Thèse de doctorat

présentée par

Thomas PAVIOT

pour l'obtention du grade

de Docteur de l'École Centrale Paris

Méthodologie de résolution des problèmes d'interopérabilité dans le domaine du Product Lifecycle Management

soutenue le 1er juillet 2010 devant le jury composé de

Abdelaziz BOURAS	*Professeur, Université Lyon 2*	*Président*
Michel GOURGAND	*Professeur, ISIMA*	*Rapporteur*
Bernard GRABOT	*Professeur, ENI Tarbes*	*Rapporteur*
Lionel ROUCOULES	*Professeur, Arts&Métiers ParisTech*	*Rapporteur*
Alain RIVIÈRE	*Professeur, Supméca*	*Examinateur*
Samir LAMOURI	*Professeur, Supméca*	*Directeur*
Vincent CHEUTET	*Maître de Conférences, Supméca*	*Encadrant*

Année 2010
Numéro d'Ordre : 2010ECAP0020

École Doctorale *Sciences pour l'Ingénieur* - ED287
École Centrale Paris

Remerciements

Je tiens en premier lieu à remercier Abdelaziz Bouras de m'avoir fait l'honneur de présider le jury qui a examiné mon travail. Merci à MM. Michel Gourgand, Bernard Grabot et Lionel Roucoules d'avoir relu et critiqué ce manuscrit. J'espère que la chance se présentera à nouveau de pouvoir confronter mon point de vue à leur Science. Merci à Alain Rivière, directeur du LISMMA, d'avoir accepté de figurer au casting de ce jury et de me témoigner de la sorte cette marque de confiance.

Je remercie particulièrement Samir Lamouri et Vincent Cheutet pour la qualité de leur encadrement. Outre leur compétence scientifique, c'est aussi la relation de confiance qu'ils ont su établir qui a rendu notre collaboration efficace. Vincent s'est montré toujours disponible, à l'écoute, critique, d'une grande rigueur et animé d'un constant souci du détail. J'ai par ailleurs apprécié en Samir un homme droit, incarnation parfaite de l'Honnête Homme tel qu'il est apparu au XVII^{ème} siècle sous les plumes de La Rochefoucault ou Blaise Pascal, doué qui plus est d'une surprenante et quasi infaillible intuition quant à la nature profonde des gens qui l'entourent et de la juste manière de les motiver. Compétence essentielle, mais rare, d'un meneur d'hommes, il a su faire en sorte que je donne le meilleur de moi-même.

Je remercie également mes collègues du LISMMA, de Supméca ou d'autres établissements pour la richesse des échanges que j'ai pu avoir avec eux. Parmi eux, je dois beaucoup à Alain Bellacicco, qui fait honneur au service public d'éducation et qui restera la figure marquante de ma carrière d'enseignant. Merci aussi à Véronique, Lionel et Christel, compagnons de route ô combien essentiels. J'associe également à ces remerciements tous les élèves que j'ai encadrés ces dix dernières années : leurs questions les plus anodines ont souvent amené la douloureuse remise en cause de certitudes que je croyais bien ancrées. En ce sens, ils ont participé à préparer le terrain de ce travail de recherche.

Enfin, je remercie famille et amis, qui m'ont toujours soutenu et fait confiance : mes parents, Claude et Michèle, mes beaux-parents, Michel et Renée, et bien sûr Émilie, dont le soutien a donné sens à ce travail et dont la présence donne sens à ma vie.

à Justin et Solène.



Claude Monet peignant dans son atelier - Édouard Manet, 1874.

Conventions typographiques utilisées dans ce document

Ce mémoire est composé de cinq *chapitres*. Chaque chapitre est identifié par un numéro. Les renvois à des chapitres dans le corps du texte sont faits de cette manière : cf chapitre 1, voir 3 etc. Chaque chapitre est composé de plusieurs *sections* et *sous-sections* identifiées par une numérotation arabe. Les renvois à des sections dans le corps du texte sont faits de cette manière : cf section 2.4, voir 2.3 etc. Des appendices figurent en fin de mémoire. Ils sont identifiés par des lettres majuscules (par exemple B ou C). Les appendices complètent la lecture du mémoire, mais ne participent pas de la réflexion scientifique présentée.

Dans la version électronique de ce document (au format PDF), des liens hypertexte permettent de naviguer dans la structure du texte : des hyperliens vers des éléments de structure en noir (chapitre 1, section 1.1 , sous-section 1.1.1), ou vers des références bibliographiques en bleu (exemple : [Botta-Genoulaz et al., 2005]). Pour ce travail, nous avons consulté *146* références.

Les notes de bas de page [1] sont utilisées pour compléter les informations figurant dans le texte (entre autres pour insérer des références ne pouvant être considérées comme scientifiques). Les liens pointant vers des sites Internet apparaissent de couleur rose. Ils ont été validés le 16 mai 2010.

Des exemples de programmes informatiques figurent dans ce mémoire. Ils sont représentés dans des boîtes à double-cadre, dans une typographie à `chasse fixe` [2]. La référence à des `classes` ou `propriétés` dans le corps du texte utilise également une typographie à `chasse fixe`. La coloration syntaxique des mots-clés ou chaînes de caractères permet de faciliter la lecture des programmes.

```
Exemple de ligne de programme informatique
```

Ce mémoire a été rédigé avec les logiciels LyX 1.6.5/LaTeX (modèle KOMA-Script Book [3]). Les références bibliographiques ont été organisées à l'aide des logiciels Papers 1.9.4 et BibDesk 1.5.2/BibTex.

1. Exemple de note de bas de page
2. http ://fr.wikipedia.org/wiki/Chasse_(typographie)
3. http ://www.tex.ac.uk/tex-archive/macros/latex/contrib/koma-script/scrguien.pdf

Table des matières

Contexte de recherche **xiii**

Structure de la thèse **xvi**

1. Problématique de recherche **1**
1.1. Product Lifecycle Management, complexité et stratégie 2
1.1.1. Définitions du PLM 2
1.1.2. Objectif du PLM 7
1.1.3. Le PLM vu selon un angle stratégique/tactique/opérationnel . . . 11
1.1.4. Interopérabilité et PLM 12
1.2. Cas particulier de la conception et de la production dans le développement de produits 12
1.2.1. Les métiers de la conception et de la production 12
1.2.2. Les ambiguïtés sémantiques comme obstacle à la communication conception/production 23
1.2.3. Les systèmes d'information comme soutien aux activités de conception et de production 27
1.3. État de l'art concernant l'interopérabilité des systèmes d'information de la conception et de la production 35
1.3.1. Étendue des problèmes d'interopérabilité 35
1.3.2. Les travaux de la littérature concernant l'interopérabilité dans le domaine du PLM 36
1.4. Vers une méthodologie générale permettant d'appréhender les problèmes d'interopérabilité dans le domaine du Product Lifecycle Management . . 39

2. Une architecture orientée services basée sur un méta-modèle d'unification **41**
2.1. Interopérabilité des systèmes d'information d'entreprise 41
2.1.1. Définitions de l'interopérabilité 41
2.1.2. Les niveaux d'interopérabilité 42
2.1.3. Niveaux d'interopérabilité et flux sémantique 43
2.2. Interopérabilité technique 44
2.2.1. Architectures envisageables 44
2.2.2. Architectures orientées services 46
2.2.3. Le médiateur d'informations orienté services 48

2.2.4. Médiation multi-échelle orientée services 49
2.2.5. Prescriptions méthodologiques pour le choix de l'architecture . . . 52
2.3. Interopérabilité sémantique 52
2.3.1. Intégration, unification et fédération de modèles 53
2.3.2. Prescriptions méthodologiques pour la mise en correspondance sémantique des modèles de données 54
2.3.3. Méta-modèle d'unification *ad hoc* / standard 54
2.3.4. Les standards du PLM 57
2.3.5. La norme ISO-10303 STEP 60
2.3.6. Le protocole d'application *Product LifeCycle Support* 64
2.3.7. Cartographie 3D du standard STEP 68
2.4. Spécifications quant au choix du modèle d'unification standard 69

3. Modèles pour l'interopérabilité sémantique 73
3.1. Aspects méthodologiques 73
3.2. Médiation intra-phase : exemple de l'interopérabilité CAO / PDM 74
3.2.1. Situation initiale - Origine du problème d'interopérabilité 74
3.2.2. Phases du cycle de vie du produit 75
3.2.3. Nature et granularité des informations à échanger 76
3.2.4. Recherche des similarités sémantiques 77
3.2.5. Modélisation des informations à échanger 79
3.2.6. Représentation UML du méta-modèle d'unification 86
3.2.7. Conclusions méthodologiques 86
3.3. Médiation inter-phase : exemple de l'interopérabilité PDM/ERP 87
3.3.1. Situation initiale - Origine du problème d'interopérabilité 88
3.3.2. Phases du cycle de vie du produit 89
3.3.3. Similarités sémantiques entre les informations : EBOM / MBOM 89
3.3.4. Nature et granularité des informations à échanger 90
3.3.5. Modélisation du produit dans STEP PLCS 91
3.3.6. Représentation multi-vues dans STEP PLCS 93
3.3.7. Diagramme UML du méta-modèle d'unification 96
3.3.8. Conclusions méthodologiques 97
3.4. Un modèle basé sur des *Tags sémantiques* pour la structuration du produit 98
3.4.1. La sémantique mélangée à la matière d'œuvre des processus de conception et de fabrication 98
3.4.2. Web collaboratif et activité de *tagging* 101
3.4.3. Cas d'étude pour le modèle de tag sémantique 102
3.4.4. Modèle de tag sémantique 103
3.4.5. Conclusion 107

4. Validation de l'architecture et de la méthodologie associée 109
4.1. Réalisation de l'interopérabilité technique 110
4.1.1. Langage de programmation 110
4.1.2. SOA et services web 111

4.1.3. Médiation en flux poussé ou flux tiré . . . 113
4.2. Réalisation de l'interopérabilité CAO/PDM . . . 120
4.3. Réalisation de l'interopérabilité PDM/ERP . . . 124
4.3.1. Implémentation de STEP PLCS . . . 126
4.3.2. Implémentation de l'architecture . . . 128
4.3.3. Résultats . . . 128
4.4. Implémentation du modèle de tags sémantiques . . . 130
4.4.1. Implémentation des classes du diagrammes UML . . . 131
4.4.2. Instanciation des items, des tags et des relations . . . 132
4.4.3. Algorithmes de traitement du modèle . . . 134
4.5. Recommandations méthodologiques relatives à l'implémentation . . . 140

5. Conclusion - Perspectives 143
5.1. Apports de la thèse . . . 143
5.2. Travaux complémentaires à envisager . . . 144
5.3. Perspectives . . . 145
5.3.1. Vers des architectures distribuées . . . 146
5.3.2. De nouvelles méthodes pour de nouvelles échelles . . . 148
5.3.3. Solution intégrée propriétaire *vs.* Applications ouvertes interopérables 149
5.3.4. Vers une machine automatisée de conception . . . 150
5.3.5. La place de l'homme face à cette évolution technologique . . . 151

A. Organisation de la norme STEP 153

B. Le langage de modélisation EXPRESS 155
B.1. Introduction . . . 155
B.2. Schéma EXPRESS . . . 155
B.3. Entités . . . 156
B.4. Typage des attributs . . . 157
B.5. Contraintes . . . 158
B.6. EXPRESS-G . . . 161

C. La bibliothèque de développement pySDAI 163
C.1. Introduction . . . 163
C.2. Contacts . . . 163
C.3. Licence d'utilisation . . . 164
C.4. Qualité . . . 164
C.5. Le schéma *Family* . . . 165
C.6. Pré-requis . . . 166
C.7. Compilation du schéma Family et import dans Jython . . . 166
C.8. Manipulation de schémas EXPRESS/STEP . . . 167
C.8.1. Initialisation . . . 167
C.8.2. Mise en place de l'environnement SDAI . . . 168
C.8.3. Création d'un modèle SDAI . . . 168

D. Mentions légales **171**

Bibliographie **173**

Table des figures

1.2.1.Actigramme de la fonction conception 16
1.2.2.Le rôle de la gestion industrielle 17
1.2.3.Actigramme de la fonction production 20
1.2.4.Interface conception/production 21
1.2.5.Besoin d'échange d'informations conception/production durant le processus de développement du produit 22
1.2.6.Les ambiguïtés sémantiques comme filtre à une bonne collaboration conception/production 26
1.2.7.Classification des logiciels utilisés en conception 28
1.2.8.Le volet applicatif du système d'information PLM 33
1.3.1.Catégorie des problèmes d'interopérabilité rencontrés dans la littérature . 35

2.2.1.Approches point-à-point/médiateur pour l'interopérabilité technique . . . 45
2.2.2.Le médiateur d'informations orienté services selon [Bénaben et al., 2008] 49
2.2.3.Médiation multi-échelle 50
2.2.4.Fonction de répartition d'un médiateur de la médiation multi-échelle . . . 51
2.3.1.Le médiateur multi-échelles orienté services - Unification des informations sémantiques 56
2.3.2.Cartographie 2D des standards du PLM [Rachuri et al., 2008] 59
2.3.3.Périmètre fonctionnel du PDM Schema 62
2.3.4.La modularité de la norme STEP : *Modules d'Application* (AM) et *Protocoles d'Application* (AP) 63
2.3.5.Cartographie 3D du standard STEP 68

3.2.1.Le projet FALSIM (Final Assembly Line) Supméca/EADS IW 75
3.2.2.La collaboration à un haut niveau granulaire 76
3.2.3.Modèle graphe des dépendances entre fichiers CAO pour une maquette numérique 82
3.2.4.La représentation d'un article dans Windchill (courtesy of PTC) 83
3.2.5.Modèle graphe de la structure produit PDM (Windchill) 84
3.2.6.Composants fabriqués/achetés 85
3.2.7.Méta-modèle d'unification pour l'interopérabilité CAO/PDM 86
3.3.1.Cas d'étude pour l'interopérabilité PDM/ERP 88
3.3.2.Modélisation du produit dans STEP PLCS 91
3.3.3.Modélisation d'une version d'un produit 92
3.3.4.Modélisation multi-vues dans STEP PLCS 94

3.3.5.Modèle EXPRESS-G `representing_part` 95
3.3.6.Modèle EXPRESS-G `representing_assembly_structure` 96
3.3.7.Diagramme UML pour le méta-modèle d'unification 97
3.4.1.Espace des matières d'œuvre/espace sémantique 101
3.4.2.Cas d'étude pour le modèle de tags sémantiques 103
3.4.3.Un premier modèle de données pour le tag sémantique 104
3.4.4.Modèle de données pour le groupe sémantique 106
3.4.5.Exemple d'instanciation du modèle de tags sémantiques 106

4.1.1.Test de mise en oeuvre des bibliothèques `soaplib`/`suds` 113
4.1.2.Validation de la continuité des données pour la médiation en flux poussé 116
4.1.3.Validation de la continuité des données pour la médiation en flux tiré . . 119
4.2.1.Webservicisation du logiciel Catia V5 122
4.2.2.Accès au service *ping* d'*Info*Engine* 123
4.3.1.Architecture fonctionnelle orientée Services 125
4.3.2.Mapping entre les nomenlcatures PDM et ERP 128
4.3.3.Résultats de l'expérimentation pour l'interopérabilité PDM/ERP 129

B.2.1Définition d'un Schéma EXPRESS 156
B.3.1Définition de l'entité `Decision_point` 157
B.4.1Types prédéfinis 158
B.5.1Contrainte locale WHERE RULE 159
B.5.2Contrainte globale d'unicité pour un attribut 159
B.5.3Contrainte de cardinalité inverse 160
B.5.4Règle globale 160
B.6.1Syntaxe EXPRESS-G 161
B.6.2Représentation EXPRESS-G de l'entité `hour_in_day` 162

C.5.1Le schéma EXPRESS-M *Family* 165
C.7.1Le schema EXPRESS family dans l'éditeur intégré à Eclipse 167
C.8.1Script Python pour l'instanciation du modèle de Famille 169
C.8.2Fichier ISO-10303-Part 21 170

Liste des tableaux

2.3.1.Correspondance bi-univoque entre les niveaux d'interopérabilité et les types de standard 58
2.4.1.Les deux catégories de problèmes d'interopérabilité 70

3.2.1.La structuration du produit dans les logiciels de CAO3D 78
3.2.2.Structure produit et visualisation 3D dans les systèmes PDM(exemple de Windchill et TeamCenter) 79
3.2.3.Vue 3D, structure produit et fichiers de définition d'un préhenseur industriel (Inventor). 80
3.3.1.Nomenclature de fabrication dans les systèmes ERP 90

Liste des symboles

AM Application Module

AP Application Protocol

APS Advanced Planning System

ASCII American Standard Code for Information Interchange

CGP Coût Global de Possession

DEX Data EXchange set

DMU Digital Mock Up

ECO Engineering Change Order

ECR Engineering Change Request

ERP Enterprise Resource Planning

IPPOP Integration of Product-Process-Organization for engineering Performance improvement

JSDAI Java Standard Data Access Interface

MBOM Manufacturing Bill Of Material (nomenclature de production)

MES Manufacturing Execuation System

PDM Product Data Management

PLM Product Lifecycle Management

PPO Product Process Organization

RPC Remote Procedure Call

SADT Structural Analysis and Design Technic

SCM Supply Chain Management

SDAI Standard Data Access Interface

SI Système d'Information

W3C World Wide Web Consortium

XML eXtended Markup Language

Contexte de recherche

En 2006, partant du constat que les activités scientifiques de ses chercheurs étaient trop éclatées, le Laboratoire d'Ingénierie des Systèmes Mécaniques et des Matériaux (LISMMA) a entrepris une démarche de rapprochement de ses équipes de recherche. Dans le fil des conclusions du mémoire d'Habilitation à Diriger des Recherches du Pr. Samir Lamouri [Lamouri, 2006], l'équipe « Optimisation des Systèmes Industriels et Logistique » (OSIL) a été rebaptisée en 2007 « Conception et Optimisation des Systèmes Industriels » (COSI). Ce changement de nom s'est accompagné d'une redéfinition des objectifs de recherche : la thématique « PLM et Logistique » a été initiée pour lier les compétences du LISMMA en production et en conception, dans l'objectif de répondre aux nouveaux défis scientifiques identifiés. Pour aller plus loin dans ce processus de fédération des énergies et des compétences, c'est sous la bannière commune « Ingénierie Numérique », créée en 2008, qu'ont été invités à collaborer les chercheurs du LISMMA se réclamant des sciences de la conception et de la production.

Parallèlement à ce changement d'organisation des activités de recherche, Supméca (établissement d'enseignement supérieur auquel est adossé le LISMMA) a entrepris une évolution de ses enseignements. La spécialisation « Usine Numérique » a été créée à la rentrée 2007 pour répondre aux besoins des entreprises industrielles et mettre en cohérence les enseignements de conception et de production.

J'ai été recruté en 2007 à Supméca, en tant que professeur agrégé de mécanique, pour assurer le développement de cette nouvelle formation. C'est dans ce cadre que j'ai souhaité mener un travail de recherche parallèlement à mes activités d'enseignement, officialisé par une inscription en thèse auprès de l'École Doctorale de l'École Centrale Paris au mois de décembre 2007.

Le travail présenté dans ce mémoire est la première thèse soutenue dans la thématique « PLM et Logistique ». Compte tenu du contexte exposé précédemment, cette recherche se situe aux interfaces de la conception et de la production, et couvre un champ de compétences relevant des 60ème (mécanique, génie mécanique, génie civil), 61ème (génie informatique, automatique, traitement du signal) et 27ème (informatique) sections définies par le Conseil National des Universités.

Thomas Paviot [4]

4. Pour toute remarque ou question relative à ce travail, contacter tpaviot@gmail.com

Structure de la thèse

Le travail de thèse présenté dans ce mémoire s'inscrit dans le domaine du Product Lifecycle Management. Il s'intéresse aux problèmes d'interopérabilité entre les systèmes d'information chargés du pilotage des activités de conception et de production des produits industriels.

Le chapitre 1 situe la problématique de notre travail de recherche. La définition du Product Lifecycle Management(PLM) est précisée. Le périmètre du travail est ensuite restreint aux domaines de la conception et de la fabrication des produits industriels. L'interopérabilité des systèmes d'information qui pilotent la conception et la fabrication est identifiée comme une nécessité compte tenu des contraintes économiques fortes qui pèsent sur les entreprises manufacturières. Les ambiguïtés sémantiques des deux domaines (conception/production) sont soulignées comme étant la source des problèmes d'interopérabilité. Un état de l'art relatif au sujet permet de dresser un panorama des approches scientifiques. Ce chapitre se conclut par un exposé des verrous scientifiques à lever et des objectifs de notre travail de thèse, centré sur les cas spécifiques d'interopérabilité CAO↔PDM et PDM↔ERP.

Le chapitre 2 définit l'architecture d'un système informatique capable d'assurer l'interopérabilité de plusieurs systèmes d'information relevant du PLM. L'interopérabilité technique est assurée, d'un point de vue fonctionnel, par une architecture orientée services (SOA) basée sur un composant tiers : le médiateur multi-échelle orienté services. L'interprétation des données échangées (ou interopérabilité sémantique) s'oriente vers l'utilisation des standards existants. La norme STEP est présentée d'une manière générale, puis un de ses protocoles d'application spécifique, intitulé *Product Lifecycle Support* (PLCS), est identifié comme étant pertinent par rapport à notre travail.

Le chapitre 3 présente la méthodologie que nous avons définie pour satisfaire aux besoins présentés en fin de chapitre 1, en s'appuyant sur l'architecture du chapitre 2. Les modèles utilisés font l'objet de la deuxième partie de ce chapitre, en particulier la modélisation du produit et de sa structuration conformément aux spécifications de la norme STEP PLCS. La *structure produit* est présentée comme étant un concept ambigu d'un point de vue sémantique, en particulier les liens de structure de type parent/enfant. Nous introduisons alors un méta-modèle de structure produit, basé sur des tags sémantiques affectés aux constituants du produit, qui peut être instancié sur des modèles existants (qu'ils relèvent ou non de la norme STEP).

Le chapitre 4 s'intéresse à la validation expérimentale de l'architecture SOA définie dans le chapitre 2, utilisant la méthodologie et les modèles présentés dans le chapitre 3.

Les détails relatifs à la résolution des problèmes d'implémentation sont présentés. Dans un deuxième temps, des scénarios d'échange de données CAO↔PDM et PDM↔ERP, conformes à des spécifications industrielles, permettent d'illustrer et valider notre travail.

Le chapitre 5 conclut ce mémoire par une synthèse des apports de cette thèse. La dernière partie ouvre la voie aux perspectives de recherche que nous avons identifiées.

1. Problématique de recherche

Ce chapitre a pour objet de décrire le périmètre du travail de thèse présenté dans ce mémoire. La recherche que nous avons menée se concentre sur l'interopérabilité des systèmes d'information relevant du Product Lifecycle Management (PLM). Cette problématique très large doit, dans un premier temps, être précisée quant aux domaines d'application de la conception et de la production des produits industriels, qui font l'objet de notre étude. Nous avons orienté cette première partie de l'état de l'art suivant deux idées-forces :

- le contexte du PLM est celui d'un accroissement de la complexité, la notion de complexité étant à prendre ici dans le sens de celle définie par l'ingénierie système ;
- les problèmes d'interopérabilité trouvent leur origine dans les ambigüités sémantiques inhérentes aux métiers concernés. Cet aspect sémantique de l'interopérabilité sera largement développé dans les chapitres suivants.

Ces deux éléments étant introduits, nous présentons ensuite de manière synthétique les systèmes d'information qui pilotent les activités de la conception et de la production. Nous mettons en correspondance bijective les fonctions de ces systèmes et les métiers associés. De la même façon, nous mettons en lumière les besoins d'interopérabilité de ces systèmes au regard des besoins métier, et les obstacles à l'interopérabilité comme conséquence des ambigüités sémantiques. Dans la deuxième et dernière partie de ce chapitre, nous présentons les travaux ayant contribué à lever certains verrous scientifiques, et nous introduisons le chapitre suivant au vu des questions encore ouvertes.

Par conséquent, le présent chapitre est structuré de la manière suivante : la section 1.1 définit préalablement le PLM en termes d'objectifs, de fonctions, d'outils et de méthodologies dans un contexte de complexité. La section 1.2 traite des caractéristiques spécifiques des métiers de la conception et de la production, de leur interface, des systèmes d'information qui leur sont associés ainsi que du besoin d'interopérabilité entre ces métiers/systèmes d'information dans un contexte PLM. La section 1.3 fait état des travaux scientifiques relatifs à ce sujet. Les limites des approches existantes sont présentées dans la section 1.4. Cette dernière section (et ce chapitre) se conclut par un exposé des pistes que nous avons explorées et que nous présentons dans la suite de ce mémoire.

1.1. Product Lifecycle Management, complexité et stratégie

L'expression anglaise *Product Lifecycle Management*, souvent désignée par son acronyme PLM, est habituellement traduite en langue française par *Gestion du cycle de vie des produits*. Après l'apparition de cet acronyme dans les années 2000 [Lamouri, 2006], le PLM est devenu un marché important en volume et en valeur [CIMdata, 2009]. La grande majorité des entreprises industrielles du secteur manufacturier ont engagé (ou ont mis en place) des projets PLM. Dans la lignée des grands groupes, les Petites et Moyennes Entreprises (PME[1]), fournisseurs ou sous-traitant des grands donneurs d'ordre, investissent aujourd'hui dans des projets de transition vers une démarche PLM [Kadiri et al., 2009].

Situer la problématique de l'interopérabilité dans le domaine du PLM requiert préalablement que la notion de PLM soit précisée. Le sens même de l'expression *Product Lifecycle Management* n'est pas clair, pas plus que sa traduction française *gestion du cycle de vie des produits*. D'autre part, la multitude de définitions différentes ne contribue pas à éclaircir ce concept. L'objectif de cette section est donc de dresser un état de l'art permettant d'aboutir à une vision claire de la notion de PLM. Dans un premier temps (voir 1.1.1), nous rappelons quelques définitions de la littérature. Au vu de l'ambigüité de ces éléments, nous proposons de structurer la notion de PLM autour de deux points : l'objectif et la couverture fonctionnelle. L'objectif du PLM (voir 1.1.2) est défini à la lumière de la notion de *complexité* alors que le champ du PLM (voir 1.1.3) est structuré suivant une approche stratégique/tactique/opérationnelle. En fin de section (voir 1.1.4), l'interopérabilité dans le domaine du PLM est mise en regard de ces définitions.

1.1.1. Définitions du PLM

L'objectif de cette partie n'est pas de dresser une liste exhaustive des nombreuses définitions du PLM que l'on peut rencontrer, mais plutôt de présenter le point de vue des grands industriels impliqués dans le domaine, ainsi que de quelques travaux scientifiques. Cet échantillon (on se reportera à [Terzi, 2005] pour un panorama complet) permettra de mettre au jour en fin de section les éléments sur lesquels se sont appuyés nos travaux.

Les besoins industriels Il nous semble cohérent d'introduire ce chapitre par l'exposé du point de vue de quelques grands industriels : ce sont leurs besoins qui tirent le marché du PLM et eux qui expriment des problématiques nécessitant des travaux scientifiques.

1. D'après la recommandation de la Commission n° 2003/361/CE du 6 mai 2003, une PME est une entreprise qui occupe moins de 250 personnes et dont le chiffre d'affaires annuel n'excède pas 50M€ ou dont le total du bilan annuel est inférieur à 43M€.

On trouvera donc ci-dessous les points de vue de représentants d'EADS, Volvo Car Corporation, Vinci Consulting et CIMData.

Le groupe aéronautique et de défense EADS est engagé depuis l'année 2007 dans un projet PLM de grande envergure intitulé *PHENIX*[2] (acronyme pour *PLM H*armonization for *EN*hanced *I*ntegration and e*X*cellence). Pour Jean-Yves Mondon[3], son responsable, ce projet doit permettre de *faire face à la complexité* des programmes. Le PLM est défini comme un *capital stratégique pour l'entreprise étendue, au cœur du système industriel*.

CIMData, cabinet de conseil exerçant dans ce domaine, définit[4] pour sa part le PLM comme une *approche stratégique qui met en œuvre un ensemble cohérent de pratiques permettant de supporter la création collaborative ainsi que l'organisation, la diffusion et l'utilisation des informations relatives à la définition du produit au travers de l'entreprise étendue, de la conception à la fin de vie, et d'intégrer les hommes, les processus, les systèmes d'organisation et d'information*.

Bertil Turesson, PLM Strategist chez Volvo Car Corporation reprend à son compte [Berkooz, 2007] le point de vue exprimé par [Stark, 2005] : *le PLM est l'activité qui consiste à gérer les produits de l'entreprise, tout au long de leur cycle de vie, de la manière la plus efficiente. Le PLM est holistique*[5] *: son champ couvre de nombreux éléments comme les processus, les données, les hommes et les applications. Cela le distingue d'activités atomiques qui s'attachent à un point particulier, comme par exemple le Product Data Management (PDM).* Selon l'auteur, le PLM peut être vu comme *un paradigme de rassemblement : il fédère de nombreux processus, disciplines, fonctions et applications jusqu'alors hétérogènes et indépendants, qui adressent un même produit avec différents vocabulaires, règles métiers, cultures ou langues*.

Ces quelques définitions permettent de situer globalement le cadre de notre discours, et font apparaître les mots-clés permettant d'illustrer les besoins des industriels : l'aspect stratégique du PLM, les phases du cycle de vie, les difficultés sémantiques, la complexité, les systèmes d'information. Nous poursuivons l'exploration de ce panorama industriel en nous intéressant à la deuxième catégorie d'acteurs du marché du PLM : les éditeurs de logiciels, qui développent des outils répondant à ces besoins.

Le point de vue des éditeurs Le marché du PLM pour l'année 2008 est estimé par CIMData[6], au niveau mondial, à une quinzaine de milliards d'euros en valeur. Les

2. http ://www2.pdteurope.com/media/54168/3. phenix - an eads programme for plm.pdf
3. http ://www.vinci-consulting.com/filerepository/publications/FR/revue_de_presse/VIN30-INT-0812a 01 Informatique 08_01_EADS.pdf
4. http ://www.cimdata.com/plm/definition.html
5. L'holisme est un terme inventé par l'homme d'état sud-africain J.C. Smuts en 1926 pour indiquer la tendance de l'univers à construire des unités structurales de complexité croissante formant un tout cohérent.
6. http ://www.cimdata.com/publications/pdf/PLM Market Growth in 2008 Mid-Year_200908 Final.pdf

prévisions pour les cinq prochaines années sont du même ordre de grandeur. D'après ce rapport, cinq éditeurs se partagent le marché : Dassault Systèmes, Siemens PLM, PTC, SAP et Oracle, cités dans l'ordre décroissant de leur part de marché. Les définitions qui suivent n'offrent aucune garantie scientifique : elles n'ont fait l'objet d'aucune validation, et il n'est pas question dans notre travail de s'y référer pour construire notre recherche. Néanmoins, ces positions permettent de situer la stratégie commerciale de chaque éditeur vis-à-vis de sa communication envers ses clients.

Dassault Systèmes propose la définition [7] suivante : *le PLM est une stratégie d'entreprise qui aide les entreprises à partager les données produit, à appliquer des procédés communs et à capitaliser les informations de l'entreprise pour le développement de produits, de la conception à la mise au rebut, et dans tous les segments de l'entreprise étendue. En incluant tous les acteurs (collaborateurs de l'entreprise, partenaires, fournisseurs, équipementiers et clients), la gestion du cycle de vie du produit permet à ce réseau de fonctionner en tant qu'entité unique de la conception à la maintenance, en passant par la fabrication.*

Pour Siemens PLM [8], *le PLM permet à l'entreprise de prendre des décisions cohérentes et pilotées par les systèmes d'information, à chaque phase du cycle de vie du produit. Les solutions PLM mettent en place une plateforme intégrée pour :*

- *optimiser les relations au travers de toutes les phases du cycle de vie et de toutes les organisations impliquées ;*
- *maximiser dans le temps la valeur du portefeuille de produits ;*
- *mettre en place un système unique pour l'enregistrement et la diffusion des données numériques.*

Selon PTC [9], *la gestion du cycle de vie des produits est une technologie qui permet de faire évoluer les informations techniques du concept au retrait et qui, ce faisant, représente un avantage certain pour l'entreprise.*

Pour SAP [10], *les logiciels vendus permettent aux entreprises clientes de :*

- *créer et proposer des produits innovants répondant ou générant de la demande ;*
- *optimiser les processus et systèmes de développement de produit pour une mise sur le marché plus rapide, tout en garantissant une conformité sectorielle, qualitative et réglementaire ;*
- *être plus flexible par rapport aux concurrents, être capable de réagir et de tirer parti des opportunités du marché.*

Enfin, d'après Oracle [11], *les solutions PLM permettent aux entreprises d'accélérer l'innovation et de maximiser la rentabilité des produits par la gestion de l'information, des processus et des décisions concernant le produit, tout au long de leur cycle de vie.*

7. http ://www.3ds.com/fr/plm-glossary/plm/
8. http ://www.plm.automation.siemens.com/en_us/plm/definition/
9. http ://www.ptc.com/solutions/product-lifecycle-management/index.htm
10. http ://www.sap.com/france/solutions/business-suite/plm/index.epx
11. http ://www.oracle.com/global/fr/applications/agile/index.html

Les points de vue des cinq acteurs majeurs du PLM en terme de solutions informatiques illustre l'évolution des besoins de leurs clients : les logiciels informatiques sont bien au cœur du PLM mais ce dernier ne s'y résume pas. En conséquence, les éditeurs cités ont fait évoluer leur modèle économique de la vente de licences (modèle *produit*) à une offre composite vente de licences/prestation de service (modèle *produit/service*) permettant à leurs clients de tirer le meilleur parti (en termes de *retour sur investissement*) des technologies proposées. L'adaptation du produit au métier du client et la définition de méthodologies efficientes quant à son utilisation sont des prestations indissociables de la vente du produit en lui-même. Ces prestations de service peuvent être assurées par la branche service de l'entreprise (ex : Dassault Data Service) ou confiées à un partenaire extérieur reconnu pour sa compétence (IBM Global Services ou SSII revendeurs/intégrateurs comme Accenture, ATOS, Cap Gemini etc.).

[Terzi, 2005] montre, en 2005, qu'aucune définition universelle et acceptée du PLM ne s'est imposée aux acteurs du marché. A la lecture des définitions actuelles, force est de constater que cette situation prévaut toujours : les points de vue exprimés dépendent du besoin spécifique des industriels ou émanent des offres commerciales des éditeurs, qui utilisent ces définitions comme un positionnement marketing de leurs produits. Une approche *objective* est donc nécessaire pour les mettre en cohérence : les travaux de la littérature scientifique sont présentés dans la section suivante.

Les travaux de la littérature scientifique Parallèlement à son essor d'un point de vue marché, le PLM s'est imposé comme une *discipline* : il constitue aujourd'hui une branche de la connaissance sujette à l'étude de la communauté scientifique. Cette discipline a été récemment consacrée par la création d'un groupe dédié au sein de l'*International Federation for Information Processing*[12]. Le journal de référence de ce groupe de travail, l'*International Journal of Product Lifecycle Management*[13], présente le PLM comme *une approche stratégique de l'entreprise permettant la gestion et l'utilisation effectives du capital intellectuel de l'entreprise.*

La définition donnée par CIMData (présentée plus haut), s'est étonnamment imposée au fil des ans comme une référence importante dans la littérature consacrée au PLM. Reprise par [Sudarsan et al., 2005], elle est toujours considérée comme acceptable par [Bernard et al., 2009].

Certains auteurs ont néanmoins proposé d'autres points de vue. Pour [Lee et al., 2008], le PLM constitue une *approche permettant d'intégrer les hommes, les processus et les systèmes d'information pour gérer le cycle de vie du produit dans sa globalité et dans toutes les entreprises impliquées.* Selon [Ducellier et al., 2007], *l'objectif principal des outils PLM est la gestion du produit et des données qui le décrivent tout au long de son cycle de vie. Le PLM englobe ainsi toutes les étapes de conception du produit (modélisation, gestion*

12. http ://www.ifip-wg51.org/
13. http ://www.inderscience.com/www/IJPLM_leaflet.pdf

de données techniques, simulation, etc.) ainsi que les composants de la chaîne numérique permettant le suivi des informations lors de la fabrication, la mise en vente et le retrait du produit. D'après [Alemanni et al., 2008], la *stratégie PLM est une solution qui met en œuvre plusieurs composants dédiés à la gestion des données relatives au produit.*

Face à l'hétérogénéité des définitions existantes, [Terzi, 2005] propose un point de vue qui s'articule suivant trois axes : stratégique, organisationnel et technique. Ainsi, *le PLM est une approche intégrée qui utilise les technologies de l'information pour permettre la gestion collaborative des données numériques relatives au produit, au cours de toutes les phases de son cycle de vie. Ainsi le PLM implique :*

- *un point de vue stratégique, selon lequel le produit est considéré comme le seul créateur de valeur pour l'entreprise ;*
- *la mise en œuvre d'une approche collaborative permettant la mise en commun de toutes les compétences de l'entreprise, distribuées parmi différents acteurs ;*
- *l'adoption d'un grand nombre de solutions informatiques et d'outils.*

Analyse des définitions Considérant tous les points de vues exposés, nous pouvons constater que le PLM associe :

- une dimension *organisationnelle* (« approche collaborative », « optimiser les processus », « être plus flexible », « optimiser les relations [...] de toutes les organisations impliquées », « son champ couvre [...] les processus » etc.) ;
- un aspect humain (« mise en commun de toutes les compétences de l'entreprise », « utilisation effective du capital intellectuel de l'entreprise », « collaborateurs de l'entreprise » etc.) ;
- l'utilisation des technologies de l'information (« les applications », « systèmes [...] pour », « composants de la chaîne numérique permettant le suivi des informations », « un grand nombre de solutions informatiques et d'outils » etc.).

Bien que ces éléments soient évoqués ou sous-entendus par chacune des définitions, nous proposons dans le paragraphe suivant d'analyser les causes de la diversité des points de vue, ainsi que l'origine des ambigüités qu'elles recèlent.

Analyse de ces définitions de vue au regard de la conception d'ontologies Au cours de son entretien avec Bertil Turesson, [Berkooz, 2007] fait remarquer que la définition donnée par son interlocuteur (et citée plus haut) est très large : elle s'étend bien au-delà de la gestion de l'information et de certaines des activités principales d'une entreprise manufacturière. Nous pensons que, devant une telle étendue fonctionnelle, tenter de définir le PLM par ce qu'il « est », ce qu'il « devrait être » ou ce qu'il « fait » se rapproche de la problématique de la conception d'ontologies (une ontologie étant définie comme *la spécification explicite d'une conceptualisation* [Gruber, 1995]). Nous proposons ainsi d'identifier les critères de conception d'une ontologie définis par cet auteur (clarté, cohérence, extensibilité, déformation d'encodage minimale, engagement ontologique minimal) aux spécifications nécessaires à une définition univoque du PLM :

- **Clarté** : la définition doit faire passer le sens voulu du terme, de manière aussi objective que possible, indépendamment du contexte. De ce point de vue, la définition de CIMData n'est pas claire : qu'est-ce qu'un « ensemble cohérent de pratiques » ? Quel est le sens que donne Terzi à l'expression « approche intégrée » ? D'autre part, la définition doit être complète. De ce point de vue, les définitions ci-dessus sont incomplètes, *i.e.* nulle définition n'inclut le sens des autres dans sa totalité ;
- **Cohérence** : rien qui ne puisse être déduit de la définition ne doit entrer en contradiction avec la définition. De ce point de vue, la notion d' « approche intégrée » peut sembler contradictoire avec « l'adoption d'un grand nombre de solutions informatiques et d'outils » ;
- **Extensibilité** : la définition doit être robuste vis-à-vis de l'émergence de nouveaux concepts (par exemple une innovation technologique). Ainsi le troisième point du raisonnement de Terzi n'est pas robuste vis-à-vis d'un éditeur de logiciels qui proposerait une solution intégrée en remplacement d'un ensemble d'outils hétérogènes (par exemple, l'approche intégrée ERP a dans le passé remplacé l'approche EAI [Bidan, 2004]) ;
- **Déformation d'encodage minimale** : la spécification doit au minimum influer sur la conceptualisation. Par exemple, les définitions du PLM qui insistent sur la technologie informatique ne doivent pas réduire la conceptualisation du PLM à cette seule dimension ;
- **Engagement ontologique minimal** : le but de la définition est de définir un vocabulaire pour décrire le domaine du PLM, si possible de manière complète, ni plus, ni moins. En ce sens, on peut considérer l'engagement ontologique de la définition du journal *IJPLM* comme non-minimal : la notion *d'utilisation effective du capital intellectuel de l'entreprise* fait appel à des connaissances du domaine de la gestion des ressources humaines (par exemple les techniques de motivation des employés, l'adéquation des hommes et des postes etc.), que nous considérons comme hors du périmètre du PLM.

Nous pensons que les cinq points exprimés précédemment constituent une grille de lecture qui permet d'expliquer le fait que les définitions proposées soient nombreuses et ambiguës. Nous proposons donc de contribuer à une définition du PLM qui soit claire, cohérente, extensible et minimale. Cette définition se fera en deux temps : tout d'abord, la section 1.1.2 précise l'objectif du PLM à la lumière de la notion de complexité issue de l'analyse systémique. Dans un second temps (cf. 1.1.3), la couverture fonctionnelle (ou champ du PLM) est structurée suivant une approche opérationnelle/tactique/stratégique qui prend en considération les nombreuses échelles temporelles de cette discipline.

1.1.2. Objectif du PLM

Tenter de définir le PLM par ce qu'il *est* (et aboutir *in fine* à une formulation commençant par « Le PLM est... ») se heurte à la subjectivité du point de vue. De plus, la définition conséquente exprimée à l'instant t n'offre aucune garantie quant à sa viabilité. Nous proposons plutôt d'introduire cette notion par son objectif stratégique, *i.e.* à long terme, formulé de manière à englober les besoins industriels exprimés.

Dans cette section, nous commençons par présenter quelques résultats issus de l'ingénierie système. Nous montrons par la suite que le contexte actuel du développement des produits peut être vu comme un système de systèmes de complexité croissante.

1.1.2.1. Les systèmes d'information

Bien que, selon les définitions précédentes, le PLM ne puisse être réduit à sa seule dimension logicielle, il se fonde grandement sur la notion de système d'information (SI). Comme il sera fait largement appel à ce concept dans ce mémoire, nous définissons dès à présent cette notion. La définition d'un SI peut-être trouvée dans le glossaire[14] du *Committee on National Security Systems* [CNSS, 2006] :

Définition 1. *Un système d'information est un ensemble de ressources organisées pour la collecte, le stockage, le traitement, la maintenance, l'utilisation, le partage, la dissémination, la mise à disposition, l'affichage ou la transmission d'informations. Les ressources sont définies comme étant le personnel, les équipements, les budgets et les technologies de l'information (i.e. les ordinateurs et équipements connexes, les logiciels mais également le service - par exemple la maintenance - et les ressources associées).*

Nous rappelons par ailleurs la distinction entre données et informations faite par [Tsuchiya, 1993] :

Définition 2. *Lorsque l'on donne un sens à une donnée par le biais d'un système interprétatif, elle devient une information.*

1.1.2.2. PLM et complexité

L'analyse des propositions précédentes, et notamment la présence de mots-clés comme « réseaux » ou encore « complexité », nous laissent à penser que le concept de *complexité* est sous-jacent à l'ensemble des points de vue exprimés bien qu'il n'apparaisse pas explicitement dans les travaux de la littérature scientifique.

Dans un contexte de sciences humaines, [Fourastié, 1966] apporte un éclairage pour appréhender les difficultés inhérentes aux phénomènes et systèmes complexes : « *La science a fui la réalité synthétique et complexe, parce qu'elle n'y trouvait pas les simples déterminismes qu'elle y cherchait. Elle a démembré ces réalités en éléments, chacun fécond du point de vue de la recherche. Mais la juxtaposition des sciences élémentaires ainsi constituées, s'est révélée inapte à former la science du phénomène complexe : un être organisé, un ensemble, une société, sont autre chose que leurs constituants, de même*

14. Également disponible à l'adresse :
http ://www.fismapedia.org/index.php ?title=Information_technology

qu'une molécule est autre chose qu'une brochette d'atomes... ainsi nous ignorons encore à peu près tout des ensembles organisés, vivants ou non [...] ».

On a parfois tendance à qualifier les systèmes de très grande taille de *complexes* alors qu'ils ne sont souvent que *compliqués*. Par exemple, ce n'est pas le volume d'une base de données qui la rend complexe, mais sa structure, les interrelations dynamiques entre ses objets, la gestion temporelle et la réactivité aux événements, le maintien de l'intégrité des données quelles que soient les transactions et les pannes matérielles ou logicielles naturellement toujours imprévisibles, mais dont les risques et les réactions doivent avoir été évalués.

Ces éléments sont à combiner avec la distribution et l'autonomie tant en matière de données que de processus. Dans le cas du SI d'une multinationale, basé sur Internet, et le web ou dans le cas du SI d'un opérateur de télécommunications, ce n'est pas le nombre d'instances ou le nombre d'événements qui induisent la complexité, mais c'est la diversité des éléments ainsi que l'hétérogénéité des architectures et de leurs entrelacements. Dans un tel système, la diversité s'entend en termes de machines, réseaux, postes de travail des utilisateurs, systèmes, intergiciels, architectures, bases de données, bases d'ontologies, applications partagées et distribuées, applications locales, etc.

Une compréhension du compliqué et du complexe est apportée par [Le Moigne, 1990] qui, dans son approche systémique, précise que l'intelligibilité du compliqué se fait par simplification alors que l'intelligibilité du complexe peut se construire par modélisation. En d'autres termes, le compliqué est lié à une accumulation d'éléments qui conduit à une accumulation de difficultés alors que la nature complexe d'un système est davantage inhérente aux combinaisons et coordinations nombreuses et variées entre éléments qu'aux éléments eux-mêmes. Une simplification hasardeuse ou la mutilation d'un système complexe peut détruire *a priori* son intelligibilité [Le Moigne, 1990]. Ainsi, un modèle de SI et la modélisation d'un SI sont tous deux complexes par essence : un SI est généralement constitué d'un grand nombre d'éléments (composants) pour la plupart fortement interconnectés.

[Simon, 1962] définit un système complexe comme *un système constitué d'un grand nombre de composants qui interagissent d'une manière non-triviale. Dans de tels systèmes, compte tenu des propriétés des composants et des lois qui gouvernent leurs interactions, il est difficile d'en déduire les propriétés du système global.* D'après [Ethiraj and Levinthal, 2004], une conséquence de la complexité est que les performances du système global peuvent révéler un système au comportement non-linéaire en réponse à un changement de l'un ou plusieurs des composants du système.

Nous proposons d'établir un parallèle entre l'ensemble des définitions du PLM présentées dans la section 1.1.1 et les considérations précédentes relatives à la complexité. Le raisonnement s'articule en deux points :

- montrer que l'organisation associée au développement de produits est un système complexe ;

- montrer que la complexité de ce système de systèmes tend à augmenter.

Le développement de produits industriels : un système complexe *Les méthodes normatives de l'ingénierie système peuvent être utilisées pour modéliser le produit, les systèmes de production et les processus de développement du produit* [Messaadia, 2008]. Cet auteur argue de plus que *l'augmentation du coût de correction des erreurs au cours du développement du système et la volatilité des exigences sont parmi les problèmes qui obligent à travailler dans le contexte de l'ingénierie système.* Pour [Morel et al., 2007], *l'essence de l'intégration des entreprises est l'interopération récursive de systèmes pour composer un système capable de réaliser un objectif dans un contexte spécifique.* La non-trivialité (voir la définition de [Simon, 1962]) des interactions dans cet ensemble d'acteurs est évidente.

L'accroissement de la complexité du « système de développement de produits » D'après [Briggs et al., 2005], dans une étude concernant le domaine de la construction navale, *l'accroissement de la complexité des systèmes et des collaborations induit le besoin, pour les acteurs impliqués, de partager l'information à travers les frontières des systèmes et des organisations.*

Plusieurs facteurs contribuent à augmenter la complexité du système de systèmes constitué :

- les contraintes fortes liées à une rude concurrence, aux objectifs de réduction des délais de mise sur le marché, à la recherche d'un *optimum* coûts/qualité/délais ont tendance à accélérer et renforcer les interactions entre acteurs de l'entreprise ;
- l'accroissement de la complexité du produit : la tendance des produits à intégrer des technologies différentes et complémentaires (par exemple les produits mécatroniques) implique une plus forte interdépendance des équipes et des systèmes d'information ;
- l'accroissement de la complexité des organisations : la dynamique fortement concurrentielle du marché augmente le recours à la sous-traitance, au travail en entreprise étendue, à des partenariats stratégiques inter-entreprise (alliance Renault/Nissan par exemple), à des fusions/acquisitions/séparations fréquentes. Ces nouvelles interactions s'étendent à l'échelle mondiale ;
- enfin, l'adoption massive d'outils numériques pour soutenir le développement des produits participe de cet accroissement de la complexité. En effet, *les technologies de l'information permettent de composer avec une complexité croissante de l'environnement et des technologies et, ce faisant, elles participent à son développement* [Galliers et al., 1997].

Nous proposons dans la section suivante de définir le PLM en s'appuyant sur cette notion de complexité.

1.1.2.3. L'objectif du PLM

Pour faire la synthèse des quelques points vus précédemment, nous proposons les deux définitions suivantes sur lesquelles nous nous baserons dans la suite de ce mémoire.

Définition 3. L'objectif du Product Lifecycle Management est la maîtrise de la complexité qui caractérise le développement et le suivi des produits. Cet objectif s'inscrit dans une stratégie d'entreprise visant à réduire les coûts, les délais et à augmenter la qualité des produits.

Définition 4. On appellera « stratégie PLM » (ou « démarche PLM ») l'ensemble constitué de l'objectif précédent, des moyens alloués pour le réaliser (humains, financiers, techniques, organisationnels) et de la méthodologie permettant de les mettre en œuvre de manière efficiente.

Ainsi, bien que le besoin initial du PLM fût de permettre le développement simultané et collaboratif des produits en prenant en compte toutes les phases de leur cycle de vie, nous pensons qu'il conviendrait mieux aujourd'hui d'évoquer cette discipline comme celle de la *gestion de la complexité*. Nous détaillons dans la section suivante comment cette gestion peut être abordée suivant des échelles de temps différentes.

1.1.3. Le PLM vu selon un angle stratégique/tactique/opérationnel

L'objectif stratégique de la définition 1.1.2.3 peut être vu suivant de nombreux sous-objectifs dont l'étude constitue une partie du champs disciplinaire du PLM. Dans cette section, nous mettons en évidence le fait que ces sous-objectifs visent des échelles de temps différentes, et structurons suivant ce point de vue l'ensemble des concepts habituellement recensés dans le domaine du PLM. Les échelles de temps en question sont classées en trois catégories : court, moyen et long terme.

- stratégique : l'ensemble des activités dont l'échéance est à long terme. On peut citer par exemple l'archivage des données qui décrivent le produit, la gestion de la traçabilité des décisions qui jalonnent le développement du produit ou le produit en lui-même etc. ;
- tactique : l'ensemble des activités dont l'échéance se situe à moyen terme. Par exemple les processus liés à un projet particulier, l'analyse de leur robustesse, l'interconnexion de systèmes d'information de nouveaux partenaires etc. ;
- opérationnel : l'ensemble des activités qui s'inscrivent à court ou très court terme. Par exemple la planification des tâches quotidiennes des concepteurs, le pilotage d'atelier, la gestion des demandes de modification de conception (*Engineering Change Request* - ECR) ou des ordres de modification de conception (*Engineering Change Order* - ECO), la génération d'un dessin 2D à partir d'un modèle 3D, les processus de notifications,

l'automatisation des tâches répétitives (par exemple les templates Catia V5), la mise en œuvre de simulations numériques permettant de caractériser le comportement du produit virtuel etc.

Dans la section suivante, nous proposons par conséquent de situer les problématiques d'interopérabilité par rapport à l'objectif stratégique du PLM, mais également suivant l'**approche stratégique/tactique/opérationnelle** précédemment présentée.

1.1.4. Interopérabilité et PLM

Dans ce contexte de complexité, **l'interopérabilité dans le domaine du PLM vise à maîtriser les interactions entre les composants du système de systèmes PLM**. De la même manière que le PLM peut être vu suivant des échelles de temps différentes, il conviendra, dans cette recherche, d'étudier l'interopérabilité selon la dimension temporelle qu'elle adresse : **nous parlerons donc, dans ce cadre, d'interopérabilité stratégique, tactique et opérationnelle**.

Dans la suite de ce mémoire, nous focalisons notre étude sur deux phases bien particulières du cycle de vie du produit : sa conception et sa fabrication. Cette restriction du domaine d'étude permet d'aboutir à des résultats de granularité fine qui seraient impossibles à obtenir, dans le cadre d'un travail de thèse, à l'échelle du cycle de vie du produit tout entier. Le choix de la conception et de la production permet en outre de faire apparaître des nuances sémantiques qui ont un impact très fort quant aux difficultés d'interopérabilité et aux moyens à mettre en œuvre pour les résoudre. Les résultats que nous présentons seront toutefois généralisables à des problèmes d'interopérabilité plus larges.

1.2. Cas particulier de la conception et de la production dans le développement de produits

1.2.1. Les métiers de la conception et de la production

Le terme de *métier* est régulièrement utilisé dans le domaine industriel, souvent d'ailleurs pour désigner l'utilisateur d'une solution informatique, en opposition aux personnes chargées de la déployer : on parlera alors d'*hommes du métier* ou de *besoins métier*. Dans la suite de ce mémoire, nous définirons cette notion de la manière suivante :

Définition 5. Le *métier* regroupe l'ensemble des savoirs et savoir-faire mobilisés par un opérateur pour mener à bien les activités relatives à un domaine identifié.

Par restriction aux domaines de la conception des produits et de leur fabrication, nous proposons les deux définitions suivantes :

Définition 6. Le *métier de la conception* regroupe l'ensemble des savoirs et savoir-faire mobilisés lors d'activités liées à la conception des produits industriels.

Définition 7. Le *métier de la production* regroupe l'ensemble des savoirs et savoir-faire mobilisés lors d'activités liées à la fabrication des produits industriels.

1.2.1.1. Le métier de la conception - Définition du produit virtuel

Dans le contexte des entreprises industrielles, la conception joue un rôle déterminant dans la stratégie de différentiation par rapport à la concurrence [Ulrich and Eppinger, 1995]. Il s'agit de concevoir une offre innovante qui satisfait les besoins des clients, tout en respectant les contraintes du triptyque coût/qualité/délai.

L'activité de conception [Koike, 2005] dresse un état de l'art exhaustif concernant l'*activité* de conception. Parmi les différentes références présentées par cet auteur concernant la modélisation de l'activité de conception, le modèle de Pahl et Beitz [Pahl et al., 1966] sera présenté et utilisé dans notre travail : les autres modèles de la littérature se distinguent de celui-ci sur des points qui n'ont pas d'influence sur le discours qui suivra.

[Pahl et al., 1966] définissent la conception comme une activité d'ingénierie dans laquelle le concepteur mobilise ses connaissances scientifiques et son expérience afin d'apporter des solutions au problème préalablement défini pour la réalisation de l'artefact : il s'agit d'une activité de résolution de problème. L'activité de l'homme du métier (le concepteur) est de trouver, dans l'espace de solutions possibles, la meilleure solution satisfaisant aux contraintes de l'étude (cahier des charges fonctionnel, exigences techniques, contraintes budgétaires, conditions de fabrication du produit etc.). [Pahl et al., 1966] définissent un processus de conception basé sur une approche systématique constituée de quatre phases :

- définition du problème : cette étape s'intéresse à la définition du contexte économique, l'expression et la validation du besoin client, la traduction de ce besoin en termes de spécifications fonctionnelles synthétisées dans un cahier des charges fonctionnel ;
- innovation/recherche de solutions possibles (*conceptual design*) : après avoir mené une étude d'analyse fonctionnelle interne (conduisant à l'architecture fonctionnelle du produit), cette phase s'attache à ouvrir le plus largement possible l'espace des solutions possibles par la recherche de principes constructifs *a priori* envisageables (séances de créativité, brainstorming, recherche de principes innovants etc.). Une analyse de faisabilité permet de choisir la meilleure solution au regard des critères coût/qualité/délai ;

- conception préliminaire (*embodiment design*) : les principes constructifs choisis permettent de définir l'architecture technique du produit. Cette architecture permet d'initier le travail collaboratif ;
- conception détaillée (*detail design*) : cette phase a pour objectif de définir complètement le produit et vérifier qu'il satisfait aux spécifications fonctionnelles.

Plusieurs allers et retours entre ces phases peuvent être nécessaires avant d'arriver à une solution acceptable : on parle alors de *boucle de conception*. Les principales décisions de conception sont prises durant les phases dites *amont* (*i.e.* avant la phase de conception détaillée) : elles ont un impact fort sur l'ensemble des autres phases du cycle de vie du produit [Verganti, 1999]. Ainsi, d'après [Asiedu and Gu, 1998] et [Barton et al., 2001], près de 80% du coût d'un produit est engagé par des décisions prises lors de ces phases amont. Les dynamiques des activités et des prises de décision de conception font que plus le projet de développement de produit avance, moins on dispose de latitude pour modifier les décisions en créant ainsi des irréversibilités [Midler, 1993]. Par conséquent, il existe un intérêt stratégique à anticiper au plus tôt l'impact des décisions de conception.

Organisation de la conception En ce qui concerne l'agencement des ressources humaines et matérielles, plusieurs structures possibles sont identifiées. Dans une approche de structure projet, [Wheelwright and Clark, 1992] définissent quatre organisations :

- fonctionnelle par métier : caractérisée par la spécialisation horizontale, avec peu d'interactions transversales, où chaque métier est rattaché directement à sa hiérarchie ;
- en coordination locale : caractérisée par l'existence d'un chef local de projet qui fait la liaison de son métier avec la hiérarchie ;
- en coordination transversale : caractérisée par l'existence d'un chef de projet transversal à toutes les équipes métier et qui fait la liaison avec la direction de projets ;
- structure autonome de projets : caractérisée par l'organisation matricielle par projet (donc multi-métier), avec l'existence d'une coordination transversale.

Appuyé sur l'exemple de Renault, [Midler, 1993] présente la structure en plateaux-projets, où toutes les ressources sont mises à disposition par l'entreprise selon une logique de projet, dans un même emplacement et avec l'objectif commun de concevoir un nouveau véhicule. Les deux dimensions organisationnelles (des activités de conception et des ressources) ne sont pas indépendantes l'une de l'autre. Par exemple, les approches organisationnelles appuyées sur le concept d'*Ingénierie Concourante ou Simultanée* (Concurrent Engineering or Simultaneous Engineering) intègrent ces deux dimensions dans le but de permettre la conception simultanée de l'artefact et de son processus de production, en opposition à l'approche classique, considérée comme peu performante, de l'exécution séquentielle de tâches [Sohlenius, 1992]. Dans l'approche séquentielle, la connaissance sur la conception est créée séquentiellement. Les activités sont réalisées selon un flux informationnel qui passe d'une fonction métier à l'autre. Par exemple, on commence à penser la fabrication après avoir défini le produit à fabriquer. En revanche, dans l'approche concourante le processus de conception est réalisée par des équipes multifonctionnelles qui sont organisées

de manière à faciliter les échanges le plus tôt possible entre les différents métiers qui participent aux efforts du projet. L'un des objectifs est de concevoir à la fois le produit et son processus productif, ce qui permet de réduire le temps global du projet et le risque de remise en cause des choix initiaux.

En s'appuyant sur les travaux de [Le Moigne, 1990], [Adreit and Mauran, 1995] modélisent la conception selon une approche systémique où le point d'entrée d'analyse est l'ensemble complexe des interactions qui s'y développent. Dans leur modèle, les auteurs proposent trois acteurs en interaction : le phénomène (par exemple l'activité de conception), le concepteur et l'environnement organisationnel. Leur objet d'étude est donc l'interaction à partir de ces trois composants fondamentaux.

Conception et connaissance Ces dernières années ont vu l'émergence de travaux relatifs à la *capitalisation* et la *diffusion* de la *connaissance* avec pour objectif de s'assurer que *les connaissances sont bien utilisées dans l'élaboration du produit final* [Louis-Sidney et al., 2009]. [Tsuchiya, 1993] définit la *connaissance* comme une *information comprise par un système interprétatif.* [Grunstein, 2002] propose un modèle en cinq points pour capitaliser les connaissances :

- repérer les connaissances ;
- préserver les connaissances ;
- valoriser les connaissances ;
- actualiser les connaissances ;
- gérer les interactions entre les quatre points précédents.

Ainsi le processus de conception visant à définir un produit virtuel est-il fortement lié aux autres acteurs de l'entreprise détenant un savoir et un savoir-faire capitalisés : au cours de ses activités, chacun génère de la connaissance qu'il est nécessaire de mettre au service des acteurs de la conception pour travailler dans une démarche efficiente de la *gestion du cycle de vie des produits.*

Conclusion La conception (c'est-à-dire l'ensemble des ressources et des méthodes dédiées aux activités de conception) vise à définir un produit virtuel conforme aux spécifications fonctionnelles issues de l'analyse du besoin client. L'organisation et la synchronisation des processus de conception tendent à être collaboratifs. Nous synthétisons sur l'actigramme de la figure 1.2.1 la fonction conception comme une activité de transformation de ressources visant à créer un produit virtuel :

- le(s) produit(s) virtuel(s) (la matière d'œuvre sortante) sont le résultat de la transformation du besoin client (matière d'œuvre entrante) ;
- un flux d'informations est généré par la fonction conception ;
- les connaissances de conception permettent de piloter de manière optimale la fonction conception ;
- l'autre sortie de cette fonction est la connaissance de conception : acquise en cours de projet et capitalisée, cette connaissance constituera le trésor de l'entreprise, et pourra valoriser d'autres projets de conception futurs ;

- sur la figure 1.2.1, la roue d'entrée grisée représente la synchronisation des flux de données et de connaissances.

Selon le modèle *Structured Analysis and Design Technic* (SADT), la *valeur ajoutée* est égale à la différence entre la matière d'œuvre sortante et la matière d'œuvre entrante. La valeur ajoutée de la fonction conception est donc la différence entre l'ensemble (produit virtuel + connaissance de conception générée) et le besoin client.

