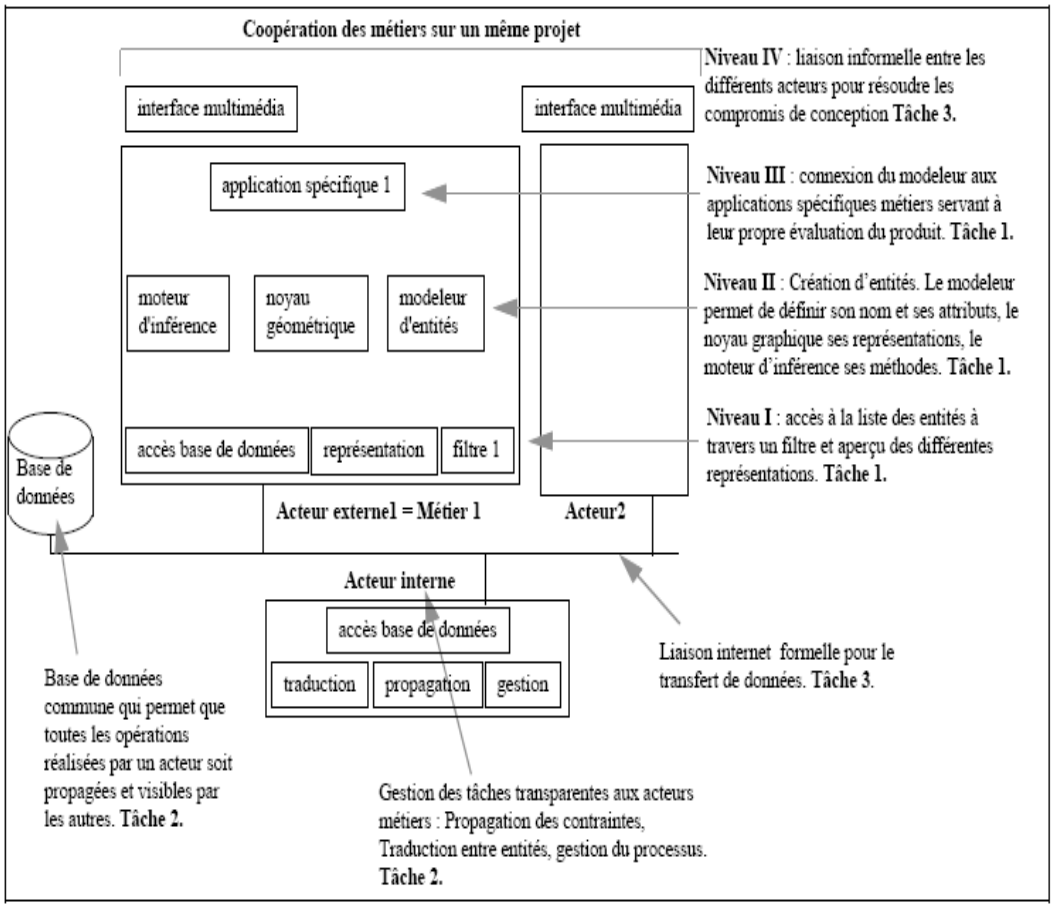

FIG. 2.3 – Architecture du modeleur de conception intégrée CODEMO (Roucoules, 1999)

Le modeleur donne, via une interface utilisateur, la possibilité de créer dynamiquement un modèle produit, ses vues Technologique, Ossature, Géométrique et Métiers (Roucoules, 1999) :

- la vue Technologique représente la décomposition structurelle du produit déduite du modèle fonctionnel du produit,
- la vue Ossature représente le produit par ses surfaces fonctionnelles,
- la vue Géométrique se construit à partir des deux vues Technologique et Ossature,
- la vue Métier représente les connaissances des acteurs métier sur le produit en créant leur propre vue.

Le modeleur CODEMO est donc une nouvelle vision de la représentation du produit où la géométrie n'est pas prépondérante et devient même plus un résultat qu'une donnée d'entrée. Le modeleur assiste les acteurs métiers dans la création des décompositions multi-vues du produit en cours de conception et présente le modèle produit de façon à ce que tous les acteurs connaissent l'état d'avancement du projet de conception. Le modeleur gère la cohérence du modèle de données (Composants, liens, relations) et de la connaissance qui lui est associée. Il s'agit aussi d'une première implémentation de modèle partagé.

2.1.4 MOKA

MOKA (Methodology and tools Oriented to Knowledge-based engineering Applications) a été développé lors du projet européen ESPRIT/AIT (MOKA, 1999a) (MOKA, 1999b). MOKA propose un modèle intégré de produit pour le développement d'applications de type KBE (Knowledge Based Engineering). Le système KBE (Knowledge Based Engineering) est une utilisation appropriée de ressources logiciels pour l'acquisition et la réutilisation de la connaissance sur un produit et un processus de manière intégrée. Une application KBE est un logiciel facilitant et automatisant des activités métiers formalisées par l'observation d'activités réelles. L'utilisation des systèmes KBE est naturellement liée à la réutilisation de la connaissance obtenue à partir de projets antérieurs.

L'ingénierie Basée sur les Connaissance (KBE : Knowledge-Based Engineering) concerne l'automatisation des processus industriels. L'exploitation du capital lié à la connaissance de conception a montré des gains importants dans le coût et les délais de conception de nouveaux produits. Dans ce contexte, le projet MOKA propose une méthodologie de formalisation de la connaissance métier. Le langage de modélisation MOKA (MML), spécialisation d'UML, est conçu pour représenter la connaissance de conception à un niveau de spécification logiciel et déploiement dans des applications KBE.

MOKA utilise deux niveaux de représentation, le premier pour la capitalisation de connaissances appelé improprement modèle informel, le second

orienté développement d'applications appelé modèle formel. Le modèle informel doit fournir la représentation de base de connaissances métier exprimées par un dialogue entre expert métier et cogniticien. Il s'agit d'un premier pas dans le processus de gestion de la connaissance.

Le modèle dit informel est donc une collection des éléments de connaissance dans des fiches appelées ICARE (Illustrations, Contraintes, Activités, Règles, Entités). Les fiches ICARE représentées dans la figure 2.4 sont structurées en unités de connaissances :

- les «illustrations» pour des commentaires et des explications complexes. Elles sont employées seulement pour des conseils et des commentaires, mais ne sont pas transformées en modèle formel,
- les «contraintes» représentent toutes sortes de liens entre entités. Elles expriment des règles métiers,
- les «activités» sont employées pour modéliser les étapes de résolution des problèmes,
- les «règles» représentent le contrôle de la connaissance,
- les «entités» sont les éléments de la connaissance qui décrivent les objets (réel et intangible, générique et concret) dans le domaine.

Le modèle formel est lui décomposé en un modèle produit et un modèle processus. Le modèle produit MOKA représenté dans la figure 2.5 est composé de cinq vues :

- la structure définit la décomposition physique hiérarchique d'un produit en assemblages, pièces et features,
- la fonction définit la décomposition fonctionnelle du produit et les principes de solution correspondants,
- le comportement inclut un modèle d'état des divers états d'un produit et de la transition d'un état à l'autre,
- la technologie inclut l'information de matériaux et de processus de fabrication,
- la représentation inclut n'importe quelle autre information technologique définie par l'utilisateur, y compris les représentations alternatives de la structure physique.

La méthodologie MOKA est un processus en trois étapes :

- formalisation des connaissances métiers via le modèle dit informel par discussions entre le spécialiste et le cogniticien dans des fiches ICARE,

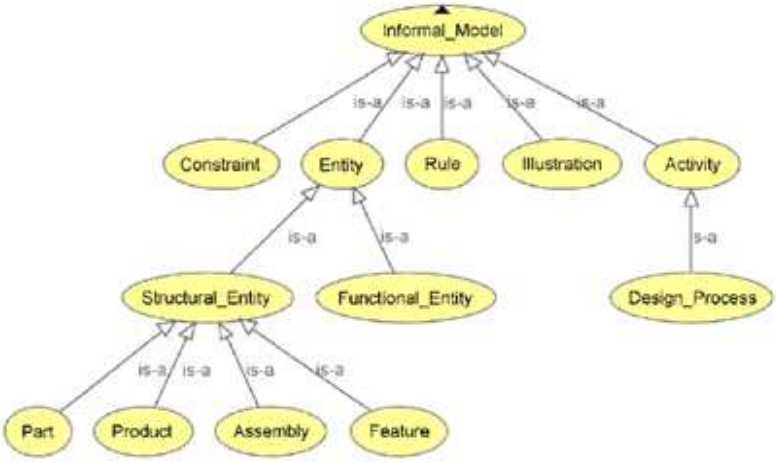

Fig. 2.4 – La structure du modèle informel de MOKA d'après (Skarka, 2007)

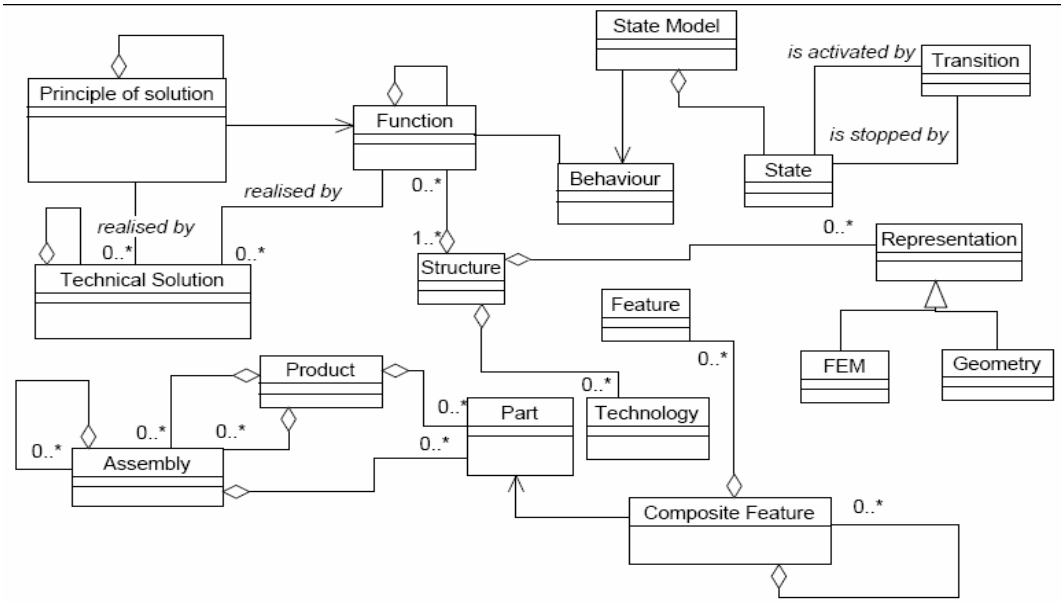

Fig. 2.5 – Le méta-modèle produit de MOKA (Stokes, 2001)

– traduction des fiches ICARE en MML,

– implémentation informatique.

Les fiches ICARE et MML sont des modèles dont les concepts peuvent être requis pour former un modèle de partage entre concepteurs.

2.1.5 PPO

Le modèle PPO (Produit Processus Organisation) est issu du projet IPPOP (IPPOP, 2008a). En réalité le modèle produit d'IPPOP véhicule une forme de concepts du modèle FBS. PPO est un modèle léger (avec peu de concepts) et évolutif. C'est le résultat de l'intégration de trois modèles : un modèle produit, un modèle processus et un modèle organisation comme le montre la figure2.6 :

1. le modèle Produit (sur lequel nous nous concentrons) est constitué de quatre concepts (Noël *et al.*, 2005) :
   – un composant permet de donner une représentation structurelle du produit. Il peut être divisé en sous-composants pour présenter un assemblage. Toutes ces concepts héritent du concept *Modelled Entity* caractérisant des entités modélisables dans des environnement multiples,

   – une interface est le lien entre un composant et une interface d'un autre composant ou une fonction,

   – une fonction représente le lien entre interfaces de composants,

   – un comportement définit l'état du modèle dans les différentes phases du cycle de vie du produit.

2. Le modèle de Processus est constitué de deux classes : la classe Projet et la classe Ressource qui se décline via une classe Acteur de processus, une classe Matériel, une classe Logiciel et une classe Information.

3. Le modèle d'Organisation définit les notions de Centre de Décision, Cadre de Décision et de Cadre de Conception.

Le modèle produit PPO est adapté au partage d'informations pendant les phases amont de la conception où la structure du produit n'est pas figée.

2.1.6 Core Product Model

Le Core Product Model (CPM), initialement développé au NIST (Fenves, 2001) (Fenves *et al.*, 2006) (Foufou *et al.*, 2005), est un support pour la mo-

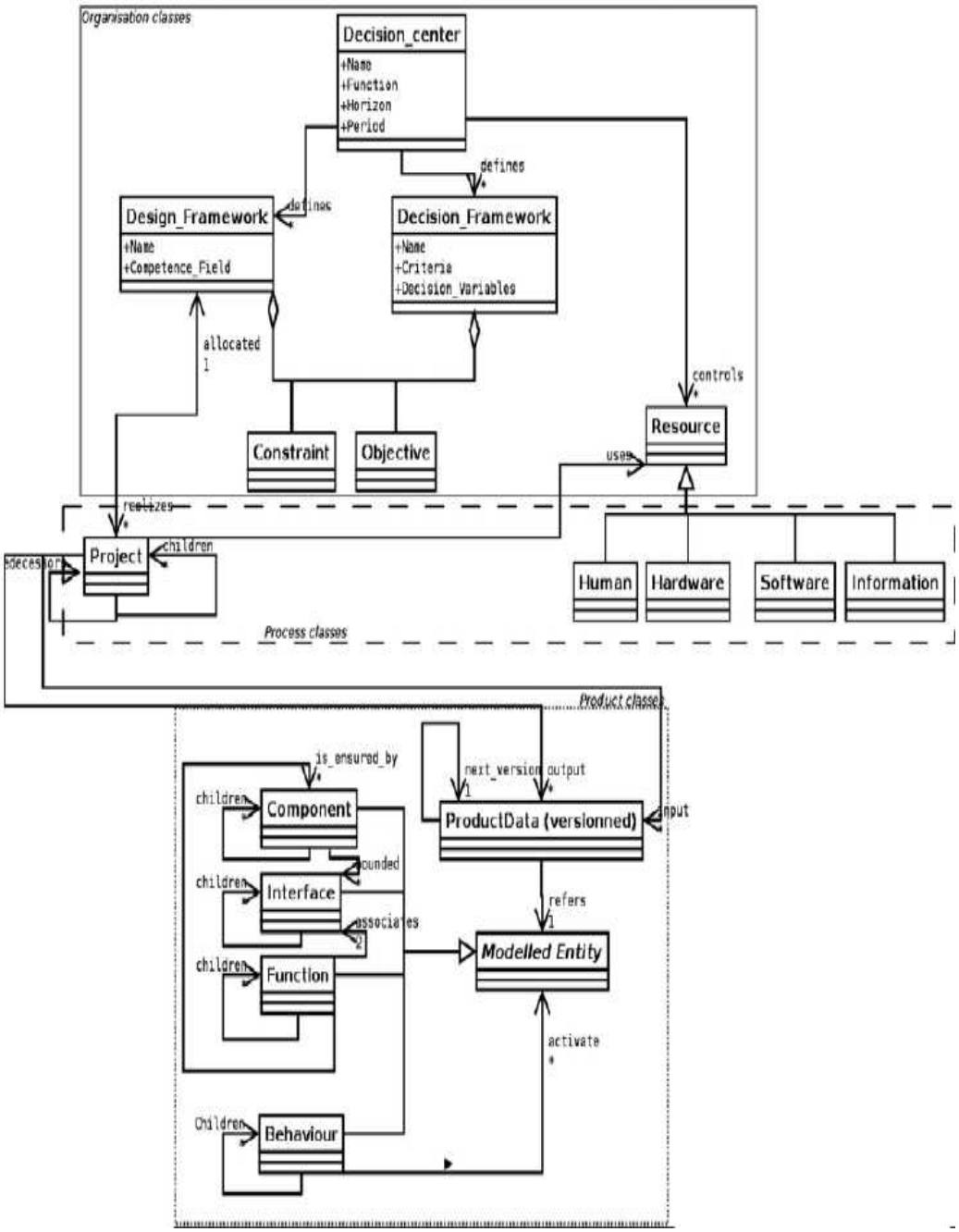

FIG. 2.6 – Le modèle produit PPO (Noël, 2006)

délisation de l'information produit. L'objectif premier du Core Product Model (CPM) est de fournir un niveau de base de modèle produit qui est ouvert, non-propriétaire, générique, extensible, indépendant de n'importe quel procédé de développement de produit et capable de capturer tout le contexte technologique généralement partagé dans le développement de produit.

Le CPM s'appuie sur quatre concepts de base :

1. sa fonction (ce qu'il doit faire),
2. sa forme (sa structure géométrique, description spatiale),
3. ses caractéristiques physiques (matériaux),
4. et son comportement (comment il met en application sa fonction). Il est ici plus abouti que les autres modèles présentés.

CPM est un modèle générique, à sémantique abstraite définie par un ensemble de concepts clés pour assurer l'interopérabilité avec d'autres modèles. Il est formalisé par un diagramme UML (figure 2.7). Dans ce diagramme on distingue trois niveaux de modélisation : un niveau conceptuel, un niveau intermédiaire, et un niveau implémentation. Le modèle CPM est un modèle conceptuel non dédié à un domaine spécifique. Ainsi, le CPM se limite aux attributs exigés pour capturer l'information générique d'un produit et pour créer des relations entre ses instances de classes. Le CPM exclut intentionnellement les attributs qui sont spécifiques à un domaine (des attributs des features mécaniques ou électroniques) ou spécifiques à un objet (des attributs spécifiques à la fonction, à la forme ou au comportement).

Le modèle intermédiaire a pour objectif de rendre le CPM directement utilisable. D'aprés (Fenves *et al.*, 2006), deux concepts génériques ont été adoptés afin de pouvoir créer les modèles intermédiaires :

- Les objets et relations sont des containers «information». Dans ces containers, l'attribut «propriétés» enregistre les paires d'attribut-valeur spécifiques du domaine ou de l'objet,
- Les objets et relations ont un attribut appelé «type», dont la valeur est une chaîne de caractères qui agit en tant que classifier symbolique spécifiant sa catégorie.

En utilisant ces deux concepts de modélisation, un artefact du type «pin» avec des valeurs d'attribut spécifiques de longueur et de diamètre est représenté dans le modèle intermédiaire suivant (Figure 2.8).

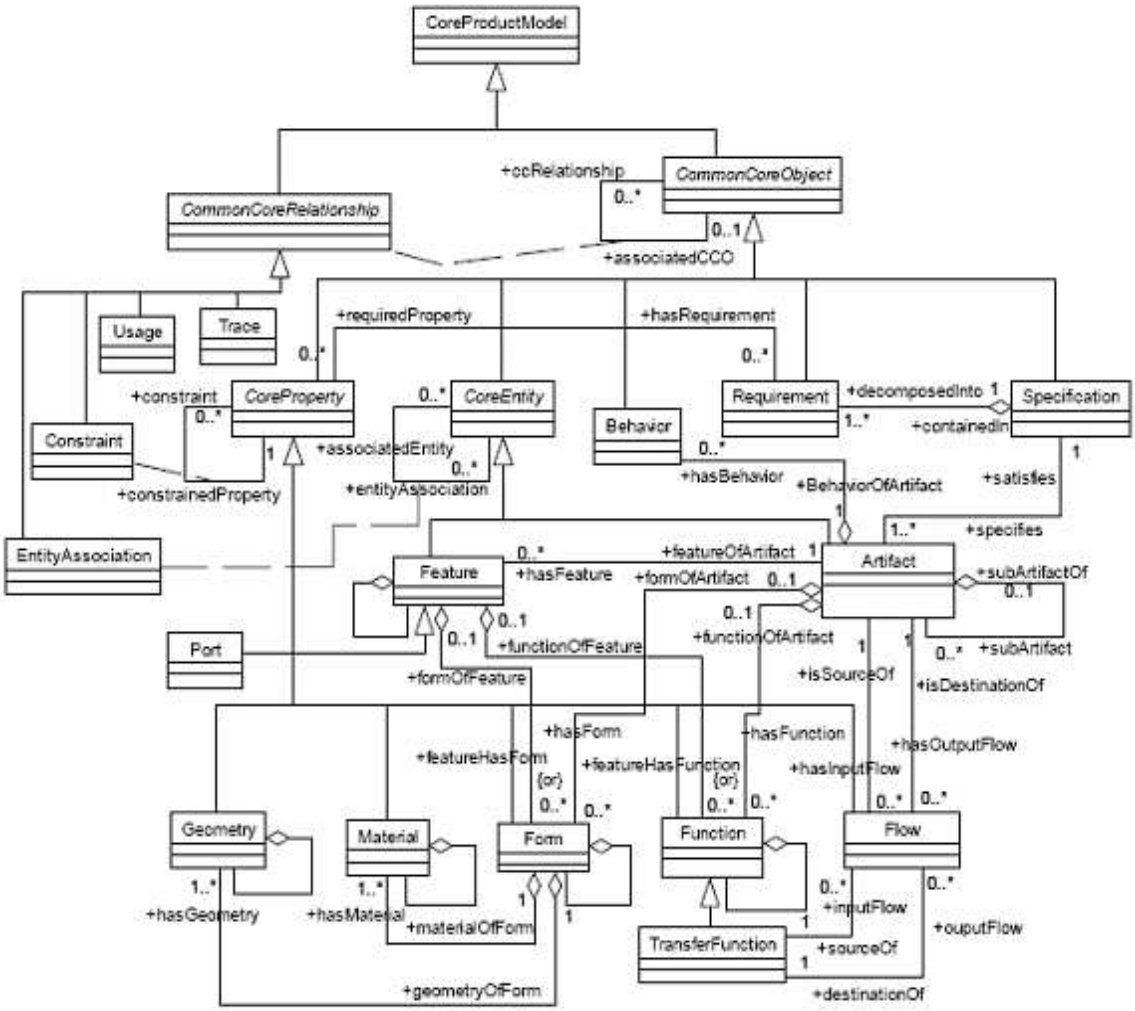

Fig. 2.7 – Un diagramme de classe UML du modèle CPM d'aprés (Fenves *et al.*, 2006)

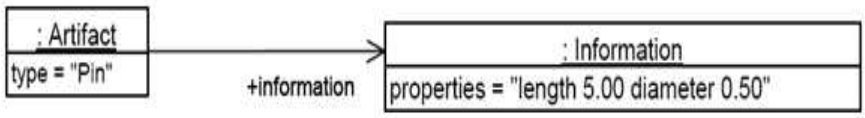

Fig. 2.8 – Une instance du modèle CPM avec les attributs et les valeurs d'aprés (Fenves *et al.*, 2006)

Une représentation en modèle intermédiaire sera généralement appropriée et suffisante pour les phases conceptuelles amont de la conception, où en général il y a un nombre restreint d'instances avec peu d'attributs. Cependant, ce modèle intermédiaire ne peut pas représenter une implémentation ou une exécution, où beaucoup d'objets sont instanciés, chacun ayant une longue liste d'attributs spécifiques à chaque application.

Le modèle du CPM doit être traduit en modèle d'exécution afin de l'appliquer à l'échelle industrielle. Pour ce faire, (Fenves, 2001) applique l'architecture dirigée par les Modèles (MDA : Model Driven Architecture) définie par l'OMG (OMG, 2003) et plus précisément le concept de traduction de modèles qui est au coeur de la MDA. Le modèle d'implémentation est obtenu donc par la traduction des Modèles Indépendants de Plateforme (PIMs), comme le CPM, aux Modèles Spécifiques Plateforme (PSMs). Cette traduction permet la génération des langages d'exécution efficaces qui spécifient le modèle générique initial.

L'architecture dirigée par les modèles, ses principes et ses apports seront étudiés en détails dans le chapitre 3. Des extensions ont été ajoutées au modèle CPM afin de fournir l'interopérabilité avec d'autres modèles tels que STEP.

2.1.7 SGDT

Pour supporter informatiquement une Gestion de données Techniques (GDT), un système de gestion de base de données est nécessaire. Un Système de Gestion de Données Techniques (SGDT) est un ensemble d'outils informatiques pour la gestion des données techniques liées à un projet de conception. Ces outils ont pour objectif de remplir les fonctions suivantes : stocker, gérer et contrôler toutes les informations et processus concernant la définition, la production et la maintenance d'un produit. L'acronyme correspondant en anglais est PDM, pour Product Data Management. Parmi les principaux outils SGDT du marché on trouve l'outil PDMLink de PTC, Teamcenter 2007 de UGS-Siemens, MatrixOne, Enovia de Dassault System et Advitium de Lascom.

Les SGDT ont été développés dans les années 80 pour faciliter la gestion d'information de façon à ce que chacun dispose de données à jour pour effectuer son travail. Un SGDT permet la gestion des informations de l'entreprise en intégrant des systèmes hétérogènes et en réalisant une gestion des données relatives aux produits. Un SGDT permet la gestion des nomenclatures, des ordres de modification et classification en vue de la réutilisation des pièces existantes. En un mot, un SGDT fédère les informations et les services dans le

cadre d'un travail collaboratif, éventuellement dans le cadre d'une entreprise étendue.

Le modèle produit partagé dans un SGDT se limite le plus souvent à une hiérarchie d'articles, c'est-à-dire une décomposition structurelle. Il est alors assez naturel d'y partager une vue unique du produit sous forme d'arbre d'assemblage/sous-assemblage/pièce. Les informations attachées à chaque article sont essentiellement des fichiers ce qui limite la granularité des informations manipulées.

Diverses descriptions des fonctionnalités existantes ou qui devraient exister dans les SGDT sont présentées dans la littérature. Nous récapitulons dans ce qui suit les principales fonctions associées à un SGDT :

- structurer les données : une description des données techniques par les attributs (d'objets et de liens) et une structuration de ces attributs offre plusieurs possibilités de tri et de recherche sur les données,
- gérer l'évolution des données : pour garder trace des diverses modifications apportées aux informations produit, divers statuts (créée, soumise, libérée, validée, etc) sont affectés à ces informations pour distinguer les diverses versions ou révisions,
- classer les données : la description des données par les attributs permet leur classification de manière simple (ex : familles, groupes) et la gestion de configuration,
- visualiser et stocker les données : le SGDT doit intégrer des outils pour la visualisation (viewer) de l'information produit comme les plans CAO. Il doit aussi assurer l'import-export de certains fichiers hétérogènes.
- protéger les données : par un contrôle des modifications et des accès (notion de sécurité des données techniques),
- distribuer les données : un SGDT doit assurer la distribution des données sur plusieurs sites via des bases de données réparties sur des sites distants et hétérogènes,
- gérer des processus de conception : les workflow permettent de connaître l'état et la position de tout élément d'un dossier dans l'entreprise. Il organise et orchestre la gestion des tâches à lancer en fonction des événements,
- assurer la réutilisation des composants existants.

Les SGDT sont principalement utilisés par les entreprises pour la gestion de leurs processus de conception et pour le partage des données entre les diffé-

rents services pendant la phase de conception du produit. Pendant cette phase le SGDT orchestre le travail de chacun et partage les tâches entre les spécialistes. Ensuite, il gère les modifications des données tout en mémorisant la composition des produits et en assurant la traçabilité de leur fabrication. Le partage de tâches fait suite à l'intégration dans les SGDT de modèles de gestion de projets.

Autres solutions apparentées aux SGDT, les logiciels de Gestion Electronique de Documents Techniques, les GEDT. Ils permettent la gestion électronique de documents techniques. Un SGDT est un GEDT intégrant un modèle de produit. Le SGDT favorise le travail collaboratif en assurant un transit efficace des informations grâce à des technologies standard. Le système autorise les utilisateurs, en fonction de leurs droits à consulter, modifier, déplacer les documents. La mise en place d'un SGDT a la réputation d'être particulièrement lourde. Les solutions commerciales proposées conduisent à des structures produit relativement figées, qui sont plus adaptées à des conceptions routinières qu'à des conceptions innovantes.

2.1.8 Synthèse des modèles de conception

Dans cette section nous avons présenté un ensemble de modèles produit et les concepts majeurs sur lesquels ils se basent. Ces modèles produit sont manipulés pour décrire un produit physique lors de l'activité de conception. Un produit peut être une pièce mécanique modélisée géométriquement par les modeleurs de conception et technologiquement par un modèle FBS ou PPO par exemple. Ces modèles techniques se basent sur une description structurelle, fonctionnelle et comportementale du produit et confirment le paradigme du modèle FBS. Le but de la modélisation produit est de pouvoir échanger et partager de l'information autour du produit. Toutefois en pratique, la méthode d'échange de données via des standards d'échanges demeure la plus couramment utilisée au dépend de ces modèles.

2.2 Standard d'échanges

2.2.1 Développement de standards d'échanges

Le besoin d'interopérabilité apparaît dés lors que les acteurs métiers ont besoin d'échanger de l'information autour du produit en cours de conception. Pour que ces modèles soient échangeables et partageables, la recherche d'un format commun pour les données XAO est un souci récurrent des experts et des industriels depuis les années 70.

Les premiers travaux ont abouti à la norme américaine IGES (Initial Graphics Exchange Specification) (IGES, 1980). IGES est apparu en 1980. Ce format d'échange est directement importable dans les logiciels de CAO 3D. Toutes les informations CAO ne sont pas pour autant transcrites avec IGES. La notion de feature par exemple est absente du formalisme IGES.

En concurrence la norme VDA-FS «Verband der Automobilindustrie Flaechen-Schnittstelle -Grafikstandard» (VDA-FS, 1986) pour l'industrie automobile a été développée. C'est un format d'échange de données 3D utilisé pour la CAO. Il a été conçu par l'union de l'industrie automobile allemande (VDA).

Au milieu des années 80, le standard français SET (Standard d'Echange et de Transfert) est impulsé (SET, 1989) par l'Aérospatiale avec l'objectif de faire mieux que IGES ou VDA. Le standard SET permet l'échange de données entre différents systèmes de CFAO et l'archivage de ces données.

Le standard DXF, sigle de Drawing eXchange Format, est un format créé par la société AUTODESK pour l'échange des fichiers DAO ou CAO entre systèmes CAO. Il a été conçu à l'origine pour sauvegarder les modélisations de DAO (Dessin Assisté par Ordinateur) d'AUTOCAD dans des fichiers ASCII. Il contient de l'information pour la visualisation des données graphiques et est supporté par la plupart des logiciels graphiques.

Au début des années 1990, est lancé le développement de STEP. STEP (STandart of Exchange of Product Model Data) apparait comme une étape majeure dans tous ces efforts normatifs sous l'égide de l'ISO (norme ISO 10303). L'objectif est de trouver une représentation simple des données du produit pour permettre l'échange de données de produit. Les échanges sont réalisés entre différents systèmes et environnements informatiques associés à l'intégralité du cycle de vie du produit comprenant la conception, la fabrication, l'utilisation, la maintenance et éventuellement la destruction finale du produit. La section suivante présente plus en détail le standard d'échange STEP.

2.2.2 Norme internationale STEP

Pour fédérer ces différentes initiatives et disposer enfin d'un langage neutre d'échange de données entre systèmes de CFAO, l'Organisation Internationale de Standardisation (ISO) a lancé le projet de norme PDES/STEP pour (Product Data Exchange Specification/STandard for the Exchange of Product model). L'objectif était de regrouper dans une norme tous les avantages des standards nationaux (IGES, SET et VDA-FS) tout en palliant les inconvénients et limitations de ces derniers. Elle a été conçue pour être extensible et réutilisable.

Ce standard permet donc de définir une représentation non ambiguë des données liées à l'ingénierie et aux produits, dans un format neutre, interprétable par tout système informatique, et couvrant tout le cycle de vie des produits sans perdre cependant l'intégrité des données (Pierra, 2000).

La norme STEP (ISO10303-1, 1994) porte sur la représentation et l'échange de données de produits et a pour objectif d'intégrer les processus de conception, de développement, de fabrication et de maintenance de ces derniers. Pour atteindre son objectif, STEP s'est dotée d'une part, d'un socle technologique de description des données via le langage de modélisation des données EXPRESS (ISO10303-11, 1994), et d'autre part d'une méthode de mise en oeuvre, le format d'échange de données neutre STEP (ISO10303-21, 1994).

La norme STEP a pour objet de produire des modèles de données standards par métier. STEP permet l'échange des tables de nomenclatures, l'historique des modifications, l'ensemble des décompositions du produit en versions multiples, et des niveaux d'autorisation. On a donc un lien fort entre la description géométrique du produit, le contrôle de sa configuration, et la gestion de ses données techniques. Les évolutions récentes de cette technologie tendent vers un rapprochement avec les outils de l'OMG, avec la standardisation de passerelles conceptuelles et techniques entre EXPRESS-STEP et UML-XMI (Plantec et Ribaud, 2005).

Le langage EXPRESS

Dans le cadre du projet STEP, le langage EXPRESS a été défini initialement pour définir des modèles de données dans les domaines techniques en vue de l'échange de données représentant de façon fiable et non ambiguë ces informations (Pierra, 2000) (Mimoune *et al.*, 2001).

EXPRESS est une représentation simplifiée des informations propres à un domaine permettant la modélisation des données, et facilitant l'échange entre concepteurs. Grâce à la représentation simplifiée (graphique) des données, les concepteurs peuvent décider des éléments qui sont pertinents et des détails qu'il convient de négliger.

Bien qu'initialement dédié à la représentation des modèles de données dans les domaines techniques, EXPRESS est à présent largement utilisé pour la modélisation des données dans différents domaines. (Ameur *et al.*, 2000) ont utilisés un modèle de données EXPRESS enrichi par l'ajout d'attributs dérivés permettant l'échange de n'importe quelle instance de n'importe quel schéma

AP 201	dessin technique explicite
AP 202	dessin technique associatif
AP 203	conception mécanique 3D avec gestion de configuration
AP 207	conception des outils et gammes pour l'emboutissage
AP 208	processus de modification du cycle de vie d'un produit
AP 209	analyse par éléments finis de structures métalliques et composites
AP 213	plans de procédés de contrôle numérique pour gammes d'usinage
AP 214	données pour la conception mécanique d'automobiles
AP 219	gammes pour machines à mesurer
AP 222	conception et fabrication de structures composites
AP 223	conception et fabrication de pièces moulées
AP 224	formes fonctionnelles utilisées lors de l'élaboration de gammes de fabrication

Tab. 2.1 – Quelques protocoles d'applications

de base de données. Ceci montre l'intérêt du langage EXPRESS pour la spécification de plusieurs applications dans le domaine de développements informatiques.

EXPRESS contient un format d'échange d'instances des modèles de données sous forme d'un fichier de caractères. Un formalisme EXPRESS-G (ISO10303-11, 1994) en parallèle de la représentation textuelle est apparu pour simplifier la présentation graphique des modèles et donner une vue synthétique des modèles de données surtout en phase de conception. La représentation textuelle des schémas EXPRESS est essentielle pour le traitement automatique des modèles par des machines pour vérification ou correction.

La norme STEP définit un protocole d'application qui décrit les informations pertinentes dans un domaine technique donné ainsi que la structuration de ces données. STEP contient plusieurs Protocoles d'Application (AP), décrits en langage EXPRESS. STEP propose en particulier les AP présentés dans le tableau 2.1.

2.2.3 STEP-NC

Nous nous sommes intéressés à STEP-NC car le cas d'étude que nous utilisons dans ce document concerne un lien entre outils de CAO et de FAO. Pour réaliser l'échange d'information et de connaissance entre les systèmes de CAO/FAO et les machines-outils à commandes numériques, une méthode d'intégration de l'information basée sur le processus de modèle de données et le

format neutre de STEP-NC a été développée dans le cadre de la normalisation ISO : c'est l'interface ISO 14649. STEP, développé depuis les années 80, est la norme pour échanger des données de produit entre les systèmes de CAO et de FAO. Pour le flux de données entres la FAO et les MOCN (les Machines-Outils à Commandes Numériques), STEP-NC (normalisé ISO 14649) a été nouvellement établie comme norme internationale.

Elle vise à remédier aux défauts de l'ISO 6983 (nommé M & G-codes) ainsi qu'à tirer profit de la puissance de calcul des contrôleurs modernes. STEP-NC combinée avec les technologies modernes de communication fournit des moyens efficaces pour développer l'intégration et l'interopérabilité des données des systèmes de fabrication.

Comparé à l'ancienne norme ISO 6983 actuellement employée comme interface entre la FAO et la MOCN, STEP-NC inclut des informations comprenant la géométrie et le plan de processus. STEP-NC (ISO 14649) se compose de : (1) la description des tâches, (2) la description de technologie, (3) la description d'outil, et (4) la description de la géométrie. La description de tâches décrit l'ordre logique des tâches exécutables ainsi que les types de données. (Suh *et al.*, 2006) présente un système pour le tournage (appelé TurnSTEP). Le système présenté est basé sur le modèle de données de STEP-NC. Bien que TurnSTEP ne soit qu'un prototype, il démontre un potentiel certain.

L'interface STEP-NC (ISO 14649) permet de spécifier «COMMENT» usiner (figure2.9b) par la description des séquences d'usinage en définissant une hiérarchie des éléments d'usinage tels que «Project», «Workplan», «Workingsteps», etc ainsi que les paramètres technologiques des opérations d'usinage. Il permet de spécifier le «QUOI» usiner (figure2.9a) en définissant les objets d'usinage (*features*) tels que les trous, les plans, le profils, les poches, etc, en incluant aussi des critères de qualité (tolérances, états de surface, etc.).

L'interface STEP-NC n'est pas largement diffusée et exploitée au niveau commercial. La norme ISO 14649 est toujours en projet. Des travaux autour de STEP-NC (Campos et Hardwick, 2006) ont essayé de construire un modèle d'information pour la traçabilité des machines-outils à commandes numériques en fabrication. Ce travail a présenté un modèle d'information pour tracer les processus de fabrication et a montré comment ce modèle peut être intégré avec d'autres modèles de données de produit CAO/FAO (figure2.10).

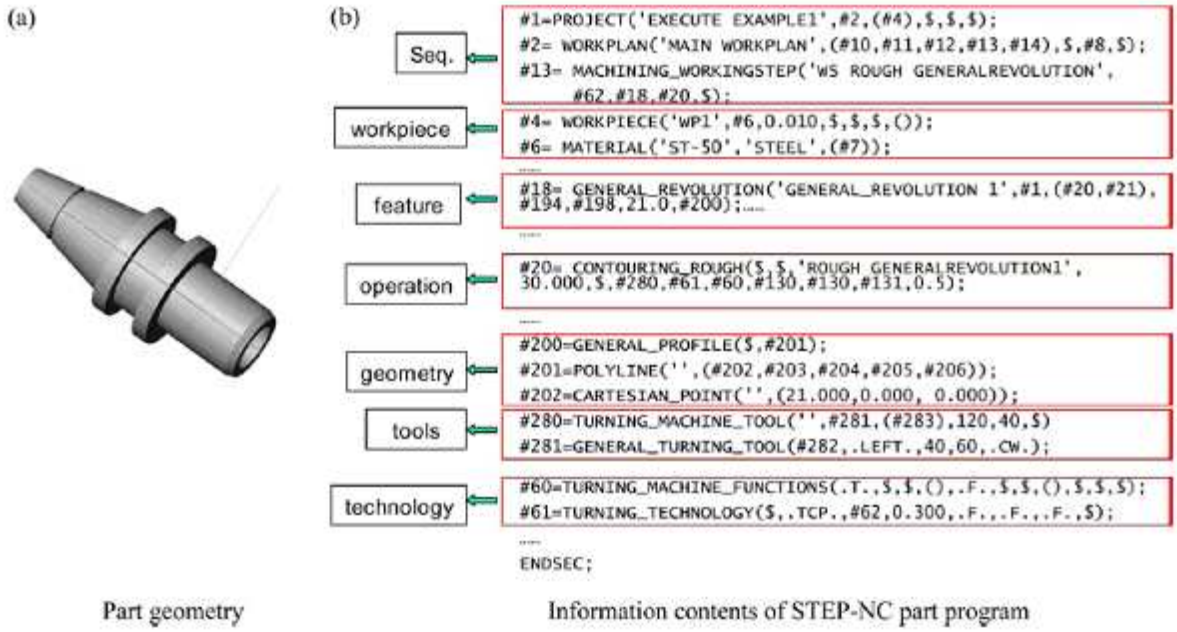

Fig. 2.9 – Un exemple de programme STEP-NC d'une pièce géométrique d'après (Suh *et al.*, 2006)

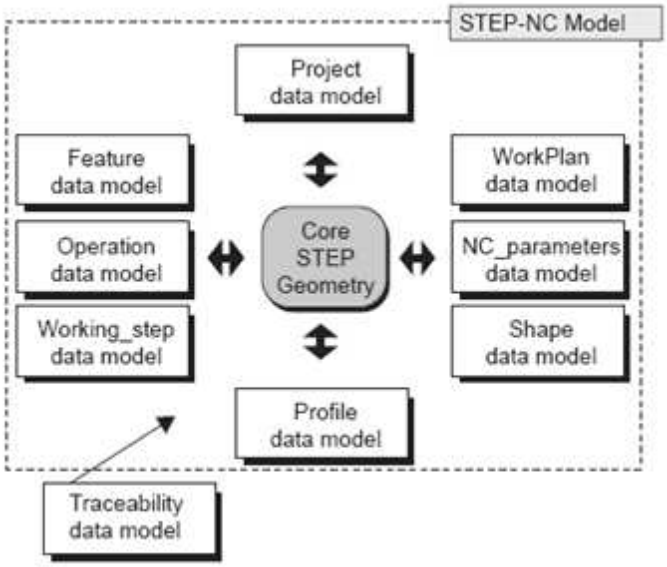

Fig. 2.10 – Modèle de Traçabilité dans STEP NC d'après (Campos et Hardwick, 2006)

Travaux autour de STEP

Des travaux ont aussi été réalisés pour le diagnostic de qualité des données de forme de modèle produit basé sur STEP pour la conception (Tanaka et Kishinami, 2006). Une méthode de diagnostic de qualité de données de forme d'un fichier neutre basé sur un modèle de produit comprenant la surface incurvée décrite en langage formel, est proposée. La méthode proposée est basée sur la norme STEP pour l'échange de données de produit en utilisant EXPRESS, qui est le langage de description des modèles de données STEP. Suite à ces travaux, une classification des critères de qualité basés sur leurs algorithmes géométriques est proposée afin de créer facilement les critères de diagnostic de qualité de données pour des erreurs géométriques.

