Mise en oeuvre d'une démarche MDA dans le cadre de développement d'applications métier

Jérémy Tatu

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL DE RHONE-ALPES

MEMOIRE

Présenté en vue d'obtenir

LE DIPLOME D'INGÉNIEUR CNAM

Spécialité : INFORMATIQUE

Option : SYSTEME D'INFORMATION

Par

Jérémy TATU

Mise en œuvre d'une démarche MDA dans le cadre de développement d'applications métier

Soutenu le 17/01/2013

JURY

PRESIDENT :	**M. Christophe PICOULEAU**	***CNAM Paris***
MEMBRES :	**M. Bertrand DAVID**	***CNAM Lyon***
	M. Claude GENIER	***CNAM Lyon***
	M. René CHALON	***Laboratoire LIRIS***
	M. Pascal REGNIAUD	***GFI Progiciels***

Résumé

L'ingénierie dirigée par les modèles représente à ce jour l'avenir du développement dans le domaine du logiciel. Des travaux restent cependant à mener dans la standardisation d'un certain nombre d'outils et dans le formalisme d'un certain nombre d'éléments pour arriver à une uniformisation des pratiques des différents acteurs dans le domaine.

Dans les dernières années, de nombreuses sociétés ont déployé la propre version de cette nouvelle démarche de manière plus ou moins dégradée. La logique de rentabilité reste en effet une problématique très importante dans le monde du développement du logiciel et la solution la plus pérenne est souvent délaissée au profit de la solution la moins coûteuse. Le niveau de complexité de cette démarche n'empêche cependant pas les industriels de se l'approprier et de la rendre productive malgré quelques raccourcis parfois regrettables.

Ce mémoire présente l'utilisation de cette démarche dans le cadre du développement d'une application métier et aspire à démontrer ses différents apports même sur un périmètre assez restreint.

Mots clés : Model Driven Architecture, transformations de modèles, modélisation, démarche 2TUP, application métier.

Summary

Currently, Model Driven Engineering represents the future in the field of software development. Some works need to be complete in the tools standardization and in the formalism of many elements in order to standardize practices.

In recent years, many companies have deployed their own version more or less successfully completed of this new approach. The logic of profitability is a very important issue in software development and the most enduring solution is often neglected in favor of the least expensive one. However, the complexity of Model Driven Engineering does not prevent companies to achieve and make productive approach despite some unfortunate shortcuts.

This paper presents the use of this approach in the development of a business application and aims to demonstrate different contributions even on a fairly limited scope.

Keywords: Model Driven Architecture, model transformations, modeling, approach 2TUP, business application.

Remerciements

En premier lieu, je tiens à remercier chaleureusement M. Bertrand David pour son accueil et la confiance témoignée tout au long de ce stage de validation de fin d'études. Merci également pour son aide précieuse dans l'organisation et la rédaction de mon mémoire.

Je tiens aussi à remercier le laboratoire LIRIS et plus particulièrement l'équipe SILEX de l'INSA pour l'accueil dans leurs locaux.

Je tiens également à remercier le FONGECIF Rhône-Alpes d'avoir sélectionné mon dossier, rendant ainsi possible la finalisation de mon cursus d'ingénieur commencé six années auparavant.

Je tiens également à remercier ma société (GFI Progiciels) et toute son équipe pour m'avoir permis de finaliser mon cursus d'ingénieur au travers de ce congé individuel de formation mais aussi pour m'avoir soutenu et assisté pendant toute cette période. Mon projet n'aurait en outre pas pu aboutir sans l'aide et l'expertise précieuse de Fabrice Daugan et du CSN de Lille.

Je tiens aussi à remercier Eléonore Gondeau (Cnam Rhône-Alpes) pour son accompagnement tout au long de ce projet et son efficacité à répondre aux moindres de mes questionnements.

Je souhaite enfin remercier l'ensemble de mon entourage pour son support précieux. Leurs encouragements réguliers m'ont donné du courage supplémentaire pour aller au bout de ce cursus important pour la suite de ma carrière.

Sommaire

Chapitre 1 : Introduction

L'ingénierie dirigée par les modèles réalise ces dernières années une poussée importante dans le domaine du génie logiciel. L'obsolescence rapide des différentes technologies et la complexité de plus en plus importante des applications participent grandement à l'avènement de ce nouveau processus de développement d'applications informatiques. L'objectif de ce mémoire est de présenter un exemple d'utilisation d'une démarche industrialisée dans le cadre du développement d'une application métier.

Pour remplir cet objectif, nous commençons tout d'abord par présenter le projet et son environnement au sein du chapitre 2. La présentation du laboratoire (2.1) et de l'équipe associée (2.2) est suivie par la description des besoins traduits au sein de l'application métier et par l'aspect organisationnel du projet (2.3).

Au sein du chapitre 3, nous nous intéressons aux définitions préliminaires nécessaires à la compréhension de cette étude. Les notions d'ingénierie logicielle dirigée par les modèles (3.1) et ces différentes déclinaisons (3.2) sont ainsi largement abordées.

Le chapitre 4 nous permet de présenter une vision industrialisée de cette nouvelle ingénierie au travers de la démarche de sélection de la méthode de développement (4.1), la sélection de l'outillage approprié (4.2) et la solution retenue (4.3).

Dans le chapitre 5, nous abordons la mise en œuvre de cette démarche MDA au travers du développement d'une application métier. Les différentes phases permettant d'aboutir à un prototype fonctionnel sont ainsi évoquées successivement : l'étude préliminaire (5.1), la capture des besoins fonctionnels (5.2), la capture des besoins techniques (5.3), l'analyse (5.4) et la conception (5.5). Nous évoquons en outre la place de la démarche MDA dans le projet (5.6) avant de présenter le développement réalisé (5.7).

Enfin, le chapitre 6 permet de faire le bilan de cette étude en abordant successivement l'utilisation de cette démarche dans un contexte industriel (6.1), l'avenir du projet initié (6.2) et les différentes perspectives associées à l'utilisation de cette démarche (6.3). Nous terminons cette étude par un bilan personnel (6.4).

Chapitre 2 : Environnement et organisation du projet

Ce mémoire de fin d'études d'ingénieur CNAM s'est effectué dans le cadre d'un congé individuel de formation de 9 mois accordé par le FONGECIF (FONds de GEstion des Congés Individuel de Formation) de la région Rhône-Alpes. Lors de cette période, mon accueil physique s'est effectué dans le laboratoire LIRIS[1] et plus particulièrement dans les locaux de l'équipe SILEX[2] sous le tutorat de M. Bertrand David.

Dans ce premier chapitre, nous commençons par présenter le laboratoire LIRIS puis son équipe SILEX. Dans un second temps, nous évoquons la nature de l'étude effectuée, sa mise en application et ses aspects organisationnels.

2.1 Laboratoire LIRIS

Le LIRIS, créé début 2003, est le résultat du regroupement de plusieurs laboratoires de recherche lyonnais travaillant dans le domaine des Sciences et Techniques de l'Information et de la Communication. Le LIRIS dispose à ce jour de cinq tutelles :

- le CNRS,
- l'INSA de Lyon,
- l'Université Claude Bernard Lyon 1,
- l'Ecole Centrale de Lyon,
- l'Université Lumière Lyon 2,

La signature d'un contrat d'association avec le CNRS (Centre National de la Recherche Scientifique) lui permet de disposer du label d'Unité Mixte de Recherche (UMR 5205).

[1] Laboratoire d'InfoRmatique en Image et Systèmes d'information (http://liris.cnrs.fr/)

[2] Supported Interactions and Learning by Experience (http://liris.cnrs.fr/equipes?id=44)

Ce laboratoire lyonnais regroupe près de 300 personnes dont 110 chercheurs et enseignants-chercheurs sur différents sites :

- INSA et Université Claude Bernard Lyon 1 situés à Villeurbanne,
- Ecole Centrale de Lyon situé à Ecully,
- Université de Lyon 2 situé à Bron

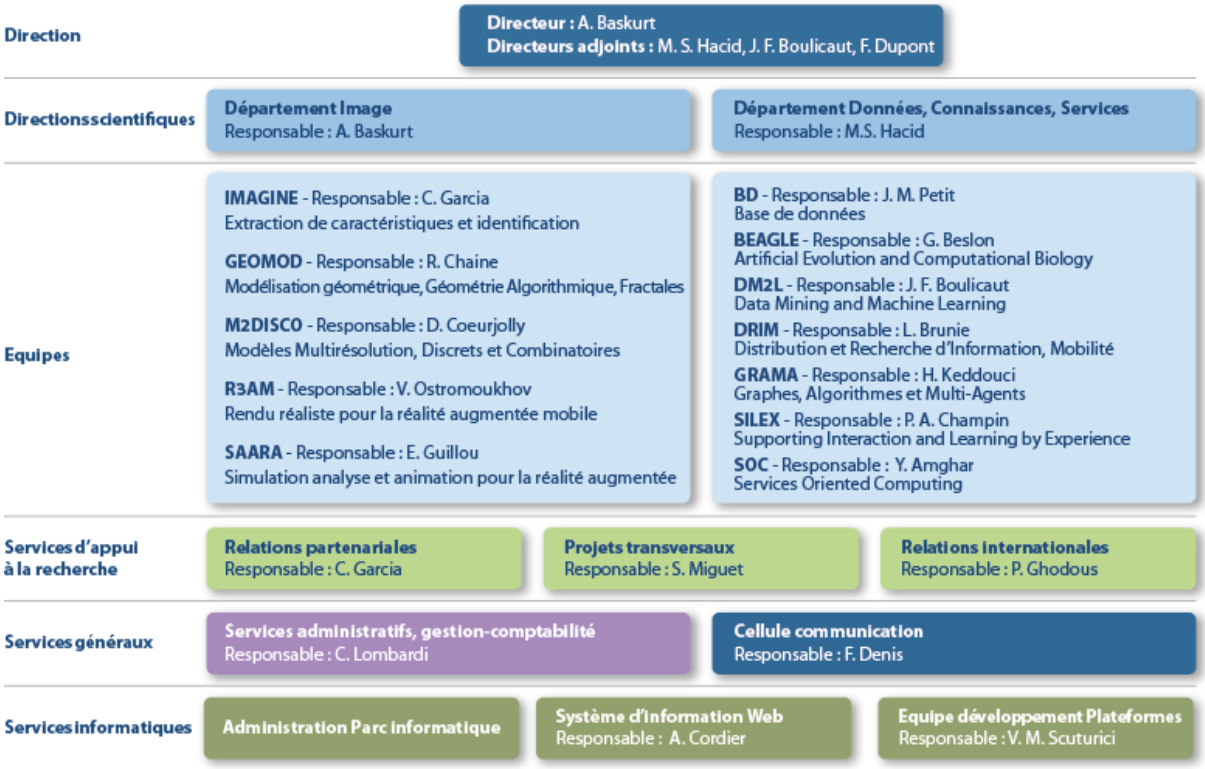

Figure 1 : Organigramme du LIRIS

2.1.1 Les instituts d'attachement

Pour le CNRS, « *le rattachement d'un laboratoire à plusieurs départements scientifiques correspond à la volonté d'un décloisonnement adapté au dynamisme de l'interdisciplinarité sur le terrain* ». En accord avec cette volonté du CNRS, le laboratoire LIRIS dispose de plusieurs instituts d'attachement.

2.1.1.1 L'INS2I[3]

L'institut d'attachement principal, l'INS2I, repose sur cinquante-neuf unités relevant des disciplines informatiques, du traitement du signal et des images, de l'automatique et de la robotique. Avec ses partenaires, l'INS2I a pour objectif de développer le champ des connaissances dans le domaine des sciences informatiques au sens large du terme.

L'institut s'implique ainsi fortement dans les enjeux scientifiques, technologiques et sociétaux liés au développement des sciences informatiques qui jouent déjà un rôle central dans la plupart des disciplines scientifiques. Il s'applique à développer les relations scientifiques avec, entre autres, les mathématiques, les sciences de l'ingénierie et des systèmes, les sciences du vivant ou les sciences humaines et sociales.

2.1.1.2 L'INSIS[4]

L'institut d'attachement secondaire, l'INSIS, est chargé d'assurer le continuum Recherche fondamentale-Ingénierie-Technologie en privilégiant l'approche système, par nature intégrative. La mission de l'INSIS consiste à développer et coordonner les recherches menées en sciences de l'ingénierie et des systèmes, tout en associant les disciplines représentées au sein de l'INSIS :

- Mécanique des matériaux et des structures, acoustique, bio-ingénierie ;
- Fluides, procédés, plasmas, transferts, combustion, thermique ;
- Micro et nanoélectronique, micro et nanotechnologies, micro et nano systèmes, photonique, électromagnétisme, énergie électrique.

L'INSIS privilégie 4 secteurs :

- L'environnement :
  - développement de procédés propres et de matériaux et structures durables,
  - réduction des nuisances sonores, des émissions polluantes et de l'impact environnemental des produits,
  - développement de nouveaux capteurs pour l'observation de l'environnement.
- L'ingénierie de la santé et du vivant, avec des recherches concernant l'imagerie, les micros et nano systèmes pour le vivant, l'ingénierie tissulaire, la biomécanique et le génie des procédés.
- Les nanotechnologies et technologies de la communication, avec l'objectif de concevoir et réaliser des dispositifs et systèmes toujours plus petits et performants, en bénéficiant des avantages de propriétés liées à la réduction de leurs dimensions.
- L'énergie, enjeu majeur pour notre société.

[3] INstitut de Sciences de l'Information et de leurs Interactions (http://www.cnrs.fr/ins2i/)

[4] INstitut des Sciences de l'Ingénierie et des Systèmes (http://www.cnrs.fr/insis/)

2.1.1.3 L'ISCC[5]

Le LIRIS dispose également d'une convention avec l'Institut des sciences de la communication du CNRS (ISCC). Créé en juin 2007, cet institut est une structure transverse et interdisciplinaire consacrée aux sciences de la communication. Les missions de cet institut concernent :

- la construction au CNRS du champ de recherche interdisciplinaire des sciences de la communication,
- le recrutement des chercheurs, la création des laboratoires et l'accueil des chercheurs étrangers,
- la mise en place pour ce nouveau champ de connaissance, au CNRS et avec l'université et les autres institutions de recherche, d'une politique structurelle interdisciplinaire,
- le renforcement de la coopération internationale,
- le financement des appels d'offres interdisciplinaires.

2.1.2 Les thématiques de recherche du laboratoire

Les activités principales du laboratoire LIRIS sont regroupées en deux départements thématiques :

- Image,
- Données, Connaissances et Services

2.1.2.1 Département Image

Les équipes du département Image développent des méthodes d'analyse des données issues de capteurs (2D, 2D+T, 3D), des outils permettant une meilleure compréhension du contenu et des modèles de représentation de données numériques multidimensionnelles. Les thématiques étudiées par ce département recouvrent les domaines de l'analyse d'image, de la modélisation, de la simulation ou encore du rendu.

Les modèles de représentation des données sont très variés et vont des grilles régulières issues des capteurs, aux graphes, maillages, modèles géométriques, modèles procéduraux, modèles statistiques ou physiques. L'hybridation de ces modèles enrichit encore l'espace des représentations. Les techniques développées s'appuient sur les compétences des équipes en traitement du signal et de l'image, modélisation géométrique, géométrie algorithmique, géométrie discrète, topologie, graphes, rendu réaliste et rendu augmenté.

[5] Institut des Sciences de la Communication du CNRS (http://www.iscc.cnrs.fr/)

2.1.2.2 Département Données, Connaissances et Services

Les activités du département Données, Connaissances et Services s'appuient sur une recherche combinant des explorations théoriques et des implantations effectives autour de pôles de compétences partagée par les cinq équipes :

- découverte de connaissances (data mining, modélisation de systèmes complexes, ingénierie des connaissances),
- ingénierie des données et des services (sécurité et confidentialité, modélisation, intégration et interrogation, composition de services).

2.2 Equipe SILEX

L'équipe SILEX, constituée d'une cinquantaine de personnes, concentre ses recherches sur l'ingénierie de l'expérience tracée, la personnalisation des Environnements Informatiques pour l'Apprentissage Humain (EIAH) et l'assistance aux personnes en situation de handicap. L'équipe est largement ouverte à la pluridisciplinarité : psychologie cognitive, sémiotique, linguistique, ergonomie, éducation, sociologie par exemple. Elle mobilise ces différentes collaborations pour étudier le système homme-machine comme un système apprenant unique. Les questions de recherche de cette équipe s'organisent en trois thèmes.

2.2.1 Thème 1 - Dynamique des connaissances et expérience tracée

L'objectif de ce thème est de proposer une ingénierie de la dynamique des connaissances exploitant les inscriptions de l'activité individuelle et collective de l'utilisateur dans l'environnement informatique. Ces inscriptions s'organisent en une représentation de l'expérience tracée, source de connaissances co-construites et disponibles pour accompagner l'activité du couple utilisateur/machine autour des connaissances. Ce thème se décline en trois facettes :

- ingénierie de la dynamique des connaissances
- trace modélisée et Système de Gestion de Base de Traces
- raisonnement à partir d'épisodes issus de l'expérience tracée

2.2.2 Thème 2 - Systèmes interactifs adaptatifs

Ce thème s'intéresse à l'Interaction entre l'Humain et la Machine (IHM), tant dans ses dimensions individuelles que collectives. Un point clé se trouve dans la définition de modèles et de méthodes pour la conception de systèmes interactifs adaptables et adaptatifs. Au-delà des interfaces classiques, des recherches sont effectuées sur les interactions « innovantes » utilisant des interfaces mobiles et portées (wearable computing), la réalité mixte et augmentée, l'informatique ambiante et ubiquitaire.

Plusieurs domaines d'application sont concernés par ces travaux : les EIAH[6] et les Serious Games, le handicap, les pratiques documentaires et audiovisuelles, mais aussi l'industrie, la ville 2.0 ou l'habitat intelligent. Ce thème se décline en trois facettes :

- conception de systèmes interactifs,
- adaptation et accessibilité,
- évaluation et usage.

2.2.3 Thème 3 - Co-conception d'EIAH situés

Les travaux de ce thème concernent la co-conception d'EIAH situés, c'est-à-dire adaptables à la fois au contexte et aux spécificités de l'apprenant. La co-conception est effectuée par les différents acteurs concernés par l'EIAH : auteurs, tuteurs, apprenants (dans un contexte individuel ou collectif), mais aussi l'EIAH lui-même. Les thématiques de recherche se développent entre autres dans les contextes du e-learning, des serious games et du mobile learning et se décline en deux facettes :

- co-conception système-humain des EIAH,
- personnalisation des EIAH.

L'approche utilisée est en lien avec l'ingénierie dirigée par les modèles. La généricité de l'approche ne s'oppose pas à la spécificité des besoins : elle permet en effet de concevoir des outils qui prennent en compte ces besoins par leur caractère adaptable et personnalisable.

2.3 Vers un outil de manipulation du paramétrage

Le but de cette étude consiste en la confrontation de l'approche MDA avec le besoin pratique de développement d'applications métier. Afin de mettre en application la confrontation de ces visions théoriques et industrielles, il a été décidé de travailler sur un projet concret de développement d'applications.

[6] Environnements Informatiques pour l'Apprentissage Humain

Dans les projets de recherche, il est parfois compliqué de trouver des exemples pratiques d'application. Dans notre cas, l'expérience acquise au sein de la société GFI Progiciels a permis de trouver un projet concret de cette confrontation. En accord avec cette société et le laboratoire, nous avons donc travaillé au cours de cette période sur la mise en œuvre d'un outil de manipulation du paramétrage du progiciel IODAS dont le développement intègre les spécificités de la démarche MDA.

2.3.1 Le progiciel IODAS

Le progiciel IODAS est un outil permettant de gérer les systèmes d'information destinés aux professionnels de l'action sociale dans les départements. L'objectif de cet outil est de faciliter le travail de chacun quel que soit son domaine d'intervention ou son rôle au sein des services du Conseil Général (travailleur social ou médico-social, agent administratif, médecin, cadre). Le produit est transversal aux différents domaines fonctionnels de l'action sociale départementale :

- Aides aux Personnes Agées (APA),
- Aide Sociale à l'Enfance (ASE),
- Maison Départementale des Personnes Handicapées (MDPH),
- Protection Maternelle et Infantile (PMI),
- Revenu Minimum d'Insertion (RMI),
- Revenu de Solidarité Active (RSA),
- Aide Sociale Générale (ASG),
- Fonds de Solidarité Logement (FSL).

2.3.2 L'outillage à développer

La philosophie de fonctionnement du progiciel IODAS est fortement basée sur les possibilités de paramétrage des différents processus de gestion. L'interprétation de la législation par les experts fonctionnels permet de proposer un paramétrage modèle associé à chaque domaine de l'action social départemental. Le produit est ensuite adaptable aux particularités de chacun des clients mais nécessite en contrepartie de nombreuses manipulations au niveau du paramétrage :

- mise à disposition d'un paramétrage modèle,
- adaptation du paramétrage en fonction des spécificités sur l'ensemble des bases du client (production, formation, qualification, etc.),

La manipulation du paramétrage représente donc une part importante dans la mise en place d'un projet client. En outre, les fonctionnalités et l'évolution régulière du paramétrage au fil des différentes versions du produit engendrent une réelle problématique de la maintenance d'un outil périphérique à IODAS.

L'application de cette étude de la confrontation entre la théorie et l'industrialisation de la démarche MDA concerne par conséquent le développement d'un système informatique permettant de mieux maîtriser les différentes manipulations de paramétrage. Beaucoup trop de projets de mise en œuvre clients sont impactés négativement par la nécessaire maintenance manuelle du paramétrage sur plusieurs bases de données. Ces opérations de recopie de paramétrage apparaissent comme chronophages, sans plus-value pour les consultants et sources potentielles d'erreur avec l'outil existant. Un outil performant et mettant en œuvre les fonctionnalités nécessaires pourraient optimisés ces opérations et ainsi améliorer la productivité des équipes.

2.3.3 Organisation de l'étude

Dans un premier temps, nous donnons les définitions préliminaires nécessaires à la compréhension des travaux effectués. Nous présentons ensuite une industrialisation de cette démarche au travers de l'exemple de la société GFI. La mise en pratique de cette vision industrielle de la démarche MDA est abordée dans le cinquième chapitre. Nous tentons ensuite de faire le bilan de la confrontation entre la théorie et la pratique industrielle de la démarche MDA. Nous finissons par évoquer l'avenir du projet démarré au sein de cette étude et les perspectives de l'utilisation de cette démarche MDA au sein de l'entité GFI Progiciels.

Ce projet, éminemment mono-personnel, m'a permis d'intervenir à tous les niveaux du processus d'analyse et conception d'une application informatique et de toucher du doigt de nombreux métiers de l'informatique :

- Chef de projet,
- Architecte informatique,
- Analyste programmeur,
- Concepteur logiciel.

2.3.3.1 Liste des tâches

Numéro de tâche	Identifiant tâche	Tâche	Charge estimé JH	Date de début	Date de fin
1	**THE**	**Analyse théorique de la démarche MDA**	30 jours	30/01/2012	12/03/2012
2	**AFR**	**Analyse fonctionnelle outil de recopie**	15 jours	06/02/2012	24/02/2012
3	**AFM**	**Analyse fonctionnelle outil de migration**	15 jours	28/02/2012	19/03/2012
4	**RSP**	**Rédaction des spécifications de l'application**	30 jours	20/03/2012	30/04/2012
5	**MAQ**	**Maquette**	2 jours	01/05/2012	02/05/2012
6	**GFI**	**Formation à la vision MDA de GFI**	10 jours	03/05/2012	23/05/2012
7	**MIHM**	**Modélisation de l'IHM**	5 jours	24/05/2012	30/05/2012
8	**MSME**	**Modélisation des services métiers**	3 jours	31/05/2012	04/06/2012
9	**EGEN**	**Etude des générateurs développés par GFI**	10 jours	05/06/2012	18/06/2012
10	**TGEN**	**Réception et test du générateur de code IODAS**	2 jours	19/06/2012	20/06/2012
11	**CIHM**	**Codage de l'IHM**	20 jours	21/06/2012	18/07/2012
12	**CSME**	**Codage des services métiers**	20 jours	19/07/2012	10/09/2012
13	**TQU**	**Tests de qualification pour le prototype fonctionnel**	3 jours	11/09/2012	13/09/2012
14	**ADO**	**Ajout d'autres domaines de paramétrage**	30 jours	14/09/2012	25/10/2012
15	**MEM**	**Rédaction du mémoire**	30 jours	17/09/2012	26/10/2012

Tableau 1 : Liste des tâches du projet

2.3.3.2 Diagramme de Gantt

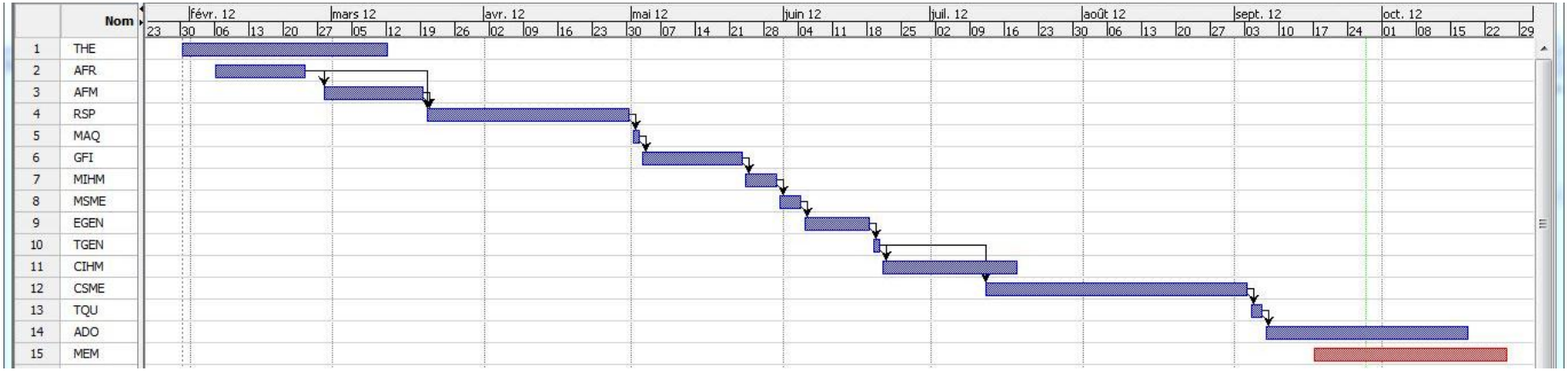

Figure 2 : Diagramme de Gantt de l'étude

Chapitre 3 : Etat de l'art et définitions préliminaires

L'ingénierie logicielle dirigée par les modèles (IDM), en anglais Model Driven Engineering (MDE), est considéré à ce jour comme une solution technologique d'avenir pour l'industrie du logiciel. Elle fait à ce titre partie des technologies clés 2010 citées par le Ministère de l'industrie dans le domaine des technologies de l'information et de la communication (TECHNOLOGIES CLÉS 2010).

Dans ce chapitre, nous souhaitons introduire les différents éléments nécessaires à la compréhension de ce travail. Nous présentons tout d'abord la notion d'ingénierie logicielle dirigée par les modèles, puis nous faisons ensuite un focus particulier sur sa composante la plus célèbre, la démarche MDA[7].

3.1 L'ingénierie logicielle dirigée par les modèles

Depuis les années 2000, nous observons un changement de paradigme en ingénierie du logiciel. L'idée centrale de la composition d'objets est de plus en plus complétée par la notion de transformation de modèles.

[7] Model Driven Architecture

3.1.1 Un peu d'histoire

A la fin des années 60, des progrès notables dans le domaine de la construction du matériel informatique rendent les ordinateurs de troisième génération de plus en plus puissants et de moins en moins coûteux. Dans le même temps, le génie logiciel relève quant à lui du domaine de l'artisanat :

- coût exorbitant de la construction de logiciels (200 millions de dollars pour fabriquer OS-360),
- délais de livraison non respectés (2 ans de retard pour les premiers compilateurs ADA),
- logiciels non évolutifs donc rapidement obsolètes (codage en assembleur),
- performances poussives des systèmes,
- fiabilité aléatoire,
- convivialité d'utilisation des systèmes discutable (interfaces homme/machine textuelles).

Pour sortir de cette crise, la notion de génie logiciel a été mise en place et a permis un bond en avant spectaculaire en maximisant la durée de vie et la qualité des logiciels, tout en minimisant les coûts et le délai. La Figure 3 donne un aperçu de l'évolution du génie logiciel dans le temps.

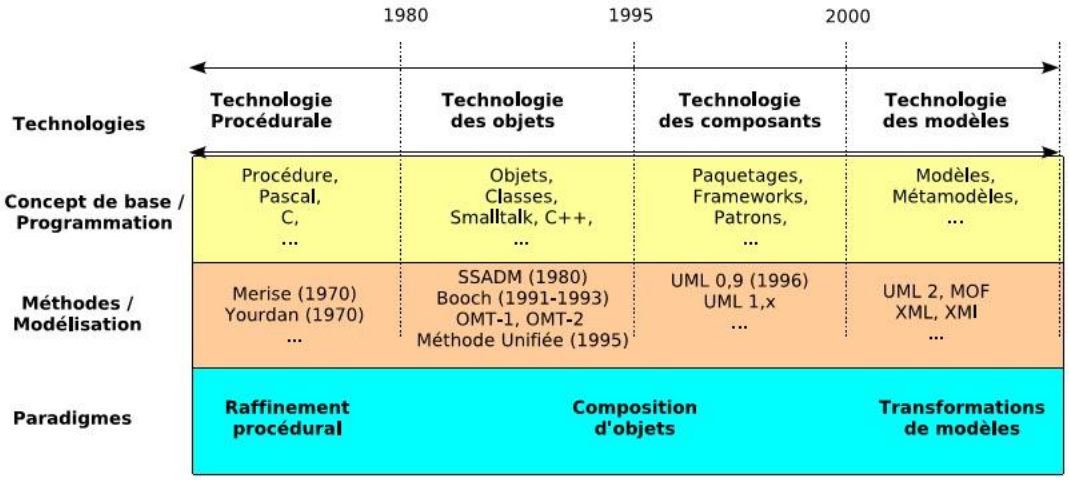

Figure 3 : Evolution du génie logiciel

Depuis le début des années 2000, différents constats dans le cadre du développement logiciel ont mis en exergue la nécessité du changement. L'évolution permanente des besoins métier de l'entreprise et des technologies rend les systèmes informatiques de plus en plus complexes et rapidement obsolètes. Les coûts de maintenance et de migration sont devenus insupportables pour la plupart des entreprises. La technologie des objets et des composants, mise en œuvre pour remplacer la logique procédurale, ne suffit aujourd'hui plus pour répondre aux nouveaux défis proposés par l'industrie du logiciel.

L'Ingénierie logicielle Dirigée par les Modèles (IDM ou MDE[8]) vise à favoriser un génie logiciel plus proche des métiers en autorisant une appréhension des applications selon différents points de vue exprimés séparément. Elle intègre également comme fondamentales la composition et la mise en cohérence de ces perspectives. De plus, elle se veut productive en automatisant la prise en charge des outils relatifs à la validation des modèles, les transformations et les générations de code.