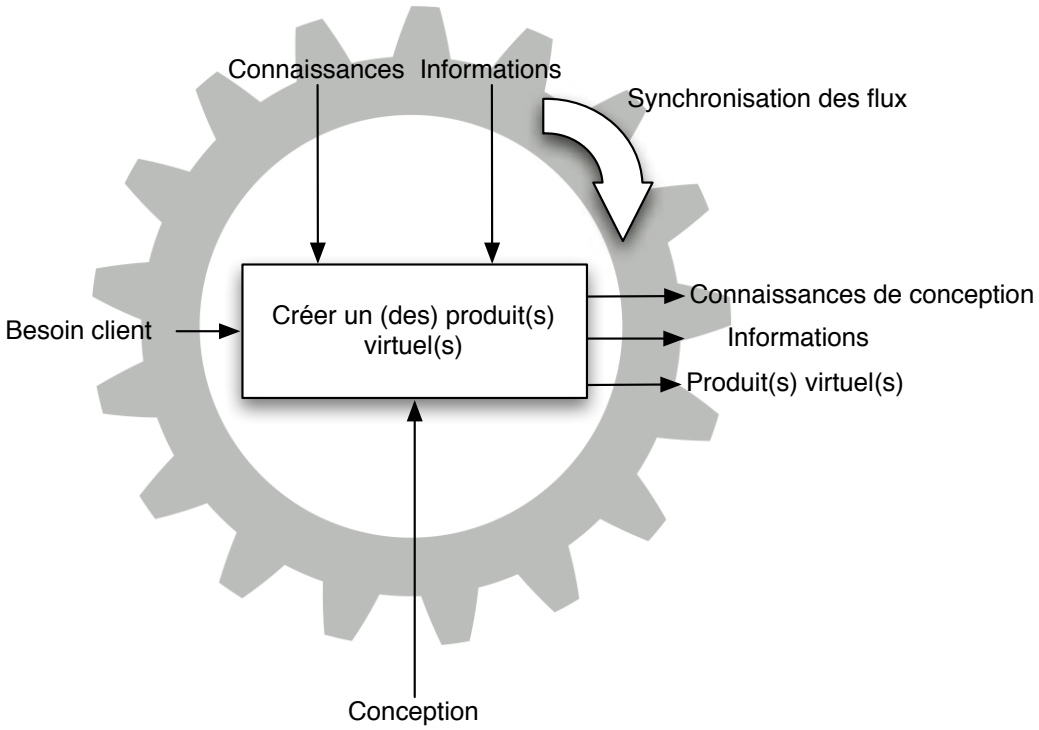


FIGURE 1.2.1.: Actigramme de la fonction conception

La section suivante présente de la même manière synthétique ce qu'on entend par le terme « production ».

1.2.1.2. Le métier de la production - Réalisation du produit physique

La production consiste à transformer des ressources en produits afin de les mettre à disposition d'un client. Un produit est un objet matériel dont la conception et la transformation sont issues d'un savoir-faire propre à chaque entreprise. L'élaboration des produits est le fruit de la consommation de ressources. Elles correspondent à l'ensemble des potentialités dont dispose l'entreprise pour réaliser ses produits. Tout produit a comme destinataire un client. La charge confiée à la gestion industrielle est d'assurer

la planification et le pilotage des ressources, tant au niveau opérationnel qu'au niveau tactique et stratégique (voir figure 1.2.2).

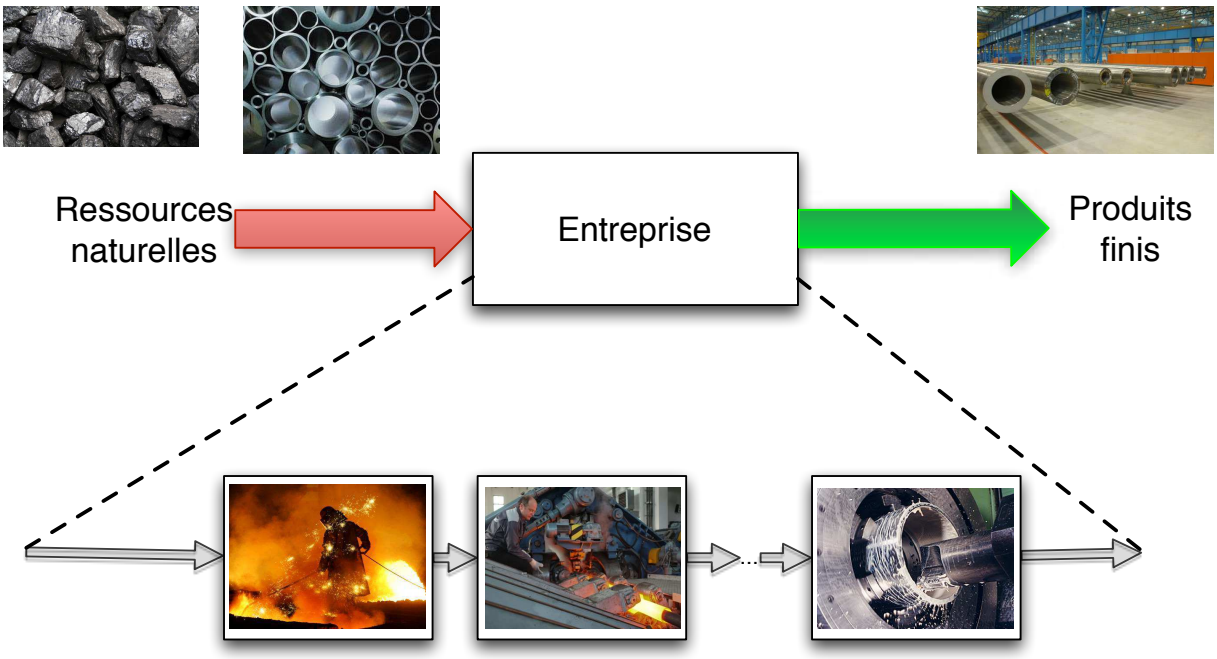


FIGURE 1.2.2.: Le rôle de la gestion industrielle

Le gestionnaire de production recueille les données concernant la stratégie générale de l'entreprise, ses sources d'approvisionnement, ses prévisions commerciales, etc. (informations relatives à l'environnement du système de production). Par ailleurs, il se préoccupe des niveaux d'en-cours, du taux de pannes des machines, des quantités produites, etc. (informations relatives à l'état du système de production). Ces informations sont, pour une large part, saisies par et pour le gestionnaire. Elles résultent d'un premier traitement de l'information.

Sur cette assise informationnelle, il doit prendre des décisions de pilotage. Au niveau opérationnel, il s'agit de régir le flux (déclenchement des ordres de fabrication, gestion des stocks, etc.) à l'aide de règles et de procédures. Plus précisément, il s'agit de répondre aux questions suivantes : que produire ou acheter (produit ou matière), pour quand, combien produire ou acheter (quantité) et enfin peut-on le faire (temps, coût, main-d'œuvre) ? Pour le niveau tactique et stratégique de plus long terme, l'objectif est d'améliorer le système de production et de faire progresser l'entreprise.

D'après [Plossl, 1993], la *maîtrise de la production signifie l'aptitude à établir des plans viables et à les mener à bien. Ceci quelle que soit la taille de l'entreprise, quel qu'en soit le nombre d'employés, le degré d'informatisation, le type de production etc.*

Il est cependant évident que ces facteurs sont interdépendants et que, si un tableau géré manuellement permet d'établir des plans viables dans une petite structure, un système informatique de type MRP (Manufacturing Resources Planning) sera nécessaire dans une grande société. La plus grande difficulté réside alors dans la mise en œuvre.

La maîtrise de la production s'appuie sur deux flux majeurs : le flux matière et le flux d'informations. Il s'agira de les synchroniser :

- le flux matière est visible, lent et onéreux. Visible du magasin matière à la zone d'expédition, le flux des pièces a toujours été à la base de la gestion de la production. Lent parce que sa mobilité dépend non seulement des temps de transformation, mais aussi des temps de transfert et d'attente. Les temps de transformation peuvent ne représenter que 5 à 20 % des temps de transit globaux, dans les entreprises manufacturières (ce résultat issu d'une enquête du CETIM [Crouhy and Greif, 1991], repris dans de nombreux ouvrages, reste encore, plus de vingt ans après, toujours d'actualité dans les entreprises qui n'ont pas parfaitement intégré le concept de production en juste-à-temps). Enfin, le flux matière est onéreux. Aujourd'hui, il est crucial d'en mesurer le coût ;
- le flux d'informations, lui, est invisible, rapide et peut être onéreux. Les informations peuvent être verbales, informatiques, enregistrées, écrites etc. Elles sont dans tous les cas beaucoup plus souvent invisibles. Exceptions faites de celles que l'on trouve dans les entreprises qui depuis quelques années ont choisi un *management visuel* [Greif, 1998]. Le flux d'informations est rapide, il l'est d'autant plus que nous comparons sa vitesse à celle du flux matières. Et enfin, le flux d'informations peut être très peu cher. Les coûts relatifs à ce flux sont liés à la gestion informatique essentiellement. Mais toute l'information et toute la communication ne nécessitent pas évidement l'utilisation de systèmes informatiques.

Construire des plans viables, avec ou sans informatique, est un objectif impératif pour maîtriser la production. A travers cette notion de planification, il apparaît donc que le temps est un facteur clé : *le temps est la plus précieuse des ressources* [Plossl, 1993]. Quelle que soit la manière de faire, le système s'attache toujours à planifier d'une manière optimale l'horizon de gestion, d'une part, et à vérifier, à l'usage, que ce qui a été réalisé correspond bien à ce qui a été prévu, d'autre part .

Dans le système de production, les gains sont directement liés à la vitesse des flux matières et des flux d'informations. Cependant, cette logique duale induit naturellement une scission entre les faits et les informations qui sont relatives à ces faits. C'est bien en cela que réside une des difficultés de la gestion industrielle.

Le gestionnaire cherchera donc en permanence à synchroniser les flux matière et d'informations ainsi qu'à rendre cohérents les événements (faits et propositions) vécus et réalisés dans les ateliers et tout au long de la chaîne logistique. Il est donc fondamental pour les entreprises de chercher à minimiser cette scission, à rapprocher les faits du quotidien, des idées et des informations qui leur sont inhérentes.

Il apparaît donc que, pour pouvoir maîtriser la production, il est nécessaire de posséder des informations récentes sur les flux, de s'appuyer sur des données issues de la production même et de l'ensemble des systèmes environnants. Cependant, ces données ne peuvent avoir de l'intérêt que dans la mesure où elles sont fiables. La difficulté est alors de deux ordres : la fiabilité de celles-ci et leur association aux faits qui les ont générées. Les hommes sont donc une composante essentielle : les opérateurs sont à l'origine des faits, ce sont eux qui donnent les informations relatives au vécu et aux faits, ce sont encore eux qui reçoivent les données issues de la gestion. Il est ainsi nécessaire d'avoir un système de pilotage et un système de retour d'informations afin de pouvoir réaliser les ajustements [Thomas, 1994].

C'est dans les systèmes de retour d'informations que les faiblesses sont les plus marquées aujourd'hui. Et quand bien même ils fonctionnent normalement, les informations qu'ils véhiculent sont totalement séparées de l'analyse du quotidien et des faits qui s'y rapportent.

On peut aujourd'hui être submergé d'informations qui ont perdu toute signification, et par là même beaucoup d'intérêt. Toutes ces données ne prennent de sens que dans la mesure où les différents éléments s'y référant sont pris en compte, où les éventuels problèmes survenus font l'objet d'une réflexion pour leur résolution et entraînent un certain nombre de modifications qui feront évoluer le système d'une manière stable.

Le concept actuel de *Lean Manufacturing* [Womack et al., 1990] conduit les entreprises à mettre en œuvre l'ensemble des actions qui permettent de coordonner et optimiser toutes les activités. Ces entreprises se dotent alors d'une organisation logistique globale performante ou *Supply Chain Management* (SCM).

Conclusion Nous synthétisons sur la figure 1.2.3 la fonction production sous forme d'actigramme :

- le(s) produit(s) physique(s) sont créés à partir de la transformation de ressources ;
- un flux d'informations est généré par la fonction production ;
- cette fonction est pilotée par deux éléments : la connaissance des gens du métier ainsi que la définition du produit virtuel (représenté sous forme d'information numérique ou papier) ;
- l'autre sortie de cette fonction est de la connaissance de production : conséquence de l'expérience, cette connaissance acquise et capitalisée permet au hommes (et aux systèmes d'informations) de maîtriser les processus de production et de travailler dans le sens de l'amélioration continue ;
- la roue dentée grisée représente la synchronisation des flux physiques et d'informations.

De manière analogue à la section précédente, la valeur ajoutée de la fonction production peut être vue comme la différence entre l'ensemble (produit physique + connaissance de production) et les ressources physiques nécessaires.

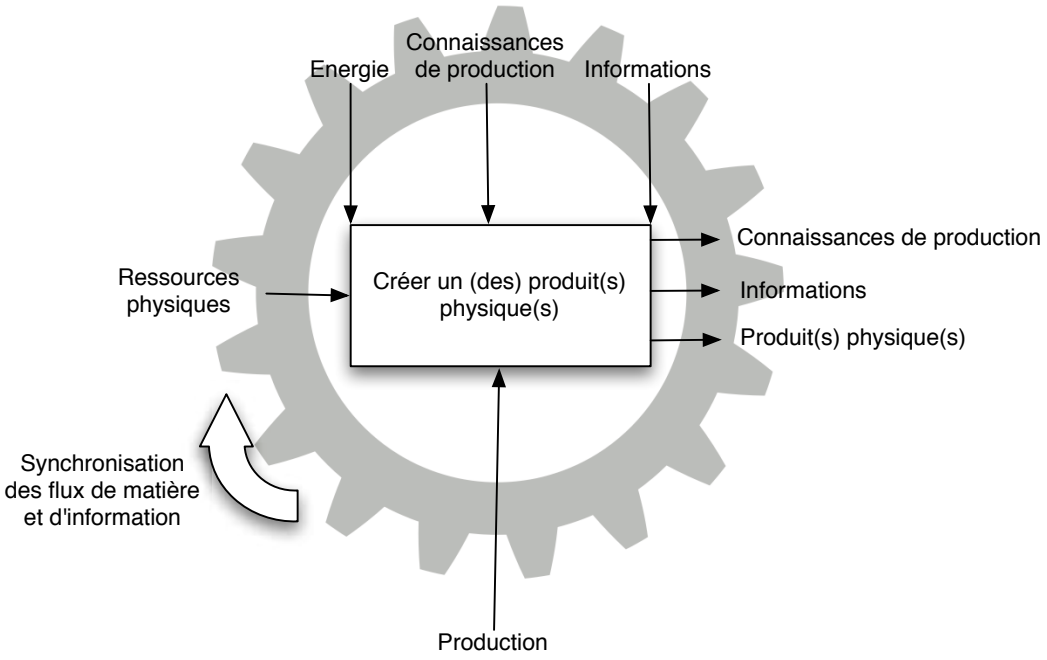


FIGURE 1.2.3.: Actigramme de la fonction production

1.2.1.3. L'interface conception/production

Les deux sections précédentes illustrent un parallèle entre la conception et la production, qui sont toutes deux des activités visant à transformer des *ressources* en *produits :*

- durant les phases de la conception, un ou plusieurs produits virtuels sont générés à partir de la connaissance présente dans l'entreprise : il est remarquable de constater que, bien que les équipes de conception ne puissent être considérées comme spécialistes de chacun des métiers concernés par les phases du cycle de vie du produit, elles doivent néanmoins être capable de digérer l'ensemble des connaissances métier de l'entreprise ;
- durant les activités de production, le produit physique est créé à partir de ressources physiques (matières premières, énergie, composants achetés) et du produit virtuel provenant de la conception. Le métier de la production génère des connaissances qui doivent être portées à l'attention des équipes de conception.

Dans un contexte PLM, l'interface conception/production se retrouve à plusieurs niveaux (voir figure 1.2.4) :

- la *synchronisation* des activités doit être assurée. Cet aspect est représenté sur la figure par l'engrènement des deux roues dentées présentes sur les figures 1.2.1 et 1.2.3 ;
- des échanges de *connaissance* : les connaissances de la fonction production doivent nourrir la fonction conception afin de s'assurer, dès les premières phases de la conception, que les choix techniques envisagés sont pertinents d'un point de vue de la gestion de la production ;
- des échanges d'*informations*, qu'ils s'opèrent ou non sous forme numérique : la fonction conception alimente la fonction production d'un ensemble de données techniques

permettant la réalisation physique du produit, ainsi que son contrôle. Dans le même temps, des informations de la fonction production peuvent contraindre *a posteriori* la fonction conception : résultats d'études d'industrialisation négatifs, demandes de modification de conception, ordre de modification de conception etc.

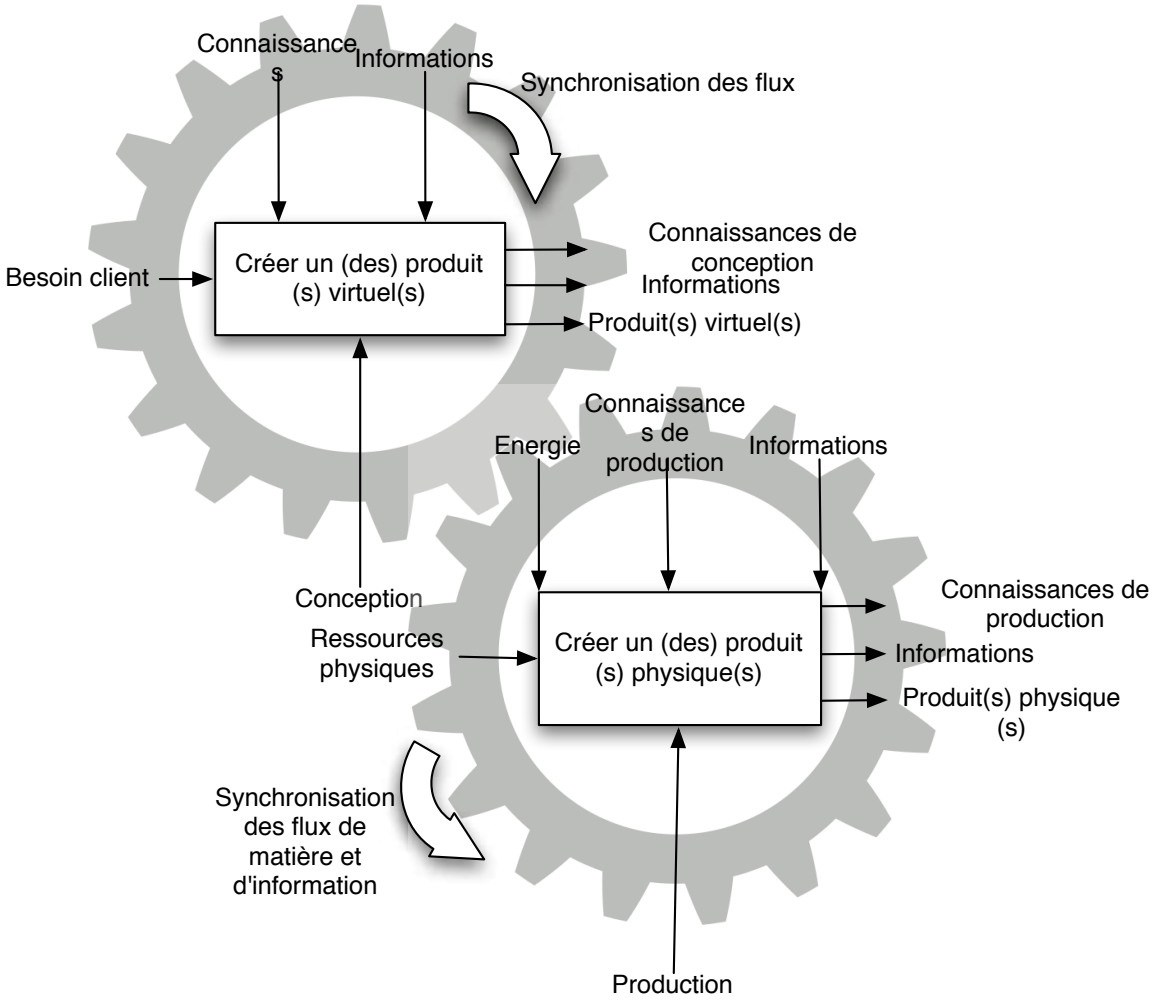


FIGURE 1.2.4.: Interface conception/production

Les activités de conception et de production sont jalonnées par un ensemble de *décisions* :

- du point de vue de la conception, les décisions concernent le choix et le dimensionnement des solutions techniques pour aboutir à un produit conforme aux spécifications fonctionnelles issues de l'analyse du *besoin* du client ;
- du point de vue de la production, les décisions concernent l'organisation de la production pour être en mesure de fabriquer le produit conformément aux prescriptions techniques de la conception, mais aussi aux *besoins du client en termes de délais de livraison.*

Ces décisions sont prises dans un contexte global d'optimisation des coûts du produit. Ainsi, bien que séparés d'un point de vue fonctionnel, les départements conception et

production se trouvent mutuellement engagés par leurs décisions, qui peuvent avoir sur la collaboration un impact à l'échelle stratégique/tactique/opérationnelle :

- **opérationnel** : la modification mineure d'un intervalle de tolérance en conception (par exemple passer de H8 à H7) peut nécessiter de modifier les opérations sur une machine, par exemple une passe supplémentaire dans une opération d'usinage. Dans ce cas, l'impact est très local en production (uniquement le fichier gamme) mais peut modifier le temps de cycle pour la fabrication de la pièce. Ce temps de cycle modifié, même de manière mineure, pourra conduire à l'apparition de goulets d'étranglement sur la ligne de production ;
- **tactique** : c'est la nomenclature de production qui pilote le calcul en besoins nets. Cette nomenclature de production est directement issue de la nomenclature de conception et donc des choix de conception : une modification de la structure produit en conception pourra donc nécessiter un travail de mise à jour de la nomenclature de fabrication ;
- **stratégique** : fabriquer ou acheter ? La décision de fabriquer un composant, en utilisant les ressources industrielles de l'entreprise, ou bien d'acheter le composant à un fournisseur a des conséquences importantes en termes d'organisation et de coûts, aussi bien aux études qu'en production. C'est un élément stratégique qui peut avoir pour conséquence de nécessiter l'investissement dans un nouvel équipement industriel (une ligne de production additionnelle, voire une usine supplémentaire), la recherche d'un nouveau fournisseur, la mise en place d'une nouvelle chaîne logistique pour l'approvisionnement de ce composant acheté. La direction de l'entreprise doit être en mesure de savoir quelle est la meilleure solution en termes de coûts.

On mesure bien la complexité de cet ensemble (conception/production) et donc, par conséquent, la difficulté des prises de décision qui peuvent avoir une dimension stratégique. Si l'on se réfère à nos définitions 3 et 4, nous concluons que **seule une stratégie PLM peut proposer à la direction de l'entreprise les indicateurs et outils d'aide à la décision dont ils ont besoin**.

Les deux sections précédentes, après avoir défini chacun des métiers de la conception et de la production, ont mis en évidence le fait que dans le contexte actuel ces deux métiers doivent pouvoir échanger des informations au cours du cycle de développement du produit. Ces informations ont pour but d'accélérer le développement du produit et réduire les risques d'erreur en les anticipant au plus tôt. Nous illustrons ce point de vue sur la figure 1.2.5 :

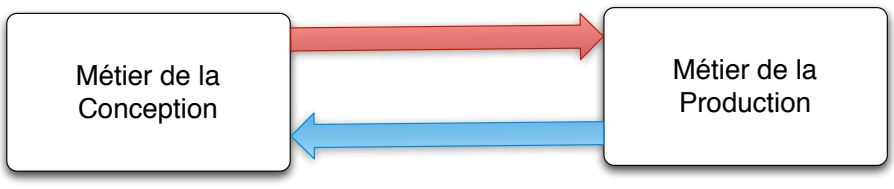


FIGURE 1.2.5.: Besoin d'échange d'informations conception/production durant le processus de développement du produit

Dans la section suivante, nous mettons en évidence les phénomènes pouvant perturber ces échanges, en particulier ceux liés à des problèmes sémantiques.

1.2.2. Les ambiguïtés sémantiques comme obstacle à la communication conception/production

Dans le contexte de complexité déjà décrit, le partage efficient des informations entre acteurs suppose une communication sémantiquement parfaite : les informations véhiculées doivent conserver leur sens quels que soient les moyens utilisés pour les transférer. Cette communication peut néanmoins être altérée par des ambiguïtés dans la perception du sens des concepts échangés.

1.2.2.1. Quelques définitions relatives à la sémantique

Dans un contexte linguistique, [Saussure, 1916] définit les éléments suivants :

Définition 8. *La sémantique est une branche de la linguistique qui étudie les signifiés. Le signifié désigne le concept, c'est-à-dire la représentation mentale d'une chose.*

L'auteur distingue dans cette définition *signification* et *valeur* : « mouton » et « sheep » (sa traduction anglaise) ont le même sens, mais non la même valeur, puisque l'anglais pour sa part distingue l'animal (*sheep*) de sa viande (*mutton*).

Définition 9. *Le signifiant désigne l'image acoustique d'un mot.*

Définition 10. *Le domaine sémantique est l'activité ou la discipline dans laquelle un mot est utilisé avec un sens donné.* Par exemple, le mot « souris » évoque un sens dans le domaine « informatique » et un autre dans le domaine « zoologie ». En d'autres termes, le signifiant « souris » correspond à deux signifiés différents dans les deux domaines sémantiques que sont l' « informatique » et la « zoologie ». Si des signifiants différents renvoient au même signifié, on parlera de *synonymie* (« cuillère » et « leurre métallique » sont des synonymes dans le domaine sémantique « pêche au carnivore »). Si un même signifiant correspond à plusieurs signifiés, on parlera d'*homonymie* (« cuillère » dans le domaine sémantique « repas » et « cuillère » dans le domaine sémantique « pêche au carnivore » sont des homonymes).

Distance sémantique Les travaux relatifs à l'extraction d'informations provenant d'Internet utilisent la notion de *distance sémantique*. Par exemple, [Vitanyi, 2005] définit une métrique sur un espace sémantique permettant de quantifier dans quelle mesure les sens de différents termes ou expressions peuvent être rapprochés. Dans notre travail de

recherche, nous n'avons pas abordé l'aspect mathématique de la distance sémantique et nous définirons cette expression de la manière suivante :

Définition 11. La distance sémantique permet de quantifier, dans un contexte sémantique donné, l'éloignement du sens de différents signifiés.

Nous utiliserons la notion de distance sémantique d'un point de vue qualitatif et relatif. Ainsi, nous dirons que :
- la distance sémantique entre deux synonymes est nulle ;
- dans le contexte sémantique « repas », la distance sémantique entre la « cuillère » et la « fourchette » est plus faible que la distance sémantique entre la « fourchette » et la « table » ;
- la distance sémantique entre une « pelle » et une « baignoire » dans le contexte « jardinage » est beaucoup plus grande que la distance sémantique entre une « cuillère » et une « fourchette » dans le contexte sémantique « repas ».

La caractérisation du contexte sémantique est importante. Par exemple, si dans le contexte « jardinage », de la terre est stockée dans une « baignoire » à l'aide d'une « pelle », la distance sémantique entre ces signifiés diminue et la dernière des trois assertions devient fausse.

Ambiguïtés sémantique [Godard, 2006], d'après les travaux de [Gillon, 2004], définit [15] la notion d'ambiguïté *sémantique*.

Définition 12. *Une forme est sémantiquement ambiguë si on peut lui faire correspondre au moins deux sens distincts. Les trois propositions suivantes portent des ambiguïtés sémantiques : « Pierre sent la rose », « Cet ours a mangé un avocat » ou encore « Sylvain a vu un homme avec un télescope ».*

Les définitions précédentes sont illustrées par des exemples de la vie courante, cependant il est possible d'identifier, de la même manière, des ambiguïtés sémantiques dans le domaine du PLM, et plus particulièrement dans l'interface conception/production (nous avons pris garde à préciser les domaines sémantiques considérés dans la section 1.2.1).

1.2.2.2. Les ambiguïtés sémantiques dans le PLM

La présentation des activités en charge de la conception et de la production nous amène à formuler les remarques suivantes :
- le domaine sémantique de la conception est caractérisé par les mots-clés **virtuel** et **fonctionnel** ;

15. http ://www.semantique-gdr.net/dico/index.php/Accueil

- le domaine sémantique de la production est caractérisé par les mots-clés **physique** et **temporel** : la gestion industrielle et la gestion de la production s'attachent à *planifier* de manière optimale (flux, stocks, coûts) la succession des opérations conduisant à la réalisation du produit physique. Cette dimension temporelle se retrouve également dans le calcul des besoins, pour lequel des relations de précédence entre composants permettent de déterminer, pour chaque opération de fabrication, les ressources à amener sur le poste de charge.

Nous avons identifié ces nuances, *a priori minimes*, comme étant fondamentales pour expliquer les difficultés d'interopérabilité conception/production. En effet, cette légère distinction entre les domaines sémantiques conduit à des ambiguïtés relatives à la notion de produit et plus particulièrement sa *définition*, ses *vues* et sa *diversité*.

La définition du mot « produit » Dans la norme française relative au vocabulaire de l'Analyse Fonctionnelle [NF-EN-1325-1, 1996], le produit est défini comme étant *le résultat d'une activité humaine, il n'est pas naturel*. Dans le protocole d'application 239 de la norme STEP [ISO-10303-239, 2005], un produit peut *exister dans le monde réel*, *être amené à exister à la suite d'un processus de réalisation* ou encore *être une fonction*. Enfin, dans le domaine de CAO 3D, le produit est généralement entendu comme étant l'assemblage d'un ensemble de géométries 3D élémentaires : on dit qu'on « charge un produit en session ». Un même concepteur sera donc confronté à des sens différents pour des propositions contenant le signifiant *produit* suivant qu'il travaille dans l'un ou l'autre de ces trois contextes. Ceci correspond bien à une ambiguïté sémantique.

Les *vues* du produit Pour les différents métiers concernés par le développement du produit, le terme *Produit* est un *homonyme* : il porte des sens différents suivant le contexte sémantique dans lequel il est utilisé. Ainsi, le département conception verra le produit comme un système dont les fonctions doivent satisfaire au besoin du client, la production comme un système qu'il faut fabriquer et stocker, la maintenance comme un système dont il faut assurer le maintien en conditions opérationnelles, les achats comme un système dont il faut acheter certains composants à des fournisseurs etc. Ce problème est connu sous le nom de *multi-vues* [Zina et al., 2006]. Chaque signifié « Produit » sera représenté par une *vue* utilisée spécifiquement par un métier :

- vue du produit « tel que conçu » ou *as-designed* en conception ;
- vue du produit « tel que planifié » ou *as-planned* et « tel que fabriqué » ou *as-realized* en production ;
- vue du produit « tel que maintenu » ou *as-maintained* en maintenance ;
- etc.

Ceci constitue bien une ambiguïté sémantique au regard de la définition 12.

Difficultés d'ordre sémantique pour la gestion de la diversité produit La diversité produit naît de la volonté conjointe de réduire le nombre de composants et de proposer au client une palette toujours plus riche de produits personnalisés [Agard, 2004], [Amilhastre et al., 2002]. Devant la logique combinatoire conduisant à un grand nombre de configurations, il convient de parler de *sémantiques (au pluriel) du produit*, étant entendu que chaque configuration est à considérer comme un domaine sémantique différent dans lequel l'instanciation d'un composant lui conférera un sens particulier. La multiplication des versions de composants et des configurations de produit, c'est-à-dire de la diversité produit, participe donc de l'accroissement des ambiguïtés sémantiques.

Conclusion Des ambiguïtés sémantiques peuvent perturber l'interface conception/-production et donc l'interopérabilité entre ces métiers. En référence à la figure 1.2.5, nous illustrons ce point de vue sur la figure 1.2.6 par une analogie avec les règles de l'optique : les ambiguïtés sémantique agissent comme une lentille qui diffuse le sens porté par les informations échangées.

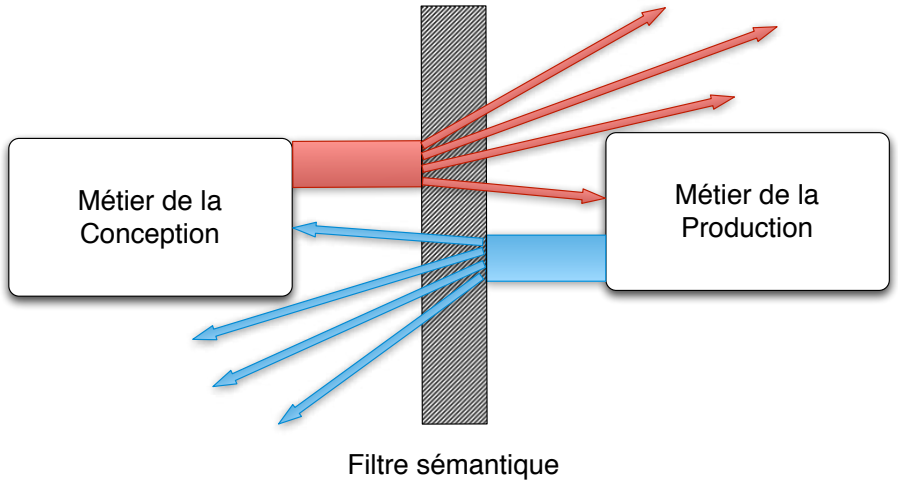


FIGURE 1.2.6.: Les ambiguïtés sémantiques comme filtre à une bonne collaboration conception/production

Cette déduction est d'ailleurs corroborée par le travail de [Jun et al., 2007], *pour qui le flux d'information devient vague dans les phases du cycle de vie postérieures au début du cycle de vie*. Ce sont ces ambiguïtés sémantiques dans un contexte complexe qui font la difficulté des problèmes d'interopérabilité et qui justifient ce travail de recherche : la constitution d'un référentiel sémantique partagé sera d'autant plus difficile que les distances sémantiques seront grandes (donc que les phases du cycle de vie concernées seront éloignées sémantiquement). En définitive, **les techniques d'interopérabilité devront s'attacher à assurer la continuité et la conservation du flux sémantique qui circule dans le système complexe PLM**.

Nous nous concentrons dans la section suivante sur la composante informatique des SI de la conception et de la production (voir 1.2.3), puis nous aborderons la question de leur interopérabilité. Les ambiguïtés concernant la modélisation du produit, sa structuration ou sa diversité seront quant à elles largement abordées dans le chapitre 3.

1.2.3. Les systèmes d'information comme soutien aux activités de conception et de production

Le paragraphe 1.1.2 a montré qu'aujourd'hui les échanges entre acteurs d'un projet s'opèrent de plus en plus de manière numérique, *i.e.* le vecteur de l'information est un support numérique (fichier, e-mail, webconference etc.). Après avoir étudié les *métiers* de la conception et de la production, cette partie s'intéresse aux systèmes d'information supports de ces fonctions. Par extension de la définition 1, nous définirons les SI de la conception et de la production de la manière suivante :

Définition 13. Le SI de la conception est l'*infrastructure de gestion des informations numériques concourant à la définition du produit*, infrastructure caractérisée par la définition 1.

Définition 14. Le SI de la production est l'*infrastructure de gestion des informations numériques permettant la réalisation physique du produit* conformément aux spécifications, infrastructure caractérisée par la définition 1.

1.2.3.1. Le système d'information de la conception

Les logiciels utilisés en conception peuvent être regroupés en trois familles de produits :
- les outils d'*authoring* : ils aident le concepteur à créer des données techniques en fonction de la tâche qui lui a été affectée. Ces données permettent de définir le produit virtuel ;
- la gestion de données techniques : permettent l'organisation, le partage et la conservation des données créées ;
- le pilotage des activités de conception : processus à courte échelle de temps (processus métiers, workflows), ou à moyen/long terme (gestion de projet).

Ces trois catégories d'outils doivent être en mises en regard des trois sorties de l'actigramme modélisant la fonction de conception (voir figure 1.2.1). Nous faisons figurer ces outils dans un tableau à deux dimensions faisant apparaître le type d'outils (authoring, gestion des données techniques, processus) ainsi que le contexte de leur utilisation conformément aux quatre phases définies par [Pahl et al., 1966](voir figure 1.2.7). Dans cette figure, nous faisons apparaître la corrélation qui existe entre les processus, les données et les outils : alors que l'on s'avance vers la phase de conception détaillée, les outils

à mettre en œuvre se spécialisent, la granularité des données s'affine et les processus associés s'inscrivent dans des horizons de temps de plus en plus courts.

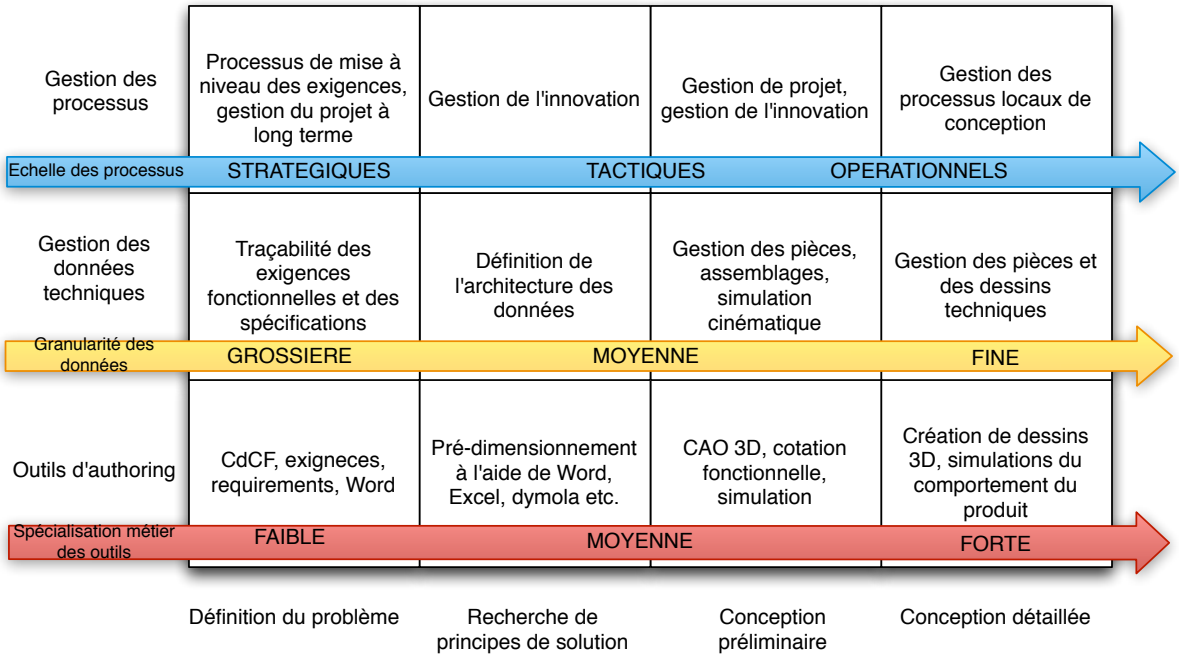


FIGURE 1.2.7.: Classification des logiciels utilisés en conception

Outils d'*authoring* Pour chacune des quatre phases précédentes, les outils logiciels utilisés sont :

- les outils classiques de bureautique (Word, Excel, Access) sont utilisés pour retranscrire les besoins clients en exigence produit. Néanmoins, on peut signaler l'existence de la suite TDC (Need, FMEA), et on peut aussi souligner le fait que Catia V6 intègre une nouvelle structuration (Requirements Functional Logical Physical - RFLP) qui vise à assurer une continuité entre les exigences (Requirements) et la définition du produit ;
- les outils de Conception Assistée par Ordinateur (CAO). Leur périmètre fonctionnel est en fait beaucoup plus réduit que ne pourrait le laisser espérer leur nom, puisqu'ils ne couvrent pas l'intégralité des activités de la fonction conception. Ils permettent aux concepteurs de produire une géométrie 2 ou 3D, encodée dans un format particulier (format propriétaire ou neutre) : Catia V5, SolidWorks, Inventor, Solid Edge, SPaceClaim etc. Ces outils tendent vers une certaines *modularité* : des outils métiers (tôlerie, fonderie, injection plastique etc.) peuvent enrichir les fonctionnalités de base de ces logiciels ;
- les outils de simulation : permettent aux ingénieurs d'exécuter une batterie d'expérimentations numériques pour caractériser et valider le comportement du produit. Ces outils se basent sur la géométrie générée, ainsi que des informations du cahier des charges fonctionnel (environnement du produit, critères de performances attendus) et produisent des résultats numériques qu'il faut ensuite interpréter. On peut citer par

exemple les outils de simulation par élément finis pour les calculs d'écoulements fluides (Fluent), de tenue mécanique (Abaqus, Ansys), de transfert de chaleur (QfinSoft) etc. Il est important toutefois de noter que ces outils de simulation tendent à intégrer les outils de CAO précédemment mentionnés (par exemple le rachat d'Abaqus par Dassault Systèmes a donné naissance à un nouvel outil Simulia, intégré à la plateforme V6).

Outils de gestion des données techniques (Product Data Management - PDM)

Les 3 familles de produits logiciels précédemment citées sont considérées comme des outils d'*authoring :* elles assistent l'ingénieur dans son processus de développement et l'aident à générer de l'information numérique. Devant la masse considérable de données que les ingénieurs doivent traiter, des outils sont apparus pour en permettre l'organisation. Du côté de la conception, c'est l'outil de Gestion de Données Techniques (ou Product Data Management - PDM) qui satisfait à ce besoin : contrairement aux outils d'authoring, le PDM ne crée pas d'information relatives à la définition du produit.

D'après [Eynard et al., 2004], un système PDM *a pour fonction de gérer les données relatives à la conception et à la fabrication des produits. Les fonctionnalités principales sont :*

- *un gestionnaire de la structure produit, qui organise et stocke les informations du produit ; il gère les structures produits et les multiples configurations du produit. Il propose des fonctionnalités permettant le versionnement ainsi que le lien entre documents et composants de la structure produit. Il garantit aussi la gestion de la maturité des données (disponible, gelé, obsolète) ;*
- *un moteur de gestion des processus qui, pour une structure produit donnée, permet d'envoyer la bonne donnée à la bonne personne et au bon moment ;*
- *beaucoup d'autres fonctionnalités sont également disponibles, comme l'intégration avec un serveur de messages électroniques, qui notifie les utilisateurs d'un événement relatif à un document.*

Pour [Audrain, 2008], *un système PDM doit réaliser les fonctions suivantes :*

- *la liste exhaustive des composants du produit ;*
- *la gestion de l'état de maturité des composants et documents associés ;*
- *l'intégration avec les outils d'*authoring *;*
- *la structuration du produit et la gestion multi-vues.*

En revanche, d'après cet auteur, *le PDM n'a pas pour objectif de gérer les pièces physiques ou de rechange*, ce qui l'éloigne de fait des domaines sémantiques relatifs au produit physique, comme par exemple la production ou la maintenance.

Le PDM peut être considéré comme *modulaire* : le logiciel Windchill, édité par PTC, offre la possibilité de n'acheter/installer/utiliser que quelques fonctionnalités parmi toutes celles offertes. Le PDM se situe donc au cœur de l'activité de conception, puisqu'il prend en charge aussi bien les données que les processus. Par ailleurs, le PDM ne doit pas être confondu avec le PLM : en reprenant notre définition du PLM, le PDM peut être vu comme un outil qui entre dans le cadre d'une stratégie PLM.

Outils de gestion de projet Les outils de gestion de projet ont pour fonction de permettre le pilotage du projet (maîtrise des délais et des coûts) ainsi que le contrôle fin du travail collaboratif. Ils offrent des fonctionnalité de type :

- gestion de planning ;
- décomposition du projet en sous-projets et tâches ;
- visualisation du plan projet sous forme de diagrammes Gantt/Pert ;
- définition des jalons et des livrables ;
- constitution des équipes, des membres et de leur rôle.

Des exemples d'outil de gestion de projet utilisables dans un contexte industriel sont Microsoft Project ou PTC Windchill ProjectLink. Les outils de gestion de projet permettent de gérer des processus sur des échelles de temps plus longues que les processus opérationnels implémentés dans le PDM.

Interactions de ces outils Les interaction entre ces 3 types d'outil (authoring, PDM, gestion de projet) sont de plusieurs ordres :

- les activités de conception (*tâches* en gestion de projet) génèrent des données projet (devis, cahier des charges, budget) ou produit (qui permettent de définir le produit de manière univoque) ;
- la modification de ces données par le concepteur peut amener au déclenchement d'un workflow (un workflow de validation par exemple).

1.2.3.2. Le système d'information de la production

Les outils informatiques ont diffusé largement dans le domaine de la production, et ont étendu la fonction production de la création de produits physiques (son activité historique) à la gestion des flux physiques et d'informations mises à disposition de tous les acteurs de l'entreprise. Le système d'information chargé du pilotage de la gestion de production concerne l'utilisation efficiente des ressources (ERP), l'exécution et le suivi de la fabrication (MES) ainsi que la planification avancée (APS) des opérations de fabrication.

Enterprise Resource Planing (ERP) Selon [Grabot et al., 2008], *les système ERP constituent aujourd'hui l'arête dorsale des systèmes d'information de la plupart des grandes et moyennes entreprises*. Pour [Tarn et al., 2002], l'objectif principal d'un ERP est *d'unifier les différents départements de l'entreprise au travers d'un système d'applications. Les informations gérées par un ERP peuvent être utilisées de différentes manières : par exemple, les directions des services production, relation client, finance/-comptabilité peuvent utiliser les informations de l'ERP comme aide à la décision. Il permet à différents services de l'entreprise de travailler de manière collaborative et de communiquer via une interface commune*. Pour leur part, [Kumar and Van Hillegersberg, 2000]

définissent le système ERP comme *un ensemble modulaire de systèmes d'information qui intègre les informations et les processus de toutes les fonctions de l'entreprise* (cette définition ramène l'ERP dans la famille des systèmes de systèmes au sens précédemment exprimé).

Dans les propos liminaires d'une étude concernant l'intégration de l'ERP avec la gestion de la chaîne logistique, [Tarn et al., 2002] détaillent l'architecture et les fonctions d'un système ERP : il consiste en une série de modules intégrés dédiés à la gestion des comptes clients (accouting), la distribution, le marketing et les ventes, la fabrication, les ressources humaines.

Manufacturing Execution Systems (MES) D'après la définition du National Institute of Standards and Technology américain [Barkmeyer et al., 1999], reprise par [Mirdamadi, 2007] dans son travail relatif au pilotage d'atelier, un MES est un *ensemble de composants matériels et logiciels permettant la gestion et l'optimisation des activités de production, du lancement de la commande jusqu'à l'obtention des produits finis. Tout en maintenant des données précises et à jour, le MES guide, initie, répond et rapporte les évènements de l'usine au moment où ils se produisent. Un MES fournit des informations critiques aux modules d'aide à la décision à travers l'entreprise.*

Les fonctions principales d'un Manufacturing Execution System sont :
- suivre et allouer les ressources (assurer que les conditions de démarrage de la production sont respectées) ;
- ordonnancer les opérations (en fonction des priorités, caractéristiques, procédés liés à un équipement) ;
- « dispatcher » le travail aux ressources (ordres de fabrication, préparation des matières premières, de la manutention etc.) ;
- gérer les spécifications (notices techniques, dessins de définition, procédures etc.) ;
- acquérir et stocker les données de production et de maintenance ;
- gérer la qualité (analyses temporelles du contrôle produit/process provenant des opérations de fabrication) ;
- gérer la maintenance (historique et traçabilité des pannes/opérations de maintenance, maintenance des équipements et des outils pour assurer leur disponibilité etc.) ;
- superviser la fabrication du produit (conditions de production, informations sur l'état des ressources etc.) ;
- analyser la performance (utilisation des ressources, disponibilité des ressources, temps de cycle etc.) ;
- organiser les flux de matières premières dans l'atelier.

Advanced Planning Systems (APS) Les systèmes MRP ont évolué à la fin des années 1990 vers le Supply Chain Management. Ces systèmes deviennent de véritables outils d'aide à la décision, dotés d'algorithmes d'optimisation, ils simulent l'ensemble du

système logistique, vérifient la disponibilité des produits et des capacités de production et de distribution nécessaires pour faire face à différents niveaux de demande. Ces outils sont couramment dénommés des *Advanced Planning Systems* (APS) par les entreprises qui les commercialisent. D'après la société CXP [16], les *APS sont des progiciels qui optimisent la planification et synchronisent les flux de la chaine logistique en tenant compte simultanément d'un grand nombre de contraintes (ressources, capacités, délais, couts, profits etc.)* .

D'après le dictionnaire APICS de l'*Association for Operations Management, les APS sont généralement utilisés pour évaluer des scénarii multiples* [Cox and Blackstone, 2002]. *Le Management sélectionne un de ces scénarii, qui devient alors la planification officielle. Les cinq fonctions principales des systèmes APS sont :*

1. *Planifier la demande ;*
2. *Planifier la production ;*
3. *Ordonnancer la production ;*
4. *Planifier la distribution ;*
5. *Planifier le transport.*

Ces outils semblent aujourd'hui capables de synchroniser l'ensemble des ressources matières et capacitaires pour aboutir à une planification dite *optimale* et ainsi apporter une réponse aux faiblesses de synchronisation des systèmes traditionnels MRP II [Thomas and Lamouri, 2000]. Les APS sont souvent présentés comme un concept innovant de planification. Or les organisations utilisent des techniques de planification avancée depuis plusieurs années [Stadtler and Kilger, 2000]. En effet, les APS reposent sur un ensemble de méthodes connues, rendues accessibles et efficaces par l'amélioration continue des technologies de calculs ; l'émergence des architectures de communication client-serveur, le développement des bases de données relationnelles, puis l'expansion de l'EDI (Échange de Données Informatisées) ont contribué a assurer le développement de ces outils. *Toutes les entreprises disposent maintenant, à un coût abordable, d'une puissance de traitements informatiques sans commune mesure avec celles dont disposaient les plus grosses entreprises, il y a une vingtaine d'années* [Giard, 2003].

Dans la section suivante, nous proposons un point de vue synthétique de ces systèmes et applications, pour en déduire des propriétés quant à leur interopérabilité.

1.2.3.3. Le système d'information PLM

Le volet applicatif du système d'information du PLM va bien au-delà de la conception et de la production : il *englobe* les systèmes de la conception et de la production, et d'une

16. http ://www.cxp.fr/

manière générale tous les systèmes permettant de supporter les phases du cycle de vie du produit. Nous représentons graphiquement ce point de vue sur la figure 1.2.8.

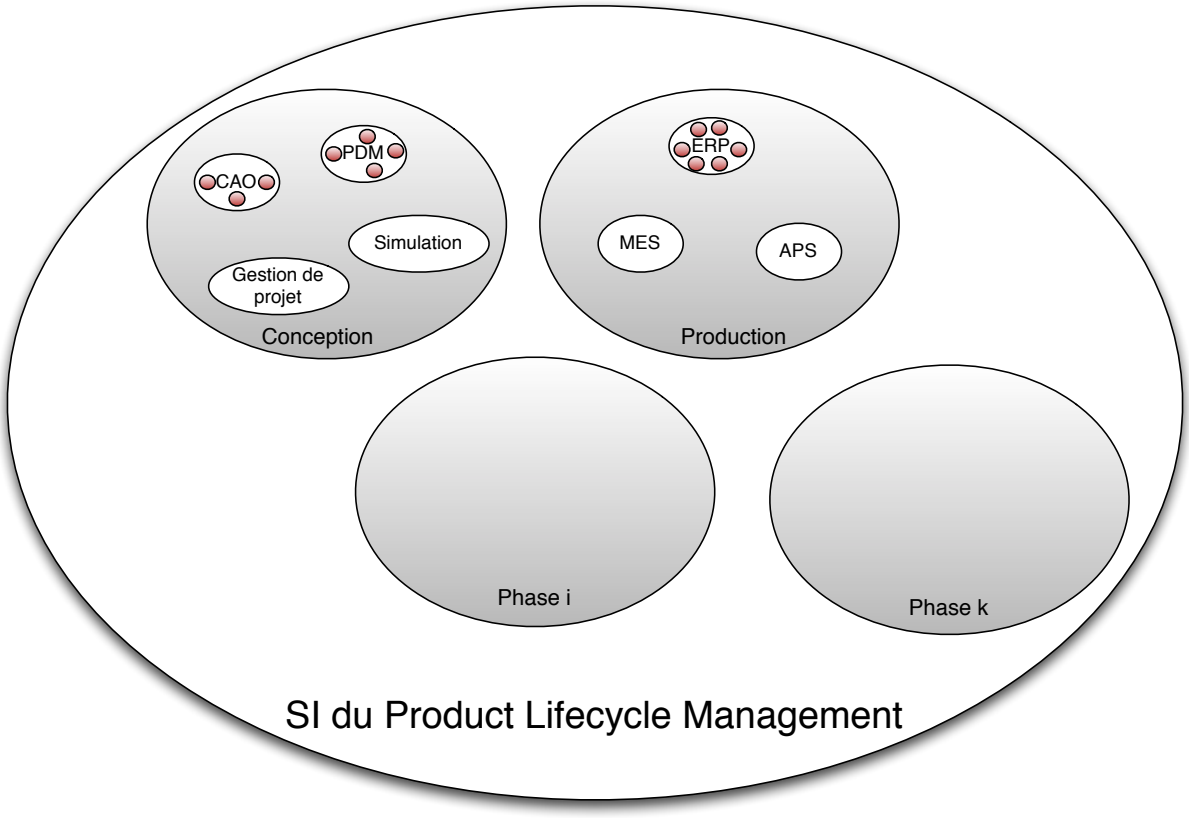


FIGURE 1.2.8.: Le volet applicatif du système d'information PLM

Dans cette figure, c'est un critère de distance sémantique qui permet de représenter la position relative de chaque bulle ainsi que sa dimension :

- chaque bulle grisée correspond à un ensemble d'applications constituant le système d'information d'*une* phase du cycle de vie du produit. Nous n'avons aucun élément nous permettant d'affirmer qu'un ensemble est plus ou moins « volumineux » ou « grand » qu'un autre : nous faisons donc l'hypothèse que les *diamètres sémantiques* sont du même ordre de grandeur et représentons les systèmes par des bulles de même dimension. Pour des raisons de mise en page, nous nous sommes limités à quatres bulles, mais il y en a évidemment autant que de phases du cycle de vie du produit ;
- la position relative de chaque bulle grisée représente la distance sémantique entre deux systèmes. Nous n'avons aucun élément nous permettant d'affirmer que la distance sémantique entre la conception et la production est plus grande qu'entre la conception et la maintenance. Nous faisons donc l'hypothèse que les distances sémantiques sont toutes du même ordre de grandeur et que les distances entre bulles grises sont égales deux à deux. Ainsi, trois bulles grises devraient être représentées comme les sommets d'un triangle équilatéral, quatre bulles comme les sommets d'un tétraèdre régulier.

Au-delà de quatre bulles, la représentation en deux ou trois dimensions correspond à la projection d'une hyperpyramide et les distances entre éléments projetés ne sont pas conservées. C'est donc le *biais* induit par la méthode de représentation qui fait que, sur la figure 1.2.8, la $k^{ème}$ phase *apparaît* plus éloignée de la conception que de la production.

- à l'intérieur de chaque bulle grisée, une bulle blanche représente une application ou un ensemble d'applications remplissant la même fonction. Seules les applications correspondant à la conception et à la production ont été représentées. Nous faisons l'hypothèse que la distance sémantique entre la CAO et le PDM est du même ordre de grandeur que celle entre l'ERP et le MES ;
- certaines de ces applications sont elles-mêmes modulaires (ERP, PDM, CAO). Ces modules apparaissent sous forme de petites bulles roses ;
- enfin, les remarques présentées dans le deuxième point (distances sémantiques du même ordre de grandeur) s'appliquent également aux bulles blanches (applications) et roses (composants modulaires des applications).

Ce point de vue permet de faire apparaître que le **système d'information du PLM intègre une dimension multi-échelle**. **Cet aspect multi-échelle que nous mettons en évidence n'a pas été pensé de manière globale, il s'est mis en place progressivement au cours du temps : nous le considérons *in fine* comme un réflexe d'auto-organisation [Le Moigne, 1977], c'est-à-dire la réponse du système à l'accroissement de sa complexité intrinsèque selon un critère de distance sémantique**.

Cet état de l'art sur les systèmes d'information permet donc de situer clairement la problématique de leur interopérabilité. A l'issue de ces deux premières sections, nous pouvons arguer que **l'interopérabilité dans le système d'information du PLM, visant à garantir le flux sémantique qui circule dans le système complexe PLM, doit être envisagée au regard de caractéristiques multi-échelles suivant deux dimensions :**

- **une dimension temporelle (stratégique/tactique/opérationnel) ;**
- **une dimensions sémantique (sémantique globale/sémantique locale).**

Il existe une forte corrélation entre ces deux dimensions : à dimension temporelle stratégique, sémantique globale ; à dimension temporelle opérationnelle, sémantique locale. C'est donc de ce point de vue que nous proposons de structurer, dans la section suivante, l'ensemble des travaux de la littérature relatifs à l'interopérabilité du SI PLM.

1.3. État de l'art concernant l'interopérabilité des systèmes d'information de la conception et de la production

Nous nous intéressons dans cette partie aux travaux relatifs à l'interopérabilité des systèmes d'information, exclusivement pour les phases du cycle de vie concernant la conception et la production. Dans un premier temps, nous séparons les problèmes d'interopérabilité en deux grandes catégories puis nous situons les travaux de la littérature relativement à ces catégories.

1.3.1. Étendue des problèmes d'interopérabilité

Pour faire suite aux constatations de la section précédente, nous reprenons la figure 1.2.8 sur laquelle nous faisons apparaître deux catégories de problèmes traitées dans la littérature (voir figure 1.3.1).

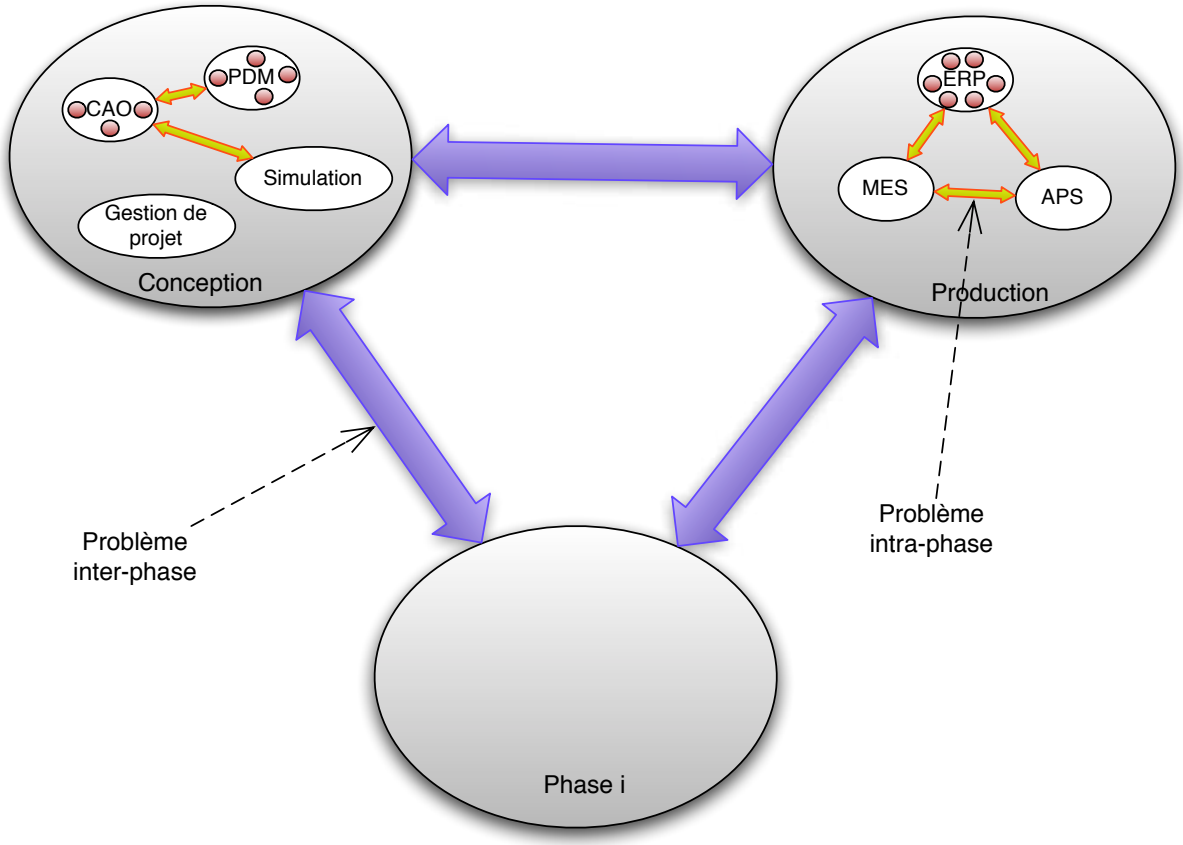


FIGURE 1.3.1.: Catégorie des problèmes d'interopérabilité rencontrés dans la littérature

Sur cette figure :
- les flèches violettes représentent les problèmes d'interopérabilité de systèmes en charge de plusieurs phases du cycle de vie du produit. Dans ce cas, les distances sémantiques sont importantes, et nous dilatons à cet effet la largeur de la flèche : la sémantique transportée a une grande étendue ;
- les flèches jaunes représentent les problèmes d'interopérabilité à l'intérieur d'un même système. Les distances sémantiques sont plus faibles : l'étendue sémantique des informations transportées est plus petite que dans le premier cas, ce que nous faisons figurer par une flèche de largeur moins importante.

Il est à noter que la largeur des flèches n'est en rien liée au volume ou au débit des données. C'est bien un critère sémantique qui est utilisé.

Nous dénommerons dans la suite de ce mémoire ces deux catégories de problèmes comme *inter-phase* ou *intra-phase* et c'est suivant cette classification que nous présentons les travaux de la littérature.

1.3.2. Les travaux de la littérature concernant l'interopérabilité dans le domaine du PLM

Nous rappelons au préalable que la totalité des auteurs qui suivent signalent l'interopérabilité des systèmes d'information comme une nécessité, et constatent unanimement leur manque d'interopérabilité comme un problème encore ouvert.

Problèmes intra-phase [Song et al., 2007] spécifient une plateforme d'intégration CAx/PDM basée sur le modèle de produit défini par le protocole d'application 203 de la norme STEP. Les résultats de ce travail sont implémentés sur la base du noyau *Cooperative Design Modeller* (CoDeMo).

[Tursi, 2009] propose une interopérabilité ERP/MES. Le modèle de données utilisé pour l'unification des modèles se base sur la représentation du produit spécifiée dans le PDMSchema. Un ensemble de règles expertes permet de contraindre la transformation des nomenclatures lors de leur transfert d'un système à l'autre. Cette approche par règles s'apparente à celle proposée par [Ou-Yang and Cheng, 2003].

[Oh et al., 2001] étudient un mapping pour l'échange de données d'un logiciel de CAO (IDEAS V6) vers un outil PDM (Metaphase). Ce travail s'appuie sur un modèle de données extrait du protocole 203 de la norme STEP et le PDM Schema.

[Baïna, 2006] étudie les interopérabilités ERP/MES et MES/CRP suivant une méthode d'ingénierie dirigée par les modèles. L'auteur utilise dans son travail un modèle de produit intelligent.

[Gallet et al., 2007] étudient l'interopérabilité CAO/Calcul/PDM dans le cas d'une chaîne de conception aéronautique. Les informations transitent *via* un ensemble de fichiers XML et le processus est largement manuel, c'est-à-dire qu'il requiert de nombreuses manipulations de la part de l'utilisateur.