(Leea *et al.*, 2003) proposent le développement d'un système de collaboration à distance de retro-ingénierie (reverse engineering) pour un environnement concourant de développement de produit et de processus. Cette approche utilise la norme internationale STEP enrichie par le développement d'un module d'enregistrement de l'information produit. Dans ce travail, un module d'enregistrement de l'information de produit pour la retro- ingénierie a été développé pour améliorer le développement de produit. Ce module peut être utilisé pour enregistrer l'information principale pendant un processus de collaboration, et peut également être employé pour d'autres échanges d'information entre systèmes hétérogènes.

Les systèmes de CAO actuels produisent des modèles basés sur des caractéristiques (feature-based) de forme de produit incluant des techniques de paramétrage et de modélisation sous contraintes. Jusqu'à présent, l'échange de données de CAO entre différents systèmes CAO a été limité à l'échange d'information géométrique. Les standards ont ignoré l'historique de conception, les paramètres, les contraintes, les caractéristiques (features) et d'autres éléments représentant l'intention de conception (design intent) réellement présents dans le modèle à transférer.

(Kim *et al.*, 2007) suggèrent une base d'implémentation pour l'échange de données CAO en conservant l'arbre de construction, basée sur l'utilisation de parties nouvellement publiées. Des études de cas qui utilisent un protocole d'application STEP employant les parties 55, 108 et 111 d'ISO 10303 ont été réalisées. Un prototype de traduction basé sur ce protocole d'application a été mis en oeuvre et examiné.

Trois nouvelles parties récemment publiées sont caractéristiques de l'avancement de STEP :

– ISO 10303-55, «représentation procédurale et hybride» fournit pour le transfert d'information de l'historique de conception,
– ISO 10303-108, «paramétrisation et contraintes pour les modèles géométriques explicites de produit» rend possible la capture et le transfert des paramètres et des contraintes, et la représentation des esquisses 2D,
– ISO 10303-111, «éléments pour la modélisation procédurale des formes de solide» fournit des représentations pour les caractéristiques de conception (design features).

Ces travaux présentent des bases pour l'échange d'historiques de conception de modèles de forme avec la paramétrisation et les contraintes. Après la publication initiale de l'édition 1 de STEP en 1994, plusieurs années d'expérience ont été exigées avant que les échanges des modèles de BREP soient fiables. L'échange des modèles procéduraux ou d'historique de conception réactive l'évolution de STEP, et une période d'étude comparable est prévisible.

De plus, l'évolution des modèles CAO ne s'arrête pas à l'évolution du modèle géométrique et STEP ne couvre donc qu'une partie des concepts modélisés dans les outils CAO.

2.2.4 Synthèse des standards d'échanges

Les standards d'échanges permettent aux concepteurs d'extraire leurs besoins en informations depuis un modèle unifié, pour construire leur propre modèle. Pour le soutien des échanges, un modèle partagé qui évolue avec la phase de conception par l'incorporation d'une manière dynamique, des modèles multi-disciplinaires d'ingénierie tout au long du cycle de vie du produit est nécessaire. Les standards d'échanges actuels restent donc inadaptés à une interopérabilité totale en conception.

2.3 Standard OMG

2.3.1 PLM Services

Les «PLM services» sont un standard développé par l'OMG pour l'échange des données produit avec des technologies de Services Web. C'est le premier standard qui propose une synthèse entre XML, les Services Web et le modèle de données de STEP. Le modèle d'information de «PLM services» est complètement conforme au «PDM Schema» (Laemmer, 2002). «PDM Schema»

est né d'un effort pour harmoniser les besoins et les modèles de données issus des protocoles d'application de STEP en offrant un modèle de données unique supportant les données gérées dans les PDM. Ce modèle devrait permettre aux concepteurs de PDM d'implanter des fonctionnalités PDM en se basant sur STEP et favoriser ainsi l'interopérabilité des PDM avec d'autres systèmes basés sur les différents protocoles d'application de STEP. Les PLM Services de l'OMG offrent une base solide pour accomplir des actions d'ingénierie collaborative : en naviguant dans des structures produit distribuées, en contextualisant les données de conception et en permettant la visualisation des données produit, etc.

Les «PLM services» définissent à la fois les données et les process attachés. Cette initiative de l'OMG est une avancée prometteuse pour réconcilier le modèle de données de STEP (ISO) très orienté technologie avec la technologie XML et les Services Web (W3C) issus du développement de l'informatique. Cependant, en se basant sur un modèle de données intégrant le «PDM Schéma», les «PLM services» manipulent des données à un niveau de granularité fin et un niveau de détail très développé des applications PLM (Product Lifecycle Management). Ceci ne satisfait pas notre problématique où les expert coopèrent sur des tâches communes en manipulant quelques paramètres spécifiques.

2.3.2 PDM Enablers

L'échange de données dans STEP n'offrait pas une solution satisfaisante aux problèmes d'utilisation coopérative d'outils CAO et SGDT. La conversion de données entraînait une perte d'information à chaque transformation. Pour pallier à ce problème, un couplage fort et une intégration plus directe sont nécessaires (Elkhalkhali, 2002) (Gzara, 2000).

«PDM Enablers» (Starzyk, 1999) est un standard OMG apparu pour supporter la gestion des données techniques, la configuration des produits, et les process de fabrication. Les informations techniques peuvent être associées à des produits spécifiques, des schémas de produits, ou plus généralement des familles de produits. La gestion de ces données se fait de la conception du produit jusqu'a sa commercialisation, et sa maintenance. Les PDM Enablers supportent l'interopérabilité des systèmes en proposant une architecture d'interfaces standards (API : Application Programming Interface) aux divers services d'un PDM opérant dans un environnement distribué.

Les PDM Enablers sont donc conçus pour répondre aux exigences suivantes :

- fournir des interfaces robustes qui permettent l'interopérabilité entre les systèmes PDM et une grande variété d'autres systèmes logiciels,
- fournir un environnement pour les interfaces de système PDM qui peuvent être aisément adaptées et entendues aux besoins du client final,

- fournir des interfaces qui sont simples et assez génériques à employer pour une large gamme d'applications avec assez de détails pour être signicatif et complet,
- rendre les services d'un PDM disponibles pour d'autres systèmes (des systèmes CAO, IAO, FAO et également d'autres PDM) à travers un environnement CORBA (Common Object Request Broker Architecture),

- fournir des interfaces directes pour spécifier la gestion des documents, la gestion des nomenclatures, la gestion des modifications et la configuration de produits (options et variantes du produit) et également l'import et l'export de fichiers d'échange STEP.

Les PDM Enablers héritent naturellement des inconvénients d'un PDM. En effet le niveau de granularité d'un PDM est insuffisant pour traiter les données dans leurs détails.

2.3.3 La norme XML

Au-delà des différentes normes officielles (soutenues par les organismes de normalisation) que nous venons de voir : IGES, VDA, SET, STEP, on trouve de nombreux formats natifs de stockage des logiciels de CAO. Les applications d'aujourd'hui ne sont plus monolithiques et doivent s'intégrer harmonieusement dans le système d'information de l'entreprise. L'objectif principal des standards du PLM (Product Lifecycle Management) est de définir un langage commun entre les différentes solutions de PLM. XML (eXtensible Markup Language) est un standard du World Wide Web Consortium (W3C) qui sert de base pour créer des langages de balisage : c'est donc un «métalangage». De par l'extension des technologies Web, XML s'est imposé pour la description de langages communs.

L'objectif de XML est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes et donc de répondre à un problème d'interopérabilité. Un document XML est un fichier texte auto descriptible. Les balises d'un document XML sont définies par l'utilisateur. Un document XML est utilisable par n'importe quelle application dotée d'un parseur. Il peut être lu sur n'importe quelle plate-forme informatique. Les données d'un document XML sont structurées sous la forme d'une arborescence. Un document XML peut

être distribué par n'importe quel protocole qui transporte du texte (comme http).

L'initiative des travaux menés par l'ISO qui consiste à rapprocher le formalisme STEP de XML a fait naître le PDML (Product Data Markup Language). Le PDML (W.C.Burkett, 2001) est un dialecte bâti sur XML conçu au début des années 2000 pour le soutien de l'échange d'informations de produit à travers tout les systèmes commerciaux (tels que les PDM). PDML compense les lacunes de STEP en adaptant EXPRESS à la philosophie du langage XML qui facilite l'échange de données en utilisant un vocabulaire et des définitions (Charles, 2005).

Cette initiative se poursuit par des travaux menés par l'ISO pour faire converger les spécifications STEP modélisées en Express avec XML et SGML. Ces travaux se sont concrétisés par le développement d'un outil appelé «FirstSTEP EXML» basé sur XML pour l'échange des données de produit. «FirstSTEP EXML» est un outil logiciel qui peut être employé pour convertir un schéma EXPRESS en son équivalent DTD de XML (PDML, 2008).

Un autre courant est issu des langages orientés objet et des méthodes de conception associées. Elles ont été unifiées quelques années plus tard, sous l'appellation UML. UML (Unified Modeling Language) est un langage graphique de modélisation des données et des traitements. UML est une norme OMG dont le rôle est de promouvoir des méthodologies en génie logiciel et des standards qui garantissent l'interopérabilité entre applications orientées objet, développées sur des réseaux hétérogènes.

2.3.4 Synthèse des standards OMG

Dans cette section, nous avons présenté les technologies développées pour favoriser les échanges d'informations. Ces technologies sont les formats neutres d'échange comme la technologie XML et les outils standards mis en oeuvre pour l'interopérabilité entre les SGDT tels que les PLM services et les PDM Enablers.

2.4 Conclusion

Dans ce chapitre, nous avons détaillé les modèles produit ainsi que les standards d'échanges comme moyen d'échange d'information autour du produit. Pour les modèles produit, un modèle reprenant les notions du paradigme

FBS nous parait satisfaisant pour la modélisation du produit en conception collaborative. Nous avons choisi de travailler avec le modèle produit PPO pour la modélisation de l'information autour du produit pendant sa phase de conception. Au niveau des standards d'échanges, nous utilisons le standard STEP pour la représentation et l'échange des modèles.

La revue des travaux réalisés autour de STEP a montré que les efforts se sont essentiellement focalisés sur l'échange et le partage des données. Rares sont les travaux qui utilisent le contenu du modèle de données STEP comme un modèle d'application (Urban *et al.*, 1999). L'insertion récente des arbres de construction démontre le retard chronique de la norme sur les outils commerciaux. L'interopérabilité ne peut être complète entre outils commerciaux récents.

De leur côté, les SGDT gèrent l'information produit pendant son cycle de vie. Les SGDT ne permettent pas une description à un niveau de granularité fin du produit. Les standards ont un niveau de détails produit très développé. Un juste milieu entre les deux paradigmes est donc nécessaire. Le besoin d'un modèle de description de données simple et évolutif qui permet de décrire dans un niveau de détail suffisant pour le partage de l'information est nécessaire. Le choix du modèle produit PPO satisfait nos besoins actuels en terme de scénario d'usage pour le partage autour du produit. Quelque soit le modèle produit choisi (PPO, CPM, etc.), des mécanismes de connexion à des représentations métiers doivent être mis en oeuvre. C'est ce que nous allons réaliser en nous appuyant sur des principes d'interopérabilité présentés dans le chapitre suivant.

Chapitre 3

Ingénierie Dirigée par les Modèles et Interopérabilité

Sommaire

3.1 Ingénierie Dirigée par les Modèles

Dans ce chapitre, nous abordons une méthodologie générique pour assurer l'interopérabilité entre outils métiers hétérogènes à travers un environnement de travail collaboratif. Le choix d'une approche d'Ingénierie Dirigée par les Modèles (IDM) afin d'assurer l'interopérabilité dans une phase de conception d'un produit manufacturier est nouvelle. Le contexte de conception de produit donne une complexité à la solution qu'on doit proposer (faire interopérer des

outils métiers hétérogènes et répartis). L'IDM a été utilisée pour la conception d'outils logiciels en offrant une démarche de génie logiciel rigoureuse et structurée. Nous voulons appliquer cette méthodologie dans le contexte de conception collaborative de produit manufacturier : ceci implique une prise en compte de la spécifité de ce contexte de développement, des types de modèles et des types de collaboration qui lui sont associés.

Le choix d'une approche de développement basée sur l'Ingénierie Dirigée par les Modèles (IDM) donne une méthodologie de travail pour réaliser l'interopérabilité : de la définition des modèles à des niveaux d'abstraction élevés jusqu'à l'établissement des règles de transformation tout en séparant les aspects métiers et d'implémentation. L'IDM ne fait aucune supposition sur les langages et les outils utilisés. L'IDM a connu une forte attention avec sa standardisation par l'OMG de la MDA (Model Driven Architecture) (OMG, 2003) qui est un cas particulier de l'IDM.

Dans cette partie, nous discutons le choix de cette approche et ses avantages. Nous définissons ses principes et ses apports quant à l'interopérabilité des outils métiers hétérogènes. Nous montrons aussi les mécanismes de transformation de modèles qui sont au coeur de l'Ingénierie Dirigée par les Modèles. Nous entrons plus en détail pour expliciter la place qu'occupent les modèles dans la mise en oeuvre de la MDA.

3.1.1 Choix de l'IDM pour l'interopérabilité

La problématique d'interopérabilité a toujours existé en conception de produits et spécialement avec des outils métiers de type CFAO. Ces outils métier n'ont pas été conçus pour interopérer mais pour travailler dans un cadre isolé où l'expert métier réalise une tâche spécifique.

Des méthodes et des outils de travail (développé dans le chapitre 2) ont été réalisés pour aider les experts métier à collaborer. Parmi eux :

- des modèles produit ont été proposés (CPM, PPO, etc.). Il faut des méthodes pour se connecter et agir sur ces modèles,
- des standards d'échanges (VDA, STEP, etc.) ont été proposés. Ces standards favorisent l'échange unidirectionnel, non dynamique mais avec une certaine perte d'information.

Notons que STEP, standard d'échange peut être interprété comme un modèle de partage (Urban *et al.*, 1999). Ces modèles de partage et standards ont été développés pour répondre aux problèmes d'interopérabilité. Il existe aujourd'hui trois approches pour réaliser l'interopérabilite entre les systèmes :

- l'approche intégrée consiste à construire un format commun pour tous les modèles afin de développer un système unique. Suite à l'action d'intégration, les deux systèmes en interaction ne forment plus qu'un seul système utilisant un modèle unique,
- l'approche unifiée consiste à conserver le modèle propre à chaque système et de définir un format commun pour assurer la communication entre eux. Chaque système conserve alors sa propre structure avant et après la communication. Les modèles produits collaboratifs et les standards se classent dans cette approche,
- l'approche fédérée consiste à construire dynamiquement des vecteurs de communication entre des systèmes qui initialement s'ignorent. Les API spécifiques font partie de cette approche puisqu'elles ne proposent pas de format commun pour la communication et nécessitent des efforts dynamiques d'ajustement et d'accompagnement.

Ces solutions pour soutenir la collaboration permettent un premier niveau de formalisation des échanges. Les standards se sont imposés et ne fournissent pas un moyen d'exprimer les différences de points de vue entre expertises métiers. La prise en compte des informations et de l'expertise métier dans la collaboration reste à développer. Pour assister les experts pendant la phase de collaboration, il faut leur fournir des moyens d'exprimer :

- les entités qu'ils veulent échanger,
- les moyens de les transformer vers une expertise cible,
- les moyens de propager et de mettre à jour les modèles en fonction des modifications éventuelles,
- le moyen de garantir la confidentialité et les savoirs faire.

Pour répondre à ces besoins, nous optons pour une approche fédérée d'outils métier qui partagent via un modèle unifié mais évolutif. Une expertise métier est un ensemble de connaissances et de savoirs faire. Les vues expertes sont parfois complémentaires. Ici naît le besoin d'une méthodologie générique indépendante des expertises : les experts doivent y spécifier les entités à partager ou à échanger et établir les correspondances entre modèles. Conserver une trace de cette correspondance afin de pouvoir propager les modifications entre modèles est aussi nécessaire.

Ces besoins invoquent les notions de modèle, de correspondance entre modèles, de transformation de modèles. Or, ces concepts sont au coeur de l'IDM. C'est ce qui justifie le choix de l'IDM. La MDA, solution particulière de l'IDM, a émergé suite à l'expansion des infrastructures informatiques et à l'émergence

des systèmes répartis qui ont fait naître un besoin d'interopérabilité entre elle (Lopes, 2005).

L'idée de base de cette méthodologie est de rendre les entreprises indépendantes de plates-formes techniques qui sont sans cesse en évolution. La modélisation de l'application se base désormais au niveau du modèle en laissant le soin de l'implémentation de bas niveau à d'autres outils.

L'hétérogénéité des collaborateurs et des outils mis en jeu lors de la collaboration augmente de manière exponentielle et amplifie le besoin d'interopérer. L'approche proposée par l'OMG (Object Management Group) pour spécifier l'interopérabilité au niveau des modèles est la MDA (Model Driven Architecture).

Avant les années 80, les besoins étaient figés et les applications étaient plus ou moins simples et avaient peu d'interaction. Les applications deviennent des familles d'applications de plus en plus complexes. Afin de répondre aux besoins des logiciels, toujours de plus en plus nombreux et consommateurs de ressources, différents paradigmes de programmation ont vu le jour. Le paradigme procédural a ainsi laissé la place au paradigme objet qui, à son tour, a laissé sa place au paradigme composant. Plusieurs méthodologies ont été proposées pour supporter le développement logiciel dans les années 80, telles que MERISE, OOD (Object Oriented Design), OMT (Object Modeling Technique) et OOSE (Object Oriented Software Engineering) (Lemesle, 2000), (Touzi, 2007). En 1997, ces méthodologies ont convergé vers le formalisme UML. Ce langage pseudo-formel est devenu une référence pour le développement orienté-objet d'un système.

3.1.2 L'approche MDA

La MDA (Model Driven Architecture) est apparue en 2000 après plusieurs années d'existence de standards de modélisation et de méta-modélisation comme UML (Unified Modeling Language). L'idée à la base de l'initiative Model-Driven Architecture (MDA) de l'OMG est qu'il devrait être possible de gérer des modèles métier hétérogènes indépendamment de toute plateforme (PIM : Platform-Independant Models) et de dériver par transformation de ces PIM des modèles spécifiques à la technologie retenue (PSM : Platform-Specific Models) et finalement du code. Mais pour certains secteurs d'activité, la valeur ajoutée d'une entreprise ne se situe pas seulement dans sa connaissance du domaine métier - le PIM- mais aussi dans l'expertise de conception et de mise en

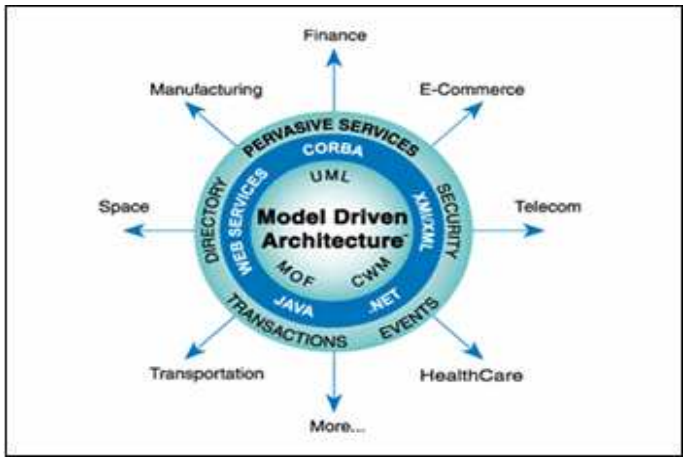

FIG. 3.1 – Le Model Driven Architecture de l'OMG

œuvre nécessaire pour faire fonctionner un système avec les contraintes d'un environnement réel -la transformation du PIM vers le PSM.

D'après (Bézivin et Gérard, 2003) la MDA fournit un processus de conception et met en oeuvre des outils pour :

- spécifier un système indépendamment de la plate-forme qui le supporte, et donc réaliser un PIM,
- enrichir ce modèle par étapes successives,
- spécifier les plate-formes,
- choisir une plateforme particulière pour le système,
- transformer la spécification du système (PIM) en une autre spécification pour une plateforme particulière (PSM),
- transformer un CIM en un PIM et un PIM en un autre PIM,
- raffiner le PSM jusqu'à obtenir une implémentation exécutable.

Les trois objectifs préliminaires de MDA sont la portabilité, l'interopérabilité et la réutilisabilité à travers une architecture de séparation des préoccupations. La figure 3.1 représente le logo de MDA et ses différentes couches de spécification : au cœur, se trouvent les techniques de base (UML, MOF, CWM), puis quelques-unes des plates-formes supportées et enfin en surface les services systèmes. A l'extérieur, sont présentés les domaines pour lesquels des composants métiers doivent être définis (Domain Facilities)(Team, 2001).

3.1.3 La Méta-modélisation

La MDA est une démarche de développement basée sur les modèles. Les modèles sont composés d'entités sur lesquelles des opérations peuvent être

appliquées de manière automatisée. La MDA est dirigée par les modèles « parce qu'elle fournit une approche de l'utilisation de modèles pour diriger la compréhension, la conception, la construction, le déploiement, l'opération, la maintenance et la modification de systèmes » (Lopes, 2005).

Il faut donc pouvoir analyser les différents éléments d'un modèle et comprendre leur structuration et leur signification.

Le MOF

Le MOF (Meta Object Facility) (OMG, 2002)(MOF, 2002) est une spécification définie par le consortium OMG qui permet de concrétiser la notion de méta-modélisation. Le MOF fournissant un langage unique pour expliciter les méta-modèles est donc un méta-méta-modèle.

Le MOF est un langage abstrait représentant un vocabulaire commun pour la définition de tous les méta-modèles. De ce fait, il permet la compréhension, la comparaison et l'interopérabilité des méta-modèles et donc des divers modèles. Ci-dessous, nous présentons les quatre concepts majeurs du MOF :

1. une classe permet de définir un concept au sein d'un méta-modèle. Une classe peut contenir des attributs et des opérations,

2. une association permet de définir une relation entre deux classes, c'est-à-dire entre deux concepts d'un méta-modèle,

3. les types de données, permettent de modéliser les autres données (types primitifs, complexes, etc.),

4. un package permet de regrouper les classes et les associations. Un méta-modèle est toujours défini par un package regroupant tous ses concepts et leurs relations.

Le MOF est représenté entièrement par un diagramme de classe UML (figure 3.2).

Les niveaux d'abstraction

Une étape majeure a été franchie avec l'adoption par l'OMG (Object Management Group) du MOF en 1997. Le MOF se retrouve au sommet d'une architecture à quatre niveaux d'abstraction pour la modélisation des systèmes (figure 3.3). Cette architecture, autour de laquelle s'est aujourd'hui formé un consensus, est composée des niveaux suivants : méta-méta-modèle, méta-modèle, modèle et information. Dans cette architecture, définie par l'OMG,

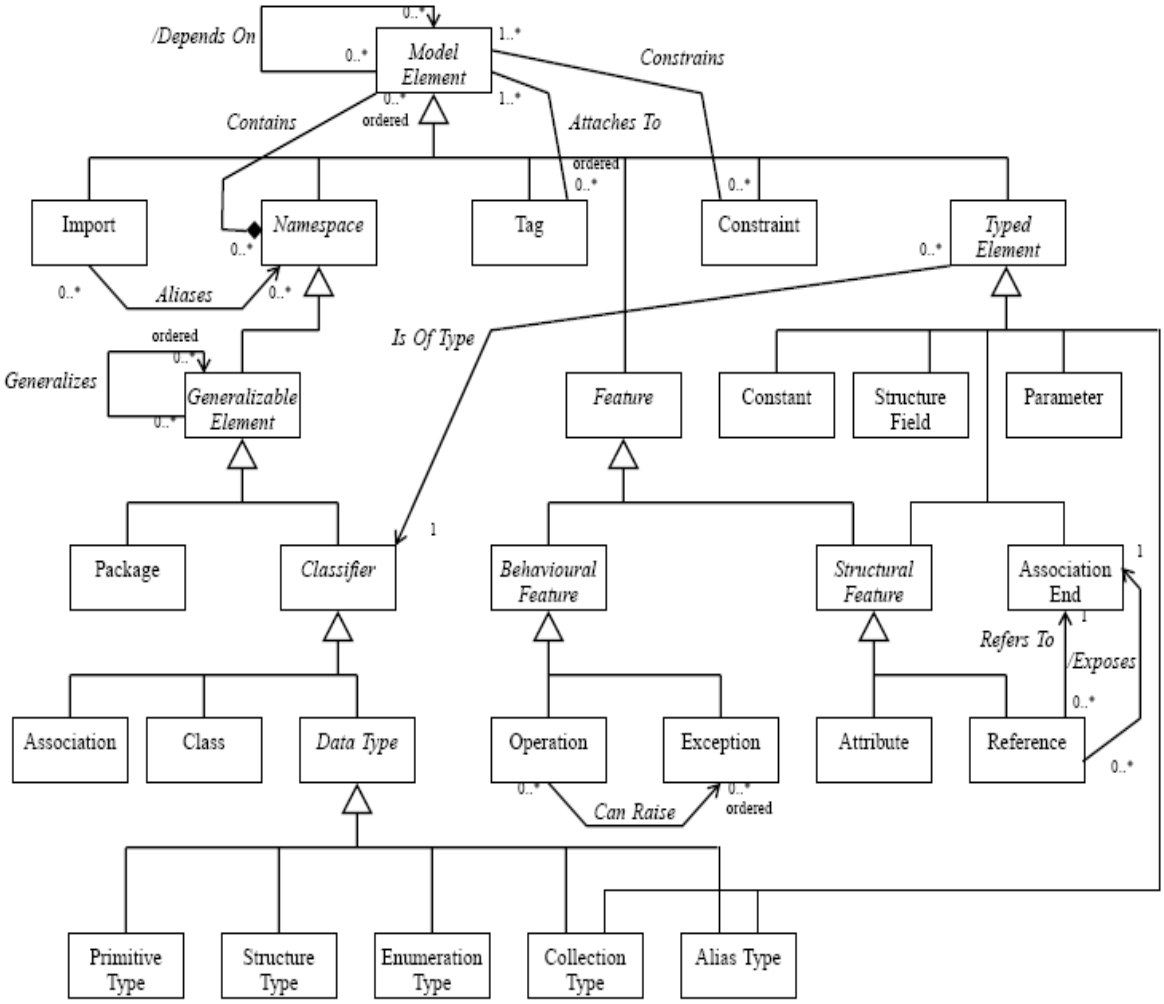

FIG. 3.2 – Le MOF version 1.4 d'aprés (OMG, 2002)

chaque niveau entretient une relation d'instanciation avec le niveau supérieur. Selon (Djebbi et Gervais, 2004), ces niveaux sont les suivants :

- au niveau M3 se trouve le langage unique de définition des méta-modèles, appelé aussi le méta-méta-modèle : en l'occurrence le MOF. Le MOF définit la structure de tous les méta-modèles qui se trouvent au niveau M2. Le MOF est réflexif, c'est-à-dire que MOF s'auto-décrit, ce qui permet de dire que le niveau M3 est le dernier niveau de la hiérarchie. Le niveau M3 correspond donc aux fonctionnalités universelles de modélisation logicielle, alors que le niveau M2 correspond aux aspects spécifiques des différentes familles, chaque aspect étant pris en compte par un méta-modèle spécifique,
- au niveau M2 se trouve les langages de modélisations ou méta-modèles : par exemple les définitions d'un MCD MERISE ou d'un diagramme d'objet UML. Notamment, le méta-modèle UML décrit dans le standard UML appartient au niveau M2. Il définit la structure interne des modèles UML. Les profils UML permettant d'étendre le méta-modèle UML, sont aussi considérés comme appartenant au niveau M2,
- au niveau M1 se trouve le modèle qui décrit certains aspects du niveau M0 que l'on veut étudier : il peut s'agir d'un diagramme de classes UML, d'un modèle conceptuel de traitement MERISE, ou de tout schéma qui représente une vue abstraite des objets modélisés. Par rapport au MDA, les PIMs et les PSMs appartiennent à ce niveau. Les modèles du niveau M1 de même famille sont exprimés dans un langage unique dont la définition est fournie explicitement au niveau M2,
- au niveau M0 se trouve le système étudié par l'utilisateur final : il peut s'agir selon le cas du système d'information d'une entreprise, ou des tâches que l'on souhaite automatiser. Le niveau M0 contient l'ensemble des informations du monde réel que l'on souhaite modéliser.

3.1.4 Les types de modèles dans MDA

Le modèle d'un système est la spécification formelle des fonctions, de la structure et/ou du comportement de ce système dans son environnement adapté à un certain contexte (Terrasse *et al.*, 2003). Un modèle est souvent représenté par des schémas et du texte. Le texte peut être exprimé dans un langage de modélisation ou en langage naturel. Dans le processus MDA, tout est considéré comme modèle, aussi bien les spécifications des applications que le code source ou le code binaire. Les quatre types de modèles identifiés sont donc les CIMs, les PIMs, les PMs et les PSMs (Bézivin et Gérard, 2003) (Miller et Mukerji, 2003) :

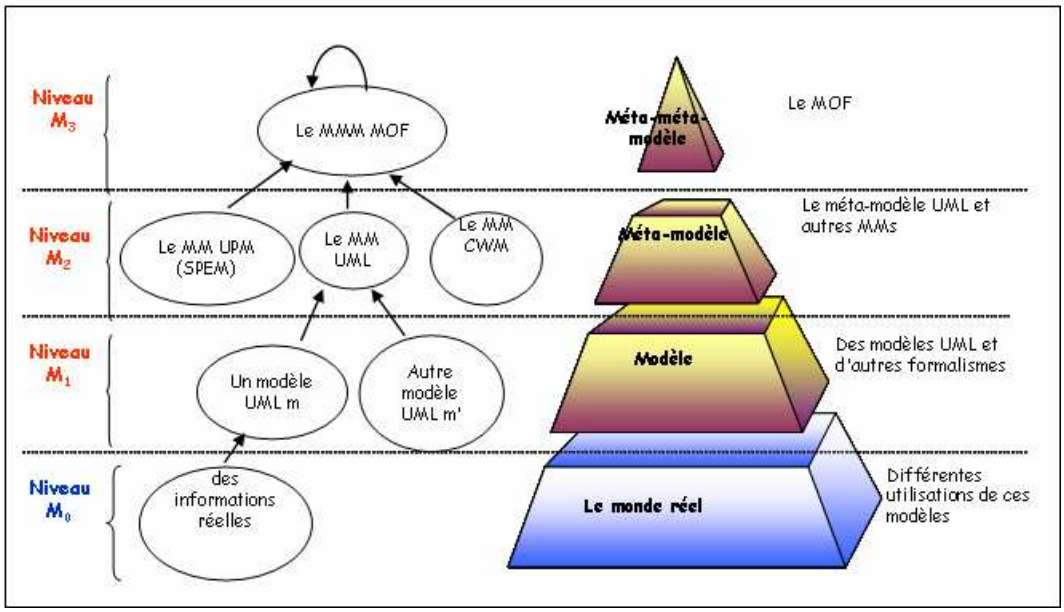

FIG. 3.3 – L'hiérarchie à 4 niveaux d'aprés (Djebbi et Gervais, 2004)

- CIM (Computation Independent Model) : appelé aussi modèle de domaine ou modèle métier. Il modélise les exigences du système. Il montre le système dans l'environnement organisationnel dans lequel il va être exécuté. Son but est d'aider à la compréhension du problème. Les exigences exprimées dans le CIM doivent être traçables dans le PIM et le PSM.
- PIM (Platform Independent Model) : décrit le système sans montrer les détails de son utilisation sur une plate-forme particulière. Un PIM peut être construit en se basant sur les points de vue Entreprise. Les PIMs représentent par exemple les différentes entités fonctionnelles d'un système avec leurs interactions. Ils représentent l'intérêt de l'application : la logique métier. Un PIM doit être raffiné par les détails d'une ou plusieurs architecture(s) particulière(s) pour obtenir un PSM.
- PM (Platform Model) : décrit la plateforme sur laquelle le système va être exécuté (modèles de composants à différents niveaux d'abstraction : CCM, C , EJB, EDOC, etc). Actuellement, il est souvent sous forme de manuels de logiciels et de matériels. Dans une démarche MDA, on se base sur les PMs pour générer les PSMs à partir des PIMs.
- PSM (Platform Specific Model) : est un modèle dépendant d'une plate-forme technologique. C'est le modèle produit par la transformation du PIM. Il spécifie comment le système va utiliser la plate-forme choisie. Les PSM servent essentiellement de base à la génération de code exécutable vers ces mêmes plate-formes techniques.

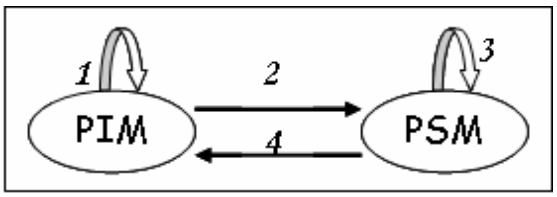

FIG. 3.4 – Les transformations de modèles dans le processus MDA

3.1.5 La transformation de modèles en MDA

La transformation de modèles est le processus qui transforme un modèle en un autre modèle du même système, mais à un niveau différent. Certaines étapes du processus MDA correspondent à la transformation d'un modèle en un autre modèle (de même type ou non), en utilisant éventuellement d'autres modèles. Plus précisément, un PIM suffisamment détaillé est projeté, par l'intermédiaire d'un PDM, vers un PSM. Quatre types de transformations différentes sont ainsi identifiés (Bézivin et Blanc, 2002) (Miller et Mukerji, 2003)(figure 3.4) :

1. Transformation de PIM vers PIM : ces transformations visent à enrichir, filtrer ou spécialiser le modèle sans utiliser d'informations dépendantes d'une plateforme. Les transformations PIM vers PIM sont généralement utilisées pour le raffinement de modèle. De telles transformations ne sont pas toujours automatiques, et demandent l'intervention du développeur pour ajouter ou soustraire des informations.

2. Transformation PIM vers PSM : ces transformations sont effectuées lorsque les PIMs sont suffisamment enrichis par des informations autorisant leur projection vers une plateforme technologique. Les caractéristiques de cette plateforme peuvent être décrites à l'aide d'un profil UML. L'opération qui consiste à ajouter des informations propres à une plateforme technique pour permettre la génération de code est une transformation PIM vers PSM. A l'heure actuelle, les plate-formes techniques visées sont : .Net, J2EE, XML et CORBA. Il apparaît clairement que ce sont les règles qui permettent ces transformations qui sont importantes et qui doivent être généralisées et capitalisées. Ces transformations peuvent donc être fortement automatisées.

3. Transformation PSM vers PSM : ces transformations s'appliquent sur un modèle spécifique et génèrent un autre modèle spécifique à la même plate-forme. En fait, une unique transformation PIM vers PSM n'est pas toujours suffisante pour permettre la génération de code, il faudra alors parfois transformer les PSM en PSM en utilisant des formalismes intermédiaires. Ces transformations sont donc effectuées pour la réalisation et le déploiement de composants.

4. Transformation PSM vers PIM : ces transformations permettent d'obtenir un PIM à partir d'une implantation existante sur une plateforme spécifique. Bien que celles-ci ne fassent pas directement partie du processus MDA, au sens où les spécifications OMG de ce processus ne les montrent pas explicitement, elles sont nécessaires à considérer dans toute stratégie de migration. Actuellement, ces transformations sont les plus difficiles à établir et à automatiser. Idéalement, le résultat de cette transformation devrait correspondre à la transformation inverse PIM vers PSM.

Une autre transformation est possible : *la transformation de CIM vers PIM*. Cette transformation peut nécessiter un enrichissement de modèle CIM pour obtenir le modèle PIM. En effet, nous pouvons dire que la distance entre CIM (exigences métiers) et PIM (exigences logiques de système) est plus grande que la distance entre PIM et PSM (exigences logiques et techniques d'un même système), ce qui rend cette transformation plus compliquée à définir (Touzi, 2007).

Mécanisme de transformations de modèles

Dans MDA, la transformation de modèles occupe un rôle majeur. Une transformation génère un modèle cible à partir d'un modèle source. Les transformations peuvent mener à des PSM ou à des PIM. Le processus de transformation est composé de trois étapes : la définition des règles de transformation, l'expression des règles de transformation et l'exécution des règles de transformation.

La transformation modèle-vers-modèle est encore en évolution alors que la transformation de modèle-vers-code est la plus diffusée et utilisée (Czarnecki et Helsen, 2003).

La figure 3.5 montre le mécanisme de transformation de modèles. En entrée on dispose d'un modèle source qui est conforme à un méta-modèle source. Les règles de transformation sont établies entre le méta-modèle source et le méta-modèle cible. Le processus de transformation prend en entrée un modèle conforme au méta-modèle source et produit en sortie un ou plusieurs autre(s) modèle(s) conforme(s) au méta-modèle cible, en utilisant les règles préalablement établies.

Pour exprimer les règles de transformation, on a besoin d'un langage de spécification des règles. Actuellement, il n'existe pas de langage standard. Quelques langages en cours de standardisation existent. Parmi ces langages on trouve, ATL (Atlas Transformation Language) (Bézivin *et al.*, 2003), Mtrans

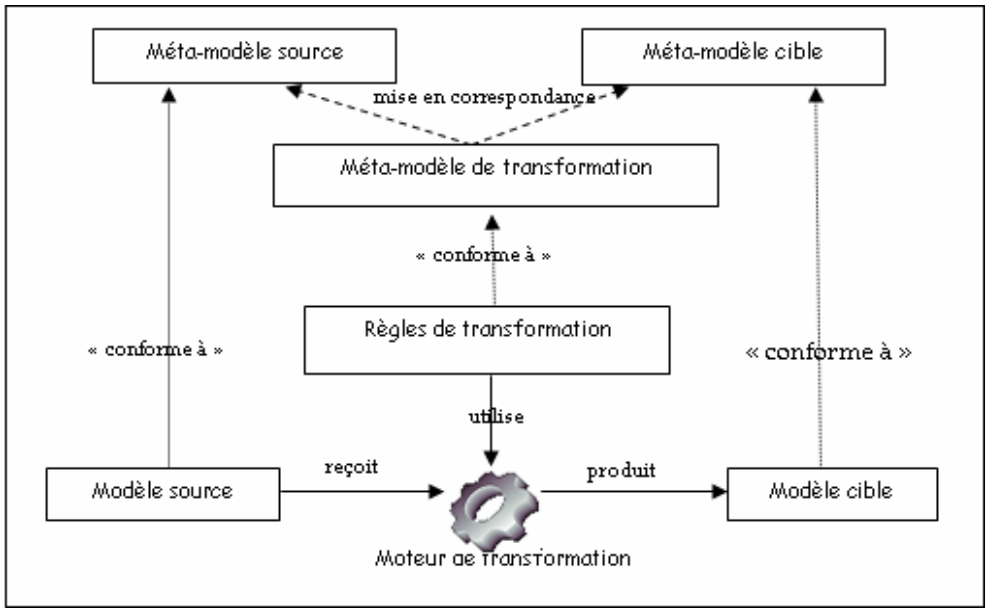

FIG. 3.5 – Architecture d'un système de transformation basé sur la méta-modélisation

(Peltier, 2003), YATL (Yet Another Transformation Language) (Patrascoiu, 2004), MOLA (MOdel transformation LAnguage) (Kalnins *et al.*, 2006) et AndroMDA (Andro, 2008).

Les outils de transformation liés à ATL sont intégrés sous forme du plugin ADT (ATL Development Tools) pour l'environnement de développement Eclipse.

Ces langages de transformations peuvent être selon plusieurs critères : bidirectionalité, tracabilité, etc. En ATL les règles de transformations ne sont pas bidirectionnelle, mais ATL fournit des supports pour permettre la création des règles bidirectionnelles. Les règles de transformation sont unidirectionnelles en YATL. ATL et YATL présentent des similarités : ils sont tous les deux basés sur OCL, langages textuels de définition de contraintes.

3.2 L'interopérabilité

Aujourd'hui, la complexité des systèmes informatiques et les environnements économiques compétitifs rendent nécessaire le développement d'architectures logicielles supportant des applications hétérogènes, tant dans les modèles d'informations traités que dans les modes d'échange et de coopération. C'est le cas des applications du e-gouvernement, du e-learning, du e-commerce ou encore des bibliothèques électroniques. Mais aussi avec les approches de col-

laboration et d'intégration ainsi qu'avec les architectures orientée services ou multi-agents.

Le problème est de faire fonctionner ensemble des technologies hétérogènes. Ce problème touche aujourd'hui chaque entreprise bien que des outils sont déjà déployés afin d'assurer l'interopérabilité des ERP, SCM, PLM, etc. Ces outils fournissent aux entreprises un minimum pour fonctionner et collaborer. Cependant beaucoup de problèmes persistent : redondance, perte, duplication d'information, etc., problèmes que les technologies d'interopérabilité visent à résoudre.

Plusieurs définitions de l'interopérabilité ont immergé. Généralement, interopérer implique qu'un système exécute une opération avec (ou pour) un autre système. (IEEE, 1990) considère l'interopérabilité comme « la capacité que possèdent deux ou plusieurs systèmes ou composants à échanger des informations puis à exploiter les informations venant d'être échangées».

(Khan, 2006) définit l'interopérabilité comme étant le fait que deux entités différentes puissent travailler ensemble. D'après (Terrasse *et al.*, février 2005), l'interopérabilité peut être définie comme l'aptitude de systèmes à pouvoir travailler ensemble sans effort particulier pour les utilisateurs de ces systèmes.

Plusieurs définitions de l'intéropérabilité mais un seul but : l'échange et l'exploitation de l'information. Trois niveaux d'interopérabilité peuvent être distingués : le niveau syntaxique, le niveau sémantique et le niveau technologique (Lewis et Wrage, 2004).