Le terme de l'ingénierie dirigée par les modèles est récent et souvent présenté comme une révolution dans le domaine de l'informatique. Les détracteurs clament au contraire que « tout cela existe depuis très longtemps ». La vérité se situant entre ces deux visions, l'IDM doit être plutôt considérée comme une évolution. Le principe de modélisation existe en effet depuis très longtemps et a déjà largement prouvé son utilité, en mécanique par exemple. Léonard de Vinci, lui-même, les utilisait déjà (Figure 4).

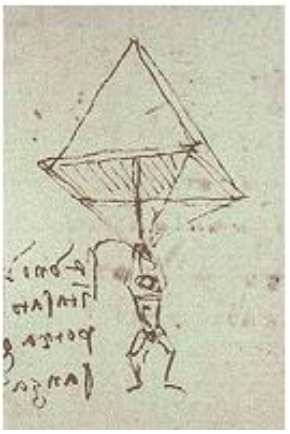

Figure 4 : Modèle de parachute par De Vinci

L'idée centrale de la composition d'objet est complétée par la notion de transformation de modèle. L'idée de systèmes logiciels composés d'objets communicants n'est pas cependant pas contradictoire avec l'idée d'un cycle de vie du logiciel considéré comme une chaîne de transformations de modèles. Cependant, les apports mutuels de la technologie des objets et de l'ingénierie des modèles devraient se cumuler mais les principes en sont fondamentalement différents (Bézivin, Sur les principes de base de l'ingénierie des modèles, 2004). Alors que l'approche objet est fondée sur les relations essentielles d'instanciation et d'héritage, l'IDM est basée sur un autre jeu de concepts et de relations.

3.1.2 Concepts socles

Le concept central de l'IDM est la notion de modèle, pour laquelle, il n'existe pas à ce jour de définition universelle (Jézéquel, Combenale, & Vojtisek, 2012). L'objectif fondateur et fédérateur de cette ingénierie est d'offrir des processus de développement logiciel complets, cohérents et intelligibles par tout informaticien. Il s'agit ainsi de spécifier totalement à des niveaux d'abstraction distincts, mais interdépendants, une application et ses composants par toute une série de manipulations automatiques (Cariou & Barbier, 2008).

[8] Model Driven Engineering

L'ingénierie logicielle guidée par les modèles permet de s'abstraire complètement des préoccupations technologiques telles que les plateformes et les langages de programmation. Elle permet aussi de capitaliser les architectures métier et applicatives. Pour mettre en œuvre cette méthodologie et ainsi rendre les modèles « productifs », trois notions apparaissent comme fondamentales :

- interprétation des modèles par les machines,
- productivité des modèles,
- séparation des différents aspects d'un système

3.1.2.1 Interprétation des modèles par les machines

Le besoin d'interprétation des modèles par les machines nécessite l'introduction des concepts de système, de modèle et de métamodèle. Ces éléments sont associés à des relations de représentation (entre un système et un modèle) et de conformité (entre un modèle et un métamodèle).

La chaise (Figure 5) représente le système, l'image de la chaise correspond à une modélisation selon un point de vue particulier et l'extrait du dictionnaire représente le langage de définition des modèles.

Figure 5 : One and three chairs (Joseph Kosuth 1955)

3.1.2.1.1 Modèles

« A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and reversibility » (Jeff Rothenberg)

Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développer. Il s'agit d'une description partielle du système selon un point de vue particulier se limitant à représenter seulement certains aspects pertinents de l'univers du système que l'on modélise (Figure 6). Le modèle doit être en mesure de répondre à des questions en lieu et place du système modélisé.

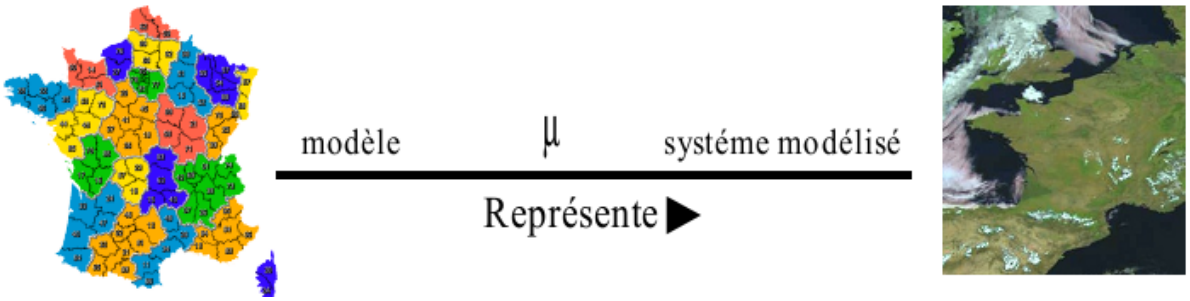

Figure 6 : Modélisation de la France

Parallèlement, la définition de modèle requiert la mise en œuvre de métamodèle représentant l'ensemble des concepts et relations nécessaires pour exprimer des modèles (Kadima, 2005).

3.1.2.1.2 Métamodélisation

« A metamodel is a model that defines the language for expression of a model » (MOF standard version 1.4)

Un métamodèle est un langage qui permet d'exprimer des modèles. Il définit les concepts ainsi que les relations entre concepts nécessaires à l'expression de modèles. Un modèle écrit dans un métamodèle donné sera dit « conforme à » ce dernier et considéré comme une instance de celui-ci. Par analogie aux langages de programmation, la relation entre ces deux éléments (Figure 7) est comparable à celle entre une variable et son type ou un objet et sa classe.

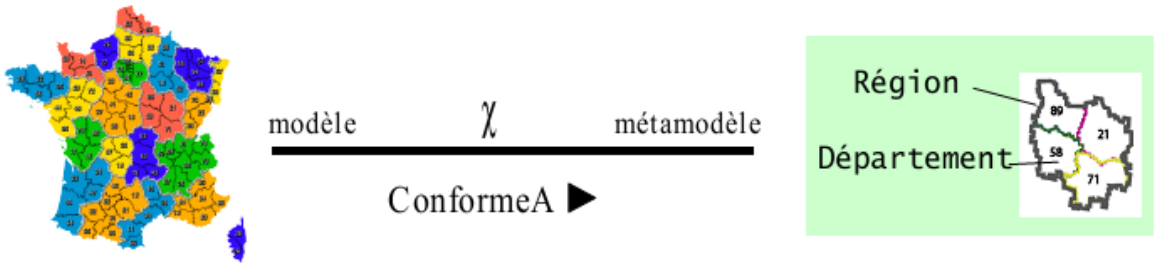

Figure 7 : Métamodélisation de la carte de France

Le formalisme de modélisation n'est cependant pas suffisant pour mettre en œuvre une démarche telle que l'IDM. Des liens de traçabilité et des transformations de modèles doivent être exprimés ce qui implique le besoin de modéliser aussi les métamodèles.

3.1.2.1.3 Espaces techniques

Un espace technique est l'ensemble des outils et techniques issus d'une pyramide de métamodèles dont le sommet est occupé par une famille de métamétamodèles similaires. Cette logique est utilisée depuis longtemps dans de nombreux domaines de l'informatique (Figure 8).

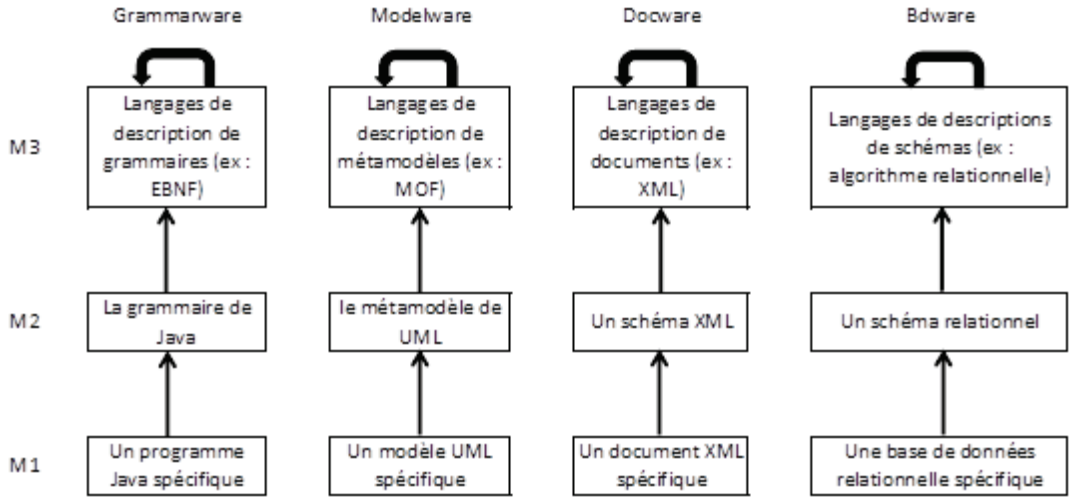

Figure 8 : Espaces techniques (Favre, Estublier, & Blay-Fornarino, L'ingénierie dirigée par les modèles : au-delà du MDA, 2006)

Le métatriangle (Figure 9) permet d'illustrer cette notion d'espaces techniques. Le système est représenté par un modèle qui est conforme à un métamodèle représentant un langage. Un modèle conforme au métamodèle est donc par extension un mot du langage.

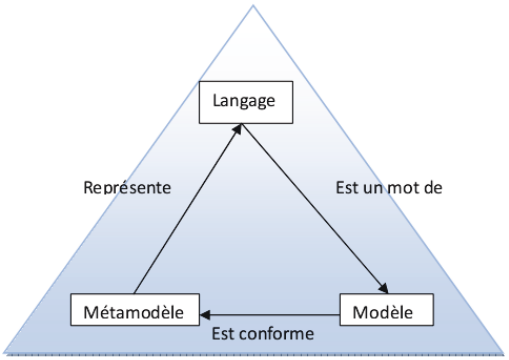

Figure 9 : Métatriangle

Ces multiples niveaux de modélisation permettent l'interprétation des modèles par les machines. Ce n'est cependant pas encore suffisant pour rendre les modèles productifs.

3.1.2.2 Productivité des modèles

La productivité des modèles nécessite la production d'artéfacts exécutables, résultat d'opérations sur les modèles connu sous la terminologie de transformations de modèles. Celles-ci sont composées d'un ensemble de règles (illustré Figure 10) qui décrivent globalement comment un modèle décrit dans un langage source peut être transformé en un modèle décrit dans un langage cible. Cette notion de transformation de modèles fait

également intervenir le concept de plate-forme. Tout système exécutable est défini dans le contexte d'une plate-forme qui sert de support à la construction et à l'exécution des systèmes. Celle-ci peut être logicielle (systèmes d'exploitation, intergiciels, machines virtuelles, environnement de programmation, XML) ou matérielles (PC, PDA, calculateurs ou chipsets).

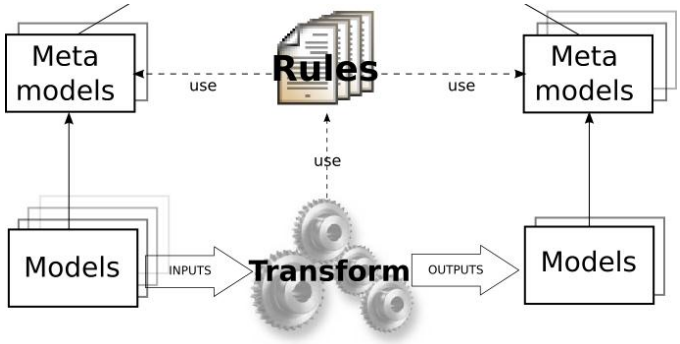

Figure 10 : Transformations de modèles

L'ingénierie des modèles englobe, entre autres, la génération de code à partir de modèles, l'élaboration de modèles à partir d'applications existantes ou l'intégration de modèles. Toutes les manipulations de modèles correspondent à des transformations de modèles qui s'appuient d'une part sur un enrichissement progressif des spécifications de l'application et d'autre part sur des transformations autorisées par le métamodèle UML. On distingue deux types de transformations de modèles :

- les transformations endogènes,
- les transformations exogènes.

3.1.2.2.1 Transformations endogènes

Les transformations de type endogène se réalisent au sein du même espace technologique. Les modèles source et cible sont dans ce cas conformes au même méta-modèle (Figure 11). Par exemple, on transforme un modèle UML en un autre modèle UML dans le cadre d'un raffinement ou de l'intégration d'un design pattern.

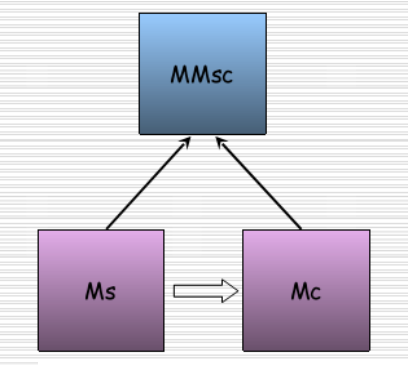

Figure 11 : Transformation endogène (Topcased, 2007)

3.1.2.2.2 Transformations exogènes

Les transformations exogènes se font dans deux espaces technologiques différents. Les modèles source et cible sont conformes à des méta-modèles différents (Figure 12). Par exemple, un modèle UML pourra être transformé en programme Java ou un fichier XML pourra être transformé en schéma de base de données relationnelle. Ce type de transformation est très utilisé dans l'IDM du fait des recommandations de séparation des aspects.

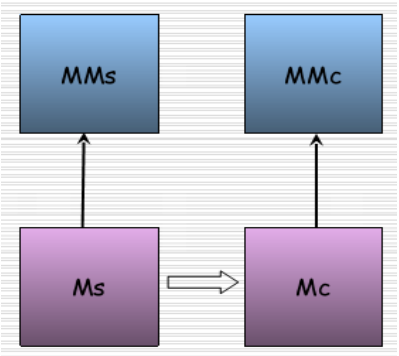

Figure 12 : Transformation exogène (Topcased, 2007)

3.1.2.3 Séparation des différents aspects d'un système

Un système, en général très complexe et donc difficile à appréhender dans son intégralité, est souvent décomposé en plusieurs sous-systèmes plus simples. Chacun de ces éléments se focalise sur un aspect particulier du système et est représenté par un modèle décrit à l'aide d'un métamodèle spécifique. Cette diversité exige l'utilisation d'un langage unificateur et d'intégration pour les outils qui vont manipuler les différents métamodèles : la notion de métamétamodèle en tant que langage unique. La conception dans l'IDM se fait donc par une modélisation hiérarchique avec quatre niveaux et nécessite l'utilisation de nombreux standards.

3.1.3 Les standards associés

L'Ingénierie logicielle Dirigée par les Modèles préconise l'utilisation de standards dans les domaines de la modélisation, de la manipulation de métamodèles et de la transformation de modèles.

3.1.3.1 Les normes de modélisation

Au niveau de la modélisation, l'IDM nécessite un socle basé soit sur le langage UML (Unified Modeling Language), soit sur des DSL (Domain Specific Language). Le choix entre les deux langages correspond d'ailleurs à une préoccupation centrale de la plupart des conférences scientifiques sur le sujet (Journées Neptune 2012).

3.1.3.1.1 UML

L'émergence de l'approche objet, dans les années 1990, a provoqué un foisonnement de modèles qui a trouvé sa conclusion à travers le langage UML (Desfray, 2009). Adopté et standardisé par l'Object Management Group (OMG) en 1997, UML est à ce jour un outil de communication incontournable dans le domaine de l'informatique. Retenu dans la grande majorité des projets informatiques et connu d'un très large public, il permet de communiquer indépendamment du domaine métier concerné. Son évolution lui permet de disposer d'une large couverture (données, traitements, besoins, architectures, composants). La dernière version diffusée à ce jour par l'OMG est la version UML 2.4.1 datant d'août 2011.

3.1.3.1.1.1 Evolutions du langage

Le langage UML correspond à la fusion des trois méthodes qui ont le plus influencé la modélisation objet au milieu des années 90 : OMT[9], Booch (du nom de son créateur) et OOSE[10]. La première version (1.0) a été proposée à l'OMG en janvier 1997 et le langage n'a cessé d'évoluer depuis cette époque (Figure 13).

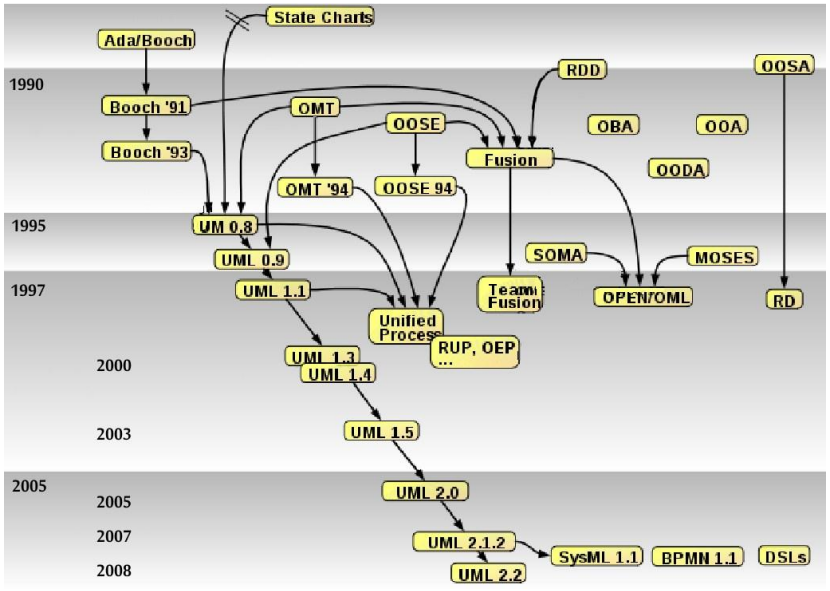

Figure 13 : Evolutions du langage UML (Unified Modeling Language, 2012)

Portée par tous les plus gros industriels du domaine de l'informatique, le langage UML évolue rapidement pour aller vers une complétude de plus en plus importante. Dans l'histoire de ce langage, la version 2.0 apparaît comme majeure car elle apporte des innovations radicales et étend largement le champ d'application d'UML. Le langage s'est enrichi de plusieurs diagrammes ce qui porte son total à 13 (Figure 14).

[9] Object Modeling Technique

[10] Object-oriented software engineering

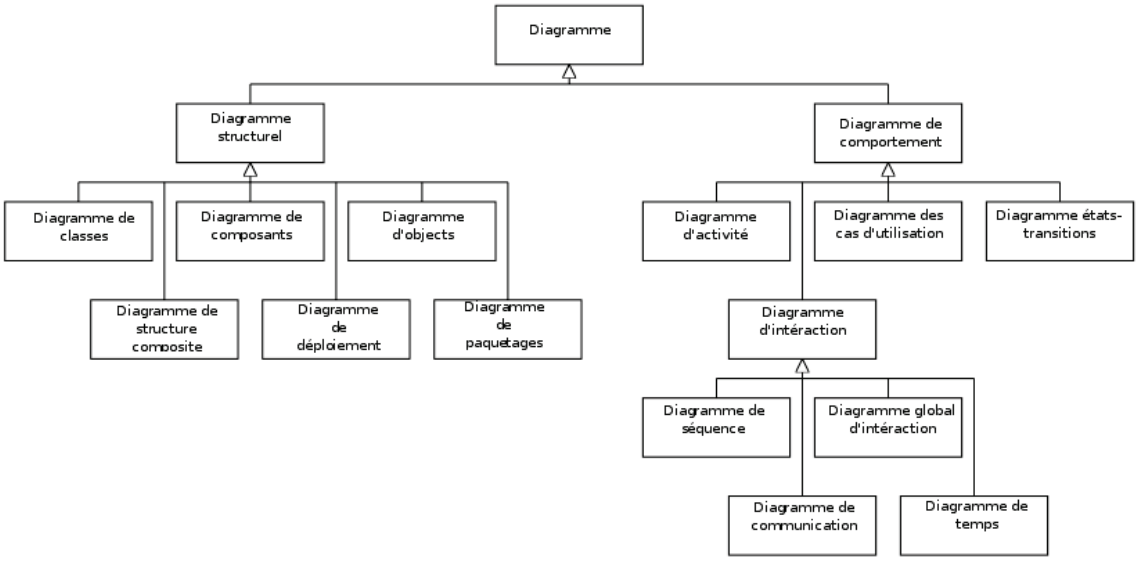

Figure 14 : Ensemble des diagrammes (UML2.0)

3.1.3.1.1.2 Profils

La prétention d'universalité d'UML ne peut pas s'affranchir de la nécessité d'adaptation à chaque contexte de son usage. Le mécanisme des profils (très enrichi sous la version UML2) permet dans ce cadre de définir des extensions permettant d'adapter le standard, sans modifier sa définition, pour cibler des domaines d'exploitation particuliers :

- préciser une terminologie propre à un domaine ou à une plateforme (terminologie EJB[11]),
- donner une syntaxe à des concepts qui n'en ont pas (actions et nœuds de contrôle dans le domaine du temps réel),
- donner une notation différente à des symboles existants (une image de bouton dans le cadre d'une modélisation d'une IHM),
- ajouter de la sémantique (gestion des priorités à réception d'un signal),
- ajouter des informations utiles à la transformation de modèles (règles de transformations vers du code).

Cette précision de la sémantique se caractérise tout d'abord par des stéréotypes qui définissent, dans le méta-modèle, des sous-méta-classes. Des tagged-values (valeurs marquées) définissent quant à elles des nouvelles propriétés. Enfin, des contraintes précisent le rôle des éléments d'UML afin d'étendre ou restreindre la sémantique du modèle obtenu. Le langage naturel peut être utilisé, mais OCL[12] est, dans ce cas, tout indiqué.

[11] Enterprise Java Beans

[12] Object Constraint Language

3.1.3.1.1.3 Un langage de contraintes : OCL

Une contrainte correspond à une restriction sur une ou plusieurs valeurs d'un modèle orienté objet. Celle-ci peut être exprimée avec un langage formel tel que OCL (Figure 15), un langage semi-formel ou en langage naturel.

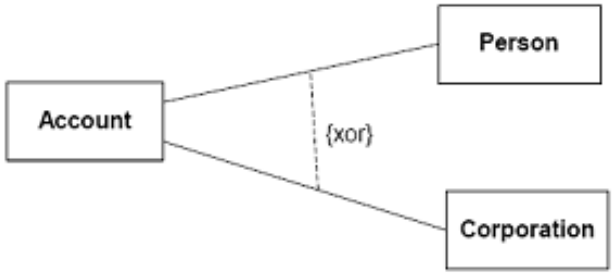

Figure 15 : Un exemple de contraintes OCL

3.1.3.1.1.4 Un langage d'action : Action Semantics

Jusqu'à sa version 1.4, UML était très critiqué parce qu'il ne permettait pas de spécifier des créations, des suppressions ou des modifications d'éléments de modèle. La nécessité de modéliser ces changements d'états, non spécifiables avec le langage OCL, ont ainsi donné naissance au langage Action Semantics (Blanc, 2005). Le langage AS permet la spécification d'actions et donc de modéliser pleinement le corps des opérations UML. A partir de la version 2.0 de UML, il est totalement intégré au métamodèle.

3.1.3.1.2 CWM[13]

Le CWM représente une démarche d'échange de méta-données entre systèmes logiciels (Tolbert, 2000). Ce métamodèle inspiré d'UML couvre le cycle de vie complet de modélisation, construction et gestion des entrepôts de données. Il représente les métadonnées aussi bien métiers que techniques les plus souvent manipulées dans les applications d'un datawarehouse (Kadima, 2005). Promu par l'OMG, il est conforme au méta-méta-modèle MOF et fait ainsi partie intégrante du noyau des standards MDA.

[13] Common Warehouse Metamodel

3.1.3.1.3 DSL[14]

Au-delà de la mécanisation de l'utilisation d'UML, l'IDM permet d'outiller des langages de modélisation dédiés à chaque aspect méthodologique ou technologique du développement. La création d'un langage spécifique à un domaine permet de proposer aux utilisateurs un environnement de travail adapté à ce domaine, c'est-à-dire manipulant directement les concepts de celui-ci (Vojtisek, 2009). Cette notion s'oppose aux langages de programmation dits généralistes, de type Java ou C, ou bien aux langages de modélisation généralistes de type UML. Le langage dédié doit répondre à certains critères :

- Il doit être universel : il doit résoudre tous les problèmes auxquels il est dédié.
- Il doit être implémenté et réalisé sur un ordinateur.
- Il doit avoir une logique naturelle.

L'avantage des DSL réside dans leur puissance d'expression, puisque conçus pour une problématique précise, et dans leur facilité de traitement lorsqu'il s'agit d'effectuer de la validation ou de la transformation. En contrepartie, un DSL nécessite de définir un méta-modèle et une notation, ce qui représente un temps de développement et d'apprentissage du langage supplémentaire.

La création de ce langage nécessite la manipulation cohérente de la syntaxe abstraite, de la syntaxe concrète et des domaines sémantiques (Jézéquel, Combenale, & Vojtisek, 2012). Les relations entre ces différentes entités (Figure 16) permettent la construction d'un DSL de type Kermeta.

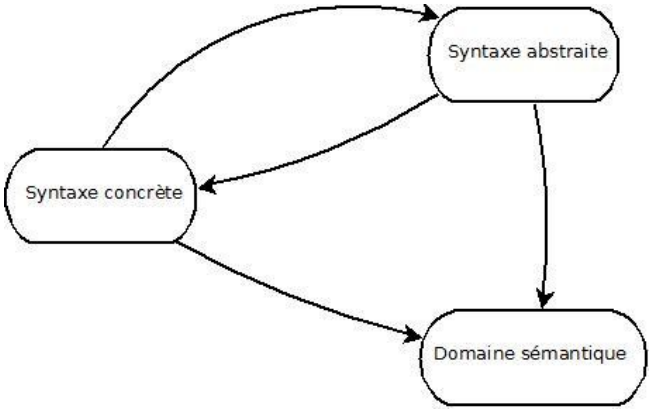

Figure 16 : Création d'un DSL

[14] Domain Specific Languages

3.1.3.1.4 MOF

Le Meta Object Facility (MOF) est un standard OMG définissant un langage abstrait pour la spécification de métamodèles dont le but est de définir un langage de modélisation unique. Le MOF, en tant que métamétamodèle unique et réflexif, permet de construire les autres modèles standards de l'OMG comme UML (Figure 17) et CWM. La puissance de ce langage se situe dans la l'interopérabilité de métamodèles différents et donc dans la possibilité de les relier par composition.

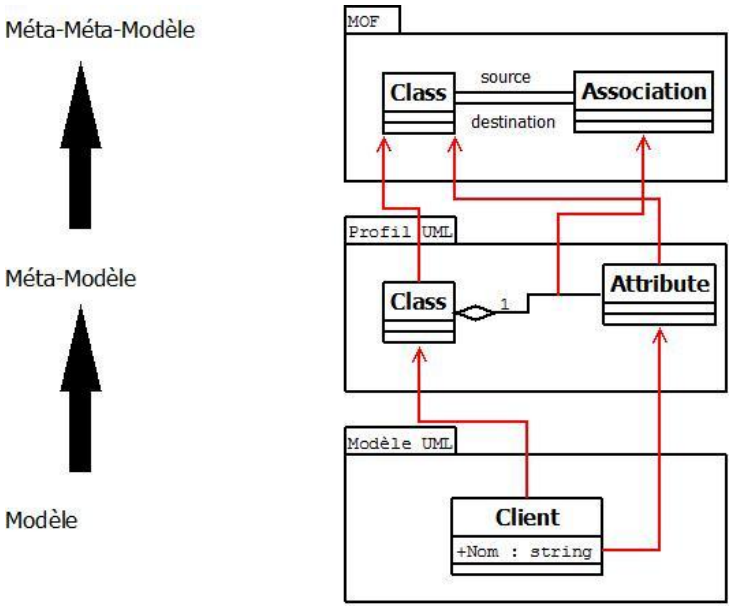

Figure 17 : Mapping entre entités de différents niveaux de métamodélisation

Constitué d'un ensemble relativement restreint de concepts objet, il définit les éléments essentiels pour construire les modèles orientés systèmes. Le MOF est basé sur un framework de modélisation objet dont les principaux concepts (Figure 18) sont les suivants :

- les classes : définition des métaobjets du MOF,
- les associations : modélisation des relations binaires entre les métaobjets,
- les types de données : modélisation des autres données,
- les paquetages : maintien de la modularité de ces modèles.

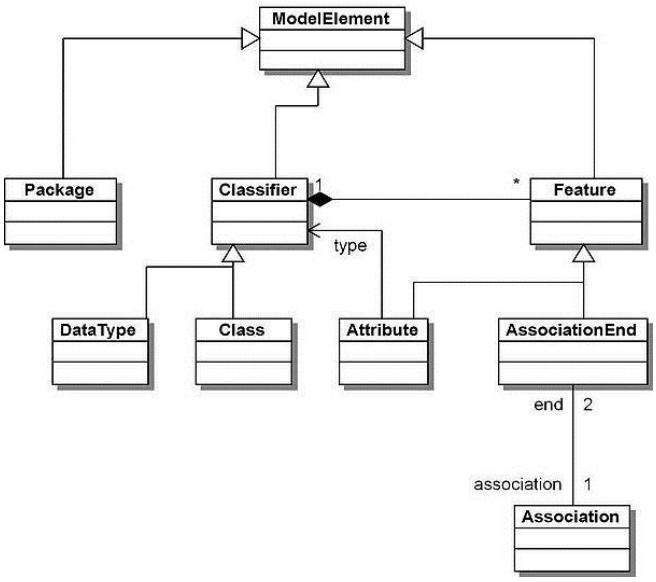

Figure 18 : Représentation du MOF

3.1.3.2 L'échange de modèles

Les modèles restent des entités théoriques abstraites fortement volatiles. Dans un souci de pérennité, il est nécessaire de leur fournir un support informatique permettant de réaliser des échanges.

3.1.3.2.1 XMI

XML Metadata Interchange (XMI) est un standard OMG permettant de décrire une instance du MOF sous forme textuelle au travers du langage XML[15]. Les métamodèles basés sur MOF sont transformés en DTD[16] XML et les modèles sont traduits en documents XML cohérents avec leurs DTD (Figure 19). XMI facilite ainsi les échanges des métadonnées UML entre les outils de modélisation en tant que procédé de sérialisation d'objets MOF.

[15] eXtensible Markup Language

[16] Document Type Definition

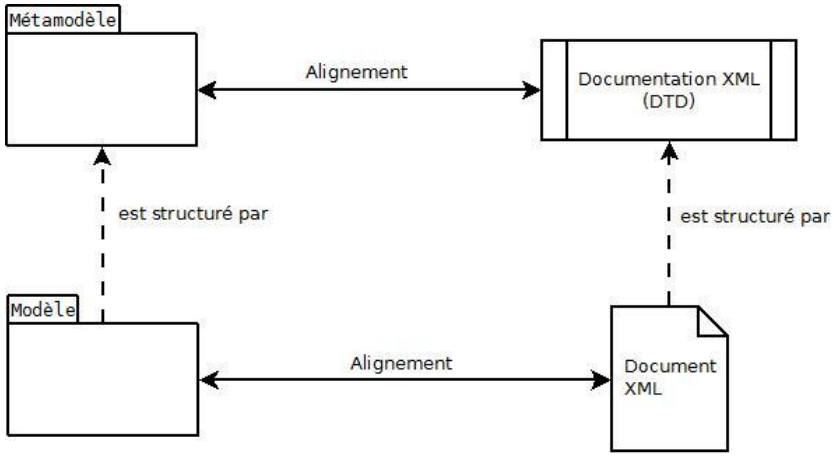

Figure 19 : Alignement entre métamodèle/modèle et DTD/document XML (Blanc, 2005)

XMI permet de représenter sous forme de document XML, les informations dont la structure est définie dans un métamodèle. La partie graphique des modèles (les diagrammes) n'étant pas définie au sein du métamodèle UML, elle n'est pas pas représentable en l'état avec XMI.