Problèmes inter-phase [Noël and Roucoules, 2008] proposent un modèle product/-process/organisation (PPO) adossé à une plateforme intitulée *Integration of Product-Process-Organization for engineering Performance improvement* (IPPOP) [17] dédiée à l'implémentation de ce modèle. Ces deux éléments visent à établir une plateforme d'intégration des applications utilisées dans le PLM. [Rose et al., 2007] intègrent à cette plateforme un module de gestion des processus collaboratifs. En revanche, d'après [Ducellier, 2008], *les solutions proposées par IPPOP ne fournissent pas une solution de remplacement des plateformes PLM. En effet, le niveau de formalisation des données dans une plateforme PLM est bien moins granulaire que celui des données issues de la plateforme IPPOP.*

[Galeta et al., 2006] proposent un modèle de produit *as-designed* destiné à être implémenté dans un ERP. Ce modèle *ad hoc* est intimement lié au système PDM choisi, donc difficilement généralisable. Les auteurs concluent que l'utilisation du modèle de données STEP PDM Schema doit être envisagée.

[Xu et al., 2008] étudient les conversions de nomenclature de conception et de production. Ils définissent un ensemble de règles de transformation ainsi qu'un algorithme permettant de transférer les nomenclatures d'un système vers l'autre. Ce travail ne permet toutefois de pouvoir traiter qu'un nombre restreint de cas d'études, sachant que les nomenclatures traitées par les règles et l'algorithme proposés doivent respecter *a priori* un ensemble de critères très restrictifs.

[Ou-Yang and Cheng, 2003] développent un environnement pour une intégration MRP/PDM. Ce travail couvre un spectre fonctionnel très large : gestion des modifications en conception, transfert de nomenclature, évaluation des coûts de stocks de pièces de rechange. Le modèle de produit utilisé dans ce cadre n'est pas précisé, la granularité de ce travail n'étant pas très détaillée.

[Cheung et al., 2010] contruisent un environnement permettant l'intégration entre un ERP et un système PDM/PLM. Cet environnement *web based* adresse aussi bien les données que les processus (*via* une nouvelle méthodologie intitulée *Workflows Activity Task Controller* permettant de synchroniser les processus pris en charge par les systèmes de type PLM, ERP ou Process Planning). L'échange d'informations entre ces systèmes repose sur des transferts de fichiers au format XML. Le modèle de données utilisé n'est pas détaillé.

[Giménez et al., 2008] construisent une ontologie produit intitulée *PRoduct ONTOlogy* (PRONTO) pour une représentation structurale du produit à même de couvrir les phases

17. http ://ippop.laps.u-bordeaux1.fr

de conception et de production. La sémantique couverte par cette ontologie est cependant assez pauvre.

Les manques identifiés dans ces différentes approches La lecture de ses différents travaux nous permet de conclure d'une manière générale que :

- dans le cas de problèmes d'interopérabilité inter-phase, l'accent est mis principalement sur des questions d'architecture. La question des modèles est soit éludée, soit abordée avec une granularité grossière. Ainsi, dans ces cas, la continuité du flux sémantique est assurée mais pas sa conservation ;
- dans le cas de problèmes intra-phase, la granularité des modèles de données s'affine. En revanche, l'extensibilité de ces modèles n'est pas traitée pas plus que les questions de plus haut niveau concernant l'architecture du SI ne sont détaillées. Dans ce cas, la conservation du flux sémantique est donc assurée mais limitée à un cas précis, et les détails concernant l'architecture du SI ne sont pas exposés : la continuité du flux ne peut être assurée.

Par ailleurs :

- les transferts d'information entre applications s'opèrent majoritairement sous forme de *transferts de fichiers*. Ces fichiers sont utilisés comme *vecteurs* de l'information, mais la gestion de la traçabilité du fichier, de son historique, de son stockage (ou de sa volatilité) n'est pas abordée ;
- dans les articles présentant des modèles de données (granularité fine), les détails liés à l'implémentation ne sont pas suffisamment précisés : il est très difficile de reproduire les résultats présentés. Bien qu'il ne soit pas permis de douter de la validité scientifique de ces résultats, les incertitudes d'implémentation empêchent de pouvoir déceler des problèmes qui n'auraient pas été identifiés ou dont le détail n'aurait pas été présenté, et limite par conséquent les possibilités d'amender, compléter ou améliorer ces travaux. La seule exception notable que nous ayons identifiée est celle du projet IPPOP : la plate-forme de développement est librement disponible. Malheureusement, aucune mise à jour semble n'avoir été apportée depuis 2005, et la technologie a évolué depuis. Nous considérons que l'absence de référentiel d'implémentation partagé, basé sur l'utilisation des dernières technologies, constitue un frein à la construction collective (à l'échelle de la communauté) et incrémentale d'un savoir-faire dans le domaine de l'interopérabilité. A cet égard, nous estimons que l'implémentation des modèles doit être traitée de manière pointue et que le choix des outils et techniques d'implémentation constitue bien un verrou ;
- le choix des modèles pour la réconciliation d'informations hétérogènes n'est pas suffisamment abordé. L'utilisation d'un modèle de données *ad hoc* ou standard ne doit pas être un postulat mais bien le résultat d'une démarche argumentée, de même que le choix d'une approche d'unification, d'intégration ou de fédération de modèles ;

Par conséquent, nous proposons dans la suite de ce mémoire une méthodologie permettant de s'affranchir de ces problèmes.

1.4. Vers une méthodologie générale permettant d'appréhender les problèmes d'interopérabilité dans le domaine du Product Lifecycle Management

La méthodologie que nous proposons permet simultanément :
- de traiter des problèmes inter-phase. Nous illustrerons ce point par l'interopérabilité PDM/ERP ;
- de traiter des problèmes intra-phase, ce qui sera l'objet de notre travail sur l'interopérabilité CAO/PDM ;
- d'adresser aussi bien les questions d'architecture du SI que celles concernant les modèles de produit ;
- d'être robuste vis-à-vis d'une extension du problème d'interopérabilité : ajout d'un composant dans le SI ou changement d'échelle sémantique ;
- de s'affranchir du *fichier* comme vecteur de l'information entre SI ;
- d'être reproductible puisque les détails d'implémentation seront largement détaillés.

Dans un contexte d'**accroissement de la complexité**, le critère qui guide notre recherche est l'**accompagnement de la mise à l'échelle temporelle et sémantique** : nous devons assurer la robustesse vis-à-vis d'un changement d'échelle de l'une et/ou l'autre de ces dimensions et garantir la flexibilité et l'agilité des solutions proposées. Dans ce cadre :
- le chapitre 2 s'intéresse à une **architecture** robuste, flexible et agile capable d'assurer la **continuité** du flux sémantique ;
- le chapitre 3 étudie les **modèles** robustes, flexibles et agiles capables de garantir la **conservation** du flux sémantique ;
- le chapitre 4 précise l'**implémentation** de l'architecture et des modèles, ainsi que leur validation.

L'ensemble des chapitres 1, 2, 3 et 4 constituera notre *Méthodologie de résolution des problèmes d'interopérabilité dans le domaine du PLM.*

Résumé du chapitre 1

Le développement de produit constitue un système de systèmes caractérisé par une complexité croissante : il implique un grand nombre d'acteurs dont les interactions sont non-triviales. Pour faire face à la tendance à l'accroissement de cette complexité, l'objectif stratégique du Product Lifecycle Management est défini comme la maîtrise de la complexité qui caractérise le développement et le suivi des produits.

Les principaux obstacles à la maîtrise des inter-relations sont d'ordre sémantique : les différences qui caractérisent les domaines sémantiques conduisent à des ambiguïtés. Le système d'information qui soutien la démarche PLM est analysé au regard des métiers concernés et de ces ambiguïtés : il est lui aussi un système complexe qui présente un double aspect multi-échelle temporel et sémantique.

Dans ce contexte, l'interopérabilité dans le domaine du PLM répond à l'objectif de maîtriser les flux sémantiques dans ce SI. Pour satisfaire à cet objectif, il convient de veiller à assurer la continuité de ces flux et de garantir leur conservation.

Nous proposons une méthodologie centrée sur l'étude de l'architecture du SI et la spécification et la définition des modèles de données. La restriction de notre étude à deux phases particulières du cycle de vie du produit, la conception et la production, permet de restreindre l'étendue de notre travail sans limiter sa portée plus générale ni occulter ou travestir la difficulté du problème.

2. Une architecture orientée services basée sur un méta-modèle d'unification

Ce chapitre vise à définir l'architecture d'un système PLM interopérable à même de respecter les contraintes de continuité et de conservation du flux sémantique formulées à la fin du chapitre 1. Nous cherchons à construire cette architecture à partir des résultats issus des travaux touchant à l'interopérabilité des systèmes d'entreprise. Les définitions relatives à cette notion sont tout d'abord précisées dans la section 2.1. La classification en *niveaux d'interopérabilité* guide notre démarche :

- le *niveau technique* nous conduit à définir dans la section 2.2 une architecture de *médiation multi-échelle orientée services* pour assurer la continuité de ce flux ;
- le *niveau sémantique* nous amène, dans la section 2.3, à proposer l'utilisation d'un *méta-modèle d'unification* pour ce qui concerne la mise en correspondance sémantique des modèles de données et donc la conservation du flux sémantique (les détails concernant ce(s) méta-modèle(s) seront présentés dans le chapitre 3).

Les résultats que nous présentons sont assortis de prescriptions méthodologiques.

2.1. Interopérabilité des systèmes d'information d'entreprise

Le chapitre 1 a permis de *construire* notre vision de l'interopérabilité par rapport aux problèmes se posant dans le domaine du PLM, sans que nous n'ayons *défini* cette notion. Nous proposons dans cette section de confronter cette construction aux définitions formelles issues de la littérature relative à l'interopérabilité des systèmes d'entreprise. Nous verrons par la suite dans quelle mesure ces travaux permettent d'envisager des *solutions* pour répondre aux problèmes identifiés à la fin du chapitre introductif.

2.1.1. Définitions de l'interopérabilité

La littérature offre plusieurs définitions différentes de l'interopérabilité. [Baïna, 2006] dresse un inventaire non-exhaustif de ces définitions que nous reprenons pour partie en

le complétant. Selon les auteurs, l'interopérabilité est :

1. *la capacité de deux ou plusieurs systèmes ou composants à échanger des informations et à utiliser les informations échangées* [Geraci et al., 1991] ;
2. *réussie si et seulement si l'interaction entre les systèmes impliqués couvre les aspects données, ressources et processus métiers en utilisant les sémantiques définies dans le domaine métier* [Chen and Doumeingts, 2003] ;
3. *la capacité à communiquer avec des systèmes pairs et accéder à leurs fonctionnalités* [Vernadat, 1996] ;
4. *l'aptitude de deux systèmes (ou plus) à communiquer, coopérer et échanger des données et services, et ce malgré les différences dans les langages, les implémentations et les environnements d'exécution ou les modèles d'abstraction* [Wegner, 1996] ;
5. dans le domaine des sciences de l'information [1], *la capacité pour des systèmes à échanger et utiliser des informations (généralement dans un réseau hétérogène composé de plusieurs réseaux locaux).*

[Kosanke, 2005] définit pour sa part le néologisme *interopérable* connexe à la notion d'interopérabilité.

Définition 15. *Interopérable* : *adjectif caractérisant le fait que deux systèmes peuvent échanger et utiliser de l'information.* On dira de deux systèmes qu'ils sont *interopérables.*

Par corollaire, nous définissons le verbe *interopérer.*

Définition 16. *Interopérer* : action mutuelle de deux systèmes en situation d'échanger et utiliser des informations. On dira de deux systèmes qu'ils *interopèrent.*

Toutes les définitions de l'interopérabilité sont équivalentes (au sens de leur clarté, leur cohérence, leur extensibilité, leur déformation d'encodage ou leur engagement ontologique) et se retrouvent complètement dans la classification suivante qui décompose l'interopérabilité en plusieurs *niveaux.*

2.1.2. Les niveaux d'interopérabilité

D'après [EIF, 2004], l'interopérabilité peut se produire à plusieurs niveaux :

- niveau *technique* (données et messages échangés) : pour pouvoir échanger des informations, il faut au préalable s'assurer que le transport des données d'un système à l'autre soit possible, c'est-à-dire qu'il existe un vecteur des données fonctionnel. L'exemple de deux personnes en communication téléphonique permet d'illustrer ce point de vue : pour que des paroles soient échangées, il est nécessaire que les deux personnes puissent parler, entendre, et que leurs téléphones/le réseau soient en état de fonctionnement. C'est ce que sous-tend l'expression « capacité à échanger » qui apparaît dans les définitions ci-dessus. Le niveau technique est une condition nécessaire mais non suffisante pour établir l'interopérabilité ;

1. http ://wordnet.princeton.edu/

- niveau *sémantique* (information et partage de services) : un vecteur d'information fonctionnel ne suffit pas, encore faut-il que les données échangées soient *comprises* par les deux systèmes. Les données, chargées de sens, deviennent alors des informations qui peuvent être traitées par les systèmes en question. Pour l'exemple de personnes en communication téléphonique, les deux parties doivent être capables d'*interpréter* le sens du message de leur interlocuteur (nous retrouvons la notion de *système interprétatif* de [Tsuchiya, 1993] présentée dans la partie concernant les connaissances en conception). C'est ce que sous-tendent les expressions « utiliser l'information » ou « niveau sémantique » ;
- niveau *organisationnel* (interactions *business unit*/processus/personnes au travers de l'organisation) : une organisation adaptée doit être prévue pour assurer l'échange des informations. Dans le cas de la communication téléphonique, l'échange des informations ne sera pas possible si l'une des personnes est absente. Il sera perturbé si l'une d'elles est occupée simultanément par une autre activité. C'est ce que sous-tendent les expressions « interopérabilité structurale » ou « niveau organisationnel ».

Il faut veiller à satisfaire simultanément tous ces niveaux pour garantir une interopérabilité complète entre SI. Dans le cadre de notre recherche, nous avons concentré nos efforts sur les deux premiers niveaux (techniques et sémantiques). Le niveau organisationnel implique des problèmes relatifs aux processus (ainsi que leur modélisation) que nous n'avons pas étudiés. Nous argumenterons néanmoins en fin de chapitre les raisons nous laissant à penser que les résultats présentés sont prêts à être enrichis d'un aspect organisationnel, et qu'ils ne sont pas contradictoires avec notre approche.

Nous proposons dès lors d'établir un parallèle entre les deux premiers niveaux d'interopérabilité et les notions de continuité et de conservation du flux sémantique que nous avons déjà présentées.

2.1.3. Niveaux d'interopérabilité et flux sémantique

Au regard de cette classification en niveaux d'interopérabilité, nous pouvons établir le parallèle suivant :

- c'est au niveau technique que se situe la problématique de continuité du flux sémantique ;
- c'est au niveau sémantique que se situe la problématique de conservation du flux sémantique.

Le premier de ces deux points (niveau technique) fait l'objet de la section 2.2, dans laquelle nous étudions les topologies de réseau permettant de construire une architecture flexible et modulaire pour la continuité du flux sémantique. La section 2.3 sera centrée sur le deuxième point (niveau sémantique).

2.2. Interopérabilité technique

2.2.1. Architectures envisageables

Dans cette section, nous étudions et comparons les types d'architectures possibles permettant de réaliser l'interopérabilité dans un ensemble E_S constitué de n systèmes d'informations.

Critères de comparaison des architectures Ns, nous proposons de prendre en compte les deux critères suivants pour choisir l'architecture la plus pertinente permetant d'assurer la continuité du flux d'information :

- **le coût global de possession** (CGP) des interfaces (coûts de développement, de déploiement, de maintenance) ;
- **l'agilité du SI** constitué : l'agilité caractérise une propriété d'adaptation du système après occurrence d'un événement. Celle-ci met en œuvre deux autres propriétés d'adaptation que sont la *réactivité* et l'*adaptabilité* [Iskanius, 2006], [Barzi, 2007].

Topologies de graphe complet / en étoile Pour garantir la continuité du flux d'information dans le système E_S, chaque couple (i, j) de systèmes doit être capable d'échanger des données, c'est-à-dire qu'il doit exister, pour la topologie de réseau choisie, au moins un chemin reliant tout couple de sommets (i, j). *In fine*, c'est donc une condition de connexité du graphe modélisant le réseau qui doit être satisfaite. Parmi toutes les topologies de type « graphe connexe », deux types d'architectures (voir figure 2.2.1) émergent de la littérature concernant le PLM [Guyot et al., 2007] :

- une architecture de type *point-à-point*, dont la topologie est celle d'un graphe complet, dans laquelle chaque système i est connecté au système j via une interface dédiée. Le nombre total d'interfaces $I_{point-to-point}$ nécessaires est égal au nombres d'arêtes d'un graphe complet présentant n sommets :

$$I_{point-to-point} = \sum (n - i) = \frac{n(n-1)}{2}$$

- une architecture de *médiation* (topologie en « étoile »), pour laquelle un $(n+1)^{ème}$ système est ajouté. Dans cette architecture, introduite par [Wiederhold, 1992], ce système additionnel est appelé *médiateur d'information*. D'après cet auteur, *ce composant a pour fonction d'assurer l'interopérabilité des systèmes*. Pour cette topologie, le nombre d'interfaces nécessaires $I_{Mediator}$ est tel que :

$$I_{Mediator} = n$$

On vérifie bien que, pour chacune des deux architectures, la condition de *connexité* du graphe est satisfaite : pour tout couple (i, j) de systèmes, il existe au moins un chemin

liant le sommet i au sommet j, i.e. l'échange d'information entre les deux systèmes i et j est assurée.

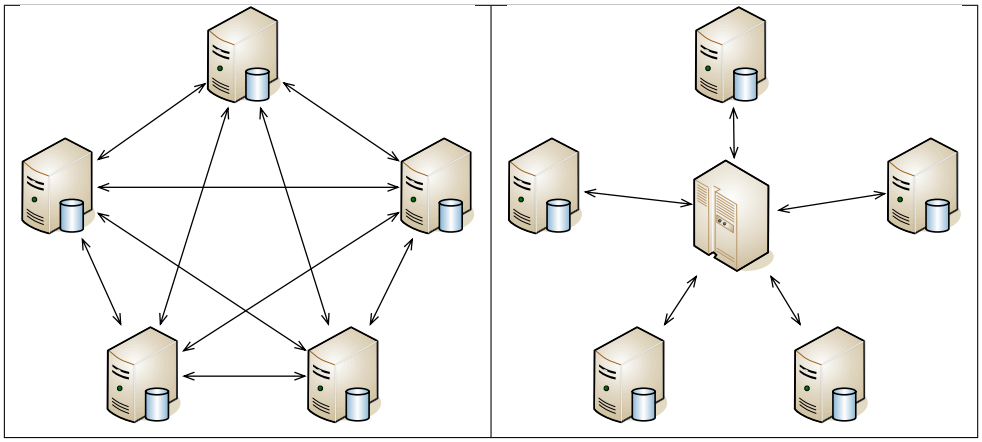


FIGURE 2.2.1.: Approches point-à-point/médiateur pour l'interopérabilité technique

Comparaison des deux types d'architecture Nous associons au critère de coût l'indicateur k_C défini comme le rapport du nombre d'interfaces des deux architrctures :

$$k_C = \frac{I_{point-to-point}}{I_{Mediator}} = \frac{n-1}{2}$$

Nous faisons l'hypothèse, en première approximation, que le CGP des interfaces croît linéairement avec leur nombre total. Dans ce cas, si $C_{point-to-point}$ est le CGP des interfaces dans l'architecture point-à-point et $C_{Mediator}$ celui des interfaces de l'architecture médiateur, nous avons :

$$\frac{C_{point-to-point}}{C_{Mediator}} \approxeq k_C \approxeq \frac{n-1}{2}$$

Le CGP des interfaces de l'architecture point-à-point est donc toujours supérieur au CGP de l'architecture de type médiateur : il est de l'ordre de n fois supérieur.

Pour ce qui concerne l'agilité, nous nous mettons dans la situation où l'ensemble E_S des systèmes d'information doit accueillir un nouveau composant. Ceci correspond à une situation industrielle où l'entreprise, par exemple, doit ouvrir son système d'information à un co-traitant ou à une filiale nouvellement acquise. Dans ce cas, le nouveau système $E_S^{'}$ comporte $n+1$ systèmes. Soient $I_{point-to-point}^{'}$ et $I_{Mediator}^{'}$ le nombre d'interfaces pour chacune des deux architectures, nous avons :

$$I_{point-to-point}^{'} - I_{point-to-point} = n$$

et

$$I'_{Mediator} - I_{Mediator} = 1$$

Le rapport k_A des nombres d'interfaces nouvelles entre les deux architectures, défini tel que :

$$k_A = \frac{I'_{point-to-point} - I_{point-to-point}}{I'_{Mediator} - I_{Mediator}}$$

est donc égal à n. Nous pouvons en conclure que l'architecture de type médiateur est n fois plus agile que l'architecture point-à-point.

En conclusion de cette section, que ce soit d'un point de vue du coût global de possession des interfaces ou de l'agilité du système E_S, le choix s'oriente nettement en faveur de l'architecture médiateur pour la réalisation de systèmes interopérables.

Depuis l'article fondateur de [Wiederhold, 1992], les architectures SOA et les *Web Services* ont permis l'éclosion du concept de *médiateur orienté services*. Nous présentons par conséquent la notion d'architecture orientée service dans la section 2.2.2 puis le médiateur orienté services dans la section 2.2.3.

2.2.2. Architectures orientées services

D'après [Booth et al., 2004], les *Web services constituent un moyen permettant de faire interopérer différentes applications informatiques, indépendamment de la plateforme ou de l'environnement dans lequel ils sont exécutés.* La tendance observée des grands éditeurs de logiciels PLM [2] à proposer aujourd'hui des solutions orientées service nous invite à étudier les notions de *service* et d'*architecture orientée services*, pour déterminer dans quelle mesure elles sont pertinentes par rapport à notre étude.

2.2.2.1. Services *web*

Les définitions de la notion de service que l'on rencontre dans la littérature font souvent référence à des solutions techniques liées aux sciences de l'information. Par exemple, [Booth et al., 2004] définissent un service web comme étant *un système logiciel conçu pour l'interopérabilité des systèmes, qui permet l'interaction au travers d'un réseau.* Pour [Massuthe et al., 2005], un service peut être vu comme un *artefact constitué d'un*

2. ARAS Innovator, Dassault, PTC, Siemens, Oracle

identifiant (id), une interface et un contrôle interne (par exemple un workflow modélisant un processus). La notion de service ayant été justement conçue pour être indépendante de toute solution technique, nous proposons donc de revenir aux bases de l'analyse fonctionnelle, telle que définie dans la norme [NF-EN-1325-1, 1996], pour asseoir notre réflexion. D'après cette norme, une *fonction de service* est *l'action demandée à un produit ou réalisée par lui, afin de satisfaire une partie du besoin d'un utilisateur donné*. En identifiant le produit au premier système (fournisseur de services) et l'utilisateur au second système (le client), cette définition devient donc :

Définition 17. Un service, du point de vue des systèmes d'information, est l'action demandée à un système (fournisseur de services) ou réalisée par lui, afin de satisfaire une partie du besoin du système client.

Définition 18. Par extension, un service web est un service distant fourni au travers d'un réseau exploitant les technologies spécifiées par le World Wide Web Consortium (W3C [3]).

Précisons qu'au regard de la norme, la notion de *fonction de service* fait partie des outils de l'analyse fonctionnelle externe : les solutions techniques mises en œuvre par le système fournisseur (le produit) sont transparentes pour le système client. Les fonctions techniques permettant de réaliser cette fonction de service seront vues en détail dans la section 4.1.2.

2.2.2.2. Architectures Orientées Services (SOA)

D'après [Gottschalk, 2000], les *architectures SOA offrent un potentiel prometteur et sont d'influence grandissante dans le cadre des architectures logicielles*. On peut considérer une telle architecture comme un ensemble de services pouvant communiquer entre eux. [Massuthe et al., 2005] distinguent les fonctions des trois acteurs de cette architecture :

- un fournisseur de service (*service provider*) : *il publie les informations relatives aux services disponibles dans un dépôt* (*repository*) ;
- un demandeur de service (*service requester*) : *interroge le fournisseur de services pour répondre à son besoin* ;
- un courtier ou intermédiaire (*broker*) de services : *permet au demandeur de services de trouver le service adéquat*.

[Massuthe et al., 2005] expliquent également le processus qui pilote les interactions entre ces trois acteurs : en fonction d'une requête du demandeur sous forme d'un service R, la tâche du courtier est de sélectionner parmi tous les services disponibles seulement les services P susceptibles d'intéresser le demandeur. Cette publication est appelée *couche publique* du SI ou *couche blanche*. Les deux services R et P ne doivent pas s'interbloquer,

3. http ://www.w3c.org

ce qui peut se produire si R et P attendent mutuellement des informations l'un de l'autre ou envoient des messages imprévus.

Pour illustrer ces concepts, nous pouvons établir une analogie avec cette situation de la vie quotidienne : un client entre dans un restaurant pour prendre son déjeuner. Nous faisons l'analogie entre :

- le SI fournisseur et le restaurant ;
- le SI demandeur et le client ;
- le courtier et le serveur.

Le restaurant liste les services disponibles sur son menu, qui est public (analogue à la couche publique du SI). Le client du restaurant exprime son besoin au serveur. Le serveur se charge ensuite d'aller en cuisine et revenir avec le plat correspondant à la demande du client. Le client n'a accès qu'aux parties publiques du restaurant, au menu, et au serveur (le *requester* accède à la *couche publique du provider* et au *broker*). Le client du restaurant n'a pas à se soucier de l'organisation de la cuisine ni qui est aux fourneaux. Par ailleurs, il n'est pas autorisé à se rendre en cuisine (le *requester* n'est pas autorisé à accéder aux fonctions internes du *provider*).

La séparation stricte entre services/fonctions internes et couche publique/couche privée des SI est connue sous le nom de *couplage faible spécifications/implémentations* : le service est indépendant des solutions techniques utilisées (langages de programmation, bases de données, types de machines etc.). Dans le cas de la restauration, ceci peut s'illustrer par le fait que la cuisine peut changer d'organisation ou les cuisiniers être remplacés sans que le service rendu n'en soit affecté et que le client ne s'en aperçoive.

Pour assurer l'échange d'informations, il faudra cependant veiller à ce que le *requester* et le *provider/broker* utilisent les même protocoles de communication (le client du restaurant et le serveur doivent parler la même langue pour se comprendre et faire en sorte que le client obtienne son plat).

Dans la section suivante, nous présentons la synthèse de la topologie en étoile de type médiateur et des architectures de type SOA.

2.2.3. Le médiateur d'informations orienté services

[Bénaben et al., 2006] définissent les trois fonctions principales suivantes pour l'interopérabilité :

- conversion et mise à disposition des données ;
- gestion des applications ;
- orchestration des processus collaboratifs.

A cet effet, [Bénaben, 2008] propose une architecture de médiateur d'informations orienté services qui répond à chacune de ces fonctions (voir figure 2.2.2). Dans le cas où tous les systèmes à faire interopérer sont orientés services alors le médiateur proposé, lui-même fournisseur et demandeur de services, sera à même de réaliser l'interopérabilité entre tous ces SI. Il est à noter que la fonction *conversion des données* qui apparaît dans le SI médiateur est reliée à la notion d'interopérabilité sémantique que nous verrons dans la section 2.3.

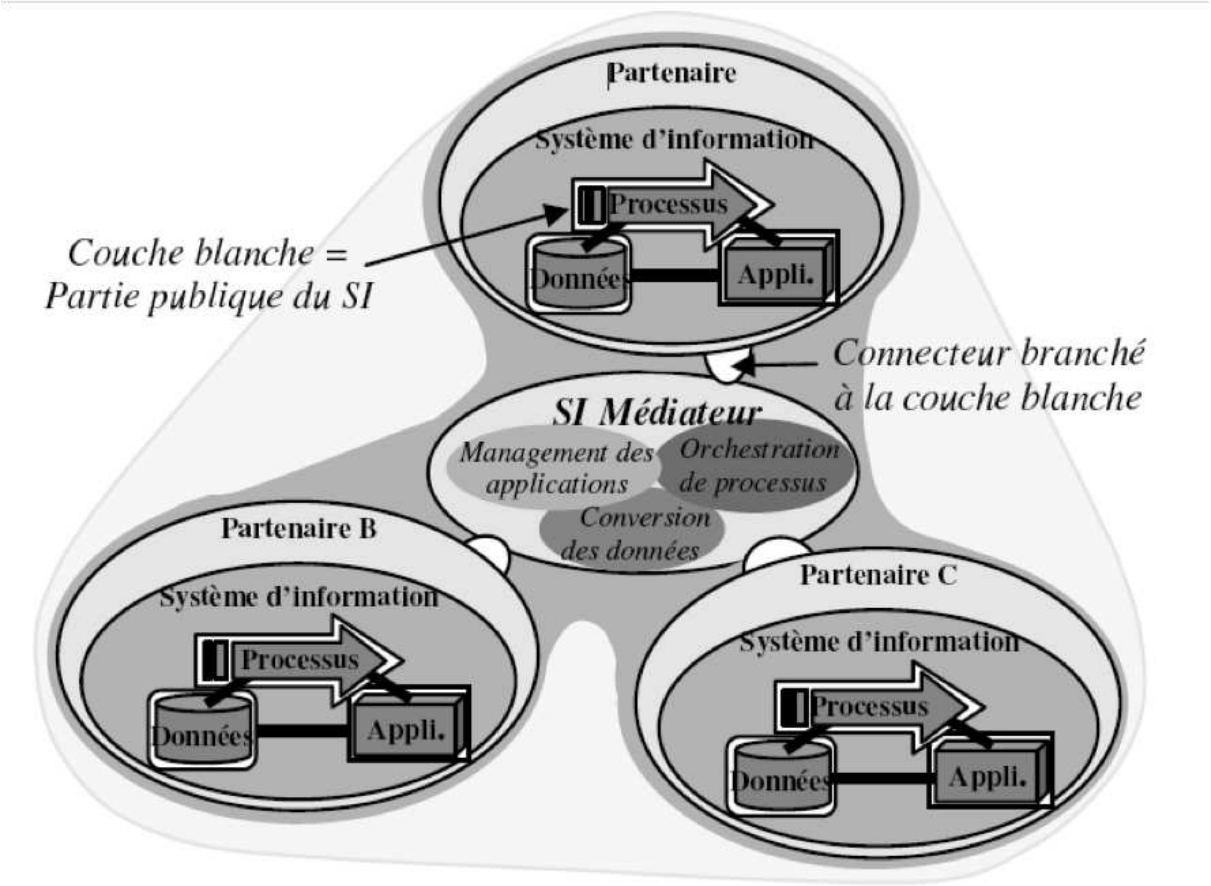


FIGURE 2.2.2.: Le médiateur d'informations orienté services selon [Bénaben et al., 2008]

2.2.4. Médiation multi-échelle orientée services

Nous adaptons ce modèle médiateur, dont nous avons vu qu'il est pertinent pour les problèmes d'interopérabilité, à l'aspect multi-échelle du système PLM et de son interopérabilite (voir figure 1.3.1). Dans cette figure, nous avons insisté sur l'aspect multi-échelle de l'interopérabilité dans le domaine du PLM, et nous avons montré qu'il existe deux types de problèmes à résoudre : inter-phase et intra-phase. Pour répondre à ce double défi de manière globale et cohérente, nous proposons une architecture basée sur deux types de médiateurs fournisseurs et consommateur de services web :

- des médiateurs intra-phase (représentés par des bulles jaunes) pour chacune des phases du cycle de vie du produit ;
- un médiateur inter-phase (représenté par une bulle violette) pour assurer l'interopérabilité entre les SI de plusieurs phases du cycle de vie du produit ;

Chacun de ces médiateurs sera situé à équidistance sémantique des applications à faire interopérer. Ce point de vue est représenté sur la figure 2.2.3. Ainsi, si n phases du cycle de vie sont à considérer, $n+1$ médiateurs seront nécessaires pour garantir l'interopérabilité dans le système : n médiateurs intra-phase et 1 médiateur inter-phase. Nous dénommons ce type d'architecture, destinée à résoudre des problèmes d'interopérabilité multi-échelle, comme de la *médiation multi-échelle*.

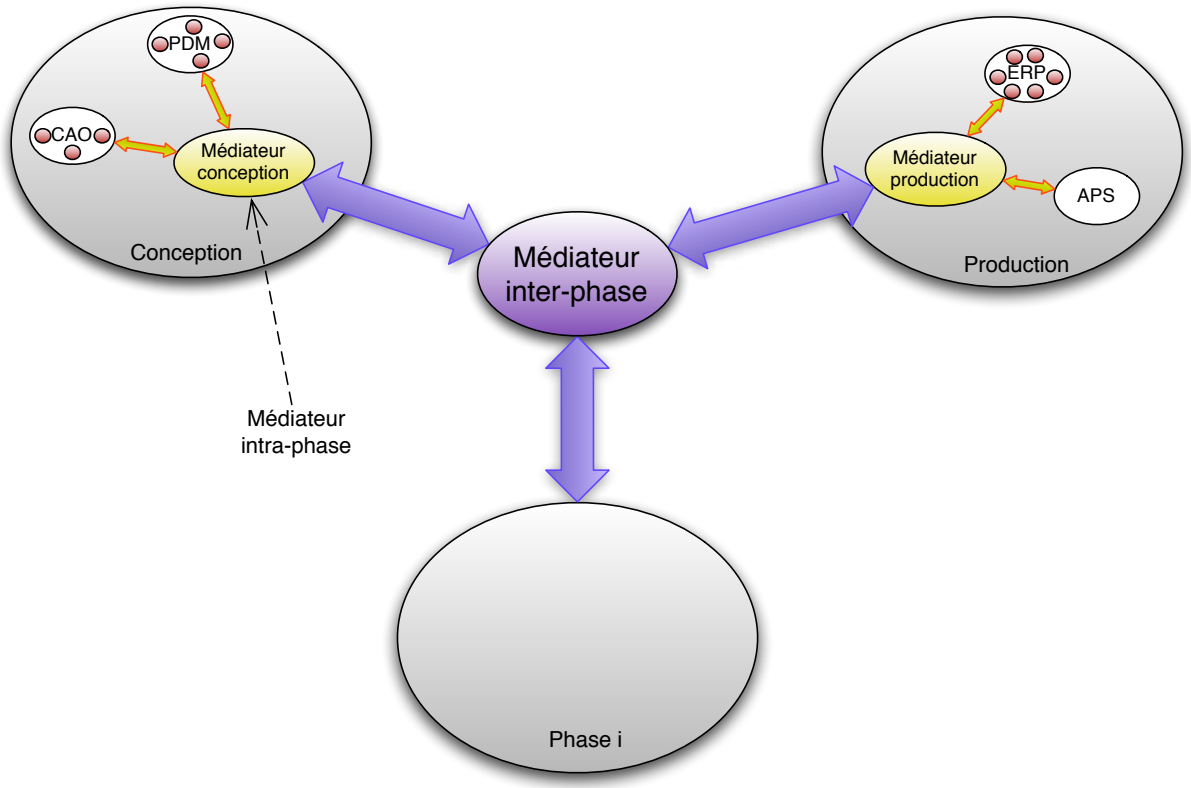


FIGURE 2.2.3.: Médiation multi-échelle

D'un point de vue fonctionnel, afin d'assurer la continuité du flux d'information (donc du flux sémantique), chaque médiateur devra être capable de :

- identifier l'origine du flux d'information ;
- identifier le destinataire de ce flux ;
- ventiler vers les différents sous-médiateurs la part de ce flux qui lui est destinée.

Nous considérons cette architecture comme cohérente puisqu'elle est indépendante de l'échelle à laquelle on observe le système : que le point de vue d'observation se situe à un niveau global (à l'échelle du système PLM entier) ou local (à l'échelle d'une phase du cycle de vie voire d'une application fortement modulaire), l'observateur voit un médiateur qui connecte un ensemble de systèmes SOA. A cet égard, **cette médiation multi-échelle présente des auto-similarités fonctionnelles** : quelle que soit l'échelle considérée, chaque médiateur agit comme un *répartiteur sémantique* (pour faire une analogie avec la circulation du flux d'énergie dans un réseau électrique). Cette fonction de répartition du médiateur i est représentée sur la figure 2.2.4 :

- le médiateur i peut être le médiateur inter-phase ou n'importe lequel des médiateurs intra-phase ;
- chaque système peut être un médiateur intra-phase ou une application orientée services.

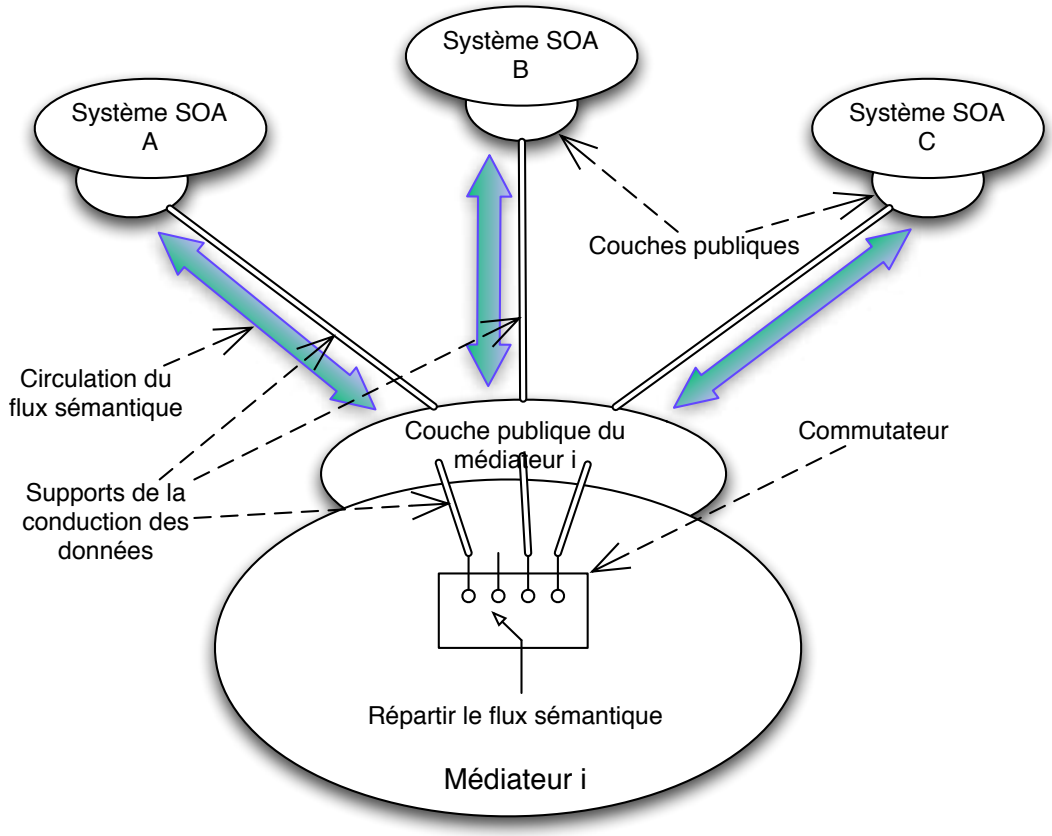


FIGURE 2.2.4.: Fonction de répartition d'un médiateur de la médiation multi-échelle

Sur cette figure, il est entendu que l'aspect technique de l'interopérabilité concerne ici simplement la *conduction* du flux sémantique : ce sont les lignes doubles qui véhiculent le flux sémantique. En revanche, la commande du commutateur représenté sur la figure 2.2.4 sera à définir à un niveau sémantique, puisque c'est cet élément qui sera chargé d'adresser au bon système la bonne information. A ce stade de la définition de l'architecture, nous

assurons donc la continuité d'un flux de données qui ne deviendront des informations qu'une fois l'étude sémantique achevée.

Cette proposition d'architecture est conforme aux prescriptions de [Vernadat, 2007], pour qui *l'agilité requiert des systèmes d'informations d'entreprise interopérables, i.e. des systèmes reconfigurables composés de plusieurs systèmes travaillant ensemble.*

2.2.5. Prescriptions méthodologiques pour le choix de l'architecture

En guise de conclusion à cette section, nous formulons les deux prescriptions méthodologiques suivantes :

- pour résoudre les problèmes d'interopérabilité dans le domaine du PLM, caractérisés par des besoins forts en terme d'agilité, **on veillera à mettre en place une médiation multi-échelle orientée services** telle que définie dans la section 2.2.4 ;
- si au moins un des SI impliqués n'est pas orienté services, l'hypothèse préalable permettant à [Bénaben, 2008] de construire son modèle de médiateur n'est pas vérifiée. Dans ce cas, on veillera au préalable à ajouter à ce(s) système(s) une couche publique centrée uniquement sur la fourniture des quelques services requis pour réussir l'interopérabilité. Nous nommerons cette approche *webservicisation d'applications non-orientées services.* L'implémentation de la *webservicisation* du logiciel Catia V5, définie dans le strict cadre de l'interopérabilité CAO/PDM, sera présentée dans le chapitre 4.

L'étude fonctionnelle de ces médiateurs est incomplète. Nous allons examiner, dans la section suivante, comment peut être assurée la *conservation* du flux sémantique. Toujours dans le cadre de l'analogie avec un réseau électrique, nous allons ajouter à ces médiateurs une fonction de *transformateur sémantique.* Contrairement au transport d'énergie électrique, nous ne tolérerons aucune perte dans les opérations de conversion des données évoquées par [Bénaben, 2008] (*i.e.* nous pourrons parler de manière analogue d'un *rendement* de 100% de cette transformation sémantique).

2.3. Interopérabilité sémantique

Cette section s'intéresse au niveau sémantique de l'interopérabilité tel que présenté dans le chapitre 2.1.1. Elle s'attache à faire le lien entre la conservation du flux sémantique et la médiation multi-échelle, pour compléter la spécification fonctionnelle du médiateur de la figure 2.2.4. Nous présentons tout d'abord les approches possibles quant à la mise en correspondance sémantique des modèles de données des différents SI à faire interopérer (voir section 2.3.1). Une fois l'approche pertinente sélectionnée, nous nous intéressons en particulier au méta-modèle d'unification, pour déterminer dans quelle mesure utiliser un

modèle *ad hoc* ou un modèle standard (voir section 2.3.3). Ensuite, nous proposons une revue des standards du PLM et retenons notre attention sur un protocole d'application particulier de la norme STEP : *Product Lifecycle Support* (voir section 2.3.6). Enfin, nous concluons cette section et ce chapitre en complétant le modèle de médiateur avec une fonction de conversion des données, qui sera spécifiée d'après les conclusions tirées de chacune de ces sous-parties.

2.3.1. Intégration, unification et fédération de modèles

La norme [ISO-14258-1998, 1998], relative à la modélisation des entreprises, précise que l'interopérabilité entre deux (ou plusieurs) systèmes d'information d'entreprise peut-être abordée de trois manières :

- **intégration** : un standard commun de modèle de données est utilisé pour tous les composants du système. Le processus d'intégration revient à fusionner les modèles de données ;
- **unification** : un méta-modèle commun à tous les composants du système fournit un moyen pour établir des correspondances sémantiques ;
- **fédération** : des modèles distincts sont associés dynamiquement. Cette approche s'appuie habituellement sur des outils semi-automatiques, basés sur des méthodes heuristiques qui comparent principalement la terminologie et la structure des données afin de détecter les couples de concepts qui sont reliés au niveau sémantique (similarité ou équivalence), et nommés mappings.

[Hoffmann, 2008] émet les réserves suivantes à l'encontre de ces trois approches :

- l'approche d'intégration *requiert de développer une nouvelle ontologie qui reflète un consensus sur le point de vue des différentes organisations qui collaborent, afin d'en limiter les incohérences. En raison de ces compromis, la nouvelle ontologie pourra n'avoir qu'une compatibilité limitée avec les ontologies source des différentes organisations.* D'autre part, ce qui n'est pas signalé par [Hoffmann, 2008], la fusion des modèles de données suppose de les connaître complètement, ce qui n'est souvent pas le cas lorsque le problème d'interopérabilité implique des applications propriétaires dont les modèles de données sont considérés comme des secrets industriels ;
- dans le cas de l'unification, l*a plupart des collaborations ne seront pas connues avant que les ontologies ne soient développées, cette approche requiert de modifier les ontologies en accord avec l'ontologie de plus haut niveau. Comme il peut y avoir plusieurs ontologies de plus haut niveau qui soient pertinentes, cette approche amènera à développer plusieurs versions d'ontologie, qu'il faudra garder à jour.* Cependant, cette conclusion de [Hoffmann, 2008], formulée dans le cadre d'un travail relevant du génie informatique, ne s'applique que partiellement au contexte du PLM : les collaborations sont connues *a priori* (les collaborations conception/production ont été présentées dans le chapitre 1) ;
- *les approches fédératives permettent l'échange parmi des ressources développées dans des buts indépendants et qui évoluent de manière indépendante ; elles semblent donc*

les plus appropriées. Cependant malgré tous les efforts investis dans les méthodes et outils pour l'alignement d'ontologies, les résultats sont encore décevants.

Le choix de l'une ou l'autre de ces approches est critique quant à la conservation du flux sémantique dans un contexte de mise à l'échelle. Compte tenu de cette analyse, nous proposons dans la section suivante un ensemble de prescriptions méthodologiques quant à l'approche à adopter dans notre cas.

2.3.2. Prescriptions méthodologiques pour la mise en correspondance sémantique des modèles de données

Dans le droit fil de notre objectif de robustesse et d'agilité vis-à-vis d'une mise à l'échelle sémantique et temporelle :

- nous déconseillons une approche d'intégration, qui conduit à une interopérabilité limitée dans les espaces temporels et sémantiques ;
- face aux efforts encore nécessaires pour mettre au point la fédération des modèles de données, nous déconseillons cette approche si l'industrialisation de la solution proposée est envisagée à court terme ;
- nous préconisons une approche d'unification des modèles de données.

Le choix ou la définition du méta-modèle d'unification (les *ontologies de plus haut niveau* de [Hoffmann, 2008]) constitue l'étape suivante de notre méthodologie. La présentation des travaux liés à l'interopérabilité dans le PLM a fait apparaître deux cas de figure : les auteurs définissent un modèle de données *ad hoc* ou utilisent et implémentent un modèle de données standard. L'objet de la section suivante est donc de déterminer dans quelle mesure il convient de choisir l'une plutôt que l'autre de ces approches.

2.3.3. Méta-modèle d'unification *ad hoc* / standard

A propos du terme « standard » Le terme « standard », abondamment utilisé dans le domaine du PLM et dans ce mémoire, peut revêtir plusieurs sens :

- « standard » est le terme anglais pour désigner une norme industrielle. Par exemple, l'*International Standard Organisation* peut être traduit par Association Internationale de Normalisation. En France, c'est l'AFNOR qui est chargé de l'édition des prescriptions techniques ou normes françaises. Par abus de langage, le terme français « standard » est utilisé dans ce sens ;
- un « standard » peut également être entendu comme une règle fixe à l'intérieur d'une entreprise pour caractériser un produit, une méthode de travail etc. Dans ce cas, le standard d'entreprise n'est pas nécessairement une norme, c'est-à-dire qu'il n'a pas fait l'objet d'un consensus au sein d'un groupe d'experts, selon un processus encadré par l'AFNOR ou l'ISO, pas plus qu'il n'a été rendu public ;

- « standard » peut aussi être utilisé comme adjectif pour qualifier une donnée technique. Par exemple, on parlera de « composant standard ». Employé dans ce cas, « standard » représente la caractéristique d'un élément de l'entreprise ayant vocation à être diffusé et utilisé tel quel.

Dans cette section, le *modèle standard* désignera un méta-modèle d'unification normalisé, c'est-à-dire ayant fait l'objet d'une publication par un organisme de normalisation à la suite d'un processus ayant abouti à un consensus entre experts. Ces modèles de données intègrent donc une sémantique « prête à l'emploi ».

Précisions sur la locution *ad hoc* Nous parlerons de modèle de données *ad hoc* lorsqu'il s'agira de créer un *nouveau* modèle de données à même de servir de méta-modèle d'unification pour un problème d'interopérabilité précis. Ce nouveau modèle de données devra intégrer toute la sémantique partagée par les SI concernés, travail qui sera à la charge des personnes chargées de spécifier et/ou implémenter ce modèle.

Comparaison des modèles *ad hoc* / standards comme candidats au rôle de méta-modèle d'unification Nous listons quelques caractéristiques de chacune de ces approches :

- l'utilisation de modèles standards est source d'une importante économie pour les entreprises industrielles qui les mettent en œuvre. Un rapport du *National Institute of Standards and Technology* [Gallagher et al., 2002] évalue les économies à plus d'un milliard de dollars par an dans le cas d'une adoption massive du standard STEP par les industries aéronautiques, automobiles et navales. Ces économies concernent la réduction conséquente des coûts d'interopérabilité des SI impliqués (coûts de migration, de maintenance, de temps) ;
- un modèle standard propose une sémantique riche, fruit du long travail collaboratif de nombreux experts. Cet aspect donne au standard une crédibilité industrielle en même temps qu'il dispense l'utilisateur de ce standard de la modélisation sémantique ;
- par contre, le modèle standard peut contenir une sémantique figée dans le temps, qui n'est donc pas pertinente pour une mise-à-l'échelle ;
- utiliser un modèle standard requiert un effort de la part de l'utilisateur pour s'approprier les concepts et les méthodes d'implémentation ;
- le modèle *ad hoc* offre une grande liberté à son instigateur. Il n'est contraint par aucune norme, dont les processus peuvent parfois être lourds à mettre en oeuvre ;
- en revanche, le modèle *ad hoc* est par définition vierge de toute sémantique. Il conviendra donc d'alimenter ce modèle d'une sémantique cohérente qui couvre le domaine considéré.

Les points précédents ne peuvent pas être considérés comme des points forts ou des points faibles d'une manière absolue. En conséquence, c'est dans le contexte spécifique qui fait l'objet de notre étude que nous formulons les prescriptions méthodologiques suivantes :

- on choisira **prioritairement un modèle d'unification standard et extensible** seul capable de garantir la mise-à-l'échelle temporelle et sémantique dans un contexte industriel. Nous rappelons qu'aussi bien [Chen and Vernadat, 2004] que [Lee and Jeong, 2006] précisent que *l'interopérabilité requiert l'utilisation des standards pour être capable de partager et échanger des informations au sein de SI et d'organisations hétérogènes et distribuées* ;
- le **cas échéant, on définira un nouveau modèle *ad hoc***. Il conviendra alors de s'entourer d'une large communauté à même de spécifier la sémantique partagée qui sera contenue dans ce modèle. Sans cette précaution, le risque est fort de créer une coquille qui restera sinon vide, du moins partiellement remplie.

Nous pouvons dès lors compléter, en termes fonctionnels, la figure 2.2.4 que nous avons présentée et dans laquelle nous avons fait apparaître la *répartition* du flux de données. Le médiateur complet est représenté sur la figure 2.3.1.

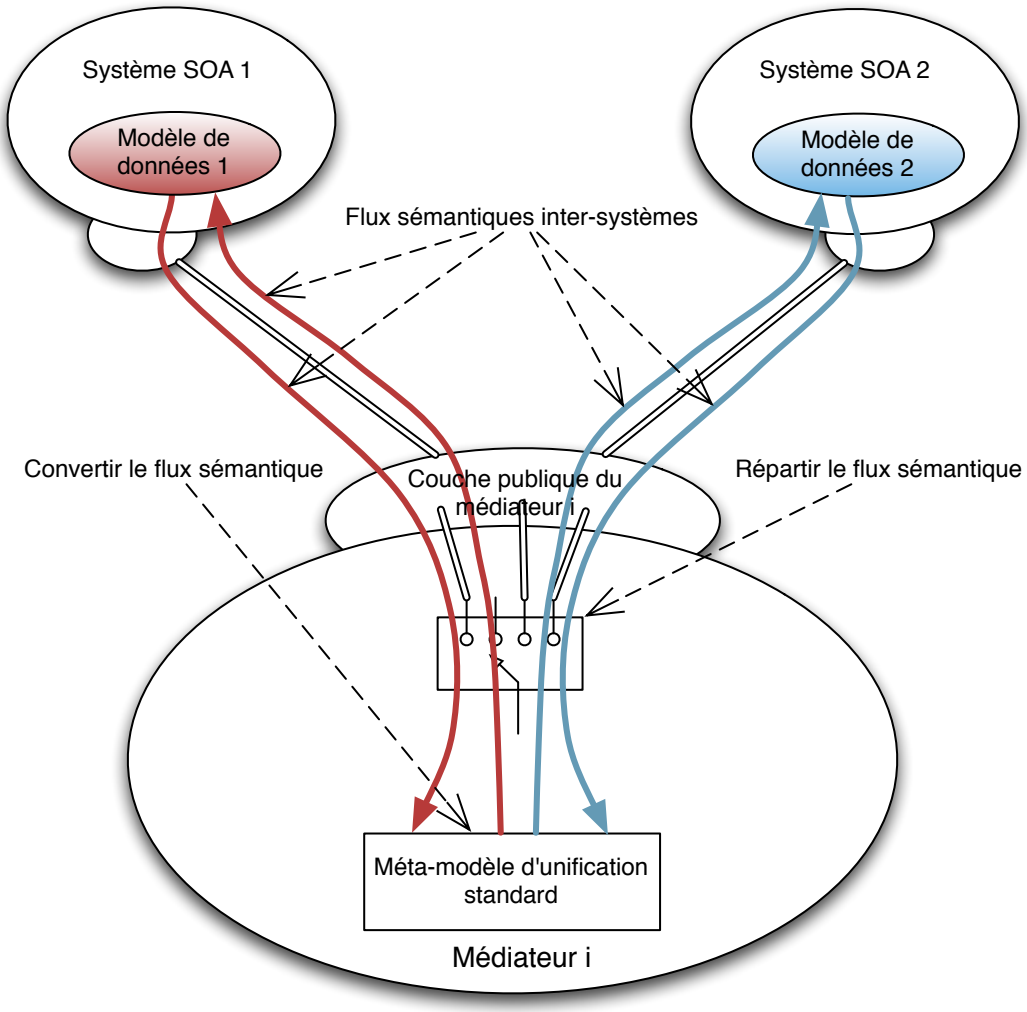


FIGURE 2.3.1.: Le médiateur multi-échelles orienté services - Unification des informations sémantiques

La conversion sémantique est donc assurée par l'élément dans lequel apparaît le label « Méta-modèle d'unification standard ». C'est ce méta-modèle qui permet au médiateur de :

- diriger les flux sémantiques entrant et sortant du médiateur et donc piloter la fonction répartition ;
- réconcilier la sémantique provenant de SI différents, ces SI pouvant être soit des applications, soit d'autres médiateurs.

La démarche conduisant au choix (ou à l'absence de choix) de ce modèle d'unification standard fait l'objet de la section suivante relative aux standards du PLM.

2.3.4. Les standards du PLM

Nous brossons dans cette section un panorama des standards à l'œuvre dans le domaine du PLM. Les états de l'art réalisés par [Chen and Vernadat, 2004] et [Rachuri et al., 2008], récents et complets, offrent une matière première qu'il n'est pas question ici de reprendre *in extenso*. Nous nous attachons plutôt à chercher dans quelle mesure leurs résultats peuvent être confrontés au contexte de notre étude. Dans ce sens, c'est le travail de [Rachuri et al., 2008], centré sur le domaine du PLM, qui retient notre attention et qui sera plusieurs fois cité dans les lignes qui suivent : il fait apparaître en filigrane, sur une cartographie 2D des standards du PLM, la notion de distance sémantique. Cette cartographie est un élément essentiel de notre processus de choix du méta-modèle d'unification. Nous présentons donc cette cartographie dans un premier temps. Dans un second temps, nous exprimons quelques désaccords avec ce point de vue, et proposons de le préciser et le compléter. Nous pourrons alors conclure quant aux préconisations méthodologiques permettant de choisir le ou les standards pertinents pour le méta-modèle d'unification au cœur de la médiation multi-échelle orientée services.

Typologie des standards [Rachuri et al., 2008] proposent une typologie hiérarchique à quatre niveaux des standard du PLM :

- **Type 0** : standards pour les langages d'implémentation ;
- **Type 1** : standards pour la modélisation des informations (par exemple EXPRESS, UML ou XML) ;
- **Type 2** : standards relatifs au contenu, au domaine sémantique : modélisation des informations relatives au produit (STEP), standards d'échange de données, visualisation des représentations du produit, standards relatifs à la sécurité informatique ;
- **Type 3** : standards concernant l'architecture des environnements de travail.

On constate un lien fort avec les niveaux d'interopérabilité déjà présentés. Les standards de type 0 concernent l'interopérabilité technique : ils seront mis en œuvre dans le chapitre 4 dédié à l'implémentation de l'architecture. Ce sont en revanche les standards du PLM relatifs à la sémantique du produit (donc des types 1 et 2) qui intéressent directement

cette section. La nuance entre les catégories 1 et 2 est très légère : par exemple, le langage de modélisation EXPRESS (présenté et utilisé dans le chapitre 3) est classé dans la catégorie 1 alors qu'il est partie intégrante de la norme STEP, classée elle dans la catégorie 2. C'est pour cette raison que nous avons associé, dans le tableau 2.3.1, les standards de type 1 et 2 au niveau sémantique de l'interopérabilité.

Niveau d'interopérabilité	Classe de standard ([Rachuri et al., 2008])
Technique	0
Sémantique	1 et 2
Organisationnelle	3

TABLE 2.3.1.: Correspondance bi-univoque entre les niveaux d'interopérabilité et les types de standard

Une fois cette classification effectuée, [Rachuri et al., 2008] dressent l'inventaire de tous les standards de type 2 et les font figurer sur une cartographie en deux dimensions (voir figure 2.3.2).

Cartographie 2D des standards du PLM Sur cette carte, les auteurs ont fait figurer les standards de type 2 suivant une double sémantique :

- sur l'axe des abscisses, les phases du cycle de vie du produit. Ces phases sont représentés suivant un point de vue temporel, du passé vers le futur ;
- sur l'axe des ordonnées, une sémantique suivant le point de vue du produit, des processus ou des services de l'entreprise (PPE).

Chaque ellipse représente donc la couverture sémantique d'un standard de type 2 dans ce référentiel ($phases$, PPE). L'analyse de cette cartographie peut dans un premier temps laisser pessimiste quant au choix du méta-modèle d'unification : il n'existe *a priori* aucun standard permettant de couvrir l'intégralité du cycle de vie du produit. De nombreuses zones blanches semblent hors de portée sémantique de tout standard. Le standard STEP quant à lui, dont on aurait pu penser qu'il fût le meilleur candidat pour unifier les données produits, apparaît éclaté et insuffisant en terme de couverture sémantique. Par conséquent, la médiation inter-phase ne serait pas possible *via* les standards, obligeant à la définition d'un modèle d'unification *ad hoc*.

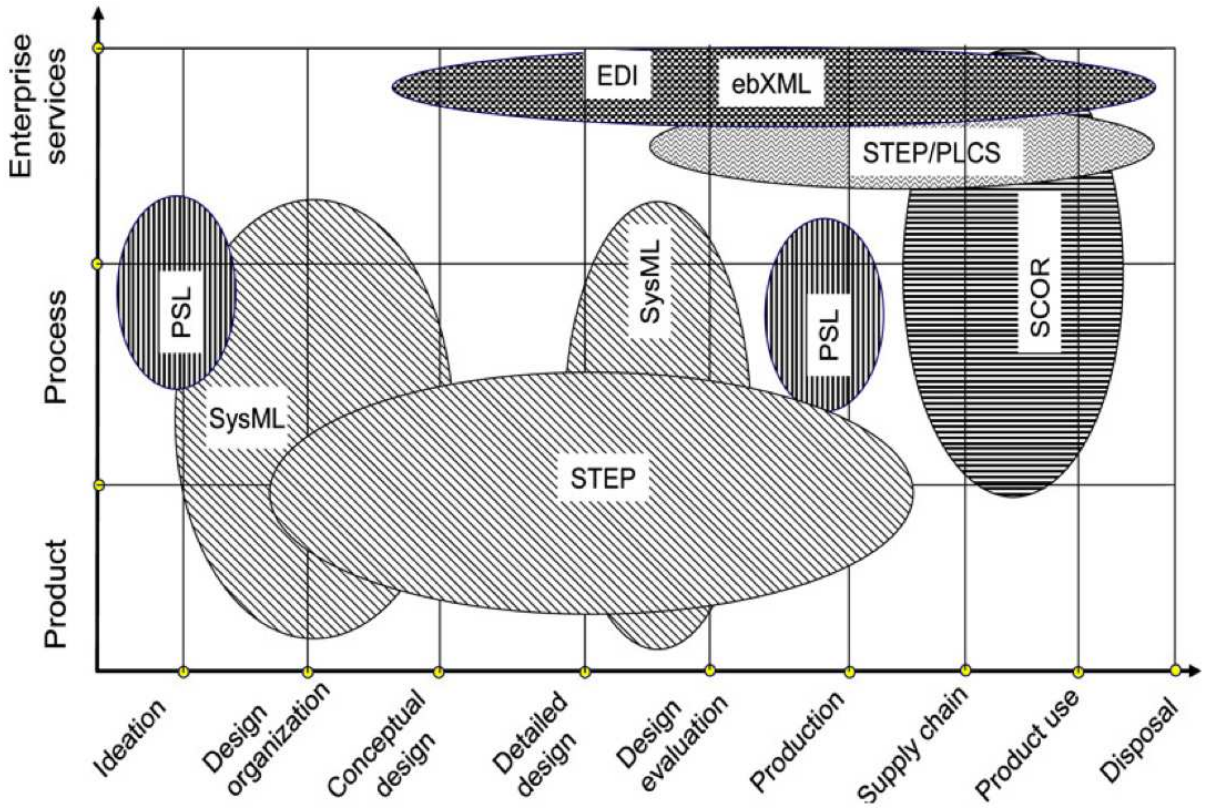


FIGURE 2.3.2.: Cartographie 2D des standards du PLM [Rachuri et al., 2008]

Nous pensons néanmoins que les conclusions issues de cette cartographie doivent être tirées après que les imprécisions suivantes ont été levées :

- les bulles grisées apparaissent comme *statiques*, *i.e.* le champ sémantique couvert par chaque standard est figé d'un point de vue temporel et sémantique. Nous proposons de montrer que cette cartographie ne tient pas compte de l'évolution du standard STEP vers des modèles de données génériques et extensibles ;
- la bulle correspondant au standard STEP apparaît scindée en deux ellipses : une de taille importante (portant la mention STEP) située au niveau *produit* et une autre de taille moindre (portant l'étiquette STEP/PLCS) située au niveau *entreprise* . Nous proposons de montrer que ce point de vue est à nuancer, PLCS étant un sous-domaine d'une norme STEP elle-même fortement modulaire mais connexe ;
- enfin, la norme STEP ne semble pas pertinente pour décrire le produit dans des phases précoces de son cycle de vie. Nous montrerons que ce point de vue est à réexaminer au vu des travaux en cours relatifs à l'intégration de l'ingénierie système dans la norme STEP.

Les deux sections suivantes visent à apporter des arguments à chacun de ces trois points. Dans la section 2.3.5, nous présentons la norme STEP. Cette norme a déjà fait l'objet de nombreuses présentations dans la littérature scientifique, mais elle est au cœur de

notre travail et nous considérerions ce mémoire comme incomplet si une section n'y était consacrée. Nous nous attacherons cependant à inscrire cette présentation dans l'histoire et l'évolution de la norme pour montrer sa tendance à la modularité et à l'extensibilité des modèles de données qu'elle définit. Dans la section 2.3.6, nous présentons un protocole d'application particulier de cette norme, PLCS.