- l'interopérabilité syntaxique concerne la capacité d'échange de l'information en proposant une intégration de premier niveau, que l'on peut appeller intégration syntaxique, en définissant notamment la nature, le type et le format des messages échangés. Elle conduit à la notion de système ouvert permettant d'assumer l'hétérogénéité des composants (interfaces, langages de programmation, etc.),
- l'interopérabilité sémantique traite de l'interprétation commune du sens de l'information échangée et de la façon dont celle-ci devra être exploitée. Elle assure que les échanges qui s'effectuent entre les composants interconnectés conservent leur sens, c'est à dire que les parties communicantes ont une compréhension commune de la signification des données et des services qu'elles échangent. En effet, des conflits sémantiques surviennent puisque les systèmes n'utilisent pas la même interprétation de l'information qui est définie différemment d'une organisation à une autre,

– L'interopérabilité technologique qui concerne la coopération entre plusieurs entités logicielles issues de différentes technologies d'implémentation.

Dans la littérature, trois grands axes de recherche marquent les travaux sur le domaine l'interopérabilité :

– la modélisation de l'entreprise s'intéresse à la représentation de l'entreprise en réseau pour mettre en évidence les besoins en interopérabilité,

– les architectures et les plateformes définissent les solutions à implémenter pour atteindre l'interopérabilité,

– les ontologies adressent le besoin en sémantique pour assurer l'interopérabilité.

L'interopérabilité peut être vue comme la capacité des entreprises à structurer, formaliser et présenter leurs connaissances et savoir-faire afin d'être en mesure de les échanger ou de les partager. Du point de vue application, l'interopérabilité vise à assurer la coopération entre deux applications sans un effort particulier d'interfaçage. Les applications initialement conçues pour fonctionner dans des environnement isolés, doivent pouvoir fonctionner ensemble.

Dans ce cas, l'interopérabilité est une aptitude cruciale des entreprises, si elles souhaitent s'inscrire dans une dynamique d'intégration. L'interopérabilité est désormais incontournable pour assurer la pérennité économique de l'entreprise. (Bondé, 2006) a classé l'interopérabilité en trois catégories par référence aux différentes solutions et approches qui ont été proposées :

– l'interopérabilité dirigée par les cadres méthodologiques : dans cette catégorie d'approche, la résolution du problème d'interopérabilité passe par la définition d'une méthodologie de transformation de modèles qui permettra d'aboutir à des systèmes interopérables. On pourrait ici citer la MDA pour présenter cette famille d'approches. La réalisation des bridges (ponts technologiques) entre les plateformes avec la technologie MDA reste une tâche difficile. La prise en compte des applications déjà existantes nécessite de la rétro-ingénierie vu les efforts de re-modélisation des applications nécessaires à la génération des bridges.

– l'interopérabilité dirigée par les modèles et les échanges de données : cette catégorie d'approche d'interopérabilité repose sur l'existence d'un langage commun permettant de définir les modèles ou données à échanger. Elle répond principalement à l'interopérabilité au niveau syntaxique. L'exemple le plus caractéristique de cette famille d'approches est l'interopérabilité par échange de fichier XMI (XML Metadata Inter-

change) (XMI, 2007). C'est aussi l'approche retenue par l'OMG pour l'échange de données et modèles entre outils MDA. XMI possède plusieurs versions de spécifications ce qui génère un problème d'interopérabilité rendant nécessaires des conversions entre différentes versions de fichiers XMI afin de pouvoir réaliser l'échange.

– l'interopérabilité centrée sur les services : cette catégorie contient toutes les approches d'interopérabilité centrées sur la notion de service. L'interopérabilité dans ces approches passe par la mise en oeuvre de services connus dans l'environnement. Un exemple de cette catégorie d'approches sont les approches des architectures orientées services (SOA). Les approches d'interopérabilité basées sur les SOA se situent au niveau de l'interopérabilité syntaxique et sémantique. Interopérabilité syntaxique, à cause du format commun de codage des messages, et interopérabilité sémantique, de par l'existence d'un contrat entre les consommateurs et fournisseurs de services.

En général, l'interopérabilité a la signification de l'autonomie et de l'environnement fédéré, tandis que l'intégration se réfère davantage aux concepts de la coordination, concordance et uniformisation. Dans un système entièrement intégré, les composants sont interdépendants et ne peuvent pas être séparés. L'interopérabilité indique que les composants sont reliés par un réseau de transmission et peuvent interagir. Ils peuvent échanger tout en continuant localement leur propre opération. Ainsi deux systèmes intégrés sont inévitablement interopérables ; mais deux systèmes interopérables ne sont pas nécessairement intégrés. Deux systèmes sont considérés comme intégrés s'il y a un format standard détaillé pour tous les composants constitutifs des deux systèmes.

L'interopérabilité est liée d'avantage à l'approche unifiée où il y a une structure de méta-niveau commun pour les modèles, donnant un moyen d'établir des équivalences sémantiques. L'interopérabilité est aussi liée à l'approche fédérée là où les modèles doivent dynamiquement s'adapter plutôt que de recourir à un méta-modèle prédéterminé (Chen *et al.*, 2008).

Synthèse Avec le tableau 3.1, (Bondé, 2006) décrit pour chaque catégorie, identifiée les niveaux d'interopérabilité abordés. Les niveaux technologique et sémantique de l'interopérabilité ne sont pas assez pris en compte dans les approches actuelles d'interopérabilité. Cependant l'approche MDA propose un mécanisme de transformations de modèles entre différents outils. Nous considérons que ce type de mise en oeuvre de l'interopérabilité couplé avec l'approche MDA est une bonne solution dans le cadre de notre travail.

Par ailleurs, les approches actuelles se basent toutes sur l'hypothèse de l'existence d'un consensus. Il est donc nécessaire de définir un cadre général

	Interopérabilité Technologique	**Interopérabilité Sémantique**	**Interopérabilité Syntaxique**
MDA	Ponts technologiques	N/A	XMI
XMI	N/A	N/A	format d'échange
SOA	N/A	Contrat	format d'échange

TAB. 3.1 – Niveaux d'interopérabilité des approches (Bondé, 2006)

d'interopérabilité dans le cas des systèmes utilisant ou exposant différentes technologies dans l'entreprise étendue et qui ont besoin d'interopérabilité.

3.2.1 Model Driven Interoperability (MDI)

La MDI définit un cadre d'interopérabilité pour le Développement Dirigé par les Modèles (MDD) des systèmes logiciels. Ce cadre guide l'application de la MDD à l'adresse de l'interopérabilité pour les applications et logiciels d'entreprise. D'aprés (Elvesæter *et al.*, 2005) INTEROP (2008), ce cadre d'interopérabilité est conçu pour accomplir les raisonnements de conception logicielle suivants :

- l'identification des problèmes d'interopérabilité rapportée aux modèles de logiciel, utilisant une approche holistique où les architectures de logiciel et les architectures d'entreprise peuvent être connexes,
- identification des composants d'architecture logiciels en expliquant les rapports entre ces composants,
- intégration des processus de développement logiciel dirigé par les modèles,
- structuration des technologies, des cadres et des méthodologies de logiciel existants.

Dans le projet ATHENA (Athena, 2004), le cadre d'intéropérabilité est structuré selon trois niveaux d'intégration. Pour chacun de ces niveaux le projet INTEROP (INTEROP, 2008) définit un modèle de référence pour décrire et soutenir l'application du MDD des systèmes logiciels :

1. l'intégration Conceptuelle qui se concentre sur les concepts, les métamodèles, les langages et les relations entre modèles pour automatiser l'interopérabilité des modèles logiciels,

2. l'Intégration technique qui se concentre sur les environnements de développement et d'exécution de logiciel,

3. l'Intégration applicable (Applicative) qui se concentre sur des méthodologies, normes et modèles de domaine. Elle nous fournit des directives, principes et modèles qui peuvent être employés pour résoudre un problème d'interopérabilité logiciel.

MDI (Athena, 2004) (Elvesæter *et al.*, 2005) proposent de lier les trois niveaux conceptuels de MDA qui correspondent aux CIM, PIM et PSM. Le but de MDI est « d'obtenir des applications informatiques interopérables et qui respectent les besoins spécifiques exprimés dans les niveaux conceptuels ».

Le figure 3.6 montre une squelette de l'architecture de la MDI qui contient les éléments suivants (Elvesæter *et al.*, 2005) :

- le CIM est inclus au sein de l'entreprise. Le CIM est présenté comme un modèle qui collecte les besoins et les caractéristiques de l'entreprise en utilisant un modèle d'entreprise. Ce modèle permet de représenter et comprendre comment l'entreprise fonctionne, afin de capitaliser ses connaissances et son savoir-faire pour une utilisation ultérieure. Ce modèle permet de représenter et comprendre comment l'entreprise fonctionne, afin de capitaliser ses connaissances et son savoir-faire pour une utilisation ultérieure,
- le PIM est inclus dans un système informatique. Le PIM est un modèle obtenu à partir des modèles d'entreprise définis dans le CIM. Le PSM représente plusieurs modèles d'application suivant la technologie choisie,
- le PSM est inclus dans une plate-forme d'exécution, qui est elle-même incluse dans le système informatique.

Afin de dépasser la difficulté inhérente à la mise en place de la méthodologie MDI, on propose de partir du niveau des modèles d'entreprises et d'utiliser des transformations horizontales pour l'interopérabilité au même niveau et des transformations verticales pour arriver au niveau de l'application informatique.

3.2.2 Les cadres d'interopérabilité en entreprise

Une entreprise est un ou plusieurs organismes partageant une mission définie et des buts pour offrir un résultat tel qu'un produit ou un service. Cette large définition couvre également l'entreprise étendue. L'entreprise étendue est définie comme étant une forme d'organisation englobant tous les partenaires,

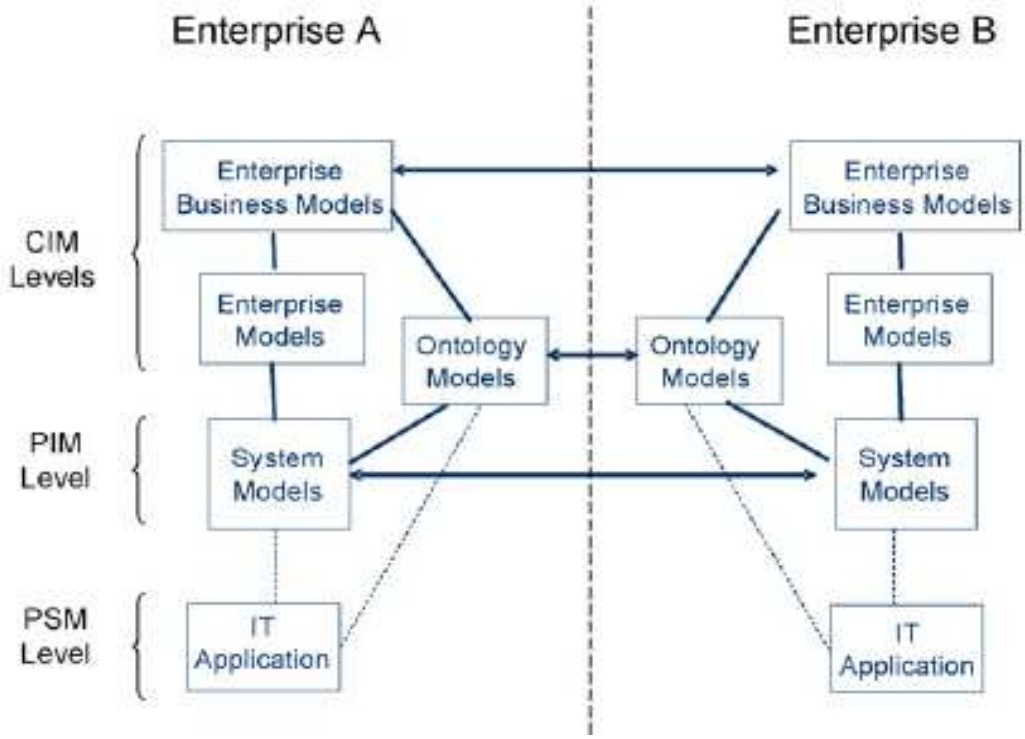

FIG. 3.6 – Architecture de la MDI d'aprés (Chen *et al.*, 2008)

fournisseurs, donneurs d'ordres, sous-traitants, concurrents. L'organisation est conçue comme un ensemble d'entités, mêlant leurs activités, en mettant en commun des ressources et en aménageant des contraintes afin d'essayer d'atteindre un ou plusieurs objectifs en commun ou pas.

Bien que ces entreprises reposent sur un ensemble de relations et un objectif commun, elles se différencient sur les points suivants :

- la nature des relations entre partenaires (sous-traitants/donneurs d'ordre, concurrents),
- les types de coordination des relations entre partenaires (horizontale, verticale, diagonale).

Ces différences induisent des différences d'ordre culturel, linguistique, métier et même technologique. Dans le cadre des entreprises étendues, l'interopérabilité se rapporte à la capacité des interactions (échange de l'information et services) entre les systèmes d'entreprise.

L'interopérabilité est significative si les interactions peuvent avoir lieu au moins à trois niveaux différents : données, services et processus, avec une sémantique définie dans un contexte donné métiers. Plusieurs travaux de recherche se sont intéressés à la définition de cadres caractérisant des niveaux de considération de l'interopérabilité dans les entreprises. Nous présentons cinq cadres de référence.

Le cadre LISI

Le cadre LISI (levels of information systems interoperability) est une approche développée pour fournir au Département de Défense Américain un modèle et un processus pour déterminer les besoins communs d'interopérabilité, évaluer la capacité des systèmes d'information à répondre à ces besoins et choisir des solutions pragmatiques et un chemin de transition pour réaliser une meilleure interopérabilité (C4ISR, 1998).

Dans le modèle LISI, les éléments de l'interopérabilité requise pour les systèmes d'information désireux d'interopérer sont décrits dans les quatre attributs connus sous le nom de PAID (voir la figure 3.7), à savoir : les procédures, les applications, infrastructures (matériel, communications, sécurité et services de système) et les données.

L'approche LISI, bien que construite avec des concepts et des modèles génériques, est appliquée pour l'interopérabilité dans le secteur militaire des USA. Cependant, elle est également employée comme base pour élaborer d'autres modèles de maturité d'interopérabilité.

Le cadre IDEAS

Le cadre IDEAS (Interoperability Development for Enterprise Applications and Software, IST-2001-37368) (IDEAS, 2003) est présenté comme une approche structurée pour la collecte et l'identification des visions et des enjeux de recherche sur l'interopérabilité des applications d'entreprise. L'interopérabilité est considérée sur trois niveaux horizontaux et une dimension transversale (Figure 3.8) :

- un niveau métier (business) : il définit le contexte métier et les processus collaboratifs des entreprises,
- un niveau connaissances (knowledge) : il concerne l'organisation, les qualifications et les sources de connaissances dans l'entreprise,
- un niveau technologies de communication (ICT systems) : il concerne les applications, les données et les infrastructures de communication,

Description	Computing Environment	Level	P	A	I	D
Enterprise	Universal	4	Enterprise Level	Interactive	Multi-Dimensional Topologies	Enterprise Model
Domain	Integrated	3	Domain Level	Groupware	World-wide Networks	Domain Model
Functional	Distributed	2	Program Level	Desktop Automatior	Local Networks	Program Model
Connected	Peer-to-Peer	1	Local/Site Level	Standard System Drivers	Simple Connection	Local
Isolated	Manual	0	Access Control	N/A	Independen:	Private

FIG. 3.7 – modèle de référence LISI

– une dimension sémantique (Semantics) : elle assure une compréhension mutuelle à tous les niveaux identifiés précédemment. Elle constitue la dimension transversale.

Le cadre IDEAS est la première initiative en Europe pour aborder l'interopérabilité d'entreprise de fabrication. Il est employé comme base pour élaborer des feuilles de route dans le cadre des projets ATHENA et INTEROP NoE. Le cadre d'interopérabilité IDEAS est basé sur les trois domaines appropriés de recherches (modélisation d'entreprise, architecture et plate-forme et ontologie)

Le cadre AIF

Le cadre AIF (Athena Interoperability Framework) (ATHENA, 2005) adopte une approche holistique de l'interopérabilité permettant de mieux comprendre et d'analyser les exigences de l'interopérabilité. Ce cadre définit trois niveaux :

– un niveau conceptuel : il définit les concepts, méta-modèles et langages utiles pour la modélisation de certains aspects de l'interopérabilité,

– Un niveau applicatif : il concerne les méthodologies, les standards et les « patterns » qui interviennent dans le développement de solutions interopérables,

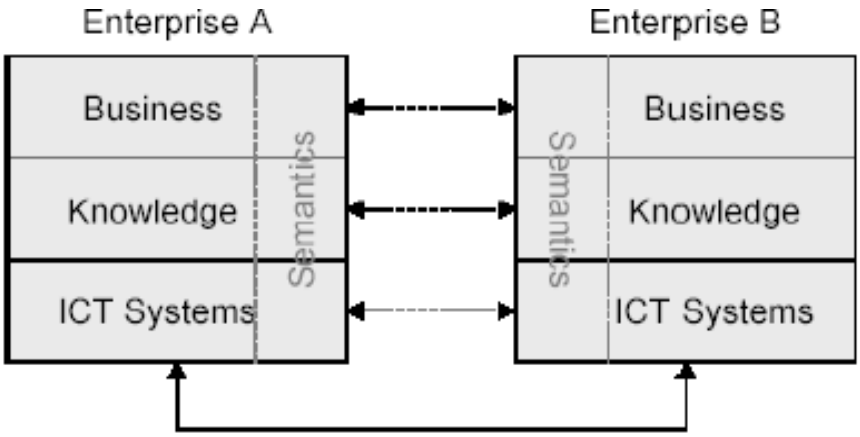

FIG. 3.8 – Le cadre IDEAS

– Un niveau technique : il décrit les plate-formes d'interconnexion de systèmes.

Le cadre E-health

Le cadre E-health (NEHTA, 2005) a été développé par NEHTA (National E-Health Transition Authority) en Australie.

Ce cadre intègre des aspects organisationnels, informationnels et techniques (figure 3.9). Le cadre d'interopérabilité crée la structure et régit son évolution avec les couches d'interopérabilité qu'elle entoure. Il crée une séparation des préoccupations parmi les couches du cadre et rassemble les groupes d'experts pour aligner et guider les couches :

– l'interopérabilité organisationnelle crée la cohésion parmi des approches de finances, de législation et métiers,

– l'interopérabilité de l'information contient tout type d'information comme des données de base,

– l'interopérabilité technique combine tous les aspects des normes avec une large approche architecturale liant les services et l'information du cadre E-health.

Le cadre EIF

Le cadre EIF (European Interoperability Framework) (EIF, 2004) vise à structurer un ensemble de recommandations et de directives pour l'interopérabilité entre les administrations européennes d'une part et entre les adminis-

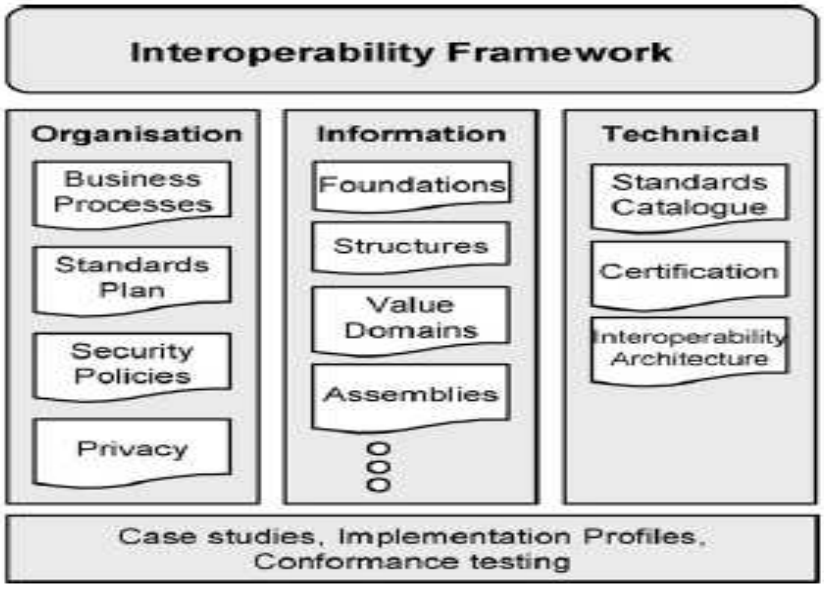

FIG. 3.9 – Le cadre E-health (Chen *et al.*, 2008)

trations et les citoyens d'autre part. L'intérêt est qu'il présente trois niveaux (appelés dimensions dans ce cadre) pour le traitement de l'interopérabilité :

- un niveau organisationnel concerne les objectifs métiers, la modélisation des interactions entre partenaires (modèles de processus) et la description de leur entreprise,
- un niveau sémantique concerne le sens des informations et des connaissances des entreprises,
- un niveau technique concerne les solutions technologiques et les outils d'interconnexion de systèmes.

3.2.3 Bilan des approches d'interopérabilité

Plusieurs cadres on été définis pour répondre à la problématique d'interopérabilité dans les domaines industriel et administratif. Des similitudes existent entre les différents niveaux de ces cadres d'interopérabilité :

- le niveau organisationnel d'EIF et le niveau métier d'IDEAS : définissent l'échange d'objectifs, de stratégies, de responsabilités et de modes de fonctionnement des entreprises,
- le niveau sémantique d'EIF et celui d'IDEAS : concernent la signification et l'interprétation des informations et des connaissances des entreprises,

– l'accord sur un niveau technique de l'interopérabilité des entreprises.

Les barrières à l'interopérabilité (Chen *et al.*, 2008) identifient les obstacles divers d'interopérabilité en trois catégories (conceptuel, technologique, et organisationnel) :

– barrières conceptuelles : elles concernent les différences syntaxiques et sémantiques d'information à échanger résultant de la diversité des modes de présentation et de communication des concepts,

– barrières technologiques : elles se rapportent à l'incompatibilité des technologies de l'information (architecture et plates-formes, infrastructure, etc.),

– barrières organisationnelles : elles se rapportent à la définition de la responsabilité (qui est responsable de quoi ?) et autorité (qui est autorisé à faire quoi ?) aussi bien qu'à l'incompatibilité de structures d'organisation (matricielle vs hiérarchique, par exemple).

Les niveaux d'interopérabilité L'interopérabilité peut avoir lieu aux différents niveaux de l'entreprise. Bien que la catégorisation suivante soit principalement donnée d'un point de vue des applications basées sur la technologie de l'information, elle s'applique aussi bien aux systèmes non-automatisés. Elle est basée sur l'architecture technique du projet ATHENA (Athena, 2004) :

– l'interopérabilité au niveau des données assure la communication entre différents modèles de données et langages de requêtes. L'interopérabilité des données revient donc à localiser et partager des informations provenant de sources hétérogènes appartenant à des bases de données différentes opérant sur des systèmes d'exploitation différents supportés par des machines différentes,

– l'interopérabilité au niveau des services concerne l'identification, la composition et le rassemblement des différentes fonctions d'applications fonctionnant ensemble (conçues et mises en application indépendamment). Le terme « service » n'est pas limité à la notion de « web service » ou une application particulière, mais s'étend pour couvrir les fonctions d'une compagnie ainsi que les entreprises en réseaux,

– l'interopérabilité au niveau des processus vise à faire fonctionner ensemble divers processus métier : un processus est défini par une séquence de services (fonctions) pour répondre à un besoin spécifique de l'entreprise. Dans le contexte de l'inter-entreprise, il est nécessaire d'étudier comment relier des processus internes de deux compagnies pour créer un processus commun,

– l'interopérabilité au niveau des métiers vise à acquérir la capacité à connecter, tant en interne à l'entreprise qu'en externe avec ses partenaires, les différentes spécifications métiers. Cette connexion doit se faire indépendamment de la vision interne d'une entreprise, de ses modèles métiers, de ses modes de décisions et de ses bonnes pratiques. Ceci facilite le développement et le partage des spécifications métiers entre les compagnies.

Les deux dimensions des cadres d'interopérabilité de l'entreprise sont présentées dans la figure 3.10 : les niveaux d'interopérabilité et les barrières d'interopérabilité (Chen *et al.*, 2008). Sur cette figure nous définissons les domaines que nous abordons dans notre travail. En effet nous sommes spécialement intéressés par le niveau données de l'entreprise. Sur le niveau données, nous étudions des solutions afin d'améliorer l'interopérabilité technologique et syntaxique. Pour l'interopérabilité technologique, nous appliquons une méthodologie basée sur l'Ingénierie Dirigé par les Modèles (IDM). Pour l'interopérabilité syntaxique, nous adoptons un format d'échange de données GAM (proche de XML).

barrières / niveau	Conceptuelle		Technologique	Organisationnelle
	Syntaxique	Sémantique		
Métiers				
Processus				
Services				
Données	Format d'échange		IDM	

FIG. 3.10 – Positionnement par rapport aux cadres d'interopérabilité

Une troisième dimension (approches d'interopérabilité) est ajoutée pour classer la connaissance et les solutions liées à l'interopérabilité d'entreprise. Le projet ATHENA présente le cadre suivant pour classer en grille les différentes dimensions à prendre en considération lors du déploiement d'une approche basée sur les principes de l'interopérabilité (figure 3.11). Nous choisissons une approche fédérée pour l'interopérabilité.

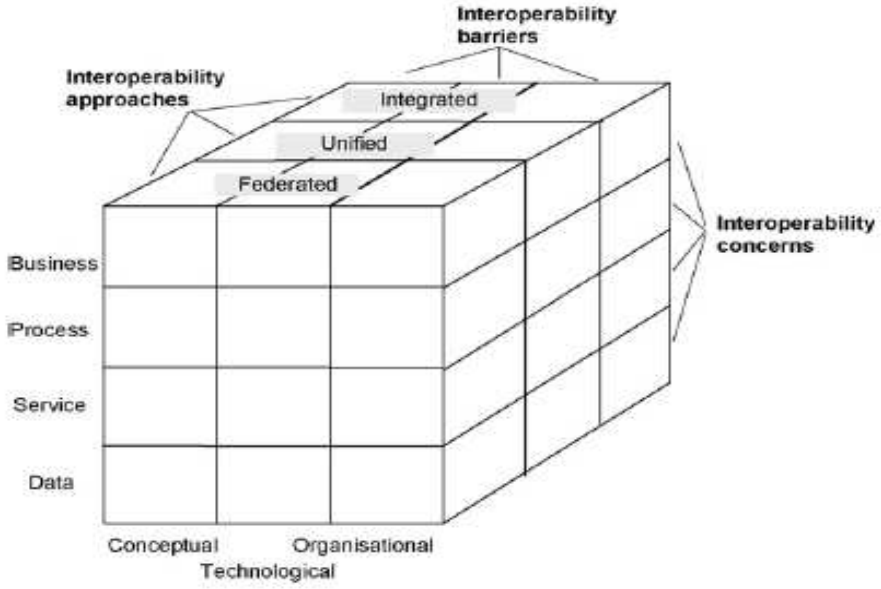

FIG. 3.11 – Le cadre d'interopérabilité pour les entreprises (Chen *et al.*, 2008)

3.2.4 L'interopérabilité dans le développement des produits manufacturiers

Le terme « interopérabilité de fabrication » se rapporte à la capacité de partager l'information technique et métier à travers une entreprise manufacturière étendue. Ce terme désigne aussi bien l'information antérieurement partagée de manière différente qui doit être véhiculée entre différents acteurs (fournisseurs, clients et.), que la complémentarité entre les acteurs de l'entreprise.

Plusieurs travaux ont reconnu l'importance de l'échange des données produit et de la modélisation de l'information comme moyen d'obtenir un certain niveau d'intégration. Par exemple (Aifaoui *et al.*, 2006) précise qu'en conception manufacturière, l'intégration de l'analyse mécanique dans le processus de conception est une condition majeure en particulier pendant les premières étapes de la conception.

Dans les systèmes PLM (Product Lifecycle Management), les informations de conception doivent être communiquées tout au long du cycle de vie du produit. Les limitations courantes de l'interopérabilité affectent inévitablement la capacité de partage de la connaissance de conception. Ce manque d'interopérabilité est coûteux pour beaucoup d'industries globalement distribuées où des

sommes d'argent significatives sont dépensées pour surmonter les problèmes d'interopérabilité.

Le logiciel de Conception Assistée par Ordinateur (CAO) est habituellement utilisé pour établir un modèle de référence du produit. Habituellement, le modèle de produit inclut la représentation 3D géométrique mais également des attributs techniques tels que les propriétés des matériaux, les caractéristiques de traitement des surfaces, des éléments de processus de fabrication. La somme de toutes ces informations représente le modèle produit. L'analyse de la fabrication du produit est accomplie par des logiciels de Fabrication Assistée par Ordinateur (FAO). Les logiciels de FAO préparent, commandent et contrôlent des Machines à Commande Numérique. L'interopérabilité entre CAO et FAO sera un gain appréciable dans les pratiques industrielles.

La conception et la fabrication d'un produit complexe fait participer typiquement plusieurs concepteurs qui sont généralement géographiquement distribués et utilisent des outils hétérogènes dédiés à leur expertise impliquant des efforts d'échange de données entre les systèmes de CAO et de FAO : des systèmes de CAO et de FAO sont encore développés de manière isolée et ne peuvent communiquer les uns avec les autres. Des problèmes persistent chaque fois que l'information du produit doit être communiquée et partagée.

Les dépenses des industries causées par les problèmes d'interopérabilité sont énormes. Des approches sont apparues pour réduire ces coûts. Dans un contexte de CAO/FAO, il y a trois méthodes pour soutenir l'échange de données : l'utilisation des modules de logiciel intégrés de CAO/FAO du même fournisseur, l'utilisation du format d'échange de données standard tel que le format IGES ou STEP et le format d'échange de données spécifique (format propriétaire). Ces trois démarches ont été détaillées au chapitre précédent.

La figure 3.12 récapitule les différentes méthodes d'échanges de données. La connexion entre un ou plusieurs systèmes de CAO à différents systèmes de FAO utilisant les types de liens décrits ci-dessus est réalisée. (Xu *et al.*, 2002) note que la connexion entre les systèmes de CAO et de FAO est itérative, exigeant des conversions de données additionnelles et un niveau important d'entrée manuelle.

L'interopérabilité parmi des systèmes de CAO/FAO est un problème bien connu dans la conception et le développement produits (Bianconi *et al.*, 2006). Actuellement l'échange de données géométriques parmi différents progiciels est effectué par des formats standards (IGES ou STEP) ou par des formats propriétaires avec toutes les pertes déjà évoquées au chapitre 2.

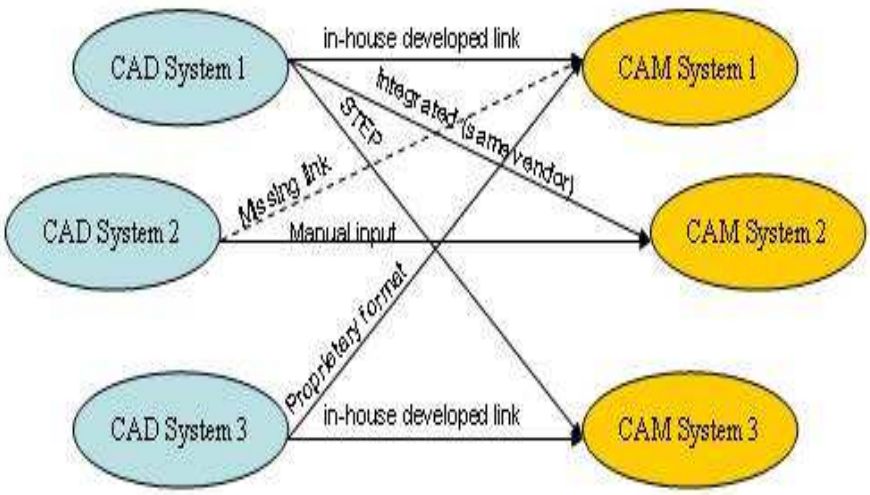

FIG. 3.12 – Type de connections entre des outils de CAO et de FAO

Des recherches se sont concentrées sur le développement des systèmes pour l'échange des modèles procéduraux (Bianconi *et al.*, 2006). Une représentation procédurale est définie en termes d'entités et de fonctions qui peuvent être reconstruites dans différents environnements sans perte d'information. (Bianconi, 2005) se focalise sur la définition de modèles procéduraux et basés-feature par la spécification d'une ontologie de feature (un ensemble d'objet, de concepts et de règles), et les spécifications d'une identification unique de mêmes entités géométriques/topologiques (nomination persistantes/cartographie de nomination). La définition d'une ontologie appropriée de feature et le problème de la nomination persistante ou de la cartographie de nomination semblent prédominants. Si des approches intéressantes sont développées par divers groupes de recherche, toutefois une solution complète et unifiée reste à venir.

D'autres recherches se sont concentrées sur le développement d'une plate-forme basée Web services pour l'échange des modèles procéduraux de CAO entre les systèmes de CAO hétérogènes (Chen et Li, 2005). La plate-forme est basée sur des commandes de modélisation neutres et les API des systèmes de CAO. La technique de services Web est employée pour construire une interface standard pour l'échange de modèle de CAO entre sites distants.

La bibliographie autour des systèmes basés-interface prouve que, pendant les dernières années, des nouvelles approches intéressantes ont émergé : parmi elles les CAD services de l'OMG. Les CAD services (Claus et Weitzer, 2002) est

un processus continu mené par l'OMG (Object Management Group). L'interface standard de CAD services est un effort pour fournir un service uniforme, simple et distribué qui permette une approche de conception centrée sur la géométrie. Les CAD services proposent une interface standard pour des outils de CAD/CAM/CAE. Cette proposition se concentre sur l'établissement des interfaces de système mécaniques de CAO qui fournissent des données de géométrie et de topologie aux outils d'analyse et de fabrication. L'intention est d'établir une série d'interfaces de haut niveau de technologie qui n'exigent pas des structures de données de bas niveau pour répondre à des requêtes d'ingénierie mécanique. Pour éviter plusieurs problèmes liés à la traduction de données, cette proposition fournit des interfaces CORBA à travers des implémentations natives de système CAO. Toutefois, les CAD Services demeurent peu implémentés dans les codes commerciaux.

Une manière de limiter les problèmes d'interopérabilité entre la CAO et les outils métiers consiste à figer la CAO avant les analyses métiers. Dans le cas des échanges de données entre les systèmes de CAO et de FAO, le modèle CAO ne doit plus évoluer. Le système de FAO prend en entrée le modèle de CAO et réalise l'opération de fabrication. Si une modification est nécessaire dans le modèle CAO lors de l'étape de fabrication, aucun support n'est fourni pour renvoyer l'information et pour mettre à jour le modèle CAO. Actuellement les ingénieurs comblent cette lacune en développant leurs propres pratiques totalement informelles.

L'intégration dans l'entreprise étendue concerne généralement des applications logicielles. Chacune de ces applications dispose d'interfaces par lesquelles elles interagissent avec les personnes et d'autres applications logicielles. Les personnes travaillant avec ces applications développent des connaissances et deviennent expertes.

Dans le projet AMIS (Automated Methods for Integrating Systems) développé par le NIST, (Ray et Jones, 2006) ont a essayé d'automatiser partiellement la translation d'une application A vers une application B en se basant sur la formalisation des interactions via des ontologies, le mapping sémantique, et la transformation des connecteurs.

La figure 3.13 représente l'architecture de l'outil qui produit des traductions. Le projet AMIS contient trois domaines principaux. Chaque élément doit être focalisé pour soutenir l'intégration automatisée.

– formulation de modèles d'action commune (Joint Action Model) : capturant les interactions métiers d'une manière appropriée pour le raisonnement machine,

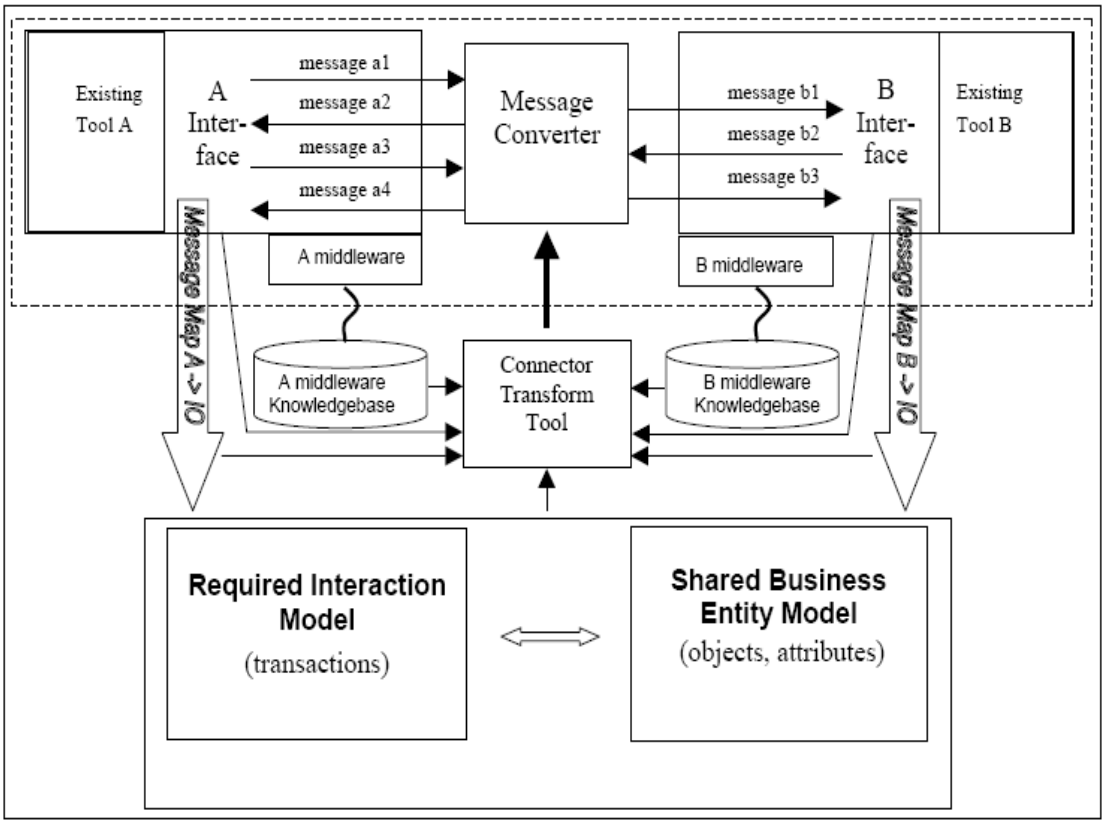

FIG. 3.13 – Une vision d'intégration (Gunendran *et al.*, 2007)

- mapping sémantique : construire des outils pour implémenter des liens sémantiques entre les modèles,
- transformation de connecteurs : construire des outils pour créer les cartes sémantiques entre les modèles.

Notre démarche utilise une approche similaire et la complète par des méthodologies IDM afin de proposer une assistance à l'interopérabilité entre outils métiers en conception collaborative. Ceci impose la formalisation des modèles métiers et des règles métiers, le tout autour d'un environnement collaboratif.

Un des problèmes majeurs lors de l'échange de l'information et de la connaissance est la disparité sémantique, qui s'oppose à l'interopérabilité entre outils de conception. Des cadres de travail pour capturer la connaissance de conception enrichie sémantiquement pour l'interopérabilité sont apparus. Des cadres sémantiques basés sur les ontologies et sur des définitions explicites des terminologies offrent une approche pour résoudre des problèmes d'interopérabilité en conception. Les applications métiers utilisent des processus qui ne partagent pas forcément la syntaxe et les définitions des concepts. Ceci

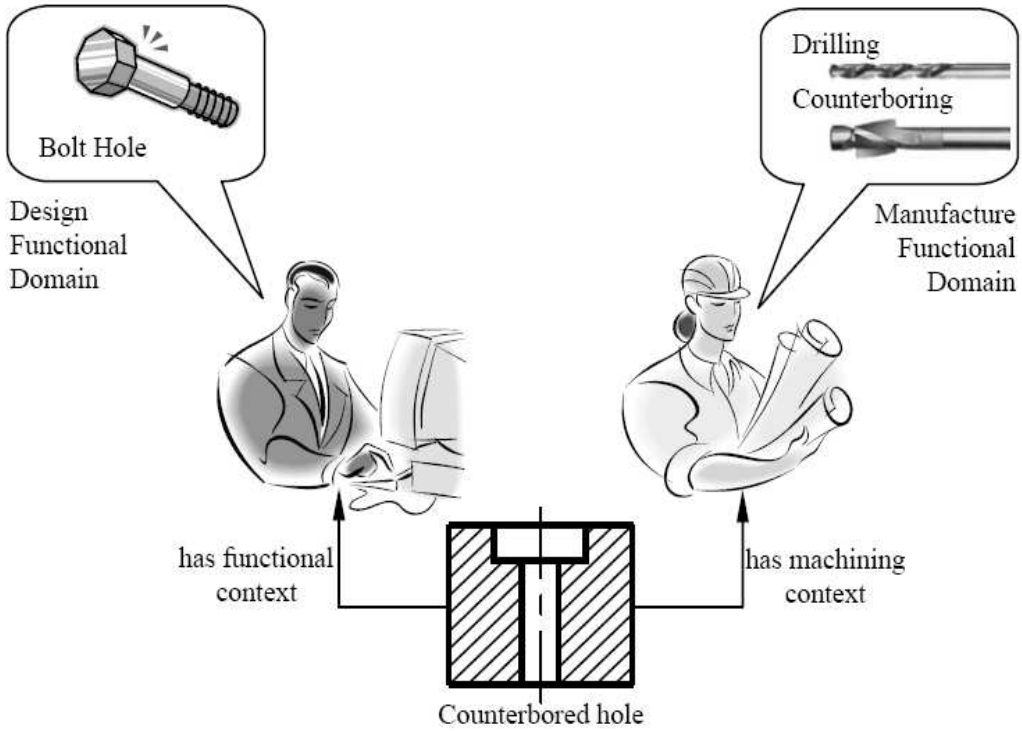

FIG. 3.14 – Un trou dans deux contexte différents

engendre un problème d'intéropérabilité sémantique. (Chungoora et Young, 2008) notent que des problèmes surgissent suite à l'emploi de termes communs pour signifier différentes choses ou de termes différents pour signifier la même chose.