3.1.3.2.2 DI

Diagram Interchange (DI) est un standard spécifique de l'OMG ayant pour objectif de représenter au format XML les parties graphiques des modèles UML. L'approche utilisée s'appuie fortement sur XMI et nécessite la création d'un nouveau métamodèle au sein duquel la représentation des idiomes graphiques est traduite sous forme de métaclasses (Figure 20).

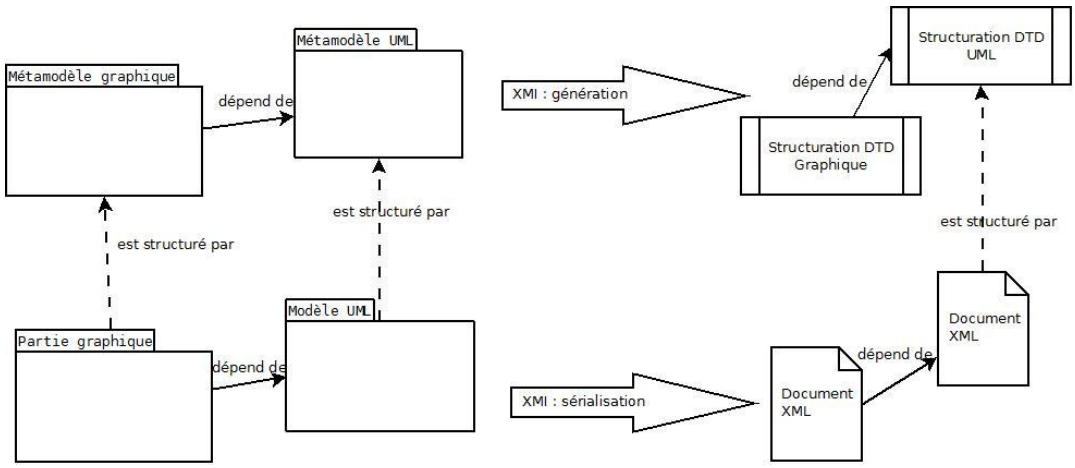

Figure 20 : Principe de fonctionnement de DI (Blanc, 2005)

Les standards XMI et DI rendent possibles la représentation des modèles au format XML et permettent ainsi de résoudre les problèmes d'interopérabilité associés. Ces éléments n'autorisent cependant pas le développement d'opérations sur les modèles.

3.1.3.3 La manipulation de modèles

La mise en place d'une opération sur les modèles nécessite la fourniture d'interfaces autorisant leurs manipulations. Celles-ci existent sous deux formes différentes : taylored et réflectives.

3.1.3.3.1 Les interfaces taylored

Les interfaces « taylored » sont adaptées à un unique type de modèle car elles offrent des opérations spécifiques permettant une navigation dans les modèles instances d'un unique métamodèle (Blanc, 2005). Par exemple, les interfaces taylored pour UML 2.0 offriront les opérations permettant d'obtenir les attributs d'une classe ou les connexions entre composants UML.

3.1.3.3.2 Les interfaces réflectives

Les interfaces réflectives sont utilisables sur tous les types de modèles car les opérations proposées sont totalement indépendantes de la structure des modèles. Elles permettent d'obtenir des informations sur le métamodèle et donc de connaître dynamiquement la structure du modèle. L'idée sous-jacente est de considérer tous les modèles comme des ensembles d'éléments (instances de métaclasse) réliés entre eux (Blanc, 2005).

3.1.3.3.3 Les outils

L'approche suivie par les standards MOF et JMI et le framework EMF (Eclipse Modeling Framework) est sensiblement la même. Des règles de génération d'interfaces taylored sont définies à partir de métamodèles et des interfaces réflectives autorisent la manipulation de tout type de modèle. Tous ces outils font en sorte que les interfaces taylored héritent des interfaces réflectives (Blanc, 2005).

L'OMG a proposé un format de représentation permettant la manipulation des modèles dans les langages de programmation orienté objet, le standard MOF vers Interface Definition Langage (MOF 2.0 IDL). JMI (Java Metadata Interface) est quant à lui un standard du JCP[17] qui définit un moyen de représenter les modèles sous forme d'objets Java. JMI couplé à MOF et UML améliore la sémantique des modèles dans un environnement J2EE (Java 2 Enterprise Edition). EMF est un framework Open Source fortement couplé à Eclipse qui est fondé sur les mêmes principes architecturaux que JMI et basé sur la version 2.0 du MOF.

[17] Java Community Process

3.1.3.4 Transformation de modèles : QVT

Dans les premières tentatives de standardisation de l'IDM par l'OMG, les transformations de modèles étaient évoquées mais en aucun cas explicitées. Dans le but de définir un standard pour la transformation de modèles, l'OMG a proposé le langage QVT composé d'une architecture à deux niveaux.

3.1.3.4.1 Partie déclarative

La partie déclarative de ce standard OMG est composé de deux langages différents :

- Le langage Relations offre la possibilité de spécifier des transformations comme des ensembles de relations entre modèles. Chaque relation contient un ensemble de motifs d'objets qui peuvent être reconnus dans les modèles source et créés dans les modèles cible.
- Le langage Core fournit une base pour la spécification de la sémantique de Relations. Cette sémantique est donnée comme une transformation Relations vers Core et peut être écrite avec le langage Relations.

3.1.3.4.2 Partie impérative

Le Langage Operational Mappings constitue la partie impérative de QVT. Sur le même principe que l'extension du langage UML à l'aide de profils, il permet l'extension du langage Relations avec des constructions impératives et OCL. Pour remplir ce besoin, OCL est étendu avec des notions familières à la programmation impérative : effets de bord et syntaxe concrète. La syntaxe de Operational Mappings fournit les constructions communes des langages impératifs (boucles, instructions conditionnelles, etc.).

3.1.3.4.3 Relation entre les langages

Le mécanisme boîte noire permet de brancher et d'exécuter du code externe au cours de l'exécution d'une transformation. Il autorise l'implémentation d'algorithmes complexes dans n'importe quel langage et la réutilisation des bibliothèques existantes. La relation entre les différents langages (Figure 21) permet d'obtenir le langage standard QVT prônée par l'OMG.

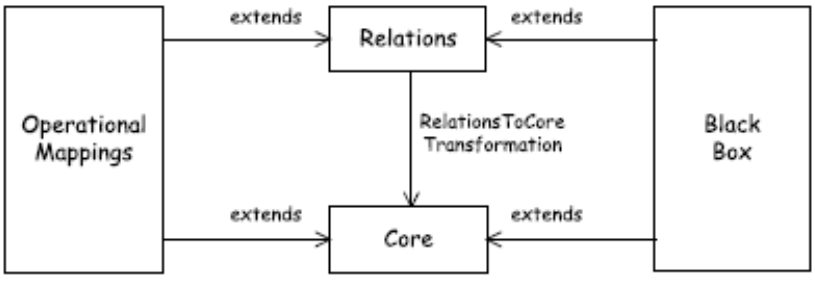

Figure 21 : Relation entre les langages dans le standard QVT

3.1.4 Les problèmes à résoudre

L'ingénierie dirigée par les modèles s'est installée dans le paysage des technologies innovantes depuis plus de 10 ans. Cette durée n'est pas encore suffisante pour l'arrivée à maturité des concepts et la délimitation stable du domaine d'application. Tous les jours de nouvelles alternatives sont proposées ce qui élargit le spectre des possibilités et remet parfois en question les outils de base. Le paradoxe de l'IDM se situe aujourd'hui entre la nécessité d'obtenir un retour sur investissement sur des projets et la mutation rapide nécessaire pour l'amélioration de cette démarche.

3.1.4.1 La dualité entre UML et les DSL

La dualité UML/Profil – DSL/MOF est exprimée au sein même de l'OMG. Hormis des cas bien identifiés où le choix d'une approche préférentielle est clairement établi, le choix reste ouvert. Le fait que Microsoft se rallie au standard UML à l'OMG et participe à sa consolidation est un facteur d'apaisement et de clarification en faveur de ce langage universel de modélisation. Cette dualité reste cependant au cœur des débats dans les différentes conférences sur l'IDM (Journées Neptune 2012).

3.1.4.1.1 Les limites de l'universalité

La technologie des profils UML permet d'étendre le standard UML pour des domaines particuliers. Les dérives auxquelles on assiste aujourd'hui sont nombreuses. La sémantique du standard UML est ignorée ou contredite à seule fin de produire un résultat graphique et outillé proche du rendu souhaité. Lorsque le domaine ciblé diverge par trop de la sémantique d'UML, la technique des profils UML doit être remplacée par celle de la construction d'un métamodèle spécifique (Desfray, 2009).

3.1.4.1.2 Débats entre UML et DSL

Chaque approche de modélisation dispose d'avantages non négligeables. La technologie des profils permet de bénéficier du standard UML en termes d'apprentissage de modèle, d'échange entre ateliers différents et de support outillé très répandu. Un faible effort de définition respectant la sémantique du standard UML produit un support outillé quasi immédiat.

Dans le cas de la construction d'un métamodèle dédié, toute liberté est offerte sur la sémantique du modèle défini. Celle-ci a cependant un coût représenté par la nécessaire mise en œuvre d'un outillage permettant d'exploiter ce métamodèle (éditeurs graphiques, générateurs, formats d'échange, etc.).

3.1.4.1.3 Les tendances

Les tensions historiques métamodèle/profils s'estompent face à une meilleure compréhension globale des deux approches et du panorama des outils. Le débat se concentre aujourd'hui sur la qualité technique des standards et outils, leur niveau de couverture et d'extensibilité, ainsi que leurs capacités de support des technologies MDA. Les organismes de standardisation, notamment l'OMG, réfléchissent à un renforcement des standards pour améliorer la qualité des métamodèles, assurer un meilleur alignement des standards et augmenter les fonctionnalités des métamodèles.

Des besoins sans cesse renouvelés et des exigences de précision croissantes militent pour l'apparition de plus de langages spécifiques (profils UML ou métamodèles spécifiques). La richesse de ces DSL augmente le pouvoir d'expression des modèles et par extension les capacités de l'IDM. L'enjeu majeur concerne la capacité d'unifier un ensemble de DSL pour offrir une couverture complète pour les types de systèmes ou de projets les plus fréquents (Desfray, 2009).

3.1.4.2 Définition et validation des modèles

Au sein de l'IDM, des difficultés apparaissent au niveau de la construction des modèles. Cette opération nécessite en effet l'intégration de plusieurs sciences complémentaires (ingénierie des besoins, acquisition des connaissances et des sciences cognitives) pour s'assurer de la complétude de la prise en compte des besoins exprimés initialement.

Un autre problème intervient au niveau de la validation des modèles. La définition du modèle est le point de départ d'un projet et doit aboutir à la prise de décision concernant le développement d'un logiciel. La modélisation, tant dans sa représentation syntaxique que dans sa sémantique, doit ainsi être de qualité. Celle-ci apparaît à ce jour très difficile à mesurer en l'absence de métriques définies.

3.1.4.3 Définition des métamodèles

Le choix du métamodèle est stratégique dans la logique de modélisation. Il influence fortement la perception des modèles réalisés et doit être en adéquation avec la finalité de la modélisation et la complexité du système à réaliser. Le défi des prochaines années correspond à la définition de métamodèles riches et rigoureux offrant des mécanismes de validation et de raisonnement en adéquation avec la nature et la complexité des systèmes à modéliser.

3.1.4.4 Hétérogénéité des modèles et métamodèles

On observe à ce jour une prolifération de métamodèles due à la nécessité de décrire différentes facettes d'un même système, différents niveaux d'abstraction ou différentes technologies. Le challenge des prochaines années consiste en la formalisation des interactions des modèles via la formalisation de leur métamodèle, dans l'interopérabilité des modèles et en général dans la gestion des métamodèles et des modèles (Favre, Estublier, & Blay-Fornarino, L'ingénierie dirigée par les modèles : au-delà du MDA, 2006).

3.1.4.5 Coûts de production et d'exploitation des modèles et métamodèles

Au bout de ces 10 années de mise en œuvre de l'IDM, l'utilité des modèles semble actée. Le débat reste cependant ouvert sur le coût engendré par leur définition et leur exploitation. L'IDM doit ainsi encore prouver sa compétitivité. Les nombreux projets mis en œuvre (IBM, Thales, etc.) laissent tout de même entrevoir de formidables perspectives de rentabilité sur les projets informatiques.

3.2 Les différentes approches de l'IDM

L'approche générique de l'ingénierie des modèles a déjà convaincu les acteurs industriels de son bien-fondé, en témoigne le ralliement non seulement de l'OMG mais aussi des poids lourds que constituent Microsoft et IBM (Favre, Estublier, & Blay-Fornarino, L'ingénierie dirigée par les modèles : au-delà du MDA, 2006). Ces différentes entités disposent cependant d'une vision différente de la mise en œuvre de cette approche (Figure 22).

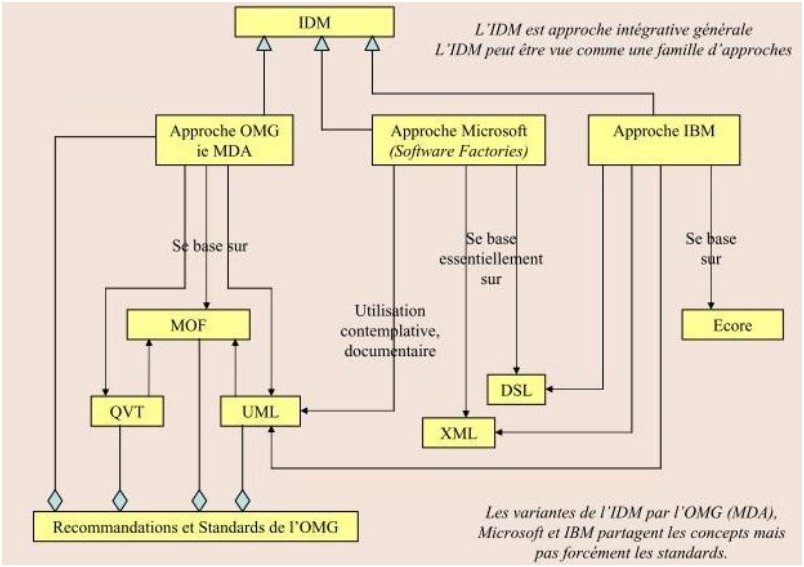

Figure 22 : Ensemble des approches de l'IDM (Rousse, 2007)

3.2.1 L'approche Microsoft (Software Factories)

Les langages de domaines – ou langages métiers, langages spécialisés - sont de petite taille, facilement manipulables, transformables, combinables… Ils sont à la base de l'automatisation de l'IDM chez Microsoft. L'approche Software Factories (Figure 23) est mis en œuvre au sein de la plateforme de génie logiciel de Microsoft (Visual Studio).

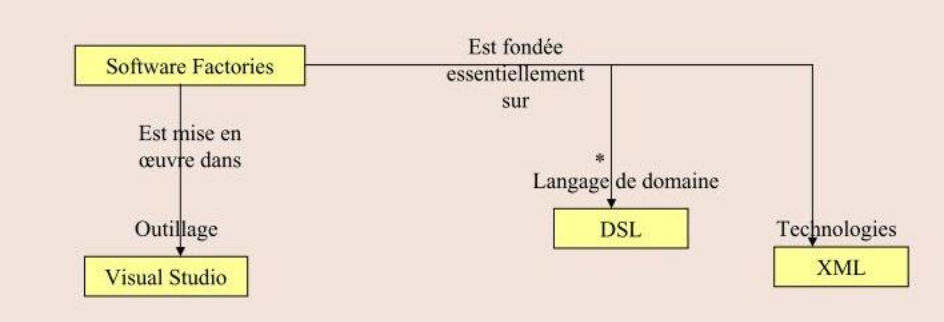

Figure 23 : Approche IDM de Microsoft (Rousse, 2007)

3.2.2 L'approche IBM

Au sein d'IBM, l'IDM doit permettre de passer de la programmation d'application à la description d'application (Journées Neptune 2012). Selon le manifeste publié sur le sujet (IBM, 2004), les 3 axes de l'IDM sont :

- Les standards ouverts : UML, XML, et autres standards.
- L'automatisation : possibilité de traitement automatique des modèles (tissage, vérification, transformation, etc.)
- La représentation directe : DSL, langages précis et outillés (éditeurs, générateurs, vérificateurs, etc.).

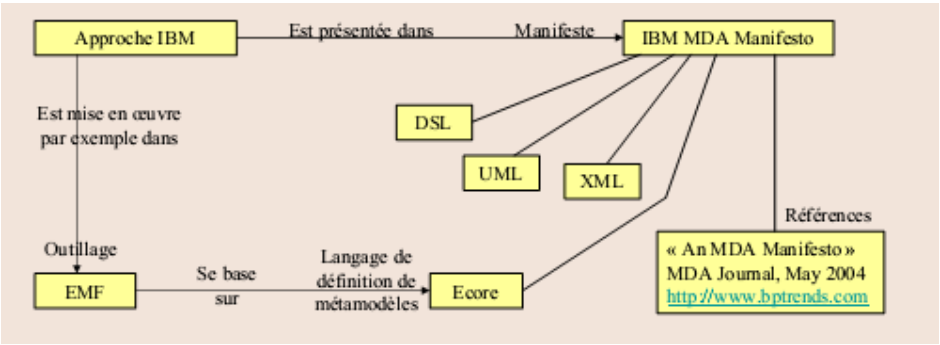

Figure 24 : Aperçu de l'approche IBM de l'IDM (Rousse, 2007)

3.2.3 La démarche ADM[18]

L'objectif de cette démarche concerne la production de standards pour la modernisation de vieux programmes informatiques grâce à des méthodes de rétro-ingénierie fondées sur les modèles. La mise en œuvre standardisée par l'OMG sous la terminologie ADM permet notamment de répondre au besoin énorme de rénovations d'applications COBOL sur le marché. Le projet REMICS[19] est un parfait exemple de l'utilité d'une telle démarche (Eveillard, Youbi, & Henry, 2009).

Beaucoup considère d'ailleurs que la notion de métamodèle propre à l'IDM permettra de révolutionner de domaine de la rétro-ingénierie (Favre & Musset, Rétro-ingénierie dirigée par les métamodèles, 2005).

3.2.4 La démarche MDA

La démarche MDA (Model Driven Architecture), proposée et soutenue par l'OMG (Object Management Group), correspond à ce jour à la réalisation la plus avancée et visible dans le domaine de l'IDM. Elle peut se définir comme la réalisation des principes de l'ingénierie des modèles autour d'un ensemble de standards comme le MOF, XMI, OCL, UML, CWM, QVT, etc. (Bézivin, Sur les principes de base de l'ingénierie des modèles, 2004). La démarche MDA correspond aux principes suivants :

- tout artefact de développement logiciel est un modèle d'un aspect d'un système,
- un modèle est décrit par un métamodèle qui est un langage spécialisé pour l'aspect considéré,
- la construction d'un système informatique revient à effectuer un tissage d'aspects.

[18] Architecture-Driven Modernization

[19] REuse and Migration of legacy applications to interoperable Cloud Services

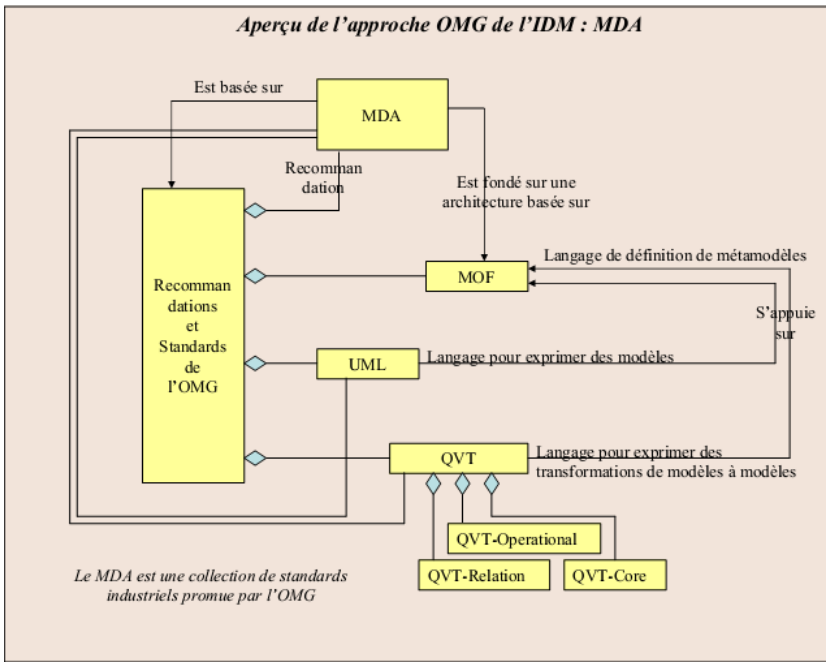

Figure 25 : L'approche MDA (Rousse, 2007)

MDA se veut une approche méthodologique qui permet une optimisation des investissements liés aux développements d'applications informatiques (Initiative MDA de l'OMG, 2001). Cette démarche reprend la philosophie générale de l'IDM et la décline pour la séparation des spécifications fonctionnelles de son implémentation sur une plateforme technologique.

3.2.4.1 Composants

La démarche MDA est représentée par un logo (Figure 26) qui met en valeur les différentes couches :

- outils de base MDA (UML, MOF et CWM),
- technologies middleware (CORBA, XMI, etc.) nécessaires pour faire communiquer les applications,
- services d'infrastructure de systèmes à objets distribués préconisés par l'OMG (niveau d'abstraction supplémentaire vis-à-vis du middleware)
- domaines étudiés par les Task Forces de l'OMG (profils UML). La couche spécifique à chaque domaine métier se base sur les profils UML et permet de disposer des frameworks spécifiques au domaine d'utilisation de l'application.

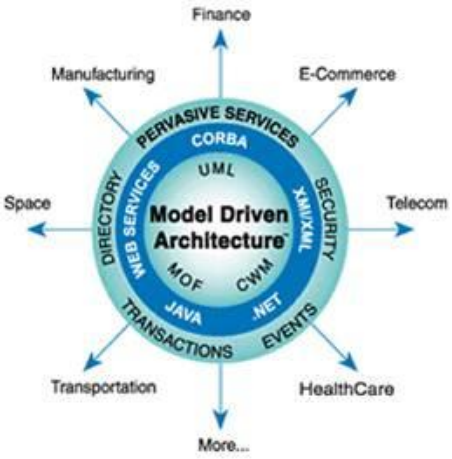

Figure 26 : Logo MDA

3.2.4.2 Architecture générale

L'OMG a défini un cadre général d'intégration de tous les métamodèles autour du MOF. Ce cadre repose sur une architecture à quatre niveaux (Figure 27) où chaque niveau entretient une relation d'instanciation avec son niveau supérieur strict :

- M0 : système réel, système modélisé,
- M1 : modèle du système réel défini dans un certain langage,
- M2 : méta-modèle définissant ce langage,
- M3 : méta-méta-modèle définissant le méta-modèle.

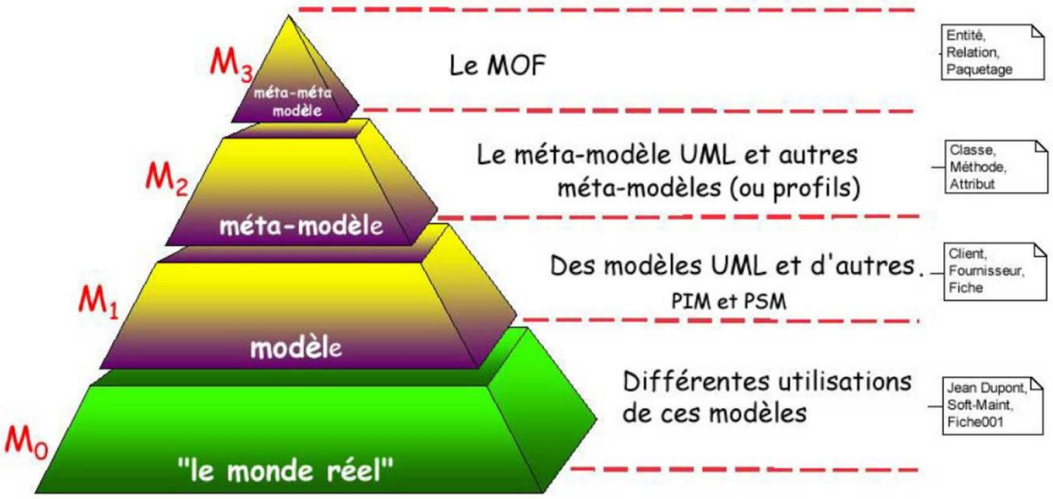

Figure 27 : Architecture MDA à 4 niveaux

L'architecture à quatre niveaux est présente dans toutes les visions de la mise en oeuvre de l'IDM. Au sein de MDA, cette architecture n'est cependant pas structurante au contraire de l'ensemble des métamodèles définis.

3.2.4.3 Le cycle de développement

Le besoin d'élaboration de modèles pérennes et indépendants des détails techniques de la plateforme d'exécution impose l'utilisation de modèles aux différentes phases du cycle de développement d'une application. Ce cloisonnement des problématiques nécessite l'utilisation d'un cycle de développement en Y.

La démarche 2TUP[20] (Figure 28) est parfaitement indiquée pour répondre à ce genre de problématique. Elle se distingue par les caractéristiques suivantes :

- Itératif : Le logiciel nécessite une compréhension progressive du problème à travers des raffinements successifs. Une solution effective est développée de façon incrémentale par des itérations multiples.
- Piloté par les risques : les causes majeures d'échec d'un projet logiciel doivent être écartées en priorité.
- Centré sur l'architecture : le choix de l'architecture logicielle est effectué lors des premières phases de développement du logiciel. La conception des composants du produit est basée sur ce choix.
- Conduit par les cas d'utilisation : le processus est orienté par les besoins utilisateurs présentés par des cas d'utilisation.

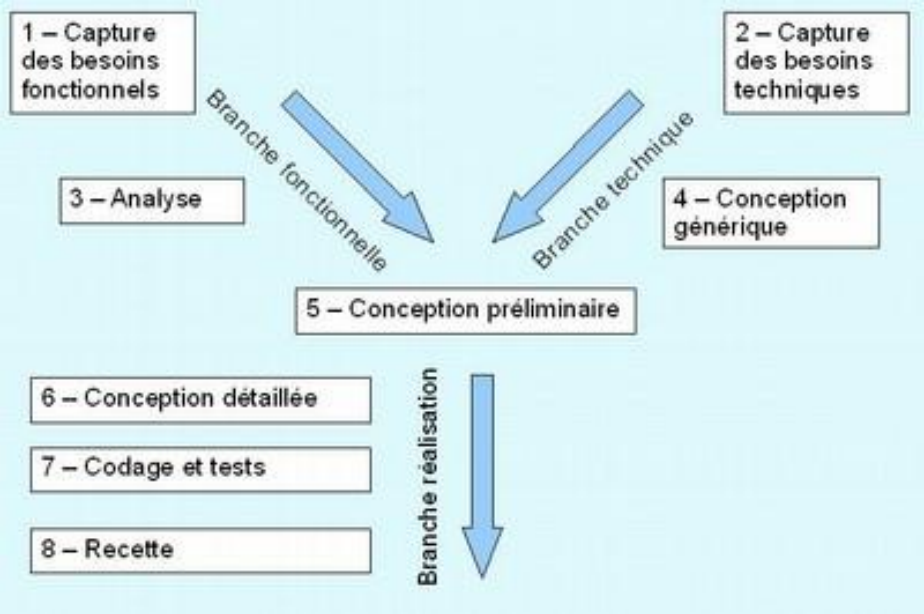

Figure 28 : Etapes de la démarche 2TUP

La démarche 2TUP apporte ainsi une réponse aux contraintes de changement continuel imposées aux systèmes d'information de l'entreprise et nécessite la mise en place de modèles à tous les niveaux (Roques & Vallée, UML 2 en action : de l'analyse des besoins à la conception J2EE, 2004).

[20] Two Tracks Unified Process

3.2.4.4 Les différents modèles

Au sein de la branche fonctionnelle, le processus spécifié (Figure 29) nécessite tout d'abord la production d'un modèle de capture de besoin (CIM[21]). Des premiers modèles PIM sont ensuite mis en œuvre pour décrire le domaine dans le cadre de l'analyse. Les raffinements successifs permettent ensuite d'obtenir des modèles de conception. Au sein de la branche technique, les modèles PDM[22] concernent le choix de la plateforme d'implémentation et de l'architecture associée à l'application. Enfin, le croisement de ces deux branches permet la création des modèles de code (PSM[23]).

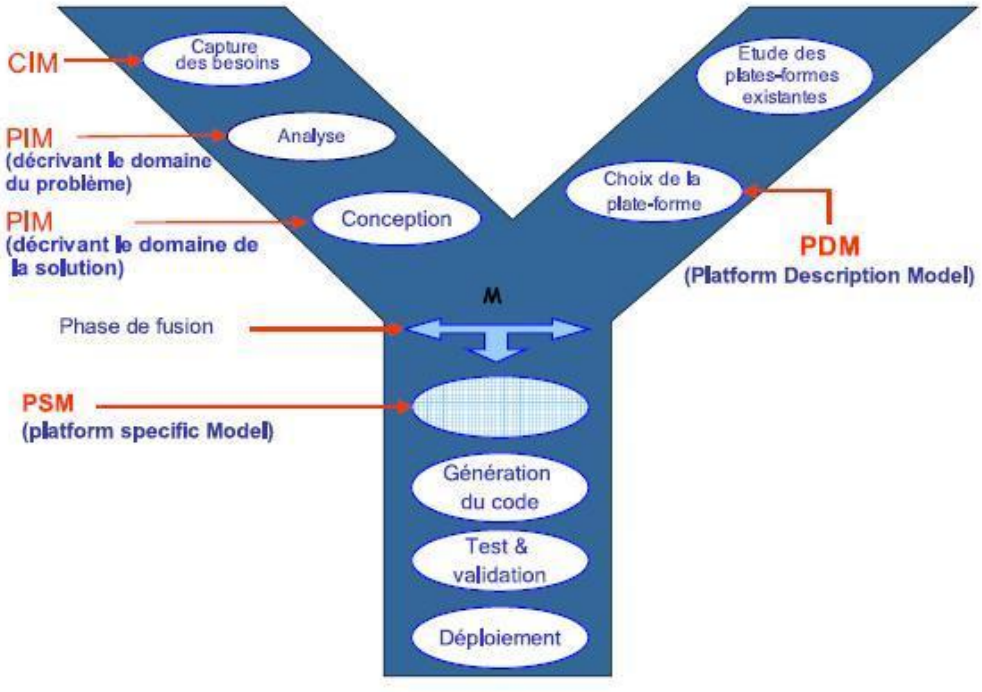

Figure 29 : Le cycle de développement en Y

3.2.4.4.1 Modèle d'exigences (CIM)

Le CIM (Computation Independent Model), parfois appelé Business Model ou Domain Model, décrit les exigences du système et la situation dans laquelle le système sera utilisé. La mise en œuvre de ce modèle est d'une importance capitale car il permet d'exprimer des liens de traçabilité avec les modèles construits dans les autres phases de développement. Il s'agit ainsi de créer un lien durable avec le besoin des clients (Blanc, 2005).