2.3.5. La norme ISO-10303 STEP

STandard for the Exchange of Product model data (STEP) vise à fournir un mécanisme permettant la description complète et univoque du produit, couvrant l'intégralité de son cycle de vie, indépendamment de tout système informatique. L'*International Standard Organization*[4], organisme international de normalisation, fixe un cadre normatif définissant ce mécanisme, repéré par le n°10303. Le comité technique TC 184/SC4[5] assure le pilotage de la rédaction et de la publication des différentes parties de cette norme. Les sections qui suivent ont pour objectif de présenter l'historique, l'architecture et les évolutions actuelles de cette norme, pour situer le contexte de la section 2.3.6 relative au protocole d'application Product Lifecycle Support.

Origines Les premiers travaux du comité technique TC 184/SC4 remontent à l'année 1984. A cette époque, la norme alors utilisée pour l'échange de données CAO, *Initial Graphics Exchange Specification* (IGES[6]), se heurte à des obstacles qui nécessitent un autre cadre de travail. Ces premiers travaux sont consacrés en 1994 par la publication d'une norme dédiée principalement (mais pas seulement) à l'échange des fichiers CAO 3D, connue sous le nom d'ISO-10303 :203 [ISO-10303-203, 1994]. Basée sur une description *Boundary Representation* (B-Rep) de la topologie des solides, cette norme a connu un grand succès et s'est progressivement imposée face à IGES : la totalité des éditeurs de logiciels CAO 3D proposent une fonctionnalité d'import/export de fichiers au format STEP203, et il est le standard de référence utilisé pour la communication de modèles CAO 3D dans des relations donneur d'ordre/sous-traitant.

Objectifs Pour répondre aux exigences formulées dans les objectifs généraux, la norme STEP contient les spécifications suivantes [Fowler, 1995] :

- elle définit un format neutre, interprétable par tout système indépendamment du système ayant généré les données ;
- elle permet l'archivage à long terme des données et l'intégration des applications autour de bases de données interopérables ;

4. *http ://www.iso.org/*
5. http ://www.tc184-sc4.org/
6. http ://www.iges5x.org/

- elle couvre un très vaste domaine de connaissance correspondant à l'ensemble des catégories de produits (pièces élémentaires, assemblages, mécanismes, etc.), selon le point de vue de tous les métiers (mécanique, électronique, etc.), et à toutes les phases du cycle de vie (conception, calcul, fabrication, maintenance, démantèlement, etc.) ;
- elle est évolutive, pour prendre en compte toutes les spécificités d'un nouveau domaine d'application ;
- enfin, elle définit des méthodes de test de conformité des données STEP générées.

Architecture L'architecture de STEP respecte celle des Systèmes de Gestion de Base de Données de la norme ANSI/SPARC. Celle-ci est décomposée en trois niveaux [Chambolle, 1999] :
- applicatif : prise en charge de la partie applicative (protocoles d'application) ;
- logique : prise en charge des structures de données génériques (ressources intégrées) ;
- physique : traitement des méthodes d'implémentation (encodage des fichiers physiques et base de données).

2.3.5.1. Évolutions passées, en-cours et futures de la norme STEP

L'histoire de l'évolution de la norme STEP au cours du temps a été marquée par des événements importants :
- certains protocoles d'application ont été mis à jour : en 2005 a été publiée la version 2 du protocole AP203, nommé AP203ed2 [ISO-10303-203ed2, 2005], qui ajoute des informations concernant les géométries paramétrées ou encore une méthode de description procédurale similaire à du CSG ;
- de nombreux protocoles d'application ont été régulièrement publiés : plus de 40 protocoles d'application ont vu le jour [7]. Les principaux domaines pris en compte par STEP concernent la conception, la fabrication, l'électronique, la construction navale, l'architecture et le génie civil ;
- cet accroissement a conduit à des redondances dans les modèles de données entre protocoles d'application ;
- dans le même temps, des griefs relatifs à l'aspect monolithique et figé des protocoles d'application ont été émis à l'encontre de STEP, freinant par là même son adoption par les industriels.

Pour répondre aux problèmes posés par les deux derniers points, la norme STEP a évolué ces dernières années dans le sens :
- d'une plus grande modularité de la norme ;
- de l'apparition de protocoles d'application présentant un modèle de données à la granularité plus grosse, mais permettant en contrepartie une spécialisation en fonction du domaine considéré.

7. Step On A Page : http ://www.mel.nist.gov/sc5/soap/

Le PDM Schema La première tentative de standardisation s'est matérialisée par la publication du schéma intitulé PDM Schema [Machner and Ungerer, 1998], défini à l'intention de la gestion de données techniques (Product Data Management - PDM). Le PDM Schema s'appuie sur quatre protocoles d'application dont les recoupements fonctionnels ont été synthétisés (cf Figure 2.3.3).

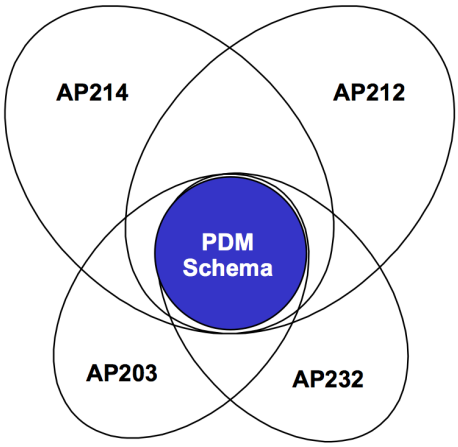


FIGURE 2.3.3.: Périmètre fonctionnel du PDM Schema

Les modules d'application : interopérabilité des protocoles d'application STEP Afin que les applications informatiques basées sur la norme STEP puissent interopérer, les protocoles d'application doivent eux-même être interopérables [Ball et al., 2008]. D'après ces auteurs, le processus de rédaction des protocoles d'application, tel qu'il a été défini dès les origines de STEP, pose deux problèmes :

- les protocoles d'application sont monolithiques et couvrent chacun une large étendue fonctionnelle (l'AP203 et l'AP214 sont des exemples de tels AP) ;
- l'absence d'un pilotage d'ensemble de la norme STEP conduit à des recouvrements : certaines informations sont modélisées dans plusieurs protocoles d'application et sont donc redondantes (il existe par exemple des redondances entre les protocoles AP203 et AP214).

Fort de ce diagnostic, l'ISO a engagé alors un travail de *modularisation* de la norme STEP [Feeney, 2002] : un ensemble de modules appelés *Application Modules* (AM) a été défini à partir des protocoles d'application (AP) existants, chacun de ces AM pouvant être appelé par d'autres AM. Le processus de construction d'un nouvel AP tel qu'édicté par l'ISO repose dorénavant sur cette base modulaire. Ceci est représenté de manière synthétique sur la figure 2.3.4.

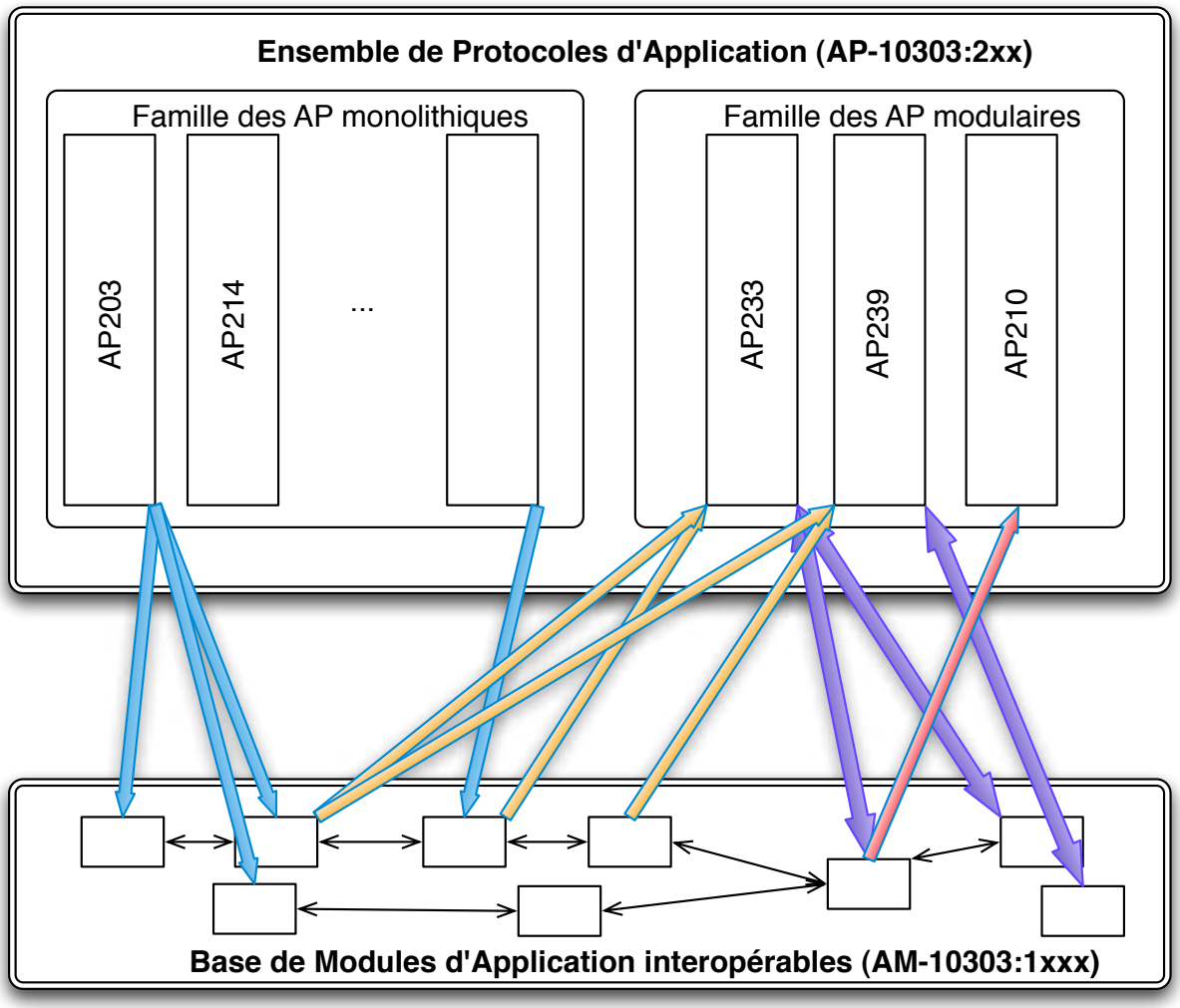


FIGURE 2.3.4.: La modularité de la norme STEP : *Modules d'Application* (AM) et *Protocoles d'Application* (AP)

Sur cette figure :

- la « famille des AP monolithiques » représente l'ensemble des AP « historiques », c'est-à-dire ceux qui ont été publiés avant que la modularisation de STEP ait été effectuée (par exemple l'AP203 ou l'AP214) ;
- la sémantique de chacun de ces AP a été scindée en plusieurs AM (les flèches bleues représentent ce processus de construction des modules à partir des AP), chaque AM étant identifié de manière unique dans la norme (par exemple AM1040, AM1021 etc) ;
- la base de modules est représentée dans le rectangle « Base d'AM interopérables ». L'interopérabilité de ces modules est symbolisée par des doubles-flèches noires en trait continu fin ;
- après que la modularisation des AP a été réalisée, l'ensemble de ces modules ont été utilisés pour créer de nouveaux AP modulaires (figurés dans la case « famille des AP

modulaires »). Ce processus de construction est représenté par des flèches de couleur orange ;
- dans le même temps, la définition de ces nouveaux AP a nécessité la création de nouveaux AM. Les doubles-flèches violettes représentent ce processus : dans le sens AP vers AM, elles représentent le fait que les modules ont été créés dans le cadre du processus de construction de l'AP ; le sens AM vers AP symbolise le fait que ces nouveaux AP reposent sur les AM nouvellement créés ;
- ces nouveaux AM complètent la base modulaire et pourront donc à leur tour servir lors de la définition ultérieure d'un autre AP modulaire (flèche rouge).

En conclusion, chaque AP modulaire repose donc sur une base de modules interopérables provenant de la sémantique des AP historiques (flèches oranges), d'autres AP modulaires (flèches rouges) et du domaine spécifique visé (flèches violettes).

Les évolutions les plus notables de ces dernières années concernent donc la publication des protocoles d'application modulaires. La section suivante présente un de ces protocoles d'application, l'AP239 [ISO-10303-239, 2005], avec pour objectif de préciser son champ sémantique (dont on verra qu'il sera particulièrement adapté pour l'interopérabilité PDM/ERP) et présenter son caractère générique et extensible.

2.3.6. Le protocole d'application *Product LifeCycle Support*

Produt LifeCycle Support (PLCS) est le nom donné à l'AP239 [ISO-10303-239, 2005]. La définition de cette norme a été placée sous l'autorité de *PLCS Inc.* [8], un consortium regroupant les ministères de la défense des États-Unis, du Royaume-Uni, de la Finlande, de la Norvège, de la Suède ainsi que les industriels *DNV*, *Bœing*, *BAE SYSTEMS*, *Rolls Royce*, *Lockheed Martin*, *SAAB*, *Hagglunds Vehicles*, *BAAN*, *LSC*, *PTC*, *Aerosystems International* et *Pennant*[9]. *Eurostep Limited* a assuré le rôle de chef de projet des travaux du consortium.

Objectifs de PLCS PLCS a pour objectif de permettre la création et la gestion dans le temps d'un ensemble cohérent d'informations relatives à la maintenance des produits. Ces informations sont utilisées pour spécifier et contrôler l'ensemble des activités de maintenance durant tout le cycle de vie du produit. *PLCS* définit un modèle de données générique et flexible qui peut être adapté à des besoins industriels spécifiques [Rosén, 2006].

8. *http ://www.eurostep.com/*
9. http ://www.plcs-resources.org/#Background+to+PLCS+Inc

Couverture fonctionnelle de PLCS L'AP239 définit un modèle de données :
- permettant de définir un produit complexe et le soutien associé ;
- requises pour assurer la maintenance d'un produit complexe ;
- requises pour la gestion des configurations d'un produit et le soutien associé.

La couverture fonctionnelle du modèle de données PLCS permet de représenter :
- les assemblages :
  - identification et représentation des pièces, leur version, la gestion de la documentation et des informations associées (par exemple les dates et les approbations assignées aux pièces) ;
  - représentation de plusieurs vues de la structure produit ainsi que les déclinaisons produit ;
  - représentation de la géométrie d'un assemblage et de ses composants ;
  - identification des positions des composants dans un assemblage ;
  - association de métadonnées relatives à une pièce ou un assemblage ;
  - représentation des interfaces entre produits,
- les états observés (ou prévus) d'un produit ;
- l'évolution d'un produit : représentation des spécifications du produit et leur satisfaction, d'une évolution potentielle ou existante du produit, de la configuration d'un produit pour un rôle donné et spécification des contraintes d'effectivité appliquées à la configuration d'un produit ;
- la planification et l'ordonnancement des tâches ainsi la gestion et l'autorisation du travail qui en découle ;
- la classification des pièces, documents et assemblages ;
- la spécification et la planification des activités liées à un produit incluant : la spécification des tâches à accomplir sur un produit, des conditions de la réalisation de cette tâche (incluant les ressources nécessaires et la localisation des ressources et des produits), la représentation du type de personne et les compétences requises pour réaliser la tâche ;
- l'historique des activités pour un produit : enregistrement de l'utilisation d'un produit et l'utilisation d'une ressource, l'enregistrement des activités menées sur un produit et l'utilisation des ressources ;
- l'historique du produit : enregistrement de l'historique des états d'un produit et des états de configuration d'un produit, localisation des données produit, observation des données produit.

En revanche, les points suivants sont hors du périmètre de l'AP239 :
- la représentation des transactions financières pour les commandes, les achats ou le retour du produit et des ressources nécessaires au support produit ;
- la représentation des transactions financières concernant le transport, l'expédition et la réception des produits et des ressources associées.

Spécialisation du modèle de données *PLCS via* les *Data Exchange Sets* (DEX) L'AP239 propose un modèle de données *générique* qui propose une couverture fonctionnelle très étendue, à la granularité assez grossière : il a vocation à être spécialisé

en fonction d'un domaine métier particulier. La norme définit le processus permettant d'encadrer la manière dont sont rédigées ces spécialisations (nommées *Data EXchange set* ou DEX) pour garantir l'échange de populations d'instances. L'ISO a délégué auprès de l'*Organization for the Advancement of Structured Information Standards* (OASIS [10]) la coordination des travaux relatifs à ces DEX. OASIS est un consortium à but non-lucratif qui pilote le développement, la convergence et l'adoption d'une multitude de standards [11] ouverts dans le domaine de la société de l'information.

Un DEX est constitué des éléments suivants :
- une introduction ;
- un domaine métier ;
- une description de tous les processus métiers pris en charge ;
- l'identification du processus dans le modèle d'activité AP239 supporté ;
- un guide d'utilisation pour le Template ;
- un ensemble de données de référence (*Reference Data*), ayant pour fonction de compléter la sémantique du modèle de données générique PLCS ;
- le sous-ensemble du modèle d'information pris en charge ;
- une formalisation EXPRESS du modèle de données ;
- le schéma XML de ce modèle (déduit du précédent).

La construction des DEX s'appuie sur la bibliothèque DEXlib [12] dont le développement est ouvert [13]. Une fois les DEX éprouvés puis validés, ils sont inclus dans la norme ISO 10303 en temps que classe de conformité ou *Conformance Class* (CC).

Un regard sur l'AP233 Un autre protocole d'application modulaire est en cours de développement : il s'agit de l'AP233, intitulé *System Engineering* (ou ingénierie système). L'AP233 a vocation à ajouter à la norme STEP la « brique » ingénierie système qui lui fait encore défaut (point par ailleurs signalé dans la cartographie 2D de [Rachuri et al., 2008]). Ce projet de norme est le fruit d'une collaboration entre le groupe *System Engineering Domain Special Interest Group* [14] de l'*Open Management Group* (OMG), par ailleurs en charge du langage de modélisation SysML, et l'*International Council on Systems Engineering* (INCOSE). Bien que l'AP233 soit dédié à un domaine sémantique différent de l'AP239, ces deux AP n'en partagent pas moins 70% de leur modèle de données [Firsch and Stirk, 2007], notamment ce qui concerne la structuration du produit, la gestion des exigences, des activités etc. L'AP233 utilise aussi le même système de DEX pour spécialiser le modèle de données générique. En conséquence, l'AP233 et l'AP239 pourront être utilisés de manière complémentaire pour couvrir l'intégralité du cycle de vie du produit. Le statut actuel de cette norme est *Draft International Standard*, nulle

10. http ://www.oasis-open.org
11. La liste de ces standards est disponible à l'adresse : http ://www.oasis-open.org/specs/index.php
12. http ://www.plcs-resources.org/dexlib/index.html
13. http ://sourceforge.net/projects/dexlib
14. http ://syseng.omg.org/

date n'étant officiellement confirmée quant à une adoption en temps qu'*International Standard.*

Nous ne détaillerons pas davantage ce projet de norme : nous avons suffisamment d'éléments pour tirer des conclusions dans le paragraphe suivant. L'ingénierie système n'entre ni dans le champ de notre recherche (centrée sur les phases de conception et de production) ni dans celui de nos compétences. Néanmoins, ce court paragraphe a permis de montrer que le nouveau processus d'AP modulaires défini par l'ISO permet d'envisager des protocoles d'applications interopérables. D'autre part, le fait maîtriser le cadre formel de la norme pour un de ces protocoles permet aisément de s'en approprier un autre : le travail que nous présenterons dans le chapitre 3, relatif à l'AP239, pourra être transposé à l'AP233 pour résoudre de manière similaire des problèmes d'interopérabilité impliquant des produits mécatroniques.

Conclusion Les éléments de cette section permettent donc d'infirmer pour partie quelques-uns des résultats de [Rachuri et al., 2008] et par conséquent d'apporter à nos points de désaccord les justifications nécessaires. En reprenant chacun des trois points évoqués dans les premières lignes de la section 2.3.4, nous pouvons donc affirmer que :

- la norme STEP présente une double dimension modulaire : les APs sont associés à des domaines sémantiques différents, les AMs garantissent l'interopérabilité des AP. En ce sens, cette norme correspond tout-à-fait à l'observation de [Hoffmann, 2008] que nous avons déjà citée : *l'approche d'unification amènera à développer plusieurs versions d'ontologie, qu'il faudra garder à jour*. Les ontologies en question sont les AP, et c'est l'ISO qui se charge de la mise à jour des protocoles d'application (par exemple l'évolution de l'AP203 vers l'AP203ed2). L'intérêt de l'utilisation d'un modèle d'unification basé sur STEP dans le domaine du PLM est donc que le travail de modélisation sémantique est assuré par un tiers (en l'occurrence l'ISO ou OASIS) qui garantit des points essentiels : la sémantique STEP est conforme à des besoins industriels ; cette sémantique évolue au gré de l'évolution des besoins ; le modèle de données est robuste et complet. Se priver de cette richesse et de cette rigueur quand on aborde un problème d'interopérabilité doit donc être préalablement justifié et doit être envisagé comme une solution de repli ;
- la norme STEP a évolué vers le développement de protocoles d'application génériques et extensibles permettant une mise à l'échelle sémantique et temporelle ;
- PLCS doit être considéré comme faisant partie intégrante de STEP. A ce titre ces deux éléments ne forment pas une sémantique disjointe comme pourrait le laisser penser la cartographie de la figure 2.3.2 ;
- bien que le début de vie du produit ne soit pas encore officiellement couvert par STEP, les avancées relatives à l'AP233 laissent augurer que ce manque sera officiellement comblé sous peu, dès la publication par l'ISO de cet AP.

Fort de ce constat, nous proposons dans la section suivante une cartographie différente pour représenter le standard STEP.

2.3.7. Cartographie 3D du standard STEP

Nous proposons de substituer à la cartographie déjà présentée une représentation tridimensionnelle de la norme STEP dans le domaine du PLM (voir figure 2.3.5). Les trois dimensions choisies sont :

- l'axe *Produit/Process/Entreprise* ;
- l'axe *Phase du cycle de vie du produit*. Ces deux axes sont les mêmes que ceux définis par [Rachuri et al., 2008] ;
- l'axe *granularité du modèle de données*.

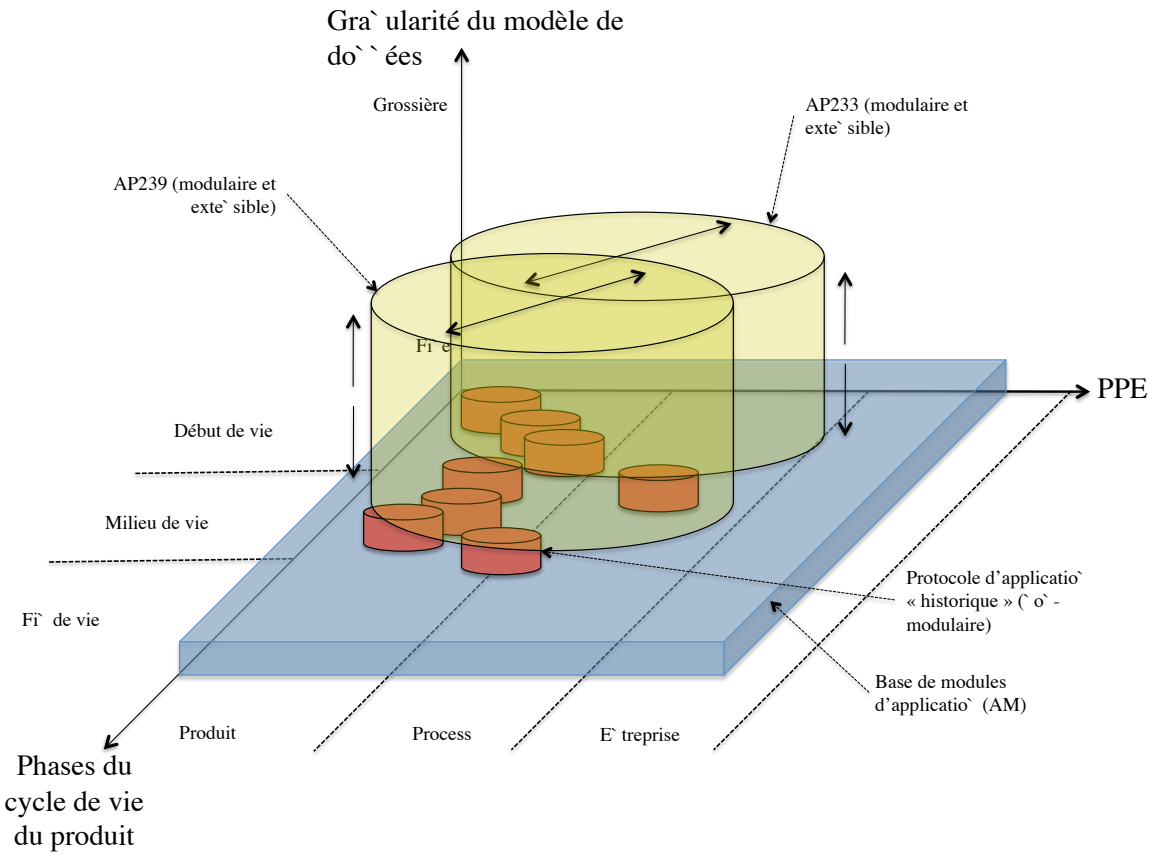


FIGURE 2.3.5.: Cartographie 3D du standard STEP

Sur cette figure, sont représentés :

- la base de modules d'application (AM) sous la forme d'un parallélépipède bleu. L'ensemble de ces AM couvre une large base dans le référentiel ($Phases$, PPE) et offrent une granularité fine ;

- les AP « historiques », c'est-à-dire non modulaires, par des petits cylindres rouges. Ils présentent la caractéristique d'être limités et figés d'un point de vue sémantique, et offrent une granularité fine (par exemple l'AP203 ou l'AP214) ;
- les AP modulaires 239 et 233 par deux gros cylindres jaunes. Ces APs offrent un modèle de donnée moins granulaire mais sont extensibles d'un point de vue sémantique, la spécialisation du modèle de données permettant d'affiner la granularité et/ou élargir le spectre sémantique couvert par l'AP. Ceci est représenté par les flèches noires sur la figure.

Cette représentation sert par la suite d'élément de base aux préconisations méthodologiques quant au choix du modèle standard d'unification, ces recommandations faisant l'objet de la section 2.4.

2.4. Spécifications quant au choix du modèle d'unification standard

Nous introduisons préalablement une classification en deux catégories des problèmes d'interopérabilité. Tout problème d'interopérabilité peut être rangé dans une des deux catégories suivantes :

- les problèmes de *transfert d'informations*, dans lesquels les informations d'un système S_i sont à transférer vers un système S_j. A l'échelle d'une phase, ces deux systèmes seront des applications. A l'échelle inter-phase, les systèmes seront des médiateurs (exemples de problèmes relèvant du transfert d'informations : transfert de l'état des stocks de l'ERP vers le PDM, de nomenclatures du PDM vers l'ERP, de données CAO vers le PDM etc.) ;
- les problèmes d'*empaquetage d'informations* : les informations de deux systèmes S_i et S_j doivent être réconciliées via le méta-modèle d'unification (empaquetées) et transférées vers un système tiers S_k (par exemple, les problèmes d'archivage à long terme des données entrent dans cette catégorie : on doit être en mesure d'archiver dans une sémantique commune l'ensemble des informations produit pour rendre possible leur interprétation à long terme [Borghoff et al., 2006]). Le flux sémantique est dans ce cas beaucoup plus important que dans le cas du transfert d'informations. Il est à remarquer que cette deuxième catégorie de problèmes peut être considérée comme un ensemble de problèmes de transfert (qu'ils soient simultanés ou séquentiels).

La médiation multi-échelle que nous proposons permet de traiter ces deux catégories de problèmes (voir tableau 2.4.1).

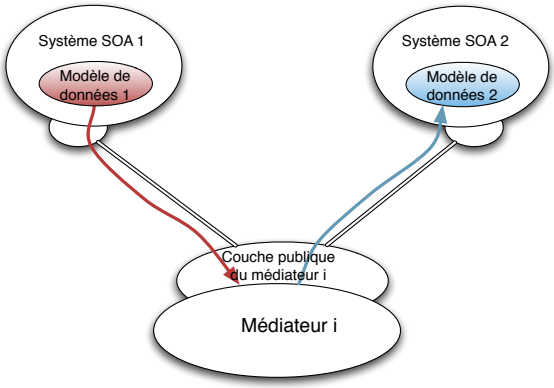
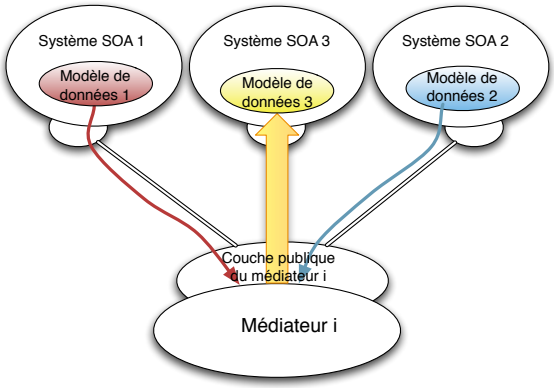
Transfert d'informations	Empaquetage d'informations
	

TABLE 2.4.1.: Les deux catégories de problèmes d'interopérabilité

Compte tenu de ces différentes observations, nos préconisations méthodologiques pour le choix du modèle standard sont les suivantes :

- dans un premier temps, **identifier la nature du problème d'interopérabilité à traiter (*transfert* ou *empaquetage*)**. Cette analyse doit conduire à **évaluer l'étendue du flux sémantique ainsi que la nature et la granularité des informations dont sera chargé le médiateur multi-échelle orienté services** ;
- dans un contexte de mise à l'échelle sémantique et temporelle, **on choisira prioritairement un modèle standard extensible** (par exemple un AP modulaire de la norme STEP). Plus le flux sémantique sera important, plus la nécessité de recourir à un tel modèle standard s'imposera. La cartographie de la figure 2.3.5 doit permettre de déterminer si la couverture sémantique de ce modèle standard ainsi que la granularité de son modèle de données correspondent à celles du premier point ;
- **pour les problème d'interopérabilité inter-phase, on s'orientera exclusivement vers un modèle standard extensible** ;
- **si la couverture sémantique ou la granularité de ce standard extensible ne sont pas suffisantes en l'état**, on cherchera à les **étendre par un enrichissement sémantique et une spécialisation du modèle de données** ;
- **si**, pour quelque raison que ce soit, **il s'avère que l'extensibilité du modèle standard ne peut être réalisée, on s'orientera alors vers un standard non-extensible**, de type AP STEP non-modulaire ;
- dans tous les cas, **l'analyse du standard par l'examen du diagramme UML ou EXPRESS-G de son modèle de données n'est pas suffisant.** C'est bien la sémantique qu'il porte qui est un critère pertinent. Par exemple, le choix du PDM Schema (qui porte une sémantique conception puisque résultant de l'intersection minimale entre les AP203, 212, 214 et 232) n'est pas adapté pour réaliser l'interopérabilité ERP/MES (qui relèvent du domaine de la production) car, ce faisant, on sacrifie la robustesse vis-à-vis d'un changement d'échelle (ce qui est le choix de [Tursi et al., 2009]). Cette décision doit être explicite et justifiée ;

- **en dernier ressort, si l'utilisation d'aucun modèle standard n'est possible, on penchera à défaut pour un modèle *ad hoc***. Le travail à accomplir sera d'autant plus grand que le flux sémantique est important. Pour cette raison, cette solution sera exclusivement réservée à des organisations qui disposent des importantes ressources nécessaires.

Après avoir déterminé le méta-modèle d'unification pertinent, la prochaine étape de notre méthodologie consiste à explorer la question du « comment ». Comment identifier la nature des informations à transférer ? Comment exploiter un modèle standard ? Comment mettre en relation les informations des différents systèmes avec le méta-modèle d'unification ? Les réponses à ces différentes questions sont développées dans le chapitre 3.

Résumé du chapitre 2

Ce chapitre vise à définir une architecture permettant de mettre en œuvre l'interopérabilité d'un SI dans un contexte de complexité multi-échelle.

Dans un premier temps, nous identifions la topologie de réseau en étoile comme étant la plus pertinente vis-à-vis du coût global de possession du SI ainsi que de son agilité. Le composant central de cette architecture a pour fonction de réaliser l'interopérabilité dans le SI : c'est un médiateur d'informations. Le développement des architectures orientées services permet d'étendre ce concept à celui de médiateur d'informations orienté services. Dans le contexte de notre étude, nous proposons une architecture basée sur de nombreuses instances de ce médiateur que nous appelons donc *médiation multi-échelle orientée services*. Cette médiation est capable d'adresser tous les problèmes d'interopérabilité du type transfert ou empaquetage d'informations.

Après une étude comparée des approches d'intégration, unification ou fédération possibles pour réaliser la mise en correspondance sémantique des informations, nous préconisons une démarche d'unification.

Le choix d'un méta-modèle d'unification basé sur les standards apparaît *a priori* pertinent. La littérature relative aux standards du PLM nous permet d'identifier la norme STEP comme un candidat sérieux au méta-modèle d'unification. Nous montrons alors que cette norme, du fait de son évolution vers des protocoles d'application génériques et extensibles, constitue une solution qu'il convient d'envisager en priorité.

Ce chapitre se termine par un ensemble de prescriptions méthodologiques quant au choix du protocole d'application le plus pertinent.

3. Modèles pour l'interopérabilité sémantique

Dans ce chapitre, nous expliquons comment, d'un point de vue méthodologique, mettre en œuvre les points théoriques présentés dans le chapitre précédent. Ces prescriptions méthodologiques se basent sur deux exemples de problèmes d'interopérabilité : CAO/PDM d'une part (voir section 3.2) et PDM/ERP d'autre part (voir section 3.3), bien que la portée plus générale de notre méthodologie permette de couvrir un panel de problèmes beaucoup plus large.

Remarque : la structure choisie pour rédiger ce mémoire diffère du déroulement temporel de notre recherche. En effet, les préconisations méthodologiques qui parsèment le chapitre 2 nous sont apparues progressivement. Elles sont le fruit d'une confrontation entre les résultats de la littérature scientifique et nos propres expérimentations. Ainsi la section 3.2, choisie pour illustrer l'interopérabilité CAO/PDM intra-phase, constitue-t-elle le premier de nos travaux [Paviot et al., 2008]. La section 3.3, relative à l'interopérabilité PDM/ERP [Paviot et al., 2009], illustre les problèmes d'interopérabilité inter-phase et s'appuie sur ces premiers travaux. Enfin, la section 3.4, qui présente un modèle basé sur des annotations sémantiques [Paviot et al., 2010], tente de lever les ambiguïtés dont nous avons pris conscience au gré de ces expériences. Tous ces travaux ont été validés par des publications scientifiques, nous les présentons donc dans ce chapitre. Cependant, les éléments relatifs à l'interopérabilité CAO/PDM n'ont pas été retravaillés dans le sens des spécifications que nous avons faites. Par conséquent, le présent chapitre doit être lu aussi bien comme la suite de la démarche méthodologique initiée dès le chapitre 1 que comme un ensemble d'éléments qui présentent comment s'est construite notre approche de la résolution des problèmes d'interopérabilité dans le domaine du PLM.

3.1. Aspects méthodologiques

D'un point de vue méthodologique, les préconisations qui concluent le chapitre 2 sont mises en œuvre de la manière suivante :

- l'identification de la phase impliquée sera basée sur l'**étude du métier des utilisateurs concernés et des SI qu'ils mettent en œuvre** ;
- la nature et la granularité des informations échangées proviendront d'une **analyse du besoin du problème d'interopérabilité** ;

- on cherchera par la suite à **évaluer la distance sémantique de ces informations, et en particulier déceler les éventuelles similarités** (les informations pour lesquelles la distance sémantique est faible par rapport à l'étendue de la phase concernée et du SI pris dans sa globalité). Ces similarités permettent en effet d'identifier les points de convergence entre les modèles de données des SI et le méta-modèle d'unification de leur médiateur ;
- les informations les plus proches (idéalement similaires) d'un point de vue sémantique seront modélisées afin de pouvoir être mises en correspondance par l'intermédiaire du méta-modèle d'unification.

La section suivante présente l'instanciation de cette méthodologie pour résoudre un problème d'interopérabilité entre logiciels CAO et systèmes PDM.

3.2. Médiation intra-phase : exemple de l'interopérabilité CAO / PDM

Le cas d'étude présenté est le fruit d'une collaboration entre les élèves-ingénieurs de Supméca Paris [1] et EADS Innovation Works (IW), centre commun de recherche du groupe EADS situé à Suresnes (92). Baptisé *FALSIM* [2] (acronyme de *Final Assembly Line SIMulation*), ce projet avait pour objectif de concevoir et simuler une ligne d'assemblage final aéronautique [Cheutet et al., 2009]. Il a mobilisé 14 étudiants pendant 3 semaines, qui ont conçu et validé un hall d'assemblage conforme aux spécifications d'EADS IW en termes de consommation des ressources et de cadences minimales à respecter.

3.2.1. Situation initiale - Origine du problème d'interopérabilité

Dans le cadre de ce projet, une grande liberté avait été laissée aux étudiants quant aux choix à mettre en œuvre pour l'assemblage des différents composants de l'avion (tronçons, ailes, empennage etc.) et comparer ainsi plusieurs *scenarii* envisageables. La maquette numérique 3D de l'avion était une donnée d'entrée du projet ayant pour fonction de permettre la conception des outillages d'assemblage, définir les tâches des opérateurs et des robots ou encore évaluer le volume enveloppe des composants lors de leurs déplacements dans l'usine (voir figure 3.2.1). Cette maquette a été fournie aux étudiants en début de projet sous la forme d'un unique et volumineux fichier STEP au format AP203 généré à partir du logiciel Catia V5. Cette maquette constituait une contrainte générale de l'étude : à ce titre, aucune modification de la géométrie de l'avion n'était autorisée.

1. http ://www.supmeca.fr
2. Vidéo de présentation réalisée par les étudiants de Supméca Paris : http ://www.youtube.com/watch ?v=h9M31N-VTQY

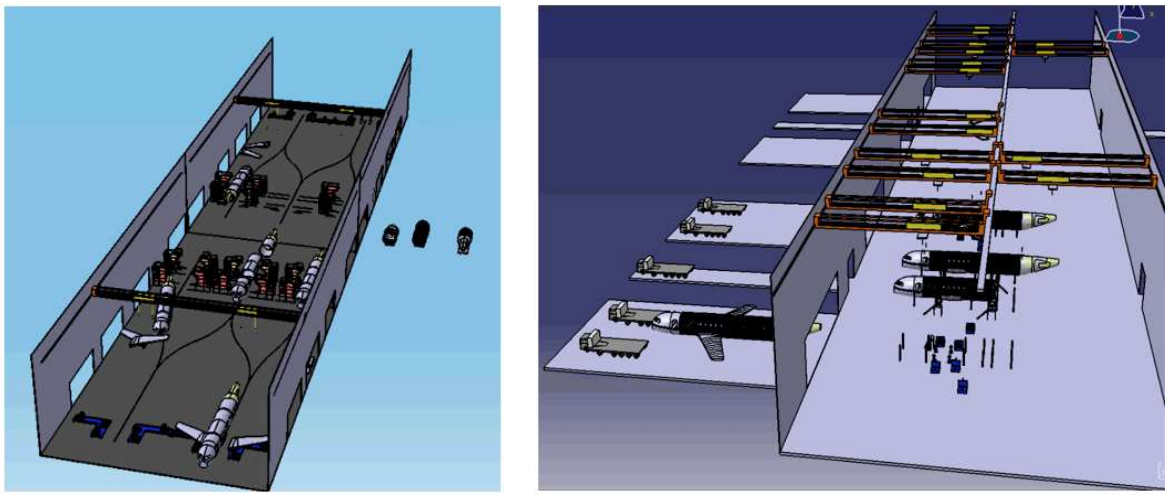


FIGURE 3.2.1.: Le projet FALSIM (Final Assembly Line) Supméca/EADS IW

Compte tenu du temps imparti ainsi que du nombre d'étudiants concernés, les contraintes de travail collaboratif étaient fortes : 5 équipes ont travaillé sur des sous-objectifs distincts suivant les préceptes de l'ingénierie concourante et simultanée. Le **partage de la maquette numérique de l'avion s'est donc révélé comme une nécessité**.

Les outils mis à disposition des étudiants étaient (entre autres) les suivants :
- Catia V5 (édité par Dassault Systèmes) pour la conception des différents éléments composant l'usine ;
- Delmia V5 (édité par Dassault Systèmes) pour la conception des gammes d'assemblage ;
- Windchill (édité par PTC) pour la gestion des données techniques et le pilotage du travail collaboratif.

Le **problème** qui s'est posé peut être formulé de la manière suivante : comment partager, dans le cadre de ce projet, une maquette numérique volumineuse en utilisant des outils d'authoring (Catia/Delmia) et PDM (Windchill) ne proposant pas d'intégration native ? C'est de cette dualité entre le besoin de partage et le travail dans un environnement logiciel hétérogène qu'est né le problème d'interopérabilité que nous avons cherché à résoudre. Cette situation particulière modélise de nombreux problèmes industriels : en effet, d'après [Bissay et al., 2008], *dans la plupart des entreprises, le SI est constitué de composants hétérogènes couvrant tous les secteurs de l'entreprise.*

3.2.2. Phases du cycle de vie du produit

Le projet FALSIM, dans sa finalité, se situait au carrefour des phases de conception et de production. Néanmoins, le problème d'interopérabilité précédent visait à permettre à tous les acteurs de :

- charger en session CAO une partie de la maquette numérique de l'avion ;
- concevoir les outillages de production qui s'appuient sur la géométrie de l'avion ;
- concevoir la géométrie du hall d'assemblage en tenant compte des contraintes d'encombrement ;
- partager ces données avec les autres membres de l'équipe.

Ces trois activités relèvent du domaine de la conception, de même que les outils utilisés. En conséquence nous avons été confrontés à **un problème d'interopérabilité intra-phase**. Nous examinons dans la section suivante la nature et la granularité des informations à échanger dans ce cadre.

3.2.3. Nature et granularité des informations à échanger

D'après le besoin exprimé précédemment, la collaboration autour de la maquette numérique devait s'effectuer au niveau des composants de l'avion ou des outillages complets, c'est-à-dire avec une granularité grossière, sans nécessité de collaborer sur des éléments de très bas niveau de la maquette numérique (entités topologiques de base, paramètres de conception etc.). Par exemple, l'équipe chargée de la robotisation de l'assemblage du tronçon arrière (voir figure 3.2.2) a dû :

- travailler à partir des trois éléments du tronçon (pavillon, barque, plancher), en gris sur la figure, provenant de la maquette numérique globale ;
- partager avec le reste de l'équipe les outillages conçus (en bleu foncé et bleu clair sur la figure) afin qu'ils puissent être instanciés dans la maquette numérique de l'usine.

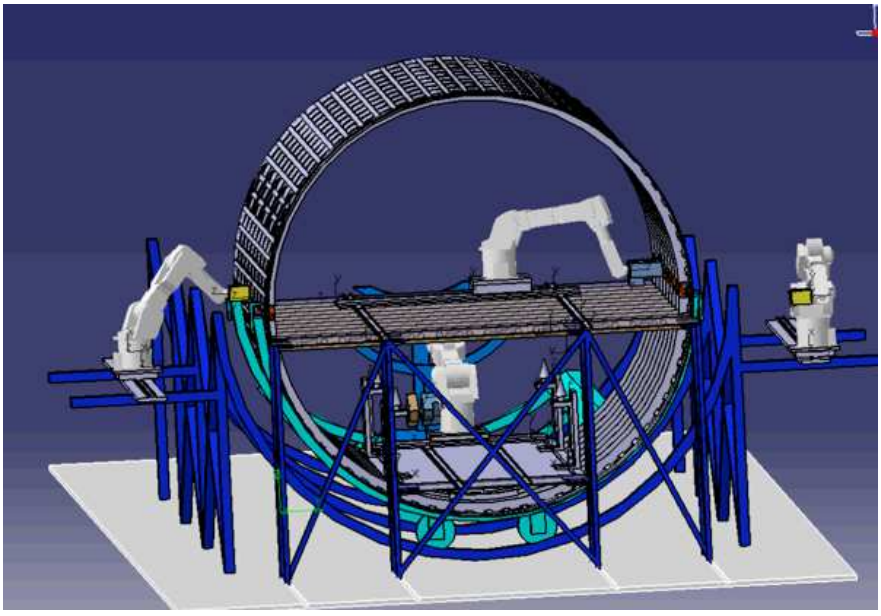


FIGURE 3.2.2.: La collaboration à un haut niveau granulaire

Ainsi, d'un point de vue sémantique :
- on devra être en mesure d'échanger toute la sémantique relative à la définition de l'avion et des outillages. Cette sémantique devra être *contenue* toute entière dans le système PDM chargé de la ventiler sur chaque poste CAO. Il faudra donc s'assurer de la *transporter* ou la *conduire* d'un système à l'autre ;
- seule une petite partie de cette sémantique doit pouvoir être *interprétée* et *comprise* par le PDM en vue du travail collaboratif : celle-ci concerne la décomposition du produit en éléments plus simples. Il faudra donc être en mesure de la *convertir* d'un système à l'autre.

C'est la partie concernant l'interprétation de la sémantique par les deux systèmes qui est dimensionnante pour les problèmes d'interopérabilité (voir à ce sujet les définitions données dans la section 2.1) : dans le cas de la collaboration autour de la maquette numérique de l'avion, l'étendue sémantique à convertir est faible. Nous sommes donc typiquement dans le cas d'un problème d'interopérabilité de type transfert tel qu'il a été défini dans la chapitre précédent : un flux sémantique faible doit être transporté d'un système à l'autre.

Dans la section suivante, nous nous intéressons à l'étude des deux systèmes impliqués (CAO et PDM), et examinons comment la décomposition du produit est réalisée, et dans quelle mesure ces éléments sont éloignés sémantiquement.

3.2.4. Recherche des similarités sémantiques

Dans cette section, notre réflexion s'inscrit d'un point de vue plus général que la simple interopérabilité de deux logiciels spécifiques. Étant guidé par le souci de l'aspect multi-échelle des problèmes d'interopérabilité, nous avons cherché une solution qui puisse s'adapter à plusieurs logiciels différents, que nous avons donc appelée *médiation multi-CAO/multi-PDM*. Nous proposons donc une étude comparée des logiciels CAO et systèmes PDM par rapport au besoin d'échanger des informations sur les composants de la maquette numérique et leurs relations.

Structuration du produit dans les logiciels de CAO 3D L'interface graphique de la plupart des logiciels de CAO propose une fenêtre graphique 3D, depuis laquelle l'opérateur de conception peut manipuler la maquette numérique, ainsi qu'une vue hiérarchique du produit présentant une topologie arborescente : cette structure permet de naviguer parmi les constituants de la maquette (voir tableau 3.2.1).

La structure CAO organise les données 3D qui décrivent le produit virtuel. La sémantique portée par ces liens de structure de type parent/enfant est largement implicite : la documentation de ces logiciels n'impose pas de procédé qui permette la hiérarchisation univoque des items (fonctionnelle ? organique ? physique ? etc.). Chaque organisation

doit spécifier sa méthodologie d'utilisation de l'outil et de la structuration conséquente pour garantir un résultat cohérent vis-à-vis des tous les interacteurs de la maquette [Tomovic et al., 2009]. Cette méthodologie, généralement archivée dans une base de connaissances sous la forme d'un recueil des « meilleures pratiques », permet d'établir un lien univoque entre les liens de structure et leur sémantique c'est-à-dire *in fine* un référentiel sémantique partagé dans tout le bureau d'études.

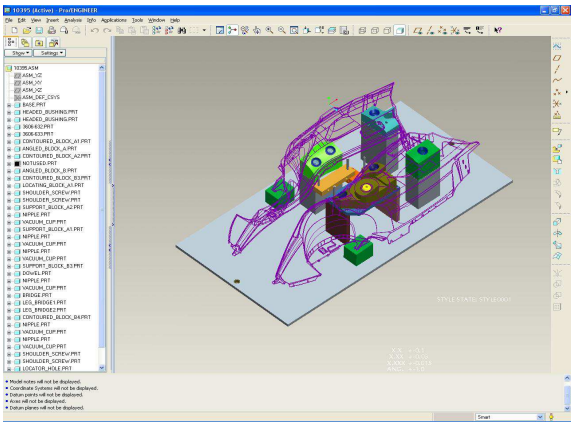
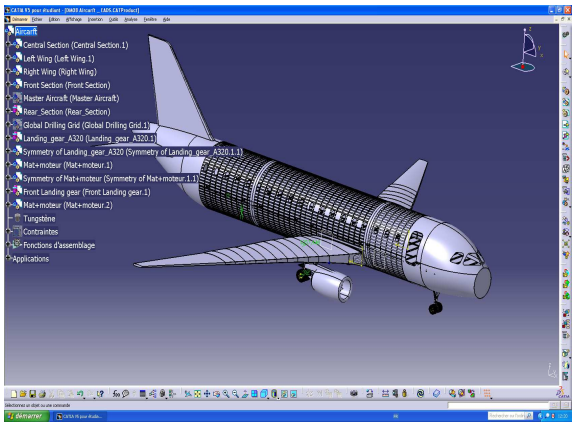
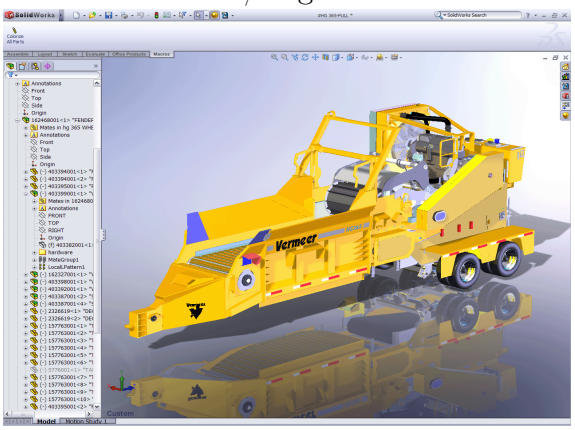
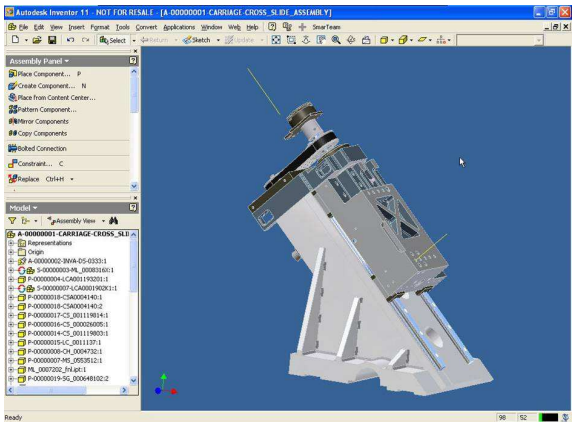
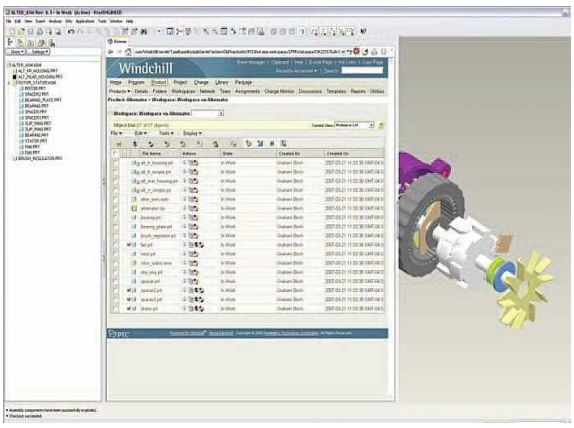
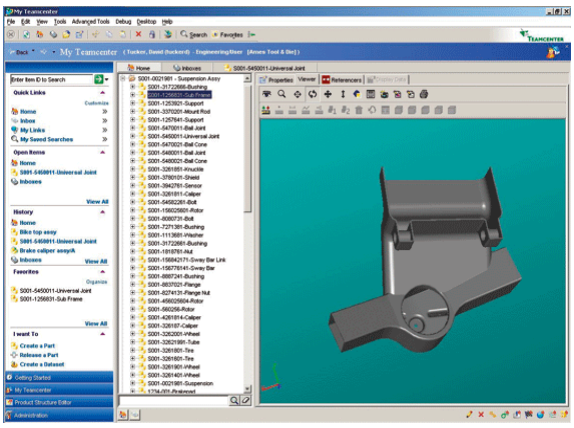
	
Pro/Engineer	Catia V5
	
SolidWorks	Inventor

TABLE 3.2.1.: La structuration du produit dans les logiciels de CAO3D

Structuration du produit dans les systèmes PDM Nous avons présenté, dans la section 1.2.3, les caractéristiques des systèmes PDM et notamment leur fonctionnalité de *gestion de la structure produit, qui organise et stocke les informations du produit* [Eynard et al., 2004]. Cette fonction apparaît dans l'interface utilisateur de ces systèmes, qu'elle soit basée sur un client lourd ou un client léger (par exemple un navigateur Internet). Dans le tableau 3.2.2, la structuration du produit apparaît sous la forme d'une structure arborescente (on voit également la fonction de visualisation 3D des composants stockés dans le PDM, évitant ainsi leur chargement dans un outil CAO).



Windchill



TeamCenter

TABLE 3.2.2.: Structure produit et visualisation 3D dans les systèmes PDM(exemple de Windchill et TeamCenter)

Similarités sémantiques et choix du méta-modèle d'unification Sans avoir eu besoin de conduire une étude très poussée sur la manière dont sont implémentés ces logiciels et sur leur modèle de donnée interne, il apparaît de très fortes similarités dans la structuration du produit opérée par ces deux systèmes. Par rapport au besoin d'interopérabilité exprimé plus-haut (échanges de haut niveau granulaire), ces similarités seront donc les deux portes d'entrée du méta-modèle d'unification.

En conclusion de ces observations :
- nous sommes confrontés à un problème d'interopérabilité de type *transfert* entre des systèmes CAO et PDM ;
- les informations à échanger concernent la stucturation du produit. Elles présentent une granularité grossière ;
- le choix d'un méta-modèle d'unification s'oriente donc prioritairement vers un méta-modèle standard extensible (AP239), accessoirement vers un méta-modèle standard qui couvre la phase de *conception*.

Dans le travail relatif au projet FALSIM, nous n'avions pas les compétences pour mettre en oeuvre PLCS. Nous nous sommes donc orientés vers un méta-modèle d'unification standard non-extensible, qui reprend la structuration du produit commune aux AP 203 et 214 et au PDM Schema.

La phase suivante consiste à modéliser les informations qui doivent être *converties* et celles qui doivent être *transportées* par le médiateur, afin de pouvoir en déduire le modèle de données UML préalable à une implémentation.

3.2.5. Modélisation des informations à échanger

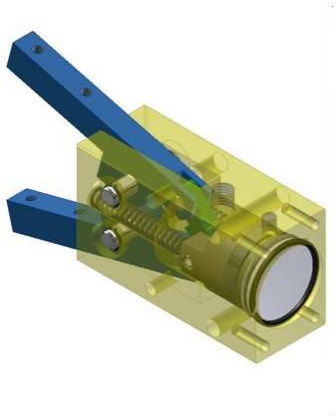
Cette étape de modélisation s'articule en deux phases :

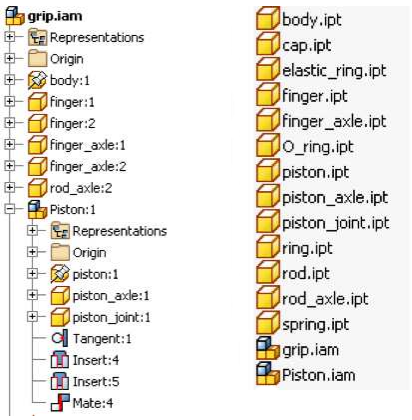
- modéliser la sémantique des informations à échanger depuis et vers chacun des deux systèmes ;
- effectuer une mise en correspondance (suivant des règles de mapping) pour tenir compte des nuances sémantiques (*similarité sémantique* n'étant pas équivalent de *synonyme*).

Modélisation des informations depuis/vers les outils CAO Bien que chaque éditeur de logiciel CAO implémente un modèle de données propriétaire (donc inaccessible), nous avons remarqué que tous les logiciels que nous avons étudiés proposent un ensemble de fonctionnalités de base très similaires, de même qu'ils partagent la même technique pour le stockage des fichiers représentant la maquette numérique d'un produit. Ainsi, la sémantique portée par la maquette numérique (*Digital Mock Up* ou DMU) est tout entière contenue dans :

- un ensemble de fichiers décrivant la géométrie volumique des composants. Ces fichiers, généralement dénommés fichiers « pièce », portent l'extension CATPart pour Catia V5, SLDPRT pour SolidWorks, IPT pour Inventor ou encore PRT pour Pro/Engineer ;
- un ensemble de fichiers décrivant le positionnement (relatif ou absolu) de ces géométries. Généralement dénommés fichiers « assemblage », ils portent l'extension CATProduct pour CATIA V5, SLDASM pour SolidWorks, IAM pour Inventor, ASM pour Pro/Engineer ;
- un ensemble de fichiers contenant des informations relatives aux différentes configurations d'une pièce ou d'un assemblage. Ces fichiers peuvent être au format CSV/ASCII, ou provenir du tableur d'une suite bureautique. Dans Catia V5, on parle de « table de paramétrage » pour dénommer ce type de fichier.

Nous expliquons cette phase de modélisation en nous basant sur un exemple plus simple que celui de l'avion : un préhenseur pneumatique industriel. L'ensemble des « fichiers CAO » de ce préhenseur contient toute la sémantique nécessaire à la reconstruction de la maquette numérique dans l'environnement CAO : la géométrie et la structuration du produit (voir tableau 3.2.3).







Vue 3D | Structure produit | Fichiers CAO

TABLE 3.2.3.: Vue 3D, structure produit et fichiers de définition d'un préhenseur industriel (Inventor).

Ces fichiers seront donc particulièrement adaptés pour *transporter* toute la sémantique de la maquette vers le PDM. Sachant que chaque fichier peut faire référence à un ou plusieurs autres fichiers, il faut être en mesure de reconstruire ces dépendances, qui ne sont pas nécessairement isomorphes de la structure produit. C'est l'objet de cette phase de modélisation.

Nous modélisons les relations entre fichiers par un graphe (voir figure 3.2.3). Les sommets de ce graphe sont des éléments de trois ensembles définis comme suit :
- F est l'ensemble des fichiers nécessaires à la définition complète de la géométrie de la maquette numérique ;
- I est l'ensemble des nœuds apparaissant dans la structure produit ;
- C est l'ensemble des fichiers définissant les configurations différentes d'une pièce ou d'un assemblage.

Chaque fichier $f \epsilon F$ peut être instancié plus d'une fois, *i.e.* il peut y avoir plusieurs occurrences de la même pièce ou du même assemblage dans la structure produit. Dans l'exemple le tableau 3.2.3, le doigt du préhenseur apparaît deux fois dans la structure produit mais il n'y a qu'un fichier « pièce » pour décrire sa géométrie. Fort de ce constat, nous pouvons écrire l'inégalité cardinale suivante :

$$0 \leqslant card(C) \leqslant card(F) \leqslant card(I) \tag{3.2.1}$$

Sur le graphe de la figure 3.2.3, quatre types d'arcs orientés peuvent être définis :
- des liens de structure (S_L) entre éléments de I. Ils modélisent les relations parent/enfant de la structure produit ;
- des liens de description géométrique (G_L) entre éléments de I et éléments de F ;
- des liens de configuration (C_L) entre éléments de F et éléments de C ;
- enfin, des liens de dépendance ($S_L^{'}$) entre éléments de F. Ces liens modélisent les interactions possibles entre fichiers résultant par exemple d'une conception en contexte.

(S_L) et ($S_L^{'}$) sont tous deux nécessaires pour caractériser l'information du produit : (S_L) définit le nombre d'instances alors que ($S_L^{'}$) définit quels sont les fichiers requis pour le chargement de la maquette numérique dans la session CAO. Par un raisonnement analogue à celui sur les sommets, nous pouvons en déduire la relation suivante concernant la cardinalité des arcs :

$$0 \leqslant card(S_L^{'}) \leqslant card(S_L) \tag{3.2.2}$$

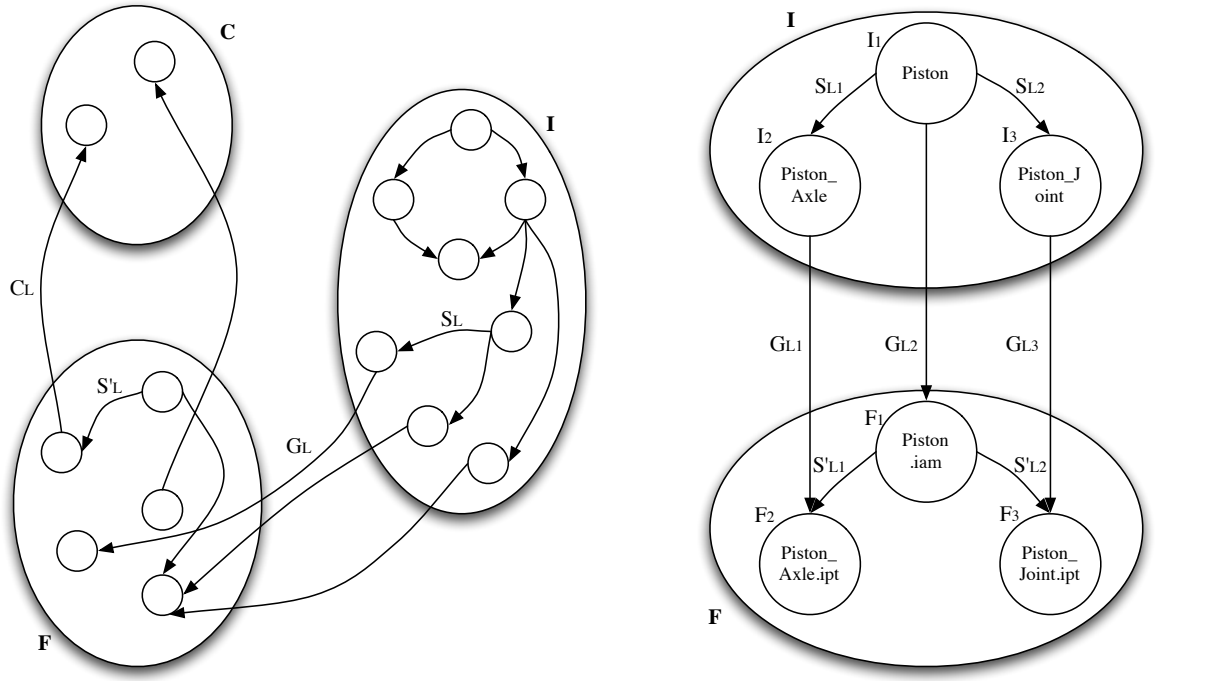
L'ensemble des arcs de description (G_L) est une application surjective de I dans F (ce qui est en cohérence avec la relation de cardinalité 3.2.1). Soit D la réunion de tous les ensembles permettant de décrire la maquette numérique :

$$D = C \cup F \cup I$$

et L l'ensemble des dépendances entres ces ensembles (*i.e.* l'ensemble des arcs du graphe) :

$$L = S_L \cup G_L \cup C_L \cup S'_L$$

Le graphe $DMU(D, L)$ contient l'information nécessaire et suffisante au transfert vers le système PDM pour être en mesure de garantir la cohérence des informations numériques.



a. Cas général **b. Instanciation du graphe pour l'assemblage du piston**

FIGURE 3.2.3.: Modèle graphe des dépendances entre fichiers CAO pour une maquette numérique

Nous définissons maintenant le sous-graphe $PS(I, S_L)$ qui modélise la structure produit CAO telle qu'elle apparaît dans l'arbre de construction. Ayant défini le graphe $DMU(D, L)$ comme portant les informations sur les dépendances nécessaires à un chargement de la maquette numérique, la sauvegarde d'une session CAO sous forme de fichiers implique que le graphe $DMU(D, L)$ est *connexe* et *orienté*. Aucun élément de I ne peut s'auto-référencer et il ne peut par conséquent exister aucune boucle dans le sous-graphe $PS(I, S_L)$, ce qui peut s'écrire :

$$si \quad \exists (I_i, I_j) \in S_L \Rightarrow (I_j, I_i) \notin S_L$$

Cette propriété est importante, puisqu'elle permettra de montrer l'isomorphisme de ce graphe avec celui modélisant les relations entre articles du point de vue du PDM. Ce point fait l'objet du paragraphe suivant.