(Gunendran *et al.*, 2007) présentent un exemple de conception d'un trou pour illustrer le problème de l'intéropérabilité sémantique (figure 3.14). Dans le domaine de conception, le concepteur considère le trou comme ayant des paramètres géométriques, il est peut être constitué par une opération géométrique (extrusion, protrusion etc). Dans le domaine de fabrication, le fabricant considère le même trou comme une opération d'usinage à effectuer. Dans ce cas, le trou a besoin d'une opération de perçage. Un même objet peut être considéré de différentes façons selon le contexte et la sémantique comme par exemple des contextes fonctionnel, géométrique, de fabrication, d'usinage, d'assemblage, etc.

3.3 Synthèse

De nos jours, l'intégration constitue une approche importante pour traiter l'hétérogénéité dans les entreprises. Actuellement, les solutions existantes sont essentiellement basées sur l'utilisation de certains standards et de plateformes collaboratives pour résoudre le problème d'intégration. Ces solutions échouent généralement car elles ne fournissent pas plus de flexibilité, d'agilité, et d'interopérabilité au niveau opératoire.

Dans ce chapitre, nous avons vu en première partie une technologie émergente qui est la MDA. Cette technologie a été appliquée en développement logiciel pour faire interopérer des outils de développement. La MDA est aussi une méthodologie de développement logiciel qui consiste à concevoir l'application à un niveau indépendant de la plateforme et ensuite à générer du code dépendant de la plateforme. Nous souhaitons appliquer cette technologie dans les domaines de conception et de fabrication de produit manufacturier afin de construire un outil qui assiste la collaboration et qui permettra l'interopérabilité en évitant les solutions d'intégration.

Dans ce travail, nous nous sommes intéressés aux échanges de données entre outils de CAO et outils de FAO. Nous fournissons un cadre approprié pour améliorer les processus de l'entreprise par l'adoption d'un environnement basé sur l'IDM, augmentant l'interopérabilité entre les outils de conception et de fabrication. L'IDM est ici employée pour formuler le savoir-faire spécifique pour permettre la réutilisation de cette connaissance dans des activités de synchronisation à un haut niveau. L'IDM a été au commencement développée pour le domaine du génie logiciel (avec la MDA). L'intérêt de l'étude courante est de démontrer l'application possible de cette technologie dans un autre domaine et de souligner ses limites pour ces nouvelles applications.

Chapitre 4

L'application de l'IDM sur des modèles de conception

Sommaire

4.1 Introduction

Pendant la phase de développement de produit, plusieurs experts sont impliqués dans la définition du produit. Ces experts développent des connaissances et du savoir faire. Les outils métiers aident les experts dans la mise en oeuvre de leur savoir et de leurs connaissances pour la définition du produit. Les experts collaborent de façon synchrone ou asynchrone. Les phases

synchrones correspondent aux réunions (co-localisées ou à distance) où les experts négocient les évolutions principales du produit en mettant en avant leur analyse métier qui a été développée pendant des phases asynchrones.

L'hétérogénéité des outils métiers implique la représentation en différents formats de fichiers et de données qui doit être harmonisée pour assurer la cohérence de la conception. La capacité de partager des modèles métiers est importante pendant la conception et la fabrication de produits. En ingénierie mécanique, les concepteurs emploient souvent des logiciels de Conception Assistée par Ordinateur (CAO) et de la Fabrication Assistée par Ordinateur (FAO). L'intégration des données issues des outils de CAO et de FAO dans les processus de conception relève un certain nombre de défis. Un des défis les plus significatifs est la synchronisation des vues métiers à travers l'éventail d'outils commerciaux de CAO et de FAO. Cette synchronisation demeure informelle en raison du manque de liens entre les outils d'assistance spécialisée. Cependant, parce qu'ils collaborent à la définition du même produit, les experts partagent leurs analyses informellement.

Dans ce chapitre, nous étudions comment nous pouvons formaliser ces pratiques, et comment cette formalisation peut être employée pour aider à la synchronisation. Dans ce chapitre, nous démontrons sur une étude de cas simple comment une telle synchronisation entre l'information issue du logiciel de CAO et de FAO peut fonctionner.

Nous allons ensuite voir l'application de concept de modélisation et de méta-modélisation (concept clé de l'IDM) avec des outils métiers de conception et de fabrication. Les outils métiers capitalisent une partie de l'expertise métier. L'expert métier travaillant avec ces outils développe une connaissance métier qui n'est généralement pas formalisée dans les outils métiers. En collaborant, l'expert métier développe aussi une connaissance autour du partage d'information avec les collaborateurs. Notre approche pour l'interopérabilité se base sur l'approche fédérée. Même si nous sommes dans une approche fédérée, nous avons besoin d'un langage de dialogue entre les différentes expertises. Ceci implique qu'il faut un modèle de partage pour tous les experts métiers. Dans notre cas, ce modèle est implémenté dans une plateforme spécifique : l'environnement GAM (Noel, 2007) qui fait l'objet de la section suivante. Les modèles métiers doivent donc être lisibles via le langage de cette plateforme. Pour permettre une meilleure collaboration entre les acteurs, une méthode d'extraction de l'information à partir des outils métiers dans un format spécifique est nécessaire.

Dans ce chapitre, nous présentons une méthode basée sur les API natives des logiciels commerciaux et en collaborant pour extraire l'information de collaboration. La figure 4.1 montre le mécanisme d'extraction de l'information depuis

les outils métier vers l'environnement GAM ainsi que les différents acteurs impliqués. Ce mécanisme se fait en trois grandes étapes :

- définition des concepts de collaboration : l'expert métier définit les concepts essentiels de partage avec une autre expertise métier. Cette étape nécessite la discussion avec les experts du domaine pour formuler les concepts commun de partage,
- implémenter l'API de l'outil métier dans GAM : l'environnement GAM offre une plateforme permettant la définition des modèles et des méta-modèles. L'API développée permet l'instanciation du méta-modèle métier,
- le développeur du logiciel métier utilise l'API GAM précédemment développée et l'API de l'outil logiciel pour implémenter le programme qui extrait le modèle de données à partager avec d'autres expertises métiers.

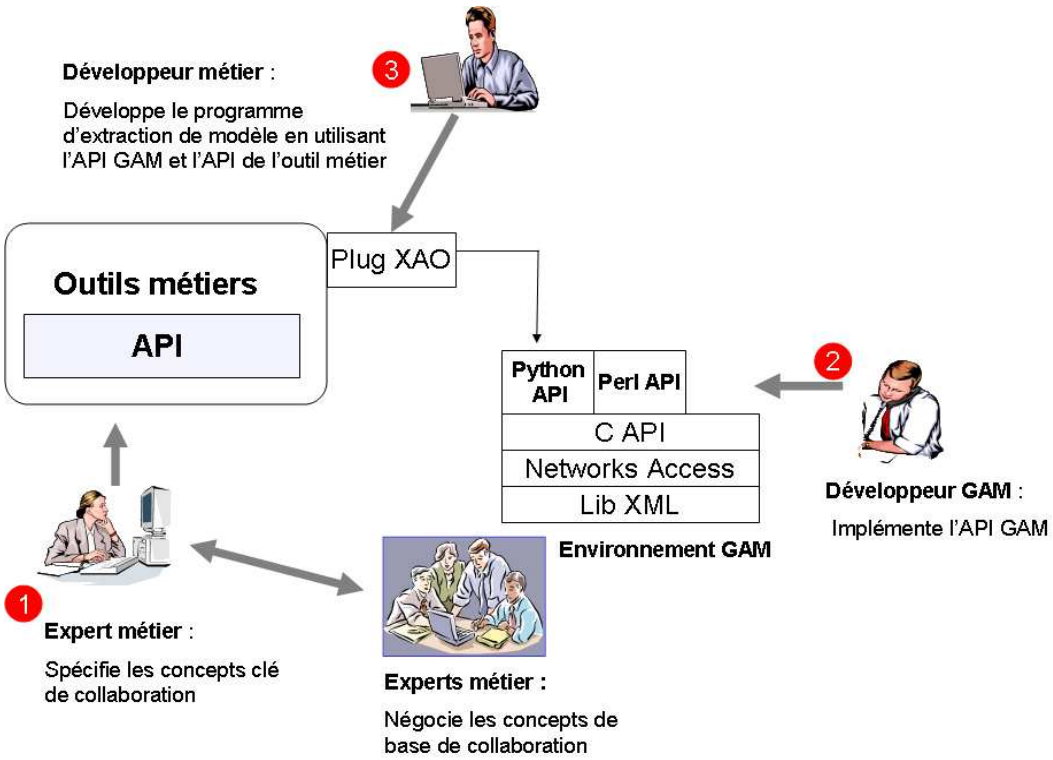

FIG. 4.1 – Méthode d'extraction des modèles XAO

Pour mieux comprendre ce mécanisme, la section 2 détaille la structure de l'environnement GAM qui nous sert de support d'implémentation pour nos propositions. La section 3 s'intéresse à l'application du mécanisme avec un

outil CAO. La section 4 reprend la même logique avec un outil de fabrication assistée par ordinateur. Le but étant d'offrir un support à la collaboration entre ces deux outils. La première étape vers cette collaboration est l'extraction de l'information métier de partage vers l'environnement de partage GAM. Le chapitre suivant discutera l'usage de cette information partagée.

4.2 L'environnement GAM

GAM est un environnement de gestion simultanée de modèles et de méta-modèles (Noel, 2007). Son développement a été initié en 1999 pour la Gestion d'Analyses Mécaniques où la modélisation des hypothèses et de leur importance dépend fortement du contexte de chaque entreprise. Les méta-modèles peuvent être définis dans le même cadre que les modèles. Un modèle spécifique de GAM le dote d'une IHM. Cette IHM (figure 4.2) permet notamment de présenter les modèles et méta-modèles suivant une vue arborescente. GAM est pensé comme une plate-forme expérimentale pour la conception collaborative dans le processus de développement de produit. Il fournit des outils pour contrôler l'information utilisée au cours du cycle de vie de produit. L'intention de GAM est d'établir un cadre de modélisation très générique permettant la gestion facile de chaque méta-modèle et modèle. Ce cadre permet entre autres la comparaison et la modification des modèles.

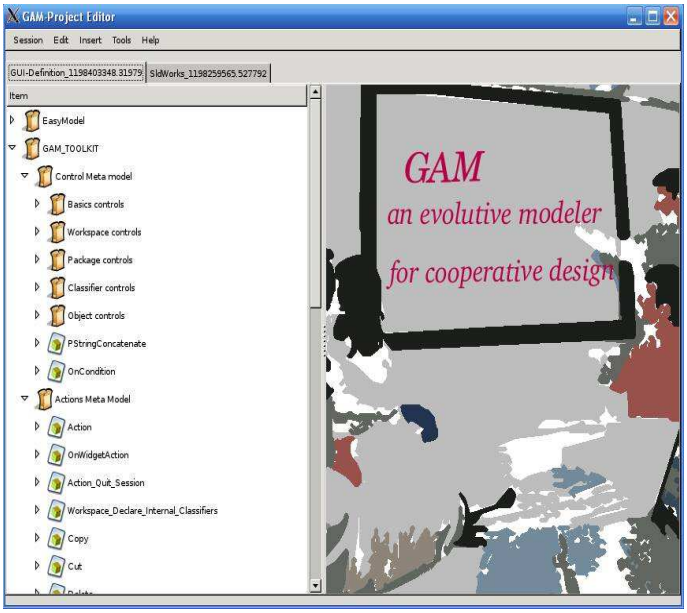

FIG. 4.2 – IHM de GAM

L'environnement GAM implémente trois niveaux d'abstraction. Au niveau 1, on trouve les modèles développés par l'utilisateur final et constitués d'un ensemble d'entités liées. Au niveau 2, on trouve les méta-modèles. La couche Méta-modèle détermine la structure d'un modèle particulier. Au niveau 3, on trouve la couche de Méta-méta-modèle qui décrit la structure des méta-modèles. Des opérateurs spécifiques fournissent quelques interfaces de protocole d'application dans plusieurs langages de programmation. Dans la version actuelle, le stockage de l'information permanente est assuré dans des dossiers via des fichiers XML, mais ceci est masqué aux utilisateurs.

L'instanciation du méta-méta-modèle GAM fournit soit une nouvelle description d'un méta-modèle ou un modèle conforme à un méta-modèle. De fait, GAM est un ensemble de modules compatibles pour soutenir la collaboration et fournit une base de données partagée. L'évolution simultanée des modèles et des méta-modèles est tracée, versionée et gérée pour analyser les actions des concepteurs. GAM prend en charge également des méta-modèles et des modèles distribués par une connexion internet via un mode client/serveur mis en oeuvre au niveau 3 et donc directement utilisable aux niveaux 1 et 2 par héritage.

4.2.1 Un modeleur de graphes

Le noyau GAM est un modeleur de graphes. Le graphe est composé de noeuds et d'arcs. Chaque noeud possède un identificateur unique qui est généré à sa création. Les arcs et les noeuds peuvent être associés à des propriétés. Ce graphe est générique et peut donc représenter n'importe quel type de modèles. Il se traduit aisément en XML et ne définit pas une sémantique spécifique tant que les noeuds ne sont pas typés.

Chaque noeud est associé à une liste ouverte d'arcs. Chaque arc a un type et un nom facultatif. Le couple (type, nom) est une clef unique d'identification de l'arc pour le noeud. Un arc est un connecteur pour le noeud vers d'autres noeuds. Ainsi un arc peut identifier une liste orientée de noeuds. La figure 4.3 décrit dans différentes couleurs ces listes orientées.

GAM fournit un ensemble d'outils pour contrôler le graphe :

- une fonction de stockage et de chargement : un graphe est archivé dans un ensemble cohérent de fichiers xml,
- subdivision de multi-document : un graphe est divisé en plusieurs documents liés. Les liens entre les documents sont automatiquement maintenus par GAM,

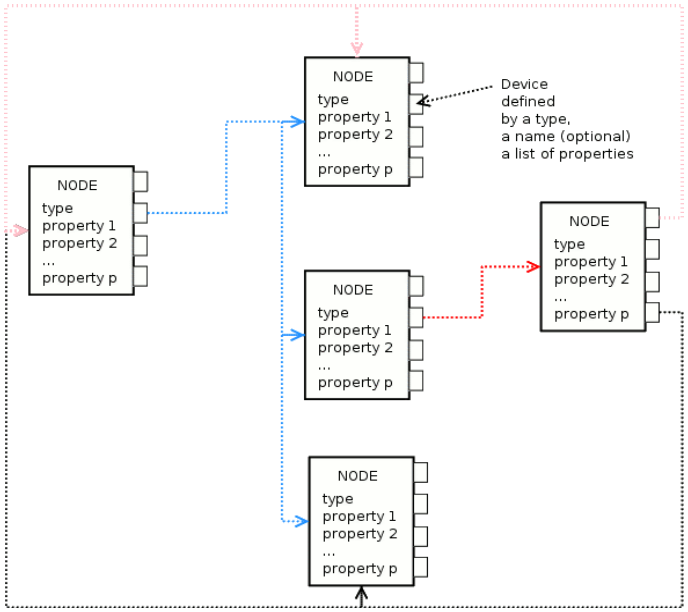

FIG. 4.3 – Modeleur de graphe (Noel, 2007)

- gestion de versionnement : un graphe a plusieurs versions au cours de son développement. Ces versions sont maintenues automatiquement,
- la gestion de la pile des modifications autorisant des UNDO-REDO infinis,
- différence entre deux graphes : la différence entre deux graphes est calculée automatiquement (simplement car chaque noeud a un identificateur unique).

Toutes ces fonctions s'appliquent à tous les graphes de manière générique.

4.2.2 Un méta-méta modèle

Le graphe défini précédemment n'a pas de structure pré-définie. GAM propose une API spécialisant les noeuds et les arcs de ce graphe pour lui associer une sémantique spécifique. Les graphes ainsi créés sont contraints par cette API.

La figure 4.4 présente le niveau 3 de l'environnement GAM. Le niveau 3 est un ensemble de concepts pour définir le méta-méta-modèle. Le modèle de méta-méta-modélisation utilisé dans GAM peut être vu comme une vision simplifiée du noyau MOF. Il est utilisé pour organiser les noeuds et les arcs du graphe. Les types de noeuds sont :

- type référencé : un ensemble fixe de noeuds se rapportant à un type de base : String, URL, réel, entier, booléen, UID (identificateur unique : pointeur vers un autre noeud),
- énumération : un type d'objet avec une valeur choisie dans une liste énumérée d'objets prédéfinis,
- class : la définition d'un type de type complexe d'objet. La classe est définie par un ensemble d'attributs et de méthodes exécutables,
- function : la définition d'une fonction active modifie le graphe. Des bindings ont été développés pour définir des fonctions dans divers langages existants (C, python, etc.),
- objet : une instance d'une classe, d'une énumération ou d'un type référencé,
- package : un ensemble des autres noeuds.

Ainsi un graphe peut être référé comme base de données orientée objet.

L'environnement GAM permet de représenter et d'exécuter des diagrammes de classes et d'objets UML. Cet environnement est un cadre qui gère un méta-méta-modèle semblable au MOF (OMG, 2002). Les objets et les classes peuvent être mis dans un même paquet. Dans GAM, il n'y a aucune séparation des niveaux habituels de méta-modèle et de modèle. Quand une classe du graphe est éditée, l'environnement met à jour automatiquement chaque objet conformément à cette classe. Cette propriété différencie GAM d'autres environnements puisque les modèles et méta-modèles sont construits et évoluent simultanément sans réification du méta-modèle.

Les modèles GAM peuvent être partagés entre des sessions distantes. Chaque fonction dans GAM est automatiquement convertie en fonction accessible via un protocole HTTP. Les mécanismes restent optimisés quand l'appel est local. Deux objets spéciaux permettent la gestion des sessions locales ou à distance :

- La session elle-même. Chaque application GAM doit initialiser une session unique. Une session contrôle une liste de zones de travail,
- La zone de travail se rapporte à un graphe qui préserve la cohérence des informations associées. Une zone de travail est locale ou distante. En mode local le graphe est chargé sur la session locale. Dans le cas du mode à distance, le graphe est contrôlé par une autre session lancée par une application distante. Pour l'utilisateur ce mécanisme est transparent.

Avec les connections à distance et un langage spécifique par domaine, GAM est une première étape pour formaliser des méta-modèles dédiés.

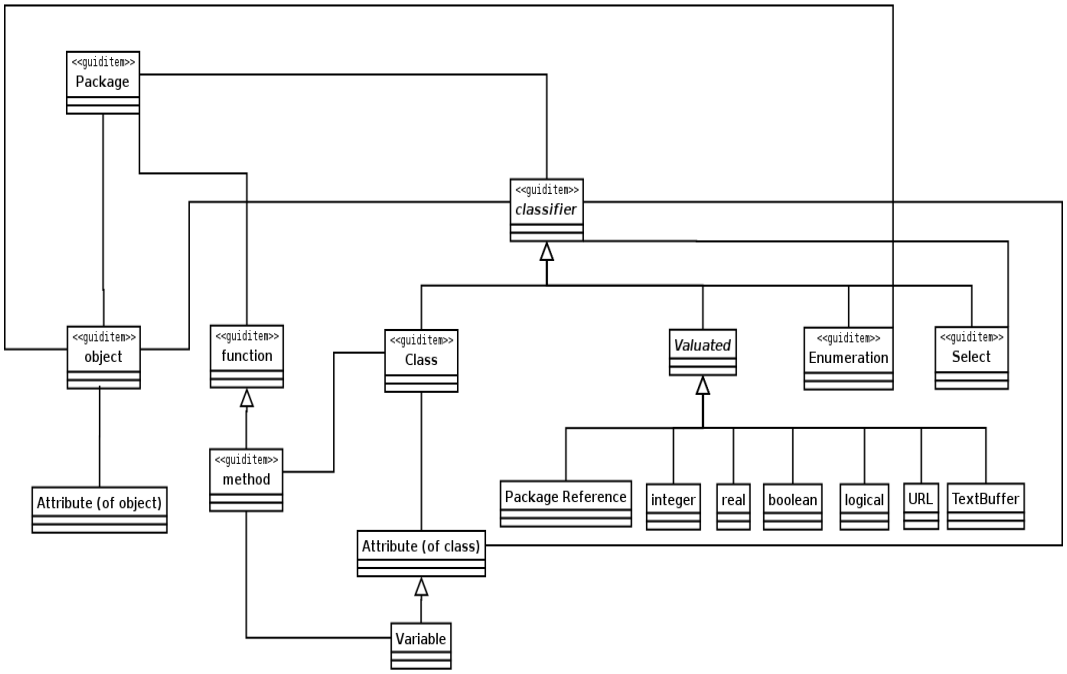

FIG. 4.4 – Le Méta-méta modèle GAM

4.2.3 Une interface utilisateur générique et adaptative

Un package spécifique a été développé dans GAM pour modéliser les interfaces graphiques utilisateurs (IHM). Ce modèle propose un ensemble de classes pour définir :

1. les *widgets* de l'IHM (bouton, image, arbre, boîte, fenêtre, etc.).

2. *contrôles* interrogeant un autre modèle géré par l'IHM. L'étiquette d'un bouton peut être le résultat d'une requête sur le modèle contrôlé.

3. *actions* provoquées par des événements associés aux *widgets* qui éditent le modèle contrôlé.

Ce méta-modèle permet la définition simple d'un modèle d'IHM. Les méthodes précédentes assurent le mapping de l'IHM dans une technologie existante (par exemple la technologie GTK (GTK, 2008) est employée), mais ceci est transparent pour l'utilisateur final.

4.2.4 Différence entre deux modèles : GAM-Diff

GAM-diff fournit un méta-modèle pour représenter la différence entre deux modèles de GAM. Un algorithme générique compare deux modèles GAM

et renvoie un nouveau modèle conforme au méta-modèle GAM-Diff. Un champ spécial est associé à chaque différence dans le méta-modèle de GAM-Diff pour permettre à des utilisateurs de négocier la fusion des deux modèles comparés. GAM-Diff est ainsi un support pour la synchronisation de sessions parallèles travaillant sur un même modèle.

Le cadre GAM permet le calcul et le stockage des différences entre deux modèles de GAM. Basés sur l'identificateur unique de chaque noeud, deux modèles, M1 et M2, sont facilement comparés pour extraire les noeuds insérés, supprimés ou modifiés. Quand un noeud est présent dans les deux modèles, ses propriétés sont balayées et comparées. Ces différences sont stockées dans un modèle conforme au méta-modèle GAM-Diff, lui-même décrit comme un *package* de GAM. Comme GAM-Toolkit pour les IHM, GAM-Diff démontre la généricité et la flexibilité des techniques de type IDM. Des modèles sont rapidement et simplement définis pour des contextes d'usages très spécifiques. La comparaison des modèles est un point clé dans le processus de synchronisation et de notification.

4.2.5 Architecture de l'environnement GAM

La figure 4.5 montre la décomposition en package de l'environnement GAM. «Graph Modeler» est le modeleur de graphe GAM qui permet la gestion de graphe. A un niveau plus haut, on trouve le noyau GAM : le module «$GAMM^3$». Ce niveau permet la définition et l'instanciation des modèles et des méta-modèles : il se charge aussi de la gestion des accès. Des modules complémentaires vont être associés à cette couche pour permettre des fonctionnalités de synchronisation ou de notification en conception. La couche «gam-ppo» définit une instanciation spécifique du modèle de collaboration PPO. Il est envisageable d'implémenter les modèles du chapitre 2, voir même par traduction d'EXPRESS d'importer des modèles STEP.

GAM-PPO Discuté au chapitre 2, le méta-modèle PPO a été conçu dans le projet IPPOP. On s'attend à ce qu'un tel modèle soit au centre de la collaboration entre les associés impliqués dans le développement d'un produit.

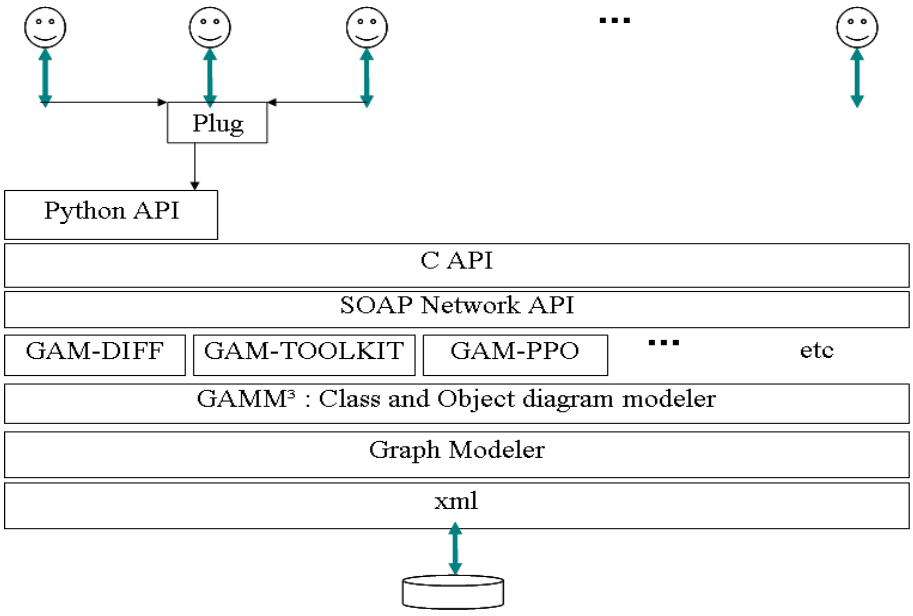

FIG. 4.5 – Décomposition en package

4.3 Outils de Conception assistée par Ordinateur

4.3.1 Cadre général

L'évolution des technologies informatiques a abouti au développement du domaine de la conception et de la réalisation de produits. La CAO outil majeur de conception de produit mécanique demeure en pleine évolution.

Une vision du modèle unique autour duquel toutes les applications métiers gravitent a toujours prévalu (Gardan, 1992). Des outils d'aide à la collaboration sont nécessaires pour permettre l'échange et le partage d'informations entre ces expertises métiers et le modèle CAO. La figure 4.6 situe la CAO comme un modèle de référence pour toutes les expertises métier dans l'entreprise qui du coup a besoin d'outils de collaboration comme les outils PDM ou autres environnements collaboratifs.

Les applications CAO sont aujourd'hui composées de plusieurs modules pour la représentation géométrique. La figure 4.7 montre les principaux modules :

- le modeleur *feature* gère les opérations spécifiées par l'utilisateur,
- le solveur de contraintes cherche à fournir une solution satisfaisante pour toutes les contraintes qui ont été posées,

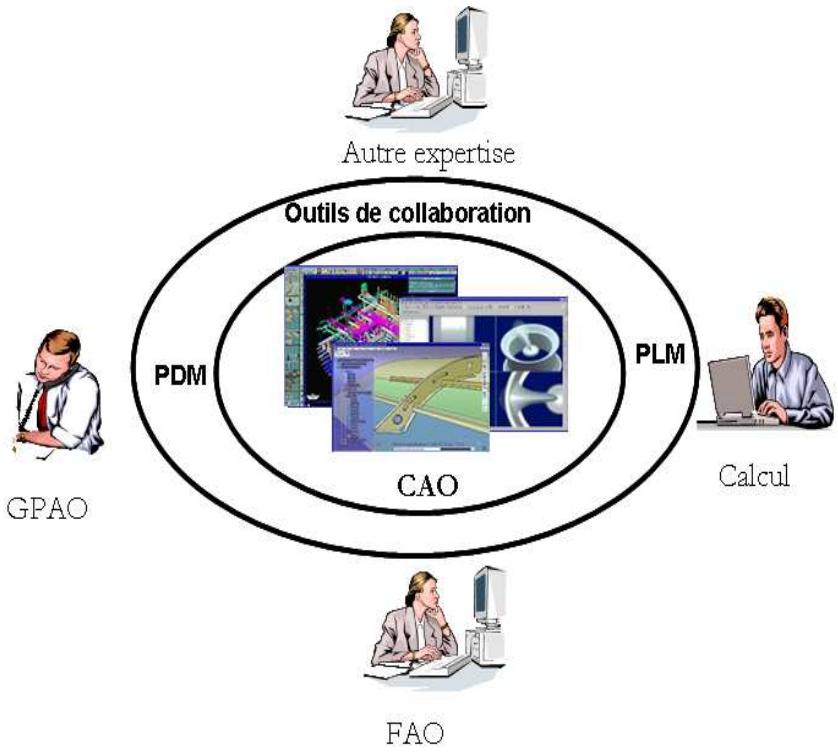

FIG. 4.6 – La CAO au coeur des toutes les activité et requiert des outils de collaboration

– le modeleur géométrique gère la représentation tridimensionnelle de l'objet (en général une représentation BRep).

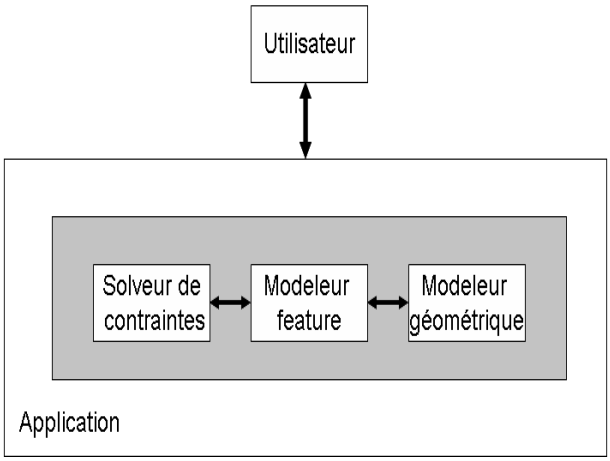

FIG. 4.7 – Vision simplifiée d'une application de CAO

(Wang *et al.*, 2002) classent les outils de conception en fonction de leur dépendance ou non au domaine d'application. Ils présentent aussi les cadres (*framework*) de conception (Figure 4.8) existants dans la littérature.

(Bettaieb, 2006) identifie quatre technologies potentielles permettant aux outils CAO de communiquer avec l'extérieur : les APIs natives, les Macros-commandes, les CAD Services et les formats standards d'échanges (détaillés au chapitre 2). Les APIs natives sont une interface de programmation d'application définie par un jeu de fonctions ou de méthodes utilisées pour accéder à certaines fonctionnalités d'un outil logiciel. Les Macros-commandes sont une variante en général simplifiée des APIs natives qui consiste à enregistrer dans une macro un ensemble d'actions qu'on souhaite exécuter à plusieurs reprises. Ces actions sont converties en code qui sera exécuté à chaque lancement de la macro.

Au total, il existe donc trois modes de communication entre outils CAO comme le montre la figure 4.9. Nous nous intéressons à la possibilité de communication offerte par les APIs natives afin d'extraire le modèle de données CAO. Elles sont en effet plus complètes que tous les autres APIs.

Les concepteurs ont souvent besoin de partager leurs données avec d'autres experts métiers afin de concevoir collaborativement le produit. Avec les formats standards les concepteurs sont obligés de partager toutes leurs données même si l'information utile est minime. Les APIs natives offrent un moyen de parcourir

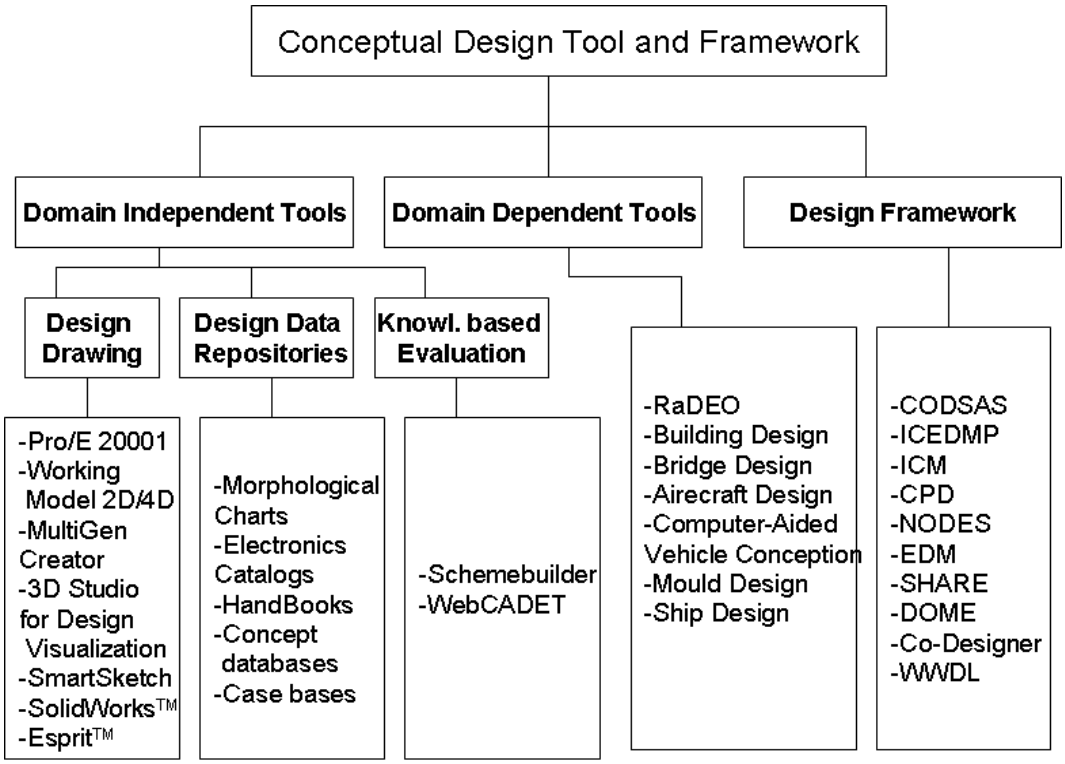

FIG. 4.8 – Classification des outils et des frameworks de conception

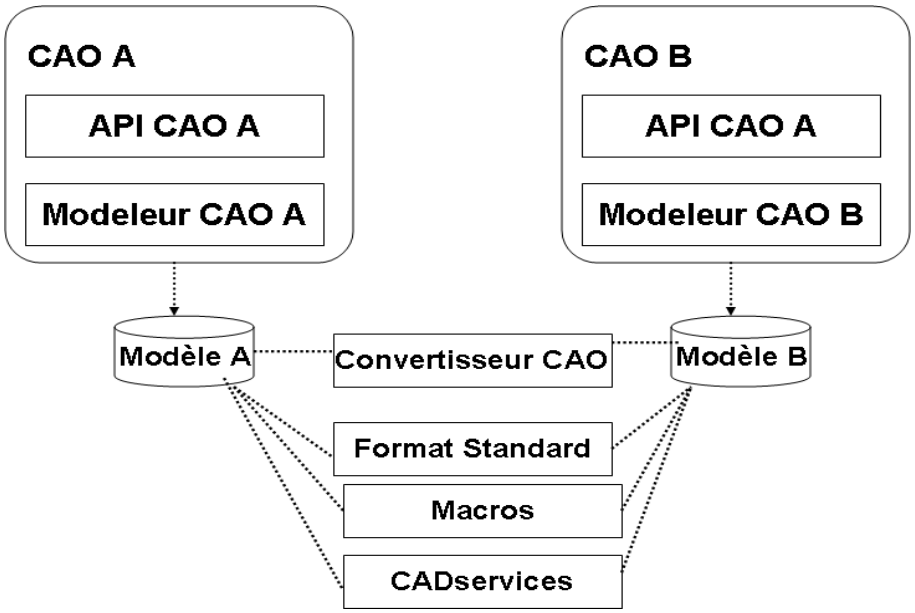

FIG. 4.9 – Les méthodes d'interopérabilité en CAO

et d'extraire toutes les informations d'un logiciel de CAO. Les équipes de concepteurs doivent se mettre d'accord sur les concepts clés à partager avec différentes expertises. Ces concepts permettent la construction du cadre de collaboration en définissant les méta-données à partager. Les développeurs utilisent ces concepts et les APIs natives offertes par le logiciel CAO pour développer des programmes qui permettent d'extraire automatiquement les données conformes aux méta-données de partage.

La figure 4.10 montre le mécanisme d'extraction d'informations dans un but de collaboration à travers des APIs natives. En effet, un outil CAO possède un modeleur géométrique et une API qui offre des fonctionnalités pour gérer son modèle. Les experts métiers définissent les concepts à partager en discutant avec d'autres experts. Ces concepts définissent le méta-modèle de partage. Ce méta-modèle peut être défini dans un environnement comme GAM et fournit une API de génération de modèle conforme à ce méta-modèle. En utilisant les fonctionnalités de lecture de données de l'API de l'outils CAO A et l'API fourni par GAM pour la génération de modèle, on extrait notre modèle de partage automatiquement. Une fois extrait, le modèle continue à être modifié.

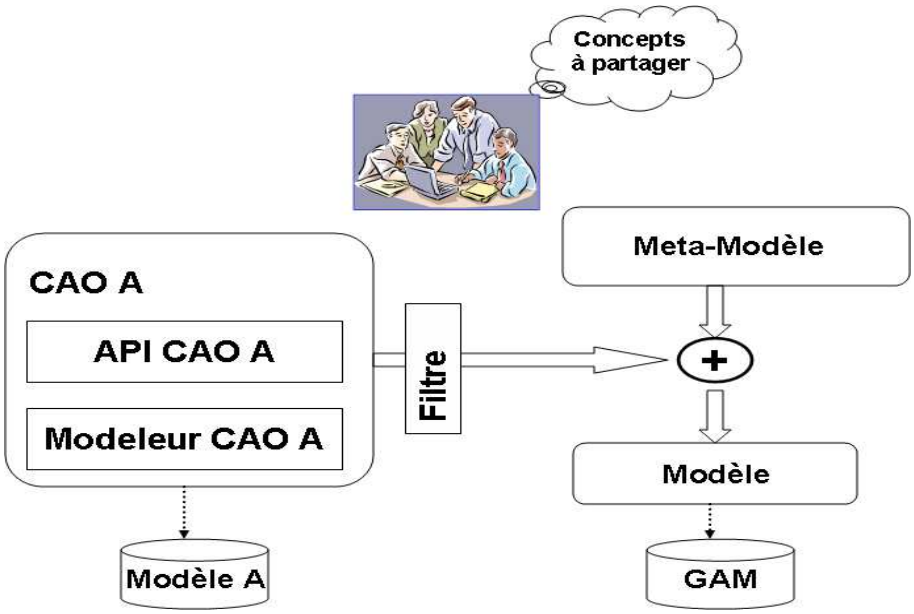

FIG. 4.10 – Première étape de l'interopérabilité : Extraction des informations à partager

Dans la section suivante, nous nous intéressons à l'application de ce mécanisme sur le logiciel CAO $SolidWorks^{TM}$. Les outils de CAO sont à l'origine conçus pour fonctionner en mode asynchrone. Nous illustrons nos propos dans la section suivante avec l'exemple du système $SolidWorks^{TM}$. A un premier

niveau d'analyse les outils CAO présentent des fonctions d'usage similaires. Les outils CAO sont en grande partie basés sur des concepts voisins. La démarche appliquée à $SolidWorks^{TM}$ sera donc une démarche répétable pour les environnements concurrents.

4.3.2 Application à SolidWorks

L'Interface de Programmation d'Application API SolidWorks (API) contient des centaines de fonctions qu'on appelle via Visual Basic (VBA) ou .NET. Ces fonctions fournissent un accès direct aux fonctionnalités de SolidWorks telle que créer une ligne ou vérifier les paramètres d'une surface. La manière la plus rapide et la plus facile de commencer à programmer avec l'API SolidWorks est d'enregistrer des macro-commandes $SolidWorks^{TM}$, qui contiennent les appels des API $SolidWorks^{TM}$ correspondant aux actions effectuées via l'interface utilisateurs. $SolidWorks^{TM}$ offre un ensemble d'APIs basées sur l'approche orientée objet. Ces API se présentent sous forme de diagramme de classes (avec des méthodes et des attributs) comme l'illustre la figure 4.11.

Dans la phase de conception, le concepteur construit la pièce comme une suite d'opérations. Ces opérations peuvent être des extrusions, des protrusions, etc. Ces éléments construisent la base de partage d'information significatifs pour un concepteur et tout autre expert métier manipulant le modèle CAO de la pièce à fabriquer. En effet, le concepteur ne va pas parler en terme de segments ou de géométrie de base pour indiquer une information à partager mais plutôt en terme de *feature* qui compose l'arbre de construction de la pièce.

Nous nous sommes intéressés à l'extraction de l'information utile à partir d'un modèle CAO donné qui est représenté dans la figure 4.12. Il s'agit d'un carter d'une pièce mécanique. Pour ce faire, nous avons identifié les concepts clés du partage à venir entre l'expertise CAO et l'expertise FAO. En effet, les éléments manipulés portent sur des opérations de construction du modèle CAO. Le concepteur et le fabricant communiquent éventuellement en discutant les entités du modèle comme les extrusions et les révolutions. La figure 4.13 présente une vision réduite de concepts clés potentiellement échangés dans un cadre de collaboration entre expertises métiers hétérogènes de conception et de fabrication. Nous réduisons volontairement l'ensemble de concepts pour permettre une implémentation réaliste dans le temps de notre travail. Les informations correspondantes doivent être extraites selon le mécanisme présenté dans la figure 4.10. Elles sont alors traduites depuis le code de CAO vers un environnement de partage d'information : en l'occurrence GAM.