[21] Computation Independant Model
[22] Platform Dependant Model
[23] Platform Specific Model

Ce modèle d'exigences n'exhibe pas les détails de la structure du système et reste indépendant de son implémentation. Il correspond en quelque sorte à la modélisation du système d'information sans parler encore de système informatique. Il place le système dans son contexte opérationnel et permet de représenter ce que le système devrait réellement assurer comme services. Les modèles d'exigences peuvent même être considérés comme des éléments contractuels, destinés à servir de référence lorsqu'on voudra s'assurer qu'une application est conforme aux demandes du client. En UML, on peut représenter un modèle d'exigences avec un diagramme de cas d'utilisation qui dispose de la faculté de définir l'ensemble des acteurs et des cas d'utilisation sans en détailler le fonctionnement.

3.2.4.4.2 Modèle d'analyse et de conception (PIM)

Le PIM (Platform Independant Model) est un modèle de haut niveau d'abstraction représentant la logique métier. Il reste totalement indépendant de la technique et de la technologie ce qui lui confère une grande portabilité. Lorsque le modèle d'exigences est réalisé, le travail d'analyse et de conception peut commencer. Il s'agit de faire une représentation informatique des exigences et donc se concentrer sur :

- la représentation de l'architecture modulaire de l'application,
- des connections entre les modules,
- le contenu des modules,
- les tâches réalisées par les modules.

UML est préconisé par l'approche MDA comme étant le langage à utiliser pour réaliser ce type de modèles. La modélisation des contraintes est exprimée à l'aide du langage OCL et le dynamisme des modèles est transcrit grâce au langage AS. Pour réaliser ce travail d'analyse et de conception, différents diagrammes UML sont utilisés :

- diagramme de composants pour l'architecture,
- diagramme de classes pour les aspects statiques,
- diagramme de séquences ou d'activités pour les aspects dynamiques.

3.2.4.4.3 Modèle de code ou de conception concrète (PSM)

Contrairement au modèle d'analyse ou de conception, le modèle de code (PSM) est lié à une plate-forme d'exécution et sert principalement à faciliter la génération de code. Fournissant plus ou moins de détails selon ses objectifs, le PSM peut être de multiples niveaux. Le premier, issu de la transformation d'un PIM, se représente par un schéma UML spécifique à une plate-forme. Les autres PSM sont obtenus par transformations successives jusqu'à l'obtention du code dans un langage spécifique (Java, C++, C#, etc.). Un PSM d'implémentation fournit, dans la mesure du possible, un large éventail d'informations relatives à une mise en oeuvre efficace et automatisée.

Les PSM peuvent être obtenus par application de profils UML, c'est-à-dire l'adaptation d'UML à un domaine particulier, ou par l'utilisation de modèles de plates-formes (PDM) lors de la transformation du PIM. Fortement liés à une plate-forme d'exécution, les modèles de code n'ont donc pas pour vocation d'être pérennes.

3.2.4.4.4 La plateforme d'exécution (PDM)

Le modèle PDM (Platform Description Model) correspond à un modèle de transformation du PIM vers un modèle d'implémentation sur une plate-forme technique donnée en tenant compte des caractéristiques architecturales désirées (Kadima, 2005). Deux étapes sont nécessaires pour la mise en place de ces éléments d'architecture technique et technologique :

- choix de la plateforme et obtention de son modèle du point de vue de l'utilisateur,
- modélisation du filtre qui va permettre de rendre possible la transformation.

3.2.4.5 Transformation de modèles

Une transformation de modèles correspond à la génération automatique d'un modèle cible à partir d'un modèle source conformément à une définition de transformation. Au sein de la démarche MDA, le plus gros défi à relever d'un point de vue technique se situe sur cette problématique. Les transformations de modèles portent en effet l'intelligence du processus méthodologique de construction d'application et les règles de qualité de développement d'applications.

3.2.4.5.1 Les différentes approches

MDA préconise de modéliser les transformations de modèles elles-mêmes. Une transformation de modèles peut être considérée comme une application et il est donc naturel de modéliser ses exigences, son analyse et sa conception et ses modèles de code afin de générer automatiquement le code de la transformation. Cependant, il ne faut pas occulter l'existence d'autres types de transformations de modèles tels que l'approche par programmation ou par template.

3.2.4.5.1.1 Approche par programmation

Cette approche consiste à programmer les transformations de modèles en utilisant les langages de programmation orientée objet et les interfaces de manipulations de modèles. Il s'agit d'un mécanisme compliqué mais qui demeure le plus utilisé du fait de son support par les outils de développement.

3.2.4.5.1.2 Approche par template

Cette approche particulière de la transformation consiste à annoter (tagger) manuellement un PIM puis, à convertir ce PIM en PSM à l'aide des règles de transformation issues des profils UML ou à l'aide des patrons de conception. Des traces de cette transformation sont conservées ce qui permet éventuellement de revenir du PSM vers le PIM. Une correspondance entre les éléments source et cible est maintenue.

Particulièrement bien outillée, l'approche par template est très utilisée pour réaliser certaines transformations de modèles. Certains langages de définition de template, comme le langage UML, sont très aboutis et permettent de définir facilement des transformations de modèles UML. Cette approche apporte par exemple un gain significatif pour la génération de texte à partir de modèle. Pour d'autres besoins de transformation, l'utilisation reste plus discutable (Blanc, 2005).

3.2.4.5.1.3 Approche par modélisation

L'approche par modélisation consiste à appliquer les concepts de l'ingénierie des modèles aux transformations des modèles et par conséquent de les modéliser. L'objectif est de rendre les transformations de modèles pérennes et productives et de les définir en fonction des plates-formes d'exécution.

Dans ce cas de figure, les transformations s'effectuent grâce aux méta-modèles. Les profils UML sont exclus de cette transformation car les méta-modèles ne peuvent être exprimés dans d'autres langages que l'UML. Cette approche de la transformation de modèles comporte deux étapes :

- spécifier les règles de transformations décrivant la correspondance entre le langage source et le langage cible,
- appliquer ces règles au PIM pour produire le PSM.

3.2.4.5.2 Les différentes transformations

Au sein de la démarche MDA, la mise en place et le maintien de liens de traçabilité entre les modèles constituent une problématique importante. L'exécution des différentes transformations de modèles (Figure 30) permet d'établir ces liens et ainsi résoudre cette préoccupation de façon automatique.

Les transformations verticales de modèles permettent d'avancer dans la réalisation de l'application (passage d'un PIM à un PSM).

Le raffinement, en tant que transformation horizontale, implique de rester au même niveau de modélisation en apportant une information complémentaire sur le modèle. Comme évoqué précédemment, les raffinements successifs du PSM permettent d'obtenir un modèle de code

Enfin, la rétro-ingénierie permet de revenir au niveau supérieur de la démarche MDA. Ce type de transformation est très utile pour la rétro-ingénierie d'applications car il permet de générer les modèles associés à celle-ci.

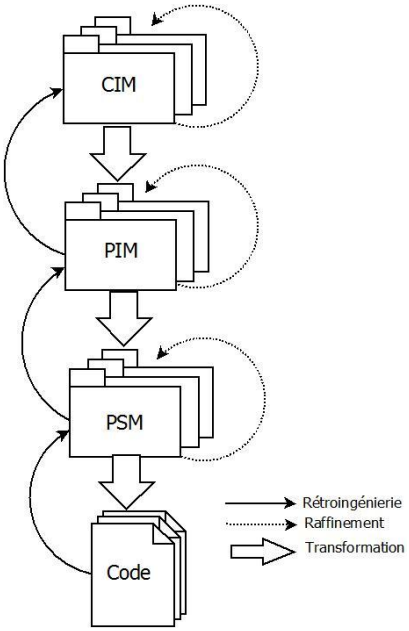

Figure 30 : Les transformations des modèles MDA

3.2.4.5.2.1 De CIM vers PIM

A ce jour, aucune méthode n'a été formalisée par la démarche MDA pour le passage automatique entre le CIM et le PIM. La transformation manuelle restera nécessaire tant qu'aucun méta-modèle n'aura été défini pour la réalisation du CIM. Les frémissements perceptibles sur le sujet (Kherraf, 2011) autorisent cependant l'espoir de disposer de ce genre de transformations dans les années à venir.

3.2.4.5.2.2 De PIM vers PIM

Les transformations de PIM vers PIM sont utilisées pour enrichir, filtrer ou spécialiser les informations des modèles sans rajouter aucune information liée à la plateforme. Ce raffinement de modèle permet par exemple de masquer des éléments afin de s'abstraire des détails fonctionnels ou de passer du modèle d'analyse à celui de conception.

3.2.4.5.2.3 De PIM vers PSM

Une fois le PIM suffisamment raffiné pour pouvoir être spécialisé vers une plate-forme donnée, il peut alors être transformé en PSM. Cette opération consiste à ajouter des informations propres à une plateforme technique. Les caractéristiques de la transformation sont stockées soit dans le modèle PDM, au sein d'un profil UML ou au sein du MOF. Le résultat de cette transformation permet les premiers modèles dépendants de la technologie et de l'architecture technique.

3.2.4.5.2.4 De PSM vers PSM

La transformation du PIM vers le PSM évoqué préalablement n'est pas toujours suffisante pour arriver à obtenir la génération de code de l'application. Le raffinement du PSM est alors nécessaire et nécessite des transformations successives de formalismes intermédiaires. Ce genre d'opération s'effectue lors des phases de déploiement, d'optimisation ou de reconfiguration.

3.2.4.5.2.5 De PSM vers PIM

Ce type de transformation est utilisé dans le but de revenir à un modèle indépendant de plateforme (PIM) à partir d'un modèle spécifique (PSM) ou éventuellement du code. Cette opération de rétro ingénierie (reverse engineering) demeure assez complexe à réaliser et parfois difficilement automatisable. Ces transformations sont néanmoins nécessaires pour permettre l'intégration d'applications existantes dans le processus MDA.

3.2.4.6 L'outillage disponible

L'intérêt grandissant de l'industrie logiciel pour la démarche MDA au cours de ces dix dernières années a engendré l'arrivée d'un nombre conséquent d'outils intégrant de façon plus ou moins rigoureuse les spécifications de l'OMG. Cette liste de taille importante pose la problématique du choix de l'outil correspondant le mieux au besoin et nécessite au préalable de répondre à ces différentes questions :

- Quels sont les interactions avec mon modèle (graphique, textuel ou guidé par un outil) ?
- Quel est le niveau de modélisation souhaité ?
- Quel type de langage est le plus approprié pour la modélisation de mon projet (UML ou DSL) ?
- Quel est mon objectif dans le domaine de la génération de code de l'application (complète ou partielle) ?

Outils de transformation de modèles	Actifsource (http://www.actifsource.com/
	AndroMDA (http://www.andromda.org/)
	Eclipse ATL (http://www.eclipse.org/atl/)
	Eclipse QVTO (http://wiki.eclipse.org/M2M/QVTO)
	Itemis/Eclipse xpand/xtend (http://wiki.eclipse.org/Xpand)
	Mia-Software (http://www.mia-software.com/)
	Obeo/Eclipse Acceleo (http://www.eclipse.org/acceleo/)
Outils de génération d'applications	Artisan Software Tools Artisan Studio (http://www.artisansoftwaretools.com/)
	Aspectize (http://aspectize.com/)
	BluAge (http://www.bluage.com/)
	Jaxio Celerio (http://www.jaxio.com/) Jaxio SpringFuse (http://www.springfuse.com/)
	Mendix (http://www.mendix.com/)
	Outsystems Agile Platform (http://www.outsystems.com/agile-platform)
	SkyWay Software SkyWay Builder (http://www.skywayperspectives.org/)
	Sodius MDWorkbench (http://www.mdworkbench.com/ & http://www.sodius.com/)
	SoftFluent CodeFluent Entities (http://www.codefluententities.com/ & http://www.softfluent.com/)
	SpringSource SpringRoo (http://www.springsource.org/roo)
Outils de définition de langage	EMFText (http://www.emftext.org/)
	Itemis/Eclipse TMF Xtext (http://xtext.itemis.com/)
	Jetbrains MPS (http://www.jetbrains.com/mps/)
	Spoofax (http://strategoxt.org/Spoofax/WebHome)
Outils de définition de domaines	Isomeris ABSE & Atomweaver (http://www.atomweaver.com/ , http://www.abse.info/ & http://www.isomeris.com/)
	MetaCase MetaEdit+ (http://www.metacase.com/)
	Obeo Designer (http://www.obeo.fr/pages/obeo-designer/en)
Outils de conceptions d'interface utilisateur	Eclipse E4/XWT (http://wiki.eclipse.org/E4/XWT)
	redView (http://redview.org/)
	wazaabi (http://wazaabi.org/)
Outils de modernisation	eclipse MoDisco (http://www.eclipse.org/MoDisco/)
	Mia-Sotware (http://www.mia-software.com/)
	Obeo Agility (http://www.obeo.fr/pages/agility/en)
Outils CASE[24] avec des possibilités d'IDM	ModelioSoft Modelio (http://www.modeliosoft.com/)
	NoMagic MagicDraw (http://www.magicdraw.com/)
	Sparx System Enterprise Architect (http://www.sparxsystems.com/products/ea/)
Outil basé sur la technologie XSLT	Xomega (http://www.xomega.net/)

Tableau 2 : Outillage MDA (Seignard, 2012)

[24] Computer Aided Software Engineering : ensemble de programmes informatiques permettant eux-mêmes de produire des programmes de manière industrielle

En 2012, les logiciels les plus populaires sur le marché s'intéressent à la construction complète d'une application. Ils ne permettent cependant pas toujours de répondre correctement à l'ensemble des besoins exprimés par les industriels. La liste de ces différents outils laisse en effet transparaître des utilisations très différentes de la démarche MDA dans le secteur du développement de logiciel.

Les différents besoins dans les domaines de la transformation de modèles, la définition de langage ou de domaines, la conception d'interfaces utilisateur ou la modernisation d'applications ont entrainé le déploiement de nombreux outils. Ces logiciels tentent d'apporter une réponse plus adéquate aux industriels dans chacun de ces domaines particuliers de la démarche MDA. Cette logique permet en outre de tenir compte des besoins spécifiques à chacun des cœurs de métier. Par exemple dans le domaine de l'aérospatiale, les contraintes sont totalement différentes de celles présentes dans le domaine du commerce électronique.

Enfin, des travaux de recherche dans différents domaines ont engendré l'arrivée de fonctionnalités MDA dans les ateliers de génie logiciel (AGL) du marché. Par exemple, les travaux réalisés sur la définition d'un langage de métamodélisation (Kermeta) sont désormais disponibles parmi les modules de la plateforme Eclipse[25]. La combinaison de différents éléments permet donc d'envisager de mettre en œuvre une version industrialisée et personnalisée de l'approche MDA au sein d'un AGL existant.

A ce jour, les possibilités d'implémentation de la démarche MDA sont très nombreuses ce qui pose la problématique de la cohérence du choix de l'outillage avec les pratiques souhaitées dans le développement. On ne parle alors plus de la démarche MDA prônée par l'OMG mais de multiples déclinaisons de celle-ci.

[25] Eclipse est un projet de la Fondation Eclipse visant à développer un environnement de développement intégré libre, extensible, universel et polyvalent.

Chapitre 4 : La démarche MDA dans un contexte industriel

Dans le chapitre précédent, nous avons introduit les notions théoriques de l'IDM et de la démarche MDA. Nous allons dorénavant présenter la mise en œuvre de ce nouveau paradigme dans un contexte industriel au travers de l'exemple d'une société du domaine informatique. Le groupe GFI[26] a été choisi car il correspond à un retour naturel vers mon contexte habituel. L'expérience au sein de cette société permet de disposer d'un certain nombre d'entrées pour étudier les pratiques industrielles « maison ».

Dans le cadre de son industrialisation, le groupe Gfi a développé des centres de services afin de garantir aux clients des gains de productivité significatifs. L'organisation des prestations fournies en centres de services permet en effet de mieux maîtriser les coûts, de gérer au plus près les évolutions et d'améliorer la réactivité du groupe. Les centres de services de Gfi se répartissent en centres de proximité (études et production), centres nearshore[27] et centres offshore[28]. Le Centre de Services National (CSN) basé à Lille est par exemple utilisé pour tous les projets ayant une partie de back office significative (supérieure à 200 000 euros).

Dans le domaine concurrentiel des sociétés informatiques, la recherche de l'amélioration de la productivité se doit d'être une préoccupation continuelle. On observe en outre une croissance importante :

- du volume de données et du périmètre des applications qui entraîne une augmentation significative des spécifications et du code,
- de l'hétérogénéité des langages de développement (Java, C#, Python, PHP, etc.),

[26] Groupe Informatique Français

[27] Externalisation qui s'effectue vers une destination proche géographiquement et/ou culturellement, tout en maintenant les avantages du modèle offshore.

[28] Transfert à l'étranger du développement d'applications informatiques par une entreprise.

- de l'hétérogénéité des architectures techniques (Hibernate, architecture orienté services, Smartphone, tablette tactile, etc.), domaine dans lequel la veille technologique est indispensable,
- de l'hétérogénéité des paradigmes de développement :
  - piloté par le code,
  - piloté par la documentation,
  - piloté par les tests,
  - piloté par les modèles.

Ces différentes problématiques poussent à se questionner sur le bien-fondé de la méthodologie, des outils et des ressources utilisés. Une réflexion autour d'une modification de l'organisation des projets de développement d'application au sein du groupe a ainsi été lancée à partir de 2006 au sein de CSN.

4.1 Sélection de la démarche de développement

Au cours de l'étude réalisée par le CSN, le développement piloté par les modèles s'est rapidement imposé comme le paradigme de développement répondant le mieux aux besoins de la société. La démarche MDA, représentant ce paradigme, représente en effet la meilleure approche pour la qualité du logiciel (code valide fonctionnellement et techniquement) et pour la productivité. Elle permet de plus d'éviter les écueils de cassure fonctionnelle (Figure 31) et de multiplicité des référentiels présentés (Figure 32).

La présence d'une cassure fonctionnelle entre le concepteur et le développeur représente une problématique importante dans les projets informatiques. La perte d'informations est amplifiée par celle-ci et engendre d'éventuels retours entre la conception et le développement chronophages pour le projet. Ce travail d'interprétation de la conception pourrait être automatisé.

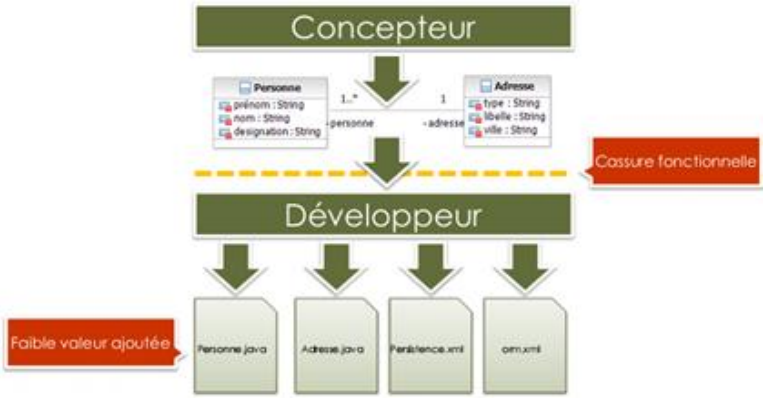

Figure 31 : Cassure fonctionnelle entre le concepteur et le développeur

La multiplicité des référentiels (Figure 32) est aussi à proscrire des projets car il devient alors complexe de déterminer quel modèle est correct et utilisable pour la suite du développement. Le travail d'unification des modèles représente ainsi un chantier important pour la bonne utilisation d'une démarche quel qu'elle soit.

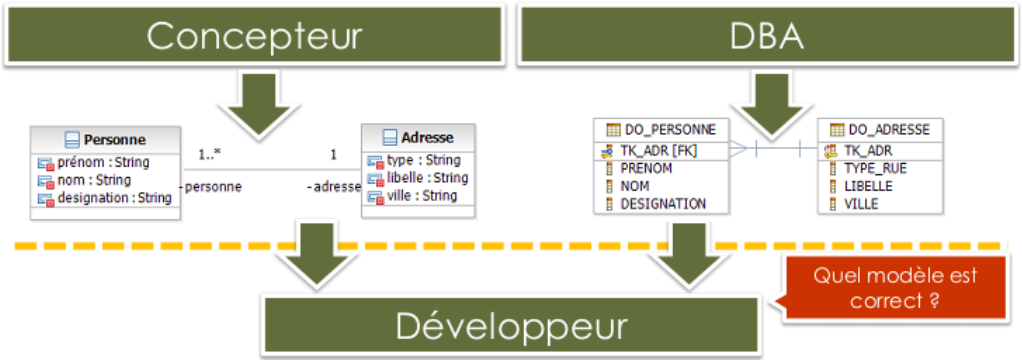

Figure 32 : Risques associés à la multiplicité des référentiels

La mise en œuvre de cette démarche nécessite de disposer d'un outillage à même de répondre aux besoins exprimés.

4.2 Sélection de l'outillage

Afin de valider le choix de la démarche MDA, une étude des différents outils présents sur le marché en 2006 a été réalisée au sein du CSN de Lille. Pour être implémenté au sein de cette structure et par extension au sein du groupe GFI, ceux-ci doivent répondre à un certain nombre de critères jugés indispensables pour la réussite de cette nouvelle approche du développement logiciel :

- basé sur UML2.1 au format EMF[29]. Les langages dédiés (DSL) sont proscrits par le CSN pour des raisons d'apprentissage pour des développements éventuellement effectués en offshore,
- performant : la génération rapide de l'application permet de ne pas bouleverser le cycle de développement,
- léger : prise en main rapide des outils de modélisation et des concepts MDA pour les nouveaux utilisateurs,
- permettant la mise au point d'un générateur adapté à un nouveau projet en moins de 10 jours,

[29] Le projet Eclipse Modeling Framework (EMF) est un framework de modélisation, une infrastructure de génération de code et des applications basées sur des modèles de données structurées (Eclipse Modeling Framework Project (EMF), 2012).

- supportant les différents types de transformation :
    - Model to Model (M2M) : raffinement de modèles ou passage d'un modèle indépendant de toute technologie (PIM) à un modèle dépendant (PSM),
    - Model to Text (M2T) : génération de code ou documentaire (Word, PDF et HTML),
    - Text to Model (T2M) : rétro ingénierie d'applications existantes.
    - Model To Documentation (M2D) : génération documentaire intégrant l'ensemble des modèles effectués.
- prix de licence abordable.

Les paragraphes suivants représentent une synthèse de cette étude ayant permis d'aboutir au choix d'un outillage adéquat. Les éléments sont organisés en fonction de la licence de l'outil en question.

4.2.1 Les outils open source

Le projet AndroMDA (XMI 2, configuration XML, properties) est rapidement apparu comme un outil non viable sur le long terme du fait d'un manque évident de synergie au sein du projet.

Le projet Acceleo (licence double Commerciale/Open Source) est basé sur EMF 2.1 mais utilise des métamodèles et des templates au format propriétaire. Ces derniers se révèlent assez peu performant et posent des problèmes d'apprentissage des équipes. Au sein de cet outillage, le choix des langages spécialisés (DSL) au détriment d'UML et un temps de mise en place d'un générateur pour un nouveau projet supérieur à 100 jours sont des freins importants.

4.2.2 Les produits soumis à licence commerciale

Le produit Objectering est basé sur une technologie XMI 2 et les métamodèles au format propriétaire. L'IHM[30] intégrable dans les projets apparaît comme trop simpliste pour l'usage souhaité par le groupe.

Au sein de l'outil MIA Software, la problématique concerne les formats propriétaires ultra présent. Cet aspect apparaît rédhibitoire car il nécessite la formation de l'ensemble des équipes sur un langage non réutilisable.

[30] Interface Homme Machine

Le produit Blu Age se positionne sur la génération totale d'une application et dispose d'arguments forts dans ce sens. Cependant, l'ultra-présence du format propriétaire et l'absence de fonctionnalités de round trip n'ont pas milité pour la mise en œuvre de cet outillage en 2006.

4.3 La solution retenue

Après cette étude poussée des solutions disponibles sur le marché, le CSN a conclu sur une absence de solutions collant réellement aux problématiques de la SSII[31]. Une approche spécifique a donc été développée et mise en place au travers de l'outil Rational Software Architect. Cet outil issu de la suite de logiciel Rational Rose de la société IBM a été considéré comme le meilleur modeleur du marché en 2006.

Conçue pour la version 2.1 du langage UML, la démarche mise en œuvre exploite les diagrammes de cas d'utilisation, d'activités, de séquences et de classes. Ces éléments sont ensuite repris pour illustrer les spécifications dans les différentes générations documentaires intégrant les normes du client. La métamodélisation des différents éléments s'effectue aussi avec ce langage standard ce qui permet par exemple de disposer d'une vision UML de l'architecture.

Dans la démarche MDA, la notion de PDM représente la modélisation de l'architecture et des spécificités technologiques d'une application. Les modèles associés représentent par conséquent l'ensemble de la logique technique de l'application. La nécessaire mise en œuvre de ces éléments a engendré la création d'une nouvelle entité : la notion de framework. Celle-ci est bien adaptée au contexte de la SSII et permet de créer une méthodologie de conception et une architecture par client. Ce framework (Figure 33) doit pouvoir être mis en place rapidement (délai allant de 2 à 15 jours suivant la complexité).

[31] Société de Services en Ingénierie Informatique

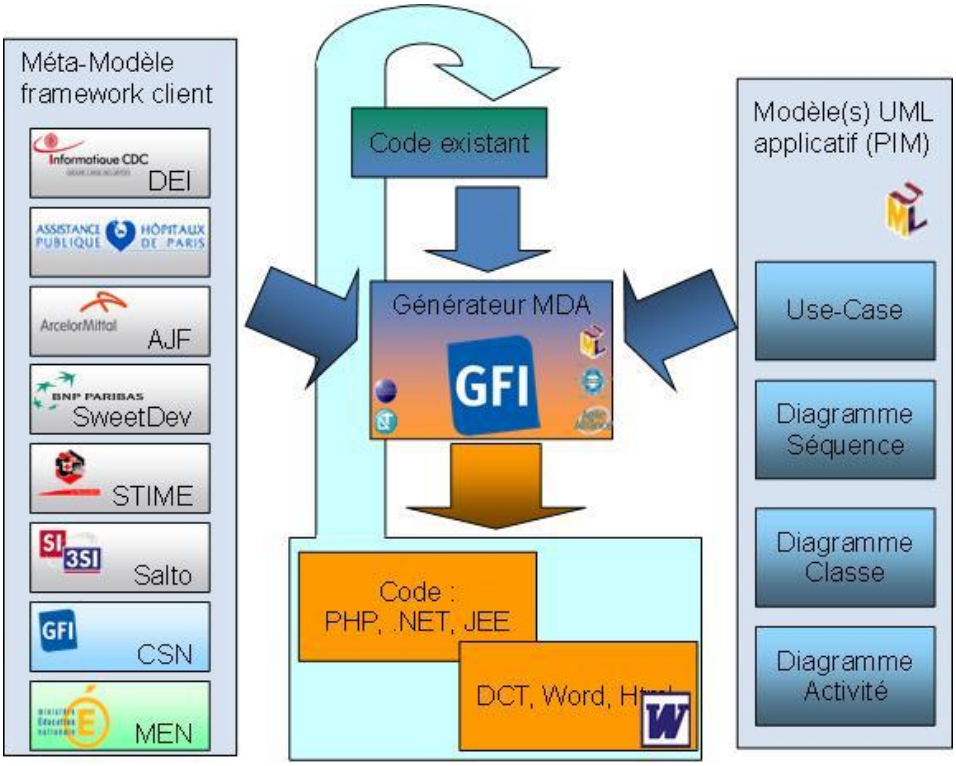

Figure 33 : Vision GFI de la démarche MDA

La solution mise en place peut se targuer de disposer d'un système de génération efficace (génération de code d'un projet en moins de 2 minutes). Extensibles et configurables, les transformations utilisent un langage de template Open Source similaire au langage JSP. Elles prennent en charge la synchronisation du code de l'application lors de multiples générations, le contrôle des encodages, le nettoyage des noms et la gestion des conflits éventuels dans la technologie ciblée. Ce système de génération permet de gérer l'ensemble des transformations nécessaires dans une démarche MDA :

- Model to Model (M2M) : raffinement du PIM apporté par différents concepteurs (Figure 34),

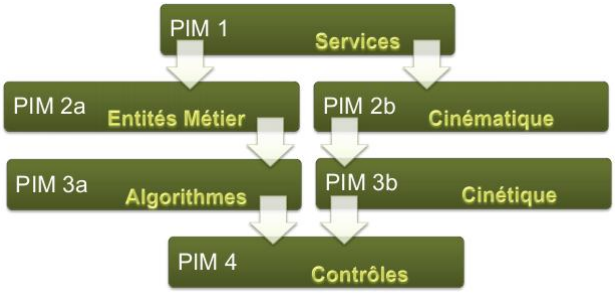

Figure 34 : Exemple de raffinement du PIM

- Model to Text (M2T) : génération de code ou documentaire,
- Text to Model (T2M) : rétro ingénierie d'applications existantes,
- Model To Documentation (M2D) : génération documentaire.

Cette nouvelle méthodologie de travail nécessite un apprentissage assez limitée. La formation des concepteurs se borne en effet à une formation au langage UML allant de 1 à 5 jours en fonction du degré de spécification. De plus, au fil des projets, de nombreux outils de validation de modèles, de formation, de documentation ont été mis en place pour l'assistance du concepteur. Ces différents éléments permettent de tendre vers une complétude de plus en plus importante de la démarche.

4.3.1 Les approches

La démarche MDA intègre deux approches différentes pouvant être mises en place en fonction de la nature du projet : l'approche descendante (Bottom Up) et l'approche descendante (Top Down).

4.3.1.1 L'approche Bottom up

Cette approche particulière de la démarche MDA concerne en général les projets sur lesquels une première analyse fonctionnelle aboutissant à l'obtention du modèle de données a déjà été effectuée.

Les projets de refonte entrent très bien dans ce cadre car la méthodologie la plus efficace consiste à partir de la base de données existante pour arriver à la représentation de l'IHM de l'application. De fait, il est plus simple de commencer à modéliser les entités métiers (cette tâche peut être automatisée avec de la rétro ingénierie) pour ensuite modéliser les services métiers. Enfin, l'IHM qui bien souvent aura été retravaillé en termes de contenu, de cinématique[32] et de cinétique[33] sera modélisé en dernier.