Modélisation des informations depuis / vers le PDM Ces données doivent être stockées dans le PDM pour assurer leur cohérence durant le processus de conception. Les données apparaissant dans la figure 3.2.3 doivent donc être mises en regard du modèle de données du PDM dans lequel elles seront stockées. Cette figure fait apparaître de manière graphique le modèle de données utilisé dans le PDM Windchill : la structure produit est définie par des liens hiérarchiques de type parent/enfant entre articles (de conception), chaque article pouvant de plus être caractérisé par un ensemble de documents liés.

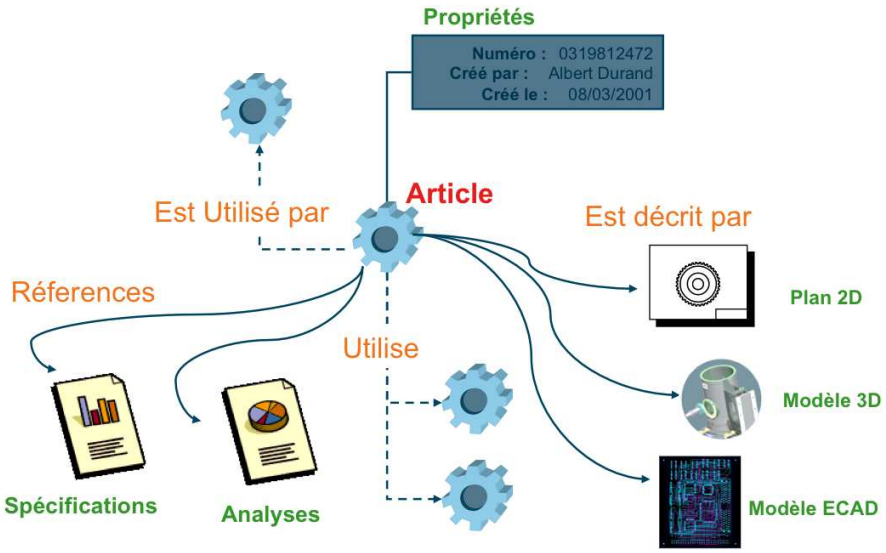
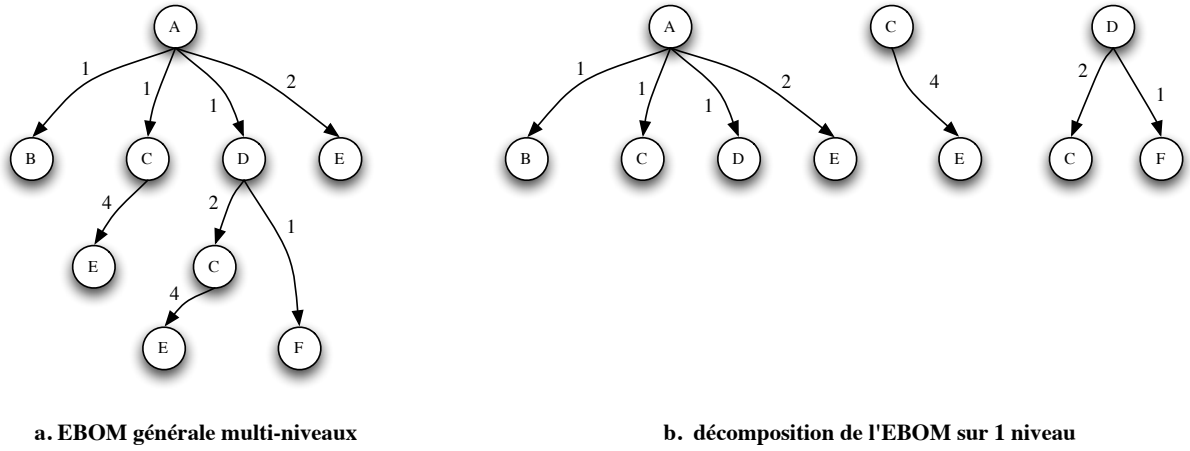


FIGURE 3.2.4.: La représentation d'un article dans Windchill (courtesy of PTC)

Le graphe $E(P, L)$ modélise cette structure produit PDM (voir figure 3.2.5), les sommets P modélisant les articles et les arcs L les liens de dépendance « Utilise »/« Est Utilisé Par ». $E(P, L)$ est un 1-graphe pondéré orienté. Le poids w_i de l'arc L_i représente le nombre d'occurrences du lien parent/enfant (la quantité). Dans la figure 3.2.5, la structure complète est éclatée en $n - 1$ structures constituées d'un seul niveau hiérarchique.



a. EBOM générale multi-niveaux **b. décomposition de l'EBOM sur 1 niveau**

FIGURE 3.2.5.: Modèle graphe de la structure produit PDM (Windchill)

Conversion des structures produits entre la CAO et le PDM (règles de *mapping*)
Les deux structures définies dans les paragraphes précédents sont modélisées par des graphes $PS(I, S_L)$ et $E(P, L)$ présentant des topologies arborescentes, connexes, orientés, sans boucle interne. Convertir une structure CAO vers PDM (ou PDM vers CAO) produit revient *in fine* à construire un isomorphisme du graphe $PS(I, S_L)$ vers $E(P, L)$ (ou $E(P, L)$ vers $PS(I, S_L)$). Pour cela, on doit s'attacher à définir :

- les règles de conversion des éléments de I en éléments de P, *i.e.* quelles sont les règles de *mapping* entre les composants de la structure CAO et les articles de la structure PDM ;
- les règles de conversion des éléments de S_L en éléments de L, i.e. les règles de *mapping* des liens de structure CAO vers les liens de structure du PDM.

La première approche que nous avons expérimentée est la plus simple : chaque sommet du graphe $PS(I, S_L)$ est mis en association bijective avec un sommet du graphe $E(P, L)$. L'utilisation de cette règle nous a amené à résoudre trois problèmes :

1. Les arcs du graphe $E(P, L)$ sont pondérés alors que ceux du graphe $PS(I, S_L)$ ne le sont pas. Par exemple, le composant **Finger** du préhenseur est instancié deux fois dans la structure CAO (donc deux arcs) alors que dans la structure PDM il n'est modélisé que par un seul arc pondéré par la valeur 2. Dans ce cas, nous devons préalablement analyser le graphe $DMU(D, L)$: les deux sommets « Finger » présents dans I sont connectés au même élément « finger.ipt » de F, *i.e.* ces deux instances peuvent être regroupées dans le graphe $E(P, L)$. Pour obtenir ce résultat, nous utilisons l'algorithme suivant : $\vee I_i \in I$, une exploration récursive permet d'obtenir tous les composants enfant $I_{k,l,m...}$. Pour chaque arc $I_i I_{k,l,m...}$, les éléments correspondants sont recherchés dans le graphe $E(P, L)$ et l'arc (PP) est créé, affecté de la pondération $w = 1$. Si cet arc existe déjà, alors w est incrémenté. L'algorithme est initialisé à partir de l'élément de plus haut niveau dans la structure, I_0 et se termine lorsque I_i n'est lié aucun élément enfant ;

2. Toutes les instances de la structure produit ne sont pas nécessairement à convertir en articles PDM. Par exemple, durant les phases de conception préliminaire, il est habituel de débuter la conception par un ensemble d'éléments géométriques (points, lignes, surfaces) qui s'étofferont par la suite d'un ensemble de fonctions volumiques. D'autre part, certaines méthodes de conception robustes préconisent l'utilisation d'esquisses pilotantes : un ensemble de paramètres et d'éléments géométriques permettent de piloter la maquette numérique complète [Bellacicco et al., 2007]. Ces esquisses ne doivent pas nécessairement figurer dans la structure produit PDM (par exemple si la structure PDM est utilisée en lien avec la nomenclature de production de l'ERP). Il faut prendre en compte ce cas de manière explicite : nous avons décidé d'adjoindre au nom de l'instance correspondante un tag _MS (pour *Master Sketch*) dédié à l'algorithme de conversion. Nous avons ainsi la règle suivante : toutes les instances de la structure CAO qui ne portent pas le tag _MS sont converties en articles PDM ;

3. Les composants utilisés par le concepteur sont-ils achetés (cas des éléments standards) ou fabriqués ? Le PDM étant capable d'*interpréter* cette sémantique, c'est une information qu'il est indispensable de lui fournir. Les catalogues de composants 3D (Traceparts [3] par exemple) proposent des modèles géométriques au format STEP de plusieurs milliers d'éléments standards (roulements, visserie, capteurs etc.). Ces éléments sont insérés dans la maquette numérique du produit, permettant ainsi un gain de temps précieux au concepteur. Dans le cas d'un transfert automatisé de la structure produit CAO vers le PDM, comment informer ce dernier que la structure produit correspondante devra être considérée comme un seul article (correspondant à la tête de nomenclature) portant la métadonnée « acheté » et non comme une structure arborescente ? Pour résoudre ce problème, nous avons imaginé la solution suivante : il suffit d'indiquer, dans le nom du composant tel qu'il apparaît dans la structure CAO, un tag _PURCHASED ou _ACHETÉ (voir figure 3.2.6). La règle associée est donc : toutes les instances portant le tag _PURCHASED ou _ACHETÉ doivent être converties en un seul article PDM caractérisé par la méta-donnée « acheté ». Le cas échéant, c'est la méta-donnée « fabriqué » qui sera utilisée par défaut pour caractériser cet article dans le PDM, et on propagera vers le PDM la structure CAO qui caractérise cet article.



FIGURE 3.2.6.: Composants fabriqués/achetés

3. http ://www.traceparts.com

Les solutions apportées aux problèmes des points 2 et 3 peuvent être considérées comme des solutions très pragmatiques, voire des artifices. Ce sont néanmoins ces premières tentatives qui nous ont mis sur la piste des étiquettes sémantiques associées au produit que nous développerons dans la section 3.4.4.

3.2.6. Représentation UML du méta-modèle d'unification

En conlusion de cette section relative à l'interopérabilité CAO/PDM, nous proposons ci-dessous le diagramme UML du méta-modèle d'unification au cœur du médiateur multi-échelle orienté services. Son implémentation sera vue dans le chapitre 4 :

- `CADDocument` modélise les fichiers CAO ;
- `CADDependency` modélise les dépendances entre fichiers CAO ;
- Part et PartMaster permettent de définir un article versionné dans la structure PDM ;
- les liens de type parent/enfant `StructureLink` permettent de construire l'EBOM dans le PDM ;
- `CADDescribeLink` est un lien d'association entre un article PDM et un fichier CAO ;
- le lien de `PartDescribeLink` permet de lier un objet de type `GenericDocument` qui décrit l'article (ceci sera notamment utilisé pour intégrer au PDM des fichiers de visualisation 3D au format 3DXML), `GenericDocument` étant également utilisé pour modéliser les fichiers de configuration.

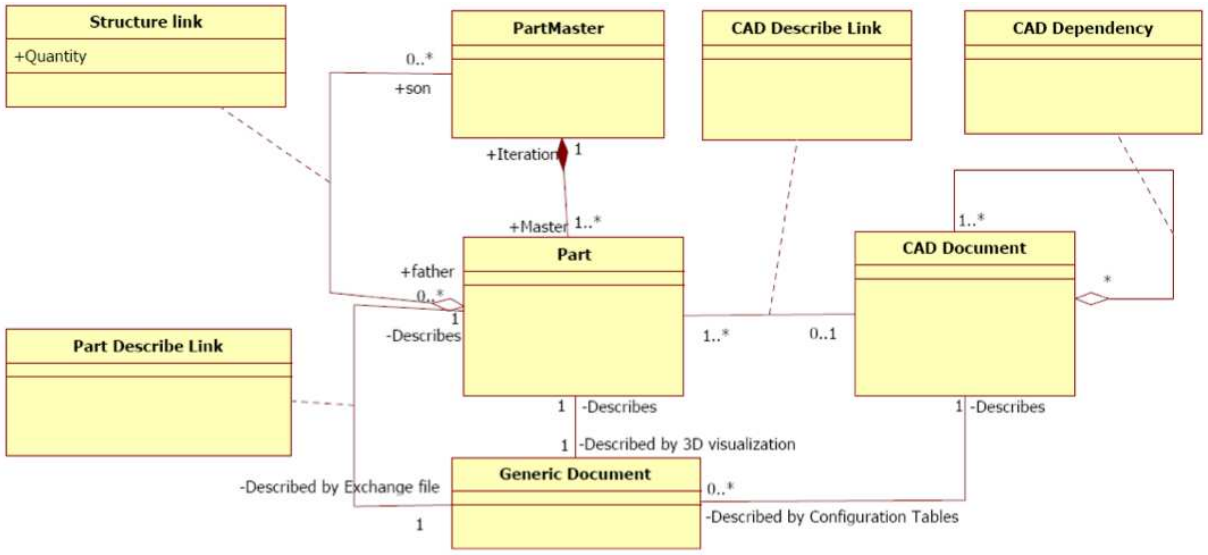


FIGURE 3.2.7.: Méta-modèle d'unification pour l'interopérabilité CAO/PDM

3.2.7. Conclusions méthodologiques

Ces premiers travaux sur l'interopérabilité nous ont permis de formuler les conclusions suivantes :

- c'est la **recherche préalable des similarités sémantiques entre informations** à échanger **qui guide la mise en œuvre du méta-modèle d'unification**. Ces similarités sont obtenues en balayant l'ensemble des informations et en évaluant leurs différences sémantiques deux-à-deux. Ce processus est d'autant plus long que le flux sémantique est étendu et les informations nombreuses ;
- **si** le problème d'interopérabilité revient à échanger des informations dont **la distance sémantique est nulle** (*i.e.* les informations sont des synonymes), **la démarche d'unification se rapproche *de facto* de celle d'intégration** : le méta-modèle d'unification sera équivalent aux modèles de données utilisés dans les deux SI ;
- **si les distances sémantiques minimales sont faibles** sans pouvoir toutefois être considérées comme négligeables, on parlera de *nuances sémantiques* : **il faudra alors modéliser (et implémenter) un ensemble restreint de règles simples permettant au médiateur d'interpréter ces nuances** ;
- **plus les distances sémantiques minimales seront grandes, plus le nombre et/ou la complexité des règles sera important**. Dans ce cas, la démarche d'unification tend à se rapprocher de celle de la fédération : les règles implémentées dans le médiateur se rapprochent des règles d'inférence permettant de raisonner sur les ontologies définies dans l'approche fédérative (à la différence près que ces règles sont statiques du point de vue unification et dynamiques du point de vue fédératif) ;
- dans le cas où les distances sémantiques minimales sont très grandes (voire infinies), le problème d'interopérabilité n'admettra pas de solution. Cette situation extrême ne peut *a priori* se produire dans le cas du PLM. Ainsi, **quelles que soient les différences entre les hommes** (différences de langue, de culture, de métier, de nationalité), **les organisations** (taille, implantation, processus etc.) **ou encore les SI utilisés, il est un élément commun : le produit**. En conséquence, ce « point commun » se traduira nécessairement par des convergences sémantiques et nous pouvons conclure que **tous les problèmes d'interopérabilité dans le domaine du PLM admettent une solution : il existe au moins un méta-modèle d'unification capable de réconcililer la sémantique des SI du domaine du PLM vis-à-vis de n'importe lequel de ces problèmes.**

Ces recherches ont été poursuivies dans le cadre de problèmes d'interopérabilité qui impliquent des SI de deux phases du cycle de vie du produit, la conception et la production. Nous avons en particulier cherché à utiliser et mettre en oeuvre un modèle standard extensible. Les résultats de ce travail sont présentés dans la section 3.3.

3.3. Médiation inter-phase : exemple de l'interopérabilité PDM/ERP

Dans cette section, nous présentons un exemple d'interopérabilité PDM/ERP. Nous mettons l'accent sur la mise en œuvre de la norme PLCS pour la construction du méta-modèle d'unification. La structuration de cette partie est la même que celle déjà utilisée

pour l'interopérabilité CAO/PDM, qui reprend *in extenso* les conclusions méthodologiques que nous avons déjà formulées.

3.3.1. Situation initiale - Origine du problème d'interopérabilité

L'AP239 est complexe, sa mise en œuvre particulièrement lourde. Pour cette raison, nous présentons un cas d'étude simple qui permettra de mettre en lumière les aspects méthodologiques ainsi que les points essentiels de l'AP239 relatifs à la structuration du produit. Le produit utilisé est celui-ci : une roue avant de bicyclette, que l'on suppose constituée de deux composants seulement, un pneu et une jante (voir figure 3.3.1).

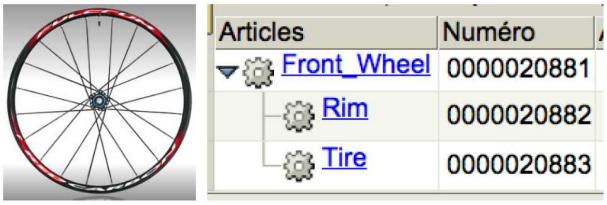


FIGURE 3.3.1.: Cas d'étude pour l'interopérabilité PDM/ERP

Sur cette figure est représentée, à côté de la photographie du produit, la nomencature de conception telle qu'elle est définie dans le système PDM. Le système ERP pilote quant à lui la production en série de cet ensemble. Le besoin que nous avons exprimé est le suivant : comment assurer la cohérence entre les informations qui décrivent le produit dans le PDM et dans l'ERP en vue de la fabrication du produit ? Comment automatiser le transfert de la structure produit du PDM vers l'ERP ? Les deux problèmes que nous souhaitons étudier sont donc les suivants :

- transférer les informations du système PDM au système ERP, c'est-à-dire un problème de type *transfert* d'informations ;
- réconcilier les informations provenant de ces deux systèmes en vue de les archiver, soit un problème de type *empaquetage* d'informations.

Comme pour le cas de l'interopérabilité CAO/PDM vu dans la section précédente, c'est une situation initiale caractérisée par l'hétérogénéité du SI qui est à l'origine du problème d'interopérabilité :

- le système PDM utilisé est Windchill, édité par PTC ;
- le système ERP disponible est OpenERP [4], édité par OpenERP S.A.

4. http ://www.openerp.com

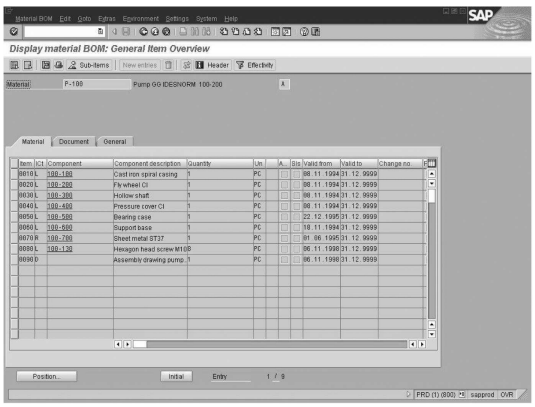
3.3.2. Phases du cycle de vie du produit

Les phases du cycle de vie du produit impliquées dans ce problème d'interopérabilité sont les phases de conception (le PDM est un système relevant du métier de la conception) et de production (l'ERP étant utilisé, entre autres, dans cette phase du cycle de vie). Nous sommes donc confronté à un problème d'interopérabilité inter-phase. Nous avons fait remarquer dans le chapitre 2 que dans ce cas le SI est le siège de flux sémantiques de plus grande étendue que pour les problèmes de type intra-phase.

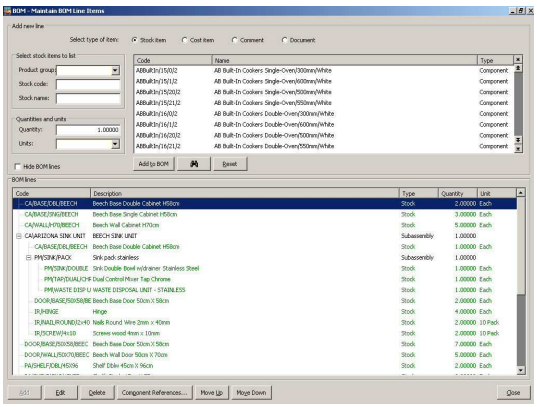
Nous recherchons, dans la section suivante, les rapprochements sémantiques entre informations. Compte tenu de la remarque précédente, on peut s'attendre à ce que les informations à échanger présentent un degré de similarité moindre que pour le cas de l'interopérabilité CAO/PDM.

3.3.3. Similarités sémantiques entre les informations : EBOM / MBOM

Les systèmes ERP (par exemple Sage, SAP ou Baan), utilisés pour la gestion des processus de fabrication (MRP), nécessitent aussi comme entrée une décomposition du produit (voir tableau 3.3.1). Cette structure est appelée nomenclature de fabrication ou Manufacturing Bill Of Material (MBOM). Ce type de structure caractérise les dépendances temporelles durant le processus de fabrication du produit ou encore la succession des étapes nécessaires à la réalisation du produit physique. Les liens de structure portent une information relative à la quantité des composants nécessaires à la fabrication du produit, information capitale pour le calcul des besoins. Dans la MBOM, les articles fantômes ont pour objectif, entre autres, de faciliter la fabrication des produits ayant une forte diversité. Au contraire de la EBOM, la définition et la compréhension de la MBOM sont largement partagées depuis [Orlicky, 1973] : le niveau de granularité de la décomposition est l'élément unique identifié dans un stock. La sémantique liée à la MBOM est sans ambiguïté : les modules MRP des systèmes ERP basent leur implémentation de la MBOM sur cette acception.



SAP



Sage

TABLE 3.3.1.: Nomenclature de fabrication dans les systèmes ERP

Nous remarquons un rapprochement sémantique évident entre la EBOM définie dans le PDM et la MBOM de l'ERP : les similarités dans la représentation des articles et des liens de structure sont le point de convergence sémantique qui seront au coeur du méta-modèle d'unification. Il faudra cependant prendre garde à choisir un méta-modèle d'unification qui permette une représentation multi-vues du produit.

3.3.4. Nature et granularité des informations à échanger

La sémantique de la structure produit PDM ainsi que celle de l'ERP devront pouvoir être transférées d'un système à l'autre, ou unifiées en vue d'un empaquetage. Comme dans le cas de l'interopérabilité CAO/PDM, ces informations présentent une granularité grossière mais, *a contrario*, concernent des phases du cycle de vie du produit pour lesquelles les distances sémantiques sont plus importantes que pour l'interopérabilité CAO/PDM.

Dans ce cas de figure (problème inter-phase conception/production, granularité grossière, représentation multi-vues du produit), la cartographie de la figure 2.3.5 fait apparaître le standard PLCS comme la solution la plus adaptée dans un contexte multi-échelle sémantique et temporelle. Remarquons qu'il serait possible de réaliser (en terme d'implémentation) cette interopérabilité en utilisant la structuration du produit telle qu'elle est définie dans le PDM Schema ou l'AP214 : l'architecture résultante porterait alors une forte contradiction sémantique interne.

Dans les sections suivantes, nous nous intéressons donc à la modélisation du produit dans PLCS afin d'identifier les connexions possibles entre les informations du PDM, de l'ERP et du méta-modèle d'unification.

3.3.5. Modélisation du produit dans STEP PLCS

Le modèle de données générique défini dans PLCS [ISO-10303-239, 2005] permet de représenter le produit et sa structuration. Dans un premier temps, nous nous intéressons à la sémantique du produit telle qu'elle est définie dans PLCS puis, dans un second temps, étudions le modèle de représentation multi-vues. La modélisation des données dans STEP est réalisée dans le langage EXPRESS dont une présentation détaillée est donnée dans l'appendice B.

Le produit La modélisation du produit dans PLCS repose sur le module d'application 10303-1017 :2004 de la norme STEP (voir l'appendice A pour l'indexation des fascicules de la norme STEP) qui définit la sémantique permettant d'identifier un produit. D'après ce module, les produits représentés par l'entité `Product` (voir figure 3.3.2) sont :

- les produits existant dans le monde réel ;
- les produits pouvant être amenés à exister à la suite d'un processus de réalisation (ceci incluant les pièces et les documents) ;
- les produits qui sont des fonctions.

Cette acception très large de la notion de produit permet de modéliser, *via* l'entité `Product`, l'ensemble des données relatives au produit (notes de calcul, documents CAO, spécifications, matières premières etc.). Afin de différencier ces types de produits, chaque instance de `Product` peut être affectée à une catégorie de produits, modélisée par l'entité `Product_category` (il est possible de hiérarchiser les catégories de produits avec l'entité `Product_category_hierarchy`). Il n'existe pas de catégories de produit pré-définies : elles sont définies par l'utilisateur en fonction de ses processus métiers.

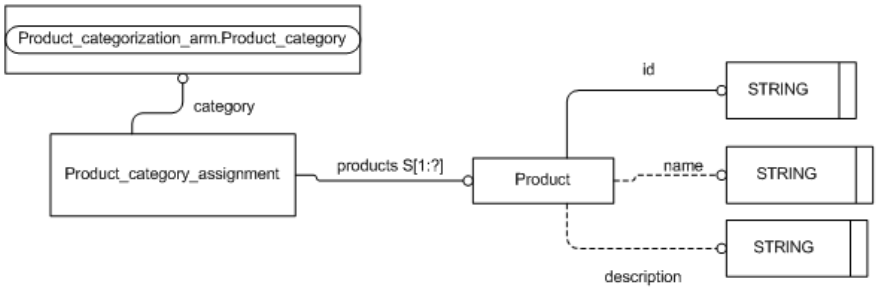


FIGURE 3.3.2.: Modélisation du produit dans STEP PLCS

L'historique du produit est représenté par un ensemble d'instances de l'entité `Product_version` (voir figure 3.3.3). L'identification des versions d'un produit est laissée à l'initiative de l'utilisateur : elle est définie par l'attribut `id` qui est du type `STRING` (une chaîne de caractères).

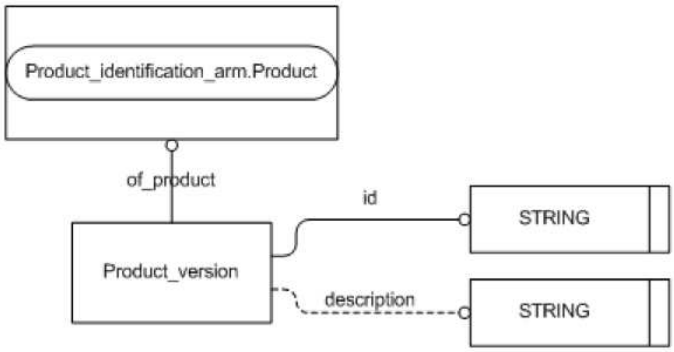


FIGURE 3.3.3.: Modélisation d'une version d'un produit

Cette généricité permet de composer avec les nombreuses conventions utilisées par les logiciels de gestion de donnée (ex : A.1, A1, 1.1, 1.1.1 etc.).

Les composants La norme PLCS définit une sous-classe particulière de l'entité `Product` : l'entité `Part`. Nous traduirons ce terme par *composant*. Le modèle EXPRESS-M de cette entité est donné ci dessous :

```
ENTITY Part_version
  SUBTYPE OF (Product_version);
  SELF\Product_version.of_product : Part;
END_ENTITY;
```

Cette spécialisation de l'entité `Product` n'ajoute ni attribut ni règle : *a priori*, la différence sémantique n'apparaît pas clairement. Il convient donc de bien lire la norme pour saisir toutes les subtilités qu'implique cette spécialisation. En effet, bien que le terme « Part » soit utilisé dans des logiciels CAO (Catia V5, SolidWorks) et souvent traduit en français par *pièce*, la sémantique PLCS montrer que cette traduction n'est pas pertinente dans ce cas. D'après le module d'application 10303-1022 :2004, l'entité `Part` est une sous-classe de l'entité `Product` permettant de modéliser :

- des pièces, c'est-à-dire *des objets discrets pouvant être réalisés à la suite d'un processus de fabrication*. Cette définition se situe à l'interface des métiers de la conception (voir 1.2.1.1) et de la production (voir 1.2.1.2). Tous les articles fabriqués par l'entreprise entrent dans cette catégorie ;
- des *matières premières*, c'est-à-dire *des matériaux bruts ou partiellement mis en formes nécessaires à l'initialisation d'un processus de fabrication*. Cette définition s'inscrit dans la sémantique du métier de la production vu dans la section 1.2.1.2, et spécialement le calcul des besoins nets : ces matières premières correspondent au dernier niveau de la nomenclature de fabrication. Exemples de matières premières : une tôle brute, des composants standards (vis, roulements), des éléments achetés ou standards (moteurs électriques, éléments de guidage etc.) ;

- des *matériaux non dénombrables*, i.e. *ne pouvant être (ou n'ayant pas besoin d'être) comptés*. Les composants de cette catégorie sont alors quantifiés par des unités. Exemple de matériaux non dénombrables : l'huile, l'eau, le sable, l'air comprimé etc. Ces éléments sont particulièrement importants pour les phases de maintenance du produit (la quantité d'huile nécessaire au fonctionnement du produit est une donnée qui peut être nécessaire pour une opération de vidange).

Ces pièces et composants servent de base à la représentation multi-vues du produit.

3.3.6. Représentation multi-vues dans STEP PLCS

Représentation multi-vues D'après le schéma EXPRESS de l'AP239, c'est l'entité `View_definition_context` qui permet de définir un point de vue sur le produit :

```
ENTITY View_definition_context;
  application_domain : STRING;
  life_cycle_stage : STRING;
  description : OPTIONAL STRING;
END_ENTITY;
```

Cette entité porte plusieurs attributs :

- `application_domain` : une chaîne de caractères qui définit le contexte sémantique considéré (par exemple « étude d'assemblage », « maquette numérique », « conception électrique », « conception mécanique » ou « planification des processus de fabrication » sont des exemples de valeur pour cette chaîne) ;
- `life_cycle_stage` : une chaîne de caractères qui identifie une étape du cycle de vie du produit (par exemple « conception », « production », « phase de recyclage ») ;
- la `description` de cette vue est une chaîne de caractères optionnelle qui permet de fournir des informations sur l'entité `View_definition_context`.

Cette entité `View_definition_context` définit donc un « point de vue » dans le plan (`lifecycle_stage`, `application_domain`). Nous définissons deux vues « PDM_View » et « ERP_View » correspondant aux points de vue de chacun des deux systèmes (voir figure 3.3.4).

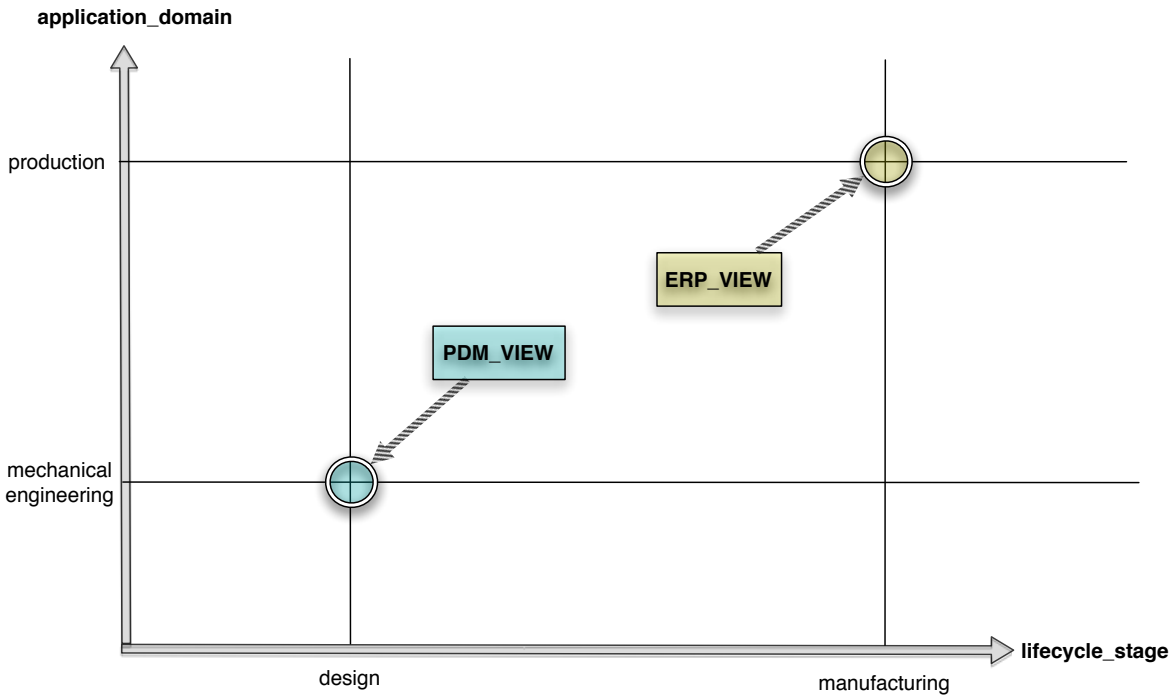


FIGURE 3.3.4.: Modélisation multi-vues dans STEP PLCS

Ainsi, les expressions « vue conception » ou « vue production » apparaissent-t-elles incompletes vis-à-vis de la sémantique de la représentation multi-vues dans PLCS.

Templates et Capabilities Un ensemble de règles d'utilisation spécifient comment instancier le modèle de données générique de l'AP239. La conformité vis-à-vis ces prescriptions permet de garantir le fait que l'implémentation de PLCS soit compatible avec celle de ses partenaires, clients ou fournisseurs. Ces règles métiers quant à l'utilisation de PLCS sont compilées dans des *Data Exchange Set* [5] (DEX). Chaque DEX définit :

- un ensemble de *capabilities* qui fournissent un guide d'utilisation relatif au déploiement du modèle de données PLCS pour représenter une information donnée ;
- pour chaque *capability,* le *template* associé qui précise le modèle de données. Ces « patrons » spécifient l'exacte instanciation d'un ensemble d'entités et de valeurs d'attributs. C'est une spécification non-ambiguë de la manière d'utiliser un ensemble d'entités, d'attributs et de données de référence PLCS dans un but caractérisé par une *capability.*

Dans un sens, on peut considéréer que les *capabilities* caractérisent une approche fonctionnelle alors que les *templates* détaillent le modèle de donnée destiné à être implémenté. Pour

5. http ://www.plcs-resources.org/plcs/dexlib/dex_index.htm

le méta-modèle d'unification dédié à l'interopérabilité PDM/ERP, nous avons identifié les *templates* `representing_part` (voir figure 3.3.5) et `representing_assembly_structure` (voir figure 3.3.6) comme étant nécessaires pour structurer le produit suivant une approche multi-vues.

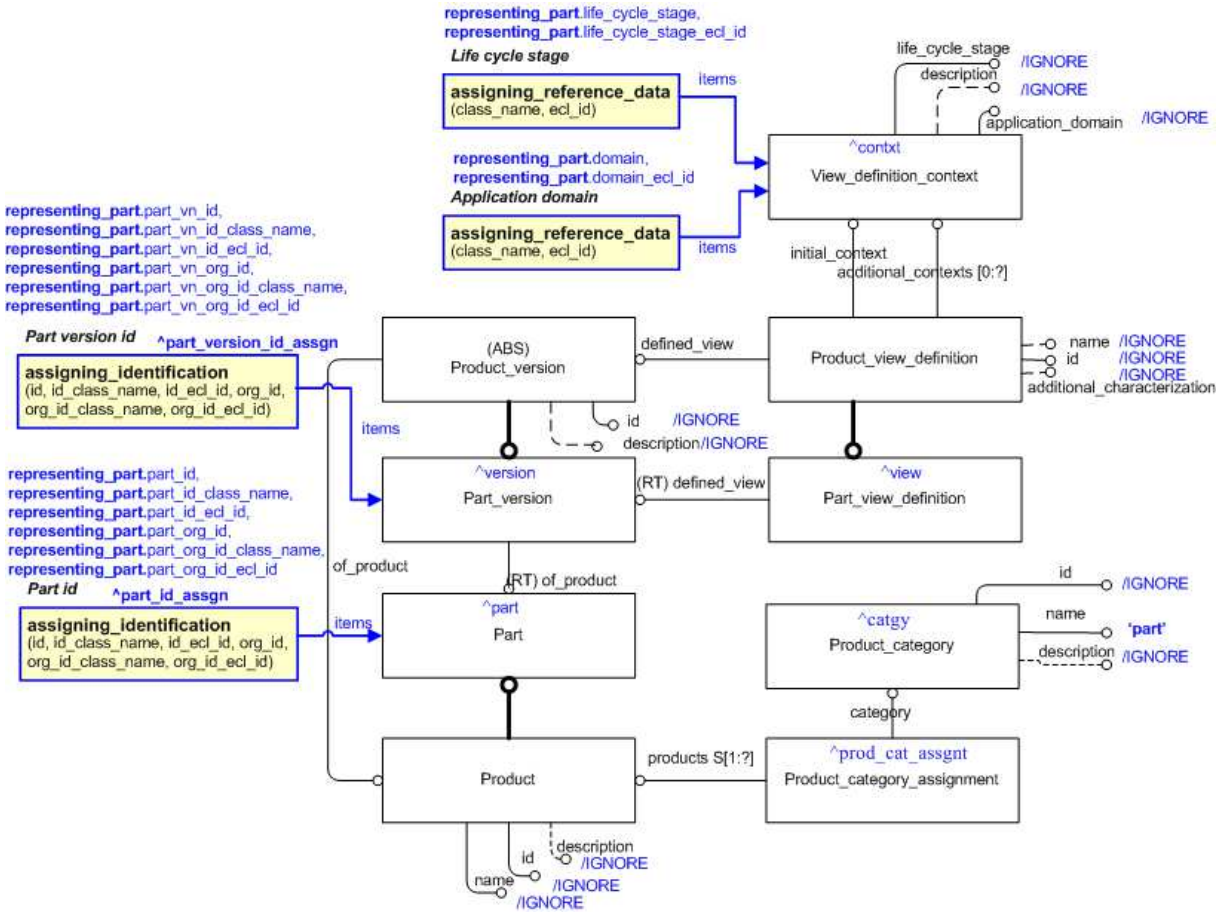


FIGURE 3.3.5.: Modèle EXPRESS-G `representing_part`

Dans le diagramme EXPRESS-G ci-dessous, la représentation des liens de structure apparaît : ce sont les entités `Next_assembly_usage` qui permettent de représenter les relations de type parent/enfant.

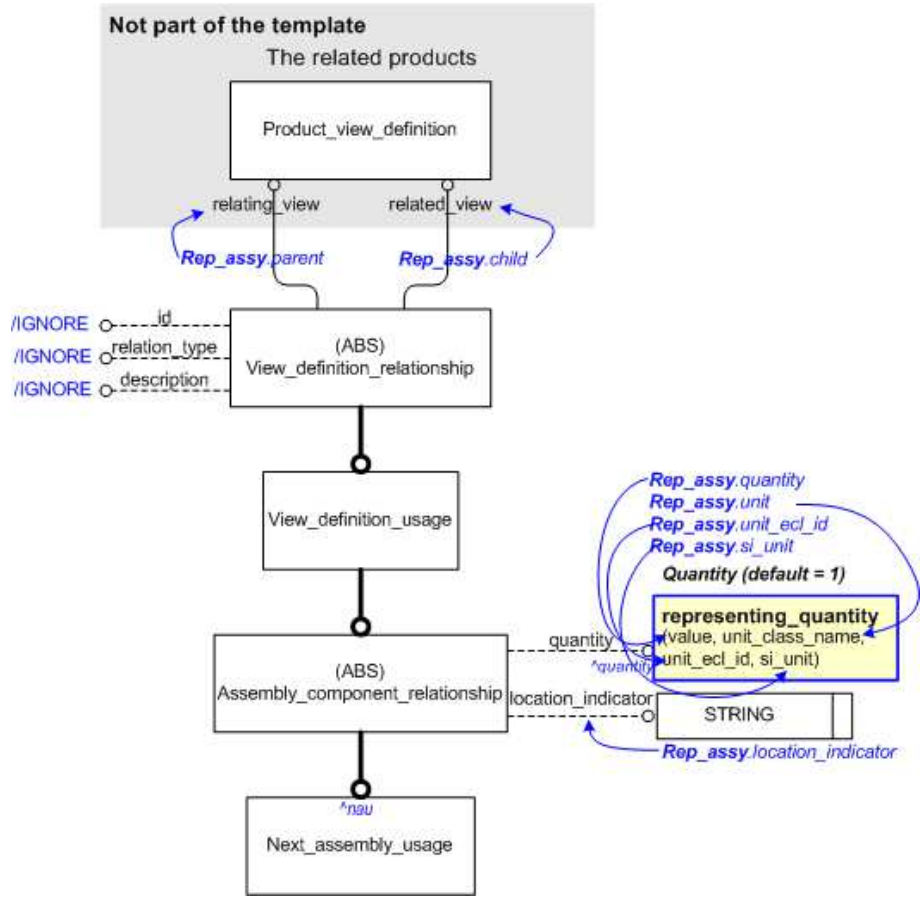


FIGURE 3.3.6.: Modèle EXPRESS-G `representing_assembly_structure`

Après avoir choisi les *templates* à utiliser, il faut rechercher un moyen de les mettre en relation, c'est-à-dire qu'il faut trouver le lien entre tous ces templates. Après cela, il devient possible de construire le méta-modèle d'unification et de le représenter sous la forme d'un diagramme UML dans un objectif d'implémentation. Ces deux points sont décrits dans la section suivante.

3.3.7. Diagramme UML du méta-modèle d'unification

D'après les deux figures précédentes, nous constatons que c'est par l'intermédiaire de l'entité `Produt_view_definition` que peut s'opérér la réunion des deux modèles précédents : cette entité est présente dans les deux *templates* choisis. Le résultat de l'union de ces deux modèles est représenté sous la forme d'un diagramme de classe UML (voir figure 3.3.7) qui sera implémenté dans le chapitre 4.

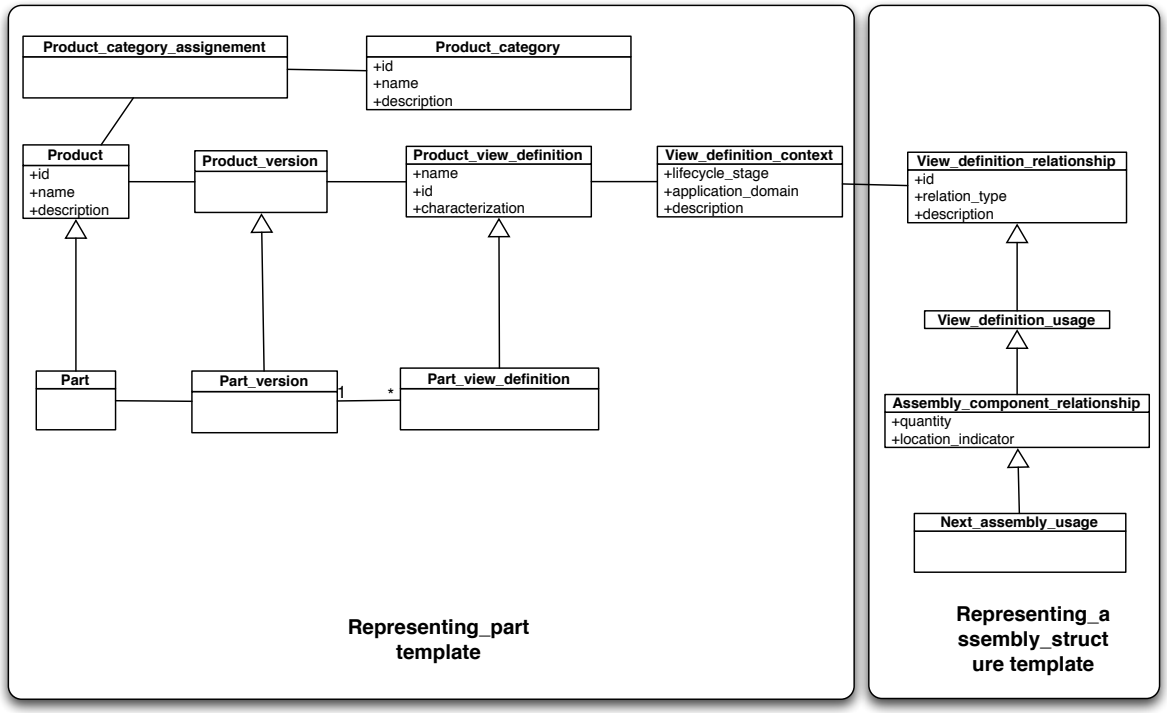


FIGURE 3.3.7.: Diagramme UML pour le méta-modèle d'unification

La définition du diagramme UML du méta-modèle d'unification conclut la première phase de ce travail. Il convient par la suite d'associer, d'un point de vue sémantique, chacune des informations provenant des deux systèmes avec les classes du méta-modèle d'unification. C'est là que le travail de la recherche des similarités sémantiques prend toute son importance : si des similarités existent, qu'elles ont bien été identifiées alors le travail de mise en relation des informations sera simplifié et il deviendra aisé de faire le lien entre le méta-modèle de la figure 3.3.7 et le modèle de graphe pour une structure arborescente présenté sur la figure 3.2.5 (sachant de plus que les structures produit du PDM et de l'ERP sont équivalentes d'un point de vue topologique).

3.3.8. Conclusions méthodologiques

La mise en oeuvre de PLCS pour résoudre le problème d'interopérabilité PDM/ERP nous amène aux prescriptions méthodologiques suivantes. Nous estimons qu'il faut :

- commencer la lecture des spécifications des standards extensibles par le chapitre dédié aux *capabilities* ;
- ensuite, identifier et étudier les *capabilities* permettant de représenter les informations préalablement identifiées ;
- étudier alors les *templates* associés et s'appropier leur sémantique ;

- enfin, fusionner ces *templates* en un seul et même diagramme UML qui sera le méta-modèle d'unification.

En définitive, il s'agit de n'utiliser que la sous-partie de PLCS qui modélise la sémantique désirée, les *templates* faisant de chaque DEX un ensemble fortement modulaire. Nous signalons par ailleurs la difficulté à s'approprier le protocole d'application de la norme STEP : les spécifications éditées par l'ISO ne constituent aucunement un guide quant à l'implémentation, et les milliers d'informations de la norme peuvent décourager de s'y plonger. S'engager dans l'utilisation de ce standard pour résoudre un problème d'interopérabilité requiert donc du temps, et il est souhaitable, si possible, de s'entourer de compétences spécifiques dans ce domaine pour accélérer le processus d'appropriation. D'autre part, nous conseillons de choisir un produit simple pour faciliter au maximum ce travail d'appropriation, et ainsi éviter d'ajouter à la complexité de la norme des difficultés liées à la complexité du produit.

3.4. Un modèle basé sur des *Tags sémantiques* pour la structuration du produit

Au travers des processus de modélisation exposés plus haut, et après l'analyse fine des modèles de données implémentés dans les outils de CAO, les ERP, les systèmes PDM ou bien encore le modèle de donnée générique PLCS, nous nous sommes rendus compte que les difficultés auxquelles nous étions confrontées provenaient pour l'essentiel d'un travail d'extraction de la sémantique contenue dans une structure produit afin de la mettre en correspondance avec celle du méta-modèle d'unification. Nous avons donc entrepris d'étudier ce problème plus spécifiquement. Le résultat de cette étude est un modèle de séparation des espaces sémantiques/espaces des matières d'œuvre, appelé *Semantic Tag Product Model*, associé à un processus d'inférence des multiples structures produit appelé *structuration sémantique du produit* (*Semantic Product Breakdown*). Dans la section 3.4.1 nous mettons en lumière le problème soulevé ainsi que notre approche pour le résoudre. Puis, dans la section 3.4.2, nous présentons la notion de *tag* issue des technologies du web sémantique. Cette manière d'organiser l'information est transposée au domaine de la conception/production des produits industriels dans les sections 3.4.3 et 3.4.4.

3.4.1. La sémantique mélangée à la matière d'œuvre des processus de conception et de fabrication

Après examen des modèles de EBOM ou MBOM utilisés dans les systèmes d'information relevant du PLM, nous constatons que ces nomenclatures sont constituées de deux ensembles d'éléments de nature très différente : les *items* et les *liens*.

Items Les *items* correspondent à des nœuds dans la structure arborescente. Les items de la nomenclature de conception sont les documents 3D qui définissent la géométrie du produit. Les items de la nomenclature de fabrication sont les articles de production. Les articles de production constituent les éléments d'un stock, le lieu de stockage pouvant être un dépôt, un centre logistique, une palette située entre deux postes de charge etc. Par analogie, nous considérons que les items de conception peuvent aussi être considérés comme faisant partie d'un stock : la base de données dans laquelle ils sont archivés, ou alors le disque dur du concepteur. Ces articles de conception ou de production sont donc caractérisés par un *volume* et entraînent un *coût de stockage*. Nous les réunissons dans la notion de matière d'œuvre des processus de conception et de fabrication.

Définition 19. La matière d'œuvre des processus de conception et de fabrication englobe tous les éléments de l'entreprise provenant d'un stock, transformés par un processus de conception ou de production, et dont le résultat de ce processus a vocation à être conservé pour une utilisation ultérieure (stock ou en-cours).

Les items sont chargés d'une sémantique qui leur est propre, généralement stockée dans les systèmes d'information sous forme de métadonnées : par exemple la masse, le nom du fournisseur ou la quantité disponible en stock pour l'ERP, la taille du fichier CAO associé, la mention fabriqué/acheté, le nom du concepteur ou l'état de maturité dans le PDM.

Liens En plus des items, les EBOM et MBOM font apparaître des liens hiérarchiques entre items de type parent/enfant (ce lien pouvant en outre être précisé par une quantité). Dans la logique MRP de la nomenclature de production, les liens figurent une relation *temporelle* : avant de pouvoir envisager la fabrication de l'élément parent, il faut s'assurer de la disponibilité en quantité suffisante de tous les éléments enfants. Du point de vue de la conception, il n'existe pas de sémantique univoque quant aux liens de dépendance parent/enfants. La création, la modification ou la maintenance de ces liens relève *in fine* d'une activité de *regroupement* :

- en production, on regroupe dans un même niveau de nomenclature tous les articles devant être disponibles au même moment pouvoir démarrer l'opération de fabrication ;
- en conception, on regroupe l'ensemble des items participant à la réalisation d'une même fonction, ou alors l'ensemble des items appartenant à un même organe ;
- du point de vue des achats, on regroupe l'ensemble des articles achetés à un même fournisseur ;
- en maintenance, on regroupe les items devant être associés à une même opération de maintenance.

Cette liste des opérations de regroupement n'est pas exhaustive, mais illustre néanmoins le fait que les liens se distinguent des items dans le sens où ils portent une sémantique qui regroupe des items. Enfin, d'un point de vue des systèmes d'information, les liens sont stockés dans des bases de données. Néanmoins, l'ordre de grandeur de l'espace de stockage nécessaire n'est pas le même que celui nécessaire à la définition des matière

d'œuvre (à titre d'exemple, les fichiers 3D nécessaires à la définition du modèle DM08 sont de 48Mb, le fichier d'assemblage seulement de 23kb).

Séparation explicite de la sémantique Au cours de son cycle de vie, le produit est représenté suivant n structures produit correspondant à autant de points de vue métier (conception, fabrication, mais aussi achats, maintenance etc.) :

- la matières d'œuvre des processus de développement du produit est ventilée sur ces n structures produit. Certains items peuvent apparaître dans plusieurs de ces structures produit, induisant une *redondance de l'information* ;
- les n vues sont *a priori* disjointes ;
- la sémantique liée au produit est éclatée suivant toutes ces vues. Elle est très largement implicite dès lors que l'on s'intéresse à la vue conception.

La difficulté de l'interopérabilité est donc la reconstruction de ce puzzle à plusieurs dimensions composé d'articles et de sémantique aux contours flous. Nous proposons donc un modèle qui soit orienté suivant une *approche explicite top/down* : nous considérons que toutes les informations et la sémantique liées au produit peuvent être contenues dans 2 espaces distincts liés par des relations élémentaires : l'*espace des matières d'œuvre* et l'*espace sémantique*. Ce modèle doit se substituer à la représentation des n structures produits qui doivent pouvoir être générées à partir d'une inférence des informations de l'espace sémantique. Cette déduction pourra être considérée comme un processus de projection de l'espace complet items/sémantique sur un espace de dimension moindre qui correspondrait à la vue désirée (voir figure 3.4.1).

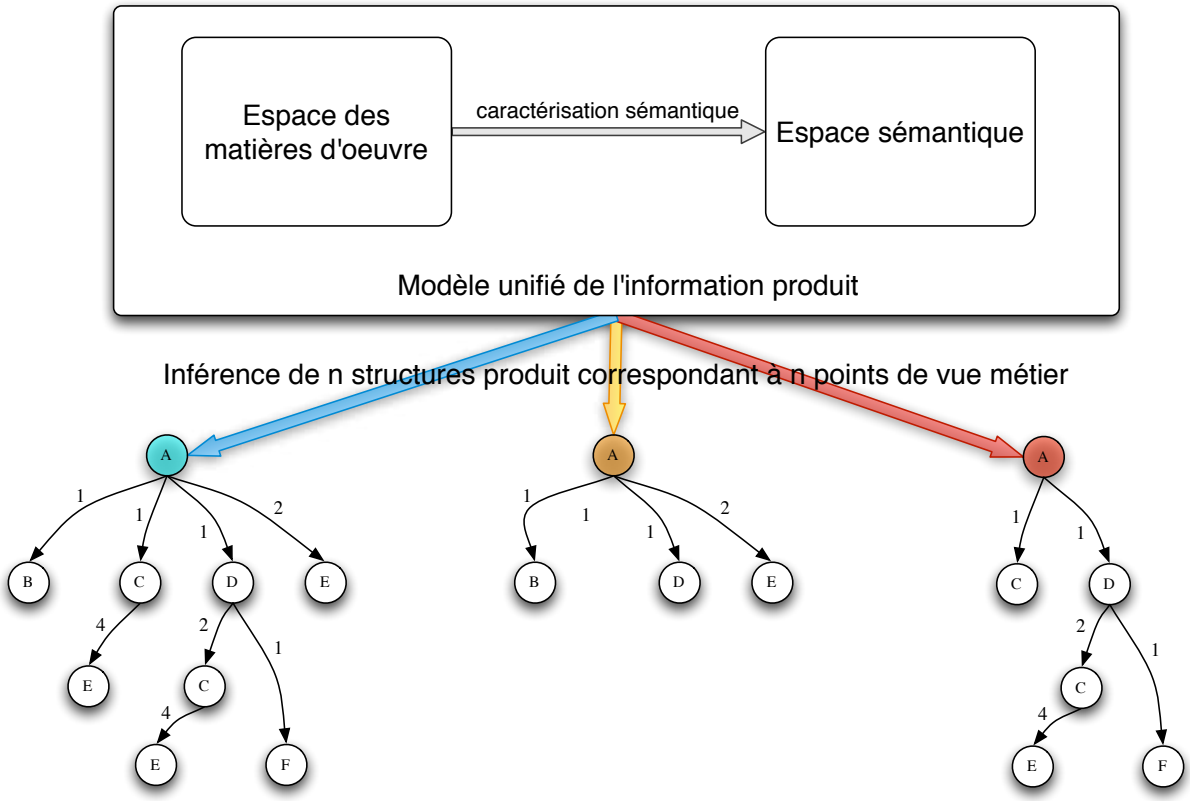


FIGURE 3.4.1.: Espace des matières d'œuvre/espace sémantique

Pour assurer le lien entre les deux espaces, nous proposons d'utiliser la notion de *tagging* ou *annotation sémantique* que nous présentons dans la section suivante.

3.4.2. Web collaboratif et activité de *tagging*

La notion de web collaboratif est apparue au moment où les technologies de l'information, en particulier celles liées au développement de l'Internet, ont transformé un internaute consommateur d'informations en producteur d'informations ayant vocation à être partagées largement [Mathes, 2004]. Lorsque des milliers d'acteurs participent à la création de contenu dans des espaces collaboratifs, l'organisation et la classification de ces informations revêtent une importance stratégique pour les opérateurs de ces plate-formes : les centaines de milliers d'informations disponibles doivent pouvoir être triées et rendues facilement accessibles afin d'amener l'information pertinente à l'utilisateur. Ainsi, une organisation préconfigurée n'est pas pertinente pour la classification de millions d'informations dont le contenu n'est pas connu *a priori*. Ces dernières années, des systèmes à base d'étiquettes sémantiques ont été popularisés. D'après [Marlow et al., 2006], *ces systèmes permettent aux utilisateurs/contributeurs d'associer spontanément des mots-clés*

à des ressources numériques (pages web, images, vidéos etc.) chacun bénéficiant de la sémantique portée par ces annotations pour organiser l'ensemble des documents. Par exemple, un internaute souhaitant partager des photographies de monuments parisiens utilisera par exemple les tags « photographie », « Paris » et « monument » pour décrire cette donnée. Les tags seront ensuite traités par le système informatique comme des *clés* permettant d'identifier les ressources pertinentes vis-à-vis de la requête d'un autre internaute. Dans ce procédé de tagging, c'est le contributeur qui extrait dans un premier temps la sémantique de la ressource qu'il/elle désire partager, chacun pouvant ensuite la compléter par ses propres annotations. Le site de partage de signets Delicious [6] est le premier à avoir industrialisé ce procédé. Ces systèmes de *tagging* permettent ainsi d'améliorer la recherche d'informations ou encore la détection d'informations non-pertinentes (*spams*) tout en introduisant de *nouvelles modalités dans la communication sociale et des opportunités pour l'extraction de données* (*data mining*) [Marlow et al., 2006].

C'est en observant ce phénomène né de l'internet et du développement des réseaux sociaux que nous avons pu constater des analogies entre ces systèmes et les informations numériques manipulées dans un contexte PLM :

- dans les deux cas, la collaboration autour de ces informations numériques relève d'activités de collaboration complexes : de nombreux acteurs peuvent créer/lire/modifier/diffuser de nombreuses informations ;
- dans les deux cas, l'organisation et la structuration des informations revêtent un caractère stratégique ;
- dans les deux cas, la sémantique portée par les informations numériques est hors de portée d'un traitement automatique par l'ordinateur.

Dans ce contexte, nous nous sommes intéressés à la transposition du procédé de tagging du domaine du PLM, avec pour objectif de répondre à la question : le modèle de double-espace matière d'œuvre/sémantique présenté dans la figure 3.4.1 pourrait-il être réalisé par des étiquettes sémantiques ? Dans la section suivante, nous proposons donc de présenter notre approche en l'illustrant à l'aide d'un cas d'étude simple.

3.4.3. Cas d'étude pour le modèle de tag sémantique

Nous proposons dans cette partie un cas d'étude permettant d'illustrer la construction du modèle. Nous avons souhaité que ce cas d'étude soit constitué du plus petit nombre possible de composants, mais également qu'il soit composé d'éléments standards et fabriqués. Nous proposons donc le produit constitué de deux tôles rectangulaires découpées **Sheet 1** et **Sheet 2**, percées puis assemblées par un boulon **Bolt** composé d'une vis **Screw 3** et d'un écrou **Nut 4**. Du point de vue de la fabrication, les éléments **Sheet 1** et **Sheet 2** sont obtenus à partir de la découpe de deux tôles brutes **Sheet 1 RM** et **Sheet 2 RM**. Pour ce produit, deux EBOM et MBOM possibles sont représentées sur la figure 3.4.2.

6. http ://delicious.com

La critique du choix de ces nomenclatures ne peut se faire indépendamment du contexte de conception et de fabrication : dans l'objet de notre étude, centré sur les problèmes d'interopérabilité de systèmes existants, nous considérons que ces deux nomenclatures sont données.

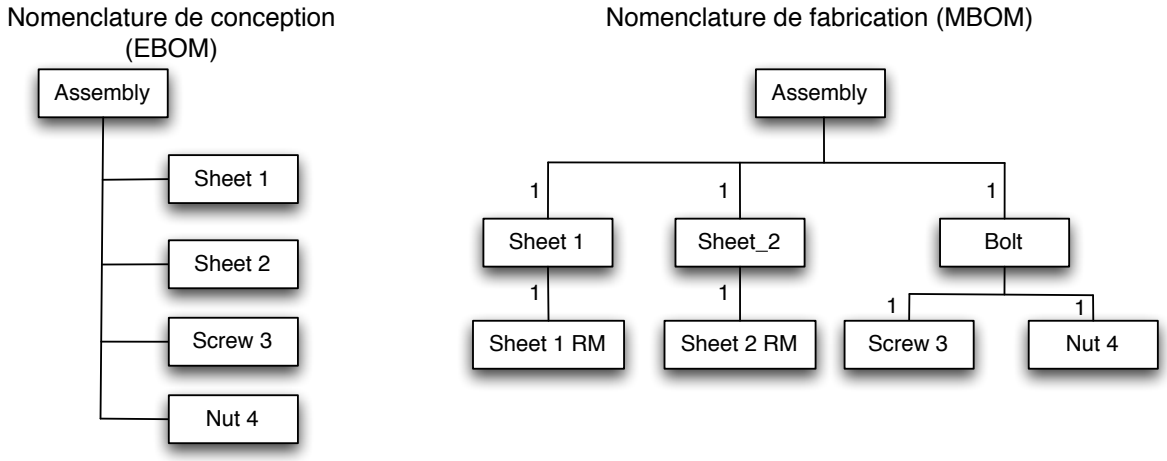


FIGURE 3.4.2.: Cas d'étude pour le modèle de tags sémantiques

3.4.4. Modèle de tag sémantique

Construction de l'espace des matières d'œuvre L'espace des matières d'euvre est construit comme étant l'ensemble des items présents dans les différentes structures produit :

$$M = (Assembly,\ Sheet\,1,\ Sheet\,2,\ Screw3,\ Nut\,4,\ Sheet\,1\,RM,\ Sheet\,2\,RM,\ Bolt)$$

Remarque : nous avons confondu dans l'ensemble ci-dessus les articles **Sheet 1** de l'EBOM et **Sheet 1** de la MBOM, leur distance sémantique pouvant être considérée comme très faible devant les autres distances en jeu.