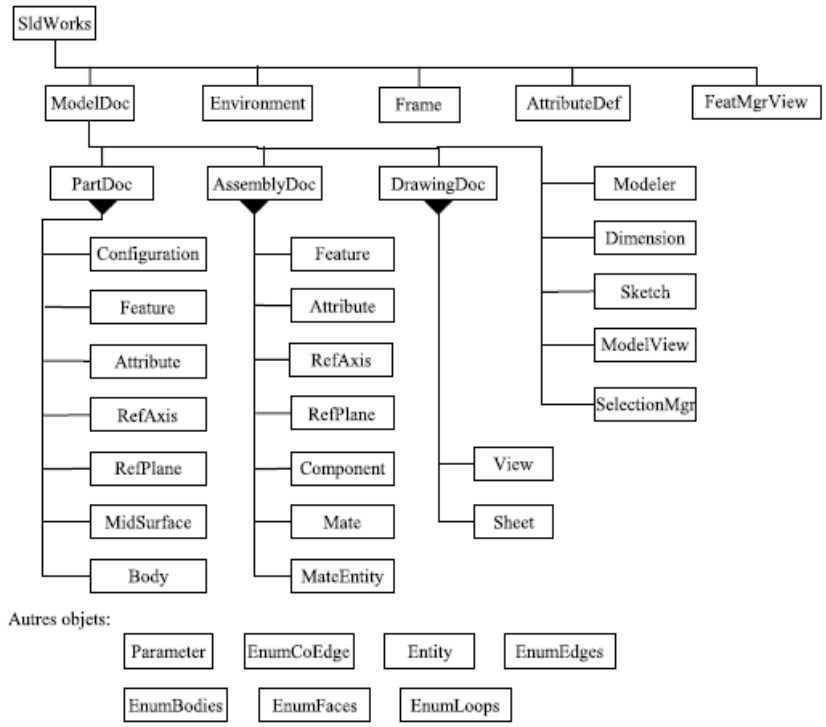

FIG. 4.11 – Méta-Modèle SolidWorks

Une première étape est d'extraire le modèle métier à partir des outils métiers. Cette étape devrait être effectuée par le développeur de logiciel métier. Le modèle CAO doit incorporer l'information telle que l'esquisse, les contraintes et les *features*, qui représentent l'intention de représentation. Actuellement, l'échange de données est réalisé par les formats standards ou propriétaires. L'arbre de construction est perdu et l'intention de représentation aussi. Afin de soutenir l'échange de données entre le modèle CAO et FAO nous limitons dans un premier temps l'extraction aux *features* :

- le modèle est alors un ensemble de *feature*,
- une *feature* est un ensemble de paramètres, de contraintes et de références,
- les paramètres représentent les variables associées à la *feature* (càd dimensions d'esquisse, distance d'extrusion, etc.),
- les contraintes identifient les relations entre les entités géométriques de la *feature* (perpendicularité, parallélisme, etc.) et les références représentent les entités géométriques explicites qui sont nécessaires pour définir le *feature* (les plans d'attache, les arrêtes de référence, etc.).

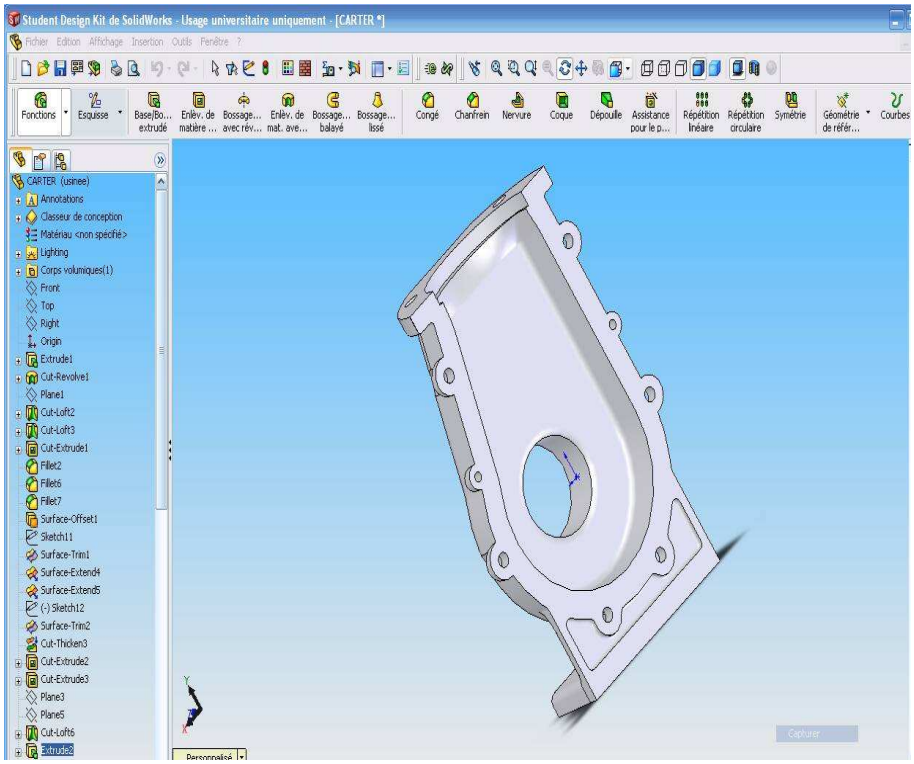

FIG. 4.12 – Représentation de la pièce CAO

Le logiciel métier devrait proposer une fonction d'export vers le modèle PPO, fonction paramétrée pour que l'utilisateur final puisse sélectionner l'étendue de l'export. Les experts spécifient l'information à extraire du modèle métier. L'information à extraire est l'information requise pour la collaboration. L'expert est libre de filtrer l'information à partager ou à garder confidentielle. Quelques étapes doivent être suivies pour lier des modèles métiers dans un environnement comme GAM. Les développeurs d'applications métiers décrivent leurs méta-modèles métier dans l'environnement GAM. Ce méta-modèle permet la génération automatique d'une bibliothèque qui liée à la bibliothèque de l'outil métier permet l'instantiation du méta-modèle sous la forme d'un modèle GAM.

L'API créée vérifie si le modèle GAM existe déjà. Si le modèle existe, les modifications apportées dans les modèles métiers sont propagées dans le modèle GAM, sans modifier les anciennes entités. Sinon, le modèle requis est produit. Cette vérification est nécessaire pour que l'environnement GAM contrôle le versionnement des modèles.

Dans l'API $SolidWorks^{TM}$ nous nous intéressons donc à la classe *feature*, ses attributs et ses méthodes. Nous générons un modèle de données conforme à l'environnement GAM. Ceci suit les étapes suivantes :

- dans l'environnement GAM on définit le méta-modèle métier dédié à la collaboration. Ce méta-modèle contient les concepts clés que le concepteur pourra partager avec d'autres expertises métier hétérogènes. La création de ce méta-modèle dans GAM offre un ensemble de fonctions pour créer des modèles conformes à ce méta-modèle. Ces fonctions sont générées automatiquement par GAM. La création de ce méta-modèle est à la charge du développeur de l'outil métier, qui spécifie un nouveau driver d'export, au même type qu'un driver STEP, IGES, SET,
- avec l'API native $SolidWorks^{TM}$, le développeur parcourt le modèle CAO. Chaque fois qu'une *feature* est rencontrée, il génère via l'API GAM une entité conforme au méta-modèle décrit à l'étape précédente. Le résultat est un modèle GAM conforme au méta-modèle précédemment défini,
- une modification du modèle CAO est prise en compte lors de ce mécanisme. En effet, l'API SolidWorks teste si l'élément est déjà créé. Si l'élément est déjà créé, des commandes de modification du modèle GAM sont appelées, sinon des commandes de création de l'élément sont exécutées. Cette étape permet donc une mise à jour du modèle GAM plutôt qu'un export. Cette procedure va plus loin que les exports traditionnelles vers STEP, IGES, etc,

Le figure 4.14 présente le modèle de $SolidWorks^{TM}$ extrait automatiquement à partir des outils de $SolidWorks^{TM}$ dans le format GAM conformément au méta-modèle que nous avons élaboré pour tester notre procédure.

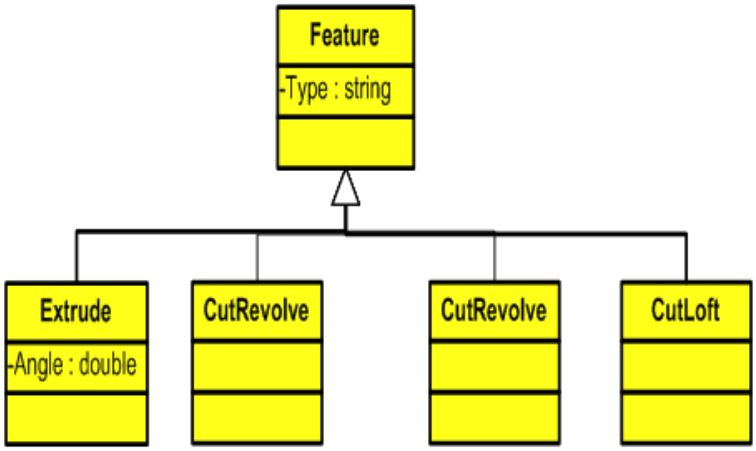

FIG. 4.13 – Méta-Modèle SolidWorks

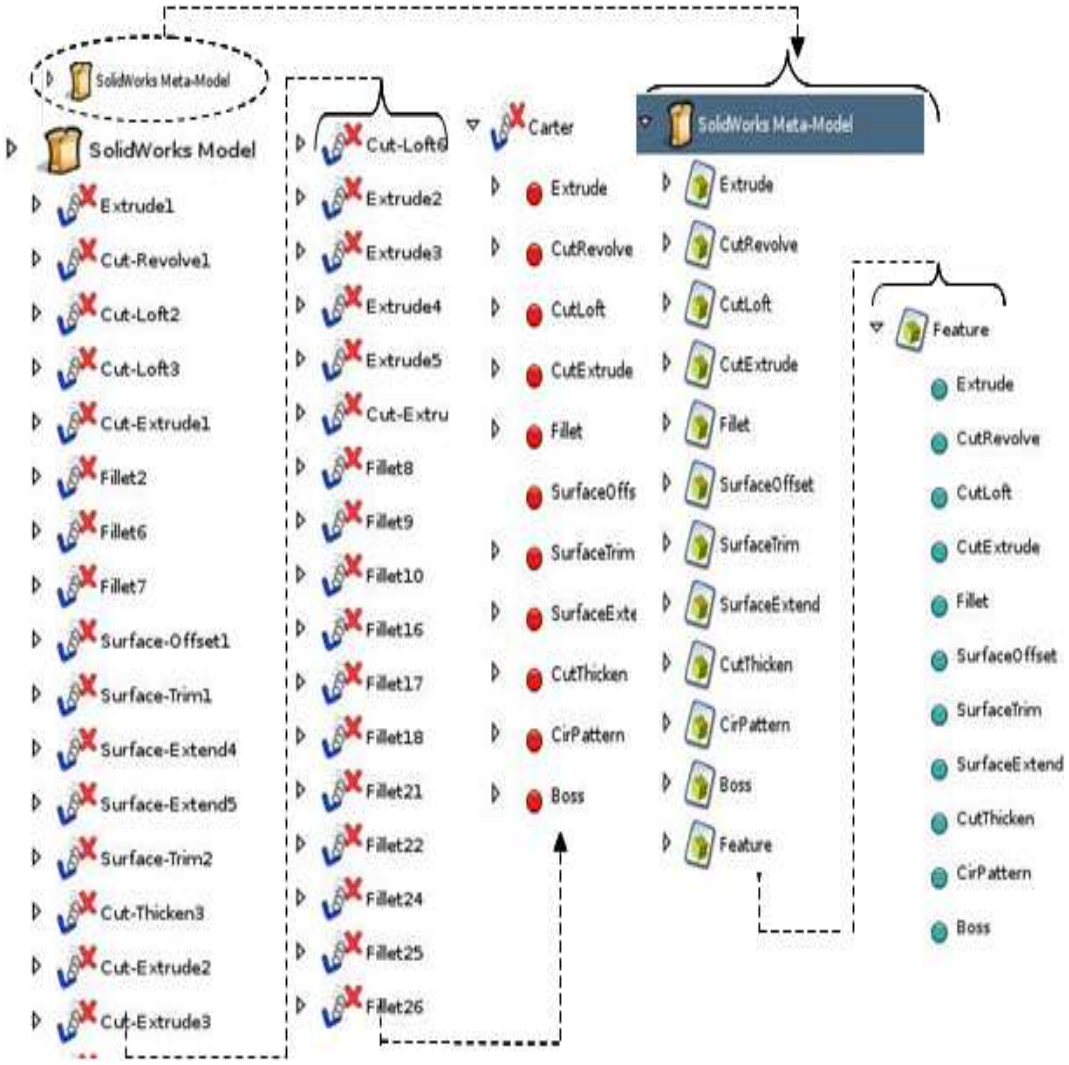

FIG. 4.14 – Exportation du modèle $SolidWorks^{TM}$ dans l'environnement GAM

4.4 Outils de Fabrication Assistée par Ordinateur

La gamme de fabrication est obtenue à l'aide d'un progiciel de Fabrication Assistée par Ordinateur (FAO) : le modèle 3D du produit à fabriquer est obtenu par la CAO. Cette modélisation en trois dimensions de la pièce à réaliser est ensuite usuellement « exportée » dans un fichier intermédiaire en utilisant un standard d'échange comme IGES, STEP, VDA, DXF ou autre. La plupart des outils de FAO sont capables de relire directement les fichiers des grands fournisseurs de CAO. Dans certains logiciels, la CAO et la FAO sont complètement intégrées et ne nécessitent pas de transfert (ProEng, CATIA, etc).

La modélisation 3D étant importée sur le progiciel de FAO puis relue par celui-ci, il est possible de passer à la programmation des trajectoires d'outils, coeur de l'activité de la FAO. Le programmeur crée les trajectoires en respectant les choix d'outils, les vitesses de coupe et d'avance, et les stratégies d'usinage à mettre en oeuvre. Le progiciel de FAO «plaque» les trajectoires des outils choisis sur la modélisation 3D et enregistre celles-ci sous forme d'équations. Les logiciels les plus évolués sont ensuite capables de reproduire graphiquement (visualisation volumique) l'action des outils dans la matière, permettant ainsi au programmeur de vérifier ses méthodes d'usinage et d'éviter a priori les collisions sur les machines-outil ou d'autres défauts basiques.

La dernière étape consiste, depuis le programme de FAO ainsi élaboré et vérifié, à générer les blocs ISO pour la machine outil. Ce programme est appelé un Post-Processeur. Il existe également des programmes indépendants pour effectuer la vérification directement à partir des blocs générés par le Post-Processeur. Le fichier ISO obtenu est transmis à la MOCN (Machine Outil à Commande Numérique), puis exécuté par cette machine

4.4.1 Partage d'information depuis $Esprit^{TM}$

$Esprit^{TM}$ (logiciel édité par la société Esprit) permet l'usinage des solides, des surfaces, et des géométrie filaires. Il permet la préparation de l'usinage en créant des programmes d'usinage pour des MOCN. Ces fonctions de base sont :

- dessiner ou importer la géométrie d'une Pièce : importer tous les fichiers CAO de format courant ou créer sa propre géométrie, surface NURBS, surface, ou solide,

- créer des séquences d'usinage : $Esprit^{TM}$ décompose tous ces usinages en séquences correspondant à l'usage d'un outil et d'une prise de pièce spécifiques,
- déclaration du parc machine : il s'agit de définir le type de machine, d'inclure la définition de tous les mouvements de la machine,
- créer des usinages : usiner les pièces en fonction des choix du spécialiste fraisage, tournage, tournage multiaxial, et électroérosion à fil selon les règles de l'art,
- simuler des usinages : simuler l'enlèvement de matière et les conditions de coupe et visualiser en temps réel l'environnement machine complet en mode solide,
- convertir des usinages en programme CN : $Esprit^{TM}$ fournit un post-processeur universel personnalisable, pour générer le programme CN de n'importe quel type de machine outil à commande numérique.

Dans le logiciel de fabrication $Esprit^{TM}$, le fabricant importe le modèle CAO et travaille dessus afin de réaliser les séquences d'usinage. L'import du modèle CAO se fait via des standards d'échanges. Les imports de fichiers natifs apparaissent dans les dernières versions seulement. Le fabricant manipule le modèle CAO pour construire des séquences d'usinage qui se basent sur les *features* géométriques mais qui ne conservent pas de lien explicite avec celle-ci. Les concepts clé de la FAO sont les opérations, les outils et les *features*. Les opérations sont appliquées à des *features* d'usinage. La figure 4.15 montre des concepts majeurs de la FAO. Les concepts clés sont les opérations d'usinage, les outils technologiques qui les réalisent ainsi que les *features* d'usinage. Les *features* de la FAO sont les séquences d'usinage. Reconstruire le lien entre la CAO et la FAO revient à construire le lien entre les *features* d'usinage et les *features* géométrique. La figure 4.15 est une vue très simplifiée du diagramme de classe $Esprit^{TM}$, mais nous avons délibérément fait cette simplification pour permettre une implémentation démonstrative de nos démarches. La figure 4.16 montre un modèle $Esprit^{TM}$ conforme au méta-modèle simplifié dans l'environnement GAM.

L'API $Esprit^{TM}$ est beaucoup plus riche et comporte une variétés de concepts résumant toute l'étendue de l'application métier. La figure 4.17 montre un extrait de l'API $Esprit^{TM}$. Nous nous intéressons donc à la classe *feature*, opération et outils technologiques ainsi que leur attributs.

Les étapes de réalisation d'un export et de mise à jour du modèle métier dans une plateforme cible comme GAM sont décrites dans la figure 4.18 et se résument aux étapes suivantes :

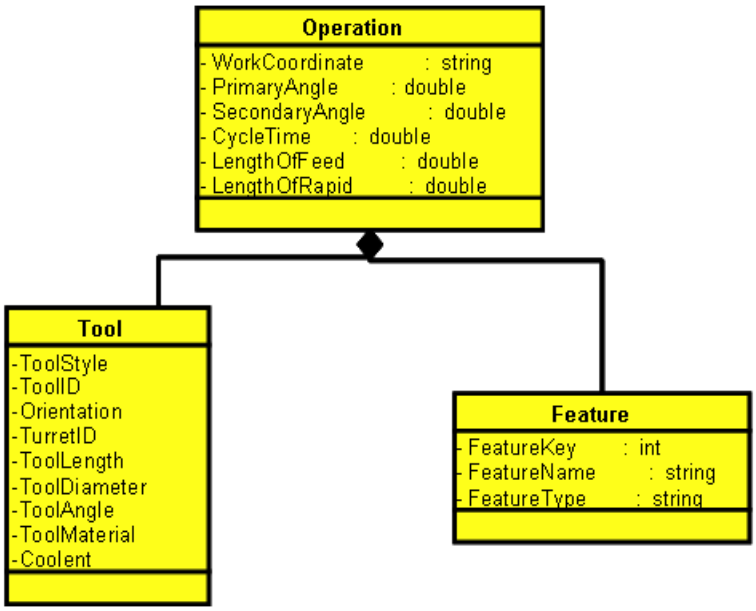

FIG. 4.15 – méta-modèle $Esprit^{TM}$

1. le développeur d'$Esprit^{TM}$ crée le module d'export en ayant comme entrée le méta-modèle métier de l'application. Le module d'export a comme sortie le méta-modèle d'export,

2. l'utilisateur travaille avec $Esprit^{TM}$ et crée des modèles en l'occurrence le modèle d'usinage du carter,

3. l'utilisateur sélectionne les concepts à partager à partir du méta-modèle d'export,

4. l'export prend en entrée les concepts à partager, le modèle métier et le module d'export comme ressource pour produire le modèle exporté GAM.

4.5 Synthèse

Les experts métier travaillent avec des outils métier pour formaliser leurs expertises. Ces outils métier contiennent donc une partie de l'expertise métier. Ces modèles instancient des concepts de haut niveau (niveau méta-modèle). L'environnement GAM nous permet de définir les concepts clés que l'expert métier veut partager avec ses collaborateurs. Conformément à ces concepts et en utilisant l'API de l'outil métier et l'API de l'environnement GAM, le modèle métier est extrait. Nous avons développé un programme permettant

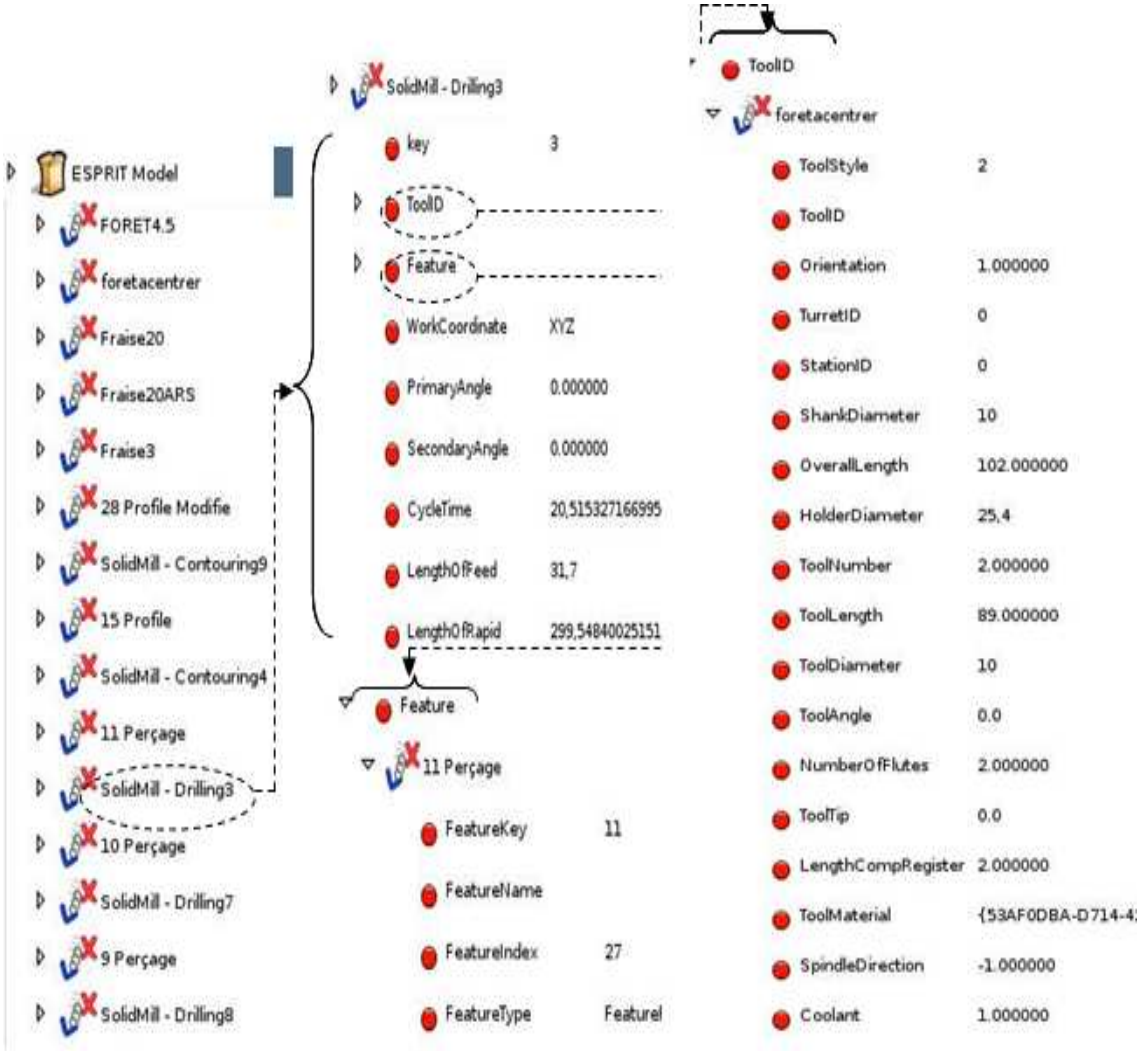

FIG. 4.16 – Instantiation du modèle $Esprit^{TM}$ dans l'environnement GAM

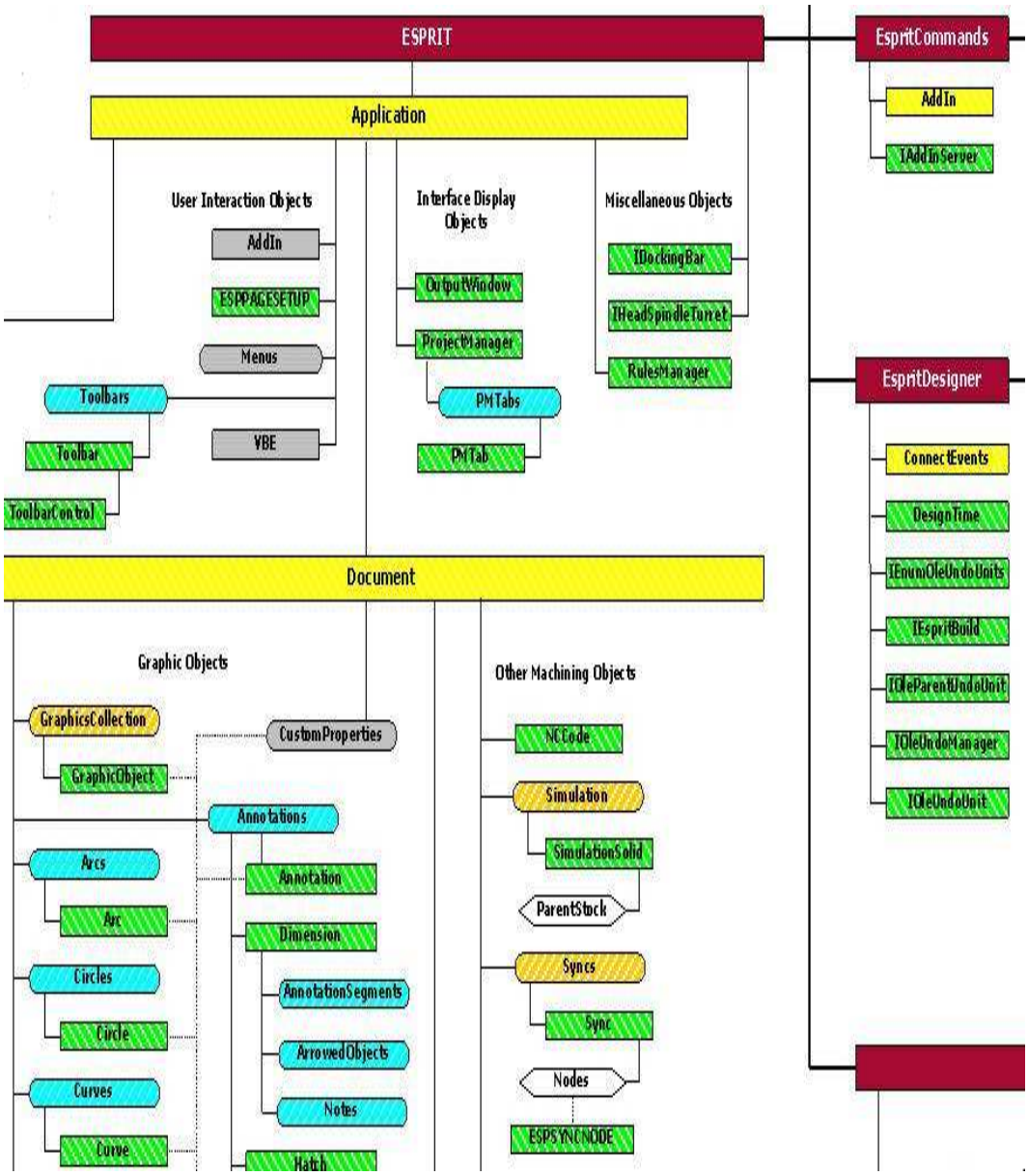

FIG. 4.17 – Un extrait de l'API $Esprit^{TM}$

le passage de l'outil métier à l'environnement GAM. Cet algorithme gère le versionnement des modèles exportés. Une première étape vers la collaboration est franchie, puisque tous les modèles sont ramenés dans un environnement unique.

Les modèles métier sont extraits dans le format de la plateforme de partage GAM. Dans cette plateforme le modèle commun de partage PPO (Produit Process Organisation) est implémenté. Pour communiquer, une vision d'un modèle commun a été choisi. Pour dialoguer avec ce modèle commun, des transformations de modèles entre modèle métier et modèle partagé s'imposent. Cette étape sera étudiée en détails dans le chapitre suivant.

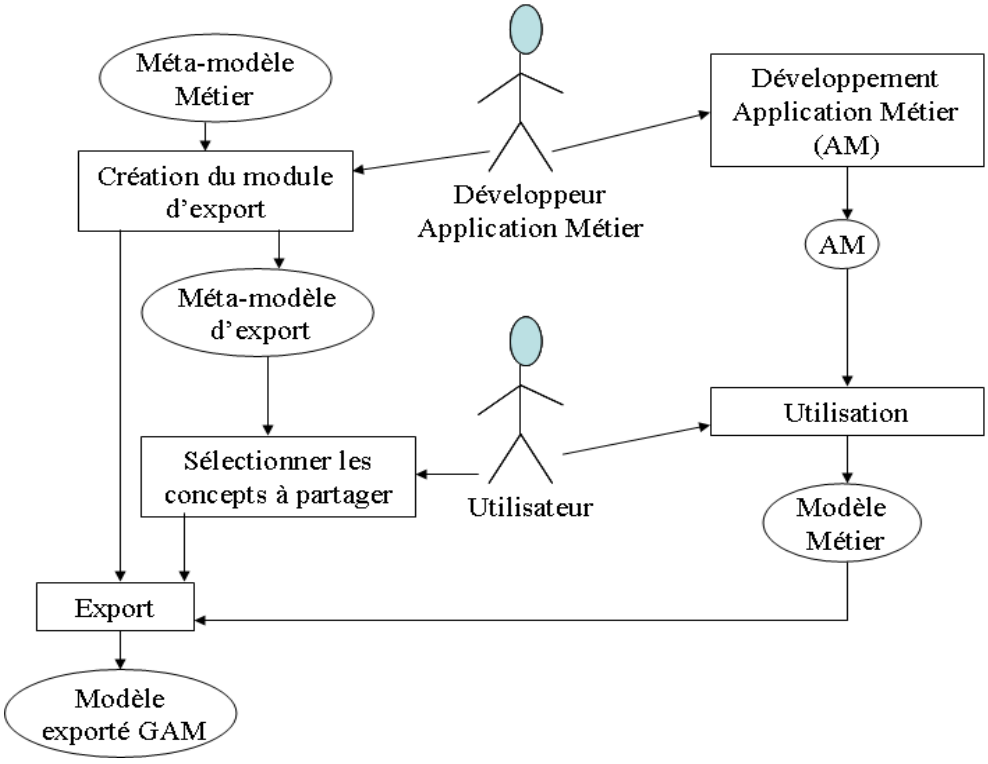

FIG. 4.18 – Le processus générique de développement de driver modèle métier

Chapitre 5

Synchronisation des modèles

Sommaire

Le chapitre précédent a présenté la démarche mise en place pour coordonner un modèle métier avec un environnement de méta-modélisation (en l'occurrence nous utilisons l'environnement GAM). Si toutes les expertises métiers procèdent ainsi, nous obtenons un ensemble de modèles GAM conformes à autant de méta-modèles tous exprimés dans le même environnement. Disposer de ces modélisations métiers dans un environnement unique comme GAM ne permet pas encore la synchronisation entre modèles métiers.

Dans ce chapitre nous présentons une approche pour la synchronisation entre des modèles experts hétérogènes. Elle utilise les techniques d'export du chapitre 4. Une étude de cas est donc appliquée aux outils $Esprit^{TM}$ et $SolidWorks^{TM}$ via le modèle de partage PPO (Produit Processus et Organisation).

La solution proposée relève d'une approche fédérée (voir section 1.1.1). Toutes les expertises métiers conservent leurs modèles usuels. Elles interagissent à travers un modèle fédérateur. Ce modèle permet l'échange et le partage d'informations entre différents modèles métiers hétérogènes en offrant une modélisation flexible de tous les concepts à partager. Nous nous concentrons sur les concepts relatifs à la description du produit.

La section 1 de ce chapitre présente l'architecture proposée pour synchroniser les modèles métiers hétérogènes. Cela sous entend des mécanismes plus évolués que les mécanismes de traduction souvent mis en avant dans les approches dirigées par les modèles. La section 2 présente les modèles et les méta-modèles utilisés dans le module de synchronisation. La section 3 décrit les différents modules de l'architecture de synchronisation. La section 4 présente un bilan de fonctionnement de l'architecture globale du module synchronisation proposé. La section 5 conclue sur les limites de la démarche et donne des pistes d'évolution du système proposé.

5.1 Architecture d'un système de synchronisation

Les modèles obtenus en provenance de plusieurs outils métiers sont hétérogènes. Le chapitre 3 a détaillé une solution de l'Ingénierie Dirigée par les Modèles (IDM) entre autre déclinée via la MDA. Notre but est de faire interopérer des outils métiers hétérogènes. Pour ce faire, on doit réaliser la synchronisation entre les différents modèles hétérogènes issus de ces outils. Nous avons vu qu'il existe différentes approches pour l'intéropérabilité : intégrée, unifiée et fédérée. Nous choisissons une approche fédérée pour l'intéropérabilité et les méthodologies de l'IDM pour réaliser la synchronisation entre modèles métiers hétérogènes.

5.1.1 Approche fédérée

D'après (Heimbigner et McLeod, 1985) (Sheth, 1999) (Park et Ram, 2004), une fédération est une architecture dans laquelle des composants sont légèrement connectés afin de partager et échanger l'information. Les composants

représentent différents utilisateurs, applications, ou autres composants d'un système d'information. L'architecture fédérée doit satisfaire deux conditions contradictoires : les composants doivent être autonomes ; cependant, les composants doit pouvoir réaliser un degré raisonnable de partage d'informations. Dans notre cas, le composant est l'outil métier. L'approche fédérée appliquée à une collection d'outils métier, en l'occurrence les outils de CAO et de FAO, conduit aux deux propriétés suivantes :

1. chaque outil métier détermine les données qu'il souhaite partager avec d'autres outils. La fédération est en charge du partage d'informations. Chaque outil doit spécifier l'information à rendre disponible.
2. un outil doit avoir la « liberté d'association » en respectant la fédération. Les composants doivent pouvoir s'inscrire ou quitter dynamiquement la fédération. Un outil doit pouvoir modifier son interface de données partagées, ajoutant de nouvelles données et retirant l'accès aux données précédemment partagées.

Chaque outil dans la fédération commande ses interactions avec d'autres outils au moyen d'un schéma d'exportation et d'un schéma d'importation. Le schéma d'exportation spécifie l'information qu'un composant partagera avec d'autres composants, alors que le schéma d'importation spécifie l'information non locale (ou véhiculaire) qu'un composant souhaite manipuler. La fédération fournit des mécanismes pour partager des données, pour combiner l'information de plusieurs composants.

5.1.2 Architecture de l'environnement

L'architecture générale de l'environnement de travail est représentée dans la figure 5.1. Plusieurs applications métier peuvent se connecter à cet environnement à travers le réseau en développant un plugin d'export de leurs modèles métiers comme décrit au chapitre 4. Chaque application doit donc fournir un minimum de degrés d'interopérabilité avec l'environnement de travail. Ainsi, à chaque application métier un modèle métier dans le format de l'environnement est créé dans l'espace privé affecté au client. Le client dispose donc d'un espace privé de travail. Chaque fois qu'il aura besoin de partager son modèle avec d'autres clients, il passe par l'espace public. Dans notre cas, le modèle de données sur lequel se base l'espace public de partage est le *modèle de fédération PPO* et plus directement sa partie produit.

Chaque expert métier établit des correspondances possibles entre ses concepts et le modèle de fédération PPO. L'environnement propose un module de synchronisation qui assiste, en première étape, la synchronisation du modèle métier et du modèle de fédération.

Plusieurs experts métiers travaillent simultanément et essayent de partager leurs données via le modèle de fédération PPO. Ceci mène à plusieurs versions qu'il faut synchroniser pour garantir un niveau d'échange et de partage fiable. L'environnement fournit aussi un moyen de synchroniser les modèles conformes au méta-modèle de fédération de l'espace partagé.

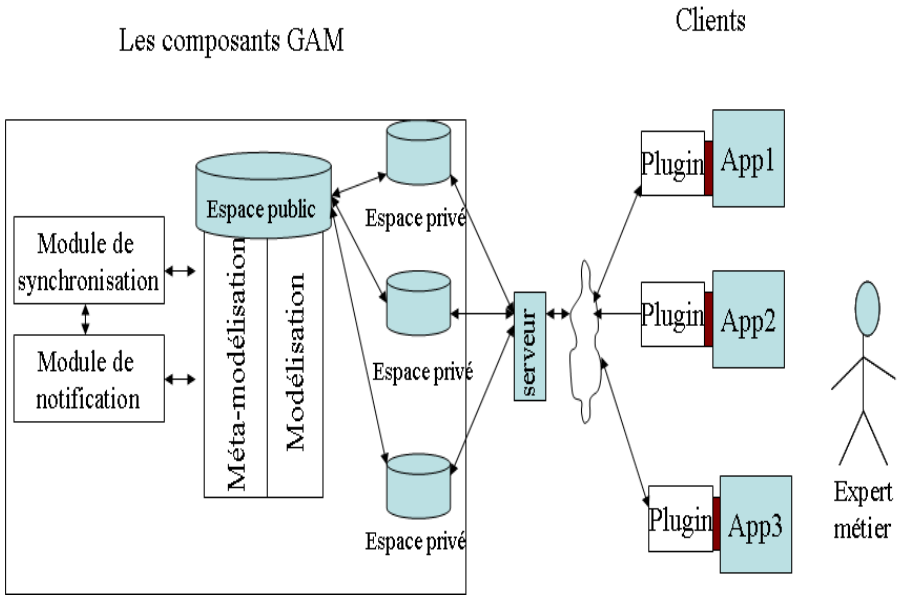

FIG. 5.1 – Architecture de l'environnement

Pour implémenter cette architecture, nous avons utilisé l'environnement GAM. Dans cet environnement nous contribuons par le développement du module de synchronisation qui permettra aux différents expertises métiers de se connecter et de partager leurs données. Le développement de ce module est possible grâce aux différents éléments que l'environnement GAM offre (détaillés au chapitre 4).

Dans la section suivante nous présentons le scénario de conception collaborative sur lequel porte notre étude de cas. Dans ce scénario, les outils métiers mis en oeuvre sont les deux applications métiers de CAO : $SolidWorks^{TM}$, et de FAO : $Esprit^{TM}$.

5.1.3 Mise en oeuvre dans un scénario de conception

Dans un processus de conception, les outils CAO et de FAO sont utilisés pour concevoir et préparer la fabrication du produit. Le scénario de conception collaborative que nous développons ici illustre le problème de l'interopérabilité entre les outils de CAO et de FAO dans le cadre d'un processus de fabrication. Nous avons deux acteurs principaux dans ce scénario :

– le concepteur est responsable de la représentation du produit dans un modeleur géométrique de CAO. Le concepteur prend en compte les besoins client et les convertir en une représentation géométrique,

– le fabricant est responsable de la fabrication des produits.

Nous proposons un environnement partagé basé sur PPO pour assurer l'interopérabilité : PPO est choisi comme modèle pour le partage et l'échange d'informations.

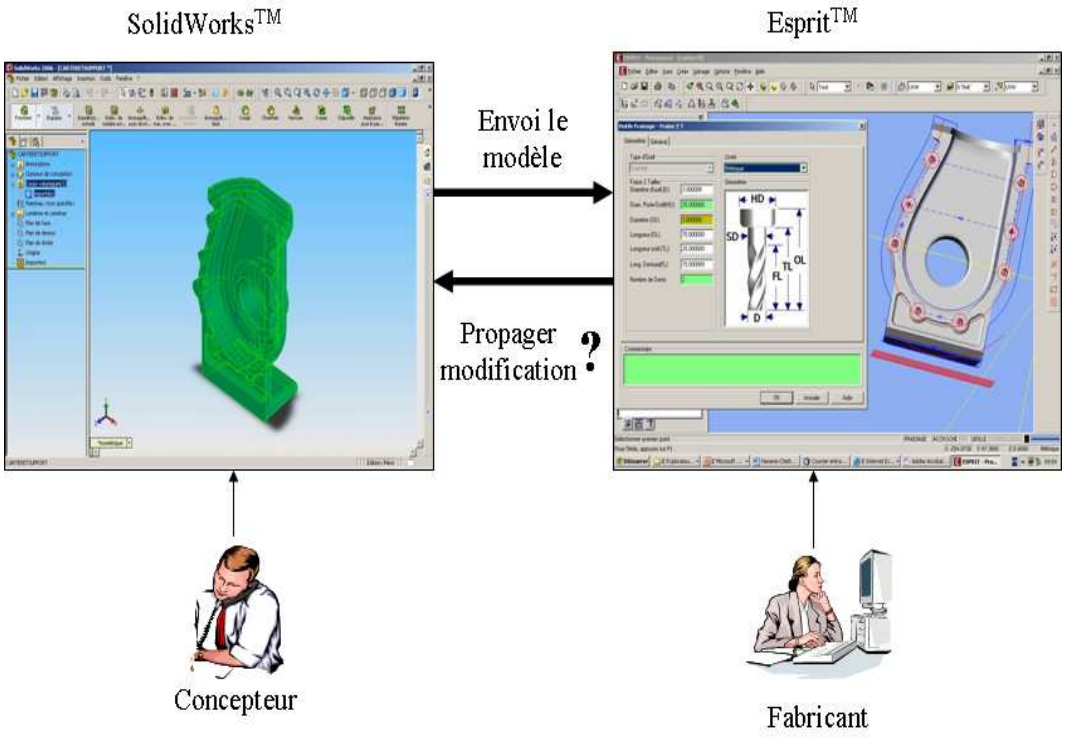

FIG. 5.2 – Destruction des liens lors de l'envoi du modèle de $SolidWorks^{TM}$ à $Esprit^{TM}$

Le concepteur modélise un produit avec les besoins et les fonctionnalités requises dans le cahier des charges. Il essaie d'exprimer l'intention de conception en réalisant un modèle CAO. Une fois constitué, le modèle CAO du produit est en général envoyé via un format standard au fabricant. Le fabricant manipule un modèle CAO en construisant dessus des séquences d'usinage sans aucun lien avec les features géométriques.

Le fabricant suit un processus que nous réduisons en première approche aux activités suivantes :

– il reçoit le modèle CAO,

- il choisit les machines et outils nécessaires à la réalisation de la pièce,
- il définit les séquences d'usinage et les modes de prise de pièce,
- il simule l'usinage des objets,
- il produit un fichier ruban (code de commande),
- il exporte le dossier de production vers l'atelier.