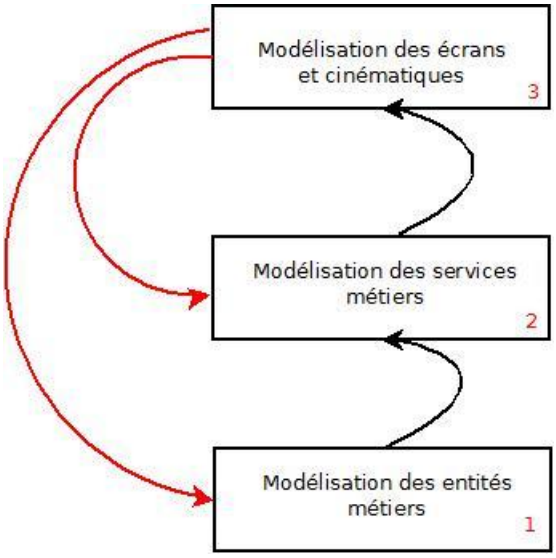

Figure 35 : Bottom Up

[32] La cinématique applicative s'intéresse aux enchaînements de l'IHM dans l'application. Elle ne permet cependant de décrire le fonctionnement interne d'une « page »

[33] La cinétique applicative s'intéresse au dynamisme interne de la page à l'aide de technologies telles que Ajax (exemple : un élément IHM mettant à jour une variable)

4.3.1.2 L'approche Top down

L'approche descendante s'applique quant à elle plus souvent sur des projets où le besoin fonctionnel n'est que très peu connu. Les nouveaux projets correspondent parfaitement à cette problématique. Le besoin fonctionnel est considéré plus simple à exprimer en partant de la modélisation de l'IHM pour ensuite décrire les services et les entités métiers. Le travail de spécification associé est ainsi réalisé de façon itérative.

La pertinence de cette approche apparaît cependant comme discutable au regard de l'absence de plasticité dans l'IHM de l'application.

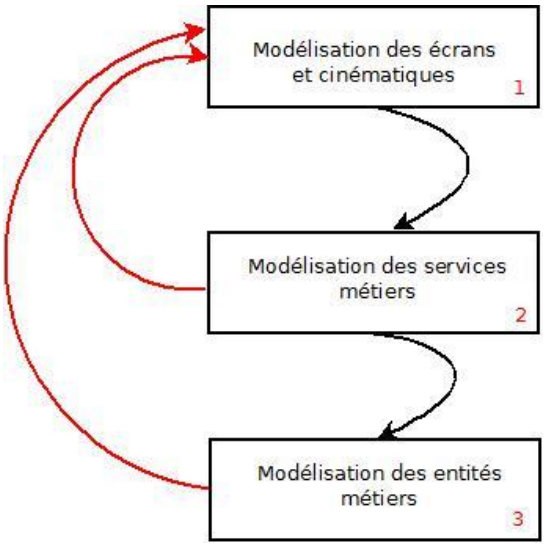

Figure 36 : Top Down

4.3.1.3 Niveaux de modélisation

Afin de répondre au mieux aux exigences des clients (avancement, acteurs, charges, etc.), un système de niveaux de modélisation a été mis en place pour le développement d'applications dans le groupe GFI. Chaque niveau hérite des apports du niveau inférieur et prend ainsi en charge des fonctionnalités plus ou moins importantes dans la génération du code (Figure 37).

Conception	Sortants	Bronze	Argent	Or	Platine
Couche Persistence	MCD , MPD	✓	✓	✓	✓
Couche Métier	Règle de gestion, branchement métier	!	✓	✓	✓
Cinématique Applicative	Branchement MVC , lotissement Use Case	!	⚠	✓	✓
Cinétique Applicative	RIA , composants personnalisés	!	!	!	✓
	DCT / SFD / Code	★	★	★	★
			★	★	★
				★	★
					★
	Formation MDA requise	0.5j	1j	2j	5j

Figure 37 : Niveaux de modélisation

Les niveaux de modélisation sont donc par ordre de fonctionnalités et de besoins de formation : bronze, argent, or et platine. Le choix parmi ces différents éléments est fait en fonction du retour sur investissement souhaité sur le projet.

4.3.1.3.1 Niveau « bronze »

Ce premier niveau de modélisation permet de :

- représenter la base de données,
- découper fonctionnellement par regroupement d'entités packagées,
- documenter clairement le modèle de données métier,
- mettre à disposition une vision d'ensemble du modèle métier.

La représentation de la base de données se fait au travers de classes UML. Les diagrammes de classes (Figure 38) permettent la génération du modèle conceptuel de données, des classes modélisées, ainsi que du mapping objet/relationnel de ces classes.

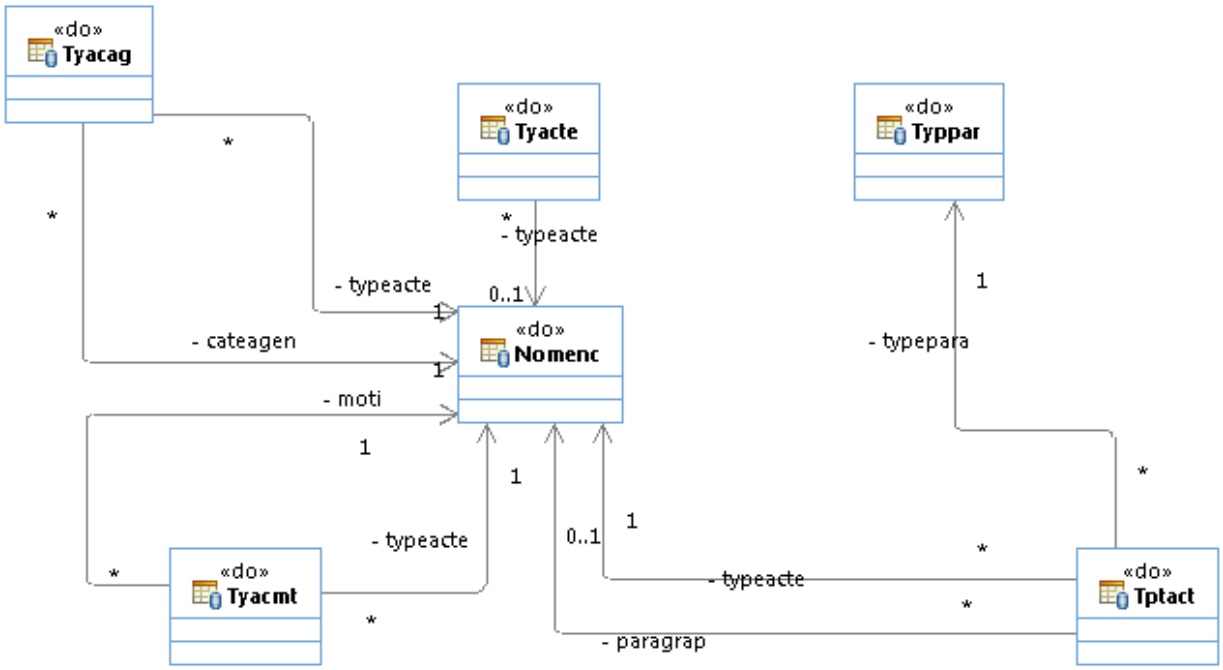

Figure 38 : Modélisation du MLD associé au concept de type d'acte

4.3.1.3.2 Niveau « argent »

Ce second niveau de modélisation permet de :

- représenter les différents acteurs et les activités qui leur sont accessibles,
- représenter la cinématique de l'application,
- identifier les pages et les actions IHM,
- lier la couche IHM et la couche métier au travers de diagrammes de séquence.

La représentation des différents menus de l'application nécessite la mise en œuvre d'un diagramme de cas d'utilisation (Figure 39). Cet élément permet en outre de définir les droits associés à chacun des utilisateurs.

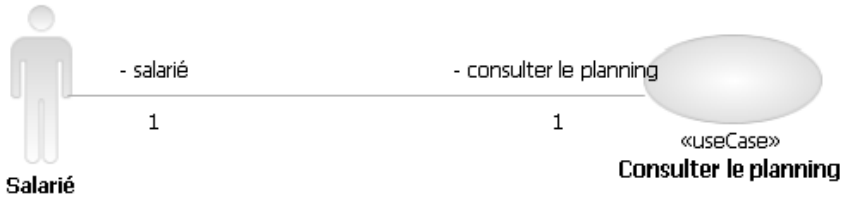

Figure 39 : Exemple de diagramme de cas d'utilisation

Pour représenter la cinématique de l'IHM, il est nécessaire de modéliser les différents éléments associés au travers de diagrammes d'activités (Figure 40). Ces informations permettent de générer la structure de l'IHM, des actions appelées et des contraintes de transition au sein de l'application.

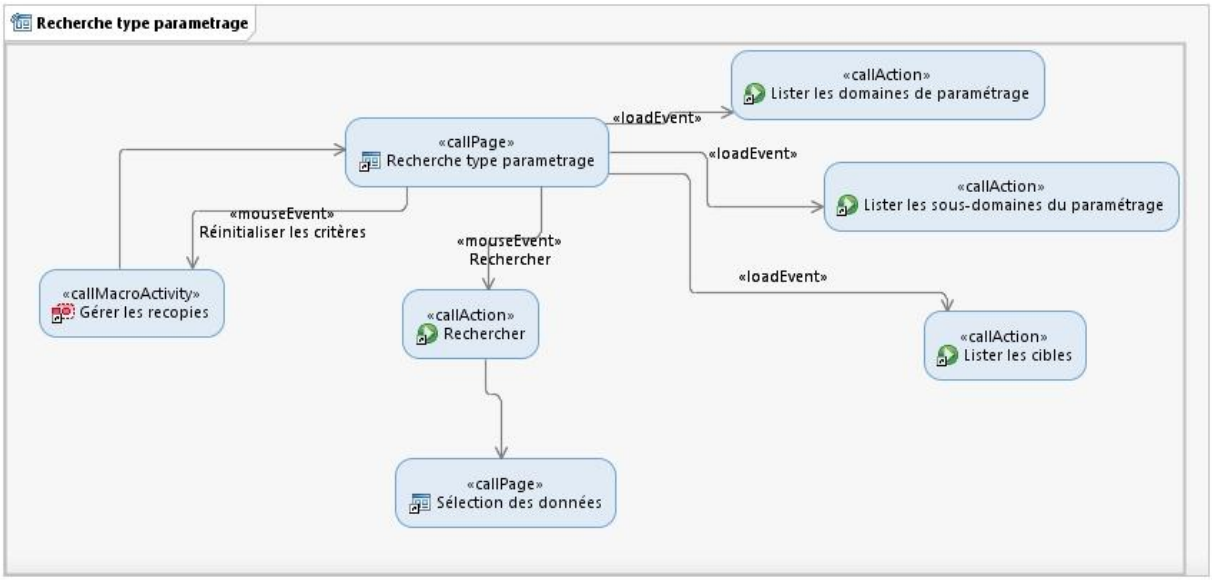

Figure 40 : Modélisation de la cinématique de l'IHM « Recherche de paramétrage »

Les services métier sont quant à eux modélisés par le biais d'un diagramme de classe tandis que leurs interactions sont traduites au sein de diagrammes de séquence (Figure 41). Ces différents éléments permettent en outre la génération sous forme de pseudo code du contenu des services métiers et la gestion de la sécurité sur ces services.

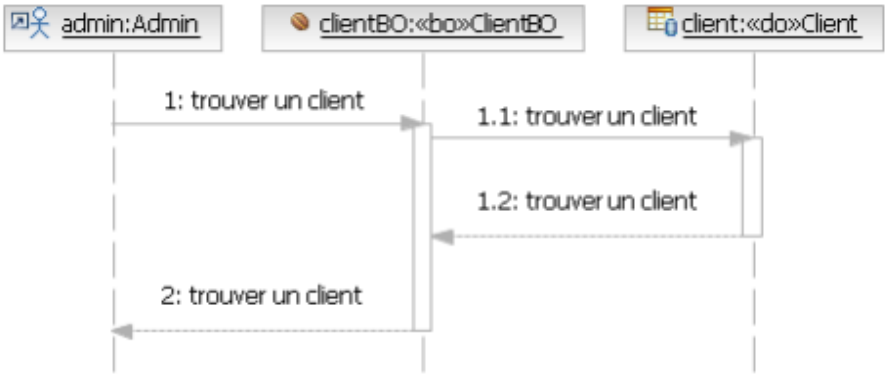

Figure 41 : Exemple de séquencement des services métiers

4.3.1.3.3 Niveau « or »

Ce troisième niveau de modélisation permet de :

- représenter le contenu de l'IHM (composants graphiques),
- lier les composants graphiques avec un contexte IHM en associant par exemple l'exécution d'une action à un événement sur un composant graphique (un clic sur un bouton),
- ajouter des contraintes d'utilisation.

Pour représenter les éléments composant une page, il est nécessaire de les ajouter comme attributs à l'entité UML "page". L'ajout de boutons, de champs de saisie et de tableaux est ainsi possible et permet de lier chacun d'eux avec un évènement et une action à exécuter.

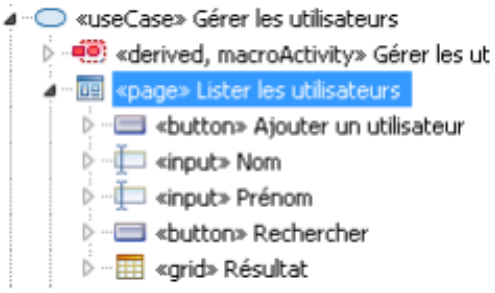

Figure 42 : Modélisation des éléments d'une page

Le contenu de l'IHM est ainsi généré sans positionnement des objets. Le mapping avec les données métier peut également être renseigné et ainsi limiter les erreurs d'alimentation des champs différents.

4.3.1.3.4 Niveau « platine »

Ce quatrième et dernier niveau de modélisation au sein de la solution mise en place dans le groupe permet de :

- représenter la cinétique applicative,
- créer des composants IHM décrivant des comportements internes et réutilisables.

Le niveau « platine » n'intègre pas de diagramme supplémentaire. Il permet l'enrichissement du contenu des pages en intégrant des opérations appelables par le biais d'actions internes. Ce genre de modélisation s'inscrit parfaitement dans le cadre de conception de projet de type RIA[34] du fait de la description du comportement interne à la page en réponse à des événements de l'utilisateur.

[34] Rich Internet Application : application Web riche

4.3.2 Les entrants

Dans les projets de mise en œuvre de la démarche MDA, le formalisme du PIM, la connaissance de l'interfaçage avec le client et la mise en œuvre des profils spécifiques représentent les différents entrants nécessaires.

4.3.2.1 Formalisme du PIM

L'objectif principal de ce niveau de modélisation concerne la description complète du métier lié à l'application. Le PIM permet ainsi d'éviter les zones d'ombre fonctionnelle et de disposer d'une documentation complète de la logique métier. De plus, l'utilisation du langage UML permet de garantir l'homogénéité des modèles entre les concepteurs.

Certaines limites liées à la complexité peuvent apparaître lors de la modélisation de l'application. Au niveau de l'IHM, l'agencement des différents composants ne peut être garanti. Les règles métier complexes, les formules mathématiques et les fonctionnalités « hors normes » ne pourront non plus apparaître au niveau de cette modélisation. Tous ces aspects devront cependant faire l'objet d'une documentation complète indispensable pour la transmission des notions souhaitées aux développeurs.

4.3.2.2 Interfaçage avec le client

Cette notion d'interfaçage avec le client permet de distinguer trois cas de figure différents. Le premier cas correspond à une utilisation du langage UML au niveau du client. Le concepteur doit alors saisir les documents de spécification dans ce langage ce qui permet de mettre à plat le problème.

Dans le second cas, le client utilise UML dans un formalisme différent du logiciel IBM Rational Software Architect. Il est alors possible d'importer ses modèles et de les rendre exploitable au travers d'une transformation M2M (Model To Model). Cette méthodologie permet de gérer les changements du client itérativement mais aussi de détecter les anomalies de conception du client.

Dans le dernier cas, le client travaille en mode Agile. La modélisation du PIM est faite directement par le concepteur car on considère que la spécification au format UML prend moins de temps qu'une description textuelle.

L'application de la démarche MDA peut être stoppée à tout moment d'un projet. Le code et les livrables ne font en effet absolument pas référence à cette démarche. Il n'est par contre pas possible de travailler avec MDA de façon intermittente. Un conflit de référentiel entre le code retouché manuellement et les modèles UML pourrait en effet apparaître dans le projet concerné.

4.3.2.3 Profils UML spécifiques

Plusieurs profils UML spécifiques ont été développés au sein du groupe GFI pour permettre la mise en application du MDA dans le cadre du développement des applications. Nous présentons dans ce cadre l'exemple de deux profils spécifiques : PIM et IHM.

4.3.2.3.1 PIM

La mise en œuvre d'un profil PIM permet de stéréotyper les classes UML selon leur utilisation. Ces informations sont exploitées par le générateur de code associé et permettent de donner un complément d'informations à la modélisation. Certaines valeurs de ce profil sont présentées à titre d'exemple (Tableau 3).

Entité « do » (Data Object)	Modélisation des entités métiers. Les attributs d'un « do » correspondent à des données qui devront être stockées sur un support tel qu'une base de données.
Entité « vo » (Value Object)	Classe qui va permettre de faire transiter des données au sein de l'application.
Entité « dao » (Data Access Object)	Interface portant les opérations ne pouvant être portées par un « do » (exemple : webservices).
Entité « cbo » (Common Business Object)	Enumération statique ou dynamique.
« action »	Modélisés au travers d'un comportement opaque, il correspond à un traitement permettant de charger ou de recevoir les données d'une page. Fait le lien entre la couche métier et l'IHM.
« page »	Eléments IHM correspondant à une page.
« macroactivity »	Mise en relation des pages et des actions modélisés.
Entité « bo » (Business Object)	Modélisation des services métiers permettant de faire le lien entre l'IHM et les entités métiers
Entité « bd » (Business Delegate)	Classe pouvant appeler plusieurs services métiers à la fois. Utilisée pour réduire le couplage entre la couche de présentation et la couche métier.

Tableau 3 : Ensemble des stéréotypes du profil « PIM »

4.3.2.3.2 IHM

Le profil présenté permet de stéréotyper les modèles de niveau PIM pour leur donner une signification en termes d'IHM. Certaines valeurs de ce profil sont présentées à titre d'illustration (Tableau 4).

« style »	Possibilité d'intégrer des styles dans des fichiers css.
« mouseEvent »	Evènements de souris (clic gauche, clic droit, roulette, double clic, etc.).
« dragAndDropEvent »	Glisser déplacer.
« keyEvent »	Evènements de clavier (presser, haut, etc.).
« loadEvent »	Evènements de chargement de la page entière ou d'un élément de la page.
« popup »	Stéréotype au niveau de la transition pour signaler le chargement de la page au sein d'un pop-up.
« input »	Champ de saisie textuel.
« list »	Liste dépliable avec plusieurs valeurs sélectionnables.
« radioGroup »	Groupe de boutons radio au sein duquel un seul est élément sélectionnable.
« radio »	Bouton radio présent dans un « radiogroup ».
« check »	Case à cocher.
« button »	Bouton.
« progress »	Barre de progression.
« searchCriteria »	Ensemble de critères de recherche.
« pageFragment»	Ensemble d'éléments graphiques, réutilisables dans les « page » et autres « pagefragment ».
« custom »	Composant réutilisable.

Tableau 4 : Ensemble des stéréotypes du profil « IHM »

Ces stéréotypes sont interprétés par les templates de génération de code au sein des différentes transformations de modèles.

4.3.3 Les sortants

Au sein des sortants de la démarche MDA, on trouve l'ensemble des transformations de modèles, des outils pour l'estimation de coût et la progression dans le projet et des outils d'analyse de la conformité de la modélisation.

4.3.3.1 Transformation M2T (Model To Text)

La démarche MDA déployée au sein du groupe ne permet pas de générer la totalité du code de l'application. L'efficacité de celle-ci est variable en fonction de la couche :

- efficacité au niveau de l'architecture : 100%
- efficacité au niveau de la couche persistance : 70 à 90%
- efficacité au niveau de la couche métier : 30 à 100%
- efficacité au niveau de la couche IHM : 80 à 100%

La génération du code de l'application peut être complète ou partielle avec la possibilité de constituer des lotissements techniques ou fonctionnels. Cette génération est caractérisée par sa rapidité (moins d'une minute pour générer une application de plus de 4 000 jours - homme).

Les fonctionnalités de cette transformation évoluent sans cesse pour prendre en compte des nouveaux besoins fonctionnels ou techniques. Les retours d'expérience des développeurs et concepteurs permettent l'évolution efficace de ce processus.

4.3.3.2 Transformation M2D (Model To Documentation)

Cette transformation permet la génération de la documentation technique associée au projet et permet ainsi de donner une feuille de route et une vue d'ensemble au développeur. Ces documents permettent en outre de disposer de l'information sur l'avancement de la modélisation (numéro de révision, date et auteur de la dernière sauvegarde sur le gestionnaire de versions).

4.3.3.3 Reverse engineering T2M (Text To Model)

Cette transformation permet de générer des modèles à partir d'une application existante mais dispose d'un faible niveau de réutilisabilité d'un projet à un autre. Ces opérations mettent souvent en évidence des anomalies de cohérence ou de complétude dans l'application existante.

4.3.3.4 Cost Estimation

La démarche MDA permet de donner une estimation des coûts de développement à partir des différents modèles mis en œuvre. Cette estimation assiste le chef de projet dans la réalisation de son planning et permet de donner au directeur de produit une idée du coût total de l'application.

4.3.3.5 Progress Estimation

En s'appuyant sur le différentiel entre le code généré et le code développé, il est possible d'obtenir une estimation de l'avancement du projet en temps réel. Le pourcentage réel de réalisation doit être cependant couplé au pourcentage de succès et à la couverture des tests pour correspondre à la réalité.

4.3.3.6 Checkstyle

Cet outillage permet de donner en temps réel la qualité de l'expression fonctionnelle et s'inscrit ainsi dans une démarche d'amélioration continue. Les règles de bonne pratique permettent de conseiller le concepteur et réduit les allers retours entre la phase de conception et de développement au regard de :

- la performance : analyse de la complexité des chaînes d'appels,
- la compréhension : présence de lexique métier, documentation,
- la complétude : analyse du modèle inutilisé,
- l'intégrité : cohérence des appels métiers, des types, etc.

4.3.4 Impacts sur les flux de production

A ce jour, la plupart des contrats de développement d'applications remportées par le groupe passent par le Centre de Services National et la démarche MDA est appliquée sur des projets de taille conséquente. L'entité de GFI Informatique ayant remporté le marché reste le maître d'œuvre du projet mais la conception et le développement se fait au sein du CSN. Ce mode de fonctionnement couplé à la démarche MDA a entraîné la modification des flux de production (Figure 43).

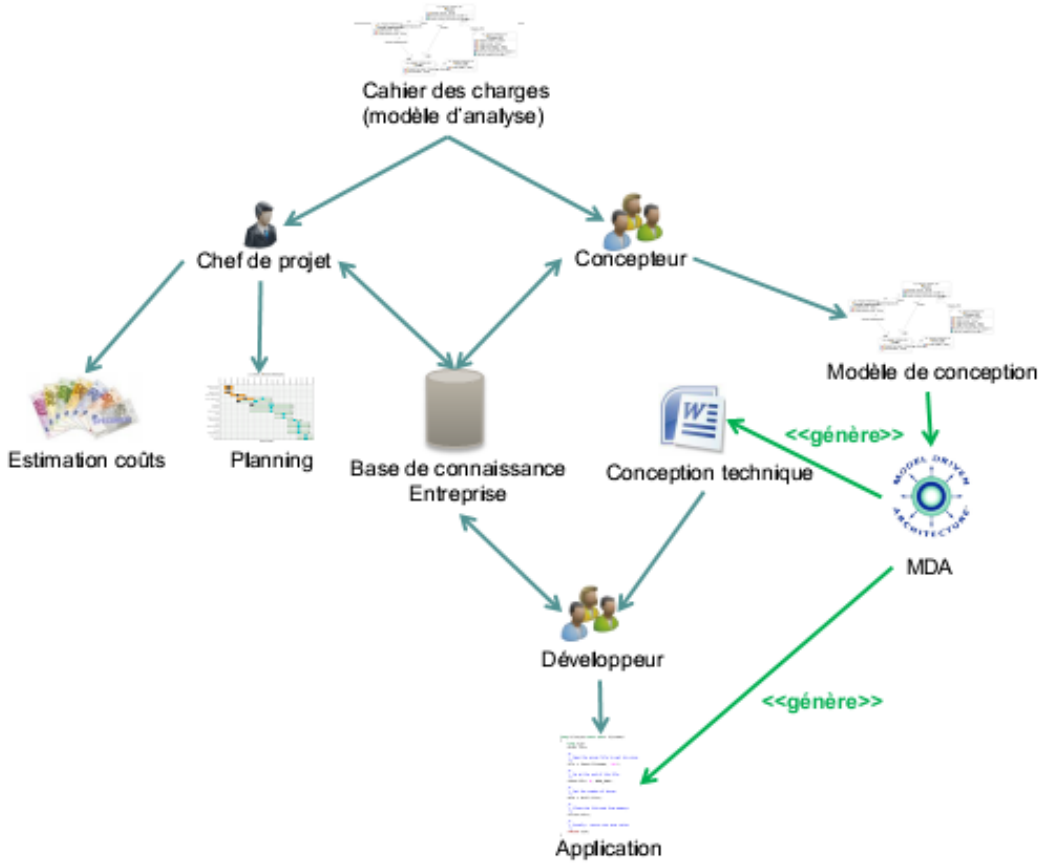

Figure 43 : Flux de production avec une utilisation basique de la démarche MDA

La préoccupation première de la démarche MDA concerne la génération d'une application grâce à un ensemble de transformations de modèles. Cependant, son rôle peut ne pas se limiter à ces opérations et impacter tous les niveaux du processus de production du logiciel :

- génération de l'estimation de coût du projet,
- assistance à la confection du planning du projet,
- ajustement de la base de connaissance de l'entreprise,
- génération des cahiers de tests.

Au sein du groupe GFI, la productivité de la démarche MDA est considérée comme proportionnelle à la complexité du projet. La rentabilité est par conséquent forte sur les gros projets ou sur des projets avec une architecture complexe. Elle apparaît au contraire comme beaucoup plus discutable sur les projets plus restreints. Cette démarche permet par contre un gain indéniable pour tous les types de projets dans le domaine de la maintenance applicative. En effet, dans le cas où la technologie d'une application doit évoluer, la logique métier précédemment modélisée reste quant à elle toujours valable et exploitable.

La mise en œuvre de cette démarche permet par ailleurs d'avancer dans l'industrialisation du cycle de vie de projet. Il est en effet possible de paralléliser les études fonctionnelles et techniques, d'automatiser certains contrôles et de réaliser des modifications concurrentes sur les référentiels.

Chapitre 5 : La mise en œuvre de la démarche MDA

Les chapitres précédents nous ont permis d'introduire deux visions différentes de la démarche MDA. La première version présentée est en effet directement issue des spécifications de l'OMG tandis que la seconde correspond à une interprétation de ces spécifications pour répondre aux besoins industriels.

Afin d'illustrer les différences dans les démarches, nous mettons en application la vision industrielle au sein d'un développement d'application métier. Dans ce chapitre, nous allons tout d'abord présenter cette réalisation au travers de la démarche 2TUP et du langage de modélisation UML. Nous analysons ensuite la place de la démarche MDA dans le projet.

5.1 Etude préliminaire

Le paramétrage représente le cœur du système du progiciel IODAS. Il permet en effet d'adapter le produit à l'ensemble des spécificités des clients. L'expérience accumulée au sein de l'entité GFI Progiciels a permis de cibler l'absence d'outils permettant la manipulation simplifiée de ces éléments. Les consultants fonctionnels, premiers utilisateurs de ces fonctionnalités de paramétrage, ne disposent pas d'un outillage répondant pleinement aux besoins de manipulation de celui-ci. L'application permettant à ce jour la recopie de paramétrage ne couvre pas l'ensemble des fonctionnalités nécessaires.

5.1.1 Analyse de l'existant

L'outil de recopie du paramétrage existant est maintenu au fil des évolutions dans le paramétrage du progiciel. Développé dans une technologie Visual Basic, il dispose de deux fonctionnalités principales :

- la recopie des éléments de paramétrage d'une base de données source à une base de données cible,
- la recopie des éléments de paramétrage sur une base de données distante par l'intermédiaire d'une génération de scripts PL/SQL[35].

L'utilisateur (consultant ou client) choisit un domaine de paramétrage et peut visualiser les éléments associés sur la base de données source et éventuellement sur la base de données cible sélectionnée. Le choix des éléments à recopier déclenche la vérification par l'outil de la possibilité d'intégration de ces éléments. L'ensemble des hiérarchies nécessaires à cette manipulation sont alors affichées à l'utilisateur pour traitement au travers d'un ensemble de messages bloquants ou d'avertissements.

Au sein de l'outil actuel, la recopie des éléments doit nécessairement se faire de manière ascendante. Les éléments de paramétrage de plus bas niveau doivent être recopiés en amont de tout paramétrage de plus haut niveau. La recopie d'un élément de haut niveau tel qu'un type de procédure nécessite dans ce cas la manipulation préalable d'une trentaine d'éléments de paramétrage.

5.1.2 Recueil des besoins fonctionnels

Nous avons pris ci-dessus l'exemple des consultants fonctionnels, mais ils ne sont pas les seuls à devoir effectuer ce genre d'opérations de manipulations du paramétrage du progiciel. Celles-ci sont en effet présentes dans de nombreuses missions des collaborateurs de l'entité Progiciels telles que la mise en place de projets fonctionnels, l'assistance téléphonique ou le déploiement de modules complémentaires.

[35] Procedural Language / Structured Query Language : permet de combiner des requêtes SQL et des instructions procédurales (boucles, conditions...)

5.1.2.1 Projets fonctionnels

Dans le cadre d'une mise en œuvre d'un projet client, l'expert fonctionnel joue un rôle essentiel pour la bonne réussite de celui-ci. Les éléments statués lors de l'analyse différentielle se traduisent dans la plupart des cas par un paramétrage spécifique à mettre en place. Celui-ci est d'abord déployé sur une base de données de « recette » pour que le client réalise les différents tests fonctionnels. Les éventuels retours sont traités sur cette base jusqu'à obtenir l'approbation totale du client. Cette validation fonctionnelle entraine le déploiement du paramétrage sur la base de données de production par le biais d'une recopie de paramétrage. Dans beaucoup de projets où les validations fonctionnelles successives se font sur des périmètres restreints, les bases de données « recette » et « production » doivent évoluer en même temps. L'outil de recopie actuel n'étant pas capable de mettre à jour des données existantes, le consultant doit souvent multiplier les modifications manuelles.

Parmi ses missions, l'expert doit aussi proposer un paramétrage modèle dans son domaine fonctionnel. Suite à l'analyse des différentes évolutions de législation, le paramétrage correspondant est mis en place sur une base de référence située sur l'un des serveurs de l'entité Progiciels. L'outil de recopie actuel ne gérant pas la modification d'éléments existants, les risques d'erreurs sont importants lors de la transmission des éléments aux différents clients. Les diffusions correspondantes sont effectuées sous la forme de scripts exécutables sur les bases clientes.

5.1.2.2 Assistance téléphonique

Certaines irrégularités de fonctionnement du progiciel signalées par le client sont issues d'incohérences dans le paramétrage. L'arrivée à un diagnostic fiable est à ce jour très compliqué et la résolution de ce genre de problématique pose un problème important au niveau de la qualité de réponse de l'assistance téléphonique.

5.1.2.3 Modules complémentaires

Le déploiement de modules complémentaires à l'utilisation du progiciel tel que des modules de paiement et des éditions bureautiques nécessitent aussi la manipulation d'élément de paramétrage. La maintenance des règles de gestion de paiement des assistantes familiales nécessitent par exemple la manipulation de milliers d'éléments à chaque évolution de la législation.