Construction de l'espace sémantique Dans l'espace sémantique, nous considérons dans un premier temps les deux domaines sémantiques « Design » et « Manufacturing », caractérisant le fait que des éléments de l'ensemble M proviennent plutôt de l'un ou de l'autre de ces domaines.

$$S = (Design, Manufacturing)$$

Construction de lien matière d'œuvre/espace sémantique Le lien M/S est une étiquette sémantique : à chaque élément de l'ensemble M, on associe une et/ou l'autre des étiquettes sémantiques « Design » et « Manufacturing ». Par exemple, l'élément **Assembly** portera les étiquettes « Design » et « Manufacturing », l'élément **Bolt** portera l'unique étiquette « Manufacturing » (sachant qu'il n'apparaît que dans la MBOM).

Le diagramme de classe de la figure 3.4.3 représente le modèle UML ainsi défini. L'association entre un item et un tag est du type `0..*` : chaque item peut porter plusieurs étiquettes, chaque étiquette est portée par 0 ou plusieurs item.

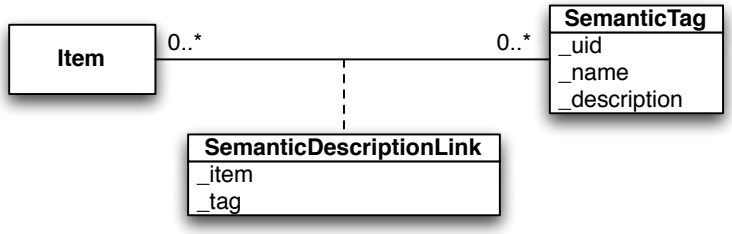


FIGURE 3.4.3.: Un premier modèle de données pour le tag sémantique

D'après [Sen et al., 2006], dans un contexte de tagging collaboratif appliqué au domaine des ressources internet partagées, ce procédé de tagging se heurte à trois problèmes :

1. des problèmes typographiques : si le choix des tags est laissé à l'initiative de l'utilisateur, des problèmes typographiques (erreurs de frappe, casse différente) peuvent compromettre la cohérence sémantique. Par exemple, si le tag « Dsign » ou « design » est affecté à l'élément Assembly, le moteur de raisonnement pourrait être perturbé pour faire le rapprochement avec le tag « Design » affecté aux autres éléments ;
2. des problèmes potentiels d'ambiguïté : l'usage d'un même acronyme peut être problématique (EDF = Electricité de France mais aussi Equipe de France) ;
3. des problèmes de structure « à plat » : ce procédé permet d'affecter des éléments à des catégories, mais pas d'organiser ces catégories de manière hiérarchique. Ainsi, il n'est pas possible, en l'état, de créer de lien de dépendance entre items dans une relation de type parent/enfant.

Les deux premiers problèmes peuvent être résolus simplement dans le cadre de la conception de produits : il suffit de contraindre les choix de l'utilisateur en limitant le nombre de tags disponibles. Pour ce qui concerne le troisième point, deux solutions peuvent être envisagées [Bateman et al., 2007] :

- l'utilisation de *tags structurés* : les mots-clés sont séparés par des signes caractéristiques (slash, dièse etc.) rendant possible la création de catégories hiérarchisées. Par exemple, une photographie d'un monument parisien pourra être taggée avec la chaîne de caractères « Paris/Pictures/Monuments/Tour Eiffel » : la ressource figurera dans

la catégorie « Tour Eiffell », sous-catégorie « Monumentsl » etc. Cette approche n'est pas pertinente pour notre étude : les nœuds de nomenclature sont des catégories mais aussi des ressources (des items),
- des tags ontologiques : les annotations sémantiques sont liées à des ontologies. C'est ce concept que nous introduisons comme « Groupes sémantiques » dans le paragraphe suivant.

Groupes sémantiques Si l'on considère le composant **Bolt** de la MBOM figure 3.4.2, ce nœud porte une double sémantique :

- il figure dans la nomenclature de fabrication, et doit à ce titre porter le tag « Manufacturing » ;
- il représente également le *regroupement* de la vis **Screw 3** et de l'écrou **Nut 4**. L'origine de ce regroupement peut provenir d'un besoin en termes de gestion des approvisionnements. A ce titre, l'item **Bolt** devra être associé à un regroupement sémantique ou *groupe sémantique*, que nous définissons ci-dessous.

Définition 20. Un *groupe sémantique* regroupe un ensemble d'éléments pour lesquels, dans un contexte sémantique spécifique, les distances sémantiques sont réduites. Par exemple, dans le contexte sémantique *Les monuments les plus haut*s, les signifiés *Tour Eiffel* et *Empire State Building* seront plus proche sémantiquement que la *Tour Eiffel* et *Notre-Dame de Paris*. En revanche, ces deux derniers seront plus proches dans le contexte sémantique *Monuments parisiens*, duquel l'*Empire State Building* sera exclu du fait de sa grande distance sémantique. Dans le contexte fonctionnel d'un aéronef, les *sièges passagers* seront proches sémantiquement du *plancher*. Il en est de même pour un contexte de proximité géographique. En revanche, dans un contexte d'assemblage de l'avion, ceux-ci pourront être éloignés sémantiquement puisque l'assemblage des sièges s'opère bien après celui du plancher.

Pour modéliser cette relation d'appartenance à un groupe, plutôt que créer de liens directs entre items (dont nous avons montré qu'ils génèrent des ambiguïtés sémantiques), nous proposons d'utiliser des liens unidirectionnels entre éléments de l'espace des matières d'œuvre et de l'espace sémantique. Le modèle de *SemanticGroup (voir Figure 3.4.4)* est donc une extension du modèle présenté sur la figure 3.4.3 : la relation d'aggégation `SemanticDescriptionLink` peut être spécialisée en deux sous-classes :

- la classe `DefineGroup` permet de spécifier que le tag associé à l'item lui confère la qualité de *groupe sémantique* ;
- la classe `InGroup` indique que l'item qui porte ce tag est l'élément d'un *groupe sémantique*.

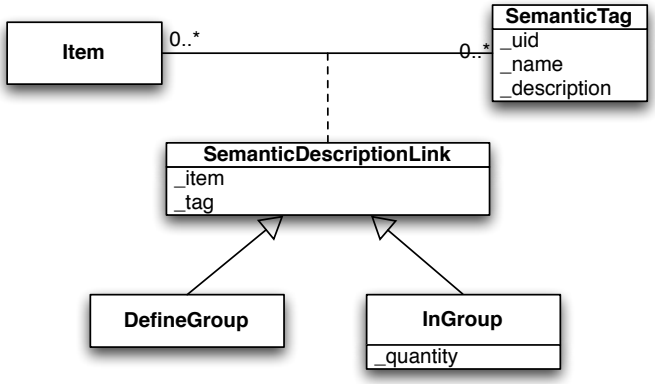


FIGURE 3.4.4.: Modèle de données pour le groupe sémantique

L'instanciation de ce modèle pour le cas d'étude considéré est représentée sur la figure 3.4.5. Les groupes sémantiques suivants ont été définis : DEsign Semantic Group 1 (DESG1), MAnufacturing Semantic Group 1 (MASG1), MASG2, MASG3 et MASG4. Sur cette figure :

- les lignes continues représentent des relations du type `SemanticDescriptionLink` (13 instances) ;
- les lignes continues doubles représentent les relations du type `DefineGroup` (5 instances) ;
- les lignes pointillées représentent les relations du type `InGroup` (11 instances).

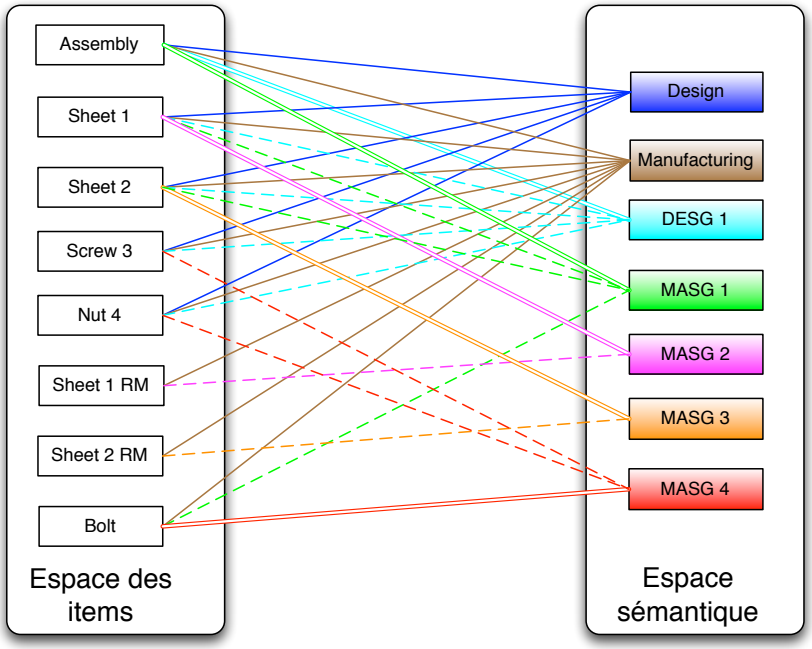


FIGURE 3.4.5.: Exemple d'instanciation du modèle de tags sémantiques

L'algorithme de déduction des structures produit à partir de ce modèle est détaillé dans la section 4.4.

3.4.5. Conclusion

Nous avons proposé un modèle permettant d'unifier les nombreuses représentations du produit dans deux espaces clairement séparés, l'espace des matières d'œuvre et l'espace sémantique, liés par un ensemble de relations de type tag. Ce modèle ne simplifie pas la représentation : il propose une autre classification des informations liées au produit et, surtout, bannit toute sémantique implicite qui serait contenue dans une vue produit. Outre la réconciliation des nomenclatures, nous pensons que ce modèle peut aussi permettre de simplifier la conception, le déploiement et la maintenance des bases de données qui stockent l'information relative au produit : les systèmes d'information du domaine du PLM s'appuient sur des bases de données relationnelles ([Chan, 1999], [Rao, 2000]) offrant de grandes capacités de stockage et d'excellentes performances. En revanche, ces bases de données peuvent être considérées comme rigides : il est en effet hors de question de modifier le modèle de données et son implémentation en base pour des produits utilisés dans le cadre de projets d'importance stratégique. Le modèle que nous proposons déplace la complexité du modèle de données implémenté vers une forme de raisonnement sur l'espace sémantique.

Résumé du chapitre 3

Ce chapitre fait suite à la définition de l'architecture multi-échelle orientée services vue dans le chapitre précédent. Il s'attache à la définition du méta-modèle d'unification et s'appuie sur l'exemple de deux problèmes d'interopérabilité :

- un problème intra-phase d'interopérabilité CAO/PDM pour permettre le travail collaboratif autour de la maquette numérique dans un contexte d'hétérogénéité des systèmes en présence ;
- un problème inter-phase d'interopérabilité PDM/ERP pour l'échange et l'unification de structures produit (EBOM et MBOM) entre ces deux systèmes, dans un contexte similaire d'hétérogénéité.

La définition de ce méta-modèle prend sa source dans l'étude du besoin. La nature et la granularité des informations à échanger entre les systèmes guident le choix du méta-modèle standard ou *ad hoc*. La recherche des similarités sémantiques permet d'identifier le point de convergence qui permettra la mise en correspondance sémantique de ces informations, ainsi que la spécification du méta-modèle d'unification. Des règles spécifiques de *mapping* permettent de composer avec les nuances sémantiques observées.

Enfin, un standard extensible (*Product Lifecycle Support*) est mis en oeuvre comme noyau du méta-modèle d'unification. La capacité de ce modèle standard à permettre des représentations multi-vues est soulignée, de même qu'est mise en exergue la difficulté de s'approprier les concepts de cette norme.

4. Validation de l'architecture et de la méthodologie associée

Nous présentons dans ce chapitre les détails relatifs à l'implémentation et au test des concepts que nous avons présentés dans les chapitres 2 et 3. Il est important de noter que nous n'avons pas développé **un** logiciel mais **un ensemble de composants logiciels interopérables**. Chacune de ces briques couvre un domaine fonctionnel strictement délimité permettant ensuite, par assemblage, de composer une application. Notre démarche de développement est la conséquence :

- d'un souci de maintenabilité du code. En effet, nos expériences dans le domaine du développement nous ont montré que, même dans le cas de développements exploratoires, la rigueur est nécessaire si l'on souhaite pouvoir garantir la pérennité de son travail ;
- de notre souhait de soumettre nos résultats à la critique : nous devons donner toutes les clés permettant de reproduire les résultats présentés ;
- de la volonté de partager les résultats de ce travail : notre attention s'est portée sur les spécifications et l'implémentation des composants afin qu'ils puissent être réutilisés ou améliorés par un tiers qui n'aurait pas participé à leur développement.

Nous avons donc consacré nos efforts de développement à la clarté, la simplicité et la concision du code. Tous les éléments dont nous pensons qu'ils contribuent à assombrir ou noyer la sémantique de notre travail ont été éliminés. Par exemple, nous n'avons développé aucune interface graphique, activité qui requiert beaucoup de temps et ne présente pas d'intérêt dans le cadre de notre recherche.

Dans la section 4.1, nous nous intéressons à la réalisation de l'interopérabilité technique, c'est-à-dire la composante de l'interopérabilité ayant vocation à assurer la continuité du flux sémantique. Nous présentons dans cette section l'ensemble des briques logicielles utilisées et nous implémentons un modèle d'architecture multi-échelle orientée services. Par la suite (voir 4.2), nous implémentons le médiateur dédié à l'interopérabilité CAO/PDM vu dans la section 3.2. L'interopérabilité PDM/ERP est vue dans la section 4.3. C'est dans cette partie que nous abordons en particulier les détails d'implémentation de PLCS. La section 4.4 est relative à l'implémentation du modèle de tags sémantiques. Chacune de ces sections sera conclue par une analyse des résultats et des conclusions que nous en tirons puis, dans une dernière section, nous proposerons les préconisations méthodologiques issues de ces expérimentations.

4.1. Réalisation de l'interopérabilité technique

L'objet de cette section est d'expliquer comment nous avons assuré l'interopérabilité technique dans le médiateur multi-échelle orienté services. Nous présentons tout d'abord le langage de programmation choisi pour conduire les expérimentation, puis les composants logiciels capables de mettre en oeuvre une architecture SOA. Cette section se termine par une validation de l'architecture.

4.1.1. Langage de programmation

Les expérimentations conduites dans cette partie ont été réalisées avec le langage de programmation Python [1]. Le langage Python nous apparaît comme un choix pertinent pour :

- aborder des problématiques d'interopérabilité : *Python est un langage de script dynamique, orienté objet et modulaire* [Sanner, 1999]. C'est un langage *interprété* (comme Perl, Php, Ruby) : les programmes écrits avec ce langage ne nécessitent donc aucune phase de compilation (contrairement à des langages comme C++ ou Java). En outre, Python propose un typage dynamique fort qui en fait un outil adapté à des environnements mouvants. A ce sujet, [Ousterhout, 1998] souligne que *les langages de script sont souvent utilisés pour étendre les fonctionnalités de composants écrits dans un autre langage. Les langages de scripts sont parfois décrits comme des langages d'agrégation ou langage d'intégration de systèmes* ;
- travailler dans un contexte exploratoire : d'après [Dubois, 1999], *la syntaxe et la sémantique de python sont claires et intuitives pour les ingénieurs et scientifiques.* Les programmes (aussi appelés scripts) Python sont concis et la possibilité d'utiliser une console interactive favorise les développements exploratoires. D'autre part, le projet IPPOP [Noël and Roucoules, 2008], présenté en introduction, propose dans sa structure un ensemble de scripts, démontrant toute la pertinence de cette approche dans un cadre scientifique. Nous pensons que l'utilisation intrinsèque d'un langage de script existant permet dans ce cas de se dispenser du développement d'un moteur de scripts *ad hoc* ;
- assurer, malgré cet aspect exploratoire, la possibilité future d'industrialiser des prototypes : Python est un langage utilisé et reconnu dans l'industrie. Des projets industriels de grande envergure basent leurs produits sur ce langage (par exemple les projets de simulation numérique Salomé [2] ou Code-Aster [3]), Google Inc. l'utilise pour son moteur d'applications en ligne *AppEngine* [4], des industriels du domaine de l'aéronautique [5] préconisent son utilisation etc.

1. http ://www.python.org
2. http ://www.salome-platform.org/
3. http ://www.code-aster.org
4. http ://code.google.com/intl/fr-FR/appengine/docs/python/
5. http ://www.python.org/about/success/usa/

Dans la section suivante, nous présentons comment nous avons utilisé ce langage pour implémenter une architecture multi-échelle orientée services.

4.1.2. SOA et services web

D'après [Booth et al., 2004], *plusieurs standards ouverts et de qualité industrielle font des webservices une technologie mature* (XML/RPC, SOAP). [Georgiev et al., 2007] précise que *ces standards représentent les informations en utilisant le langage à balises XML.*

Protocoles pour l'implémentation des webservices : XML/RPC et SOAP *Simple Object Access Protocol* (SOAP) est un protocole d'appel de procédure distante (Remove Procedure Call ou RPC) orienté objet, bâti sur XML, permettant de mettre en œuvre l'échange de donnée par services web. Il a été initialement défini par Microsoft et IBM, mais est devenu une référence dans le cadre des architectures de type SOA depuis les spécifications officielles [6] du *World Wide Web Consortium* (W3C). SOAP permet d'invoquer des fonctions (ou fournisseurs de service) situées sur un système distant, les spécifications du W3C définissant, dans un format XML, le format des informations transitant sur le réseau.

SOAP offre donc un standard permettant de garantir la continuité du flux sémantique dans l'architecture multi-échelle orientée services. Python propose des composants logiciels permettant de mettre en oeuvre ce protocole.

Client et serveur SOAP avec Python : le couple suds / soaplib soaplib [7] est un composant logiciel dédié au langage de programmation python qui implémente le protocole SOAP. Le paquet *soaplib* peut aussi être utilisé pour le client SOAP. Un autre paquet est cependant plus adapté pour mettre en œuvre rapidement un client SOAP : le package *suds* [8]. L'intérêt de ces deux bibliothèques de développement est qu'elles ajoutent une couche d'abstraction au-dessus du protocole SOAP, permettant ainsi au développeur de se focaliser sur les services sans avoir à se soucier de la manière dont son générés et échangés entre systèmes les paquets XML.

Utilisation conjointe de soaplib et suds Pour valider ces deux outils, nous avons déroulé le processus de test suivant :

- un serveur SOAP, implémenté avec soaplib, propose un service `ping` qui retourne la date du jour sous forme d'une chaîne de caractères ;

6. http ://www.w3.org/TR/soap/
7. http ://wiki.github.com/jkp/soaplib/
8. https ://fedorahosted.org/suds/

- un client SOAP, implémenté avec suds, accède au service `ping` et affiche la date renvoyée par le serveur ;

Ce test est exécuté sur une machine unique. Nous faisons l'hypothèse que si ce test simple est réussi, c'est-à-dire si le client est capable d'afficher la date envoyée par le serveur, nous aurons validé ces deux outils au regard de la conformité de leur implémentation du protocole standard SOAP. En effet, si l'un des deux au moins n'implémente pas rigoureusement SOAP, l'interopérabilité technique ne pourra pas être réalisée et le test échouera ;

Le script Python permettant mettre en oeuvre le serveur SOAP est nommé `simple_server.py` :

```
1  from soaplib.service
2  import soapmethod
3  from soaplib.serializers.primitive import String, Integer, Array
4  from soaplib.wsgi_soap import SimpleWSGISoapApp
5  import datetime
6
7  class PingService(SimpleWSGISoapApp):
8    @soapmethod(_returns=String)
9    def ping(self):
10     ''' Returns a simple string with the local time '''
11     print 'Received SOAP Request'
12     today = datetime.datetime.today()
13     return '%s'%today
14
15 # Setting up the server
16 from wsgiref.simple_server import make_server
17 server = make_server('localhost', 7889, PingService())
18 server.serve_forever()
```

Le client qui appelle ce service est lancé par le script `client.py` (on notera au passage la concision du code source permise par suds et donc sa rapidité de mise en œuvre) :

```
1  from suds.client import Client
2
3  # Set up client from WSDL file location
4  client = Client('http://localhost:7889/wsdl')
5
6  # call the ping request and print server response
7  print client.service.ping()
```

Ces deux listings illustrent le fait que la couche basse XML du protocole SOAP est de fait cachée : les bibliothèques suds et soaplib se chargent de la génération des messages XML à échanger.

Après avoir démarré le serveur, le client est lancé : la communication s'établit entre le serveur et le client (voir 4.1.1), et le client affiche bien la chaîne de caractères escomptée. Le test est réussi.

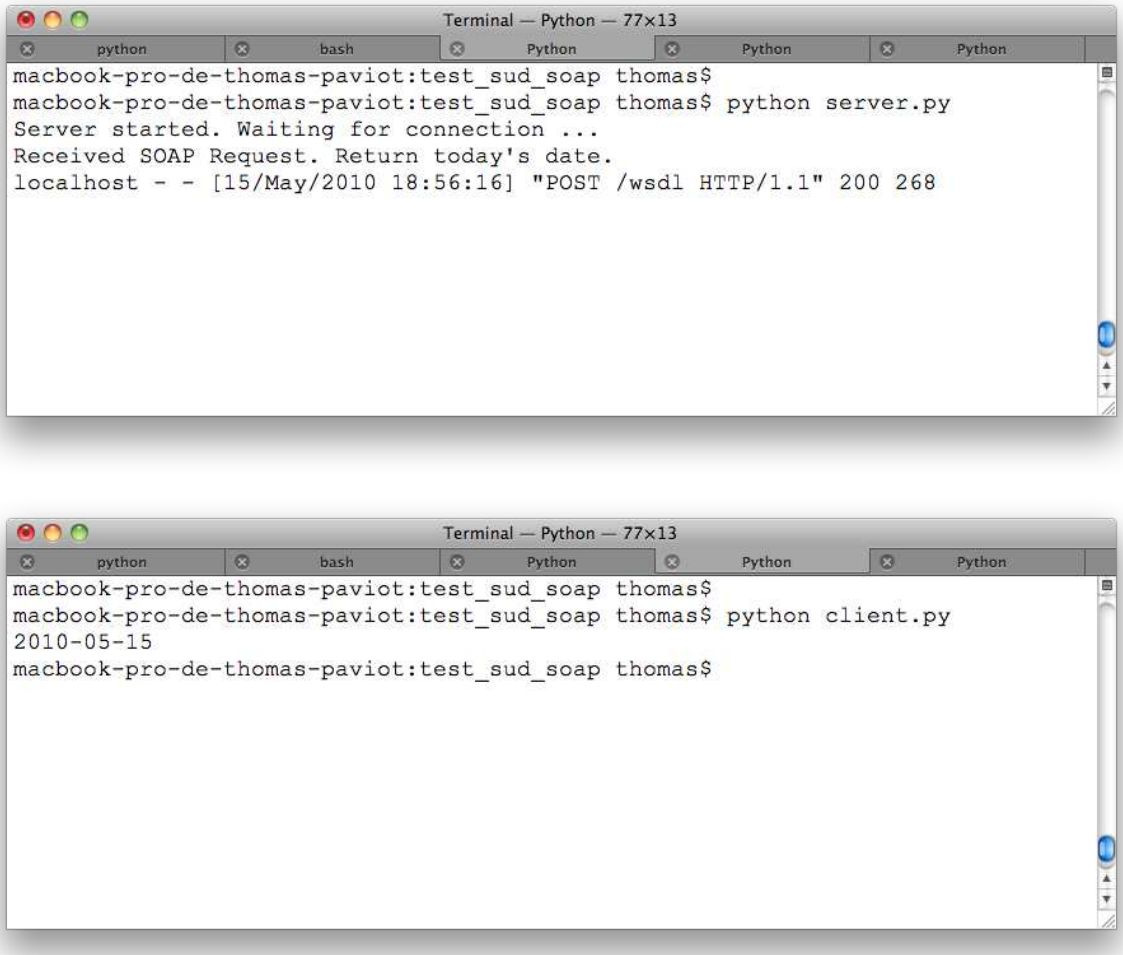


FIGURE 4.1.1.: Test de mise en oeuvre des bibliothèques `soaplib`/`suds`

Ces deux bibliothèques étant validées, nous les utilisons dans la suite de nos développements, notamment pour valider le médiateur multi-services présenté dans le chapitre 2, ce qui fait l'objet de la section suivante.

4.1.3. Médiation en flux poussé ou flux tiré

Nous proposons dans cette section de valider expérimentalement l'architecture multi-échelle orientée service présentée dans le chapitre 2. Nous avons fait remarquer, à la fin de ce chapitre, que les problèmes d'interopérabilité pouvaient se distinguer suivant deux catégories : les problèmes de transfert et les problèmes d'empaquetage, ces derniers pouvant par ailleurs être considérés comme un ensemble de problèmes de transferts simultanés ou séquentiels. Nous nous intéressons donc dans cette section à l'étude

exclusive du problème de transfert d'informations entre deux systèmes SOA1 et SOA2 par l'intermédiaire d'un médiateur M. Nous considérons alors deux types de problèmes à expérimenter :

- la **médiation en flux poussé** : dans ce cas, le médiateur M *décide* du transfert d'informations du système SOA1 vers le système SOA2. D'un point de vue de la relation entre systèmes, M est un client pour les serveurs SOA1 et SOA2 ;
- la **médiation en flux tiré** : le médiateur M *répond* à une requête du système SOA1 pour initier le transfert d'informations de SOA1 vers SOA2. Dans ce cas, M est un client pour le système SOA2 et il est dans le même temps un serveur pour le client SOA1.

Que ce soit dans le cas de la médiation en flux tiré ou dans celui du flux poussé, nous proposons de valider ces deux problèmes suivant un cas d'échange de données volontairement très simple : l'objectif est de transférer une chaîne de caractères « FromSOA1 » du système SOA1 vers le système SOA2. Il n'y a dans ce test aucun aspect sémantique, étant entendu que nous cherchons dans cette partie à réaliser l'interopérabilité technique, c'est-à-dire garantir la continuité du flux de données entre les systèmes. Nous faisons l'hypothèse que si les systèmes sont capables d'échanger des chaînes de caractères alors ils seront capables de la même manière d'échanger tous les autres types de données supportés par le protocole SOAP. Si ces deux expérimentations sont validées alors, quelle que soit l'échelle choisie, le nombre de systèmes ou encore le type de problèmes impliqués dans le problème d'interopérabilité, nous pourrons arguer que ce travail valide notre architecture multi-échelle orientée services.

Médiation en flux poussé Le modèle de système SOA1 est implémenté dans le fichier `SOA1.py`. Le serveur fournit un service web `get_string` qui renvoie la chaîne de caractères « FromSOA1 » :

```
1  from soaplib.service import soapmethod
2  from soaplib.serializers.primitive import String
3  from soaplib.wsgi_soap import SimpleWSGISoapApp
4
5  STR_SOA1 = "FromSOA1"
6
7  class StringServiceSOA1(SimpleWSGISoapApp):
8    @soapmethod(_returns=String)
9    def get_string(self):
10     ''' Returns the string STR_SOA1 '''
11     return STR_SOA1
12
13 from wsgiref.simple_server import make_server
14 server = make_server('localhost', 7880, StringServiceSOA1())
15 print 'Server SOA1 started. Waiting for connection ...' server.
       serve_forever()
16 server.serve_forever()
```

Le système SOA2 est lancé par le script `SOA2.py`. On aura pris soin au préalable de changer le numéro de port de connexion pour pouvoir exécuter ces différents scripts sur la même machine (port 7880 pour SOA1, port 7890 pour SOA2). Ce serveur implémente un service web qui prend comme paramètre une chaîne de caractères puis l'affecte à la variable globale `STR_SOA2` :

```
from soaplib.service import soapmethod
from soaplib.serializers.primitive import String
from soaplib.wsgi_soap import SimpleWSGISoapApp

STR_SOA2 = "Default_EmptyString"

class StringServiceSOA2(SimpleWSGISoapApp):
  @soapmethod(String)
  def set_string(self,received_string):
    ''' Only take a string '''
    global STR_SOA2
    print "STR_SOA2 initial value %s"%STR_SOA2
    STR_SOA2 = received_string
    print "STR_SOA2 set to %s"%STR_SOA2

from wsgiref.simple_server import make_server
server = make_server('localhost', 7890, StringServiceSOA2())
print 'Server SOA2 started. Waiting for connection ...'
server.serve_forever()
```

Ce médiateur *pilote* de manière séquentielle le processus de transfert d'informations de SOA1 vers SOA2 :

- dans un premier temps, il se connecte aux deux serveurs ;
- il adresse ensuite une requête au serveur SOA1 pour récupérer la chaîne de caractères « FromSOA1 » ;
- enfin, il adresse une requête au serveur SOA2 pour affecter la chaîne « FromSOA1 » à la variable STR_SOA2 du deuxième système.

```
from suds.client import Client
# Set up 2 clients
client_SOA1 = Client('http://localhost:7880/wsdl')
print "Connection to SOA1 successful."
client_SOA2 = Client('http://localhost:7890/wsdl')
print "Connection to SOA2 successful."
# get the string from SOA1 temp_string = client_SOA1.service.get_string()
print "Got string from SOA1"
# send the string to the second SOA2
client_SOA2.service.set_string(temp_string) print "Moved string to SOA2"
```

Sur la figure 4.1.2, les résultats obtenus sont présentés sous forme de captures d'écran qui montrent :

- sur la première image, le serveur SOA1, démarré depuis un terminal ;
- sur la deuxième image, le serveur SOA2, démarré depuis un deuxième terminal ;

- sur la dernière image (celle du bas de la figure), l'exécution du script Médiateur.py dans un troisième terminal après que les deux serveurs ont été démarrés. Le fait de démarrer chacun des serveurs/clients dans des terminaux distincts assure la séparation des processus (les processus étant à considérer ici dans le domaine de l'informatique, c'est-à-dire un ensemble de tâches qui s'exécutent simultanément dans des espaces mémoires différents).

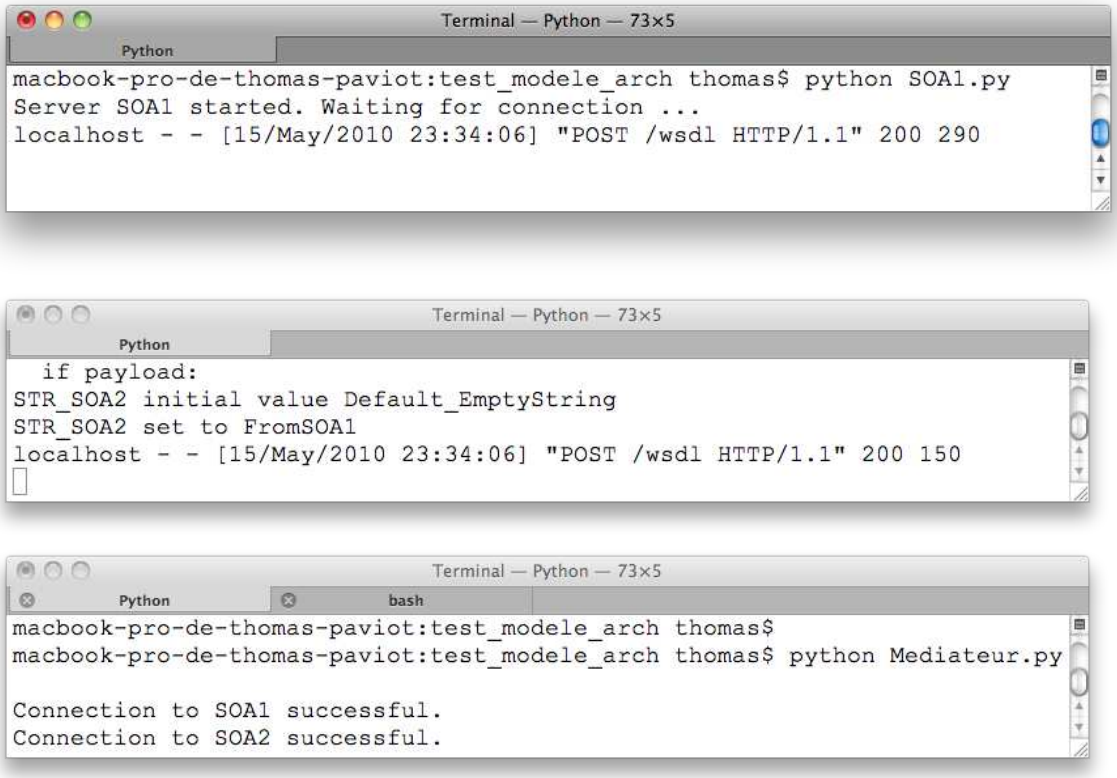


FIGURE 4.1.2.: Validation de la continuité des données pour la médiation en flux poussé

Nous constatons que le processus se déroule comme escompté : la deuxième image de la figure précédente témoigne que la donnée du serveur 1 (la chaîne de caractères « FromSOA1 ») a bien été transférée au système SOA2. Dans le paragraphe suivant, nous expérimentons le deuxième type de problème : la médiation en flux tirés.

Médiation en flux tiré Dans le cas des flux tirés, la base développée dans le paragraphe précédent peut être réutilisée. Le système SOA1 réalise les même fonctions que dans le cas précédent mais il convient d'effectuer les modifications suivantes par rapport au cas du flux poussé :

- il faut adjoindre au médiateur un serveur SOAP qui le transforme en fournisseur de services ;
- le système SOA2 devient quant à lui le client du médiateur M.

Par rapport au cas précédent, le script Mediateur.py devient :

```
1  from soaplib.service import soapmethod
2  from soaplib.serializers.primitive import String
3  from soaplib.wsgi_soap import SimpleWSGISoapApp
4  from suds.client import Client
5
6  def get_string_from_SOA1():
7      ''' Connect to SOA1 serveur and request string '''
8      # Connect to SOA1
9      print 'Connecting to SOA1 server.'
10     client_SOA1 = Client('http://localhost:7880/wsdl')
11     print 'Connection established. Requesting string from SOA1.'
12     # get the string from SOA1
13     tmp_string = client_SOA1.service.get_string()
14     print 'String received from SOA1.'
15     return tmp_string
16
17 class ServiceMediator(SimpleWSGISoapApp):
18     @soapmethod(_returns=String)
19     def get_string(self):
20         ''' Returns the string from SOA1 '''
21         print 'Mediator : received request.'
22         string_to_return = get_string_from_SOA1()
23         print 'Mediator : return request.'
24         return string_to_return
25
26 from wsgiref.simple_server import make_server
27 server = make_server('localhost', 7850, ServiceMediator())
28 print 'Server Mediator started. Waiting for connection ...'
29 server.serve_forever()
```

Nous faisons remarquer que dans ce listing :

- nous implémentons une stricte séparation entre la couche publique et la couche privée du système Médiateur : le service `get_string()` est localisée dans la couche publique du système (et donc visible par n'importe quel client SOAP autorisé à établir une connexion) alors que la fonction `get_string_from_SOA1()` fait partie de la partie privée : elle n'est pas visible depuis l'extérieur, constitue une fonction technique interne du système et n'est donc pas un service ;
- la connexion au client SOA1 est réalisée « à la volée », au moment précis où le médiateur reçoit la requête. Cette imbrication des fonctions peut poser un problème dans le cas où la connexion avec le serveur SOA1 ne peut s'établir : le serveur attend alors la connexion et devient indisponible (nous avons écrit dans le chapitre 2 qu'il est occupé à une autre activité et la communication est perturbée). Ce problème peut être résolu en utilisant les techniques de la programmation multi-tâches : le serveur et la connexion au client sont démarrés au lancement du script dans des tâches distinctes, une pile *First In/First Out* (*Queue* en anglais) étant alors utilisée pour partager les informations entre tâches, elles-mêmes synchronisées via un gestionnaire d'événements. Nous ne

détaillons pas le développement relatif à cette partie, mais sa réalisation a été testée à l'aide du paquet *multiprocessing*[9] intégré à Python.

Le script correspondant au système SOA2 est pour sa part considérablement simplifié : SOA2 se contente d'appeler le service correspondant fourni par le médiateur :

```
1 from soaplib.service import soapmethod
2 from suds.client import Client
3
4 # Connect to the mediator M
5 client_M = Client('http://localhost:7850/wsdl')
6
7 # Then request the string
8 received_string = client_M.service.get_string()
9 print "Received string from Mediator : %s"%received_string
```

Comme précédemment, chaque script est lancé depuis un terminal différent :

- dans un premier temps, les serveurs sont démarrés (SOAP1.py et Mediateur.py) ;
- ensuite, le système SOA2 est lancé afin qu'il puisse initier le processus de transfert via la requête du service `get_string()` adressée au médiateur.

Les résultats sont présentés dans la figure 4.1.3 :

- l'image du haut est une capture d'écran du terminal exécutant le système SOA1 ;
- l'image du milieu représente l'exécution du script Mediateur.py : on constate bien que le médiateur répond à la requête du système SOA2 et qu'il se connecte avec succès au système SOA1 ;
- sur l'image du bas, nous vérifions que le système SOA2 a bien récupéré la chaîne de caractères « FromSOA ».

9. *http ://docs.python.org/library/multiprocessing.html ?highlight=processing#module-multiprocessing*

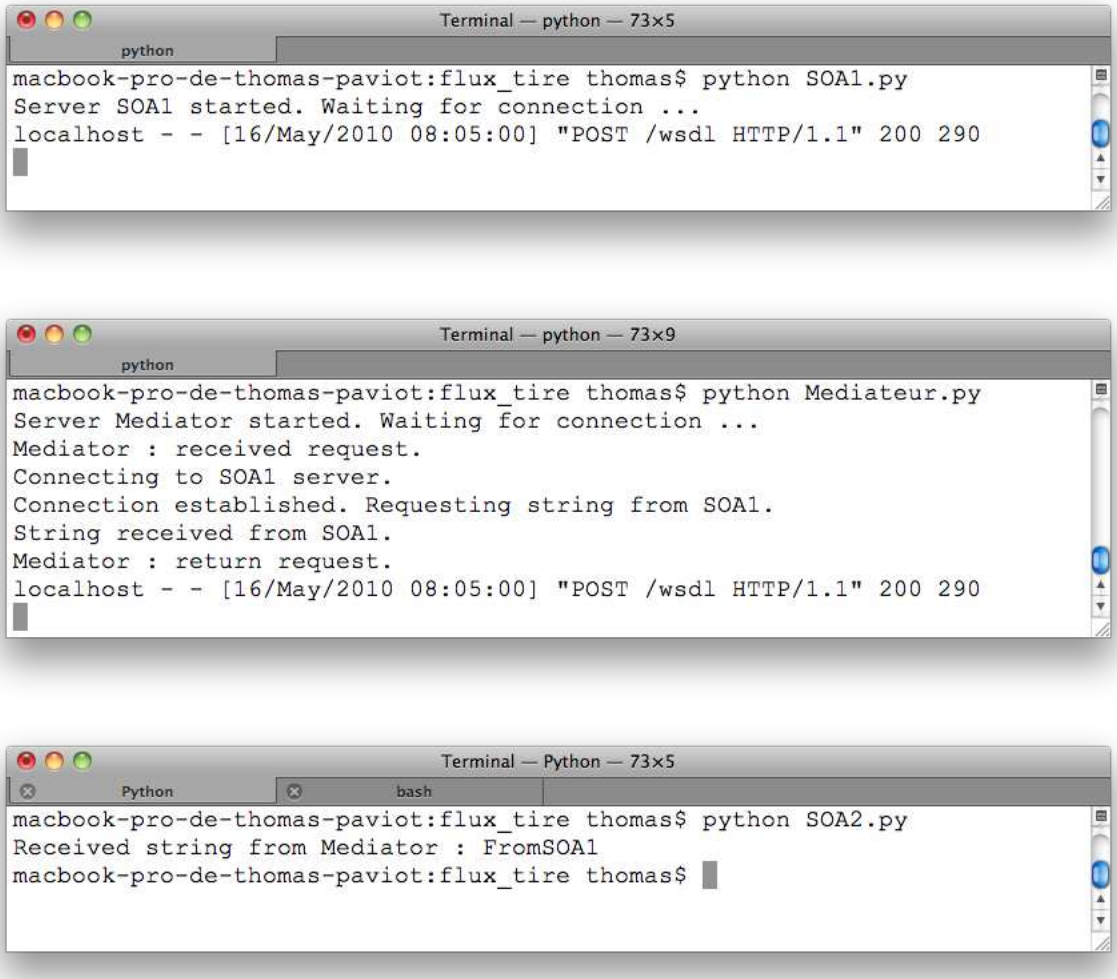


FIGURE 4.1.3.: Validation de la continuité des données pour la médiation en flux tiré

Nous dressons dans le paragraphe suivant un bilan de ces expérimentations et leurs conséquences en termes méthodologiques.

Conclusion / Recommandations méthodologiques Nous avons validé les deux cas correspondant à la médiation en flux poussé ou en flux tiré. Nous tirons de ces expérimentations les conclusions suivantes :

- la **médiation en flux poussé est plus simple en termes d'implémentation : le médiateur pilote seul le processus de transfert d'informations**. En revanche, cette solution présente un inconvénient : il **faut être en mesure de prévoir les besoins en informations** des différents systèmes, considérés alors comme passifs vis-à-vis de ce processus, **pour leur adresser la bonne donnée au bon moment. Cette technique sera réservée pour les cas où les transferts impliquent peu**

de SI ou bien lorsque les besoins d'informations sont connus à l'avance et donc prévisibles ;

- la **médiation en flux tiré est plus difficile à implémenter** car elle requiert de développer, pour chaque système, à la fois un client et un serveur traduisant le fait que chaque système est consommateur et fournisseur de services web. **La synchronisation des systèmes devra être assurée par l'utilisation des techniques de la programmation multi-tâches associées à des gestionnaires d'événements**. Cette **solution est beaucoup plus souple : le médiateur se charge uniquement de transférer les informations « à la demande », et n'a pas besoin de connaître au préalable le moment précis où les systèmes auront besoin des informations leur permettant de fonctionner. Cette solution apparaît plus pertinente d'un point de vue de la mise-à-l'échelle** : chaque système est en effet responsable des informations qu'il demande et fournit aux autres SI. De ce point de vue, la médiation multi-échelle orientée services est **plus modulaire, flexible et agile**.

4.2. Réalisation de l'interopérabilité CAO/PDM

Nous présentons dans cette section l'implémentation du médiateur capable de résoudre le problème d'interopérabilité CAO/PDM tel qu'il a été défini dans le chapitre 3. Dans un premier temps, nous présentons la technique de « webservicisation » du logiciel Catia V5. Puis nous détaillons la connexion au services web de Windchill, système PDM utilisé dans le cadre de cette expérimentation. Enfin, dans un dernier paragraphe, nous présentons nos résultats ainsi que les conclusions que nous en tirons.

« Webservicisation » d'applications non-orientées services Dans le cadre de l'interopérabilité CAO/PDM vue dans la section 3.2, nous utilisons le logiciel Catia V5, édité par Dassault Systèmes, qui possède la caractéristique de ne pas être orienté service. L'OMG définit un ensemble de services web dédiés aux activités de CAO [OMG, 2005] [10] mais force est de constater qu'à l'heure actuelle aucun éditeur de solution CAO n'a implémenté, même partiellement, les spécifications de l'OMG. Le successeur de V5, Catia V6, est annoncé comme étant orienté services, mais nous n'avons pas pu expérimenter cet outil.

Nous proposons donc un artifice permettant tout de même d'utiliser un outil CAO non-orienté services dans le cadre de notre étude, qui s'appuie sur des outils orientés services. Nous appelons cette technique « webservicisation d'applications non SOA », qui consiste à ajouter au-dessus du logiciel CAO une surcouche qui implémente les quelques webservices dont nous avons besoin. Nous nous sommes aperçus qu'à défaut de proposer un ensemble de services, les outils CAO fonctionnant sous système d'exploitation Windows proposent

10. http ://www.omg.org/technology/documents/formal/cad

une interface de programmation permettant d'automatiser des tâches répétitives. Ces interfaces de programmation sont prévues pour être utilisées *via* le langage Visual Basic ou des scripts VBA directement depuis le logiciel. Ces scripts VBA sont habituellement appelés Macros Catia V5. La gestion de ces API est réalisée par la technologie COM de Microsoft : un serveur COM, inscrit dans la base de registre du système d'exploitation lors de l'installation du logiciel, fournit un ensemble de fonctions accessibles via un serveur COM. Ces fonctions ne constituent pas des services, puisqu'elles sont dépendantes d'une technologie d'implémentation propriétaire et non standardisée. Néanmoins, il est possible d'accéder à ces API à partir de n'importe quel langage de programmation offrant la possibilité d'implémenter un client COM. La technologie COM ayant été popularisée par Microsoft, tous les langages offrent aujourd'hui des composants logiciels spécialisés capables de lancer un client COM qui puisse se connecter à tout serveur COM présent sur l'ordinateur.

Ainsi, la webservicisation en question se fait suivant la technique suivante :

1. identifier les données à échanger ou les actions distantes à piloter ;
2. identifier, dans l'API COM, les classes/méthodes/fonctions pertinentes par rapport à ce besoin. L'interface COM de Catia V5 offre de l'ordre de 4000 fonctions. Il est évidemment hors de portée de ce travail de recherche de construire un serveur SOAP qui couvre l'intégralité de ces fonctions. En revanche, pour des besoins liées à l'interopérabilité et à l'échange d'informations entre systèmes CAO et PDM, seule une toute petite partie de ces fonctions est réellement nécessaire : ouvrir un fichier CAO, récupérer l'arborescence (liens de dépendances), générer un fichier au format STEP ou 3DXML, récupérer les matrices de position des pièces ou les éléments d'inertie ;
3. identifier, dans les spécifications OMG CAD Services, les services correspondant à ces besoins ;
4. mettre en correspondance les services CAD Services et les fonctions COM ;
5. implémenter, sur l'ordinateur sur lequel est installé le logiciel, un serveur SOAP qui fait le lien entre les services CAD et les fonctions et objets COM.

L'architecture technique résultante est représentée sur la figure 4.2.1.

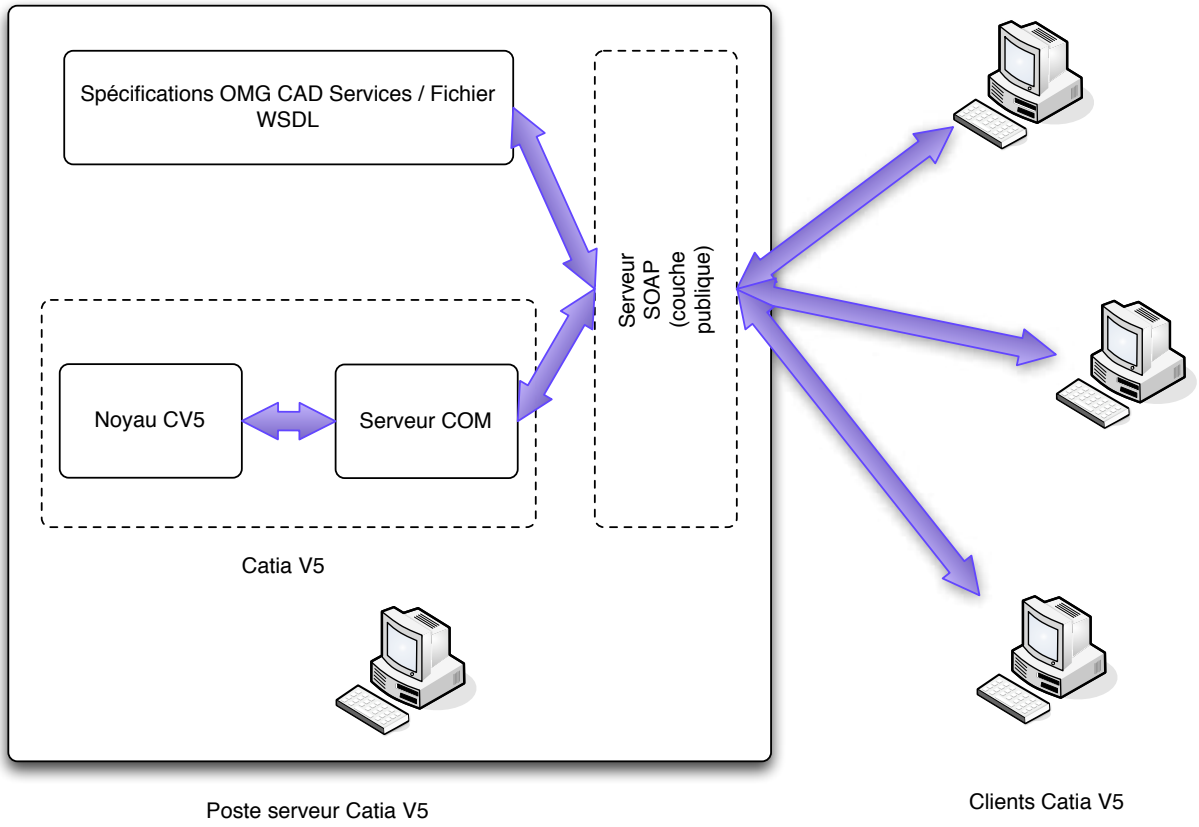


FIGURE 4.2.1.: Webservicisation du logiciel Catia V5

Il est à noter que cette technique peut être utilisée pour webserviciser toutes les applications qui utilisent la technologie COM (les outils de la suite bureautique Office, etc.). A titre d'information, sur la station de travail utilisée pour les expérimentations, plus de 2000 serveurs COM sont inscrits dans la base de registre Windows.

Connexion aux WebServices de Windchill Dans un premier temps, il convient de vérifier que la connexion au serveur Info*Engine peut s'établir : pour cela, Info*Engine propose un service dénommé *ping* qui permet d'obtenir une réponse de présence du serveur. Le code suivant permettant de se connecter au serveur, d'appeler le service *ping* et d'afficher la réponse du serveur.

```
from suds import WebFault
from suds.client import Client
from suds.sudsobject import Object
from suds.transport.https import HttpAuthenticated

# Necessary to resolve missing reference issues in the xsd schema
from suds.xsd.doctor import ImportDoctor, Import
imp = Import('http://schemas.xmlsoap.org/soap/encoding/')
d = ImportDoctor(imp)

# Server location, user connexion settings.
# Note : the url points to the WSDL file
# Replace host, 'my_username' and 'my_password' with your settings.
url = 'http://host/Windchill/servlet/RPC?CLASS=com.infngine.soap'
credentials = dict(username='my_username', password='my_password')

# Connect to the server, get the WSDL file, open the connection
t = HttpAuthenticated(**credentials)
client = Client(url, transport=t, doctor=d)

# call the ping request and display server response
ping_response = client.service.ping()
print ping_response
```

FIGURE 4.2.2.: Accès au service *ping* d'*Info*Engine*

L'exécution de ce script produit la sortie suivante, indiquant que le serveur *Info*Engine* fonctionne :

```
1 <wc:COLLECTION xmlns:wc="http://www.ptc.com/infoengine/1.0">
2 <PingResponse NAME="output" TYPE="Object" STATUS="0">
3  <wc:INSTANCE>
4  <vmname>the_server_name</vmname>
5   <date>4/30/10 4:24 AM</date>
6   </wc:INSTANCE>
7   </PingResponse>
8  </wc:COLLECTION>
```

Nous pouvons ainsi connecter le médiateur à la quarantaine de services proposés par Windchill.

Conclusions Nous avons validé le médiateur orienté services pour le cas de l'interopérabilité CAD/PDM pour le cas spécifié dans le chapitre 3 [Paviot et al., 2008]. Ainsi, il est possible de transférer une maquette numérique de sa station CAO locale vers le système PDM, et d'utiliser ensuite le système PDM et le médiateur pour ventiler les différentes parties de la maquette vers les stations des concepteurs. En revanche, nous nous sommes limités à un aspect statique : nous n'avons pas expérimenté plus avant la

dynamique de la conception collaborative (récupération/vérouillage/versionnement etc.). Une raison essentielle explique ce manque : nous nous sommes rendu compte, après avoir commencé les développements, que le méta-modèle choisi, mélange de modèle *ad hoc* et de sémantique STEP, ne se prêtait à aucune évolution et que l'intégration de l'aspect dynamique nécessitait de reprendre l'intégralité du modèle de données. C'est une des raisons qui explique nos préconisations méthodologiques tendant à utiliser un modèle standard extensible : il se peut qu'une évolution future non-prévue soit rendue impossible par un modèle de données rigides, conçu et implémenté dans un contexte différent.

Dans le cadre de nos développements, nous avons développé une couche applicative capable d'adresser indifféremment les logiciels Catia V5 ou SolidWorks. Les développements relatifs à ce composant sont librement disponibles sur la page :

http ://svn.gna.org/viewcvs/decade/trunk/src/

Le transfert de structures produit, la génération automatiques des fichiers de visualisation 3D, de génération des fichiers et le lien bi-directionnel CAO/PDM a fait l'objet d'un logiciel dont le code source est disponible sur la page :

http ://svn.gna.org/viewcvs/cadshuttle/trunk/1.X/src/

4.3. Réalisation de l'interopérabilité PDM/ERP

L'architecture fonctionnelle a été implémentée suivant le diagramme suivant. Pour les tests qui ont été réalisés, les systèmes PDM et ERP choisis sont ceux disponibles au laboratoire LISMMA : Windchill PDMLink/Project link comme solution PDM et OpenERP, système ERP libre développé avec le langage Python.

La boite de la figure 4.3.1 représente les frontières de l'implémentation, et l'environnement est composé de :

- un accès de type XML/RPC (eXtended Markup Language/Remote Procedure Call) pour le transfert de données de/vers OpenERP. Le client/serveur de communication présent dans OpenERP est base sur ce protocole, et la partie serveur peut être facilement atteinte avec le module Python xmlrpclib pour créer des composants, des MBOMs, obtenir des informations, etc. ;
- un accès de type SOAP pour le transfert de données de/vers le PDM permettant l'accès aux services web. PDMLink et ProjctLink offrent tous les deux environ 40 services web permettant l'interopérabilité avec un autre outil (création de workflow, de composants et de produits, extraction d'informations, etc.) ;
- le « noyau de traduction » est destiné à fusionner les données provenant de ces systèmes et il est responsable de la cohérence des données conception/production. Il est basé sur un modèle de données créé à partir de l'AP239 ;

- la manipulation des objets de l'AP239 de STEP (entités, types, fonctions, génération de fichiers STEP, etc.) est réalisé par le composant pySDAI ;
- enfin, un ensemble de services web est implémenté pour donner à l'utilisateur un contrôle total sur le comportement de la solution.

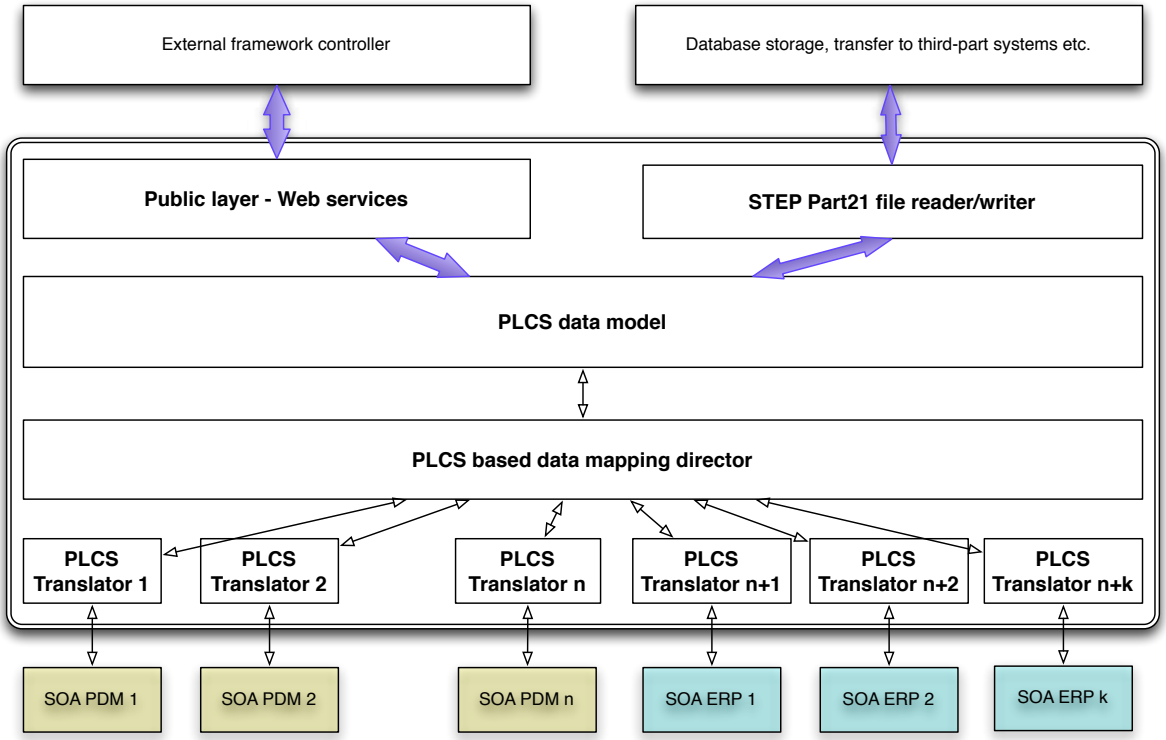


FIGURE 4.3.1.: Architecture fonctionnelle orientée Services

Cette section présente l'environnement d'interopérabilité PDM/ERP et le modèle de données produit PLCS. Pour éviter les redondances de données, une traduction volatile entre les objets de l'ERP et du PDM est réalisée. L'orchestration et le pilotage des flux de données et des workflows sont réalisés suivant les besoins de l'entreprise, *i.e.* ils permettent à l'utilisateur final d'un des systèmes de récupérer ou de mettre à jour les données des autres systèmes. L'environnement est composé de :

- un ensemble de traducteurs PLCS, qui convertissent les objets provenant du PDM ou de l'ERP en objets PLCS ;
- un chef d'orchestre de la traduction de données, qui fusionne les informations des traducteurs en une description PLCS cohérente ;
- une complète représentation sémantique du PDM et de l'ERP qui utilise le modèle de données spécialisé PLCS ;
- une interface publique, prévue pour recevoir les requêtes utilisateur et renvoyer les informations appropriées. Cette interface est la partie basée sur les services web de l'environnement ;

- un gestionnaire de fichier STEP, basé sur pySDAI, aussi bien capable d'encoder les données produits dans un fichier STEP Part 21 que de récupérer des informations d'un tel type de fichier.

La section suivante détaille l'implémentation de PLCS.

4.3.1. Implémentation de STEP PLCS

Standard Data Access Interface (SDAI) EXPRESS a pour fonction de modéliser des données, les représenter et spécifier leur intégrité via des contraintes. Il se distingue des autres langages à objets (C++, Java, Ruby, Python etc.) dans le sens où il ne permet pas de décrire des instances d'entité. A ce titre, il n'est pas possible d'écrire des programmes informatiques en EXPRESS qui pourraient être compilés/interprétés pour être exécutés sur un ordinateur (comme c'est également le cas pour UML par exemple) : un schéma ne peut pas être exploité directement, il doit être traduit dans un langage de programmation [Plantec, 1999]. Ce point complique, dans des activités de recherche, l'implémentation et l'expérimentation des idées/projets s'appuyant sur le formalisme EXPRESS/STEP.

Le *Standard Data Access Interface* (SDAI) définit une interface de programmation d'application (*Application Programming Interface* ou API) pour les données EXPRESS, comme par exemple les protocoles d'application de STEP. STEP définit les opérations et modèles de données SDAI dans la Part 22 de l'ISO-10303 [ISO-10303-22, 1998]. Le lien entre EXPRESS et un langage de programmation est appelé *binding*, traduit dans la version française de la norme par le terme *liant*. A défaut d'une meilleure traduction, nous utiliserons le terme original *binding* pour désigner ce lien. *Les bindings* précisent comment les opérations apparaissent dans un langage de programmation spécifique. Les *bindings* sont de deux natures, *early binding* et *late binding* (que l'on pourrait traduire par liant précoce et liant tardif) *:*

- l'*early binding* nécessite une phase préalable de compilation du schéma EXPRESS. L'ensemble des entités/attributs/contraintes/fonctions sont disponibles dans le langage cible et l'exécution du code est rapide ;
- lors du *late binding*, la connexion aux entités/méthodes s'opèrent durant l'exécution du programme via un dictionnaire. Le *late binding* permet d'éviter la compilation du schéma, qui peut être longue si le schéma est important. En revanche, il est moins souple en matière de programmation puisque l'ensemble des entités/méthodes/etc. disponibles n'est pas défini au préalable.

Des *bindings* SDAI existent pour les languages C++ [ISO-10303-23, 2000], C [ISO-10303-24, 2001] et Java [ISO/TS-10303-27, 2000]. Des travaux pour des *bindings* CORBA/IDL et FORTRAN furent initiés mais abandonnés. Le développement de prototypes concernant EXPRESS/STEP nécessite donc :

- le choix d'un langage de programmation (C, C++ ou Java) ;

- un environnement adéquat (compilateur C, C++ ou Java, éditeur de code, debugger etc.) ;
- un composant logiciel (bibliothèque de développement) EXPRESS/STEP conforme aux spécifications des Part 23, 24 ou 27 de l'ISO-10303.

La création et la manipulation des instances EXPRESS, conformément à un schéma donné, nécessite donc des composants logiciels dédiés. Ces difficultés quant à l'implémentation d'EXPRESS/STEP sont soulignées dans la littérature. Dans SDAI, la représentation des instances d'entités et leur persistance est assurée par :
- une session ;
- un dépôt (*repository*) ;
- des modèles SDAI (*sdai_model*).

Pour les deux premiers points, un grand nombre d'outils de qualité sont disponibles, qu'ils soient libres ou propriétaires, commerciaux ou non. En revanche, ce n'est pas le cas pour le troisième point : un petit nombre d'entreprises se sont spécialisées dans l'implémentation de STEP et ont développé des compétences/composants logiciels de qualité industrielle. On peut citer de manière exhaustive EuroStep Group AB [11], Jotne EPM Technology AS [12], Step Tools Inc. [13], LKSoftware Gmbh [14] et PROSTEP AG [15].

Pour toutes ces raisons, nous avons développé notre propre composant logiciel permettant la manipulation des données EXPRESS/STEP, intitulé pySDAI (voir appendice C).

Enregistrement des instances d'entités EXPRESS dans des fichiers ASCII ou XML Il est possible d'enregistrer les instances d'entités EXPRESS dans des fichiers, dans l'optique de les échanger (le E de l'acronyme STEP). La norme STEP définit deux protocoles pour enregistrer ces instances :
- dans des fichiers ASCII à encodage textuel clair [ISO-10303-21, 2002] ;
- dans des fichiers au format XML [ISO-10303-28, 2007].