Le fabricant est souvent amené à proposer des modifications de la conception initiale pour prendre en considération ses capacités de production et optimiser le coût. Il peut modifier le diamètre d'un perçage en fonction de sa base d'outils. Ce diamètre a en général été fixé dans la représentation CAO par la dimension d'une feature de type extrusion, sans que ce soit une contrainte de conception forte.

La figure 5.2 met en évidence qu'il n'y a en général aucun soutien à la propagation d'une telle information d'un outil comme $Esprit^{TM}$ vers un outil de CAO comme $SolidWorks^{TM}$. Pourtant les suggestions doivent être intégrées au modèle CAO original et renvoyées au fabricant. L'information de fabrication n'est pas attachée au modèle CAO géométrique. Chaque demande de modification a besoin d'une modification du modèle CAO et sa re-émission. Ceci conduit à des redondances, à des pertes d'information, et à des boucles de conception qui demeurent informelles.

Les données échangées sont statiques et à sens unique. Elles n'incorporent pas tous les détails tels que les esquisses, les contraintes et les features, qui participent à la représentation CAO.

L'architecture de l'environnement de travail GAM appliquée au scénario de travail est illustré dans la figure 5.3. Les plugins permettent aux concepteurs métier d'avoir leurs modèles métiers sous format de l'environnement GAM. Ainsi on obtient deux modèles : Esprit-GAM et SolidWorks-GAM. Les concepteurs voulant partager leur données doivent passer par l'espace commun régi par le méta-modèle de fédération PPO. Une transformation et une mise à jour de modèles est nécessaire. Ainsi on obtient deux nouveaux modèles : Esprit-PPO et SolidWorks-PPO. L'espace commun contrôle deux modèles métiers homogènes conformes à PPO, à synchroniser via un processus de synchronisation de modèles homogènes.

Nous présentons donc en parallèle le détail de l'approche de synchronisation ainsi que l'application à notre cas d'étude. Le but étant de véhiculer l'information et les mises à jour de la FAO vers la CAO.

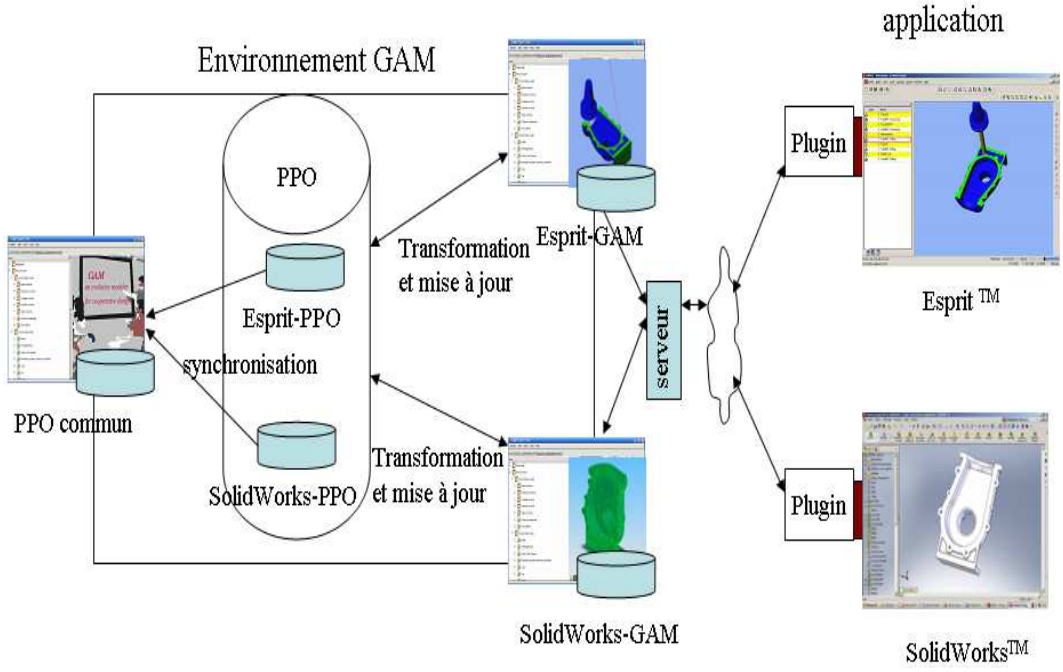

FIG. 5.3 – Architecture du module de synchronisation appliquée au scénario

5.1.4 Méthode de synchronisation

Le module de synchronisation prend en entrée deux modèles métiers hétérogènes sources et produit en sortie un modèle conforme au méta-modèle cible. La figure 5.4 montre que dans un module de synchronisation entre deux représentations métier hétérogènes deux processus de transformation (et de mise à jour) sont nécessaires. Ces deux processus génèrent deux modèles cibles homogène à synchroniser par le sous module «synchronisation des modèles homogènes».

La figure 5.5 résume l'architecture du module de synchronisation proposé. Les correspondances sont établies entre le méta-modèle source et le méta-modèle de fédération, c'est-à-dire entre l'ensemble des concepts du modèle source et celui du modèle de fédération. Le processus de transformation prend en entrée un modèle conforme au méta-modèle source et produit en sortie un ou plusieurs autre(s) modèle(s) conforme(s) au méta-modèle de fédération, en utilisant des règles préalablement établies.

On envisage de connecter un outil au système en modélisant le méta-modèle associé et en définissant les correspondances entre ce méta-modèle et le méta-modèle de fédération. Une fois cette architecture mise en place, on obtient deux modèles conformes au méta-modèle de fédération. Il faut synchroniser ces modèles homogènes pour obtenir une version cohérente de l'information de partage réunissant les deux vues métiers. Un processus de synchronisation

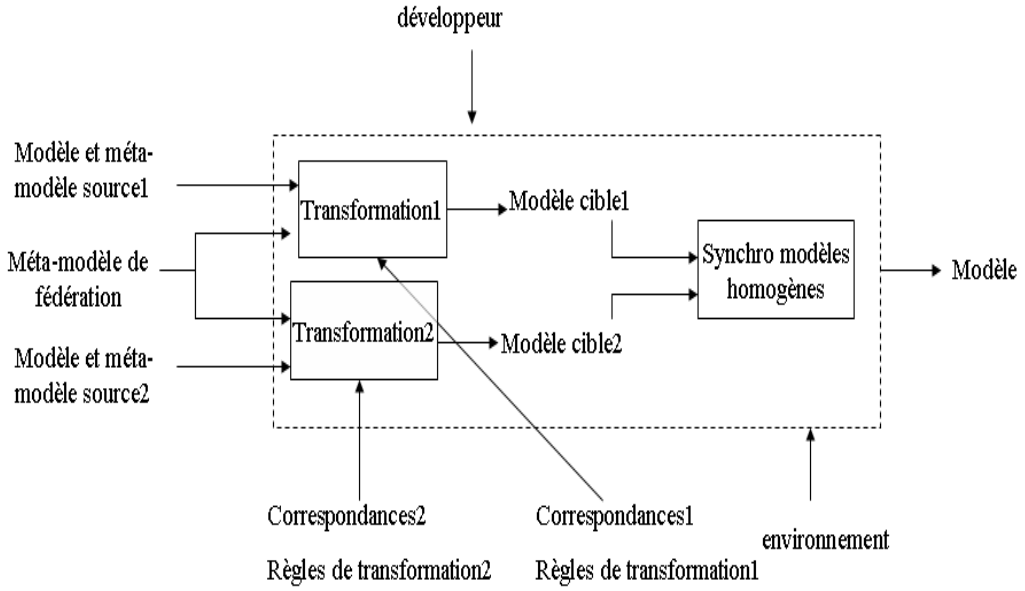

FIG. 5.4 – Module de synchronisation

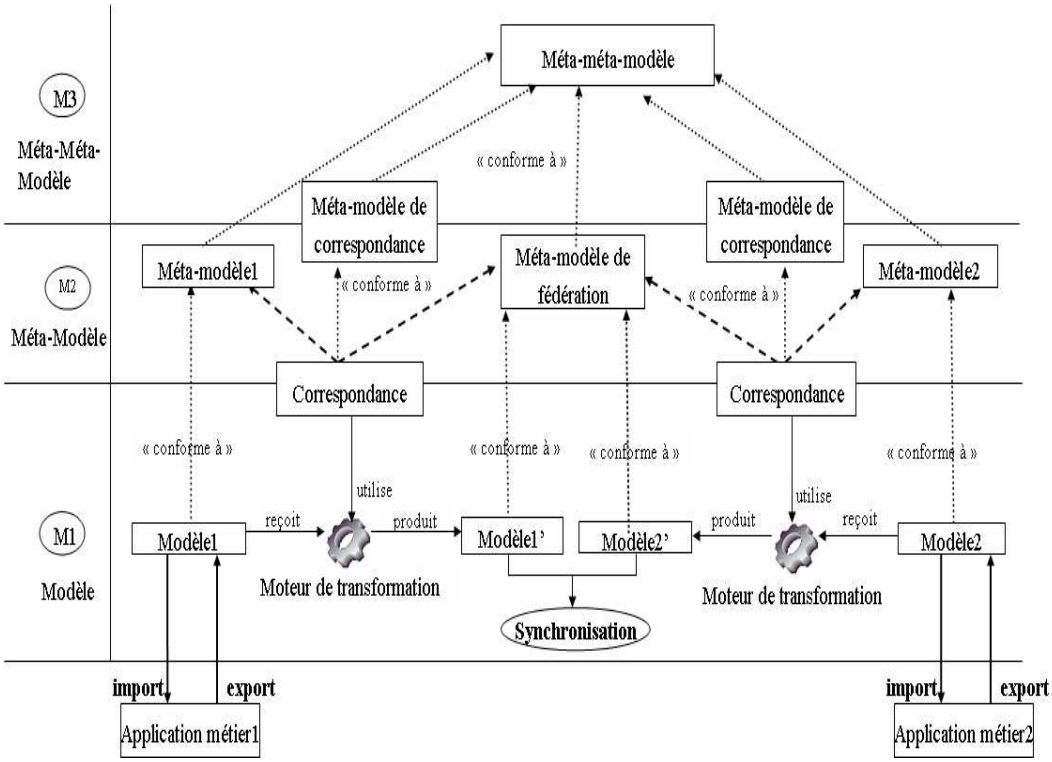

FIG. 5.5 – Architecture du module de synchronisation

de modèles homogènes est donc nécessaire. Le processus de transformation des modèles sources doit prendre en compte la propagation des modifications apportées au deux modèles sources Modèle1 et Modèle2 pour compléter cette architecture et la faire évoluer d'une architecture de transformation à une architecture de synchronisation.

Cas d'étude

L'application au cas d'étude donne la figure 5.6. Les applications métier mises en oeuvre sont $SolidWorks^{TM}$ et $Esprit^{TM}$. Le méta-modèle de fédération est celui de PPO. Dans cette figure on note :

- un niveau M1 : les modèles $SolidWorks - GAM$ et $Esprit - GAM$ obtenus par export à partir des outils métiers, et par transformation de modèles,
- un niveau M2 : les méta-modèles $SolidWorks^{TM}$ et $Esprit^{TM}$ ainsi que le méta-modèle de fédération PPO,
- un niveau M3 : à ce niveau se trouve le méta-méta-modèle. Dans notre cas, celui spécifique à GAM.

  Le méta-modèle de correspondance manipule des éléments du niveau méta-modèle et du niveau méta-méta-modèle. Dans la figure 5.6, on a placé ce méta-modèle entre les deux niveaux méta-modèle et méta-méta-modèle pour expliciter cette propriété. Les applications métier génèrent des modèles métier. L'environnement GAM permet la manipulation d'une telle architecture.

Les correspondances (mapping) sont établies entre les méta-modèles source et le méta-modèle de fédération. Le moteur de transformation exécute cette correspondance. Le résultat des transformations est un modèle conforme au méta-modèle de fédération PPO. La figure 5.6 montre que les résultats des transformations sont les modèles Esprit-PPO et SW-PPO qui doivent être synchronisés.

En résumé notre module de synchronisation se compose de trois parties :

1. les modèles et méta-modèles :
   - les modèles et méta-modèles métiers,
   - méta-modèle de fédération,
   - méta-modèle de correspondance,
2. les opérateurs ou moteurs qui contient :
   - le module d'import,

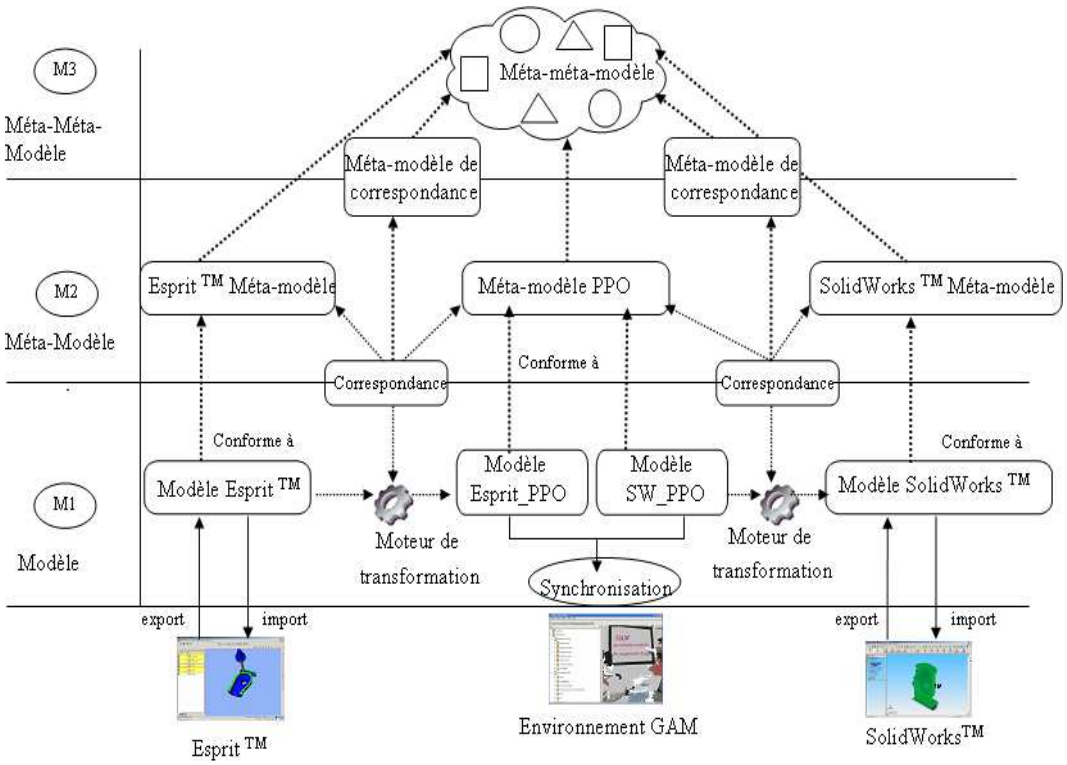

FIG. 5.6 – Architecture du module de synchronisation : cas d'étude

- le module d'export,
- le moteur de transformation,

3. le processus de synchronisation de modèles homogènes.

Dans les sections suivantes nous détaillons chacune de ces parties en montrant la mise en oeuvre dans le cadre de notre cas d'étude.

5.2 Les modèles et les méta-modèles utilisés

Un modèle métier contient un ensemble d'objets et des relations entre ces objets. Chaque modèle est défini comme un ensemble d'objet O_i, auxquels nous associons un nom et un type ainsi que des attributs (nom, valeur) :

$$M = \left\{ \left(nom_i, O_i, type_i, \left(nomattribut_{i_j}, valeurattribut_{i_j} \right)_{j \in [1,n]} \right) \right\}_{i \in [1,m]}$$

Le type d'un objet O_i peut être :
- atomique : boolean, integer, char, string, float, ...,
- composé : définit la classe que l'objet instancie.

Nous utilisons, le concept de *classe* pour définir un méta-modèle MM. Chaque entité du méta-modèle est définie par un ensemble de caractéristiques ou attributs.

$$MM = \left\{ \left(nom_i, C_i, \left(nomattribut_{i_j}, typeattribut_{i_j} \right)_{j \in [1,n]} \right) \right\}_{i \in [1,m]}$$

Le *typeattribut* est atomique : boolean, integer, char, string, float, . . .

Dans le cas de paradigme Entité-Relation (E-R), les modèles produit sont des représentations R_i conformes à des $schema_i$. Chaque concepteur possède sa propre interprétation cognitive des objets partagés. La figure 5.7(b) décrit les interactions en conception usuellement identifiées au niveau cognitif (Bettaieb, 2006). Cette interaction entre concepteurs identifiée au niveau cognitif peut être formalisée par les correspondances entre les concepts métier établis au niveau méta-modèle (figure 5.7(a)).

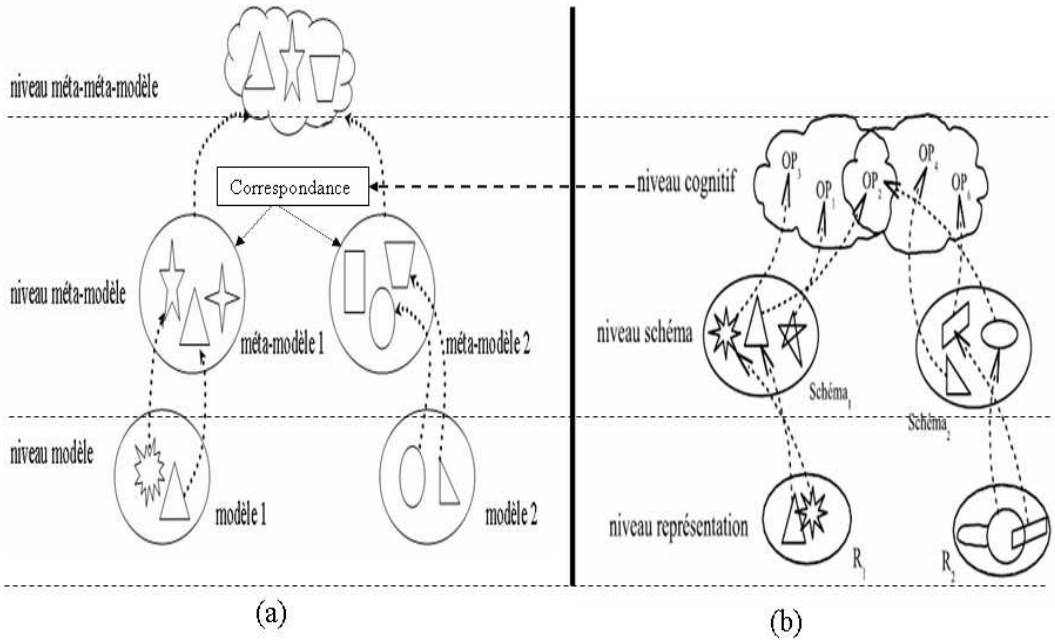

FIG. 5.7 – Interaction au niveau cognitif traduite par des correspondances

5.2.1 Les modèles et méta-modèles métiers

Le méta-modèle métier représente les concepts à partager et n'est donc qu'une partie de l'information qui réside dans le modèle de l'application métier. Dans notre cas d'étude un méta-modèle de partage $SolidWorks^{TM}$ est spécifié par le concepteur. Le module d'export permet de générer le modèle métier conforme au méta-modèle $SolidWorks^{TM}$ formulé dans l'environnement GAM. De même pour l'outil $Esprit^{TM}$ ou un méta-modèle de partage $Esprit^{TM}$ est

spécifié par le fabricant et un modèle conforme à ce méta-modèle est généré conformément au chapitre 4.

5.2.2 Le méta-modèle de fédération

Nous choisissons d'employer le Méta-Modèle PPO (IPPOP, 2008b) comme *méta-modèle de fédération* pour les échanges d'information autour des modèles métiers. Ce méta-modèle est détaillé dans le chapitre 2. La figure 5.8(a) décrit le méta-modèle produit de PPO dans GAM.

La figure 5.8(b) rappelle la partie produit du modèle PPO et montre que la description de chaque entité du modèle produit de PPO peut être affinée par un qualificatif « Alternative », « Commun » ou « Vue ». Un Composant Commun (CC) est un composant dont la description est la même pour tous les experts de conception, un Composant Alternative (AC) est décomposé en composants alternatifs définissant des configurations différentes, et un Composant Vue (VC) est décomposé en plusieurs représentations spécifiques des expertises différentes.

Les concepteurs et les fabricants interagissent l'un avec l'autre au travers de ce modèle. Pour pouvoir interagir avec le méta-modèle de fédération, chaque application métier doit établir des correspondances entre son méta-modèle métier et le méta-modèle de fédération.

5.2.3 Le méta-modèle de correspondance ou Mapping

Le méta-modèle de correspondance ou *mapping* permet la mise en correspondance des différents éléments de deux modèles. Le *mapping* est établi au niveau des méta-modèles métier et du méta-modèle cible. Dans notre approche fédérée, le méta-modèle cible est le méta-modèle fédérateur. La mise en correspondance est une opération fondamentale dans le processus de synchronisation. Elle s'effectue au niveau méta-modèle pour spécifier à un haut niveau d'abstraction les concepts qui sont équivalents dans deux modèles hétérogènes. Cette mise en correspondance est définie par la formalisation suivante qui couple deux entités associées :

Si N est l'ensemble des entités d'un modèle

$$N = \left\{ \left(nom_n, C_n, \left(nomattribut_{n_i}, typeattribut_{n_i} \right)_{i \in [1,a]} \right) \right\}_{n \in [1,h]}$$

et si R est l'ensemble des entités de l'autre modèle

$$R = \left\{ \left(nom_r, C_r, \left(nomattribut_{r_j}, typeattribut_{r_j} \right)_{j \in [1,b]} \right) \right\}_{r \in [1,k]}$$

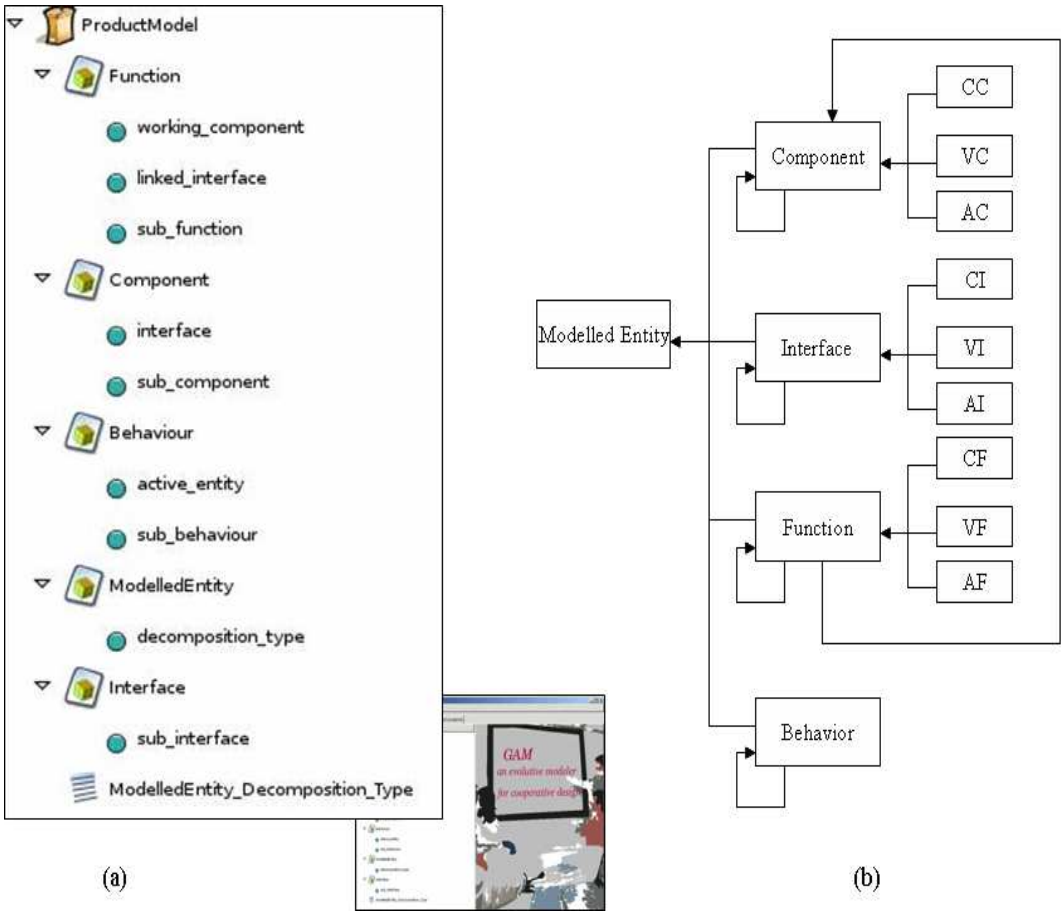

FIG. 5.8 – Méta-modèle produit de PPO

alors *mapping (N,R)* représente l'ensemble des correspondances entre N et R

$$mapping(N,R) = \{(C_m, (left_m, typeleft_m), (right_m, typeright_m),$$
$$((attributleft_{m_i}, typeattributleft_{m_i})_{i\in[1,a]},$$
$$(attributright_{m_j}, typeattributright_{m_j})_{j\in[1,b]}, F_m)\}_{m\in[1,k]}$$
$$mapping(N,R) = \{C_m, (C_n, C_r), (nomattribut_{n_i\, i\in[1,a]}, nomattribut_{r_j\, j\in[1,b]}, F_m)\}$$

La figure 5.9 représente les différentes possibilités de correspondances entre méta-modèles. Les correspondances peuvent être un vers un, un vers plusieurs et plusieurs vers un. Une fonction F_m permet d'exprimer le lien entre les attributs des entités liées :

– un vers un :

$$F_m(nomattribut_{n_1}) = nomattribut_{r_1}$$

– 1 vers n :

$$F_m(nomattribut_{n_1}) = \{nomattribut_{r_j}\}_{j\in[1,b]}$$

– n vers 1 :

$$F_m(\{nomattribut_{n_i}\}_{i\in[1,a]}) = nomattribut_{r_{j1}}$$

F_m peut être une équation mathématique qui prend un ou plusieurs attributs en entrée (*left*) et les combine pour produire un ou plusieurs attributs cibles (*right*).

Le méta-modèle de correspondance permet la spécification des inter-relations (correspondances) entre les éléments de deux méta-modèles. Nous présentons un méta-modèle de correspondance ou de *mapping* comme un ensemble de correspondances entre classes. Les correspondances entres classes peuvent être affinées par des correspondances entre attributs. La classe «Correspondance Classe» est modélisée par une classe-association caractérisant la relation entre les «Classe *left*» et les «Classe *right*». La classe «Correspondance Attribut» est modélisée par une classe-association caractérisant la relation entre les «Attribut». On gère l'historique de ce méta-modèle avec l'attribut «version» du «*mapping*». (Lopes, 2005) définit un méta-modèle qui spécifie des correspondances bidirectionnelles. Cet élément est souhaitable mais difficile à réaliser, car il suppose d'inverser les fonctions F_m.

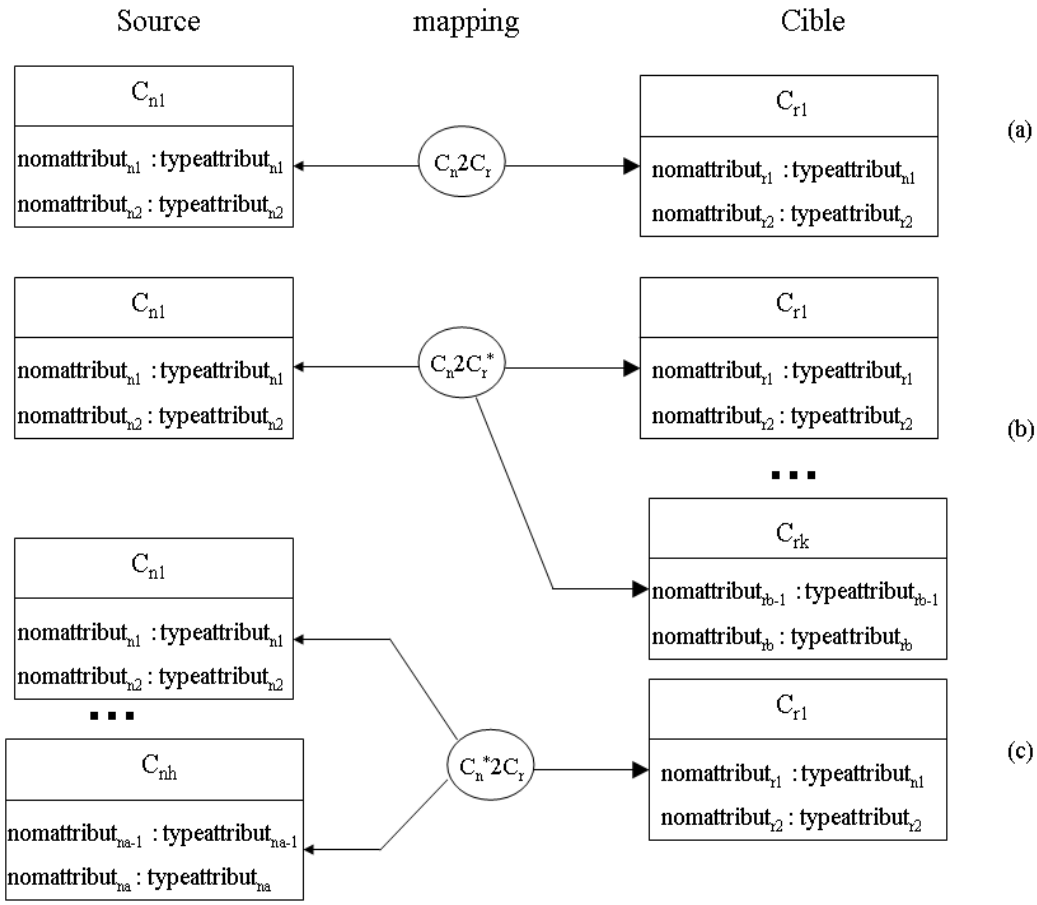

FIG. 5.9 – Les types de correspondances : (a) un-vers-un, (b) un-vers-plusieurs, (c) plusieurs-vers-un

Dans ce méta-modèle, nous considérons que la correspondance est unidirectionnelle. Dans notre cas, un méta-modèle source est mis en correspondance avec un méta-modèle cible. Ainsi, de manière générale, les deux méta-modèles sont désignés par le méta-modèle gauche et le méta-modèle droit. La figure 5.10 présente une première approche de méta-modèle pour la spécification de correspondances. Ce méta-modèle est constitué des concepts suivants :

- *mapping* est l'élément principal contenant toutes les correspondances entre les méta-modèles source et cible,
- *Correspondance Classe* permet de spécifier la correspondance entre deux ou plusieurs classes des méta-modèles sources et cibles avec un sens donné *left* ou *right*,
- *Correspondance Attribut* identifie la correspondance entre deux ou plusieurs attributs de l'élément «Classe» et ceci dans un sens donné *Left* ou *right*. La *Correspondance Attribut* a une méthode qui spécifie les règles de transformations entre les attributs,
- *Classe* identifie une classe source ou cible du méta-modèle,
- *Attribut* identifie l'attribut de la classe *Classe*.

Le méta-modèle de correspondance est constitué de :

- *Correspondance Classe* et *Correspondance Attribut* qui appartiennent au niveau méta-modèle,
- *Classe* et *Attribut* qui appartiennent au niveau méta-méta-modèle.

Le besoin de concepts de niveau méta–méta-modèle est nécessaire pour référencer des éléments au niveau méta-modèle. L'environnement GAM permet de mélanger les niveaux modèles, méta-modèle et méta-méta-modèle. Les différents niveaux évoluent simultanément. On peut donc utiliser des concepts du niveau méta-méta-modèle au niveau méta-modèle.

La figure 5.11 montre un exemple de mise en relation de deux modèles : un modèle «Modèle1» conforme au méta-modèle «Méta-Modèle1» et un modèle Modèle2 conforme au méta-modèle «Méta-Modèle2». Cette mise en relation est réalisée au niveau méta-modèle au travers d'un Mapping. Les concepteurs identifient les correspondances au niveau des concepts et donc au niveau méta-modèle. Les correspondances sont alors effectuées entre des classes et des attributs.

La figure 5.12 montre le méta-modèle de mapping spécifique à l'exemple.

L'objet *Corres*(0,0) contient deux correspondances entre attributs : (*attr(0,1,1)*, *attr(1,0,2)*) et (*attr(0,1,2)*,*attr(1,0,1)*). La méthode transformation() spécifie les règles de transformations entre les attributs. Ici $transformation1(attr(0,1,1)) = attr(1,0,2)$ et

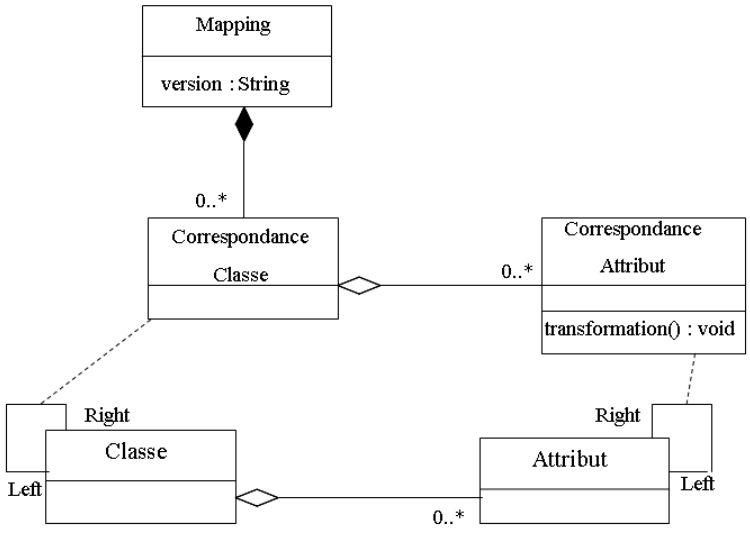

FIG. 5.10 – Méta-modèle de correspondances

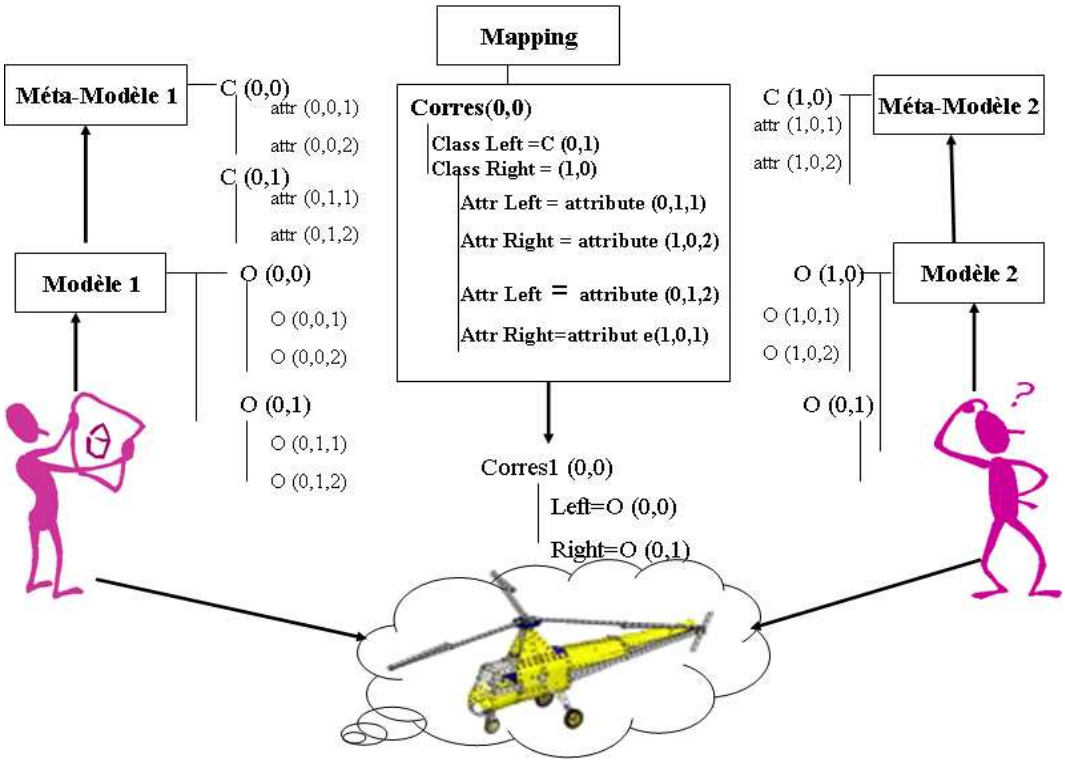

FIG. 5.11 – Exemple de mise en relation entre deux modèles

$transformation2(attr(0,1,2)) = attr(1,0,1)$. La correspondance au niveau modèle est une instance du Mapping niveau méta-modèle. On peut donc spécifier les correspondances entre objets.

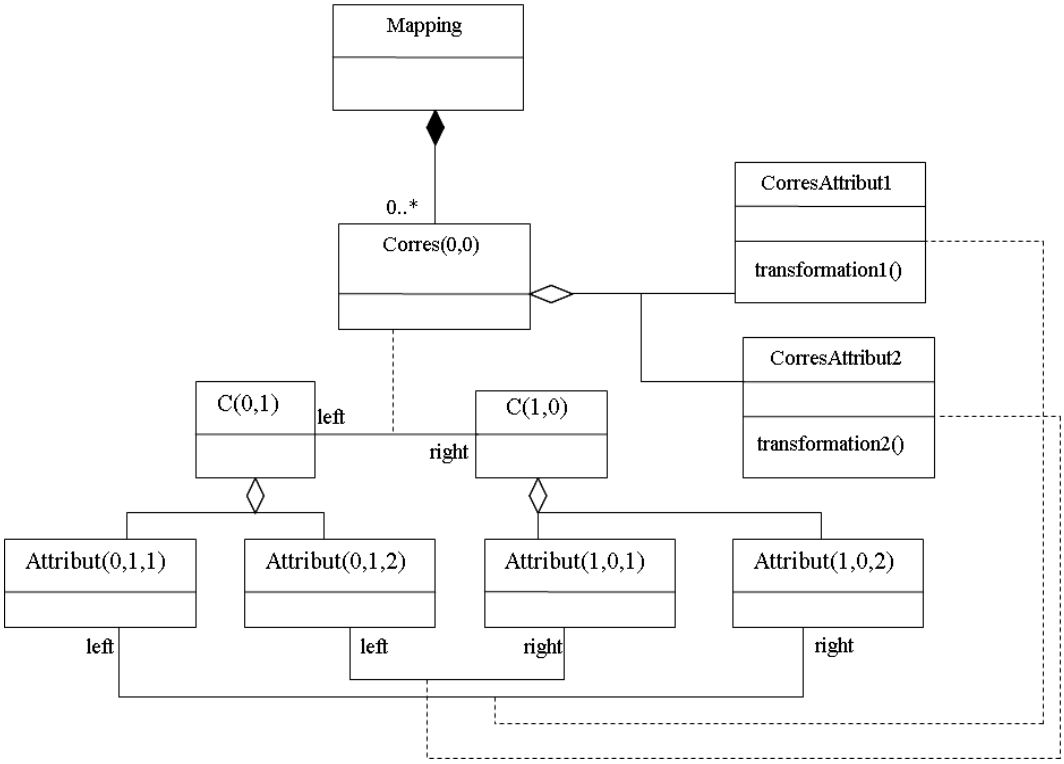

FIG. 5.12 – Méta-modèle de mapping de l'exemple

Mise en oeuvre

Le développeur du logiciel $SolidWorks^{TM}$ doit spécifier les règles de connexion entre le méta-modèle de $SolidWorks^{TM}$ et le méta-modèle PPO. De la même manière, le développeur du logiciel $Esprit^{TM}$ doit spécifier les correspondances entre le méta-modèle $Esprit^{TM}$ et le méta-modèle de PPO. La figure 5.13 montre une vue simplifiée de la connexion entre l'outil de CAO et de FAO à travers le méta-modèle PPO.

En langage métier, nous allons reconstruire le lien entre les *features* géométrique et les *features* d'usinage. On considère que les features géométriques ou d'usinages, sont des interfaces au sens PPO de la pièce géométrique traduite par une entité «component». Cette sémantique se traduit par les correspondances suivantes :

– la classe «Feature» dans le méta-modèle $SolidWorks^{TM}$ correspond à la classe «interface» dans le méta-modèle PPO,

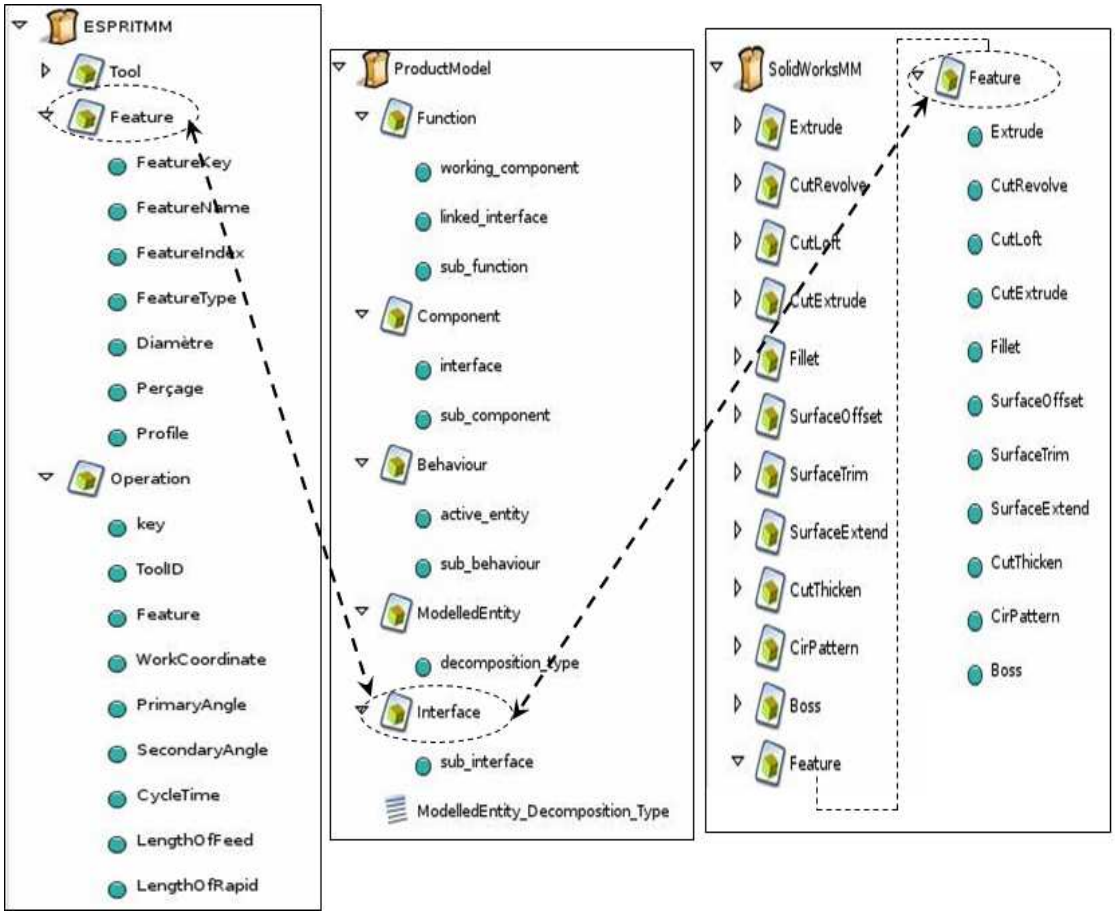

FIG. 5.13 – Correspondances générique entre $Esprit^{TM}$, PPO et $SolidWorks^{TM}$

– la classe «Feature» dans le méta-modèle $Esprit^{TM}$ et la classe «interface» dans le méta-modèle de PPO.