5.1.2.4 Synthèse des besoins

A ce jour, le constat est simple. Les interventions des différents collaborateurs sont affectées en termes de productivité et de qualité par l'absence d'un outillage de manipulation du paramétrage couvrant l'ensemble des fonctionnalités nécessaires. L'application de la démarche MDA aura donc pour objectif de tenter d'apporter une réponse à cette problématique en mettant à disposition les fonctionnalités suivantes (Figure 44) :

- la recopie des éléments de paramétrage d'une base de données source à une base de données cible,
- la recopie des éléments de paramétrage sur une base de données distante par l'intermédiaire d'une génération de scripts.

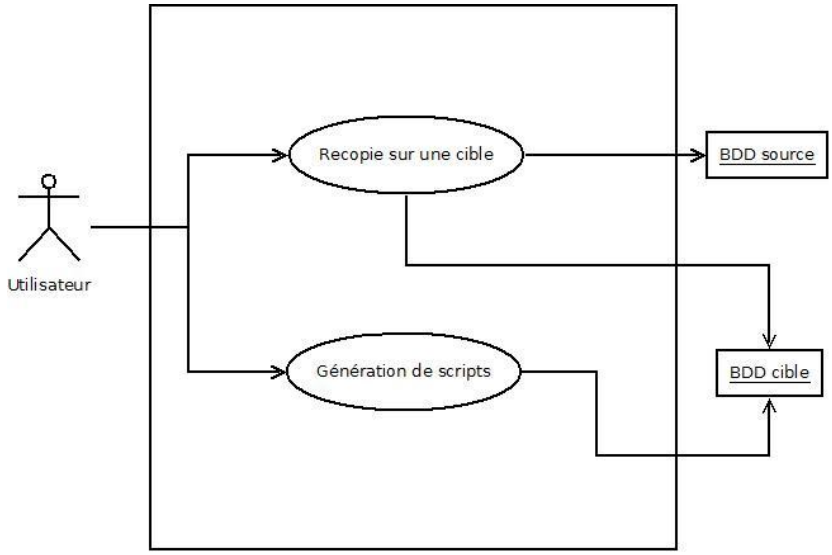

Figure 44 : Diagramme de contexte statique de l'application

L'outil existant permet de faire ces différentes opérations mais de manière non optimale. Afin de garantir une réelle valeur ajoutée dans le travail quotidien, l'application mise en œuvre dans ce projet doit nécessairement apporter des fonctionnalités supplémentaires considérées comme stratégiques :

- garantir la prise en charge des dépendances entre les différents éléments de paramétrage,
- autoriser la modification de paramétrage existant.

5.1.3 Recueil des besoins opérationnels

Chaque utilisateur de cette application doit s'authentifier sur la base de données source et éventuellement sur la base de données cible. Cette double opération d'identification permet de certifier les droits de l'utilisateur sur ces deux bases. Les opérations de manipulation de paramétrage ne concernant pas en effet l'ensemble des utilisateurs du progiciel, elle nécessite la mise en œuvre d'une habilitation particulière permettant de restreindre son utilisation à une certaine catégorie d'utilisateurs : les administrateurs.

Les éléments de paramétrage manipulés dans cette application sont déjà intégrés au sein du modèle de données du progiciel IODAS. Par conséquent, le volume de données propre à cette application est assez faible et peut être intégré dans la base de données existante sans risque de détérioration de la performance de celui-ci.

A l'intérieur de cette application, chaque échange entre les différents composants nécessite une réponse immédiate. Les différents messages rencontrés sont donc de nature synchrone. Par exemple, l'application sollicite l'affichage d'informations provenant de la base de données source et attend son retour pour afficher les données de cette entité.

5.1.4 Les grands choix techniques

Dans ce projet, les choix techniques sont fortement contraints par la nécessaire intégration au progiciel existant. Au sein de cet outil de gestion de l'Action Sociale Départementale, le gestionnaire de base de données utilisée correspond au produit Oracle dans sa version 10. En complément, l'expérience d'un premier outil de recopie peut être réutilisé :

- utilisation d'un lien entre les bases de données pour les communications (databaselink),
- génération de scripts au format SQL[36],
- utilisation de l'outil SQL PLUS pour l'intégration des scripts sur une base de données distante.

5.2 Capture des besoins fonctionnels

Un cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier (Roques, UML 2 par la pratique, 2005). Le diagramme d'utilisation (Figure 45) permet ainsi de visualiser les différents acteurs et leurs fonctionnalités souhaitées.

[36] Structured Query Language

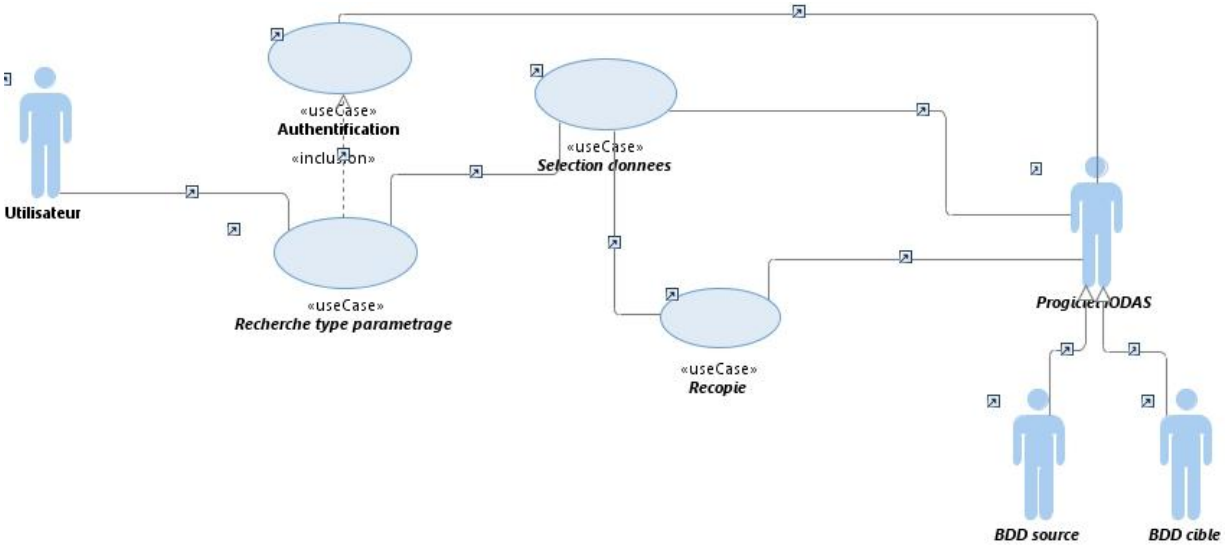

Figure 45 : Diagramme de cas d'utilisation

Au sein de cette application, la recherche de paramétrage en fonction de différents critères est disponible pour un utilisateur ayant réussi l'opération d'authentification. A la suite de cette opération, l'utilisateur pourra alors sélectionner les données à recopier sur la base de données de son choix.

5.3 Capture des besoins techniques

Comme évoqué précédemment, la capture des besoins techniques est fortement impactée par le choix d'intégration de l'application au progiciel IODAS. Les différentes technologies utilisées dans celui-ci doivent être impérativement implémentées dans l'application de cette étude :

- données au format de la base de données Oracle,
- application développée en conformité avec le framework spécifique Astre Iodas.

5.3.1 Le framework spécifique Astre Iodas

Le progiciel IODAS dans sa version Web est basée sur un framework spécifique commun à l'ensemble des applications de l'entité Solutions. Nommé « Astre » et directement issu des spécifications de la Plateforme Java J2EE de SUN Microsystems, il représente une ensemble d'outils et de règles permettant de développer une application Web en technologie Java.

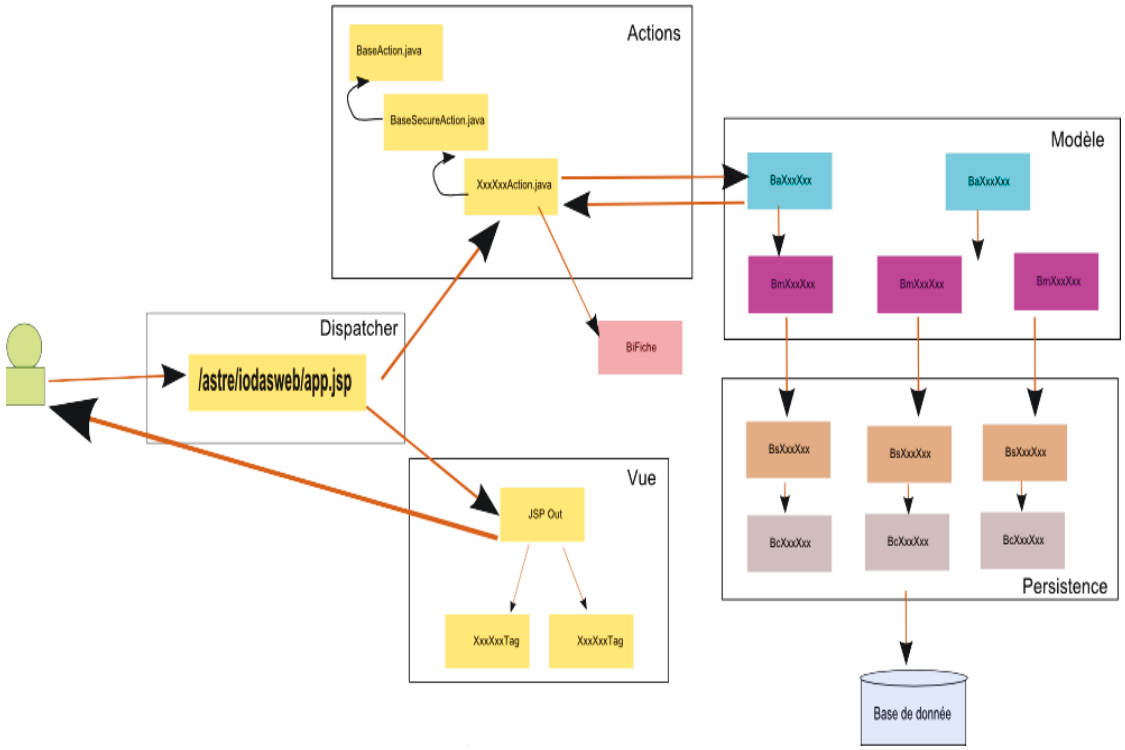

Figure 46 : Architecture Astre IodasWeb

L'utilisateur fait une requête au serveur d'application en demandant une action au dispatcher. Celui-ci représente le contrôleur central de toute l'application web et permet la communication entre les différents serveurs mis en œuvre pour le bon fonctionnement de cette application.

5.3.1.1 Architecture N-Tiers

L'architecture N-Tiers permet de centraliser sur un serveur la partie fonctionnelle des applications autrefois distribuée sur les postes clients dans le modèle Client / Serveur. Afin de répondre à cette exigence, le nouveau serveur, appelé « serveur d'application », utilise un modèle de développement « objet » ou composants et permet une centralisation de l'administration de l'application. Le poste client, allégé de la fonction traitement, réalise alors simplement la partie affichage. Cette architecture apporte les avantages des systèmes centralisés (sécurité, administration, exploitation) et du client/serveur graphique (IHM, liens bureautiques...) et constitue le point de rencontre adéquat entre l'informaticien et l'utilisateur.

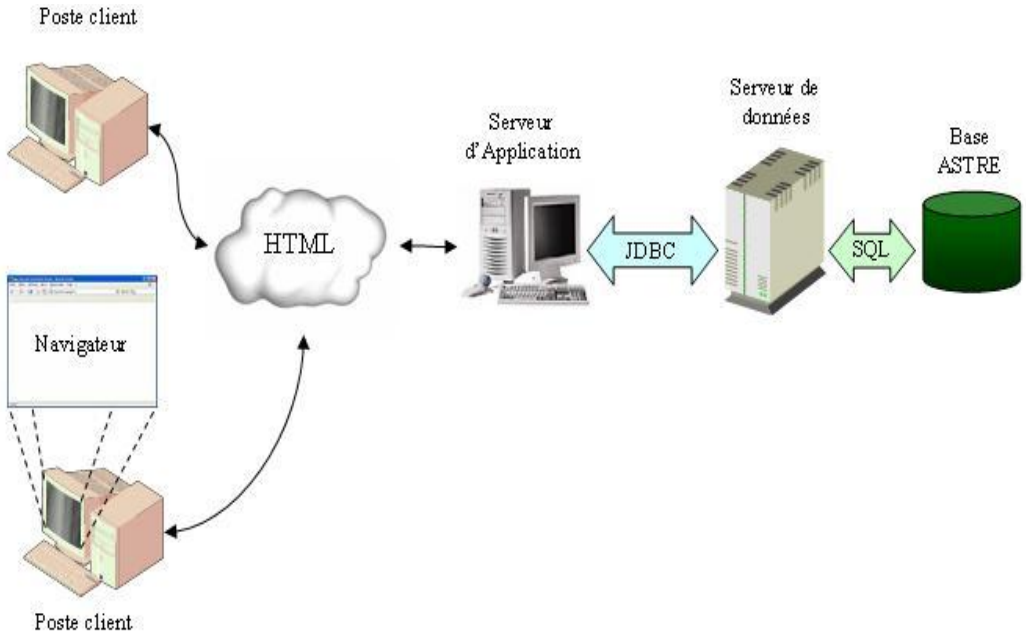

Figure 47 : Architecture N-Tiers

L'utilisation de cette architecture permet de grandement faciliter le déploiement des applications :

- absence d'exécution sur le poste client permettant une montée en charge techniquement aisée,
- portabilité et scalabilité des composants Java permettant de prendre en compte des architectures plus ouvertes.

La conception en mode objet, à base de composants Java documentés, offre une plus grande facilité de maintenance et d'évolution. Ces composants élémentaires sont utilisables dans plusieurs contextes (transactions de nature différente, batchs…). Les concepteurs bénéficient ainsi d'un catalogue de composants permettant d'accélérer sensiblement l'évolution et la maintenance des applications métiers. En ce qui concerne l'ergonomie de l'application, la navigabilité est facilitée et les fonctionnalités standards du W3C[37] sont respectées grâce à l'utilisation intensive de feuilles de style.

En ce qui concerne les communications à l'intérieur du framework Astre, une architecture de de type MVC[38] a été mise en œuvre.

[37] World Wide Web Consortium

[38] Modèle Vue Contrôleur

5.3.1.2 Architecture MVC

Le patron de conception MVC est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective. Il permet de garantir :

- l'indépendance de la logique de traitement (logique métier) par rapport à l'IHM,
- une grande stabilité dans l'utilisation des ressources,
- une maintenance facilitée par la séparation des couches composantes,
- l'intégrité des données.

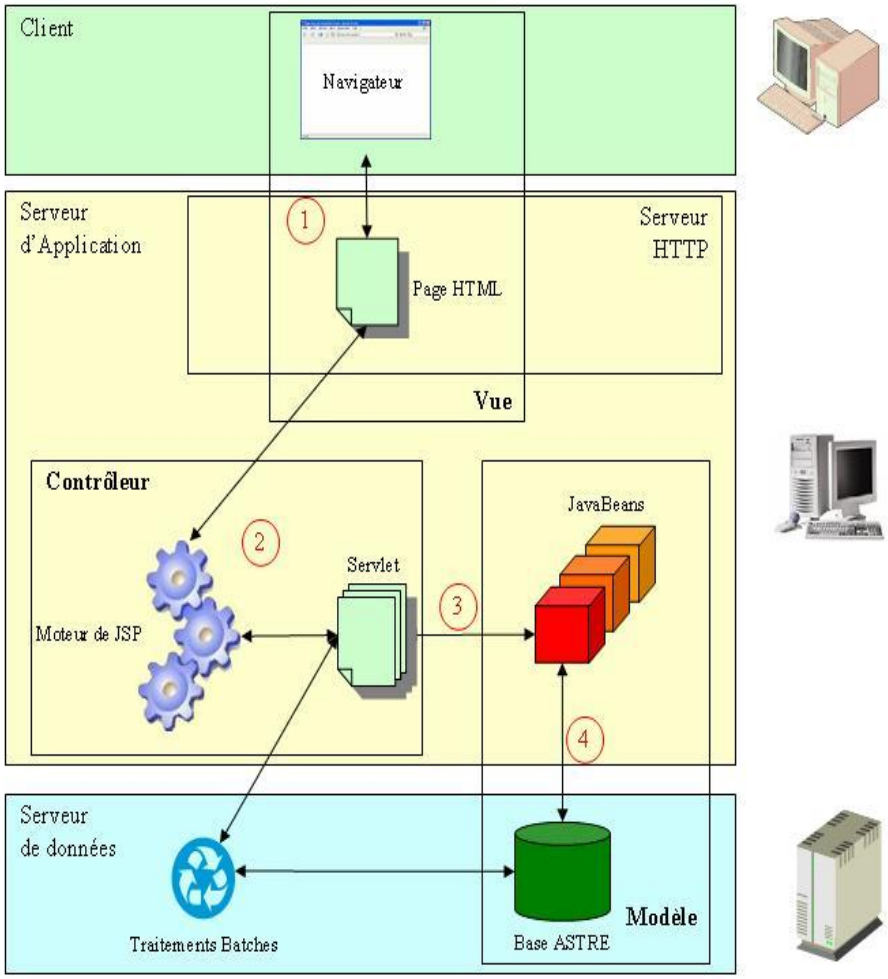

Figure 48 : Architecture MVC

Le framework Astre Iodas permet de distinguer 2 groupes de composants distincts : les composants métier et de persistance unitaire mappés sur une table et les composants métier relatifs à un cas d'utilisation.

5.3.1.2.1 Couche « Modèle »

La couche de mapping objet-persistance en base a été réalisée à l'aide d'une série de composants typés. Ces composants sont générés par un outil maison (AstreGenerator) à partir des tables présentes dans la base de données. Chaque composant de cette couche dispose d'un rôle différent dans le bon fonctionnement de l'application :

- BC (bean component) : composant de persistance de mapping des colonnes d'une table en propriétés Java.
- BS (bean spécifique) : composant de persistance proposant des fonctionnalités de mise à jour, suppression, d'ajout de données dans une table.
- BM (bean métier) : composant métier
- BA (bean application) : composant de portée applicative reprenant un ou plusieurs BM afin de gérer les aspects transactionnels.

5.3.1.2.2 Couche « Contrôleur »

Cette couche permet de piloter le flux d'entrée de l'application au travers d'actions Java. Elle gère pour cela les informations de la session de l'utilisateur, les spécificités de l'IHM en appelant des méthodes spécifiques, le lancement d'un traitement permettant l'insertion, la modification ou la suppression d'un objet métier. La transaction est en outre finalisée par cette action Java.

5.3.1.2.3 Couche « Vue »

La couche « Vue » exploite la notion de bean interface. Ce composant sert à la communication entre le contrôleur et la vue et dispose d'une durée de vie égale à celle de la requête http. Le « bean interface » peut stocker n'importe quel type d'objet (en particulier des listes pour les combos) :

- les informations données par le contrôleur et les transmet ensuite à la vue,
- les erreurs applicatives pour les afficher dans la vue.

5.3.2 Le stockage des données

Comme évoqué précédemment, le stockage des données du progiciel se fait au sein d'une base de données Oracle en version 10. Les éléments spécifiques à l'application (tables, vue, index et séquences) sont intégrés au sein d'un schéma de bases de données existant et déjà exploité par le progiciel.

5.4 Analyse

L'analyse est une étape clé dans le processus de fabrication d'un logiciel. Elle permet en effet une première ébauche de la structure objet du projet et nécessite donc d'être réalisée avec le plus grand soin.

5.4.1 Regroupement en packages

Les classes ayant une forte dépendance ou connectivité sont regroupées au sein de la notion de package qui permet de donner une vision globale plus claire de l'application. Dans l'exemple de cette étude, on distingue trois packages :

- Progiciel IODAS,
- Paramétrage,
- Recopie.

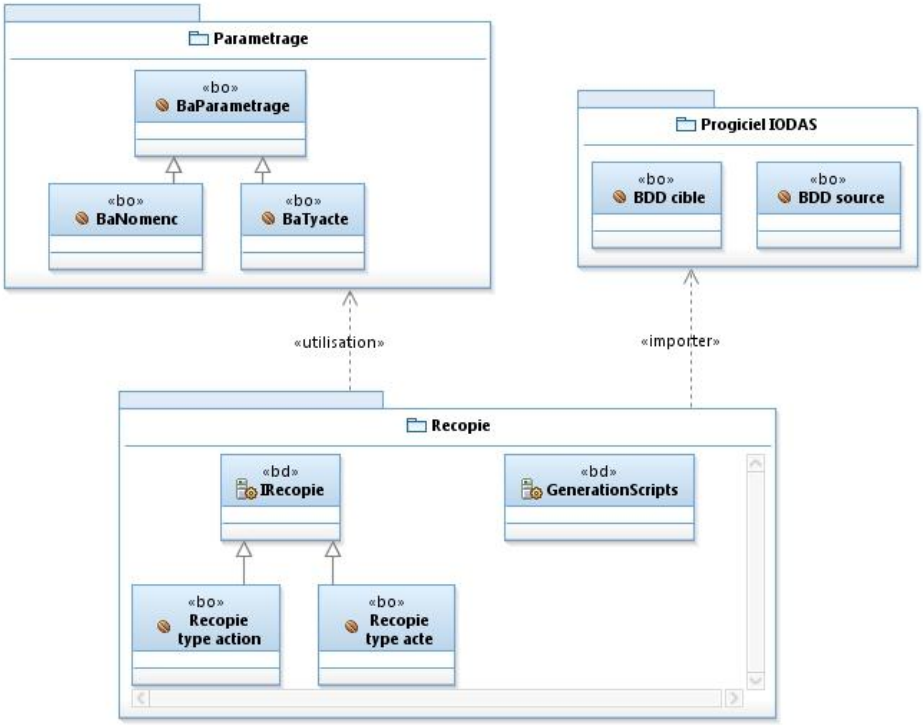

Figure 49 : Packages de l'application

Le package « Progiciel Iodas » contient l'ensemble des classes relatives aux notions de bases de données source et cible. Le package « Paramétrage » correspond quant à lui à l'ensemble des classes relatives aux éléments de paramétrage. Enfin, le package « Recopie » regroupe les interfaces représentatives des nouvelles fonctionnalités de manipulation de paramétrage. La mise en œuvre de celles-ci implique par ailleurs une évolution dans le modèle de données du progiciel.

5.4.2 Modification du modèle de données du progiciel

L'application de recopie de paramétrage souhaitée permet la manipulation d'éléments de paramétrage déjà présents dans la base de données liée au progiciel IODAS. Les fonctionnalités de recopie nécessitent quant à elles d'enrichir le modèle de données de certaines tables spécifiques. Une partie de la logique métier associée à ces recopies est en effet stockée afin de limiter les évolutions futures dans le code de l'application. Contrairement à l'outil de recopie existant actuellement, il n'est en effet pas souhaitable de gérer un modèle de données à part.

5.4.2.1 Domaines de paramétrage

Les différents domaines de paramétrage disponibles à la recopie sont référencés au sein de la table RECDOM.

Champs de la table	Type de données
ID (identifiant informatique)	INTEGER
CODEDOMA (code du domaine)	VARCHAR2(10)
LIBLDOMA (libellé du domaine)	VARCHAR2(60)
TABLPIVO (table pivot du domaine)	VARCHAR2(60)
PRODANIS (module IODAS)	VARCHAR2(10)
SOUSDOMA (restriction sous-domaine)	VARCHAR2(2000)
TABLCOMP	VARCHAR2(8)
RECO	VARCHAR(1)
CHAMCOMP	VARCHAR2(20)

Tableau 5 : Détail de la table des domaines de paramétrage

Les éléments de cette table sont considérés comme des occurrences internes et nécessitent dans ce cadre d'être inclus dans la livraison du progiciel. Chaque ajout de nouveaux domaines de paramétrage au fil des versions nécessitera l'alimentation des éléments correspondants dans cette table spécifique.

5.4.2.2 Dépendances entre les éléments de paramétrages

La recopie des éléments d'un domaine de paramétrage nécessite la manipulation de multiples concepts de paramétrage ayant des liens entre eux. La logique fonctionnelle d'agencement de la manipulation des objets métiers est donc stockée au sein d'une nouvelle table nommée RECDEP.

Champs de la table	Type de données
ID (identifiant informatique)	INTEGER
ID_DOMA (identifiant du domaine de paramétrage)	INTEGER
ID_DOMADEPE (identifiant du domaine de paramétrage dépendant)	INTEGER
TABLLIEN (Table de lien entre les dépendances)	VARCHAR2(6)
CHAMLIENDOMA (Champ sur la table du domaine principal permettant le lien)	VARCHAR2(20)
CHAMLIENDEPE (Champ sur la table du domaine dépendant permettant le lien)	VARCHAR2(20)
TYPEDEPE (Type de dépendance (S=Bloc fonctionnel secondaire, B=Blocage, A=Avertissement))	VARCHAR2(1)
SENSDEPE : Sens de la dépendance (A=Ascendant, D=Descendant)	VARCHAR2(1)
NUMEORDR (Numéro d'ordre)	INTEGER

Tableau 6 : Détail de la table des dépendances entre les domaines de paramétrage

Les éléments de cette table sont aussi considérés comme des occurrences internes et nécessitent donc d'être inclus dans la livraison du progiciel. Chaque ajout de dépendances entre les domaines de paramétrage nécessitera l'alimentation des éléments correspondants dans cette table spécifique.

5.4.2.3 Compte-rendu de la recopie de paramétrage

La loi "informatique et libertés" de la CNIL[39] impose que les organismes mettant en œuvre des fichiers garantissent la sécurité des données traitées. Cette exigence se traduit par un ensemble de mesures que les détenteurs de fichiers doivent mettre en œuvre, essentiellement par l'intermédiaire de leur direction des systèmes d'information ou de leur responsable informatique.

La notion de trace informatique correspond par conséquent à une préoccupation importante des clients du progiciel IODAS. La recopie de paramétrage est en outre une activité sensible dans la vie du progiciel et nécessite la sauvegarde de traces.

[39] Commission nationale de l'informatique et des libertés

Champs de la table	Type de données
ID (identifiant informatique)	INTEGER
ID_ACTR (identifiant d'acteur)	INTEGER
CODEMESS ((A)vertissement / (B)locage / (C)onsignation)	VARCHAR2(1)
TYPERECO (type de recopie (I/U/D))	VARCHAR2(1)
CODEPRIN (code du paramétrage principal recopié)	VARCHAR2(10)
LIBLPRIN (libellé du paramétrage principal recopié)	VARCHAR2(30)
ID_DOMA (identifiant du type de paramétrage recopié)	INTEGER
CODESECO (code du paramétrage secondaire recopié)	VARCHAR2(10)
LIBLSECO (libellé du paramétrage secondaire recopié)	VARCHAR2(30)
PRIN (Recopie de l'élément principal (O/N))	VARCHAR2(1)
TABL (Table impacté)	VARCHAR2(6)

Tableau 7 : Détail de la table des comptes-rendus de la recopie

Toutes les opérations sont donc consignées dans la nouvelle table nommée RECCOM. Celle-ci permet de stocker l'ensemble des opérations réalisées lors d'une recopie de paramétrage. Cette table est alimentée exclusivement par l'application et doit être exploitable pour de la restitution d'informations.

5.5 Conception

L'étape de conception permet d'obtenir une ébauche de l'IHM au travers de la modélisation de ces différents éléments. Les communications entre l'IHM et la couche métier sont mises en œuvre dans le même temps.

5.5.1 Modélisation de l'IHM

Dans cette logique de modélisation, la dynamique applicative est représentée au sein d'un diagramme d'activité stéréotypée « MacroActivity ». Il permet de modéliser l'IHM et la cinématique de l'application.

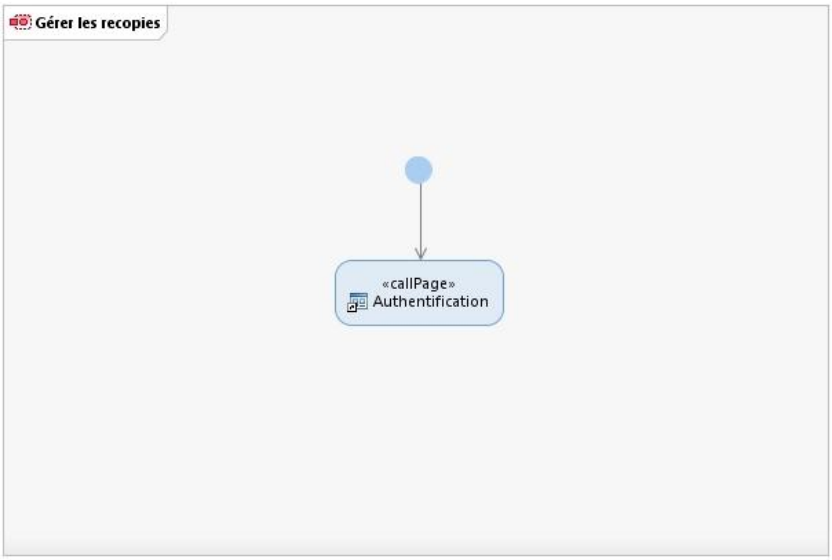

Figure 50 : Dynamique applicative

La modélisation d'un élément de l'IHM de l'application se traduit par la création d'une activité stéréotypée « page » qui est positionnée comme un élément du cas d'utilisation. L'application de recopie débute par l'affichage de la page d'authentification.

5.5.1.1 Authentification

La page d'authentification est un passage obligatoire dans cette application. Elle permet en effet de ne donner accès aux fonctionnalités de recopie qu'aux utilisateurs disposant des droits associés.

Au sein de cette page, on retrouve des champs saisissables permettant la saisie de l'identifiant et du mot de passe de l'utilisateur. Une liste déroulante permet de lister l'ensemble des bases de données disponibles et le bouton permet de lancer l'action relative au test d'authentification. L'utilisation de ces différents composants est illustrée dans le diagramme d'activité (Figure 52).

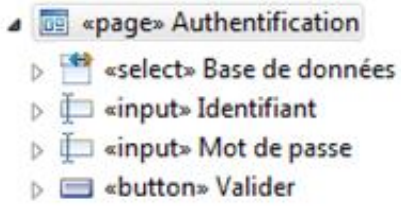

Figure 51 : Composants IHM de la page d'authentification

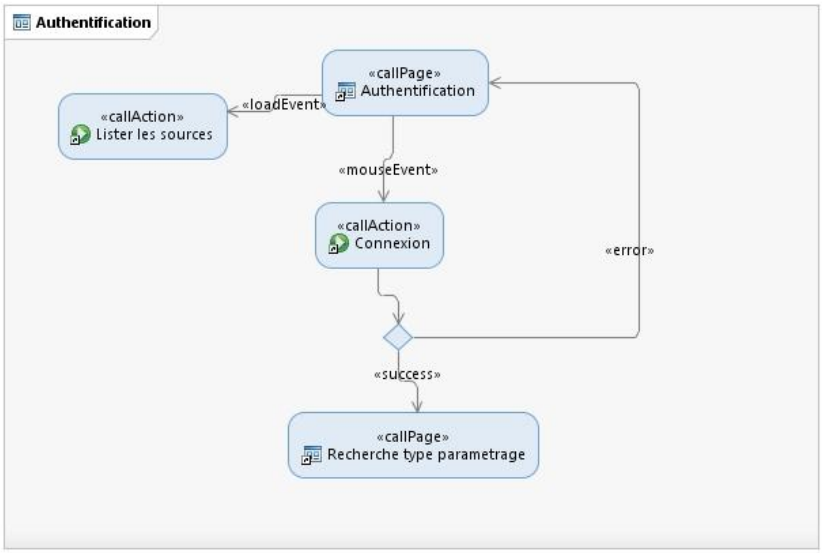

Figure 52 : Diagramme d'activité de l'authentification

L'appel de la page « authentification » entraîne le chargement de la liste des bases de données source accessible. La saisie des différents éléments nécessaires (identifiant et mot de passe) associé à un clic de souris entraîne le déclenchement de l'action correspond à la connexion. La réussite de cette connexion déclenche l'affichage de la page « Recherche type paramétrage » tandis que dans le cas d'un échec la page d'authentification est reproposée à l'utilisateur.