Etalonnage du composant logiciel pySDAI L'étalonnage de pySDAI vise, dans un premier temps, à valider que le composant est apte à être utilisé pour les expérimentations qui suivent. L'étalonnage a été réalisé à partir de :
- un ensemble de tests unitaires, c'est-à-dire une batterie de tests élémentaires permettant de valider la réaction du composant par rapport à des résultats attendus et connus *a priori*. L'ensemble de tests en question concerne la vérification du type des arguments passés, l'instanciation correcte, la vérification ou violation de contraintes locales ou globales, la bonne réalisation des fonctions ;

11. http ://www.eurostep.com/
12. http ://www.epmtech.jotne.com/
13. http ://www.steptools.com/
14. http ://www.lksoft.com/
15. http ://www.prostep.com/ ?L=1

- un ensemble de fichiers STEP de référence [16] connus pour leur conformité aux schémas de l'AP203 et 214. Le composant effectue le bon diagnostic de validité (i.e. diagnostiquer un nombre d'erreurs égal à 0).

4.3.2. Implémentation de l'architecture

Le *mapping* choisi est une correspondance bi-univoque entre EBOM et MBOM (voir figure 4.3.2).

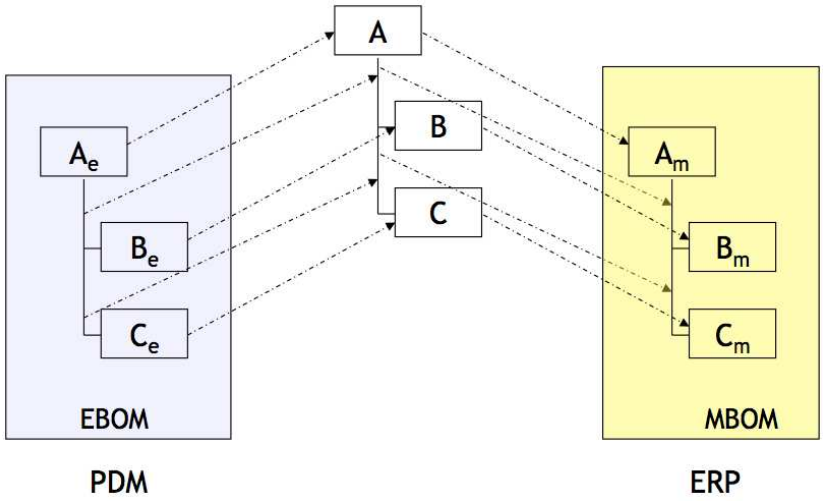


FIGURE 4.3.2.: Mapping entre les nomenlcatures PDM et ERP

4.3.3. Résultats

Le résultat de notre travail est représenté sur la figure 4.3.3. Nous avons adjoint au médiateur un serveur HTTP qui permet à un utilisateur de piloter le médiateur via une une interface accessible depuis un navigateur internet. Par l'intermédiaire de cette interface, il est possible de consulter les structures produits stockées dans le PDM ou l'ERP et également de commander le transfert de nomenclatures d'un système à l'autre. Évidemment, il reste possible de connecter le médiateur à un système tiers sur lequel est implémenté un client SOAP, ce qui est représenté sur la figure 4.3.3 par le terminal Windows de couleur noire dans lequel est lancé un script Python permettant de récupérer et afficher les nomenclatures.

16. http ://cax-if.org/library/index.html

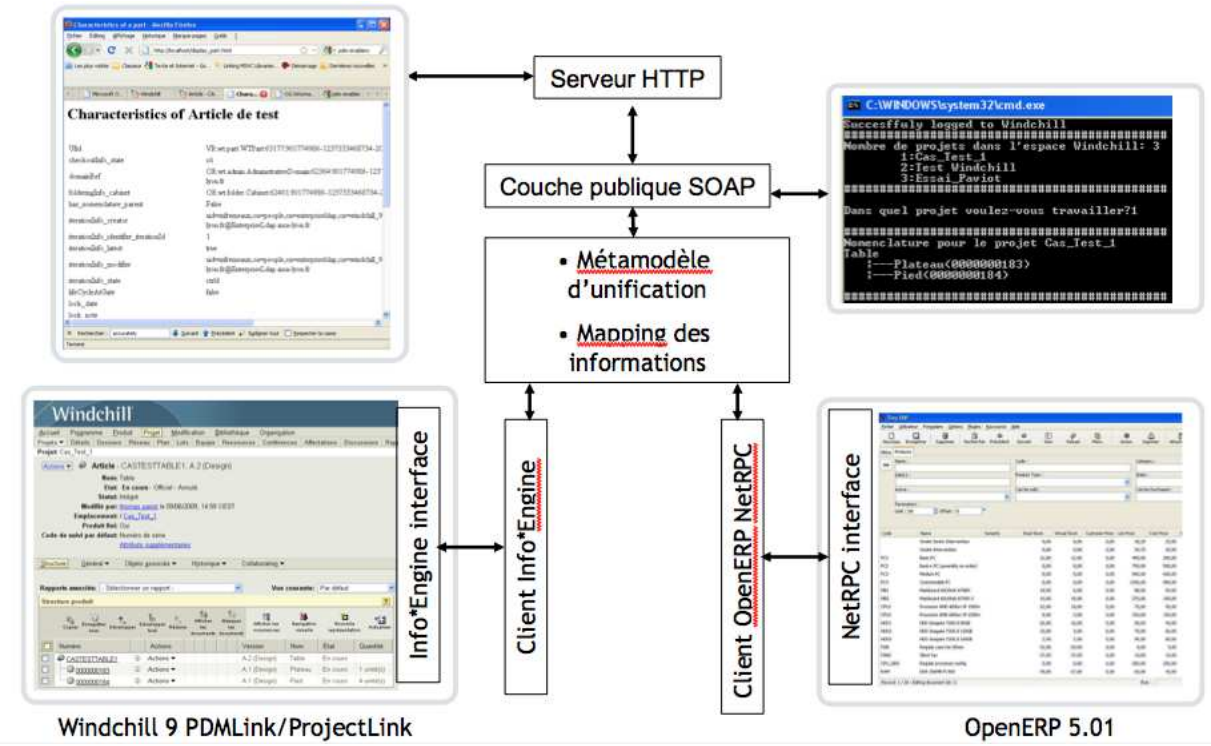


FIGURE 4.3.3.: Résultats de l'expérimentation pour l'interopérabilité PDM/ERP

Ainsi, à défaut d'avoir automatisé complètement le processus de transfert d'informations, nous avons rendu aisée la possiblité laissée à un utilisateur d'interagir avec le médiateur.

Conclusions Nous avons validé l'échange de structures produit du PDM vers l'ERP pour un produit simple. De même, nous avons validé la réconciliation de ces structures en vue d'un empaquetage des informations sous la forme d'un fichier STEP.

Un problème est apparu au cours de ces expérimentations : le processus d'échange des informations n'est pas *atomique* : si un dysfonctionnement interrompt le processus d'échange alors qu'il n'est pas terminé, la MBOM qui apparaît dans l'ERP n'est de ce fait pas synchrone avec la EBOM originale. Dans ce cas, la solution que nous nous sommes attaché à développer crée davantage de problèmes qu'elle n'en résoud. Le problème de la *sécurité* de cette transaction est donc essentiel, et c'est une question qu'il faudrait étudier finement.

D'autre part, nous n'avons pas abordé le problème de la diversité produit : le travail sur un cas d'étude plus complexe se révèle être une nécessité pour valider ce point. Le travail sur ce nouveau cas serait par ailleurs l'occasion de valider la robustesse et l'agilité de notre solution vis-à-vis de la mise à l'échelle : dans quelle mesure l'extension du méta-modèle d'unification défini perturbe-t'elle la médiation déjà en place ?

Enfin, nous avons réussi à surmonter les problèmes d'implémentation de la norme STEP grâce au composant logiciel pySDAI. C'est la partie de notre travail qui a demandé le plus d'efforts et d'investissement, et nous tenons à partager ces développements pour éviter à un tiers, dans la mesure du possible, d'avoir à reprendre à l'avenir ce travail dans son intégralité. Le code source du composant pySDAI est disponible sur la page :

http ://pysdai.svn.sourceforge.net/viewvc/pysdai/trunk/

4.4. Implémentation du modèle de tags sémantiques

Le modèle UML de la figure 3.4.4 est implémenté dans le langage de programmation python. Nous avons baptisé *Product Semantic Tagging* (PST) le composant qui rassemble l'ensemble de nos développements concernant le modèle de tags sémantiques, et qui propose les fonctionnalités suivantes :

- créer des instances pour chacune des classes et des relations définies dans le modèle UML ;
- filtrer les items par rapport aux tags auxquels ils sont liés ;
- reconstruire des structures produit multi-vues à partir d'un algorithme basé sur l'analyse de ces tags.

Les sections qui suivent précisent comment ces fonctions ont été implémentées et validées au regard des résultats attendus.

4.4.1. Implémentation des classes du diagrammes UML

Les classes `TaggedItem`, `SemanticTag`, et `SemanticDescriptionLink`, `DefineGroup` et `InGroup` sont implémentées de la manière suivante :

- Implémentation de la classe `TaggedItem`

```
1  class TaggedItem(object):
2    """ A class defining an item about to be tagged """
3    all_items = []
4
5    def __init__(self , name=''):
6      object.__init__(self)
7      self.all_items.append(self)
8      self._name = name
9      print 'Item %s created'%name
10
11   @property
12   def name(self):
13     return self._name
```

- Implémentation de la classe `SemanticTag`

```
1  # Loading unique identifier builtin python module
2  from uuid import uuid4
3
4  class SemanticTag(object):
5    def __init__(self , name , description=''):
6      self._uid = uuid4() # Sets a unique id for this tag
7      self._name = name
8      self._description = description
9      print "SemanticTag '%s' created."%name
10
11   def get_uid(self):
12     return self._uid
13
14   def get_name(self):
15     return self._name
16
17   def get_description(self):
18     return self._description
```

- Implémentation des classes `SemanticDescriptionLink`, `DefineGroup` et `InGroup`

```
1  #
2  # Base class
3  #
4  class SemanticDescriptionLink(object):
5    """ Connects an item to a semantic tag - Superclass of DefineGroup and
         InGroup """
6    # properties shared by all instances
7    _semantic_description_link_instances = []
8    _define_group_instances = []
9    _in_group_instances = []
```

```
10
11    def __init__(self , item , tag):
12      self._item = item
13      self._tag = tag
14      if isinstance(self , DefineGroup):
15        self._define_group_instances.append(self)
16        print "'%s' ------> defined as group '%s'"%(item.get_name(),tag.
              get_name())
17      elif isinstance(self,InGroup):
18        self._in_group_instances.append(self)
19        print "'%s' ------> in group '%s'"%(item.get_name(),tag.get_name())
20      else:
21        self._semantic_description_link_instances.append(self)
22        print "'%s' ------> tagged '%s'"%(item.get_name(),tag.get_name())
23
24    def get_tag(self):
25      return self._tag
26
27    def get_item(self):
28      return self._item
29
30    def get_all_connections(self):
31      return self._semantic_description_link_instances
32
33    def __del__(self):
34      """ If a link is removed, the connection is removed from the
            connections list """
35      self._semantic_description_link_instances.remove(self)
36
37  #
38  # DefinedGroup and InGroup classes inherit from SemanticDescriptionLink
39  #
40  class DefineGroup(SemanticDescriptionLink):
41    """ Used to define a semantic group """
42    def __init(self , *kargs):
43      SemanticDescriptionLink.__init__(self,*kargs)
44
45  class InGroup(SemanticDescriptionLink):
46    """ Used to define a semantic group """
47    def __init(self , *kargs):
48      SemanticDescriptionLink.__init__(self,kargs)
```

4.4.2. Instanciation des items, des tags et des relations

Les classes précédemment définies permettent par la suite d'instancier une population d'objets. Cette instanciation est présentée dans les paragraphes qui suivent.

Instanciation des item L'instanciation d'un `TaggedItem` peut se faire depuis une console python :

```
macbook-pro-de-thomas-paviot:semantic_tag thomas$ python
Python 2.6.1 (r261:67515, Feb 11 2010, 00:51:29)
[GCC 4.2.1 (Apple Inc. build 5646)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from PST import *
>>> my_tagged_item = TaggedItem('an item')
TaggedItem 'an item' created.
>>>
```

Si le nombre d'items à créer est important, on préférera une création par lots à l'aide de la fonction `create_tagged_item_list` :

```
1 def create_tagged_item_list( list_str ):
2   """ Returns a list of tagged item """
3   tagged_item_list = []
4   for tagged_item_name in list_str:
5     new_tagged_item = TaggedItem(tagged_item_name)
6     tagged_item_list.append(new_tagged_item)
7   return tagged_item_list
```

Cette fonction sera utilisée de la manière suivante :

```
>>> from PST import *
>>> my_item_list = create_tagged_item_list(['first_item','second_item','
    third_item'])
TaggedItem 'first_item' created.
TaggedItem 'second_item' created.
TaggedItem 'third_item' created.
>>> print my_item_list
[<PST.TaggedItem object at 0x100585ad0>, <PST.TaggedItem object at 0
    x10052e9d0>, <PST.TaggedItem object at 0x10052ea10>]
>>>
```

Instanciation des tags sémantiques De la même manière que pour les objets de la classe `TaggedItem`, les instances de la classe `SemantigTag` pourront être construites depuis la console :

```
>>> from PST import *
>>> my_tag = SemanticTag('my_tag_name','my_tag_description')
SemanticTag 'my_tag_name' created.
>>> my_tag.get_name()
'my_tag_name'
>>> my_tag.get_description()
'my_tag_description'
```

Ou *via* un traitement par lots en utilisant la fonction `create_semantic_tags` :

```
1 def create_semantic_tags(list_str):
2   ''' Returns a list of semantic_tags '''
3   tag_list = []
4   for tag_name in list_str:
5     new_tag = SemanticTag(tag_name)
6     tag_list.append(new_tag)
7   return tag_list
```

Dans ce cas, on créera aisément un ensemble d'instances de `SemanticTag` à partir de leur nom :

```
>>> from PST import *
>>> my_tag_1,my_tag_2,my_tag_3=create_tags(['tag1','tag2','tag3'])
SemanticTag 'tag1' created.
SemanticTag 'tag2' created.
SemanticTag 'tag3' created.
>>> my_tag_1.get_name()
'tag1'
>>> my_tag_2.get_name()
'tag2'
>>> my_tag_3.get_name()
'tag3'
```

Instanciation de relations sémantiques Les relations sémantiques permettent de connecter des items à des tags sémantiques. Nous avons défini plusieurs fonctions pour réaliser ces connexions entre instances de `TaggedItem` et instances de `SemanticTag`. Les connexions possibles sont au nombre de trois : `SemanticDescriptionLink`, `DefineGroup` et `InGroup` :

```
>>> from PST import *
>>> my_tag = SemanticTag('tag')
SemanticTag 'tag' created.
>>> my_tagged_item = TaggedItem('item')
TaggedItem 'item' created.
>>> my_rel = SemanticDescriptionLink(my_tagged_item,my_tag)
>>> 'item' ------> tagged 'tag'
```

Un ensemble de fonctions pour traiter par lots ces relations sémantiques a également été développé.

4.4.3. Algorithmes de traitement du modèle

Nous détaillons ci-dessous deux algorithmes que nous avons définis : les « filtres » et les « reconstructions de structure produit ».

Filtres Les « Filtres » ont pour fonction de balayer l'ensemble des items et de retourner ceux qui portent une étiquette sémantique donnée. Ainsi l'algorithme implémenté parcourt l'ensemble des connexions de type `SemanticDescriptionLink`, vérifie si le tag associé à cette connexion sémantique correspond à celui passé en paramètre, et ajoute dans une liste l'instance de `TaggedItem` :

```
1  def filter_by_tag(tag,verbose=True):
2    ''' returns items that match the tag passed as parameter '''
3    matched_items = []
4    for connection in SemanticDescriptionLink.
         _semantic_description_link_instances:
5      if connection.get_tag() == tag:
6        item = connection.get_item()
7          if verbose:
8            print "Found : item '%s' matches tag '%s'"%(item.get_name(),tag.
                 get_name())
9          matched_items.append(connection.get_item())
10   return matched_items
```

Reconstruction de structure produit Cette partie fait directement écho au problème introduit dans la section 3.4 : comment générer la structure produit à partir d'informations sémantiques ? Dans le listing suivant, c'est la fonction `get_view` qui est chargée d'afficher la structure produit à partir d'un tag passé en paramètre :

- `get_view` recherche tout d'abord l'item qui est à considérer comme la tête de nomenclature ;
- l'appel récursif à la fonction `traverse_item` permet de parcourir l'ensemble des enfants qui portent le même tag.

C'est une structure complète et non à un seul niveau qui est retournée par cette fonction.

```
1  def get_view(view_tag):
2    ''' returns the structure view    '''
3    print "###### generating '%s' view ######"%view_tag.get_name()
4    # First get the set of items that match the tag
5    matched_items = filter_by_tag(view_tag,verbose=False)
6    # From the list of matched items, look for the one which do no belong to
         any group
7    # this item will be the root node of the view
8    root_node = None
9    for item in matched_items:
10     if get_groups_for_item(item)==[]:
11       root_node = item
12   print 'Root Node for view %s: %s'%(view_tag.get_name(),root_node.
         get_name())
13   traverse_item(root_node)
14
15 level = 0 #global variable to store recursion level
16
17 def traverse_item(item):
```

```
18    ''' Build the product structure according to the '''
19    global level
20    print '\t'*level,item.get_name()
21    level += 1
22    # first find the groups defined for this item
23    item_groups = get_group_item(item)
24
25    if len(item_groups) > 0:
26      # Find the child item for this group
27      children = get_children_of_group(item_groups[0])
28      for child in children:
29        traverse_item(child)
30    level -= 1
```

Résultats Les éléments présentés dans la section 3.4.5 sont instanciés de la manière suivante :

```
1  # Create items
2  print "========== Creating items ==========="
3  tagged_item_names = ['Tole1','Tole2','Tole1_Prime','Tole2_Prime','Vis3','
       Ecrou4','Boulon','Assemblage']
4  tole_1, tole_2,tole_1_prime,tole_2_prime,vis_3,ecrou_4,boulon,assemblage =
       create_tagged_item_list(tagged_item_names)
5  print ""
6  # Create tags
7  print "========== Creating tags ==========="
8  tag_design = SemanticTag(name = 'Design' , description ='Design context')
9  tag_manufacturing = SemanticTag(name = 'Manufacturing' , description = '
       Manufacturing context')
10
11 # Create groups
12 print "========== Creating Group tags ==========="
13 group_DEG1 = SemanticTag(name = 'DEG1')
14 # the assembly
15 group_PRG_Main = SemanticTag(name = 'PRG_Main')
16 group_PRG1 = SemanticTag(name = 'PRG1')
17 # the tole1
18 group_PRG2 = SemanticTag(name = 'PRG2')
19 # the tole2
20 group_PRG3 = SemanticTag(name = 'PRG3')
21 # the bolt
22 print "========== Connect items to tags ==========="
23 # Connect items to the tag_design tag
24 connect_items_to_tag([assemblage,tole_1,tole_2,vis_3,ecrou_4],tag_design)
25 # Connect items to the manufacturing tag
26 connect_items_to_tag([assemblage,tole_1,tole_1_prime,tole_2,tole_2_prime,
       boulon,vis_3,ecrou_4],tag_manufacturing)
27 # connect items to groups name
28 DefieGroup(assemblage,group_DEG1) # the assembly is a design group
29 DefineGroup(assemblage,group_PRG_Main) # the assembly is also a
       manufacturing group
```

```
30 DefineGroup(tole_1,group_PRG1) # tole 1
31 DefineGroup(tole_2,group_PRG2) # tole 2
32 DefineGroup(boulon,group_PRG3) # boulon
33 # connect items that belongs to groups
34 item_list_to_group([tole_1,tole_2,vis_3,ecrou_4],group_DEG1)
35 item_list__to_group([tole_1_prime],group_PRG1)
36 item_list__to_group([tole_2_prime],group_PRG2)
37 item_list__to_group([vis_3,ecrou_4],group_PRG3)
```

L'exécution de ce script depuis un terminal génère la sorte suivante :

```
========== Creating items ==========
TaggedItem 'Tole1' created.
TaggedItem 'Tole2' created.
TaggedItem 'Tole1_Prime' created.
TaggedItem 'Tole2_Prime' created.
TaggedItem 'Vis3' created.
TaggedItem 'Ecrou4' created.
TaggedItem 'Boulon' created.
TaggedItem 'Assemblage' created.
========== Creating tags ==========
SemanticTag 'Design' created.
SemanticTag 'Manufacturing' created.
========== Creating Group tags ==========
SemanticTag 'Design' created.
SemanticTag 'DEG1' created.
SemanticTag 'PRG_Main' created.
SemanticTag 'PRG1' created.
SemanticTag 'PRG2' created.
SemanticTag 'PRG3' created.
========== Connect items to tags ==========
'Assemblage' ------> tagged 'Design'
'Tole1' ------> tagged 'Design'
'Tole2' ------> tagged 'Design'
'Vis3' ------> tagged 'Design'
'Ecrou4' ------> tagged 'Design'
'Assemblage' ------> tagged 'Manufacturing'
'Tole1' ------> tagged 'Manufacturing'
'Tole1_Prime' ------> tagged 'Manufacturing'
'Tole2' ------> tagged 'Manufacturing'
'Tole2_Prime' ------> tagged 'Manufacturing'
'Boulon' ------> tagged 'Manufacturing'
'Vis3' ------> tagged 'Manufacturing'
'Ecrou4' ------> tagged 'Manufacturing'
'Assemblage' ------> defined as group 'DEG1'
'Assemblage' ------> defined as group 'PRG_Main'
'Tole1' ------> defined as group 'PRG1'
'Tole2' ------> defined as group 'PRG2'
'Boulon' ------> defined as group 'PRG3'
'Tole1' ------> in group 'DEG1'
'Tole2' ------> in group 'DEG1'
'Vis3' ------> in group 'DEG1'
```

```
'Ecrou4' ------> in group 'DEG1'
'Tole1_Prime' ------> in group 'PRG1'
'Tole2_Prime' ------> in group 'PRG2'
'Vis3' ------> in group 'PRG3'
'Ecrou4' ------> in group 'PRG3'
```

Trois requêtes de type filtre sont testées :

- filtrage par le tag `tag_design`
- filtrage par le tag `tag_manufacturing`
- filtrage par les tag `tag_design` ET `tag_manufacturing` correspondant à l'intersection des deux ensembles précédents.

```
1 print "======= Request 1 : find items with the tag 'Design'"
2 items_with_tag_design = filter_by_tag(tag_design)
3 print "======= Request 2 : find items with the tag 'Manufacturing'"
4 items_with_tag_manufacturing = filter_by_tag(tag_manufacturing)
5 print "======= Requets 3 : find items with the tag 'Design' and '
    Manufacturing'"
6 r = list(set(items_with_tag_design).intersection(set(
    items_with_tag_manufacturing)))
7   for item in r:
8     print "Found : '%s'"%item.get_name()
```

Ce qui produit la sortie suivante :

```
======= Request 1 : find items with the tag 'Design'
Found : item 'Assemblage' matches tag 'Design'
Found : item 'Tole1' matches tag 'Design'
Found : item 'Tole2' matches tag 'Design'
Found : item 'Vis3' matches tag 'Design'
Found : item 'Ecrou4' matches tag 'Design'
======= Request 2 : find items with the tag 'Manufacturing'
Found : item 'Assemblage' matches tag 'Manufacturing'
Found : item 'Tole1' matches tag 'Manufacturing'
Found : item 'Tole1_Prime' matches tag 'Manufacturing'
Found : item 'Tole2' matches tag 'Manufacturing'
Found : item 'Tole2_Prime' matches tag 'Manufacturing'
Found : item 'Boulon' matches tag 'Manufacturing'
Found : item 'Vis3' matches tag 'Manufacturing'
Found : item 'Ecrou4' matches tag 'Manufacturing'
======= Requets 3 : find items with the tag 'Design' and 'Manufacturing'
Found : 'Assemblage'
Found : 'Tole1'
Found : 'Tole2'
Found : 'Ecrou4'
Found : 'Vis3'
```

Enfin, nous demandons la reconstruction des EBOM et MBOM de la façon suivante :

```
print "======= Request 4 : Display the product structure for the tag '
    Design'"
get_view(tag_design)
print "======= Request 5 : Display the product structure for the tag '
    Manufacturing'"
get_view(tag_manufacturing)
```

Qui produit le résultat :

```
======= Request 4 : Display the product structure for the tag 'Design'
##### generating 'Design' view ###
### Root Node for view Design: Assemblage ###
Assemblage
        Tole1
        Tole2
    Vis3
        Ecrou4
======= Request 5 : Display the product structure for the tag '
    Manufacturing'
##### generating 'Manufacturing' view ###
### Root Node for view Manufacturing: Assemblage ###
Assemblage
    Tole1
        Tole1_Prime
    Tole2
        Tole2_Prime
    Boulon
        Vis3
        Ecrou4
```

Nous constatons que les résultats concernant les structures produit sont bien en conformité avec les EBOM et MBOM définies sur la figure 3.4.2.

Conclusions Nous avons pu mettre en oeuvre un algorithme simple pour la reconstruction des structures produit. Cependant, nous avons dû utiliser des opérations d'union ou d'intersection entre ensembles pour filtrer les items suivant un critère de tags multiples (par exemple tag1 ET tag2, tag1 OU tag2). Nous avons expérimenté deux autres approches, qui ne sont pas détaillées dans ce mémoire, pour espérer produire des résultats plus étoffés :

- nous avons employé une technique consistant à surcharger les opérateurs de la classe `SemanticTag` pour être capable de créer des opérations booléennes multiples entre tags, par exemple l'expression booléenne ((tag1 and tag2) or tag3). Cette approche limite tout de même l'étendue des règles que nous pouvons prendre en charge aux seules opérations booléennes ;
- nous avons aussi mené quelques expérimentations consistant à utiliser les graphes conjonctifs RDF associés à des requêtes SPARQL (ceci en utilisant la bibliothèque de

développement pour python rdflib [17]) : la syntaxe des requêtes SPARQL est difficile à prendre en main et, de toute manière, il faut tout de même utiliser un algorithme pour reconstruire les structures produit.

Par conséquent, l'utilisation d'un moteur d'inférence s'avère nécessaire dans le cas où les règles d'affectation des tags (par exemple des contraintes d'unicité ou d'exclusivité) sont plus poussées que dans l'exemple que nous avons proposé. Ce moteur d'inférence doit cependant pouvoir s'intégrer à notre environnement de développement, et être interopérable avec les briques logicielles que nous avons développées.

Enfin, nous avons pu entrevoir au cours de ces développements une perspective intéressante : avec le même composant logiciel PST, il est possible d'affecter des tags sémantiques à n'importe quel objet instancié depuis le langage de programmation Python. Ceci est vrai pour les instances de classes définies par l'utilisateur, mais également pour les fonctions, qui sont considérées par Python comme des objets. Il est ainsi possible d'affecter des annotations sémantiques à des fonctions et donc reconstruire une structure de fonctions à partir du même algorithme que celui que nous avons développé pour la reconstruction de structures produits. Nous voyons là une implication importante : cette technique pourrait être utilisée pour reconstruire des processus multi-vues à partir de l'étiquetage sémantique de fonctions élémentaires. Nos premières expérimentations dans ce domaine sont prometteuses mais pas suffisamment avancées pour faire l'objet d'un compte-rendu dans ce mémoire. Nous allons néanmoins poursuivre ce travail avec l'espoir qu'il permette de réconcilier les données produit et les processus qui leurs sont associés.

4.5. Recommandations méthodologiques relatives à l'implémentation

Les expérimentations que nous avons menées nous amènent aux préconisations suivantes pour ce qui concerne l'implémentation de modèles dans le cadre de la résolution de problèmes d'interopérabilité. Ainsi, il faudra dans la mesure du possible :

- **utiliser des langages de scripts dynamiques multi-plateformes** : ce sont les seuls à offrir à la fois une grande souplesse d'utilisation (pas de phase de compilation nécessaire par exemple) et à garantir la **portabilité** des programmes (les scripts s'exécutent de la même manière sur des plateformes différentes sans modification), critère d'une importance cruciale pour assurer la robustesse de la solution à un problème d'interopérabilité multi-échelle. Notre expérience nous a appris que Python est un langage de programmation particulièrement pertinent dans ce contexte ;
- **baser les développements sur des composants logiciels libres dont l'activité est avérée** ;

17. http ://www.rdflib.net/

- **baser les développements sur des standards ouverts.** On s'assurera au préalable que les composants logiciels qui implémentent ces standards sont disponibles pour le langage de programmation choisi ;
- **dans le cadre expérimental d'un travail de recherche scientifique, utiliser des outils métier libres** pour lesquels l'accès au modèle de données est possible et qui permettent aux collègues de reproduire et vérifier les résultats ;
- dans le cadre d'une expérimentation menée en partenariat avec une entreprise industrielle, chacun des points précédents n'est pas nécessairement possible, en particulier celui concernant les outils métiers libres : le contexte industriel peut imposer des outils propriétaires pour des techniques, historiques (les systèmes à rendre interopérables sont installés/utilisés/maintenus depuis de longues années) ou commerciales (accord de partenariat avec un éditeur particulier). Il est néanmoins possible de rendre ces outils propriétaires interopérables en s'appuyant sur des composants logiciels libres et des techniques de contournement de problèmes réputés comme sans solution (techniques de webservicisation COM ou HTTP, accès aux DLL via ctypes etc.).

Résumé du chapitre 4

Ce chapitre détaille l'implémentation de l'architecture multi-échelle orientée services et du composant chargé de la médiation des systèmes, ces deux points ayant été présentés dans les précédents chapitres. Nous validons dans un premier temps l'architecture en étudiant deux catégories de problèmes qui modélisent les problèmes généraux d'interopérabilité visant à échanger des informations :

- la médiation en flux poussé : le médiateur pilote le transfert des informations entre les deux systèmes ;
- la médiation en flux tiré : un système adresse une requête au médiateur qui se charge alors de récupérer l'information auprès du système identifié comme possédant cette information, pour la restituer *in fine* au système demandeur.

Nous présentons les résultats concernant les problèmes d'interopérabilité CAO/PDM et PDM/ERP, et nous détaillons l'implémentation du modèle d'annotations sémantiques du produit. Tous ces développements sont conduits à l'aide du langage de programmation Python, identifié comme pertinent pour notre contexte de recherche. Ces développements sont organisés sous la forme de composants logiciels (ou bibliothèques), placés sous licence libre, qu'il est donc possible d'utiliser pour résoudre des problèmes d'interopérabilité qui revêtent une autre nature, ou qui concernent d'autres phases du cycle de vie du produit que celles sur lesquelles nous avons basé notre travail.

5. Conclusion - Perspectives

Ce mémoire a décrit un travail de recherche relatif à l'interopérabilité des systèmes d'information dans le domaine du Product Lifecycle Management. Nous avons essayé de présenter ce mémoire en balayant un large spectre suivant deux dimensions :

- du global vers le local : nous avons fait le lien entre des problématiques scientifiques de haut niveau (architectures, modèles) et des considérations pragmatiques concernant l'implémentation ;
- mutli-disciplinarité : nous avons emprunté à divers champs de la connaissance (sciences de la production, de la conception, de l'informatique, jusqu'à la linguistique) pour construire un ensemble cohérent vis-à-vis de la problématique exposée.

Outre l'aspect scientifique, nous avons fait l'effort de donner à ce mémoire une dimension pédagogique dans un souci constant de capitalisation de l'expérience : nous avons souhaité donner au lecteur toutes les clés lui permettant de reproduire les résultats présentés, et ainsi faire en sorte que la poursuite de cette recherche puisse se faire dans les meilleures conditions. A cet effet, nous reprenons à notre compte la citation liminaire du mémoire d'Habilitation à Diriger des Recherches du Pr. Samir Lamouri [Lamouri, 2006] : « le savoir est la seule ressource qui croît lorsqu'on le partage », ce partage étant dans notre cas facilité par le fait que notre recherche a été menée hors de tout cadre contractuel avec une entreprise privée.

Ce dernier chapitre dresse le bilan de notre travail et l'inscrit dans un processus de recherche qui est toujours en cours. La section 5.1 résume les grandes lignes de notre contribution au regard des problèmes soulevés en introduction. La section suivante (voir 5.2) ouvre sur les perspectives de recherche que nous souhaitons mettre en œuvre à court terme pour pallier à certaines insuffisances. Enfin, nous terminons ce mémoire par l'exposé (voir 5.3) de quelques éléments prospectifs qui nous semblent se dessiner à plus long terme et auxquels nous nous préparons d'ores et déjà.

5.1. Apports de la thèse

Compte-tenu de notre connaissance de l'état de l'art relatif à la littérature scientifique sur le PLM et aux problématiques d'interopérabilité, nous pensons apporter les contributions suivantes :

1. nous proposons un point de vue nouveau sur le PLM et l'interopérabilité des SI qui s'appuie sur la notion de complexité ;
2. nous introduisons la notion de médiation multi-échelle orientée services, dont nous soutenons qu'elle peut être mise en oeuvre pour résoudre tout type de problème d'interopérabilité dans le domaine du PLM ;
3. nous précisons la position sémantique de chacun des protocoles d'application de la norme STEP sur une cartographie 3D ;
4. nous mettons en oeuvre le protocole d'application 239 de la norme STEP intitulé *Product Lifecycle Support* ;
5. nous introduisons un modèle d'annotations sémantiques des composants du produit qui vise à séparer la sémantique du produit de ce que nous avons appelé ses « matières d'oeuvre » pour générer des structures produit dans un contexte multi-vues ;
6. enfin, nous offrons à la communauté un ensemble de composants logiciels dédiés aux problèmes d'interopérabilité impliquant des systèmes CAO/PDM/ERP, en espérant qu'ils puissent ainsi s'étoffer du travail d'autres chercheurs ;

Le travail relatif à l'intégration CAO/PDM présenté dans la section 4 a fait l'objet d'une publication scientifique [Paviot et al., 2008]. L'architecture fonctionnelle et technique basée sur un modèle d'unification STEPAP239 (cf chapitres 2 et 3) a été validée scientifiquement [Paviot et al., 2009], de même que le modèle d'annotations sémantiques des composants du produit présenté dans la section 3.4.4 [Paviot et al., 2010].

Nous avons construit ce mémoire avec le souci de la cohérence, mais tous les éléments présentés n'offrent pas le même degré de maturité. La rédaction de ce mémoire est donc l'opportunité de poser un jalon dans un processus de recherche toujours en cours. Nous détaillons dans la section suivante les points sur lesquels nous souhaitons insister pour poursuivre ce travail.

5.2. Travaux complémentaires à envisager

Bien que visant une portée générale, notre travail de recherche souffre encore de questions ouvertes qu'il conviendrait d'explorer plus en détail :

1. devant l'étendue des problèmes d'interopérabilité, nous nous sommes concentré sur les données relatives au produit, en occultant consciemment la partie processus : le lien entre les données et les processus reste encore à étudier ;
2. l'interopérabilité entre les domaines de la conception et de la production s'est limitée au module MRP de l'ERP. Sans doute serait-il intéressant de chercher à unifier davantage d'informations, pour amener au décideur des informations financières dès les premières phases de conception ;

3. nous souhaiterions profiter de la proximité entre l'AP239 (que nous avons étudié) et l'AP233 (qui reste à étudier) pour étendre nos recherches au domaine des produits mécatroniques, et ainsi initier au sein du LISMMA une recherche conjointe avec les collègues spécialistes de ce métier ;
4. la notion de distance sémantique a été présentée et utilisée d'une manière qualitative : pour aller plus loin, nous souhaitons quantifier cette notion, car nous pensons qu'elle est au coeur du problème du traitement automatique des données relatives au produit. Nous devrons d'autre part utiliser toute les technologies récentes de modélisation et d'encodage des connaissances. Ces travaux pourraient permettre d'aller beaucoup plus loin dans notre travail sur les tags sémantiques ;
5. du côté de l'implémentation, nous avons été confronté à des difficultés liées aux différences fondamentales entre les techniques de la programmation orientée objets (POO) et celles liées à la modélisation des connaissances. Nous ne sommes pas satisfait de l'ajout d'une « couche » connaissance sur un langage objet et appelons de nos voeux un langage de programmation qui soit intrinsèquement basé sur la sémantique et qui remette en cause les notions de base de la POO. L'implémentation de règles et d'algorithmes d'inférences en serait grandement facilitée ;
6. enfin, nous souhaiterions mettre en conformité notre méthodologie avec les méthodes standardisées de l'ingénierie dirigée par les modèles (*Model Driven Engineering*).

Dans la section suivante, nous présentons les évolutions à plus long terme auxquelles nous nous préparons.

5.3. Perspectives

Cette partie concerne les évolutions possibles du PLM compte tenu de la veille technologique et scientifique que nous avons menée durant notre recherche. Rien ne préjuge de leur réalisation, rien ne permet d'affirmer leur vérité. Cependant, les éléments qui suivent sont argumentés. Cette section constitue donc un exercice de « prospective pragmatique » ayant pour but de contribuer au débat relatif aux voies de recherches qui pourraient être explorées à l'avenir.

TRIZ comme aide à la prospective La théorie de résolution des problèmes inventifs (TRIZ, [Altshuller et al., 1999]) fournit un ensemble d'outils favorisant la créativité dans des phases de recherche de solutions (la deuxième phase du modèle de Pahl et Breizh présenté dans la section 1.2.1.1). Parmi ceux-ci, les huit lois d'évolutions des systèmes techniques sont pertinentes dans une étude prospective pour dessiner les évolutions probables d'un système [Crubleau and Richir, 2001]. Un système technique ne respectant pas ces lois d'évolution sera considéré comme non-viable et sera inévitablement remplacé par un système concurrent qui les respecte.

Genrich Alstshuller, inventeur de cette méthode, définit un système technique comme *une combinaison d'éléments réunis de manière à former un ensemble, qui possède une propriété globale qu'aucun élément pris séparément ne possède.* Le système d'information du PLM est donc un système technique au regard de cette définition, et nous pouvons appliquer les concepts de TRIZ à ce système. Parmi ces huit lois, trois retiennent notre attention dans cette dernière partie :

- transition vers le super système (loi n°6) : lorsqu'un système technique a épuisé ses possibilités de développement, il intègre un « super-système » dont il devient une partie. Son développement se poursuit à l'intérieur de ce « super-système » ;
- transition vers le micro niveau (loi n°7) : tout système a tendance à se diviser en parties de dimension plus petites. Les liaisons entre ces parties ont tendance à se dynamiser ;
- augmentation de la contrôlabilité et du dynamisme (loi n°8) : un système technique tend vers un niveau de contrôlabilité accru, pour atteindre un niveau d'auto-contrôle. L'évolution du système tend ainsi vers une diminution de l'intervention humaine, cette évolution se décomposant en quatre phases :
  - phase 1 (initiale) : le système requiert l'intervention humaine à tous les niveaux ;
  - phase 2 : diminution de la fonction humaine au niveau de l'exécution ;
  - phase 3 : diminution de la fonction humaine au niveau du contrôle ;
  - phase 4 : diminution de la fonction humaine au niveau de la prise de décision.

Les systèmes d'échelle mésoscopique ont donc tendance à s'automatiser et à évoluer simultanément vers des échelles micro et macroscopiques. Cette prédiction de la multiplication du nombre d'éléments dans le système et de leurs interrelations conduit à anticiper un accroissement de la complexité du système PLM. Les conséquences de cet accroissement prévisible sont examinées dans les quelques sections qui suivent.

5.3.1. Vers des architectures distribuées

Le concept de *grid computing* ou *cloud computing* a émergé récemment [Wang et al., 2008]. D'après [Foster and Kesselman, 2004], *une grille est un système d'information de très grande dimension, constitué de plusieurs ordinateurs distribués suivant plusieurs organisations, qui peut être étendu à l'échelle de l'internet tout entier.* [Krauter et al., 2002] classent ces grilles en trois catégories :

- les grilles de calcul (*computational grids*), pour améliorer les performances des applications ;
- les grilles de données (*data grids*) pour améliorer l'accès aux données ;
- les grilles de services (*service grids*), pour améliorer les services.

Des exemples de systèmes basés sur une architecture distribuée sont déjà en service. Le projet World Community Grid [1], démarré en 2005, met en relation des scientifiques ayant besoin de fortes capacités de calcul avec des particuliers prêts à céder une partie de la

1. http ://www.worldcommunitygrid.org/

puissance de leur ordinateur personnel. Basé sur une technologie développée par IBM, des simulations lourdes (en médecine, physique, météorologie, astrophysique, physique nucléaire) sont exécutées simultanément, et de manière synchronisée, sur des milliers d'ordinateurs personnels. Par exemple, la phase 1 du *Clean Energy Project*, visant à concevoir de nouvelles structures cristallines pour des panneaux solaires à haut rendement, s'est achevée le 12 octobre 2009, à peine 1 an après son lancement, et a associé de l'ordre de 120 000 ordinateurs ayant contribué à un temps global équivalent à 2 000 ans de calcul (si le calcul avait été mené sur un seul ordinateur). Ce projet entre dans la catégorie des grilles de calcul.

Une autre évolution en cours se situe dans le domaine des bases de données. Le modèle dominant sur le marché est aujourd'hui constitué des bases de données relationnelles qui utilisent un système de requêtes de type *System Query Language* (SQL). Les leaders du marché [2] proposent des serveurs de base de données reposant sur une architecture centralisée (Oracle 10g2, Microsoft SQL server, IBM DB2). Cependant, les dernières années ont vu l'éclosion de produits ayant des caractéristiques particulières en terme de montée en charge. Le projet Cassandra [3], soutenu par la *Apache foundation* (éditeur du serveur web éponyme) a pour objectif de proposer un gestionnaire de base de données qui n'utilise pas SQL et qui soit basé sur une architecture distribuée : les données sont stockées sur des grappes pouvant être constituées de plusieurs milliers d'ordinateurs. Ce programme a été placé sous licence libre par la société Facebook et se trouve adapté à des bases de données comptant plusieurs milliards d'entrées, dont le type est une chaîne de caractères, devant faire face à de très fortes contraintes de montée en charge ou de mise à l'échelle (la société Facebook est passée de 0 à plusieurs millions de comptes en moins de 2 ans). Cassandra est soutenu par les acteurs majeurs du secteur de l'internet moderne (réseaux sociaux). La plus grosse grille de données mise en œuvre en production avec Cassandra représente 150To de données réparties sur 150 machines. Cette architecture offre en particulier la possibilité d'être robuste vis-à-vis de la défaillance d'un *data center* complet, et offre des performances accrues par rapport à du SQL.

Du côté de l'industrie logicielle, le travail collaboratif basé sur des architectures centralisées est une réalité depuis le milieu des années 1980. Le premier outil de gestion collaborative de code source, Concurrent Version System [4] (CVS) a été publié en 1986. C'est à cette occasion qu'ont été été introduits les premiers mécanismes de verrouillage/récupération/intégration/traçabilité des modifications/rôles/équipes que l'on trouve implémentés dans les systèmes PDM modernes. Ces dernières années, des outils basés sur des architectures distribuées ont fait leur apparition : Mercurial [5] en 2005, Bazaar [6] en 2006 ou encore Git [7] en 2005, ce dernier outil étant utilisé pour le développement

2. http ://www.lemondeinformatique.fr/actualites/lire-marche-des-bases-de-donnees-oracle-a-accru-son-avance-en-2006-22744.html
3. http ://cassandra.apache.org/
4. http ://ximbiot.com/cvs/
5. http ://mercurial.selenic.com/
6. http ://bazaar.canonical.com/
7. http ://git-scm.com/

collaboratif du noyau du système d'exploitation GNU [8]/Linux.

Dans le domaine qui nous intéresse plus directement, celui du développement des produits industriels, des travaux de recherche récents ont été menés dans le domaine des architectures distribuées : [Lee et al., 2009] proposent une grille de calcul permettant l'amélioration du processus MRP. [Jubertie, 2007] expérimente une architecture distribuée dédiée aux applications de Réalité Virtuelle, en utilisant la programmation par contraintes pour la définition du placement de la grappe d'ordinateurs.

L'alliance de l'ensemble grille de calcul/données/services appliquées au cas particulier du développement de produit et de la gestion des informations produit pourrait conduire à des projets de recherche liées à une « grille PLM » ou *grid PLM*. Les expérimentations associées à ce type de recherches nécessiteront cependant d'importantes ressources matérielles à disposition des seules grandes entreprises ou de groupements d'équipes de recherche.

5.3.2. De nouvelles méthodes pour de nouvelles échelles

La tendance du système à enfler (transition vers le super-système) et, dans le même temps, à se flexibiliser en éléments de taille de plus en plus petite amène un changement d'échelle : les systèmes actuels, d'échelle mésoscopique, évolueront simultanément vers des échelles microscopiques et macroscopiques. L'histoire des sciences physique nous enseigne que les changements d'échelle ont amené de grands bouleversements dans la pensée scientifique, les modèles existants ayant été battus en brèche par ces changements d'échelle. Ce sont les avancées dans le domaine de la physique statistique et de la thermodynamique qui au XIX^{ème} siècle ont permis de comprendre et décrire au niveau macroscopique les propriétés des gaz (systèmes complexes de particules). Puis au XX^{ème} siècle, les physiques quantiques et relativistes ont permis la transition entre le niveau mésoscopique et le niveau microscopique et macroscopique. Enfin, dans la deuxième partie du XX^{ème} siècle, les travaux liés à la dynamique des systèmes complexes fortement non-linéaires ont permis d'expliquer et comprendre les découplages entre phénomènes locaux et globaux (par exemple pour la modélisation des turbulences en météorologie).

Appliquées au domaine du PLM, les questions suivantes nous paraissent dignes d'intérêt :

- le comportement du système global ainsi constitué est-il *prédictible* ? Le fait de connaître et/ou contrôler les interactions entre ses constituants suffit-il à garantir sa prédictibilité ? En d'autres termes, peut-on s'assurer de la connaissance des états futurs du système connaissant son état initial et toutes les règles qui gouvernent ses interactions ? Nous serions tenté de répondre par la négative à cette question. Par exemple, les équations de Navier Stokes, qui modélisent les interactions entre volumes fluides élémentaires sont connues depuis plus d'un siècle et demi. Pourtant, la résolution de ces équations

8. http ://www.gnu.org/

en vue de connaître avec certitude l'état d'un fluide fait encore l'objet de nombreux travaux mathématiques [9]. Considérons un autre exemple faisant intervenir des êtres humains : celui d'une partie d'échec. La situation initiale de l'échiquier est parfaitement connue. Les règles sont connues et simples. Le processus est élémentaire (chaque joueur joue tour à tour). Le plateau est petit, le nombre de pièces est limité. Pourtant, même dans ce contexte très bien connu et très fortement cadré, avec très peu d'inconnues et de perturbations (contrairement à un projet industriel) il est impossible, au cours d'une partie, de prévoir avec certitude la position des pièces au bout de n coups. Même si des outils permettent de piloter les projets industriels, cette notion de prédictibilité nous semble être une question ouverte.

- en corollaire du point précédent, le système est-il *stable* ? existe-il des *instabilités* conduisant à un comportement *chaotique* ?
- comme pour toute installation industrielle critique, comment évaluer et maîtriser les *risques* ? Quels sont les moyens de *supervision* à mettre en œuvre ?
- quelle est la relation entre la *dynamique locale du système* et son comportement *global* ? Existe-t-il un *découplage* entre des phénomènes dynamique locaux et globaux ?

La réponse à ces questions nécessitera peut-être l'utilisation de techniques mathématiques avancées, voire l'analogie ou la collaboration avec des sciences ayant déjà proposé des éléments de réponse pour des problèmes analogues.

5.3.3. Solution intégrée propriétaire *vs.* Applications ouvertes interopérables

Dans ce contexte d'accroissement de la complexité, l'impact de la moindre ambiguïté sera de plus en plus problématique. Tout voile jeté sur la sémantique véhiculée dans le système sera préjudiciable au bon fonctionnement du SI. A cet égard, nous nous interrogeons sur la stratégie des éditeurs de solutions propriétaires, qui filialisent des entreprises concurrentes par le biais de rachats et intègrent dans leurs logiciels les nouveaux outils ainsi acquis. La solution qui en résulte couvre un spectre fonctionnel plus large, mais sous une forme monolithique et une approche « boîte noire ». Nous faisons le pari que ces sociétés devront faire évoluer leur modèle économique pour garantir à leurs clients la transparence sémantique de leurs produits. Elles devront ouvrir leurs modèles le plus largement possible et se poser ainsi en maîtres de la complexité multi-échelle. Le contexte technologique favorise aujourd'hui l'émergence sur le marché de nouveaux entrants qui savent inventer des modèles économiques basés sur la technologie multi-échelle, la simplicité, l'ouverture et le respect des standards. C'est ainsi que la société Google a relégué en quelques années au « rang de dinosaure » le champion de l'informatique mondiale de la fin du XX^{ème} sicècle, Microsoft, qui voit aujourd'hui son navigateur internet phare emprunter la pente du déclin [10]. Dans le domaine de l'ingénierie logicielle, nul ne peut se prétendre à l'abri de pareille mésaventure, quelles que soient sa taille et ses performances économiques.

9. http ://www.claymath.org/millennium/
10. Page mise à jour chaque mois : http ://en.wikipedia.org/wiki/Usage_share_of_web_browsers

A cet égard, nous considérons le modèle du logiciel libre comme pertinent d'un point de vue scientifique et industriel pour garantir l'interopérabilité des systèmes d'information, et nous pensons qu'il devrait recevoir toute l'attention de la communauté du PLM.

5.3.4. Vers une machine automatisée de conception

La huitième et dernière loi d'évolution des systèmes techniques laisse présager de l'évolution du système d'information du PLM vers une machine de conception automatisée, qui ne requerrait idéalement (au sens de l'idéal formulé par Alshuller) aucune intervention humaine. Nous voyons à long terme se dessiner une telle machine d'une grande complexité, qui serait la conséquence :

- des travaux relatifs à la capitalisation et la diffusion des connaissances ;
- des progrès quant au raisonnement sur ces bases de connaissance ;
- des progrès dans l'interopérabilité ;
- de l'évolution vers des architectures distribuées de type grille.

Nous proposons d'illustrer ce point de vue par une analogie entre une telle machine et un autre élément d'une grande complexité : le cerveau humain. Ainsi :

- les très nombreux composants de cette machine de conception seraient analogues aux neurones, briques élémentaires très spécialisées capables de former un système intelligent lorsqu'elles sont connectées par milliards ;
- l'interopérabilité entre les SI qui composent la machine pourraient être vues comme les connexions entre neurones : deux SI échangent des informations *via* des réseaux informatiques, les neurones échangent quant à eux des informations par l'intermédiaire de messages électriques véhiculés par les synapses ;
- l'encodage de la connaissance de l'entreprise serait équivalente à l'ADN des neurones : c'est de cette connaissance, entre autres, que naîtrait le produit final. Un problème dans l'encodage de ces connaissances conduirait à des aberrations sur le produit final ;
- la mémoire totale de cette machine serait distribuée sur chacun des éléments du système, et non contenue toute entière dans un endroit précis du système (de la même manière qu'il n'a pas encore été trouvé de zone restreinte du cerveau dans laquelle serait localisée toute la mémoire d'un être humain).

Cette évolution vers un système automatisé de conception, même si elle peut être raisonnablement prédite par les lois de TRIZ, est-elle pour autant souhaitable ? Quelles peuvent être les conséquences de cette évolution sur des hommes qui utilisent aujourd'hui ces outils comme une assistance (on parle bien de nos jours de Conception *Assistée* par Ordinateur) ?

5.3.5. La place de l'homme face à cette évolution technologique

Nous voyons dans l'évolution actuelle du PLM, et plus globalement celle du développement collaboratif des produits industriels, de fortes similitudes avec l'évolution des systèmes de production initiée dans la deuxième partie du XIX^{ème} siècle et qui s'est poursuivie au cours du XX^{ème} siècle.

Que l'on songe à l'essor de l'industrie automobile américaine de la fin du XIX^{ème} siècle sous l'impulsion d'Henry Ford, au fantastique redressement de l'industrie japonaise exsangue après-guerre ou à sa capacité à surmonter le choc pétrolier de 1973, les vainqueurs de la compétition économique ont su inventer de nouvelles méthodes d'organisation du travail s'appuyant sur une maîtrise technologique éprouvée : taylorisme, kaizen, lean manufacturing etc. Au cours des dernières dizaines années, l'automatisation de l'outil de production s'est ainsi accompagné de nouvelles missions pour les hommes, de nouvelles responsabilités, de nouvelles compétences, de nouvelles manières d'interagir avec leurs collègues et leur environnement technologique. Aux images d'Épinal des *Temps Modernes* de Charlie Chaplin ou de l'usine Renault-Billancourt des années 30, dans lesquelles des centaines d'hommes accomplissaient des tâches répétitives, s'est progressivement substituée l'image d'ateliers largement automatisés placés sous la responsabilité de techniciens experts. L'atelier d'usinage de l'usine Aérolia de Méaulte illustre cette évolution : d'imposantes machines UGV produisent en un temps record des pièces d'une complexité et d'une précision inouïes, sous la surveillance bienveillante d'opérateurs de production chargés de leur supervision. Les gains de productivité ainsi dégagés par les nouvelles méthodes de production ont permis à la France de maintenir la part de l'industrie dans la valeur ajoutée alors que, dans le même temps, l'emploi industriel connaissait une chute importante : depuis 1990, le nombre d'emplois industriels directs a chuté de 20% d'après l'INSEE [11]. Nous pensons que cette tendance est structurelle. Demain des machines de Conception Grande Vitesse en lieu et place des techniciens de bureaux d'études ?

Le point actuel des développements du PLM nous semble coïncider avec le début de la seconde révolution industrielle (datée de la fin du XIX^{ème} siècle) : l'information s'annonce comme le carburant de l'industrie de demain comme le pétrole et l'électricité ont jadis bouleversé le paysage industriel. Les gisements d'informations sont inépuisables (livres, savoirs, savoir-faire, bases de données), ces informations sont transformées, raffinées, transportées. La maîtrise totale de cette chaîne logistique de l'information semble être le dessein des travaux scientifiques liés au PLM. Encore partielles, voire balbutiantes, des technologies de rupture émergent, qui obligent les entreprises industrielles à des réajustements organisationnels. Ces changements s'opèrent pour l'instant de manière incrémentale, mais la révolution n'a pas encore eu lieu : de nombreux verrous scientifiques et technologiques restent à lever. Il ne fait aucun doute qu'ils seront surmontés avec succès. Ainsi le rêve d'Aristote, chantre du développement intégral de l'individu rendu possible par le machinisme, serait-il sur le point d'être réalisé. Il exprimait ainsi son idéal dans *Politique* :

11. INSEE - Visage industriel 2009

> *Si la navette courait d'elle-même sur la trame, l'industrie n'aurait plus d'ouvrier. Si chaque outil pouvait exécuter sur sommation, ou de lui-même, la tâche qui lui est propre, l'architecte n'aurait plus besoin de manoeuvre, ni le maître d'esclave.*

La première phrase de cette citation nous interroge quant au sens de nos recherches. Que vont devenir les concepteurs chargés de tâches de conception routinière, dénoncées comme un frein à l'innovation et un gaspillage au sens du *lean design* ? Qu'adviendra-t-il des employés chargés de conduire des tests sur des prototypes physiques, présentés comme coûteux, longs et par conséquent appelés à disparaître ? Disposerons-nous d'un système de formation capable de permettre à chacun de trouver sa place dans ces nouvelles organisations ? A ce sujet, Daniel Cohen [Cohen, 2009] explique dans son dernier ouvrage les raisons du déclin de l'industrie britannique au cours du XIX$^{\text{ème}}$ siècle :

> *L'Angleterre, siège de la première révolution industrielle, perdra progressivement son ascendant sur le monde industriel. L'Angleterre n'a pas su adapter son système de formation. Les élites continueront de fréquenter les collèges chics où l'on apprend l'art des codes sociaux. En France et en Allemagne, les grandes écoles d'ingénieurs, créées pour combler le retard avec l'Angleterre, fourniront les cadres de la seconde révolution industrielle, celle de l'électricité et du moteur à explosion.*

Ainsi serait-il possible d'espérer allier développement technologique et humain, à la condition expresse que l'ensemble de la communauté scientifique ait pleinement conscience de ses responsabilités dans la transmission de son savoir.

A. Organisation de la norme STEP

La norme ISO 10303 se décompose en sous-ensembles appelés fascicules. Ces fascicules portent une référence telle que :
- le premier numéro de la référence est 10303 (le numéro de la norme ISO) ;
- le séparateur « - » ;
- le deuxième numéro est un nombre entre 10 et 1999 qui indique la nature du fasciule : environnement, modèle de données intégrés, top parts ;
- le séparateur « : » ;
- l'année de publication du document (4 chiffres).

Voici des exemples de cette indexation : 10303-239 :2005, 10303-1017 :2004 etc. La référence à un élément de la norme STEP permet donc, à la seule lecture de son indexation, d'en déterminer l'objectif.

Environnement Composé des parties :
- 1x : méthodes de descriptions des données,
- 2x : méthodes d'implémentation,
- 3x : méthodes et environnement pour les tests de conformité.

Modèles de données intégrés
- Resources intégrées :
  - 4x et 5x : ressources intégrées génériques
  - 1xx : intergated application resources
  - PLIB ISO 13584-20 : bibliothèque de composants
- 5xx : Application Integrated Constructs (AIC)
- 1xxx : Application Modules

Top Parts
- 2xx : protocoles d'application (AP) ;
- 3xx : absracts test suits (ATS) pour les APs ;
- 4xx : méthodes d'implémentation des APs.

B. Le langage de modélisation EXPRESS

Les données de la norme ISO-10303 STEP sont décrites et modélisées à l'aide du langage EXPRESS [ISO-10303-11, 1994]. Nous proposons d'introduire ici les concepts-clés permettant de s'approprier un modèle de données STEP à partir de sa description dans ce langage.

B.1. Introduction

EXPRESS a été choisi dès la naissance de la norme STEP comme le langage de description des données. EXPRESS est caractérisé par les points suivants :

- il est, comme UML, un langage de modélisation ayant une approche objets ;
- il est textuel et traitable par machine, satisfaisant à cet égard aux contraintes originelles de STEP ;
- il propose des contraintes d'intégrité, logiques et fonctionnelles.

Les notions de schéma, d'entité, d'attributs, de relations, de fonctions et de procédures, et d'interfaçage entre schémas sont détaillées dans les paragraphes suivants. Nous appuierons les concepts présentés sur des exemples issus du schéma EXPRESS du protocole d'application 239 de la norme STEP.

B.2. Schéma EXPRESS

Un schéma EXPRESS permet la définition de modèles de données modulaires pouvant être fractionnés en plusieurs schémas. Pour les modèles de données de taille importante, plusieurs schémas interconnectés composent l'intégralité du modèle de données. Ces sous-schémas correspondent à une décomposition fonctionnelle du modèle de données général. Bien que ces sous-schéma puissent être sauvegardés dans des fichiers séparés, ils ne doivent pas être confondus avec les sous-fichiers d'autres langage objet (comme par exemple les fichiers d'en-tête du C++, qui n'obéissent d'après la norme à aucune contrainte fonctionnelle). L'exemple ci-cessous (voir figure B.2.1) est un extrait du schéma

correspondant à la description de types d'organisation dans l'AP239. Chaque schéma est encadré par les balises SCHEMA et END_SCHEMA. Le nom du schéma suit immédiatement le mot-clé SCHEMA. L'interfaçage entre différents schémas se fait à l'aide des mots-clés USE FROM ou REFERENCE FROM.

```
SCHEMA Organization_type_arm;
USE FROM Person_organization_arm;     -- ISO/TS 10303-1011

[...]

END_SCHEMA;  -- Organization_type_arm
```

FIGURE B.2.1.: Définition d'un Schéma EXPRESS

B.3. Entités

Un schéma EXPRESS est composé d'un ensemble d'*entités*. L'entité a pour fonction de modéliser les éléments existant dans le domaine sémantique considéré (les entités sont utilisées pour modéliser la notion de produit, d'organisation, de composant etc.). L'entité est désignée par son *nom*. Comme tout langage objet :

- une entité est équivalente à la classe d'un langage à objets. EXPRESS ne permet cependant pas de définir des méthodes sur ces classes ;
- chaque entité peut avoir un ou plusieurs *attributs* (propriété représentant une caractéristique de l'entité) ;
- le langage EXPRESS autorise différents types d'*héritage* pour les entités : héritage simple, multiple ou répété.

L'exemple de la figure B.3.1, extrait de l'AP239, représente la description de l'entité `Decision_point` du schéma `Task_Specification_arm` (permettant de représenter une tâche) :

- la définition de l'entité est balisée par les mots-clés ENTITY et END_ENTITY ;
- `Decision_point` est le nom de l'entité,
- le mot-clé SUBTYPE OF indique que l'entité `Decision_point` est une *spécialisation* de l'entité `Structured_task_element` (on dit aussi qu'elle *dérive* de l'entité `Structured_task_element`). Par conséquent, elle *hérite* de l'ensemble des attributs de cette super-classe,
- `condition` est un attribut de l'entité `Decision_point`. Il prend pour valeur une instance de l'entité `Condition` et il est obligatoire (i.e. l'instance de cette entité sera considérée comme non valide si cet attribut n'est pas précisé),
- `true_case_element`, `false_case_element` et `unknown_case_element` sont des attributs optionnels (i.e. l'instance sera considérée comme valide même s'ils ne sont pas spécifiés) prenant comme valeur une instance de l'entité `Task_element`.

```
ENTITY Decision_point
  SUBTYPE OF (Structured_task_element);
   condition : Condition;
   true_case_element : OPTIONAL Task_element;
   false_case_element : OPTIONAL Task_element;
   unknown_case_element : OPTIONAL Task_element;
END_ENTITY;
```

FIGURE B.3.1.: Définition de l'entité `Decision_point`

Remarque : le mot-clé ABSTRACT permet de définir des classes abstraites, dont l'instanciation est interdite et qui ont pour unique fonction d'être spécialisées.

B.4. Typage des attributs

Dans l'exemple précédent, les attributs de l'entité sont des instances d'autres entités. Les attributs doivent cependant permettre de décrire des grandeurs mesurables, des éléments conditionnels, des champs de texte etc. A cet effet, EXPRESS fournit un ensemble de types prédéfinis qui peuvent être :

- simples : BINARY, BOOLEAN (valeur TRUE ou FALSE), LOGICAL (les valeurs aurotisées étant TRUE, UNKNOWN ou FALSE), REAL (nombre réel de taille et de précision quelconques), INTEGER (nombre entier de taille quelconque, souvent limité en pratique à 32 bits dans les implémentations), STRING (chaîne de caractères de longueur quelconque acceptant tous les caractères Unicode[ISO/CEI-10646, 2003] ;
- des collections ou aggrégats : LIST (ensemble ordonnée), SET (ensemble non ordonnée), BAG (ensemble avec répétition), ARRAY (tableau à une dimension) ;
- des énumérations : ENUMERATION ;
- des sélections : SELECT (élément à choisir parmi un ensemble d'éléments).