Comme le montre la figure 5.14, la classe «Feature» dans le méta-modèle $SolidWorks^{TM}$ généralise tout les features géométriques comme la classe «extrusion» et «cut-revolve». De même, la classe «Feature» dans le méta-modèle $Esprit^{TM}$ généralise toutes les features d'usinage comme la classe «perçage» et «profil». Cette correspondance donne donc un grand degré de liberté dans l'association des entités géométriques et d'usinage.

Une partie de l'activité de coopération consiste à résoudre des conflits entre acteurs métier sur des features du modèle produit. Le concepteur et le fabricant coopèrent pour régler des conflits sur des features avec leurs paramètres spécifiques. Dans cet exemple les paramètres significatifs à partager sont l'attribut «Mesure». La valeur spécifiée par les concepteurs ne satisfait pas le fabricant pour des raisons de disponibilité des machines-outils. On opte donc pour une correspondance à un niveau de granularité plus fin où on pourra spécifier les correspondances au niveau des attributs.

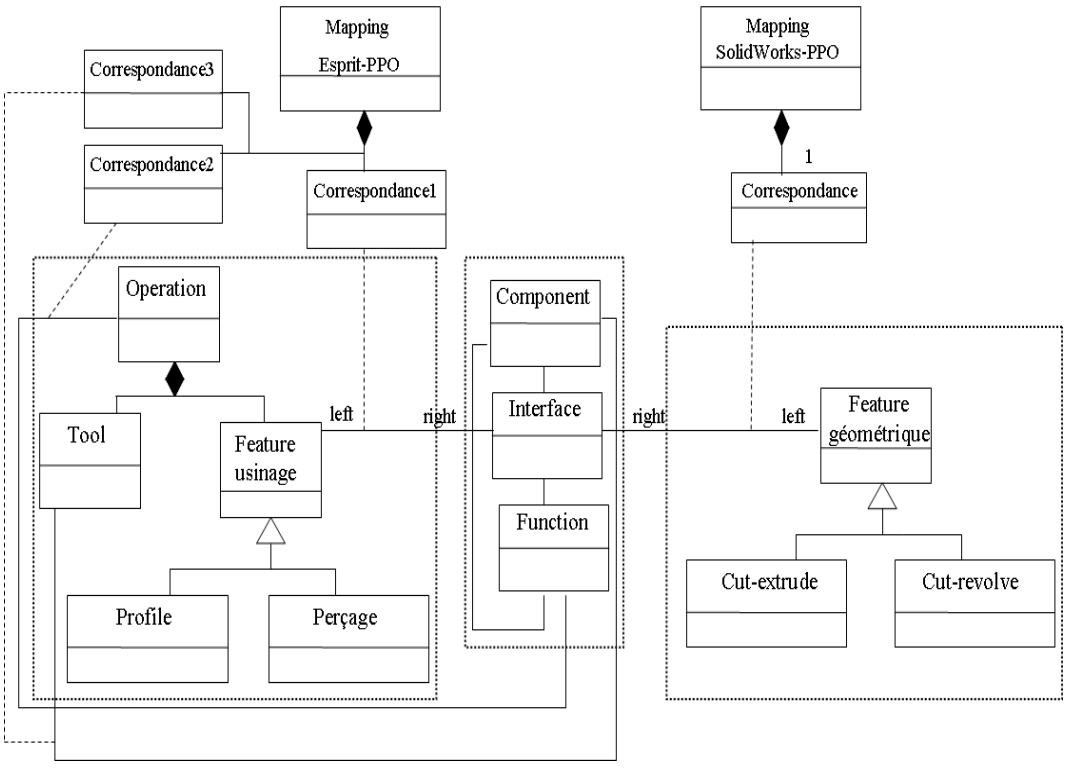

FIG. 5.14 – Méta-modèles de correspondances générique entre $Esprit^{TM}$, PPO et $SolidWorks^{TM}$

La figure 5.15 montre que le même *feature* a différentes identifications selon le point de vue expert. Pour le concepteur la *feature* est de type «Cut-

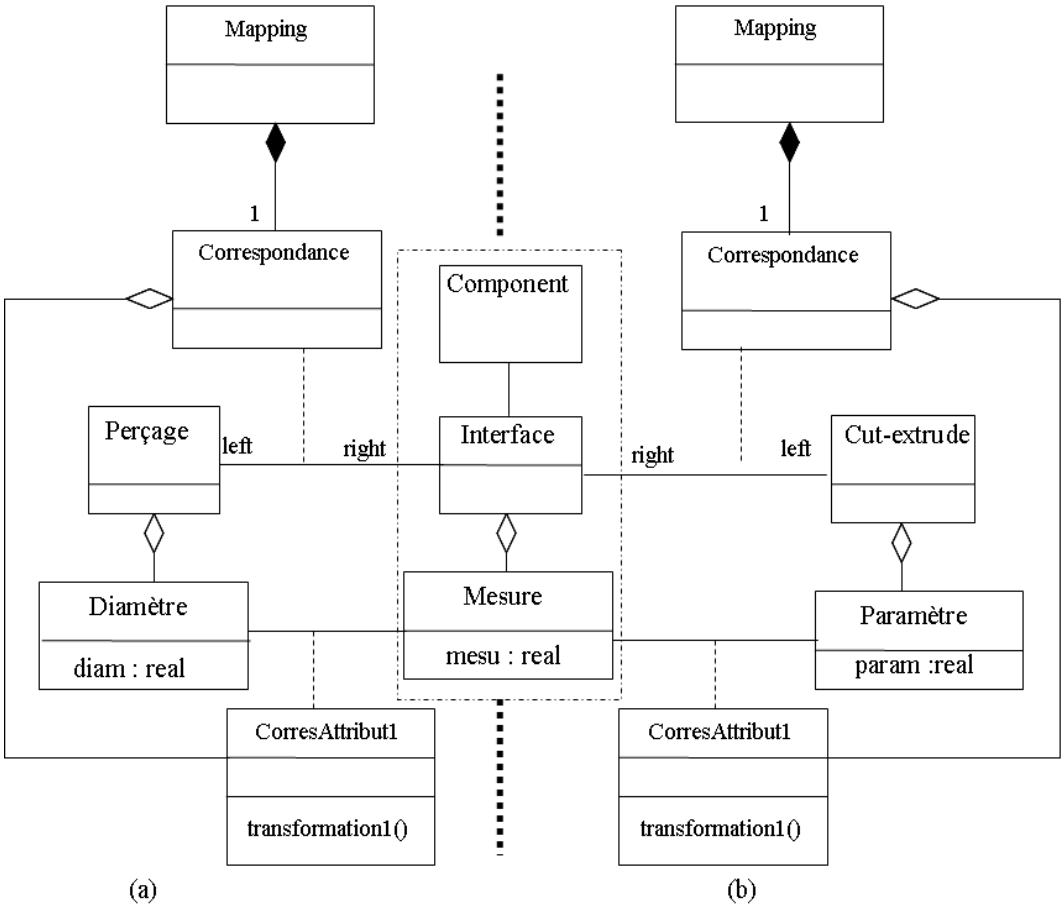

FIG. 5.15 – Méta-modèles de correspondances entre features de $Esprit^{TM}$,PPO et $SolidWorks^{TM}$

Extrude» en se référant à l'opération de CAO qui a permis la création de cette *feature*. Pour le fabricant c'est un «Perçage» au regard de l'opération de fabrication. L'interopérabilité se joue donc à un niveau syntaxique et à un niveau sémantique. Dans cet exemple l'attribut «Diamètre» de la classe «Perçage» correspond à l'attribut «Paramètre» de la classe «Cut-Extrude».

La figure 5.15(a) montre un exemple de correspondance entre la feature d'usinage perçage représentée par la classe «perçage» et la classe «Interface» du méta-modèle PPO. Cette correspondance contient une correspondance au niveau des attributs liant le diamètre de la classe «perçage» et la mesure de la classe «Interface». Dans PPO une interface n'existe que pour décrire un composant. Les modèles CAO et FAO sont modélisés par des entités de type «Component» en PPO. Les features géométriques et les features d'usinage sont modélisées par des interfaces de cette entité «Component». La méthode transformation1 représente la règle de transformation entre les deux attributs : en l'occurrence il s'agit d'une simple égalité.

La figure 5.15(b) montre un exemple de correspondance entre la feature géométrique «cut-extrude» et la classe «Interface» du méta-modèle PPO. Cette correspondance contient une correspondance au niveau des attributs : Paramètre de «Cut-extrude» et Mesure de «Interface».

La figure 5.16 présente la mise en oeuvre de la correspondance présentée dans la figure5.15(a). La feature d'usinage «10Perçage» est mise en correspondance avec l'interface «10Perçage». L'attribut «Diamètre» de la feature «10Perçage» correspond à l'attribut «Mesure» de l'interface «10Perçage».

La figure 5.17 présente la mise en oeuvre de la correspondance présentée dans la figure 5.15(b). La feature géométrique «Cut-extrude4» est mise en correspondance avec l'interface « Cut-extrude4». L'attribut «Paramètre» du feature «Cut-extrude4» correspond à l'attribut «Mesure» de l'interface « Cut-extrude4».

Enfin la figure 5.18 présente le modèle PPO commun résultat de la synchronisation des deux modèles homogènes : modèle PPO d'Esprit et modèle PPO de SolidWorks. La feature d'usinage «10Perçage» correspond à la feature géométrique «Cut-extrude4». La correspondance entre ces deux feature se modélise dans PPO par :

- créer une interface1 qui représente une interface du composant,
- créer deux sous interfaces «10Perçage» et «Cut-extrude4» représentant les deux vues métier,

– valuer l'attribut *decomposition-type=view* pour spécifier que ces deux sous interfaces représentent deux vues différentes du même feature.

A travers cet exemple, on a ramené le problème de synchronisation entre modèle hétérogènes à une synchronisation entre modèles homogènes couplés au travers du modèle PPO commun. Dans la section 3.4 nous présentons une méthode pour formaliser et structurer cette synchronisation.

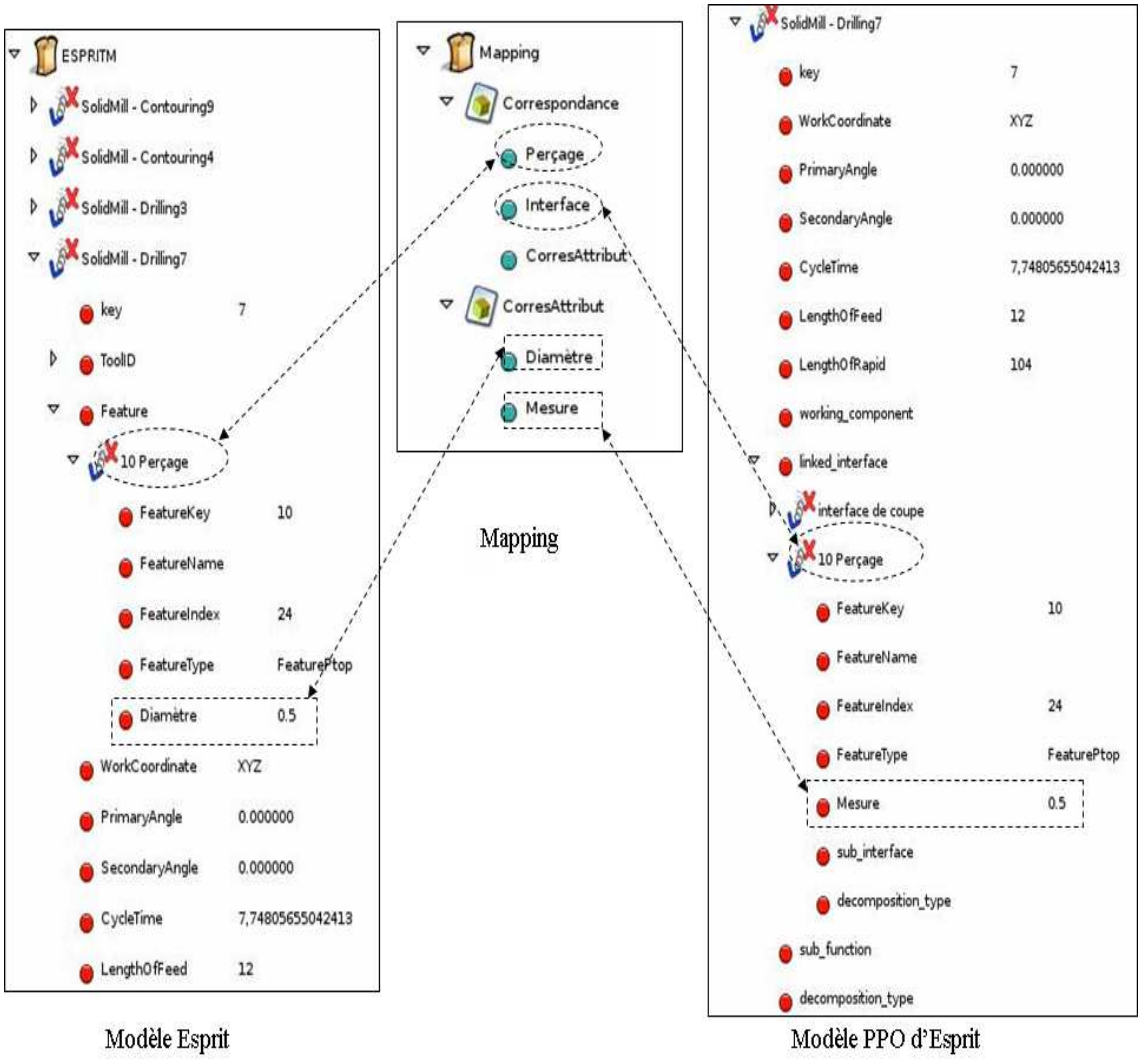

FIG. 5.16 – Correspondance entre Esprit et PPO

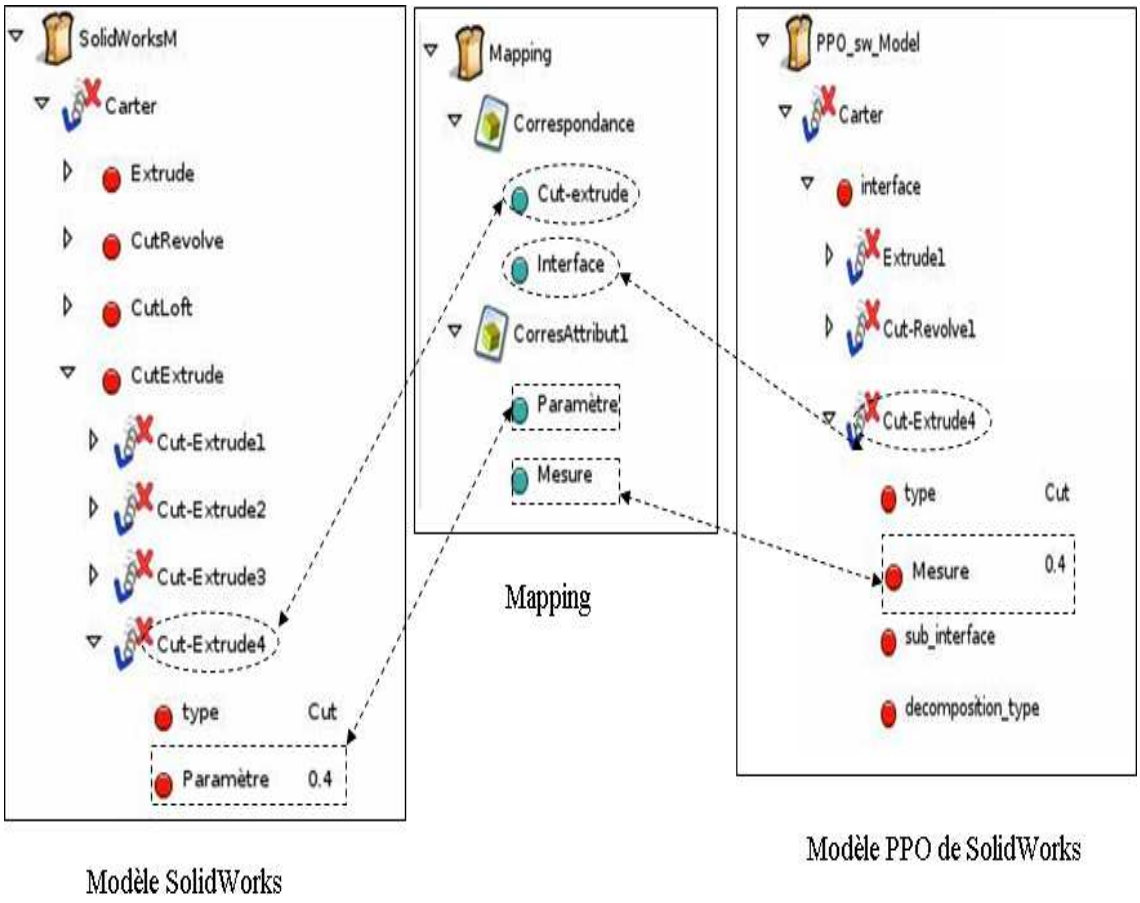

FIG. 5.17 – Correspondance entre SolidWorks et PPO

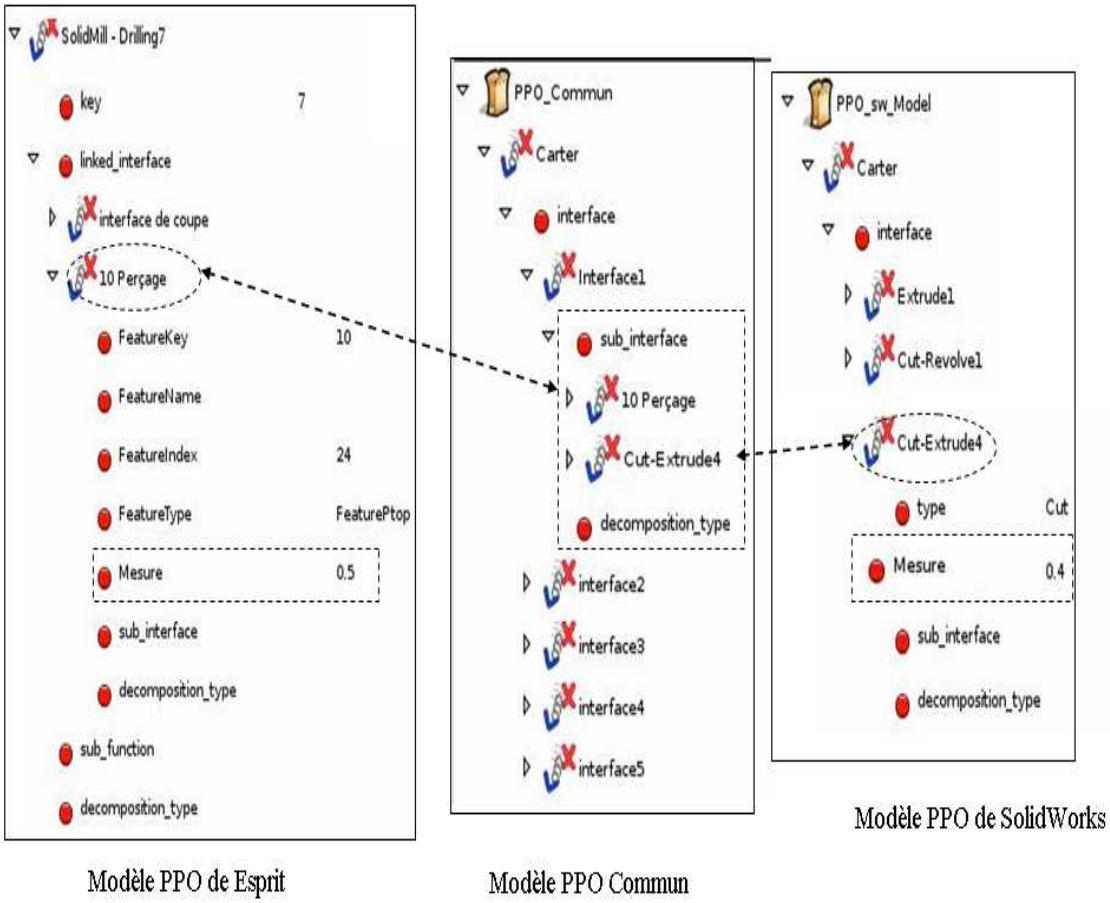

FIG. 5.18 – Correspondance entre Esprit et SolidWorks

5.3 Les modules de l'architecture

5.3.1 Le module d'export

Ce module a fait l'objet du chapitre 4. Ici nous détaillons plus avant son fonctionnement. L'export des modèles métier à partir d'outils métier est une première étape vers l'interoperabilité. La figure 5.19 représente l'architecture du module d'export. Le développeur métier est en charge de développer le module d'export. Pour cela il dispose de :

- l'API métier qui permet de manipuler les différents éléments du modèle métier,
- le méta-modèle métier qui décrit la connaissance des experts,
- l'application métier qui est le logiciel métier.

En utilisant ces entrées, le développeur métier produit un module d'export et un méta-modèle d'export. Le module d'export permet de gérer tous les éléments du méta-modèle d'export.

L'utilisateur qui peut être un concepteur CAO conçoit un modèle métier conforme au méta-modèle métier. Pour réaliser son export il dispose du module d'export comme ressource. L'utilisateur contrôle l'export en choisissant les éléments à exporter parmi le méta-modèle d'export. Le résultat de cette opération est un méta-modèle exporté conforme au méta-modèle d'export.

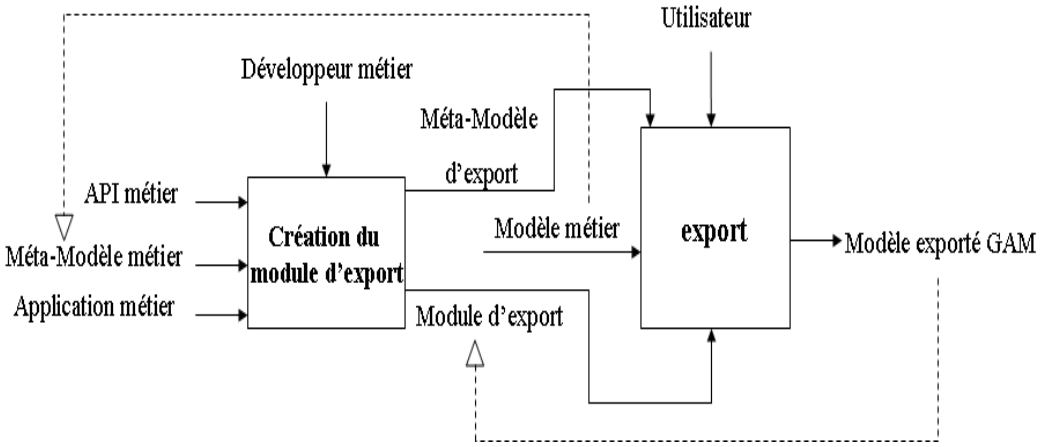

FIG. 5.19 – Module d'export

Pour réaliser l'export, le développeur a besoin des éléments suivants :

1. liste des objets d'un type donné ainsi que la manière de les parcourir,
2. accès aux attributs de l'objet,
3. accès aux relations entre objets.

L'utilisateur sélectionne les éléments à partager à partir de la gamme complète fournie par le développeur. Ces éléments forment le méta-modèle

d'export de l'utilisateur. Le méta-modèle d'export est un ensemble d'entité C_n définie par un nom_n et un ensemble de couple $(nomattribut_n, typeattribut_n)$. On obtient le formalisme suivant :

$$MMexport = \left\{ \left(C_n, nom_n, \left(nomattribut_{n_i}, typeattribut_{n_i} \right)_{i \in [1,a]} \right) \right\}_{n \in [1,h]}$$

Le modèle métier de l'application est formalisé comme suivant :

$$Model = \left\{ \left(O_i, nom_i, type_i, \left(nomattribut_{i_j}, valeurattribut_{i_j} \right)_{j \in [1,k]} \right) \right\}_{i \in [1,b]}$$

Le but du module d'export est de générer un modèle conforme au méta-modèle d'export. Pour manipuler la description du modèle dans une application métier, les méthodes suivantes sont nécessaire :

- Object-New(C_n) crée un feature O_i conforme à la classe C_n,
- TYPE(O_i) retourne le type d'un feature O_i,
- FirstFeature() est appliqué sur le modèle et retourne le premier feature du modèle métier,
- O_i.GetNextFeature() retourne le feature suivant O_{i+1},
- O_i.GetName() retourne le nom du feature O_i,
- O_i.GetTypeName() retourne le type du feature O_i,
- O_i.Get-NB-ATT() retourne le nombre d'attributs du feature O_i,
- O_i.Get-ATT-Value (*nomattribut*) retourne la valeur de l'attribut *attributename* du feature O_i,
- O_i.Set-ATT-Value (*nomattribut*, *valeurattribut*) modifie la valeur de l'attribut *nomattribut* du feature O_i avec la valeur *valeurattribut*,

En parallèle l'environnement de travail (type GAM) doit fournir des méthodes de manipulation des modèles et des méta-modèles :

- *gam-find-object(nomobjet)* retourne la référence de l'objet s'il existe et None sinon,
- *object-new(nomobjet, typeobjet)* crée un objet de nom nomobjet et de type typeobjet. Exemple *extrusion = object-new("Extrusion", extrusion).*

Avec ces méthodes, l'algorithme d'export est donné par le pseudo-code suivant :

Export-Model() :

Feature = Model.FirstFeature()

While Feature != None

Select Case Feature.GetTypeName()
for n =1 to n=h (h nombre de classe de MMexport
Case $MMexport.C_n.nom_n$
if find (O_n)=None then
O_n= Object-New(C_n)
for i =1 to i=a (a nombre d'attributs de C_n)
$nomattribut = O_n.nomattribut_{n_i}$
valeurattribut=Feature.Get-ATT-Value(nomattribut)
O_n.Set-ATT-Value (nomattribut, valeurattribut)
Case Else
End Select
Feature = Feature.GetNextFeature()
Loop

Le module d'export doit vérifier si chaque élément existe déjà dans le modèle exporté ou pas. Si l'élément existe, ses attributs sont mis à jour, sinon l'élément est créé. Un extrait de l'application de cet algorithme sur l'application SolidWorks et l'application Esprit donne le tableau 5.2 :

Dim swFeat As SldWorks.Feature	*Dim ToolTech As EspritTechnology.Technology*
Do While Not swFeat Is Nothing	*For Each ToolTech In Document.Tools*
Case "Extrusion"	*ToolID = ToolTech.Item("ToolID").Value*
depth1 = swFeat.GetDefinition().GetDepth(True)	*ToolStyle = ToolTech.Item("ToolStyle").Value*
uid =gam-find-object(Extrusion)	*uid =gam-find-object(ToolID)*
if uid = None	*if uid = None*
extrution = SolidWorks-extrusion-new(Extrusion)	*Tool = Esprit-tool-new(ToolID)*
SolidWorks-extrution-set-depth1(extrution, depth1)	*Esprit-tool-set-toolstyle(Tool, ToolTech.Item("ToolStyle"))*
Set swFeat = swFeat.GetNextFeature	

TAB. 5.1 – Un extrait de l'alghorithme d'export de SolidWorks et d'Esprit

5.3.2 Le module d'import

L'import des modèles métier depuis des modèles partagés est une étape nécessaire pour propager les modifications dans l'application métier elle même.

La figure 5.20 représente l'architecture du module d'import. Le développeur métier est en charge de développer le module d'import. Pour cela il dispose de :

- l'API métier qui permet de manipuler les différents éléments du modèle métier,
- le méta-modèle métier qui décrit la connaissance des experts,
- l'application métier qui est le logiciel métier.

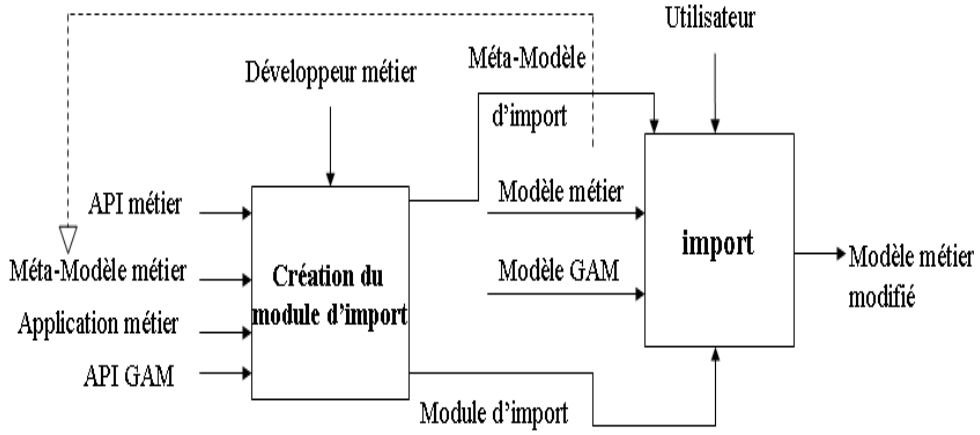

FIG. 5.20 – Module d'import

En utilisant ces entrées, le développeur de l'application métier produit un module d'import et un méta-modèle d'import. Le module d'import permet de gérer tous les éléments du méta-modèle d'import.

L'utilisateur sélectionne les éléments à importer conformément à son méta-modèle d'import à partir du modèle partagé (modèle GAM). Pour réaliser son import, il dispose du module d'import comme ressource. Le résultat de cette opération est la mise à jour du modèle métier dans l'application métier. Pour réaliser un module d'import on a besoin des méthodes suivantes :

- une méthode *Feature=find-feature(nomfeature)* qui permet de trouver le fichier de nom nomfeature et retourne sa référence,
- une méthode O_i=*Model.Object(i)* qui permet de récupérer dans un modèle GAM l'objet d'indice i,
- une méthode O_i*.Get-ATT-Value(nomattribut)* retourne la valeur de l'attribut *attributename* du feature O_i,
- *Feature.Set-ATT-Value (nomattribut, valeurattribut* modifie la valeur de l'attribut *nomattribut* du feature,

Un algorithme d'import est donné par le code suivant :

Import-Model() :

for n =1 to n=b

O_n = *Model.Object(n)*

While O_n *!= None*

Feature=find-feature(O_n*.name)*

for i =1 to i=k

$nomattribut = O_n.nomattribut_{n_i}$

valeurattribut=O_n*.Get-ATT-Value(nomattribut)*

Feature.Set-ATT-Value (nomattribut, valeurattribut)

5.3.3 Le moteur de transformations

Le moteur de transformation exécute le méta-modèle de correspondance. La figure 5.21 représente l'architecture du moteur de transformation. Le développeur est en charge de développer le moteur de transformation. Pour cela il dispose des :

- méta-modèle source,
- méta-modèle cible,
- méta-modèle de correspondances entre le méta-modèle source et le méta-modèle cible,
- API métier cible qui permet de manipuler les différents éléments du modèle métier cible.

Pour une correspondance de type n vers 1, le mapping de deux modèles N et R présenté dans la section 2.3 est donné par l'expression suivante :

$$mapping(N, R) = \{C_m, (C_n, C_r), (nomattribut_{n_i i\in[1,a]}, nomattribut_{r_1}, F_m)\}$$

En utilisant ces entrées le développeur produit le modèle cible conforme au méta-modèle cible et respectant le méta-modèle de correspondances. Pour réaliser le moteur de transformation on a besoin de méthodes suivantes :

- une méthode Open-Model(ModelName) ouvre le modèle au nom *ModelName*
- une méthode liste=Model-Get-Content(ModelName) retourne une référence sur le modèle,
- une méthode nb=Model-Content-Nb(liste) retourne le nombre de fils du modèle,

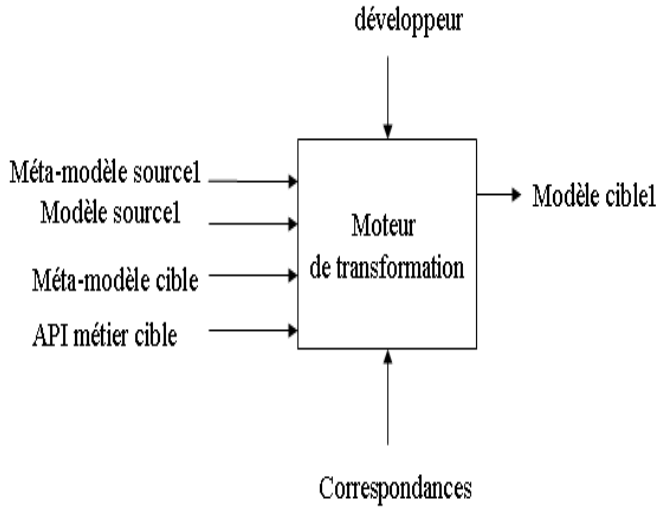

FIG. 5.21 – Moteur de transformation

- une méthode O_i=*Model.Get-Object(i)* qui permet de récupérer dans un modèle l'objet d'indice i,
- une méthode objclassifier=O_i.Get-Classifier() retourne le classifier C_n de l'objet,
- une méthode Object-New(C_n) crée un feature O_i conforme à la classe C_n,
- une méthode O_i.GetName() retourne le nom du feature O_i,
- O_i.Get-NB-ATT() retourne le nombre d'attributs du feature O_i,
- une méthode O_i*.Get-ATT-Value(nomattribut)* retourne la valeur de l'attribut *attributename* du feature O_i,
- *Feature.Set-ATT-Value (nomattribut, valeurattribut* modifie la valeur de l'attribut *nomattribut* du feature,
- une méthode *attributelist*=O_i*.Get-Attributes()* retourne la liste des attributs de l'objet O_i,

Nous présentons le pseudo-code qui permet de réaliser une transformation simple :

Moteur-Transformation() :

Open-Model(Correspondance)

Open-Model(ModelName)

liste=Model-Get-Content(ModelName)

nb=Model-Content-Nb(liste)

For i=1 to i=nb
O_i=Model.Get-Object(i)
objclassifier=O_i.Get-Classifier()
classifiername=objclassifier.GetName()
attributelist=O_i.Get-Attributes()
For j=1 to i=m (m nombre de classe de correspondances)
If classifiername = C_{nj}.GetName()
Obj=Object-New(C_r)
$Obj.Set - ATT - Value(nomattribut_{r_{j1}}, F_m(attributelist))$

Un extrait simplifié des transformations décrites dans la figure 5.14 est donner par le tableau 5.2 :

last=gamsession-document-find-last(None,"ESPRITM",0)	*last=gamsession-document-find-last(None,"SolidWorksM",0)*
gam-open-document(work,"./",last)	*gam-open-document(work,"./",last)*
liste=gam-workspace-document-get-content(work,last)	*liste=gam-workspace-document-get-content(work,last)*
Component=ppo-component-new("Carter")	*Component=ppo-component-new("Carter")*
nb=gam-package-content-nb(work,liste)	*nb=gam-package-content-nb(work,liste)*
while i<=nb :	*if uid = None*
obj= gam-package-content-get-child(work, liste,i)	*obj= gam-package-content-get-child(work, liste,i)*
objclassifier=gam-object-get-classifier(work,obj)	*objclassifier=gam-object-get-classifier(work,obj)*
classifiername=gam-object-get-name(work,objclassifier)	*classifiername=gam-object-get-name(work,objclassifier)*
if classifiername=="Operation"	*if classifiername=="Extrude"*
Operation1=PPO-Esprit-operation-new("Operation1"')	*Extrude1=PPO-SolidWorks-extrude-new("Extrude1"')*
attributelistO=gam-class-get-attributes(work,obj)	*attributelistO=gam-class-get-attributes(work,obj)*
...	...
ppo-function-working-component-insert-item(Component,Operation1)	*ppo-component-interface-insert-item(Component,Extrude1)*

TAB. 5.2 – Un extrait des transformations d'Esprit2PPO et de SolidWorks2PPO

5.3.4 module de synchronisation de modèle homogènes

Bien que dans cette phase les modèles soient homogènes, une mise en correspondance de différents éléments des deux modèles est nécessaire. La figure 5.22 propose une procédure de synchronisation de modèles homogènes. Cette procédure est basée sur trois opérateurs : l'opérateur de correspondance, l'opérateur de comparaison et l'opérateur de fusion.

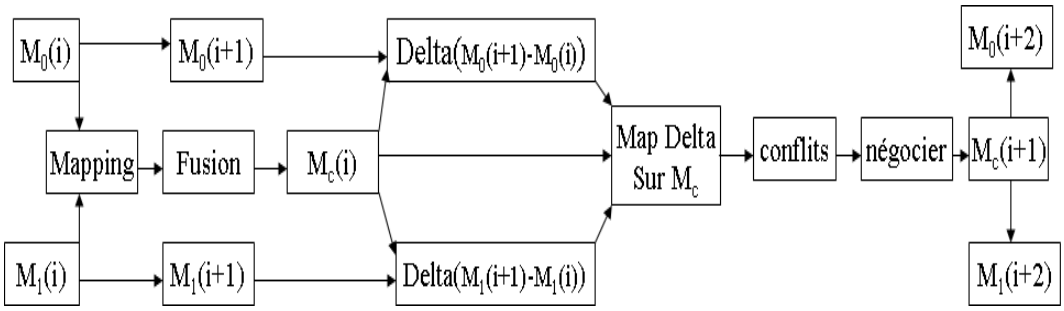

FIG. 5.22 – Procédure de synchronisation de modèles homogènes

Pour synchroniser deux modèles homogènes, les étapes suivantes sont nécessaires :

- associer les représentations initiales $M_0(i)$ et $M_1(i)$,
- fusionner les modèles $M_0(i)$ et $M_1(i)$ ce qui mène à créer $M_c(i)$. Cette fusion est effectuée manuellement en manipulant les modèles à travers l'interface de GAM. L'interface nous donne la possibilité de supprimer, copier/coller ou de modifier des éléments des modèles. Nous construisons ainsi la $M_c(i)$,
- comparer $M_k(i)$ et $M_{k+1}(i)(\forall k \in [0,1])$. Cette comparaison est effectué avec le module GAM-DIFF du cadre GAM. Nous obtenons donc $Delta(M_{k+1}(i) - M_k(i))$,
- relier les états dans $M_c(i)$,
- identifier les conflits de base par application. D'autres travaux sont en cours pour l'identification de conflits (Sadeghi *et al.*, 2007),
- Négocier les modifications à propager dans le modèle $M_c(i+1)$. Les modifications sont propagées de façon manuelle. L'experts copie/colle ou supprime les modifications.

5.4 Synchronisation

5.4.1 Bilan du fonctionnement de l'environnement proposé

Dans cette section, nous présentons les différents éléments du module de synchronisation de modèles hétérogènes. La figure 5.23 résume le processus complet de synchronisation de modèles hétérogènes. Les étapes de ce processus sont les suivantes :

- exporter le modèle métier $M_{metier0}(i)$. Cette opération crée la réprésentation $R_0(i)$,
- associer la représentation $R_k(i)$ et le modèle $M_k(i)(\forall k \in [0,1])$. Le *Mapping* est un ensemble d'entités correspondantes : $Mapping(R_k(i), M_k(i)) = \{E_j, C_j\}$,
- exporter le modèle $M_{metier0}(i)$ modifié et créer ainsi la représentation $R_0(i+1)$,
- comparer $R_k(i)$ et $R_{k+1}(i)(\forall k \in [0,1])$. Cette comparaison est effectuée avec le module GAM-DIFF du cadre GAM. Nous obtenons donc $Delta(R_{k+1}(i) - R_k(i))$,
- vérifier si les éléments de *Delta* font partie du *Mapping*. S'il existe un élément $E_j \in Delta(R_{k+1}(i) - R_k(i))$ et $E_j \in Mapping(R_k(i), M_k(i))$, alors mettre à jour l'élément correspondant C_j,
- synchroniser les modèles homogènes $M_k(i)(\forall k \in [0,1])$.

La figure 5.24 décrit un exemple de synchronisation d'un modèle métier avec le modèle de fédération PPO.