5.5.1.2 Recherche d'un type de paramétrage

Les fonctionnalités mises en œuvre dans cette application nécessitent la mise en place de possibilité de restreindre le champ de la recopie. Il est ainsi nécessaire de sélectionner un ou plusieurs éléments de paramétrage que l'on souhaite recopier sur une autre base de données. La page de recherche (Figure 53) permet de filtrer ces informations selon différents critères.

Au sein de cette page, on trouve des boutons permettant de faire une recherche, de réinitialiser les critères de sélection ou de se déconnecter de l'application. Les éléments permettant de réaliser les filtres sur les éléments de paramétrage sont aussi présents et sont regroupés au sein d'un même composant graphique. L'utilisation de ces différents composants est illustrée dans un diagramme d'activité (Figure 54).

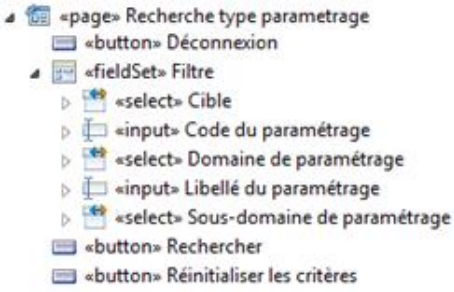

Figure 53 : Composants de la page de recherche de paramétrage

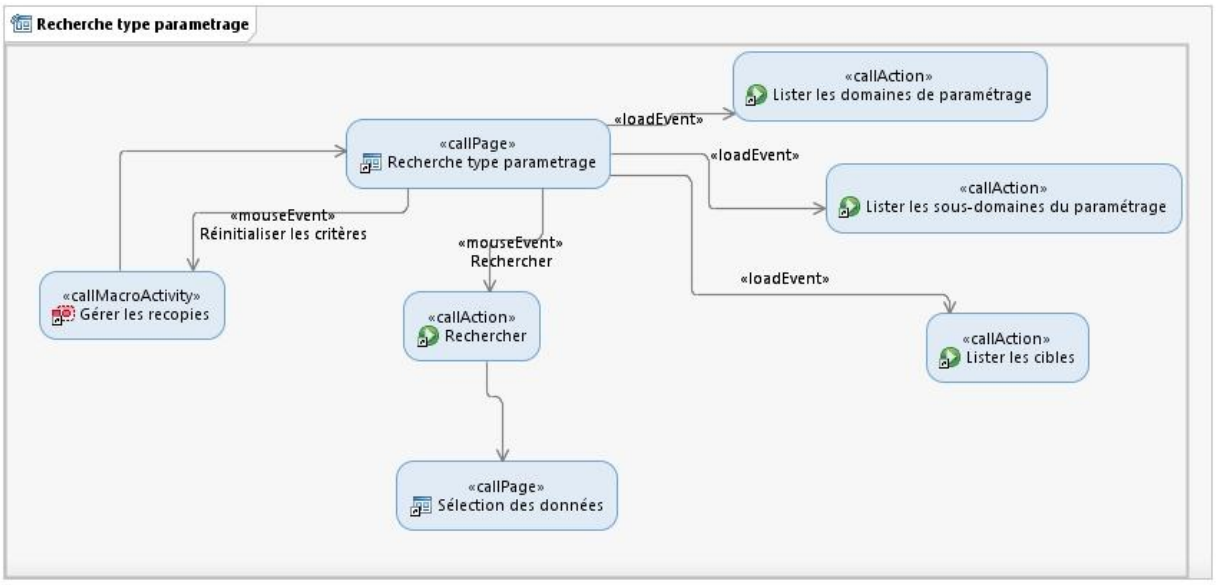

Figure 54 : Diagramme d'activité de la recherche de paramétrage

De nombreuses listes déroulantes sont présentes dans cette page et nécessitent le chargement des valeurs associées. L'action de réinitialisation des critères vide l'ensemble des saisies faites préalablement. L'action de recherche permet quant à elle l'affichage des éléments correspondant aux différents critères de filtre mis en œuvre et autorise alors la sélection des données.

5.5.1.3 Sélection du paramétrage

L'application de recopie du paramétrage donne la possibilité de sélectionner tout ou partie des résultats de la recherche.

Au sein de cette page, les nombreux boutons présents permettent d'accéder à différentes fonctionnalités :

- réalisation d'une recopie sur une base de données cible,
- génération de scripts,
- sélection de l'ensemble des éléments de paramétrage,
- désélection de l'ensemble.

Le résultat de la recherche précédente est présenté au sein d'un nouveau tableau de données au sein duquel la première colonne permet de sélectionner l'élément correspondant. L'utilisation de ces différents composants est illustrée dans un diagramme d'activité (Figure 56).

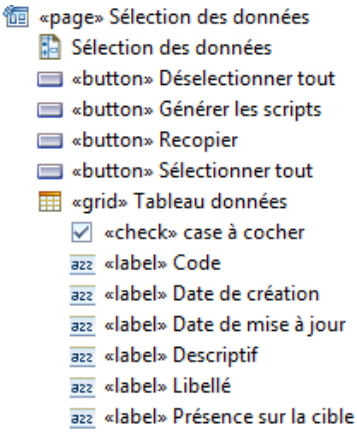

Figure 55 : Composants de la page de sélection de données

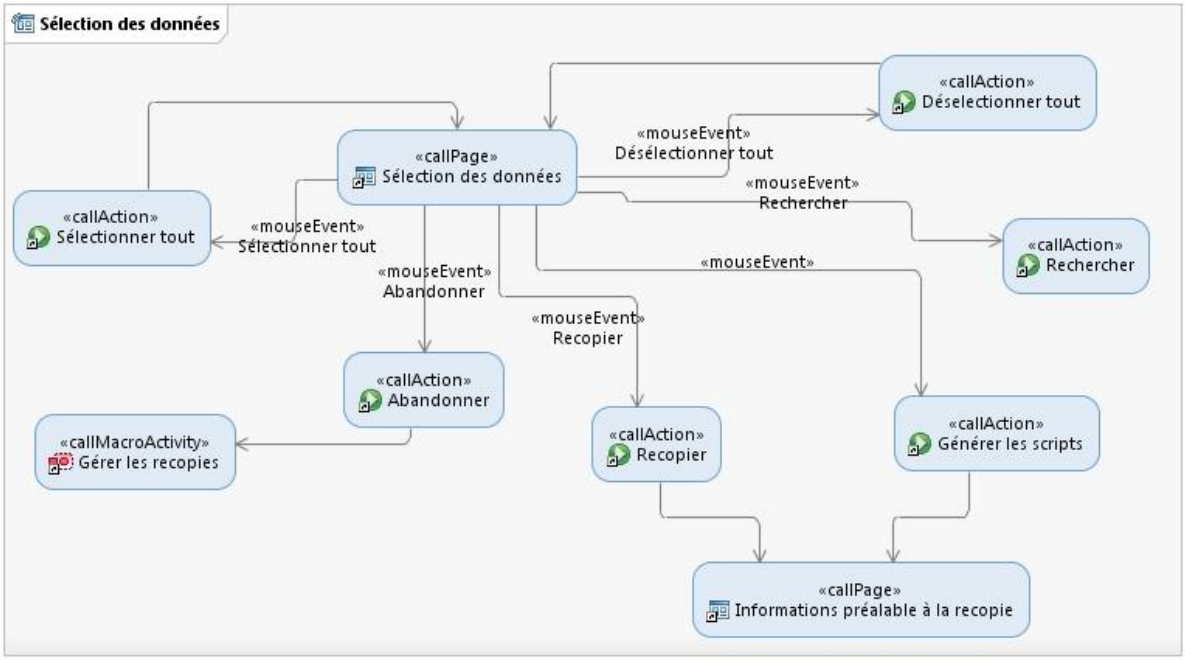

Figure 56 : Diagramme d'activité de la sélection du paramétrage

Au sein de cette page, l'utilisateur peut avoir de multiples interactions avec les éléments de paramétrage :

- l'action « Sélectionner tout » permet par exemple de cocher l'ensemble des résultats de la recherche,
- l'action « Désélectionner tout » représente son contraire,
- l'action « Rechercher » permet de relancer la recherche pour la prise en compte de nouveaux critères de recherche,
- l'action « Abandonner » permet d'abandonner la recopie de paramétrage initiée préalablement,
- l'action « Générer les scripts » donne l'accès à la fonctionnalité de génération de scripts et aboutit à l'affichage de la page d'informations préalables à la recopie,
- l'action « Recopier » donne l'accès à la fonctionnalité de recopie sur une base de données cible et aboutit à l'affichage de la page d'informations préalables à la recopie.

5.5.1.4 Informations préalables à la recopie

Dans le cadre d'une recopie d'éléments de paramétrage, il apparaît essentiel de conserver la maîtrise sur la nature des opérations à réaliser pour mener à bien cette opération. Cette page permet dans ce cadre de synthétiser l'ensemble des informations préalables à la recopie.

L'ensemble des informations préalables à la recopie est affiché au sein d'un nouveau tableau de données. La lecture de ces différents éléments doit permettre à l'utilisateur de statuer entre la nécessité d'abandonner ou de valider les opérations de recopie du paramétrage. L'utilisation de ces différents composants est illustrée dans un diagramme d'activité (Figure 58).

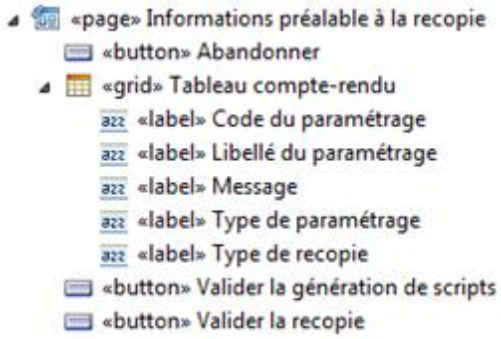

Figure 57 : Composants de la page d'informations préalable à la recopie

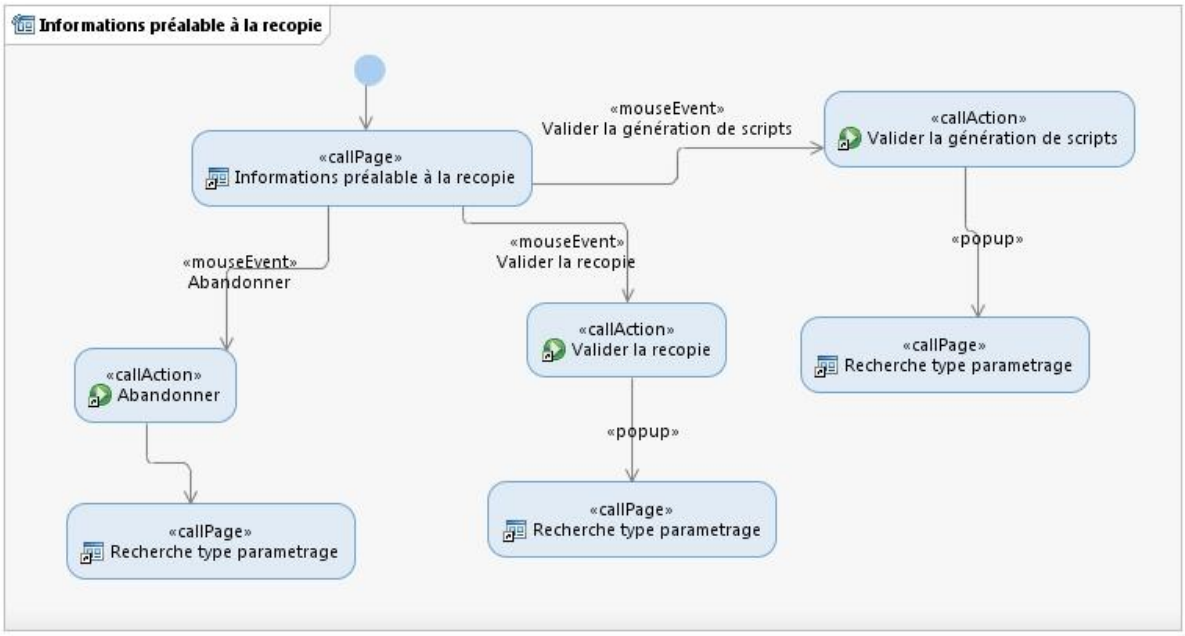

Figure 58 : Diagramme d'activité des informations préalable à la recopie

L'ensemble des actions choisies par l'utilisateur au sein de cette page permet de revenir à la page de recherche de paramétrage. Les traitements associés sont cependant bien différents en fonction du choix réalisé par l'utilisateur. La validation d'une recopie sur une base de données cible entraîne par exemple le déploiement des différents éléments de paramétrage sélectionnés. L'utilisateur est prévenu de la finalisation de ces opérations au travers d'une fenêtre secondaire (pop-up).

5.5.2 Modélisation des actions

Au sein de la vision MDA de GFI, une action est modélisée au travers d'un comportement opaque stéréotypée « action » et doit être positionnée comme un élément du cas d'utilisation. Il s'agit d'un traitement permettant de charger ou de recevoir les données d'une page. Elle est appelée lors du déclenchement d'un évènement de la couche présentation et permet de faire le lien entre la couche métier et l'IHM.

L'action permettant de rechercher les éléments de paramétrage (Figure 59) fait appel à l'interface de recherche. Le traitement passe ainsi de la page de recherche de paramétrage à la couche de présentation grâce à l'appel d'une interface stéréotypée « Business Delegate ».

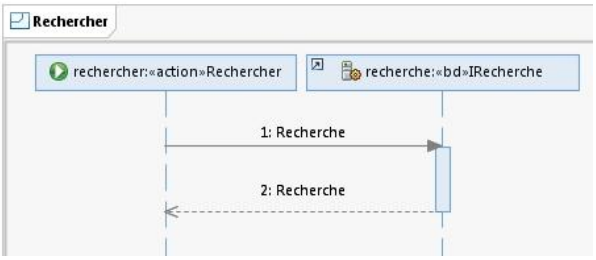

Figure 59 : Exemple de diagramme de séquence illustrant les actions

5.5.3 Appel des services métiers

Dans cette logique de modélisation, l'appel des services métiers est mis en place au travers de différentes interfaces stéréotypées.

5.5.3.1 Les interfaces « Business Delegate »

Les interfaces stéréotypées « Business Delegate » permettent d'appeler plusieurs services métiers à la fois. Elles sont utilisées généralement pour réduire le couplage entre la couche de présentation et la couche métier. La modélisation des appels effectués vers la couche métier est traduite au sein d'un diagramme de séquence.

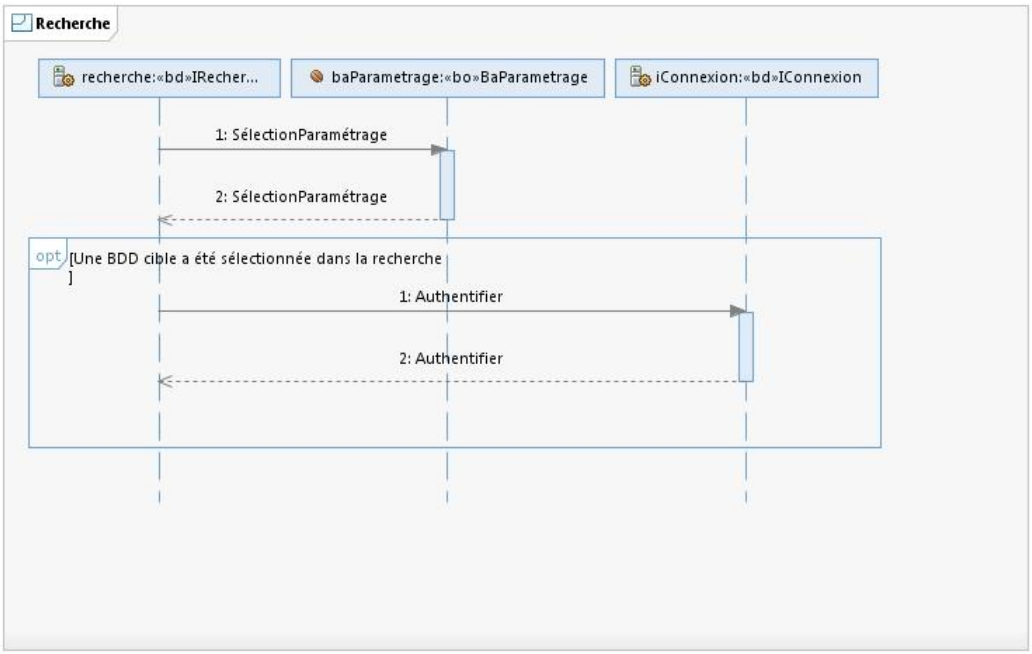

Figure 60 : Diagramme de séquence de l'interface de recherche de paramétrage

Au sein de la Figure 60, l'interface « recherche » permet de faire appel à la fois à l'interface représentant les éléments de paramétrage et selon le contexte (sélection d'une cible) à l'interface gérant l'authentification.

5.5.3.2 Les interfaces « Business Object »

Les interfaces stéréotypées « Business Object » permettent de faire le lien entre l'IHM et les entités métiers. Ces interfaces vont porter les algorithmes permettant de manipuler chacune des classes métiers.

L'interface « BaParametrage » permet de faire le lien avec les services de persistance de données (Figure 61). Ces différents éléments de paramétrage du progiciel sont modélisés au travers de classes stéréotypées « Data Object ». Le modèle de données du progiciel est ainsi transcrit en langage UML.

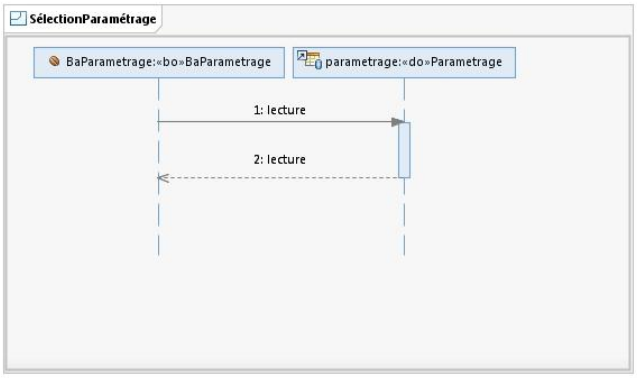

Figure 61 : Diagramme de séquence de sélection du paramétrage

5.5.4 Modélisation de la couche de persistance des données

La modélisation des entités métiers consiste à créer des classes stéréotypées « Data Object ». Les attributs associés à cette classe correspondent aux données qui devront être stockées sur un support tel qu'une base de données. Ces différentes entités sont représentées au sein d'un diagramme de classes.

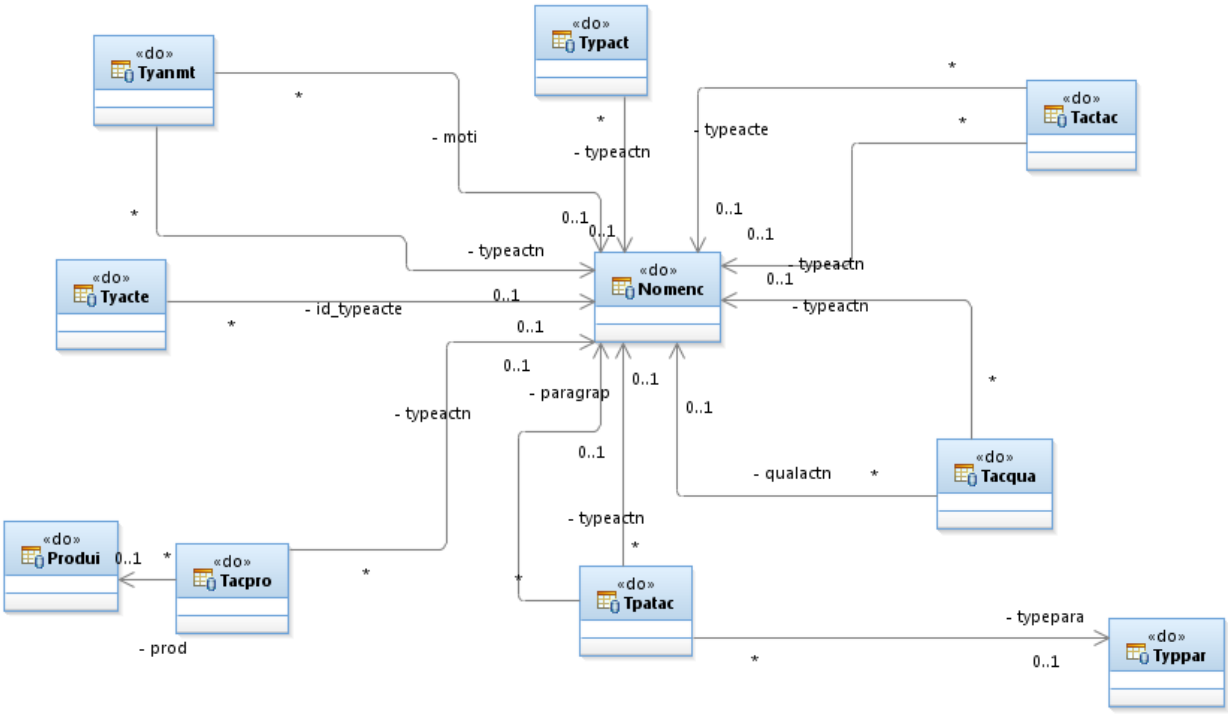

Figure 62 : Modélisation des tables associées au concept de type d'action

Le domaine relatif aux types d'action intègre de nombreux éléments de paramétrage. Les différentes classes sont liées entre elles par des associations de type dirigé permettant ainsi de stocker l'information relative aux clés étrangères. Cette modélisation est fortement exploitée par le générateur de code mis en place dans le cadre de ce projet.

5.6 La place de la démarche MDA dans le projet

La mise en œuvre de la démarche MDA dans le groupe GFI nécessite un investissement de départ plus ou moins important en fonction du niveau de modélisation. La prise de décision associée dépend fortement du retour sur investissement souhaité sur le projet et du coût initial jugé supportable. La démarche MDA couvre alors plus ou moins largement le champ de l'application.

5.6.1 Niveau de modélisation pour le projet

Dans le cadre de cette étude, le choix s'est porté sur le niveau de modélisation bronze. Le coût et le manque de disponibilités des équipes du CSN de Lille ont plaidé pour une limitation du champ de la démarche MDA à la couche de persistance des données. Le code applicatif correspondant est généré automatiquement et permet la manipulation de l'ensemble des objets métier.

Le niveau de modélisation étant plus restreint que les prévisions de départ, la modélisation de l'ensemble de l'application réalisée n'a pas servi pour la génération de code. Le développement de l'IHM et des communications internes à l'application ont ainsi dû être réalisés manuellement en tenant compte des spécificités du framework Astre Iodas. Cette découverte tardive a engendré un décalage dans la mise en œuvre des fonctionnalités attendues pour l'application.

5.6.2 Développement d'un générateur de code spécifique

Le développement du générateur de code au format Iodas nécessite de multiples connaissances dans le domaine de l'architecture technique et de la transformation de modèles. Celui-ci n'a pas ainsi pu être réalisé dans le cadre de cette étude et a nécessité l'intervention des intervenants du CSN de Lille. Les travaux consistant à la définition des besoins, la réception et les tests de qualification du développement réalisé sont quant à eux rentrés dans les réalisations effectuées.

Le générateur de code intègre les spécificités du framework Astre Iodas relatives à la couche de persistance des données. Cette livraison est composée :

- d'un métamodèle permettant la modélisation des tables IODAS en tant que classes UML,
- des templates de transformation de modèle permettant d'obtenir à la fois de la génération de code et de documentation.

La génération des quatre composants nécessaires (Ba, Bm, Bs et Bc) pour le bon fonctionnement de l'application autorise ainsi la mise en œuvre d'un prototype fonctionnel réutilisable dans la suite du projet.

5.7 Présentation du prototype fonctionnel

L'ampleur de la tâche et le timing limité ont entrainé la mise en œuvre d'un prototype fonctionnel visant à prouver l'intérêt de la démarche MDA dans le développement de cette application métier. Il se concentre sur une fonctionnalité (recopie sur une base de données cible) et sur un périmètre restreint mais représentatif des différents concepts de paramétrage (type d'action). Il permet d'instancier le modèle, de tester les différentes règles de gestion définies et de dérouler les scénarios d'utilisation de l'application cible.

5.7.1 Spécifications fonctionnelles

La mise en œuvre d'une fonctionnalité de recopie de paramétrage oblige à définir des règles de gestion spécifiques à chacun des concepts de paramétrage (plus de soixante au total). Les différents éléments de paramétrage doivent en effet être manipulés dans un ordre précis du fait de leur forte corrélation. En outre, il est nécessaire de définir le niveau de dépendance entre chaque concept de paramétrage et ses éléments associés.

5.7.1.1 Les modes de recopie

Dans le but de définir des règles de gestion métier permettant la manipulation correcte de ces éléments de paramétrage, trois modes de recopie ont été définis :

- Déclenchement d'un bloc fonctionnel secondaire :
    - En cas d'absence de l'élément secondaire sur la base de données cible, la recopie de l'élément principal entraîne le déploiement de l'élément secondaire.
    - En cas de présence de l'élément secondaire sur la base de données cible, la recopie de l'élément principal entraîne la mise à jour éventuelle de l'élément secondaire.
- Avertissement :
    - En cas d'absence de l'élément secondaire sur la base de données cible, le lien entre l'élément principal et l'élément secondaire n'est pas répercuté mais n'empêche pas la recopie d'aboutir.
    - En cas de présence de l'élément secondaire sur la base de données cible, le lien entre l'élément principal et l'élément secondaire est répercuté mais l'élément secondaire n'est pas mis à jour.

- Blocage :
    - En cas d'absence de l'élément secondaire sur la base de données cible, la recopie sur la cible n'aboutit pas.
    - En cas de présence de l'élément secondaire sur la base de données cible, le lien entre l'élément principal et l'élément secondaire est répercuté sur la cible mais l'élément secondaire n'est pas mis à jour.

Pour illustrer ces différentes notions, nous prenons l'exemple de la recopie d'un type d'action (Tableau 8) qui nécessite la manipulation potentielle de nombreux éléments de paramétrage d'horizons différents.

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Types d'action	Paramètres Motifs d'interruption/annulation Qualités dans l'action		Types d'acte Produits

Tableau 8 : Spécifications de la recopie d'un type d'action

Les paramètres, les motifs d'interruption/annulation et les qualités dans l'action feront par conséquent l'objet d'une intégration sur la base de données cible lors de la recopie d'un type d'action. La présence des types d'acte et des produits associés sur la base de données cible est par contre indispensable pour la finalisation de cette opération. Ces notions sont accompagnées d'une logique d'agencement permettant de maîtriser l'ordre de recopie de ces différentes notions.

5.7.1.2 Algorithme récursif d'agencement

La logique d'agencement des différents éléments de paramétrage est stockée au sein d'une nouvelle table du modèle de données du progiciel. Ce mode de fonctionnement permet d'avoir une interaction plus facile avec ces éléments en cas d'évolutions mais nécessite l'écriture d'un algorithme récursif.

La recopie de certains éléments de paramétrage nécessitent en effet la présence préalable d'autres éléments (dépendance ascendante). Pour exemple, la recopie d'un paramètre nécessite la présence préalable de la nature de paramètre associée.

5.7.1.3 Algorithme de manipulation des objets métier

Dans l'objectif d'une programmation optimum, la manipulation des objets métier doit être commune à tous les éléments de paramétrage. Un algorithme de manipulation des objets métier couvrant l'ensemble des spécificités de chaque élément doit donc être défini dans l'application de recopie. Le code résultant est générique et donc plus facilement maintenable au fil des évolutions.

5.7.2 Description du prototype

La mise en œuvre de ce prototype fonctionnel répond à un double objectif. Il permet tout d'abord de démontrer l'efficacité et les apports de la démarche MDA dans un contexte industriel. Il permet ensuite de confirmer que les fonctionnalités mises dans cette application sont conformes aux besoins.

5.7.2.1 Authentification

Comme évoquée précédemment, l'application développée dans le cadre de cette étude doit s'intégrer pleinement dans le progiciel Iodas. Cette intégration permet la réutilisation d'éléments déjà mis en œuvre. La gestion de l'authentification (Figure 63) est ainsi héritée de l'intégration dans le progiciel.

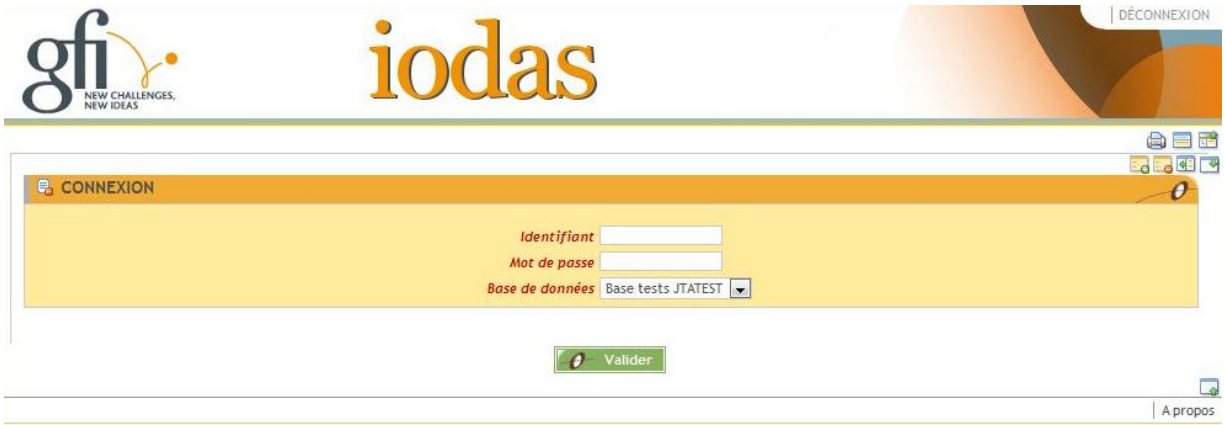

Figure 63 : Ecran d'authentification

Dans le cas d'une authentification correcte et d'une habilitation en cohérence avec les fonctionnalités proposées, l'application propose deux sous-menus correspondant aux fonctionnalités principales décidées :

- Recopier sur une cible,
- Génération de scripts.

5.7.2.2 Recherche d'un type de paramétrage

Dans les deux fonctionnalités de l'application, la recherche du type de paramétrage (Figure 64) que l'on souhaite recopier correspond à un passage obligatoire et nécessite de définir les critères de recherche associés.