Par ailleurs, le mot-clé TYPE permet de définir des types personnalisés. Dans l'exemple ci-dessous, deux éléments sont définis :

- le type `message_definer_select` : il définit une sélection d'une parmi deux instances possibles de type `Organization` ou `Person_in_organisation` ;
- l'entité `Message` : les attributs `id` et `message_type` sont des chaînes de caractère. L'attribut `defined_by` est une instance du type `message_definer_select` définie au préalable.

```
TYPE message_definer_select = SELECT
  (Organization, Person_in_organization);
END_TYPE; (* declared in: Message_arm *)

ENTITY Message;
  id : STRING;
  message_type : STRING;
  contains : SET OF Content_item;
  defined_by : OPTIONAL SET OF message_definer_select;
  purpose : OPTIONAL STRING;
END_ENTITY; (* declared in: Message_arm *)
```

FIGURE B.4.1.: Types prédéfinis

B.5. Contraintes

Une des spécificités du langage EXPRESS, par rapport aux autres langages à objets, est qu'il permet d'associer aux valeurs des attributs ou à leur type des *contraintes d'intégrité*. Cette fonctionnalité du langage est d'une grande importance dans la modélisation des données STEP, puique ces contraintes d'intégrité permettent de modéliser des *règles métier*. Les contraintes permettent donc de représenter la sémantique d'une donnée d'une manière beaucoup plus explicite qu'avec UML. Ces contraintes peuvent être :
- *locales* : elles s'appliquent à chaque instance d'une même entité,
- *globales* : elles concernent toutes les instances d'une même entité.

La contrainte assertionnelle WHERE RULE représente un prédicat devant avoir la valeur *vraie*. Nous avons fait figuré (voir figure B.5.1) la définition de l'entité `Local_time` ainsi que celle du type `hour_in_day`. L'entité `Local_time` permet de représenter l'heure locale à partir d'entiers représentant les heures, les minutes, les secondes, ainsi que le décalage par rapport à l'heure GMT. Afin de se prémunir contre d'éventuelles erreurs dans la représentation de l'heure, l'attribut `hour_component` est contraint à être un entier compris dans l'intervalle $[0, 24[$. Dans la définition du type `hour_in_day`, c'est le mot-clé WHERE qui spécifie une contrainte locale, WR1 étant l'identifiant de cette contrainte. On peut spécifier plusieurs contraintes, les suivantes portant alors les identifiants WR2, WR3 etc. Si la contrainte est respectée (*i.e.* $hour_{in_{day}} \in [0, 24[$), la contrainte WR1 est évaluée comme vraie, et l'entité correspondante considérée comme valide. Le cas échéant, la règle est évaluée à Faux et la contrainte est violée. L'entité correspondante n'est pas validée. Cette contrainte est bien *locale* : c'est à l'instance spécifique que s'applique la contrainte, et c'est donc une entité parmi une multitude qui peut être considérée comme non valide, les autres étant valides. Il est à noter par ailleurs que le fait de spécifier une contrainte ne garantit pas que la population d'instances satisfait à ces contraintes. Il faudra vérifier *a posteriori*, par un diagnostic à l'aide d'un outil approprié, que l'ensemble des contraintes est bien respecté.

```
TYPE hour_in_day = INTEGER;
WHERE
  WR1 : {0 <= SELF < 24};
END_TYPE; (* declared in: Date_time_arm *)

ENTITY Local_time;
  hour_component : hour_in_day;
  minute_component : OPTIONAL minute_in_hour;
  second_component :  OPTIONAL second_in_minute;
  zone : Time_offset;
END_ENTITY; (* declared in: Date_time_arm *)
```

FIGURE B.5.1.: Contrainte locale WHERE RULE

Le mot-clé UNIQUE définit une contrainte globale d'*unicité* qui permet de spécifier que le ou les attributs concernés doivent avoir une valeur unique sur l'ensemble des instances de l'entité concernée. Par exemple (voir figure B.5.2), pour la définiton de l'entité `Language`, l'attribut `language_code` est spécifié comme devant être unique (mot-clé UNIQUE, référence de la contrainte UR1) pour chaque instance de cette entité. Par conséquent, la contrainte UR1 sera considérée comme violée si deux instances au moins de l'entité `Language` portent la même valeur pour l'attribut `language_code`. Dans le cadre de la modélisation des produits, la contrainte UNIQUE sera utilisée pour l'attribut définissant l'identifiant de ce produit (par exemple un numéro de série). La vérification de cette contrainte globale, nécessitant de parcourir l'ensemble des instances, peut être coûteuse en terme d'implémentation (mémoire, temps CPU) si plusieurs centaines de milliers d'instances ont été crées.

```
ENTITY Language;
  language_code : STRING;
  country_code :  OPTIONAL STRING;
UNIQUE
  UR1 : language_code;
END_ENTITY; (* declared in: Multi_linguism_arm *)
```

FIGURE B.5.2.: Contrainte globale d'unicité pour un attribut

Le mot-clé INVERSE définit une contrainte de cardinalité inverse. La cardinalité inverse permet de contraindre le nombre d'entités d'un certain type qui référencent une entité donnée dans un certain rôle. Cette cardinalité s'exprime soit sous la forme $SET[Borne_{inf}, Borne_{sup}]$, soit en exprimant son domaine par une entité, la cardinalité étant alors exactement de 1. Dans l'exemple de la figure B.5.3, l'entité `Descriptive_document_property` est une sous-classe de `String_representation_item`, elle même dérivée de l'entité abstraite `Representation_item`. Chaque instance de l'entité portera donc deux

attributs : `name`, `string_value`. La contrainte de cardinalité inverse `valued_characteristic` stipule que la liste associée à l'attribut `item` de l'instance `Document_property_representation` (non-représentée sur la figure B.5.3) porte une et une seule référence à une instance de `Descriptive_document_property`.

```
ENTITY Representation_item ABSTRACT SUPERTYPE;
  name : STRING;
END_ENTITY; (* declared in: Foundation_representation_arm *)

ENTITY String_representation_item SUBTYPE OF (Representation_item);
  string_value : STRING;
END_ENTITY; (* declared in: Foundation_representation_arm *)

ENTITY Descriptive_document_property SUBTYPE OF
       (String_representation_item);
INVERSE
 valued_characteristic : SET [1:1] OF
                         Document_property_representation FOR items;
END_ENTITY; (* declared in: Document_properties_arm *)
```

FIGURE B.5.3.: Contrainte de cardinalité inverse

Il est à remarquer que, les bornes inférieures et supérieures étant égales à 1, on aurait aussi pu exprimer cette cardinalité inverse par la forme plus simple :
`valued_characteristic : Document_property_representation FOR items;`

Enfin, le dernier type de contrainte globale est définie par le mot-clé RULE. Elles s'expriment sous forme de prédicats s'appliquant à l'ensemble d'une (ou plusieurs) population(s) d'entités dans un schéma donné. La population d'entités auxquelles s'applique la contrainte figurent dans l'en-tête de la règle. Dans le corps de la règle, ces noms d'entités représentent l'ensemble des instances correspondantes. Dans l'exemple de la figure B.5.4, la règle globale `part_version_constraint` s'appliquera à toutes les instances de l'entité `Product_version`. Le détail de la contrainte est exprimé par une assertion WHERE rule (WR1).

```
RULE part_version_constraint FOR (Product_version);
WHERE
  WR1 : (* snippet *)
END_RULE; (* declared in: Part_and_version_identification_arm *)
```

FIGURE B.5.4.: Règle globale

B.6. EXPRESS-G

La représentation textuelle des schémas EXPRESS est essentielle pour le traitement automatique des modèles. Elle est en revanche difficilement lisible. Un formalisme graphique, EXPRESS-G, a donc été défini pour donner une vue synthétique des problèmes à modéliser. Ce symbolisme de représentation n'est que partiel et ne permet pas d'exprimer les contraintes d'intégrité. La nature des attributs ainsi que leur type peuvent être représentés. La lecture des diagrammes EXPRESS-G doit être faite à la lumière de la syntaxe de la figure B.6.1.

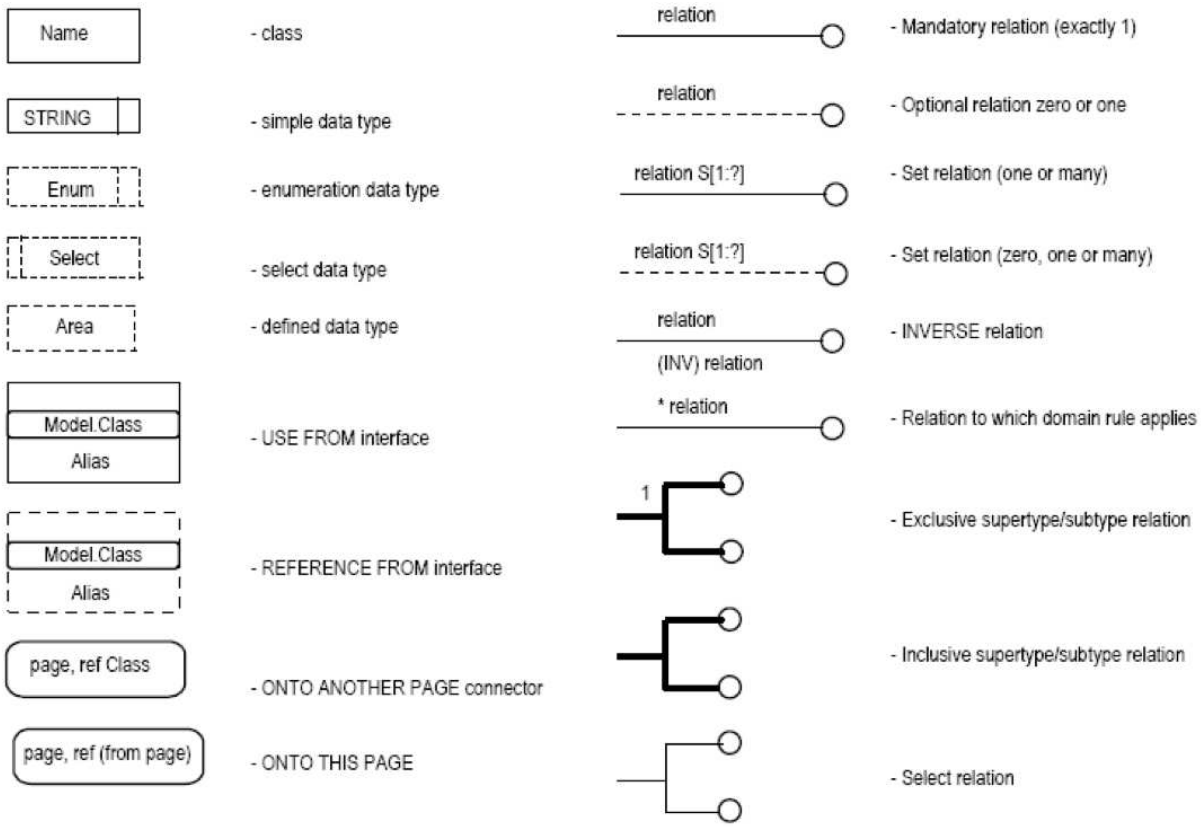


FIGURE B.6.1.: Syntaxe EXPRESS-G

La syntaxe de représentation graphique EXPRESS-G ne permet pas de faire apparaître les contraintes, qu'elles soient locales ou globales. Ainsi, la représentation EXPRESS-G d'un schéma EXPRESS agit comme un filtre qui appauvrit la sémantique du schéma : cette représentation permet de faire apparaître l'architecture globale d'un schéma (les relations entre entités - agrégation, spécialisation), mais elle doit être accompagnée de la lecture du schéma textuel. La figure B.6.2 représente graphiquement l'entité `hour_in_day` [ISO-10303-239, 2005]. Dans la norme STEP, les représentations graphiques sont données pour le schéma complet, et non entité par entité.

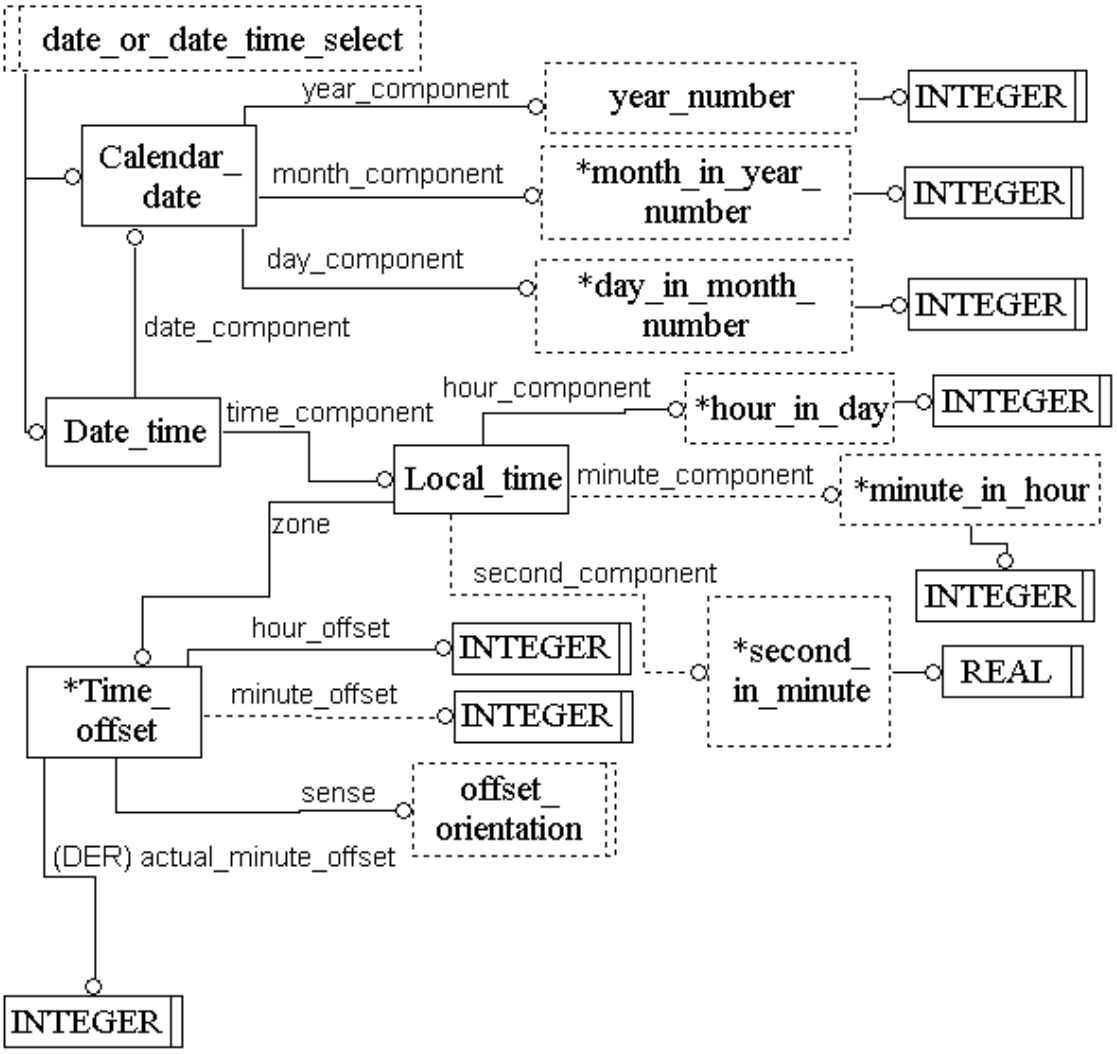


FIGURE B.6.2.: Représentation EXPRESS-G de l'entité `hour_in_day`

C. La bibliothèque de développement pySDAI

C.1. Introduction

pySDAI [1] est un composant logiciel destiné à la manipulation, *via* des scripts Python, des données représentées par un modèle au format EXPRESS. pySDAI s'appuie sur le binding Java de SDAI (cf 4.3.1), intitulé JSDAI [2], et propose une interface de programmation basée sur une architecture à double couche (*double-layer API*). Parmi les fonctionnalités couvertes, on peut citer :

- accès aux interfaces de programmation définies par la norme SDAI [ISO-10303-22, 1998] ;
- instanciation d'entités EXPRESS conformément à un schéma (cf B.2) ;
- écriture d'une population d'instances dans des fichiers *clear text encoding* [ISO-10303-21, 2002] ou XML [ISO-10303-28, 2007] ;
- validation des contraintes locales et globales (cf B.5) ;
- wrapper Jython permettant de simplifier l'accès à SDAI ;
- sérialization des entités EXPRESS pour un transfert de Jython vers CPython ;
- pySDAI intègre une version pré-compilée et prête à l'emploi des schémas EXPRESS/STEP les plus courants dans le domaine du PLM : PDMSchema, AP203, AP239.

pySDAI a été conçue pour faciliter le prototypage d'applications nécessitant la manipulation de données EXPRESS/STEP. L'objectif de cette annexe est d'introduire à la mise en œuvre de cette bibliothèque.

C.2. Contacts

Pour toute question relative à pySDAI, contacter Thomas Paviot par e-mail à l'adresse tpaviot@gmail.com.

1. http ://sf.net/projects/pysdai
2. http ://www.jsdai.net

C.3. Licence d'utilisation

La bibliothèque pySDAI est distribuée selon les termes de la licence *GNU Affero General Public License*[3] dans sa version 3 (AGPLv3). AGPL est une licence approuvée par l'Open Source Initiative (OSI)[4]. Cette license place pySDAI dans la catégorie des logiciels libres et garantit par conséquent les quatres libertés fondamentales du logiciel libre énoncées par la Free Software Foundation[5] :

- la liberté d'exécuter le programme pour tous les usages,
- la liberté d'étudier le fonctionnement du programme,
- la liberté de redistribuer des copies (ce qui comprend la liberté de vendre des copies),
- la liberté d'améliorer le programme et de publier ses améliorations.

La license AGPL présente une particularité importante à noter : c'est une licence dite *virale* en ce sens que tous les programmes utilisant pySDAI devront être également distribués selon les termes de cette licence (JSDAI étant distribué sous AGPLv3, pySDAI l'est également). Il n'est donc pas possible d'utiliser pySDAI/JSDAI pour développer des programmes propriétaires, à moins d'avoir demandé et obtenu l'autorisation expresse du titulaire des droits patrimoniaux de l'œuvre (pour le cas de JSDAI, cette autorisation se monnaye, et on parle alors de double-licence). Le téléchargement et l'utilisation de pySDAI implique l'acceptation par le licencié de tous les termes du contrat de licence. On se reportera pour plus d'information au texte complet de la license AGPLv3[6].

C.4. Qualité

La démarche qualité qui sous-tend le développement de pySDAI s'appuie sur les éléments suivants :

- conformité à la convention de développement *PEP8-Style Guide for Python Code*[7],
- documentation fournie du code source (en anglais),
- série de tests unitaires permettant de garantir les fonctionnalités,
- hébergement du projet sur une plateforme communautaire permettant le suivi des dysfonctionnements (*bug tracker*) et des modifications du code source (*subversion repository*).

3. http ://www.gnu.org/licenses/agpl.html
4. http ://www.opensource.org/
5. http ://www.fsf.org
6. http ://www.gnu.org/licenses/agpl-3.0.txt
7. http ://www.python.org/dev/peps/pep-0008/

C.5. Le schéma *Family*

Le schéma EXPRESS utilisé comme fil rouge de ce document est connu sous le nom de *Family* (cf. Figure C.5.1). Il est généralement utilisé comme support à l'introduction au language de modélisation EXPRESS. Il permet de modéliser une famille composée d'hommes et de femmes liés par des relations de type parent/enfant. Ce schéma définit trois entités :

1. l'entité ***Person*** est une entité abstraite, i.e. elle ne peut être instanciée et a vocation à être spécialisée. Cette entité possède trois attributs : l'attribut *name,* obligatoire, de type chaîne de caractères (STRING), ainsi que deux attributs optionnels (OPTIONAL) *mother* et *father*, instances des entités ***Female*** et ***Male***. Le caractère optionnel des attributs *mother* et *father* implique que l'instanciation des sous-classes de ***Person*** ne nécessite pas que ces attributs soient définis pour que l'instance soit considérée vomme valide.
2. l'entité ***Male*** est une spécialisation de l'entité ***Person***.
3. l'entité ***Female*** est une spécialisation de l'entité ***Person***.

Dans la suite de cette annexe, nous représenterons par une population d'instances la famille suivante : Thomas est le fils de Claude et Michèle. Claude est le fils de Marcel et Mariette, Michèle est la fille de Jean et Jeannette. Ces instances seront exportées dans un fichier au format Part 21.

```
SCHEMA family;

ENTITY Person
 ABSTRACT SUPERTYPE OF(ONEOF(Male,Female));
   name:STRING;
   mother:OPTIONAL Female;
   father:OPTIONAL Male;
END_ENTITY;

ENTITY Female
 SUBTYPE_OF(Person);
END_ENTITY;

ENTITY Male
 SUBTYPE OF(PERSON);
END_ENTITY;

END_SHEMA;
```

FIGURE C.5.1.: Le schéma EXPRESS-M *Family*

C.6. Pré-requis

Cette étape vise à mettre en place l'environnement de travail nécessaire pour mettre en œuvre pySDAI. L'installation et la configuration de cet environnement se fait en trois points :

1. Télécharger et installer Eclipse 3.5 (Galiléo) : Eclipse[8] est un outil de développement logiciel. Eclipse se base sur une architecture modulaire permettant d'installer et utiliser une série de modules (*plugins*) mis à disposition par des éditeurs. L'installation/désinstallation/mise à jour de ces modules se fait de manière transparente via les fonctionnalités de mise à jour d'Eclipse. La page de téléchargement d'Eclipse est disponible à l'adresse http ://www.eclipse.org/downloads/. Il est recommandé d'installer la version *Eclipse Classic* (en version 3.5.2 ou supérieure) ;
2. Télécharger et installer Jython 2.5.1 : Jython[9] est une implémentation du language de programmation Python en java. Jython permet ainsi d'accéder à l'ensemble des API java à l'aide de scripts Python. La version nécessaire pour exécuter pySDAI est la version 2.5.1. Elle est disponible en téléchargement à l'adresse : http ://www.jython.org/downloads.html ;
3. Télécharger et installer pySDAI.

Le code source du pySDAI est disponible dans un dépôt *subversion*[10] *(SVN)* hébergé par le site communautaire sourceforge.net[11]. SVN est un outil de gestion de code source, qui permet de travailler de manière collaborative sur un projet de développement tout en conservant la traçabilité de l'historique des modifications apportées[12]. Pour récupérer sur le disque local une copie du dépôt SVN, il faut, depuis une console, taper la commande suivante :

```
svn co https://pysdai.svn.sourceforge.net/svnroot/pysdai/trunk pysdai
```

C.7. Compilation du schéma Family et import dans Jython

Le schéma EXPRESS à traiter doit tout d'abord être compilé dans Eclipse (voir figure C.7.1).

8. http ://www.eclipse.org/
9. http ://www.jython.org/
10. *http ://subversion.tigris.org/*
11. http ://sourceforge.net
12. svn book (http ://svnbook.red-bean.com/)

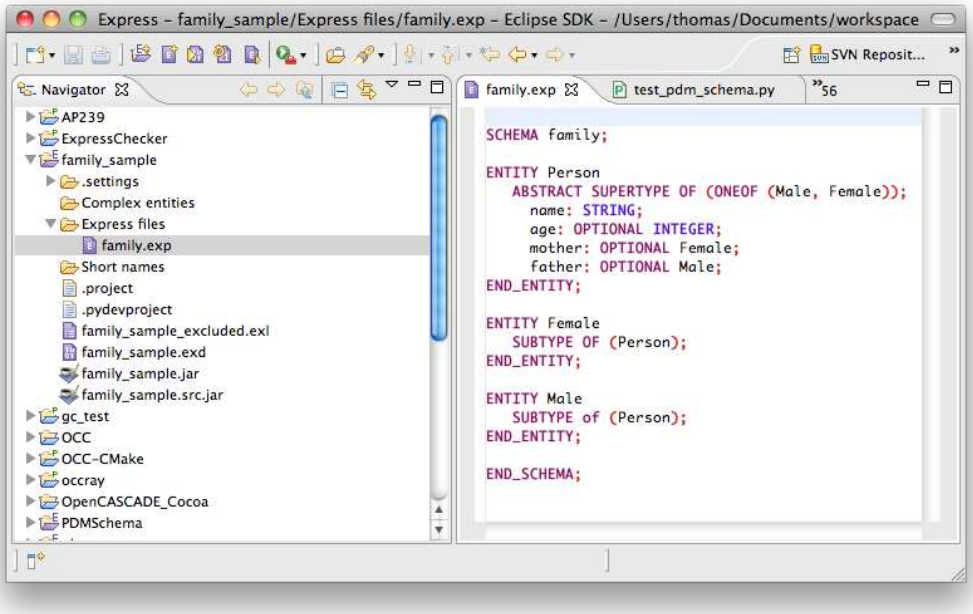


FIGURE C.7.1.: Le schema EXPRESS family dans l'éditeur intégré à Eclipse

C.8. Manipulation de schémas EXPRESS/STEP

pySDAI comprend, dans la distribution de base, quelques schéma EXPRESS prêts-à-l'emploi ou pré-compilés. La compilation d'un schéma EXPRESS sera vue dans la section suivante. La création d'une population d'instances du PDM Schéma est détaillée ci-après étape par étape. Naturellement, le même processus sera utilisé pour travailler avec d'autres schémas.

C.8.1. Initialisation

Les deux lignes suivantes chargent les modules nécessaires :

- au travail en mode *early binding* (ligne 1) : cette ligne sera commune à tous les scripts *early binding* ;
- à la manipulation des entités définies dans le PDM Schema (ligne 2).

```
1 from pySDAI.Core.EarlyBinding.EBWrapper import *
2 from pySDAI.schemas.PDMSchema import *
```

C.8.2. Mise en place de l'environnement SDAI

Afin de pouvoir créer des instances, il faut dans l'ordre :

- ouvrir une *session* SDAI (ligne 4) (la ligne 3 permet seulement l'affichage à l'écran des messages SDAI) ;
- démarrer une *transaction* en mode lecture / écriture (ligne 5) ;
- creer un *dépôt* (*repository*) dans lequel seront stockées les instances (lignes 6 et 7).

```
3 SdaiSession.setLogWriter(PrintWriter(sys.stdout,True))
4 session = SdaiSession.openSession()
5 transaction = session.startTransactionReadWriteAccess()
6 repo = session.createRepository("",None)
7 repo.openRepository()
```

C.8.3. Création d'un modèle SDAI

Dans le contexte SDAI, les entités sont créées dans un conteneur appelé *modèle.* Le modèle est instancié à partir de deux arguments (ligne 8) :

- le nom du modèle (une chaîne de caractères) ;
- le schéma à utiliser. Le type à passer en argument est toujours de la forme `SNom_du_schema` (attention à bien respecter la casse : S majuscule, suivi de la première lettre du nom du schéma en majuscule, suivi des autres lettres en minuscule). Ainsi, le schéma utilisé étant ici `pdm_schema` (tel qu'il est défini dans le fichier EXPRESS associé), l'argument passé sera `SPdm_schema`.

Enfin, il faut veiller à demander le début d'un accès en lecture/écriture à ce modèle (ligne 9). Il est possible de créer plusieurs modèles, portant des noms différents ou non (l'identifiant du modèle n'est pas son nom), sachant qu'il y aura un fichier par modèle (fichier ASCII ou XML).

```
8 model = repo.createSdaiModel("mon_modele", SPdm_schema)
9 model.startReadWriteAccess()
```

Exemple de création de modèle Le script de la figure C.8.1 correspond à l'implémentation de ce modèle.

```
if __name__=='__main__':
    SdaiSession.setLogWriter(PrintWriter(sys.stdout,True))
    ## first open a session
    session = SdaiSession.openSession()
    transaction = session.startTransactionReadWriteAccess()
    # Create repository
    repo = session.createRepository("",None)
    repo.openRepository()
    descriptions = repo.getDescription()
    descriptions.addByIndex(1,"Family sample - A simple getting started
        example")
    # Authors
    authors = repo.getAuthor()
    authors.addByIndex(1, "Thomas Paviot")
    # Define organization
    organizations = repo.getOrganization()
    organizations.addByIndex(1,"pySDAI")
    repo.setOriginatingSystem(session.getSdaiImplementation().getName()+"
        "+session.getSdaiImplementation().getLevel());
    repo.setAuthorization("Thomas Paviot")
    # early binding
    model = repo.createSdaiModel("Model1", SFamily)
    model.startReadWriteAccess()
    # Create Male instances
    thomas = model.createEntityInstance(CMale)
    thomas.setName(None,'Thomas')
    claude = model.createEntityInstance(CMale)
    claude.setName(None,'Claude')
    jean = model.createEntityInstance(CMale)
    jean.setName(None,'Jean')
    marcel = model.createEntityInstance(CMale)
    marcel.setName(None,'Marcel')
    # Create Female instances
    michele = model.createEntityInstance(CFemale)
    michele.setName(None,'Michele')
    jeannette = model.createEntityInstance(CFemale)
    jeannette.setName(None,'Jeannette')
    mariette = model.createEntityInstance(CFemale)
    mariette.setName(None,'Mariette')
    # Set child/father relations
    thomas.setMother(None,michele)
    thomas.setFather(None,claude)
    michele.setMother(None,jeannette)
    michele.setFather(None,jean)
    claude.setMother(None,mariette)
    claude.setFather(None,marcel)
    # Export model to a STEP Part28 file
    repo.exportXml(None,'my_family.p28')
    transaction.endTransactionAccessAbort()
    repo.closeRepository()
    repo.deleteRepository()
    session.closeSession()
```

FIGURE C.8.1.: Script Python pour l'instanciation du modèle de Famille

Fichier généré Le fichier résultant est reproduit sur la figure C.8.2.

```
ISO-10303-21;
HEADER;
/* Generated by software containing
* JSDAI (TM) from LKSoft (www.lksoft.com, www.jsdai.net)
* JSDAI Runtime Version 4.0.0 (Build 270, 2009-07-22T20:05:26)
*/
FILE_DESCRIPTION(
/* description */ ('Family sample - A simple getting started example'),
/* implementation_level */ '2;1');
FILE_NAME(
/* name */ ' ',
/* time_stamp */ '2010-04-05T12:04:02',
/* author */ ('Thomas Paviot'),
/* organization */ ('pySDAI'),
/* preprocessor_version */ ' ',
/* originating_system */ 'JSDAI MULTIPLE Version 4.0.0 (Build 270,
    2009-07-22T20:05:26)',
/* authorization */ 'Thomas Paviot');
FILE_SCHEMA(('FAMILY'));
ENDSEC;
DATA;
#1=MALE('Thomas',#5,#2);
#2=MALE('Claude',#7,#4);
#3=MALE('Jean',$,$);
#4=MALE('Marcel',$,$);
#5=FEMALE('Michele',#6,#3);
#6=FEMALE('Jeannette',$,$);
#7=FEMALE('Mariette',$,$);
ENDSEC;
END-ISO-10303-21;
```

FIGURE C.8.2.: Fichier ISO-10303-Part 21

D. Mentions légales

L'appendice C de ce mémoire (texte et figures) sont placées sous licence Creative Common [1] BY-NC-SA [2].

D'après le texte de cette licence, vous êtes libre :
- de reproduire, distribuer et communiquer cette création au public,
- de modifier cette création.

Selon les conditions suivantes :
- Paternité (BY) : vous devez citer le nom de l'auteur original de la manière indiquée par l'auteur de l'œuvre ou le titulaire des droits qui vous confère cette autorisation (mais pas d'une manière qui suggérerait qu'ils vous soutiennent ou approuvent votre utilisation de l'œuvre),
- Pas d'Utilisation Commerciale (Non Commercial) : vous n'avez pas le droit d'utiliser cette création à des fins commerciales,
- Partage des Conditions Initiales à l'Identique (Share Alike) : si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.

Le texte complet de la licence CC-BY-NC-SA est disponible en langue française à l'adresse http ://creativecommons.org/licenses/by-nc-sa/2.0/fr/legalcode.

Les extraits de code source publiés en annexe sont placés sous licence libre AGPLv3 ou GPLv3 (voir l'entête des fichiers).

Catia® et 3DXML® sont des marques déposées de Dassault Systèmes S.A. Pro/Engineer® et Windchill® sont des marques déposées de Parametric Technology Corporation. Inventor® est une marque déposée d'Autodesk Inc. JSDAI® est une marque déposée de LKSoft Gmbh. OpenERP® est une marque déposée de Tinty Sprl. R/3® est une marque déposée de SAP AG. Python® est une marque déposée de la Python Software Foundation.

1. http ://fr.creativecommons.org/
2. http ://creativecommons.org/licenses/by-nc-nd/2.0/fr/

Bibliographie

A

[Adreit and Mauran, 1995] Adreit, F. and Mauran, P. (1995). *L'interaction en conception in Le communicationnel pour concevoir (Caelle, J. et Zreik).* Paris :Europia.

[Agard, 2004] Agard, B. (2004). Modélisation des familles de produits : État de l'art. *Mécanique & Industries*, 5 :275–288.

[Alemanni et al., 2008] Alemanni, M., Alessia, G., Tornincasa, S., and Vezzetti, E. (2008). Key performance indicators for PLM benefits evaluation : The Alcatel Alenia Space case study. *Computers in Industry*, 59(8) :833–841.

[Altshuller et al., 1999] Altshuller, G., Shulyak, L., and Rodman, S. (1999). *The Innovation Algorithm : TRIZ, systematic innovation and technical creativity.* Technical Innovation Ctr.

[Amilhastre et al., 2002] Amilhastre, J., Fargier, H., and Marquis, P. (2002). Consistency restoration and explanations in dynamic CSPs–Application to configuration. *Artificial Intelligence*, 135(1-2) :199–234.

[Asiedu and Gu, 1998] Asiedu, Y. and Gu, P. (1998). Product life cycle cost analysis : state of the art review. *International journal of production research*, 36(4) :883—908.

[Audrain, 2008] Audrain, P. (2008). Gestion des actifs industriels : intégration d'un logiciel product data management avec un logiciel enterprise asset management. *Ateliers MICADO, 21 octobre 2008, ENS Cachan.*

B

[Baïna, 2006] Baïna, S. (2006). Interopérabilité dirigée par les modèles : Une Approche Orientée Produit pour l'interopérabilité des systèmes d'entreprise. *Thèse de doctorat de l'Université Henri Poincaré, Nancy I.*

[Ball et al., 2008] Ball, A., Ding, L., and Patel, M. (2008). An approach to accessing product data across system and software revisions. *Advanced Engineering Informatics*, 22(2) :222–235.

[Barkmeyer et al., 1999] Barkmeyer, E., Denno, P., Feng, S., Jones, A., and Wallace, E. (1999). NIST Response to MES Request for Information. *NIST Response to RFI-3 : MES Models*, pages 2–4.

[Barton et al., 2001] Barton, J., Love, D., and Taylor, G. (2001). Design determines 70 percents of cost ? A review of implications for design evaluation. *Journal of Engineering Design*, 12(1) :47—58.

[Barzi, 2007] Barzi, R. (2007). Le concept de l'agilité à l'épreuve de la pme. In *Actes de la 16ème Conféence Internationale de Management Stratégique*.

[Bateman et al., 2007] Bateman, S., Brooks, C., McCalla, G., and Brusilovsky, P. (2007). Applying collaborative tagging to e-learning. *Proceedings of ACM WWW*, 7.

[Bellacicco et al., 2007] Bellacicco, A., Sellakh, R., and Rivière, A. (2007). Méthode de conception collaborative appliquée à la maquette 3d. *Conception et Production Intégrées, 22-24 octobre 2007, Rabat-Maroc*.

[Bénaben, 2008] Bénaben, F. (2008). Du processus collaboratif au médiateur des systèmes d'information des partenaires. In *Actes de la 7ème Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - Paris, France*, volume 2, pages 837–846.

[Bénaben et al., 2006] Bénaben, F., Touzi, J., Rajsiri, V., and Pingaud, H. (2006). Collaborative Information System Design. *AIM Conference*.

[Bénaben et al., 2008] Bénaben, F., Touzi, J., Rajsiri, V., Truptil, S., Lorré, J., and Pingaud, H. (2008). Mediation Information System Design in a Collaborative SOA Context through a MDD Approach. *MDISIS Workshop, CAISE 08 conference, Montpellier*.

[Berkooz, 2007] Berkooz, G. (2007). Viewpoint : Bertil Turesson on PLM. *International Journal of Product Lifecycle Management*, 2(1) :104—107.

[Bernard et al., 2009] Bernard, A., Laroche, F., and Cunha, C. D. (2009). Models and methods for knowledge formalisation in a PLM context. *3rd International Congress Design and Modelling of Mechanical Systems CMSM'2009*.

[Bidan, 2004] Bidan, M. (2004). Fédération et intégration des applications du Système d'information de gestion. *Système d'Information et Management*, 9(2) :5–24.

[Bissay et al., 2008] Bissay, A., Pernelle, P., Lefebvre, A., and Bouras, A. (2008). Approche de capitalisation des connaissances à l'aide d'un système PLM. *7ème Conférence Internationale de MOdélisation et SIMulation MOSOM'08, 31/3-2/4 208, Paris-France*.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orhcard, D. (2004). Web Services Architecture. *W3C Working Group Note, 11/2/2004*.

[Borghoff et al., 2006] Borghoff, U., Rödig, P., Scheffczyk, J., and Schmitz, L. (2006). *Long-term preservation of digital documents : principles and practices*. Springer Verlag, Berlin.

[Botta-Genoulaz et al., 2005] Botta-Genoulaz, V., Millet, P., and Grabot, B. (2005). A survey on the recent research literature on ERP systems. *Computers in Industry*, 56(6) :510–522.

[Briggs et al., 2005] Briggs, T., Rando, T., and Daggett, T. A. (2005). Re-Use of Ship Product Model Data for Life-Cycle Support. *Proceedings of the SNAME Maritime Technology Conference & Expo 2005, Houston, TX*, pages 1–12.

C

[Chambolle, 1999] Chambolle, F. (1999). Un modèle produit piloté par les processus d'élaboration : Application au secteur automobile dans l'environnement STEP. *Thèse de doctorat de l'Ecole Centrale Paris*.

[Chan, 1999] Chan, S. (1999). Architecture choices for erp systems. In *Proceedings of the Americas Conference on Information Systems*.

[Chen and Doumeingts, 2003] Chen, D. and Doumeingts, G. (2003). European initiatives to develop interoperability of enterprise applications - basic concepts, framework and roadmap. *Annual Reviews in Control*, (27) :153–162.

[Chen and Vernadat, 2004] Chen, D. and Vernadat, F. (2004). Standards on enterprise integration and engineering–state of the art. *International Journal of Computer Integrated Manufaturing*, 17(3) :235—253.

[Cheung et al., 2010] Cheung, W., Maropoulos, P., and Mattews, P. (2010). Linking design and manufacturing domains via web-based and enterprise integration technologies. *Int. J. Computer Applications in Technology*, 37(3/4) :182—198.

[Cheutet et al., 2009] Cheutet, V., Paviot, T., and Lamouri, S. (2009). Conception et simulation d'une ligne d'assemblage final aéronautique. *11ème Colloque National AIP PRIMECA, 22-24 avril 2009, La Plagne*.

[CIMdata, 2009] CIMdata (2009). PLM Market Growth in 2008, Mid-Year.

[CNSS, 2006] CNSS (2006). NATIONAL INFORMATION ASSURANCE (IA) GLOSSARY.

[Cohen, 2009] Cohen, D. (2009). *La prospérité du vice, Une introduction (inquiète) à l'économie*. éd. Albin Michel, Paris.

[Cox and Blackstone, 2002] Cox, J. and Blackstone, J. (2002). *APICS Dictionnary 10th edition*. Terry College of Business, University of Georgia, Cox and Blackstone.

[Crouhy and Greif, 1991] Crouhy, M. and Greif, M. (1991). *Gérer simplement les flux de production*. Éditions du Moniteur, Paris.

[Crubleau and Richir, 2001] Crubleau, P. and Richir, S. (2001). La prospective technologique et TRIZ, de nouveaux outils pour le concepteur. *XVème Congrès Français de Mécanique, 3-7/9/2001, Nancy*.

D

[Dubois, 1999] Dubois, P. (1999). Ten good practices in scientific programming. *Computing in Science & Enginerring*, 1(1) :7–11.

[Ducellier, 2008] Ducellier, G. (2008). Gestion de règles expertes en ingénierie collaborative : application aux plateformes PLM. *Thèse de Doctorat de l'Université de Technologie de Troyes*.

[Ducellier et al., 2007] Ducellier, G., Eynard, B., and Caillaud, E. (2007). Infrastructure PLM pour la capitalisation et la réutilisation de données en conception mécanique. *18ème Congrès Français de Mécanique, Grenoble, 27-31 août 2007*.

E

[EIF, 2004] EIF (2004). European Interoperability Framework - White Paper. pages 1–40.

[Ethiraj and Levinthal, 2004] Ethiraj, S. and Levinthal, D. (2004). Modularity and innovation in complex systems. *Management Science*, pages 159–173.

[Eynard et al., 2004] Eynard, B., Gallet, T., Nowak, P., and Roucoules, L. (2004). UML based specifications of PDM product structure and workflow. *Computers in Industry*, 55(3) :301–316.

F

[Feeney, 2002] Feeney, A. (2002). The STEP modular architecture. *Journal of Computing and Information Science in Engineering*, 2 :132.

[Firsch and Stirk, 2007] Firsch, H. and Stirk, C. (2007). *ISO STEP-AP233 Transition Development to Enablement, http ://www.ap233.org/ap233-public-information/presentations/ap20v3.ppt*. OMG.

[Foster and Kesselman, 2004] Foster, I. and Kesselman, C. (2004). *The grid : blueprint for a new computing infrastructure*. Morgan Kaufman.

[Fourastié, 1966] Fourastié, J. (1966). *Idées majeures.(Pour un humanisme de la société scientifique)*. Gonthier.

[Fowler, 1995] Fowler, J. (1995). *STEP for Data Management, Exchange and Sharing*. Technology Appraisals (ISBN 1-871802-36-9).

G

[Galeta et al., 2006] Galeta, T., Kljajin, M., and Karakasic, M. (2006). Product model suited for the ERP system. *9th International Design Conference-Design 2006*.

[Gallagher et al., 2002] Gallagher, M., O'Connor, A., and Phelps, T. (2002). *Economic Impact Assessment of the International STEP Standard in Transportation Equipment Industries*. National Institute of Standards and Technology.

[Gallet et al., 2007] Gallet, T., Guyot, E., Ducellier, G., and Eynard, B. (2007). Interopérabilité des logiciels CAO et calcul avec un SGDT dans une chaîne de conception aéronautique. *5ème Conférence Internationale Conception et Production Intégrées CPI2007, 22-24 octobre 2007, Rabat - Maroc*.

[Galliers et al., 1997] Galliers, R., Mingers, J., and Jackson, M. (1997). Organization theory and systems thinking : the benefits of partnership. *Organization*, 4(2) :269–280.

[Georgiev et al., 2007] Georgiev, I., Ovtcharova, J., and Georgiev, I. (2007). Modelling web services for PLM distributed system. *International Journal of Product Lifecycle Management*, 2(1) :30–49.

[Geraci et al., 1991] Geraci, A., Katki, F., McMonegal, L., B., M., Lane, J., P., W., Radatz, J., Yee, M., H., P., and Springsteel, F. (1991). *IEEE Standard Computer Dictionary : Compilation of IEEE Standard Computer Glossaries*. IEEE Press, Piscataway, NJ, USA.

[Giard, 2003] Giard, V. (2003). *Gestion de la production et des flux*. Economica, Collection Gestion, Paris.

[Gillon, 2004] Gillon, B. (2004). *Semantics : a reader*, chapter Ambiguity, indeterminacy, deixis and vagueness., pages 157–187. Oxford University Press.

[Giménez et al., 2008] Giménez, D., Vegetti, M., Leone, H., and Henning, G. (2008). PRoduct ONTOlogy : Defining product-related concepts for logistics planning activities. *Computers in Industry*, 59(2-3) :231–241.

[Godard, 2006] Godard, D. (2006). *L'ambigüité*, pages 157–187. GDR Sémantique & Modélisation.

[Gottschalk, 2000] Gottschalk, K. (2000). Web services architecture overview. *IBM white paper, IBM developer Works , 1er septembre 2000*.

[Grabot et al., 2008] Grabot, B., Mayère, A., and Bazet, I. (2008). *ERP Systems and Organisational Change : A Socio-technical Insight*. Springer Verlag.

[Greif, 1998] Greif, M. (1998). *L'Usine s'affiche. La communication visuelle au service du progrès.* Éditions d'organisation.

[Gruber, 1995] Gruber, T. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43(5) :907–928.

[Grunstein, 2002] Grunstein, M. (2002). De la capitalisation des connaissances au renforcement des compétences dans l'entreprise étendue. *Premier colloque du groupe de travail Gestion des Compétences et des Connaissances en Génie Industriel, Nantes*.

[Guyot et al., 2007] Guyot, E., Ducellier, G., Eynard, B., Girard, P., and Gallet, T. (2007). Product data and digital mock-up exchange based on plm. In *Proceedings of the PLM07 conference*.

H

[Hoffmann, 2008] Hoffmann, P. (2008). Similarité sémantique inter-ontologies basée sur le contexte. *Thèse de doctorat de l'Université Claude Bernard - Lyon 1*.

I

[Iskanius, 2006] Iskanius, R. (2006). *An agile supply chain for a project-oriented steel product network*. Thèse de doctorat de l'Univesrité de Oulu.

[ISO-10303-11, 1994] ISO-10303-11 (1994). *Systèmes d'automatisation industrielle et intégration. Représentation et échange de données de produits. Partie 11 : méthodes de description : manuel de référence du langage EXPRESS.* International Standard Organization.

[ISO-10303-203, 1994] ISO-10303-203 (1994). *Systèmes d'automatisation industrielle et intégration – Représentation et échange de données de produits – Partie 203 : Protocole d'application : Conceptions 3D contrôlées de configuration des pièces mécaniques et des assemblages*. International Standard Organization.

[ISO-10303-203ed2, 2005] ISO-10303-203ed2 (2005). Systèmes d'automatisation industrielle et intégration - représentation et échange de données de produits - partie 203 : protocole d'application : conceptions 3d contrôlées de configuration de pièces mécaniques et des assemblages. *ISO-10303-203ed2*.

[ISO-10303-21, 2002] ISO-10303-21 (2002). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produits - Partie 21 : méthodes de mise en application : encodage en texte clair des fichiers d'échange*. International Standard Organization.

[ISO-10303-22, 1998] ISO-10303-22 (1998). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produit - Partie 22 : méthodes de mise en application : interface normalisée d'accès aux données*. International Standard Organization.

[ISO-10303-23, 2000] ISO-10303-23 (2000). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produits - Partie 23 : méthodes de mise en application : liant de langage C++ à l'interface d'accès aux données normalisées*. International Standard Organization.

[ISO-10303-239, 2005] ISO-10303-239 (2005). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produits - Partie 239 : protocole d'application : cycle de vie du produit et modules d'application*. International Standard Organization.

[ISO-10303-24, 2001] ISO-10303-24 (2001). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produits - Partie 24 : méthode de mise en application : liant de langage C à l'interface d'accès aux données normalisées*. International Standard Organization.

[ISO-10303-28, 2007] ISO-10303-28 (2007). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produits - Partie 28 : méthodes d'implémentation : représentations XML de schémas et de données EXPRESS en utilisant des schémas XML*. International Standard Organization.

[ISO-14258-1998, 1998] ISO-14258-1998 (1998). *Systèmes d'automatisation industrielle - Concepts et règles pour modèles d'entreprise*. International Standard Organization.

[ISO/CEI-10646, 2003] ISO/CEI-10646 (2003). *Technologies de l'information - Jeu universel de caractères codés sur plusieurs octets (JUC)*. International Standard Organization/CEI.

[ISO/TS-10303-27, 2000] ISO/TS-10303-27 (2000). *Systèmes d'automatisation industrielle et intégration - Représentation et échange de données de produits - Partie 27 : méthode de mise en application : liaison de langage de programmation Java à l'interface d'accès aux données normales avec extensions Internet/Intranet*. International Standard Organization/TS.

J

[Jubertie, 2007] Jubertie, S. (2007). Modèles et outils pour le déploiement d'applications de Réalité Virtuelle sur des architectures distribuées hétérognèes. *Thèse de Doctorat de l'Université d'Orléans*, pages 1–144.

[Jun et al., 2007] Jun, H., Kiritsis, D., and Xirouchakis, P. (2007). Research issues on closed-loop PLM. *Computers in Industry*, 58(8-9) :855–868.

K

[Kadiri et al., 2009] Kadiri, S. E., Pernelle, P., Delattre, M., and Bouras, A. (2009). Current situation of PLM systems in SME/SMI : Survey's results and analysis. *6th International Conference on Product Lifecycle Management, Bath - GB*.

[Koike, 2005] Koike, T. (2005). Les interfaces pour l'intégration de la logistique dans les projets de conception. *Thèse de doctorat de l'Institut National Polytechnique de Grenoble*.

[Kosanke, 2005] Kosanke, K. (2005). ISO Standards for Interoperability : a comparison. *Pre-Proceedings of the INTEROP-ESA*, pages 59–67.

[Krauter et al., 2002] Krauter, K., Buyya, R., and Maheswaran, M. (2002). A taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Software Practice and Experience*, 32 :135—164.

[Kumar and Van Hillegersberg, 2000] Kumar, K. and Van Hillegersberg, J. (2000). ERP experiences and evolution. *Communications of the ACM*, 43(4) :22–26.

L

[Lamouri, 2006] Lamouri, S. (2006). *Synchronisation des prises de décisions dans une chaîne logistique : robustesse et stabilité*. Mémoire d'Habilitation à Diriger des Recherches, Ecole Doctorale d'Informatique et Electronique de Paris.

[Le Moigne, 1977] Le Moigne, J. (1977). *Théorie du système général. Théorie de la modélisation*. Presses Universitaires de France.

[Le Moigne, 1990] Le Moigne, J. (1990). *La modélisation des systèmes complexes*. Dunod Paris.

[Lee et al., 2009] Lee, H.-G., Park, N., Jeong, H.-I., and Park, J. (2009). Grid enabled MRP process improvement under distributed database environment. *The Journal of Systems & Software*, 82(7) :1087–1097.

[Lee and Jeong, 2006] Lee, S. and Jeong, Y. (2006). A system integration framework through development of ISO 10303-based product model for steel bridges. *Automation in Construction*, 15(2) :212–228.

[Lee et al., 2008] Lee, S., Ma, Y., Thimm, G., and Verstraeten, J. (2008). Product lifecycle management in aviation maintenance, repair and overhaul. *Computers in Industry*, 59(2-3) :296–303.

[Louis-Sidney et al., 2009] Louis-Sidney, L., Cheutet, V., and Lamouri, S. (2009). Discussion about the establishment of a support process for logistics of design. *3ième Congrès International Conception et Modélisation des Systèmes Mécaniques (CMSM'2009), 16-18 mars 2009, Hammamet - Tunisie.*

M

[Machner and Ungerer, 1998] Machner, B. and Ungerer, M. (1998). Mapping the user requirements from the VDAORG with the STEP PDM Schema. *ProductData Journal*, 5(1) :8–9.

[Marlow et al., 2006] Marlow, C., Naaman, M., Boyd, D., and Davis, M. (2006). Position paper, tagging, taxonomy, flickr, article, toread. *Collaborative web tagging workshop at WWW*, 6.

[Massuthe et al., 2005] Massuthe, P., Reisig, W., and Wolf, K. (2005). An operating guideline approach to the SOA. *nnals of Mathematics, Computing and Teleinformatics*, 1(3) :35—43.

[Mathes, 2004] Mathes, A. (2004). Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication.*

[Messaadia, 2008] Messaadia, M. (2008). Ingénierie système et systèmes de production manufacturière : intégration de l'évolution des exigences dans le PLM. *Thèse de doctorat de l'Université Toulous III - Paul Sabatier.*

[Midler, 1993] Midler, C. (1993). *L'auto qui n'existait pas : management des projets et tansformation de l'entreprise.* InterEditions, Paris.

[Mirdamadi, 2007] Mirdamadi, S. (2007). Pilotage d'un atelier de production en temps réel à base de simulation de flux à événements discrets. *Thèse de doctorat de l'Ecole des Mines d'Albi.*

[Morel et al., 2007] Morel, G., Panetto, H., Mayer, F., and Auzelle, J. (2007). System of enterprise-Systems integration issues : an engineering perspective. *IFAC Conference on Cost Effective Automation in Networked Product Development and Manufacturing, IFAC-CEA'07, Monterrey - Mexico.*

N

[NF-EN-1325-1, 1996] NF-EN-1325-1 (1996). *Vocabulaire du management de la valeur, de l'analyse de la valeur et de l'analyse fonctionnelle - Partie 1 : analyse de la valeur et analyse fonctionnelle.* NF-EN-1325-1.

[Noël and Roucoules, 2008] Noël, F. and Roucoules, L. (2008). The PPO design model with respect to digital enterprise technologies among product life cycle. *International Journal of Computer Integrated Manufacturing*, 21(2) :139–145.

O

[Oh et al., 2001] Oh, Y., Han, S., and Suh, H. (2001). Mapping product structures between CAD and PDM systems using UML. *Computer-Aided Design*, 33(7) :521–529.

[OMG, 2005] OMG (2005). *Computer Aided Design Services Specification*. Object Management Group.

[Orlicky, 1973] Orlicky, J. (1973). Net change material requirements planning. *IBM Systems Journal*.

[Ou-Yang and Cheng, 2003] Ou-Yang, C. and Cheng, M. (2003). Developing a PDM/MRP integration framework to evaluate the influence of engineering change on inventory scrap cost. *The International Journal of Advanced Manufacturing Technology*, 22(3) :161–174.

[Ousterhout, 1998] Ousterhout, J. (1998). Scripting : Higher-level programming for the 21st century. *IEEE computer*, 31(3) :23–30.

P

[Pahl et al., 1966] Pahl, G., Beitz, W., and Wallace, K. (1966). *Engineering design : a systematic approach*. Springer Verlag.

[Paviot et al., 2009] Paviot, T., Cheutet, V., and Lamouri, S. (2009). Design and logistics it federation through product lifecycle support standard. *6th International Conference on Product Lifecycle Management, Bath - GB*.

[Paviot et al., 2010] Paviot, T., Cheutet, V., and Lamouri, S. (2010). Semantic tags for generative multiview product breakdown. *(Accepté) 7th International Conference on Produt Lifecycle Management, Bremen - Germany*.

[Paviot et al., 2008] Paviot, T., Morenton, P., Cheutet, V., and Lamouri, S. (2008). Multicad/multipdm integration framework. *5th International Conference on Product Lifecycle Management, Seoul - Korea*.

[Plantec, 1999] Plantec, A. (1999). Utilisation de la norme STEP pour la spécification et la mise en œuvre de générateurs de code. *Thèse de doctorat de l'Université de Bretagne Occidentale*.

[Plossl, 1993] Plossl, W. (1993). *La nouvelle donne de la gestion de la production*. AFNOR Gestion, Paris.

R

[Rachuri et al., 2008] Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S., Foufou, S., and Sriram, R. (2008). Information sharing and exchange in the context of product lifecycle management : Role of standards. *Computer-Aided Design*, 40(7) :789–800.

[Rao, 2000] Rao, S. (2000). Enterprise resource planning : business needs and technologies. *Industrial Management and Data Systems*, 100(1) :81–88.

[Rose et al., 2007] Rose, B., Robin, V., Girard, P., and Lombard, M. (2007). Management of engineering design process in collaborative situation. *International Journal of Product Lifecycle Management*, 2(1) :84–103.

[Rosén, 2006] Rosén, J. (2006). Federated Through-life Support, Enabling Online Integration of Systems within the PLM Domain. *Proceedings of the 1st Nordic Conference on Product Lifecycle Management*.

S

[Sanner, 1999] Sanner, M. (1999). Python : a programming language for software integration and development. *J. Mol. Graph. Model*, 17 :57–61.

[Saussure, 1916] Saussure, F. (1916). *Cours de linguistique générale*.

[Sen et al., 2006] Sen, S., Lam, S., Rashid, A., Cosley, D., Frankowski, D., Osterhouse, J., Harper, F., and Riedl, J. (2006). Tagging, communities, vocabulary, evolution. *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 181—190.

[Simon, 1962] Simon, H. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6) :467–482.

[Sohlenius, 1992] Sohlenius, G. (1992). Concurrent engineering. *CIRP Annals-Manufacturing Technology*, 41(2) :645–655.

[Song et al., 2007] Song, H., Eynard, B., Roucoules, L., Lafon, P., and Charles, S. (2007). Beyond geometric CAD system : implementation of STEP translator for multiple-views product modeller. *International Journal of Product Lifecycle Management*, 2(1) :1–17.

[Stadtler and Kilger, 2000] Stadtler, H. and Kilger, C. (2000). *Supply Chain Management and Advanced Planning : Concept Models, Software and Case Studies*. Springer Verlag, Berlin.

[Stark, 2005] Stark, J. (2005). Product lifecycle management : 21st century paradigm for product realisation. *Springer*.

[Sudarsan et al., 2005] Sudarsan, R., Fenves, S., Sriram, R., and Wang, F. (2005). A product information modeling framework for product lifecycle management. *Computer-Aided Design*, 37(13) :1399–1411.

T

[Tarn et al., 2002] Tarn, J., Yen, D., and Beaumont, M. (2002). Exploring the rationales for ERP and SCM integration. *Industrial Management and Data Systems*, 102(1) :26–34.

[Terzi, 2005] Terzi, S. (2005). Elements of Product Lifecycle Management : Definitions, Open Issues and Reference Models. *Thèse de doctorat de l'Université Henri Poincaré, Nancy-I et Politecnico di Milano, Italia*.

[Thomas, 1994] Thomas, A. (1994). *La connaissance du vécu en atelier de production*. Thèse de doctorat de l'INP Lorraine, Nancy.

[Thomas and Lamouri, 2000] Thomas, A. and Lamouri, S. (2000). Flux poussé : MRP et DRP. *Techniques de l'Ingénieur, AG5100*.

[Tomovic et al., 2009] Tomovic, C., Walton, A., Ncube, L., Grieves, M., Birtles, B., and Bednar, B. (2009). Measuring the Impact of Product Lifecycle Management : Process Plan, Waste Reduction and Innovations Conceptual Frameworks, and Logic Model for Developing Metrics. *Product Realization*, pages 1—14.

[Tsuchiya, 1993] Tsuchiya, S. (1993). Improving knowledge creation ability through organizational learning. In *Proceedings of International Symposium on the Management of Industrial and Corporate Knowledge ISMICK'93, Compiègne - France*.

[Tursi, 2009] Tursi, A. (2009). Approche basée sur les ontologies pour l'interopérabilité centrée sur le produit des systèmes d'entreprise de production. *Thèse de doctorat de l'Université Henri Poincaré, Nancy I et Politecnico di Bari, Italia*.

[Tursi et al., 2009] Tursi, A., Panetto, H., Morel, G., and Dassisti, M. (2009). Ontological approach for products-centric information system interoperability in networked manufacturing enterprises. *Annual Reviews in Control*, 33(1) :238—245.

U

[Ulrich and Eppinger, 1995] Ulrich, K. and Eppinger, S. (1995). *Product design and development*. McGraw-Hill New York.

V

[Verganti, 1999] Verganti, R. (1999). Planned flexibility : linking anticipation and reaction in product development projects. *Journal of Product Innovation Management*.

[Vernadat, 1996] Vernadat, F. (1996). *Enterprise modelling and integration : principles and applications*. Chapman & Hall.

[Vernadat, 2007] Vernadat, F. (2007). Interoperable enterprise systems : principles, concepts, and methods. *Annual Reviews in Control*, 31(1) :137–145.

[Vitanyi, 2005] Vitanyi, P. (2005). Automatic Meaning Discovery Using Google. *New Scientist*.

W

[Wang et al., 2008] Wang, L., von LASZEWSKI, G., KUNZE, M., and Tao, J. (2008). Cloud Computing : a Perspective Study. *Service Oriented Cyberinfrastruture Lab, Rochester Inst. of Tech–Dezembro de*.

[Wegner, 1996] Wegner, P. (1996). Interoperability. *ACM Computing Survey*, 28(1) :285–287.

[Wheelwright and Clark, 1992] Wheelwright, S. and Clark, K. (1992). *Revolutionizing product development : quantum leaps in speed, efficiency, and quality*. Free Press.

[Wiederhold, 1992] Wiederhold, G. (1992). Mediation in the architecture of future information systems. *IEEE computer*, 25(3) :38–49.

[Womack et al., 1990] Womack, J., Jones, D., and Roos, D. (1990). *The machine that changed the world : how Japan's secret weapon in the global auto wars will revolutionize western industry*. HarperPerennial.

X

[Xu et al., 2008] Xu, H., Xu, X., and He, T. (2008). Research on Transformation Engineering BOM into Manufacturing BOM Based on BOP. *Applied Mechanics and Materials, v10-12, e-Engineering and Digital Enterprise Technology*, pages 99–103.

Z

[Zina et al., 2006] Zina, S., Lombard, M., Lossent, L., and Henriot, C. (2006). Generic Modeling and Configuration Management in Product Lifecycle Management. *International Journal of Computers, Communications and Control*, 1(4) :126—138.

Résumé

Le travail de recherche présenté dans ce mémoire s'intéresse aux problèmes d'interopérabilité dans le domaine du Product Lifecycle Management (PLM) et vise à proposer une méthodologie capable d'y remédier.

Nous définissons l'objectif stratégique du PLM comme la maîtrise de la complexité qui caractérise le développement et le suivi des produits. Dans ce cadre, l'objectif de l'interopérabilité est le contrôle des interactions entre constituants de ce système complexe PLM. Nous montrons que cet objectif est atteint si on assure la continuité et la conservation du flux sémantique qui circule dans le système. Notre étude est restreinte aux seuls domaines de la conception et de la production, mais a l'ambition de pouvoir s'appliquer à d'autres domaines.

La continuité du flux sémantique est assurée par une architecture que nous avons baptisée « médiation multi-échelle orientée services ». Le cœur de cette architecture, le médiateur, se charge d'orienter les flux sémantiques vers les systèmes concernés, et met en correspondance sémantique les informations échangées en s'appuyant sur un méta-modèle d'unification. Nous montrons que pour assurer la robustesse, la flexibilité et l'agilité du système ainsi constitué, il faut privilégier le choix d'un méta-modèle standard générique et extensible. A cet effet, nous proposons une cartographie du standard STEP permettant de choisir le méta-modèle pertinent.

Nous illustrons le choix et la définition de ce méta-modèle dans le cas de deux problèmes d'interopérabilité des domaines de la conception et de la production : l'interopérabilité CAO/PDM et l'interopérabilité PDM/ERP. Nous expliquons notre démarche dans la mise en oeuvre de la norme PLCS. Par ailleurs, nous introduisons un modèle d'annotations sémantiques du produit qui permet de reconstruire des vues multiples correspondant à différents besoins métiers.

Pour la validation de nos propositions et de nos résultats nous proposons des démonstrateurs qui s'appuient sur des composants logiciels dont l'implémentation est détaillée, le code source de ces composants étant librement accessible.

Mots-clés : Product Lifecycle Management (PLM), interopérabilité, STandard for Exchange of Product Data (STEP), Product Lifecycle Support (PLCS), Service Oriented Architecture (SOA), Python.