- dans l'étape 1 de l'activité de collaboration, nous avons le modèle et le méta-modèle métier ($Esprit^{TM}$), le modèle et le méta-modèle de correspondance et le méta-modèle de PPO. Utilisant ces entrées, nous pourrions produire le modèle cible PPO,
- dans une activité de collaboration, un expert fait plusieurs modifications et a besoin de partager ses informations plusieurs fois afin de collaborer efficacement. Dans l'étape 2, la modification est calculée grâce au module «GAM-diff» de l'environnement GAM. Comme montré dans le modèle de GAM-diff, la modification concerne l'attribut «angle». La valeur de cet attribut des deux cotés est proposée,
- la synchronisation du modèle d'$Esprit^{TM}$ et du modèle de PPO a pu être réalisée à l'étape 3. À l'étape 3, nous prenons comme entrée le modèle de GAM-diff et le modèle de correspondance afin de produire une

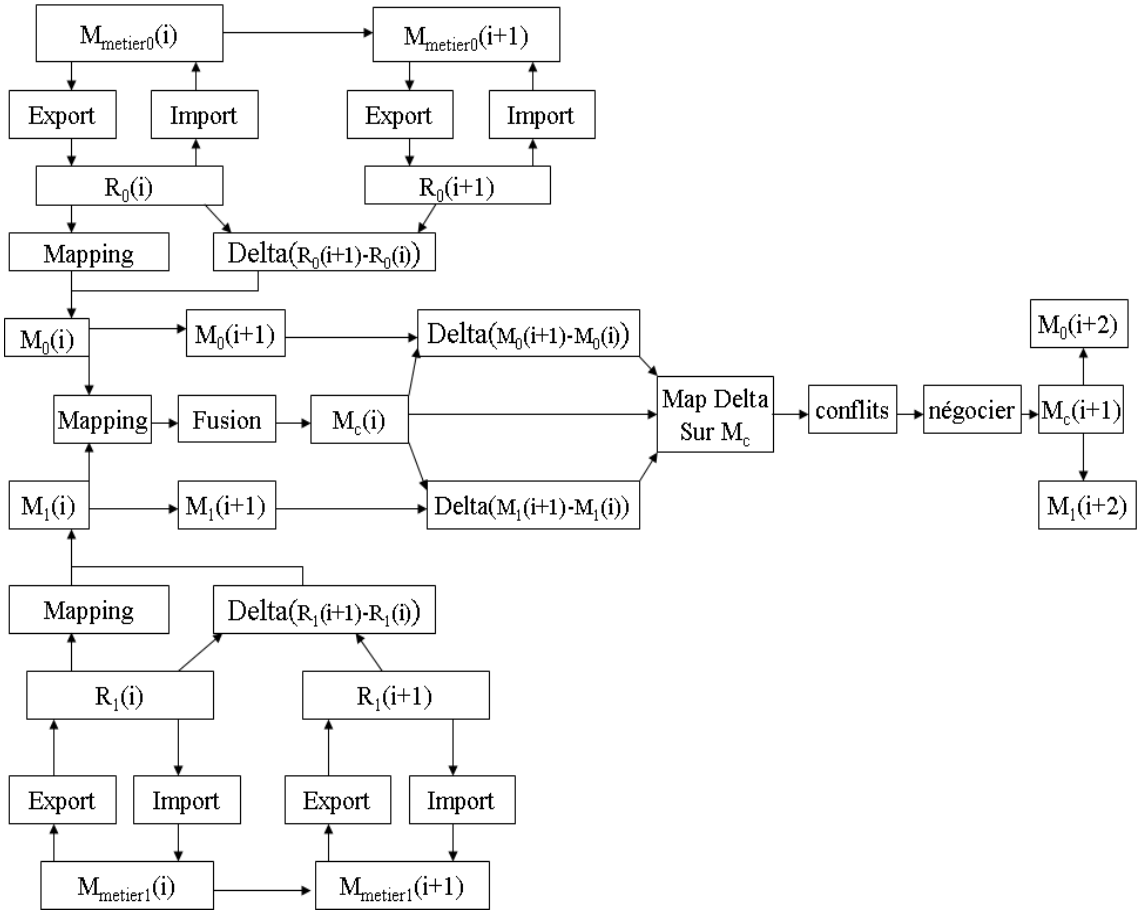

FIG. 5.23 – Le processus entier d'un module de synchronisation de modèles métier hétérogènes

nouvelle version du modèle PPO. Le processus recherche pour chaque objet dans le modèle de GAM-diff la modification effectuée. Il vérifie si cet objet est impliqué dans le modèle de correspondance et propage la modification.

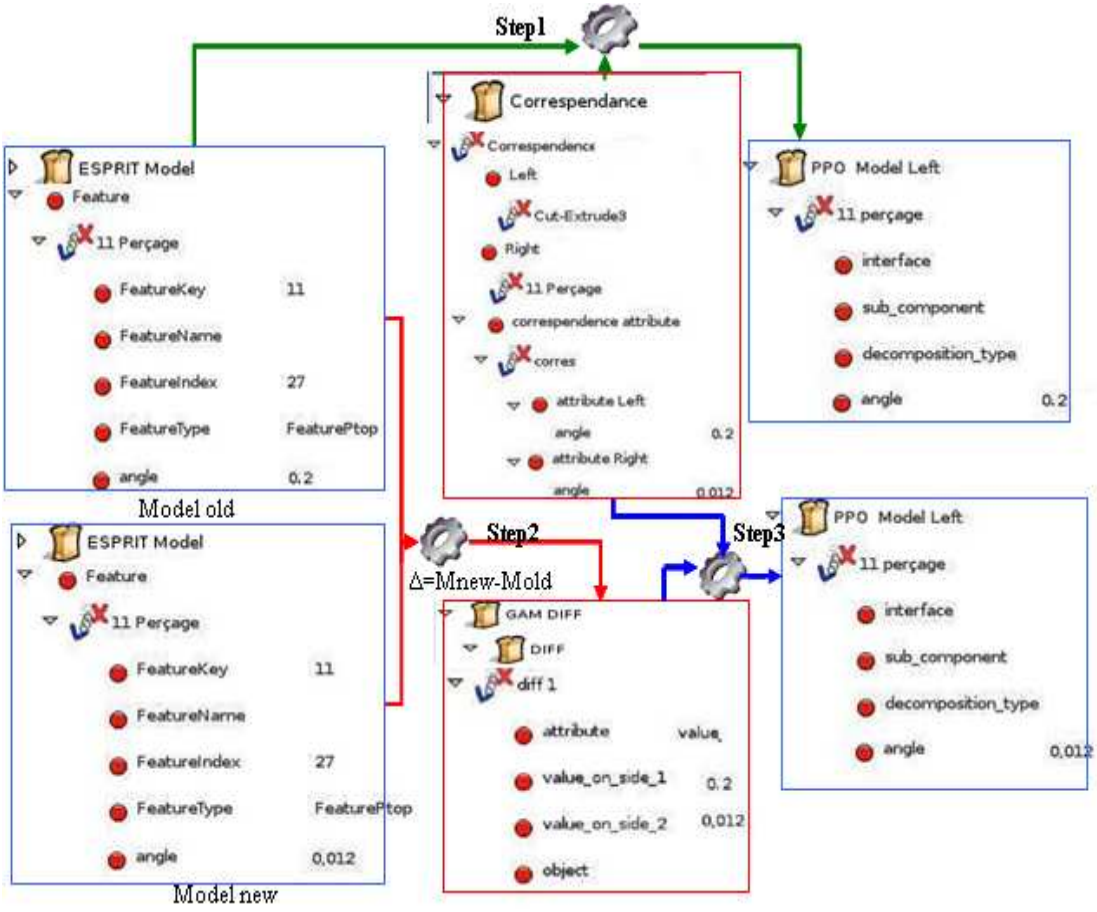

FIG. 5.24 – Exemple de synchronisation d'un modèle métier $Esprit^{TM}$ avec PPO

5.4.2 Prise en compte des démarches hors IDM

Comme démontré précédemment, l'approche IDM fournit une aide pour modéliser et gérer les correspondances entre les modèles de produit issus d'outils hétérogènes. La correspondance entre les modèles hétérogènes est une approche associative pour relier différents objets des modèles métiers. Les objets représentant les différents points de vue métiers du même produit sont reliés et le transfert de l'information par l'intermédiaire de la correspondance parmi eux est possible. Les modèles dans différentes vues peuvent partager l'information par l'intermédiaire de l'environnement de conception collaborative PPO.

Évidemment, l'amélioration de l'interopérabilité entre les outils experts devrait augmenter leur capacité de collaboration et fournir un processus de développement de produit plus facile. Les experts ont besoin d'un environnement flexible pour formaliser les nouvelles connaissances au sujet de ces liens. L'approche IDM peut fournir l'environnement de formalisation. Cependant la pratique du développement de produit mène à la perte de correspondance entre les modèles métier. Elle doit être alors reconstruite entièrement.

La correspondance entre modèles est une connaissance non formalisée. Elle relève de l'expertise métier. Dans certains cas cette correspondance est simple à identifier. Nous venons d'étudier le cas d'un outil de conception et d'un outil de fabrication où la correspondance se fait au niveau des feature géométriques. Dans le cas d'autres outils métier la correspondance est moins évidente à identifier et pourtant de nombreux travaux portent sur la transformation d'un modèle métier à un autre. Les règles métier sont donc existantes. Il faudra procéder à des compléments de travaux d'identification de ces correspondances pour permettre l'interopérabilité. La section suivante présente quelques travaux allant dans ce sens.

La synchronisation entre les modèles métier a besoin d'une étape pour l'identification de lien (Bettaieb et Noel, 2006). Le lien entre les modèles experts est complexe et difficile à reconstruire. La figure 5.25 montre trois modèles métier hétérogènes :

- le modèle CAO, le modèle d'élément fini et le modèle de mécanisme,
- le modèle élément fini est employé pour valider et optimiser le produit,
- le modèle de mécanisme est employé pour évaluer la cinématique et la dynamique.

Le lien entre ces modèles a besoin d'une étape de reconnaissance. Pour des modèles métiers spécifiques, des approches ont été développées pour extraire automatiquement la correspondance. (Noël *et al.*, 1995) développent une structure de données et des algorithmes spécifiques pour effectuer la classification de maillage sur une CAO 3D. La classification de maillage est issue des travaux de (Schroeder, 1991) pour la génération automatique de maillages à partir de modèles CAO.

Dans le cadre du lien CAO FAO, quelques approches ont été développées pour identifier automatiquement les *features* pour la fabrication (Li *et al.*, 2006). L'identification de *feature* est une approche de modélisation géométrique (*solid modeling*) qui se focalise sur la conception et l'implémentation

des algorithmes pour détecter l'information de fabrication des modèles CAO. La reconnaissance automatisée de *feature* est un domaine de recherche actif dans la modélisation géométrique et est considérée comme un élément critique pour l'intégration de la CAO et de la FAO. Des approches dans le contexte de FEM/CAO ou dans le contexte de CFAO sont spécifiques à ces métiers et principalement basées sur les règles géométriques difficilement représentables au travers de l'IDM. La combinaison des techniques de reconnaissance de lien entre modèles hétérogènes spécifiques et l'IDM est essentielle pour traiter le problème de la correspondance entre les modèles hétérogènes.

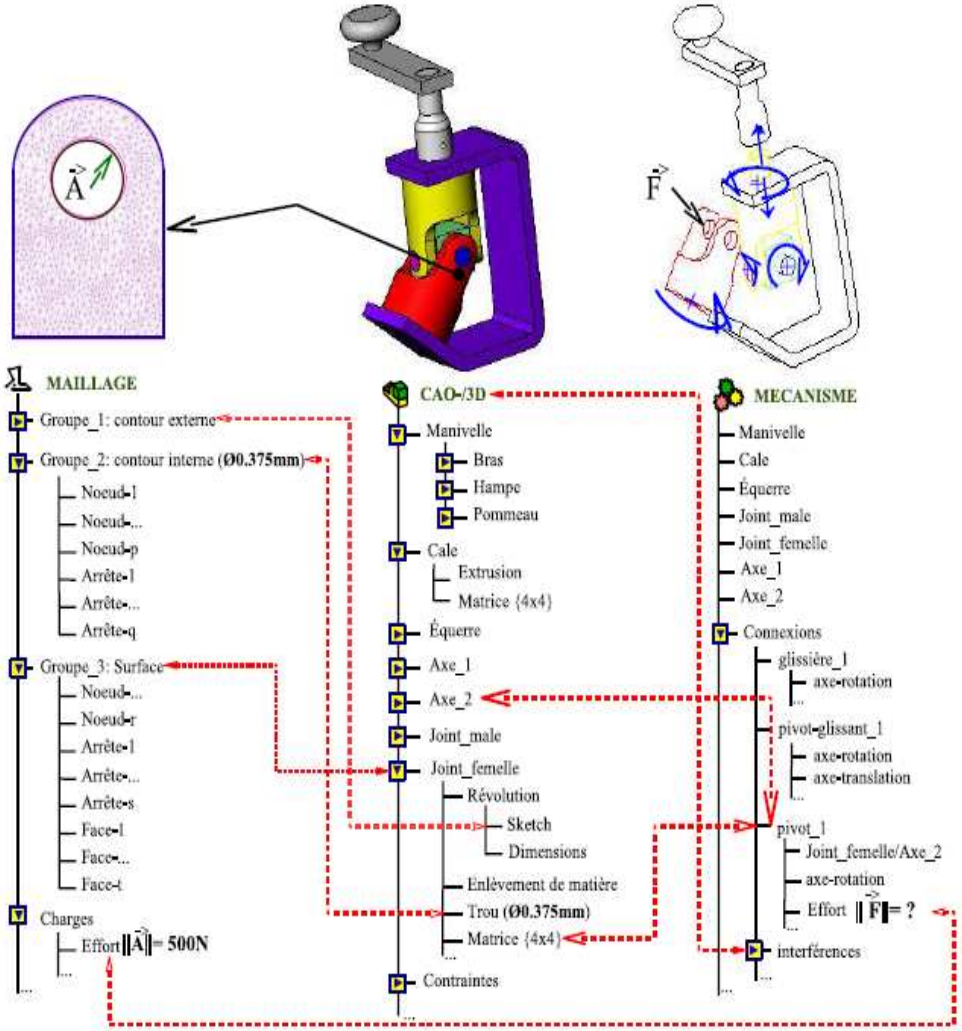

FIG. 5.25 – Les modèles initiaux et les liens entre leurs entités (Bettaieb, 2006)

5.4.3 Vers une architecture étendue

La figure 5.26 montre tous les éléments dont on a besoin pour accomplir le module de synchronisation. En effet, l'application de l'IDM dans le domaine de conception de produit manufacturier révèle quelques difficultés vis à vis des types de modèles mis en oeuvre. Les modèles sont parfois complémentaires et ne représentent pas un autre point de vue du même élément. Une étape de conversion des connaissances métier est alors nécessaire pour savoir quoi correspond à quoi. Le recourt à des travaux complémentaires dans le domaine pour pouvoir formaliser le mapping entre ces deux modèles est nécessaire. La figure montre que d'autres travaux de recherche complète l'IDM pour accomplir le mapping.

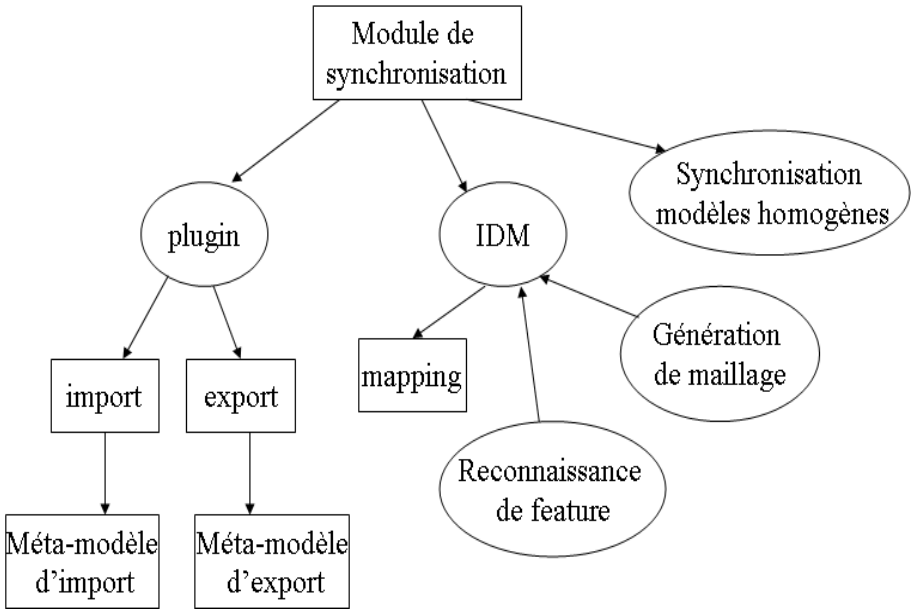

FIG. 5.26 – Module de synchronisation

5.5 Conclusion

Ce chapitre étudie comment formuler et organiser la synchronisation des modèles hétérogènes. L'approche proposée permet à des experts métier de formuler leur connaissance. La connaissance partagée exprimée améliore la collaboration efficace et formalise les compétences de l'expert tout en préservant la confidentialité.

L'IDM semble s'appliquer pour la synchronisation des modèles de conception des produits manufacturés. Nous démontrons cet intérêt dans de nouveaux

domaines pour l'IDM habituellement appliquée au génie logiciel. L'implémentation de cette approche est effectuée pour un scénario démonstratif de l'interaction entre les outils de CAO et de FAO dans l'environnement de collaboration PPO.

Le résumé de notre contribution est :

1. la création des plug pour connecter une nouvelle représentation et pour extraire le modèle métier dans le format de GAM.

2. la proposition d'un méta-modèle de mapping qui aide à la création de la correspondance entre le modèle métier et le modèle de fédération.

3. la synchronisation des modèles métier hétérogènes avec une approche fédérée en s'appuyant sur les méthodologies de l'IDM pour la gestion des interactions entre les outils métier,

4. la mise en évidence de la limite de l'application de l'approche IDM dans un domaine de conception de produit manufacturier où la mise en correspondance doit être complétée par d'autres travaux d'expertise métier.

Conclusion générale

Dans ce travail nous apportons des éléments de réponse à la problématique de l'interopérabilité dans le domaine de la conception collaborative de produits manufacturiers. Notre contribution majeure est la proposition et l'implémentation d'une méthodologie de connexion entre outils métier hétérogènes à travers un environnement collaboratif. Cette méthodologie se décompose en plusieurs étapes :

1. développement d'un plugin qui permet l'exportation du modèle métier de l'expert dans l'environnement de collaboration.

2. transformation et mise à jour du modèle exporté en modèle conforme au méta-modèle de fédération de l'environnement de collaboration : définition d'un méta-modèle de correspondances.

3. synchronisation des modèles métier homogènes.

La problématique de l'interopérabilité se pose dés que les experts utilisant des outils hétérogènes sont amenés à coopérer. Les problèmes d'interopérabilité sont d'ordre technologiques, sémantiques et syntaxiques. Pourtant les experts coopèrent. Dans le chapitre 2, nous avons détaillé les différentes méthodes et outils existants pour l'échange et le partage de l'information en conception. Dans ce domaine, deux grandes catégories existent : les modèles produits et les standards d'échanges. Il ne suffit pas de modéliser l'information mais il faut aussi pourvoir la partager. Pour les modèles produit, nous avons choisi de travailler avec le modèle produit de PPO qui reprend les notions du paradigme FBS et qui représente un compromis efficace entre le niveau de granularité très fin du modèle produit de STEP et le niveau de détails des SGDT.

Si nous choisissons de partager l'information autour du produit avec un modèle produit donné (ici modèle produit de PPO), nous devons fournir les moyens aux modèles métier hétérogènes de communiquer avec ce modèle produit. Les problèmes d'interopérabilité sont d'ordre technologique et syntaxique. Nous choisissons une approche fédérée pour l'interopérabilité. Le modèle produit choisi est le modèle de fédération. Dans ce travail nous ne traitons pas

l'interopérabilité sémantique bien que nous reconnaissons son existence. Le chapitre 3 présente la méthodologie IDM (Ingénierie Dirigée par les Modèles) et spécialement la MDA (une implémentation de l'IDM) comme solution pour l'interopérabilité en génie logiciel où les modèles sont figés et les règles de correspondance sont explicites et peu variables. Nous avons proposé une méthodologie de synchronisation pour l'échange d'informations entre un expert CAO et un expert FAO. Cette méthodologie a été implémentée pour un scénario d'usage avec les outils métier $SolidWorks^{TM}$ et $Esprit^{TM}$.

Les difficultés rencontrées lors de la mise en place d'une telle approche sont :

1. pendant la phase de transformation de modèles hétérogènes : la transformation entre modèles métier hétérogènes se base sur l'établissement de correspondance entre les concepts de modèle métier et les concepts du modèle de fédération en premier lieu. La mise en correspondance entre le modèle de l'expert métier et le modèle de fédération est une étape délicate à cause de la nature du modèle métier et du modèle de fédération. Cette étape nécessite une réflexion et une connaissance du modèle métier et du modèle de fédération pour pouvoir associer les concepts.
2. pendant la phase de synchronisation de modèles homogènes : la mise en correspondance des modèles homogènes nécessite une connaissance des deux expertises métiers pour pouvoir associer les deux vues différentes du même produit. Dans notre cas d'étude, il faut savoir que telle extrusion du modèle CAO correspond à tel perçage du modèle FAO pour pouvoir les associer dans le modèle de fédération PPO sous formes de sous-interface avec deux vues différentes du même élément. Dans notre cas d'étude, les correspondances sont établies entre des features géométriques et des features d'usinage. Ces correspondances restent donc explicites.

Si on prend le cas de modèles aux éléments finis, l'établissement de correspondances entre ce modèle et celui de la CAO est moins évident à expliciter. L'application du principe de transformation de modèles, concept clé de l'IDM est possible si on est entre expertises métier où les correspondances entre concepts de partage sont clairs et formalisables. Ceci limite l'application de l'IDM entre applications métiers de conception de produit manufacturier.

L'IDM n'est à ce jour pas capable de formaliser la transformation complexe décrite dans des algorithmes de génération de maillages. Dans nos travaux futurs nous nous concentrerons sur la définition de processus de raffinement de règles métier pour qu'elles soient suffisamment explicites pour être décrites avec un modèle de correspondances.

Nous souhaitons combler le manque de correspondances directe et explicites entre certaines expertises métier de conception de produit. Aujourd'hui, l'IDM ne peut pas être à elle seule le noyau d'un module de synchronisation générique de notre environnement. Nous devrons regarder les travaux en matière de reconnaissance de features pour l'usinage ou encore les travaux de génération automatique de maillage pour les intégrer dans une méthodologie de synchronisation étendue.

Bibliographie

N. Aifaoui, D. Deneux et R. Soenen : Feature-based interoperability between design and analysis processes. *Journal of Intelligent Manufacturing*, 17:13–27, 2006.

Y. A. Ameur, G. Pierra et E. Sardet : An object approach to represent behavioural knowledge in heterogeneous information systems. *In Proceeding of the International Conference on Object-Oriented Information Systems*, p. 315–339, London, 2000.

Andro : 2008. URL http://www.andromda.org/.

Athena : Athena consortium, athena general description v10, public document. 2004. URL http://www.athena-ip.org/.

D. ATHENA : Report on methodology description and guidelines definition version 1.0, athena integrated project, deliverable d.a1.3.1. Rap. tech., 2005.

S. Bettaieb et F. Noel : A generic architecture to synchronise design models issued from heterogeneous business tools : towards more interoperability between design expertises. *Computer with engineering*, 2006.

S. Bettaieb : *Contribution à la spécification d'un environnement de conception collaborative intégrateur d'expertises métier hétérogènes.* Thèse de doctorat, Institut National Polytechnique de Grenoble, 2006.

F. Bianconi : Towards a procedural cad model for data exchange : Problems and perspectives. *Congreso Internacional Conjunto XVII Ingegraf-XV ADM : De la Tradición al Futuro, Seville : University of Seville*, June 2005.

F. Bianconi, P. Conti et L. D. Angelo : Interoperability among cad/cam/cae systems : A review of current research trends. *In Proceedings of International Conference on Geometric Modeling and Imaging GMAI 2006*, p. 82–89, London, July 2006.

L. Bondé : *Transformations de Modèles et Interopérabilité dans la Conception de Systèmes Hétérogènes sur Puce à Base d'IP*. Thèse de doctorat, Université des Sciences et Technologies de Lille, 2006.

L. Bouzguenda : *Coordination multi-agents pour le Workflow interorganisationnel lâche*. Thèse de doctorat, IRIT, 2006.

J. Bézivin et X. Blanc : Mda : standards et travaux. *Développeur Référence*, 2002.

J. Bézivin, G. Dupé, F. Jouault, G. Pitette et J. Rougui : First experiments with the atl model transformation language : Transforming xslt into xquery. *2nd OOPSLA Workhop on Generative Techniques in the context of Model Driven Architecture*, October 2003.

J. Bézivin et S. Gérard : A preliminary identification of mda components. 2003.

C4ISR : Levels of information systems interoperability (lisi). *C4ISR Interoperability Working Group, Department of Defense*, 1998.

J. G. Campos et M. Hardwick : A traceability information model for cnc manufacturing. *Computer-Aided Design*, 38:540–551, 2006.

E. Chapa : *Outils et structure pour la coopération formelle et informelle dans un contexte de conception holonique*. Thèse de doctorat, l'Institut National Polytechnique de Grenoble, 1997.

S. Charles : *Gestion Intégrée des données CAO et EF-Contribution à la liaison entre conception mécanique et calcul de structures*. Thèse de doctorat, l'Université de Technologie de Troyes, 2005.

D. Chen, G. Doumeingts et F. Vernadat : Architectures for enterprise integration and interoperability : Past, present and future. *Computers in Industry*, 2008.

X. Chen et S. M. Li : A web services based platform for exchange of procedural cad models. *In Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design*, May 2005.

N. Chungoora et R. I. M. Young : *Semantic Interoperability Requirements for Manufacturing Knowledge Sharing*. Springer London, 2008.

R. Claus et I. Weitzer : Cad services-an industry standard interface for mechanical cad interoperability. Rap. tech., Octobre 2002.

K. Czarnecki et S. Helsen : Classification of model transformation approaches. *proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, 2003.

F. Danesi, N. Gardan et Y. Gardan : Collaborative design : from concept to application. *Proceedings of the Geometric Modeling and Imaging GMAI'06*, 2006.

O. Djebbi et M. P. Gervais : Mda : Vers l'industrialisation de la construction d'applications réparties. Mémoire de D.E.A., 2004.

J. Dragonas : *Modélisation déclarative collaborative Systèmes collaboratifs pour la modélisation déclarative en synthèse d'image*. Thèse de doctorat, Universite de limoges, 2006.

EIF : European interoperability framework, white paper. 2004. URL http://www.comptia.org.

I. Elkhalkhali : *Système intégré pour la modélisation, l'échange et le part de données de produits*. Thèse de doctorat, l'Institut National des Sciences Appliquées de Lyon, 2002.

B. Elvesæter, A. Hahn, A.-J. Berre et T. Neple : Towards an interoperability framework for model-driven development of software systems. *In in Proc. of the 1st International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'05)*, ISBN 978-1-84628-151-8, p. 409–420, Geneva, Switzerland, 2005. Springer London.

S. J. Fenves : A core product model for representing design information. *USA : National Institute of Standards and Technology, NISTIR 6736, Gaithersburg, MD 20899, USA,*, 2001.

S. J. Fenves, S. Foufou, C. Bock et R. D. Sriram : Cpm : A core model for product data. *Manufacturing Systems Integration Division, National Institute of Standards and Technology*, 2006.

P. Flynn et A. Jain : Cad-based computer vision : From cad models to relational graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:114–132, 1991.

G. Foucault : *Adaptation de modèles CAO paramétrés en vue d'une analyse de comportement mécanique par éléments finis*. Thèse de doctorat, Université du Québec, MONTRÉAL, 2007.

S. Foufou, S. J. FENVES, C. BOCK, S. RACHURI et R. D. SRIRAM : A core product model for plm with an illustrative xml implementation. *proceedings of the international conference on product life cycle management - PLM'05*, july 2005.

Y. Gardan : *La CFAO*. Hermès Science Publications, 1992.

J. S. Gero : Design prototypes : A knowledge representation scheme for design. *AI Magazine*, 11 :4:26–36, 1990.

J. Gero et U. Kannengiesser : A function-behaviour-structure view of social situated agents. *In A Choutgrajank, E Charoenslip, K Keatruangkamala and W Nakapan (eds), CAADRIA03*, p. PP 707–715, Bangkok, 2003. Rangsit University.

S. Gorti, A.Gupta, G. Kim, R. Sriram et A. Wang : An object-oriented representation for product and design process. *Computer Aided Design*, 30 :1:498–501, 1998.

GTK, 2008 : URL http://www.gtk.org/.

A. Gunendran, R. Young, A. Cutting-Decelle et J. Bourey : Organising manufacturing information for engineering interoperability. *In Interoperability for Enterprise Software and Applications Conference*, Madeira Island, Portugal, 2007.

L. Gzara : *Les patterns pour l'ingenierie des systemes d'information produit*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 2000.

D. Heimbigner et D. McLeod : A federated architecture for information management. *ACM Transactions on Information Systems (TOIS)*, 3:253–278, 1985.

J.-M. Hoc : Towards a cognitive approach to human-machine cooperation in dynamic situations. *International Journal of Human-Computer Studies*, 54:509–540, April 2001.

IDEAS : A gap analysis -required activities in research, technology and standardisation to close the rts gap- roadmaps and recommendations on rts activites. *IDEAS, Deliverables*, 2003.

IEEE : Ieee : Standard computer dictionary - a compilation of ieee standard computer glossaries. 1990.

IGES : Initial graphics exchange specification : Ansi y 14.26m. USA, 1980.

INTEROP : European virtual laboratory for enterprise interoperability, 2008. URL http://interop-vlab.eu/.

IPPOP : http ://ippop.laps.u-bordeaux1.fr/index.php. visited on April 2008a.

IPPOP : http ://ippop.laps.u-bordeaux1.fr/index.php, visited on april 2008. 2008b.

ISO10303-1 : Industrial automation systems – product data representation and exchange – part 1 : Overview and fundamental principles. ISO/IEC, Geneva, Switzerland, 1994.

ISO10303-11 : Industrial automation systems and integration – product data representation and exchange – part 11 : Description methods : The express language reference manual. ISO/IEC, Geneva, Switzerland, 1994.

ISO10303-21 : Industrial automation systems and integration – product data representation and exchange – part 21 : Implementation methods : Clear text encoding of the exchange structure (physical file). ISO/IEC, Geneva, Switzerland, 1994.

P. Jagou : *Concurrent Engineering : La maitrise des couts, des delais et de la qualite*. Hermes, Paris, 1993.

R. Johansen et J. Charles : *GroupWare : computer support for business teams*. New York, USA, 1988.

A. Kalnins, E. Celms et A. Sostaks : Tool support for mola. electr. notes theor. comput. sci. 152 : 83-96 (2006). *Electr. Notes Theor. Comput. Sci.*, 152:83–96, 2006.

H. S. Khan : Achieving reusability through interoperability. *In Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems(ECBS'06)*, 2006.

J. Kim, M. J. Prattb, R. G. Iyerc et R. D. Srirama : Standardized data exchange of cad models with design intent. *Computer-Aided Design*, 2007.

L. Laemmer : Product lifecycle management services v1.0 : Request for proposal. *OMG Document : mantis/2002-10-01*, 2002.

P. Laug : Topologie et maillage des surfaces paramétrées à partir d'une modélisation b-rep. *In 17 ème Congrès Français de Mécanique*, 2005.

R. Leea, J. Tsaib, Y. Kaoc, G. C. Lind et K. Fane : Step-based product modeling system for remote collaborative reverse engineering. *Robotics and Computer Integrated Manufacturing*, 19:543–553, 2003.

R. Lemesle : *Techniques de Modélisation et de Méta-modélisation*. Thèse de doctorat, Université de Nantes, 2000.

A. Leont'ev : The problem of activity in psychology. *In The concept of activity in soviet psychology*, p. 37–71, 1981.

G. A. Lewis et L. Wrage : approaches to constructive interoperability. Rap. tech., Carnegie Mellon Software Engineering Institute, 2004.

W. Li, S. Ong et A. Nee : Integrated and collaborative product development environment : Technologies and implementations. *World Scientific publishing, Series On Manufacturing Systems & Technology*, 2, 2006.

D. Lopes : *Étude et applications de l'approche MDA pour des plate-formes de Service Web*. Thèse de doctorat, Université de Nantes, 2005.

J. Marsot : Conception et ergonomie. Département Ingénierie des Equipement de Travail, centre de Lorainne de l'INRS, 2002.

G. Miaoulis : *Contribution à l'étude des Systèmes d'Information Multimédia et Intelligent dédiés à la Conception Déclarative Assistée par l'Ordinateur - Le projet MultiCAD*. Thèse de doctorat, University of Limoges, 2002.

J. Miller et J. Mukerji : Mda guide version 1.1. Rap. tech. Document number omg/,12, 2003.

M. E. H. Mimoune, G. PIERRA et Y. A. AMEUR : Une approche pour l'échange entre bases de données hétérogènes basée sur des méta-modèles génériques exprimés en langage express. *Actes de la Journée de Travail Bi-Thématique du GDR-PRC I3*, p. 229–246, 2001.

MOF : Mof 2.0 query / views / transformations omg. Rap. tech. Document : ad/2002-04-10, 2002. URL http://www.omg.org/docs/ad/02-04-10.pdf.

MOKA : Moka : A framework for structuring and representing engineering knowledge. 1999a. URL http://www.kbe.coventry.ac.uk/moka/miginfo.htm.

MOKA : Moka user guide, délivrable du consortium moka. 1999b. URL http://www.kbe.coventry.ac.uk/moka.

C. Mundutéguy et F. Darses : *Facteurs de transgression d'un mode de coopération prescrit par l'organisation pour un mode de coopération adapté auproblème à résoudre. Le Travail collectif. Perspectives actuelles en ergonomie*. Toulouse, Octarès, 2000.

NEHTA : Towards an interoperability framework, version 1.8. August 21 2005.

K. Nezamirad, P. G. Higgins et S. Dunstall : Human collaboration in planning and scheduling. *7th International Workshop on Human Factors in Planning, Scheduling and Control in Manufacturing, The University of Groningen, The Netherlands*, june 2005.

F. Noel : *Mailleur auto-adaptatif pour des surfaces gauches en vue de la conception intégrée*. Thèse de doctorat, l'Institut National Polytechnique de Grenoble, 1992.

F. Noel : L'environnement gam. 2007. URL http://www.g-scop.inpg.fr/GAM/.

F. Noël : A product-process-organisation integrative model for collaborative design. *In Innovation in Life Cycle Engineering and Sustainable Development*, p. 407–418, 2006.

F. Noël, J. Léon et P. Trompette : A data structure dedicated to an integrated free-form surface meshing environment. *Computer & Structures*, 57(2):345–355., 1995.

F. Noël, L. Roucoules et D. Teissandier : Specification of product modelling concepts dedicated to information sharing in a collaborative design context. *In Advances in Integrated Design and Manufacturing in Mechanical Engineering*, p. 135–146, 2005.

OMG : Metaobjectfacility(mof) specification, document formal/2002-04-03. Rap. tech., april 2002. URL http://www.omg.org/technology/documents/formal/mof.htm.

OMG : Mda guide version 1.0.1, document number : omg/2003-06-01 edition. Rap. tech., Object Management Group, 2003.

G. Pahl et W. Beitz : *Engineering design, a systematic approach.* 2nd edition, Springer ed, 1996.

J. Park et S. Ram : Information systems interoperability : What lies beneath ? *ACM Transactions on Information Systems (TOIS)*, 22:595–632, 2004.

O. Patrascoiu : Yatl : Yet another transformation language. *In Proceedings of the 1st European MDA Workshop MDA-IA*, January 2004. URL http://www.cs.kent.ac.uk/pubs/2004/1829.

PDML, 2008 : URL http://xml.coverpages.org/pdml.html.

M. Peltier : *Techniques de Transformation de Modèles Basées sur la métamodélisation.* Thèse de doctorat, UFR Sciences et Techniques, Université de Nantes, 2003.

G. Pierra : Représentation et echange de données techniques. *Mécanique & industries*, 1:397–414, 2000.

A. Plantec et V. Ribaud : Un procédé de validation des métamodèles par les métadonnées. *Premières Journées sur l'Ingéniérie Dirigée par les Modèles*, 2005.

S. Ray et A. Jones : Manufacturing interoperability. *journal of Intelligent Manufacturing*, 17:681–688, 2006.

V. Rose, B. Girard, P. Lombard et M. Robin : Modelling collaborative knowledge to support engineering design project manager. *In In 17th IMACS World Congress, Scientific Computation, Applied Mathematics and Simulation*, 2005.

L. Roucoules : *Méthodes et connaissances : contribution au développement d'un environnement de conception intégrée*. Thèse de doctorat, L'Institut National Polytechnique de Grenoble, 1999.

M. Sadeghi, F. Noel et K. Hadj-Hamou : Conflict management in multi-view integrated product modelling systems based on a meta-constraint approache. *In Proceedings of DET2007, 4th International Conference on Digital Enterprise Technology*, Bath, United Kingdom, 19-21 September 2007.

W. J. Schroeder : *Geometric triangulations, with application to fully automatic three-dimensional mesh generation*. Thèse de doctorat, Rensselaer Polytechnic Institute, Troy, New York, 1991.

SET : Z 68-300, industrial automation - external representation of product definition data - data exchange and transfert standard specification. AFNOR-Association Française de NORmalisation, France, 1989.

J. Shah et M. T. Rogers : Functional requirements and conceptual design of the feature-based modeling system. *Computer Aided Engineering*, 5:5–15, 1988.

A. Sheth : Changing focus on interoperability in information systems : from system, syntax, structure to semantics. *Interoperating Geographic Information Systems- Kluwer Academic Publishers, Norwell, MA*, 47:5–29, January 1999.

W. Skarka : Application of moka methodology in generative model creation using catia. *Engineering Applications of Artificial Intelligence*, 20:677–690, 2007.

D. Starzyk : Step and omg product data management specifications : A guide for decision makers. *OMG Document mfg/99-10-04, PDES, Inc. Document MG001.04.00*, 1999.

M. Stokes : Managing engineering knowledge ; moka : Methodology for knowledge based engineering applications. *Ed. Professional Engineering Publishing*, 2001.

S. H. Suh, D. H. Chung, B. E. Lee, S. Shin, I. Choi et K. M. Kim : Step-compliant cnc system for turning : Data model, architecture, and implementation. *Computer-Aided Design*, 38:677–688, 2006.

H. Takeda, M. Yoshioka, T. Tomiyama et Y. Shimomura : Analysis of design process by function, behavior and structure.

F. Tanaka et T. Kishinami : Step-based quality diagnosis of shape data of product models for collaborative e-engineering. *Computers in Industry*, 57:245–260, 2006.

A. B. M. D. Team : Model driven architecture : A technical perspective. Rap. tech. Document number ab/2001-02-04, 2001.

M. Terrasse, M. SAVONNET, G. BECKER et E. LECLERCQ : Uml-based metamodeling for information system engineering and evolution. *In Proc. of the 9th International Conference on Object-Oriented Information Systems, OOIS'03*, p. 83–94, 2003.

M. Terrasse, M. Savonnet, G. Becker et E. Leclercq : *Preface de : Interoperability of Enterprise Software and Applications*. Springer-Verlag, Genève, Suisse, février 2005.

J. Touzi : *Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des entreprises*. Thèse de doctorat, Ecole des Mines d'Albi Carmaux, 2007.

S. D. Urban, K. Ayyaswamy, L. Fu, J. Shah et J. Liang : Integrated product data environment : data sharing across diverse engineering applications. *International Journal of Computer Integrated Manufacturing*, 12(6):525–540, 1999.

VDA-FS : Verband der automobilindustrie flaechen-schnittstelle : Din 66301, din-deutsches instiut für normung. Germany, 1986.

H. Voelcker et A. Requicha : Geometrical modeling of mechanical parts and processes. *IEEE Computer*, 12:48–57, 1977.

L. Wang, W. Shen, H. Xie, J. Neelamkavil et A. Pardasani : Collaborative conceptual design-state of the art and future trends. *Computer-Aided Design*, 34:981–996, 2002.

W.C.Burkett : Product data markup language : a new paradigm for product data exchange and integration. *Computer Aided Design*, 33:489–500, 2001.

K. Weiler : *Topological structures for geometric modelling*. Thèse de doctorat, Rensselaer Polytechnic Institute, Troy, New York, août 1986.

XMI : Mof 2.0/xmi mapping, version 2.1.1. Rap. tech., 2007. URL http://www.omg.org/cgi-bin/doc?formal/2007-12-01.

X. Xu, U. Weiss et G. Gao : The integration of cad/cam/cae based on multi model technology in the development of cylinder head. *International Journal of Automotive Technology*, 3, 2002.

Resumé. Le travail collaboratif fait participer des équipes qui doivent partager et échanger l'information sur le produit tout en travaillant avec différents outils métiers. Les processus complexes de conception du produit exigent l'utilisation d'un ensemble d'outils métiers hétérogènes tels que les outils de CAO et de FAO. Notre travail de thèse propose une approche fédérée basée sur l'Ingénierie Dirigé par les Modèles pour l'interopérabilité des outils métiers. Dans ce travail nous utilisons le modèle PPO (Produit Processus Organisation) comme modèle de fédération pour l'échange et le partage d'informations. Ce travail décrit comment nous structurons la synchronisation entre outils métiers hétérogènes. Un cas d'étude est appliqué à l'outil de FAO $Esprit^{TM}$ lui permettant d'interopérer avec l'outil de CAO $SolidWorks^{TM}$ à travers l'environnement collaboratif PPO.

Mots-clés : Conception Collaborative, interopérabilité, transformations de modèles, synchronisation, méta-modélisation, IDM.

Abstract. Collaborative work involves many teams member which must share and exchange information about the product while working with various business tools. The complex process of product design requires the use of heterogeneous tools such as CAD and CAM tool. Our work of thesis proposes an approach based on the Model Driven Architecture (MDA) for the interoperability of business tools. In this work we use the PPO model (Product Process Organization) as a federation model for information exchange. This work describes how we structure the synchronization between heterogeneous business tools. A case study is applied to $Esprit^{TM}$ CAM tool enabling $Esprit^{TM}$ to interoperate with the $SolidWorks^{TM}$ CAD tool throughout the PPO collaborative environment.

Keywords : Collaborative design, Interoperability, model transformation, synchronisation meta-modelisation, MDE.