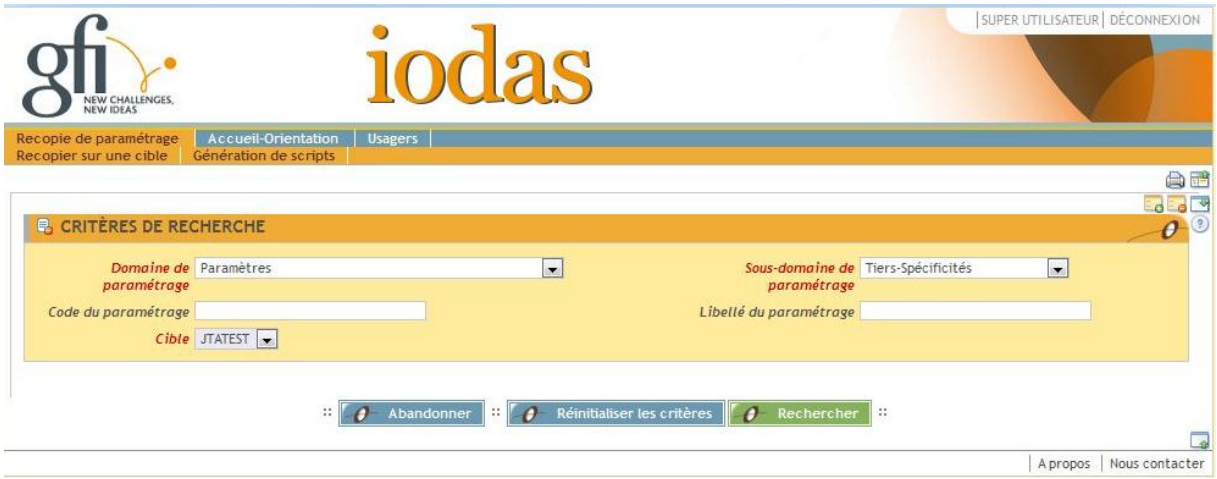

Figure 64 : Page de recherche de paramétrage

Au sein de cette page de recherche d'un type de paramétrage, l'utilisateur dispose de plusieurs choix :

- la sélection du domaine de paramétrage,
- la sélection éventuelle d'un sous-domaine de paramétrage,
- la sélection d'une base de données cible,
- le filtre des résultats en agissant sur le code et/ou le libellé du paramétrage,
- le filtre des résultats en agissant sur le libellé du paramétrage.

La recherche selon les critères définis permet alors de visualiser un ensemble d'éléments de paramétrage à recopier.

5.7.2.3 Sélection du paramétrage

L'ensemble des éléments répondant aux critères de sélection prédéfinis sont alors affichés en tant que nouveau tableau dans la page de l'application. L'information concernant la présence de chaque élément sur la base de données cible permet de disposer d'un premier niveau d'informations à ce stade de la recopie.

Figure 65 : Ecran de sélection du paramétrage

L'utilisateur peut sélectionner un ou plusieurs éléments de paramétrage à recopier (Figure 65). Cette opération entraîne alors l'affichage d'informations préalables à la recopie correspondant à l'ensemble des opérations nécessaires pour aboutir à la finalisation de la recopie du paramétrage.

5.7.2.4 Informations préalables à la recopie

L'affichage des informations préalables à la recopie entraîne l'apparition d'un nouveau tableau dans la page listant l'ensemble des opérations à réaliser sur la base de données cible :

- insertion d'un nouvel élément de paramétrage sur la base de données cible,
- mise à jour d'un élément de paramétrage existant sur la base de données cible,
- suppression d'un lien entre deux éléments de paramétrage existant sur la base de données cible et non présent sur la base de données source.

En fonction des informations trouvées sur la base de données cible, deux cas de figure différents peuvent alors apparaître :

- certains éléments de paramétrage indispensables sont absents de la base de données cible, la recopie du paramétrage est considérée comme impossible à finaliser,
- tous les éléments de paramétrage indispensables sont présents et la recopie du paramétrage est considérée comme possible à finaliser.

5.7.2.4.1 Cas d'une recopie impossible

Dans le cas d'une recopie de paramétrage impossible, les éléments bloquants sont affichés en rouge et nécessitent une action de l'utilisateur.

Figure 66 : Cas d'une recopie impossible

La recopie du type d'action (Figure 66) ne peut pas être validée en raison de l'absence d'un produit associé et considéré comme indispensable pour la finalisation des opérations. Ce cas de figure correspond concrètement aux spécifications mises en place pour la gestion de la recopie d'un type d'action. Les produits associés sont en effet considérés comme bloquants pour la recopie de ce concept de paramétrage.

L'utilisateur doit donc préalablement réaliser une recopie du produit concerné avec l'application. Suite à cette opération, la recopie de ce type d'action est alors considérée comme possible.

5.7.2.4.2 Cas d'une recopie possible

Dans le cas d'une recopie de paramétrage possible, l'utilisateur dispose de la possibilité de valider la recopie et ainsi de déployer les différents éléments de paramétrage sur la base de données cible.

Figure 67 : Cas d'une recopie possible

Dans ce cas (Figure 67), la validation de la recopie du type d'action est considéré comme possible. Cette opération nécessite le déploiement de différents éléments de paramétrage dans l'ordre suivant :

- mise à jour des éléments relatifs au type d'action sélectionné (tables NOMENC et TYPACT),
- suppression du lien avec le paramètre trouvé sur la base de données cible et non présent sur la base de données source (table TPATAC),
- insertion du motif d'interruption/annulation (table NOMENC),
- insertion du lien entre le type d'action et le motif d'interruption/annulation (table TYANMT).

Le résultat de ces différentes opérations permet alors de disposer d'un paramétrage identique pour ce type d'action entre les deux bases de données.

5.7.3 Perspectives

La réalisation d'un prototype fonctionnel permet de disposer d'une vision plus concrète de la réalité d'une industrialisation de la démarche MDA. Il est important de capitaliser sur cette expérience réussie pour ce projet en cours mais aussi pour l'avenir.

Chapitre 6 : Bilan et perspectives de cette étude

Ce dernier chapitre permet de dresser le bilan de cette étude et d'évoquer les différentes perspectives associées. Dans un premier temps, nous réalisons le bilan de l'utilisation de la démarche MDA dans un contexte industriel. Dans un second temps, nous évoquons l'avenir du projet initié dans le cadre de cette étude. Dans un troisième temps, nous analysons les perspectives associées à l'utilisation de cette démarche dans une société éditrice de logiciels. Enfin, nous terminons ce chapitre par un bilan personnel.

6.1 Bilan de l'utilisation de MDA dans un contexte industriel

Dans le domaine industriel et plus particulièrement dans le secteur de l'édition de logiciels, la course à la rentabilité est une problématique fondamentale dans la survie des entreprises. Chaque société fait face à une concurrence accrue qui oblige à trouver de nouvelles solutions pour l'amélioration de la productivité.

Dans ce cadre, les leviers d'optimisation de la rentabilité sont présents dans différents domaines. L'externalisation des développements permet par exemple de disposer d'une main d'œuvre à coût réduit. Le changement des outils de développement peut permettre une augmentation de la productivité des équipes. Enfin, la modification des méthodologies de travail (démarches agiles, MDA, etc.) est aussi un moyen de tendre vers une meilleure rentabilité dans les projets informatiques.

6.1.1 Des gains indéniables

Grâce à son architecture de modélisation, toutes les mises en œuvre industrielles de la démarche MDA apportent d'importants gains en permettant de capitaliser les efforts à tous les niveaux :

- la pérennité du savoir-faire afin d'éviter de recommencer la modélisation des fonctionnalités et du comportement du système chaque fois qu'une nouvelle technologie est adoptée,
- les gains de productivité afin de permettre au développeur de réduire les coûts de mise en œuvre des applications nécessaires à son métier,
- la prise en compte des plates-formes d'exécution afin de permettre le déploiement d'une même application sur plusieurs plates-formes.

L'utilisation de la démarche MDA par les grands pontes de l'industrie du logiciel (IBM, Microsoft) n'est donc pas simplement due à un effet de mode ou à un attrait pour la nouveauté. Aucune plate-forme universelle n'existe cependant pour sa mise en œuvre dans les entreprises (Bouara, Barbouche, & Kouider El Ouahed, 2008) et les visions de la démarche peuvent être différentes d'une société à l'autre.

6.1.1.1 Une meilleure productivité

La démarche MDA court après l'objectif d'une succession de transformations de modèles techniques et fonctionnels aboutissant à la génération automatique du code de l'application. Pour y parvenir, le coût initial de mise en place de la plate-forme ne doit pas être négligé. Il s'amortit généralement sur les premiers cas d'utilisations si le projet est de taille suffisamment importante voire sur le premier projet. Dans l'exemple de cette étude, les coûts de mise en place du générateur de code variaient fortement en fonction du niveau de modélisation souhaité :

- environ 3 jours pour une modélisation de niveau « bronze » (exploitation de la couche de persistance de données),
- environ 10 jours pour une modélisation complète des différentes couches de l'application.

Dans cette démarche, l'ensemble du code généré n'est donc plus à écrire par le développeur et la quantité de code se retrouve réduite grâce à la factorisation déduite de ces différents modèles. Les charges de développement sont ainsi maîtrisées et les gains sont substantiels.

Dans l'exemple sélectionné (Figure 68), deux équipes de développement ont réalisé le même projet avec des démarches différentes :

- centrée sur les modèles,
- centrée sur le code.

L'effort estimé au préalable est sensiblement le même pour les deux démarches. Au niveau de la réalisation, l'utilisation d'une approche centrée sur les modèles permet un gain de productivité de l'ordre de 35%, diminue le temps de développement et encourage l'utilisation de patrons de conception. Le retour sur investissement de cette conception se situe ainsi à tous les niveaux de la fabrication d'un logiciel.

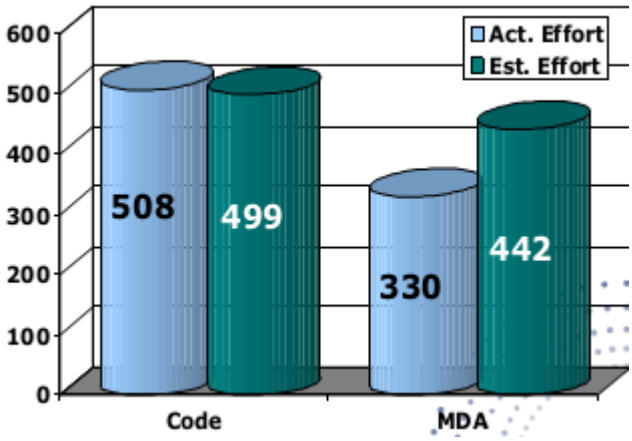

Figure 68 : Différences entre l'effort estimé et réalisé pour deux démarches de développement (Staron)

Cette démarche dispose en outre de l'avantage d'être parfaitement adaptée à des projets dont le développement est externalisé. L'ensemble du cadre de l'application est généré en exploitant les modèles et seul le code technique non modélisé reste à écrire par les équipes de développement. Le niveau de puissance descriptive du code est quant à lui augmenté et le volume de « colle » est fortement diminué. Le code restant à écrire par le développeur représente alors une valeur ajoutée fonctionnelle.

6.1.1.2 Une meilleure qualité

Une mise en œuvre efficace de la démarche MDA assure aussi une meilleure qualité de code et donc de meilleures performances de l'application associée. Dans un contexte idéal, le code généré automatiquement est :

- correct : il ne comporte pas d'erreur de syntaxe,
- efficace : le code généré n'implique pas de pertes de performances et n'ajoute pas d'anomalies supplémentaires à l'application,
- fonctionnel : il a été validé par des architectes logiciels par une démonstration de faisabilité (POC[40]),
- maintenable : il répond aux standards de qualité définis dans les conventions de codage.

[40] Prouf Of Concept

Cette qualité de code est cependant fortement dépendante de la qualité des modèles. Au même titre que le code peut avoir une note de qualité (par exemple dans Sonar), une note de qualité du modèle peut être envisagée. Des contrôles sur les modèles doivent dans tous les cas être réalisés en amont de toute génération du code pour éviter les différents retours entre la conception et le développement. Dans cette logique, la notion de « checkstyle » vérifiant la qualité de l'expression fonctionnelle a été implémentée dans la vision GFI de cette démarche. Elle n'autorise la génération de code que dans le cas où les modèles répondent correctement aux différents critères mis en place :

- documentation associée,
- conformité au méta-modèle,
- absence de modèle non utilisé.

De plus, le code produit par les différentes transformations de modèles est homogène et répond ainsi parfaitement aux normes et bonnes pratiques du développement. Toute optimisation est immédiatement utilisée par l'ensemble de l'application. De plus, la fiabilité du code technique est éprouvée par les architectes logiciels et constitue une préoccupation de moins pour le développeur. Dans le cas de notre prototype fonctionnel, la génération de la couche de persistance de données de code assure une totale homogénéité du code et aussi une totale cohérence au fonctionnement du framework spécifique Astre.

6.1.1.3 Une pérennité garantie

L'indépendance des modèles vis-à-vis de l'évolution de la technique (PIM) permet de conserver tout le savoir-faire métier. Dans les applications informatiques, il s'agit en effet de la notion la plus stable dans le temps. Les technologies utilisées correspondent quant à elles à des notions beaucoup plus sensibles et évolutives. Le fondement de la démarche MDA concerne la séparation des préoccupations techniques et fonctionnelles et permet dans ce cas de capitaliser sur le métier d'une part et sur l'aspect technique d'autre part. De plus, la compréhension du modèle plus aisée sans la connaissance de la technologie permet une meilleure maintenabilité de ces différents éléments.

Dans le cas de l'application de cette étude, les modèles PIM représentent les tables accueillant les éléments de paramétrage du progiciel. Le modèle de données évolue par fines touches au fur et à mesure des changements de version et sont donc associées à de nouvelles fonctionnalités dans le progiciel. Les répercussions sur la modélisation seront donc infimes en rapport à l'effort fourni dans le cadre de ce projet. Le plus important reste surtout l'indépendance de ces modèles vis-à-vis d'un gestionnaire de base de données. Des évolutions de version ou un changement radical d'outils n'auront ainsi aucun impact sur la conformité de ces modèles.

6.1.1.4 Une grande réactivité

Dans un projet d'application, les besoins en termes d'évolution fonctionnelle peuvent apparaître à tout moment du cycle de développement. La démarche MDA est parfaitement adaptée à l'évaluation rapide des différentes répercussions de celle-ci car elle répond aux différentes valeurs et principes de l'agilité dans les projets informatiques (17 experts du développement logiciel, 2001).

Principes de l'agilité	Démarche MDA
Satisfaire le client est la priorité	L'énergie est concentrée sur le métier et non sur la technique.
Accueillir les demandes de changement « à bras ouverts »	La génération de code à cycles courts permet des remises en cause, même tardives.
Livrer le plus souvent possible des versions opérationnelles de l'application	Le processus d'analyse est intrinsèquement itératif et incrémental.
Assurer une coopération permanente entre Client et Equipe projet	L'analyste/développeur est l'interlocuteur direct de l'utilisateur sans intermédiaire.
Construire des projets autour d'individus motivés	L'analyste/développeur est la personne en charge.
Privilégier la conversation en face à face	La plupart des informations sont recueillies lors de séances de travail communes et sont stockées dans le modèle.
Mesurer l'avancement du projet en termes de fonctionnalités de l'application	L'application fonctionnelle est générée en permanence.
Faire avancer le projet à un rythme soutenable et constant	On adresse la complexité dès le début en allant rapidement en profondeur.
Porter une attention continue à l'excellence technique et à la conception	L'énergie est dépensée essentiellement au service du métier : l'accessoire est généré.
Favoriser la simplicité	La conception est générée, la plate-forme de génération garantit l'homogénéité qualité.
Responsabiliser les équipes : les meilleures architectures, spécifications et conceptions émergent d'équipes auto organisées.	La polyvalence de l'analyse impose une hiérarchie « plate »
Ajuster, à intervalles réguliers, son comportement, ses processus pour être plus efficace	Pas de réponse spécifique (faire des rétrospectives)

Tableau 9 : Les réponses de la démarche MDA aux principes de l'agilité

Dans cette démarche, le cycle de développement très court et la proximité de l'analyste facilitent le retour des différents utilisateurs et renforcent ainsi l'acceptabilité des applications développées.

6.1.1.5 Une uniformisation de la modélisation

L'application de la démarche MDA apporte une certaine uniformisation de la modélisation sur les différents projets du groupe. Tous les concepteurs logiciels formalisent dorénavant les besoins métiers ou techniques à l'aide du langage UML et des profils spécifiques mis en œuvre (PIM, IHM, etc.).

Cette généralisation à tous les projets permet en outre d'envisager une capitalisation du savoir-faire. Des éléments correspondant aux spécificités d'une architecture et d'une technologie pourront par exemple être réutilisés dans les modèles d'architecture d'un autre projet. La modélisation de services communs à différentes applications (la gestion de l'authentification par exemple) peut aussi faire l'objet d'une réutilisation, même partielle, dans les modèles d'analyse et de conception. L'investissement initié au départ est fortement rentabilisé au fur et à mesure des projets et permet d'envisager un gain significatif dans les projets suivants.

En ce qui concerne l'application de cette étude, les éléments correspondant à la couche de persistance des données et à l'architecture pourront faire l'objet d'une réutilisation simple dans les projets futurs de l'entité Progiciels.

6.1.1.6 Un apprentissage facilité

La pratique actuelle de l'IDM demande de la part des développeurs un niveau de formalisme élevé en raison des niveaux d'abstraction concernés : manipulation de graphes, abstractions multi-domaines, méthodes formelles (Canals, Millan, & Rault, 2011). Le déploiement des différents concepts associés à la démarche MDA nécessite par conséquent un apprentissage conséquent pour les concepteurs quel que soit le langage de modélisation utilisée (UML ou DSL).

La solution GFI prend quant à elle le parti de fortement simplifier l'apprentissage des différents intervenants. L'absence de l'utilisation de langages de contraintes (OCL) ou d'actions (AS), jugés trop complexes pour un apprentissage rapide, tend en effet à raccourcir les besoins de formation des concepteurs logiciels. Celui-ci se résume en effet à l'utilisation du langage UML dans les outils déployés et varie de 0,5 à 5 jours selon le niveau de modélisation sélectionné pour le projet. La prise en main et la compréhension du fonctionnement de la démarche MDA s'en trouve ainsi fortement facilitée et beaucoup plus accessible.

6.1.2 Des raccourcis dommageables

La confrontation entre la production et le coût est une problématique importante dans le domaine du développement du logiciel. La refonte des processus de production doit forcément en tenir compte même si cela doit entraîner l'abandon de fonctionnalités jugées comme pertinentes et intéressantes pour la vie d'un projet. Dans le domaine des sociétés de services en informatique, le meilleur rendement économique du projet reste la référence absolue pour adouber un nouveau mode de fonctionnement.

6.1.2.1 Une logique de modélisation minimaliste

Au sein de la démarche prônée par le groupe GFI, la logique de modélisation reste minimaliste et se cantonne à l'utilisation du langage UML agrémenté de quelques profils spécifiques. Le code de l'application ne peut pas être généré en totalité et le taux de couverture varie selon la couche :

- la totalité de l'architecture,
- de 70 à 90% de la couche de persistance des données,
- de 30 à 100% de la couche métier,
- de 80 à 100% de la couche IHM.

L'absence d'utilisation de langages de contrainte dans la modélisation nécessite de conserver des spécifications non modélisables et forcément non transformables. Cette logique de modélisation minimaliste permet de simplifier le travail des concepteurs qui constituent une denrée rare dans les sociétés de services informatiques. Les développeurs sont quant à eux plus nombreux et leur travail représente un coût moindre. Ce raisonnement implique donc de conserver une part de développement humain dans les projets d'applications.

Cependant, se restreindre à ce niveau de la démarche MDA implique de perdre tout un pan important de cette démarche. La génération de code est loin d'être exhaustive et pose la question de la maintenance de ces modèles. Ceux-ci ne doivent pas seulement servir à la génération du cadre de l'application en déployant les fichiers permettant au développeur de réaliser le code de celle-ci. Dans cette logique de modélisation ne couvrant pas totalement le périmètre de l'application, il peut être parfois tentant de laisser de côté cet aspect « modèles » dans des phases suivantes du cycle de vie du logiciel telles que la maintenance évolutive.

6.1.2.2 La problématique des applications multiplateforme

Dans le cas d'une application utilisable sur de multiples plateformes, la modélisation indépendante de toute plateforme (PIM) reste unique. Les transformations réalisées avec de multiples modèles des architectures à déployer (PDM) doivent aboutir à la génération d'autant de modèles dépendant de la technologie (PSM) que de plateformes souhaitées pour l'application. La démarche MDA permet ainsi de générer et de maintenir plusieurs applications aux fonctionnalités identiques mais évoluant sur des plateformes et avec des technologies différentes. Le réseau social Facebook est par exemple disponible au travers d'une application Web et d'une application mobile sur des supports divers (Android, OIS, Windows Mobile). Toutes ces applications disposent cependant des mêmes fonctionnalités de partage et de communications.

Dans ce cas de figure, la vision GFI de la démarche MDA peut s'avérer problématique. Les développements manuels induits par une modélisation restreinte seront en effet nécessaires sur toutes les plateformes permettant de faire tourner l'application. Le travail initial de développement sera ainsi multiplié et toutes les évolutions fonctionnelles relatives à la maintenance évolutive devront aussi faire l'objet de prise en compte sur toutes les plateformes.

6.1.2.3 La perte de plasticité au niveau de l'IHM

Au sein de la vision GFI de la démarche MDA, la notion de PSM est fortement délaissée au profil de la notion de PIM annoté. Ce fonctionnement n'est cependant pas spécifique à ce groupe car la maintenance des modèles PSM implique de prévoir des opérations de rétro-ingénierie généralement coûteuses mais permettant de revenir au niveau de modélisation supérieur. Le principe mis en œuvre nécessite donc de stéréotyper les éléments associés à l'IHM au sein de ces modèles PIM :

- page,
- bouton,
- pop-up,
- évènements de souris,
- évènements de clavier, etc.

Le cycle de développement est ainsi raccourci par rapport à la vision théorique de la démarche MDA. Le code de l'application est directement issu de la transformation des modèles PIM agrémentés des informations relatives à l'architecture et aux technologies exploitées (PDM).

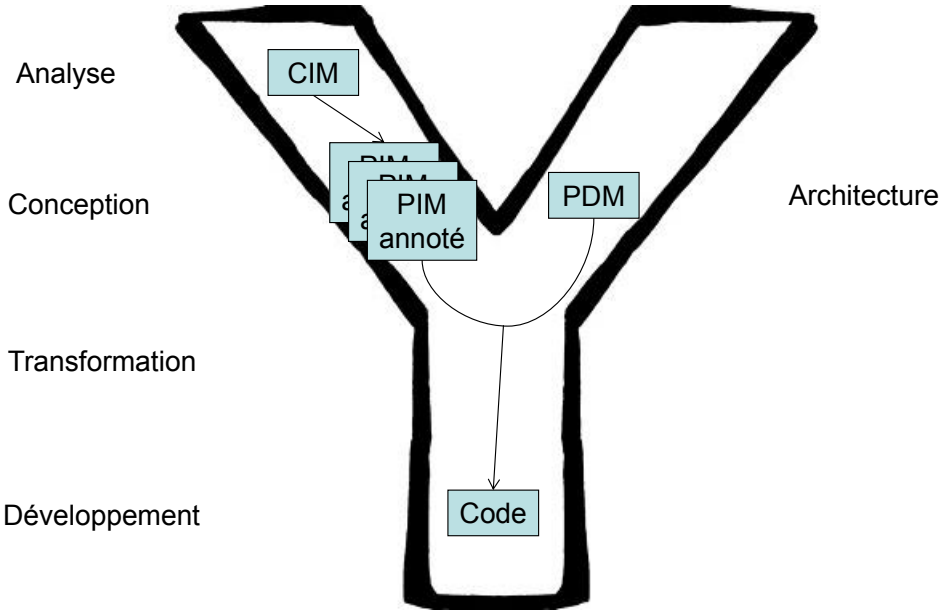

Figure 69 : La vision GFI de la démarche

Ce fonctionnement particulier implique cependant une perte de plasticité au niveau de l'IHM. Celle-ci correspond à la capacité d'une IHM à s'adapter à son contexte d'usage (<utilisateur, plate-forme, environnement>) dans le respect de son utilisabilité.

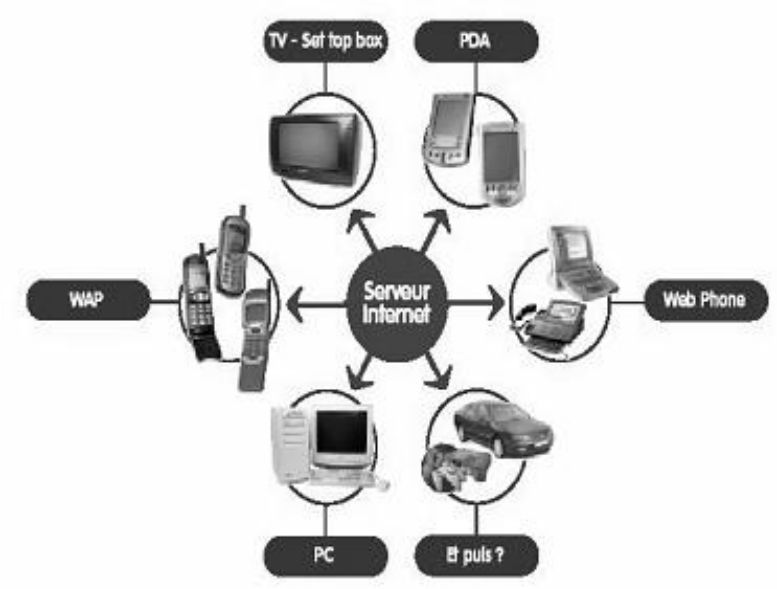

Figure 70 : Diversité des contextes d'utilisation

Dans le fonctionnement prôné dans le groupe, la capacité d'adaptation aux contraintes matérielles et environnementales n'existe ainsi pas. L'IHM ne fait pas l'objet d'une modélisation à part entière et chaque nouvelle plateforme implique le développement de nouveaux stéréotypes « IHM » correspondant aux spécificités de celle-ci.

6.1.2.4 La question de la qualité de l'application

La logique économique omnipotente dans les sociétés de services informatiques implique que tout investissement réalisé doit justifier d'un retour significatif. Il est toutefois dommage que cette logique se fasse régulièrement au détriment de la qualité du logiciel. La génération automatique de code sera en effet toujours plus performante que la programmation humaine. « En informatique, le problème se situe le plus souvent entre la chaise et le clavier ».

Malgré ces raccourcis, les mesures effectuées sur la qualité du code dénotent une réelle amélioration de celui-ci par rapport à des projets n'implémentant pas la démarche MDA. La logique pourrait simplement être poussée un peu plus loin et permettre d'arriver à un niveau plus proche du zéro défaut. Cet argumentaire pourrait alors justifier un coût de réalisation légèrement plus élevé pour le projet mais sous-entend une implication plus importante des concepteurs logiciels. La modélisation de l'application serait en effet nécessairement beaucoup plus complète.

6.1.3 Synthèse

La mise en œuvre de la démarche MDA au sein du groupe GFI correspond à une vision épurée et donc industrialisée de cette nouvelle démarche. Les notions les plus complexes sont laissées de côté pour obtenir des résultats plus rapidement et simplement. Les objectifs en termes de gains sur la qualité de l'application et sur la productivité sont remplis et valident ainsi l'utilité de cette démarche. Les objectifs de la démarche MDA sont aussi atteints car la génération de code est largement supérieure à 50% de l'application. En dessous de ce seuil, on se situe alors plus dans de la génération « naïve » de code et on n'est pas « centré sur les modèles » mais on reste « centré sur le code ».

On peut cependant constater avec regrets que l'ensemble des projets ne peuvent bénéficier de ces différents apports. Le retour sur investissement associé à cette démarche ne se situe en effet que sur des projets conséquents (supérieur à 500 jours – homme) ou sur des projets disposant d'une architecture complexe. Tous les projets du groupe ne sont donc pas labellisés « démarche MDA ».

La mise en œuvre de cette démarche reste encore jeune et peu connue par les différentes entités du groupe. Chaque projet permet d'avancer dans sa visibilité dans le groupe et dans sa couverture des différentes technologies et architecture. L'étude de ce document a par exemple permis de faire connaître cette démarche dans l'entité Progiciels et d'intégrer des spécificités d'un framework utilisé dans de nombreux progiciels en technologie Web du groupe.

Cette nouvelle démarche ne peut cependant être vue comme un remède à tous les maux dans le domaine du développement du logiciel. Un projet difficilement réalisable ne pourra être sauvé par l'introduction de cette nouvelle méthodologie de travail. A l'inverse, les amalgames faciles sur l'impact de l'utilisation de cette démarche sur l'échec d'un projet applicatif doivent être évités.

6.2 L'avenir du projet

Le projet initié correspondait au départ à un simple exemple d'application de la démarche MDA dans un contexte industriel. L'expérience accumulée dans l'entité Progiciels a permis de produire un prototype fonctionnel répondant aux besoins de multiples utilisateurs du progiciel. Par conséquent, le travail réalisé dans le cadre de cette étude va aboutir à une réelle application de gestion. Celle-ci représente un intérêt fort pour l'entité GFI Progiciels qui a donc décidé de transférer ce projet dans un centre de services situé à Casablanca pour la finalisation des développements.

6.2.1 Avancement du projet

La mise en œuvre du prototype fonctionnel a nécessité le développement de l'ensemble de l'IHM et des différentes classes permettant de respecter le fonctionnement du framework spécifique Astre Iodas. Le cœur de l'application est ainsi déjà mis en œuvre et nécessite simplement quelques ajustements.

La modélisation de l'ensemble des éléments de paramétrage est réalisée et a permis de générer l'ensemble des classes représentant la couche de persistance des données. En outre, les éléments indispensables pour l'intégration dans le progiciel sont disponibles et fonctionnels :

- modularisation de cette application permettant de restreindre l'accès de cette application à certains interlocuteurs,
- création des outils permettant de rendre disponible ces fonctionnalités qu'à une certaine catégorie d'utilisateurs,
- mise en œuvre d'un modèle de données spécifiques :
    - table de stockage des domaines de paramétrage (type d'action, type de procédure, etc.),
    - table de stockage des dépendances entre les domaines de paramétrage,
    - table de stockage des comptes rendus de la recopie de paramétrage,
    - table temporaire de stockage des informations préalables à la recopie,
    - vue permettant de rassembler l'ensemble des éléments de paramétrage,
    - séquences Oracle permettant l'incrémentation dans les tables précitées.

Les travaux restant à réaliser concernent essentiellement le codage des différentes règles de gestion associées à la manipulation des éléments de paramétrage. Le prototype fonctionnel couvre en effet un périmètre restreint sur ce chantier (type d'action). La seconde fonctionnalité souhaitée dans l'application (génération de scripts) fait aussi partie des éléments restant à développer. Il n'a cependant pas été abordé dans le prototype fonctionnel réalisé.

6.2.1.1 Spécifications

L'ensemble des spécifications relatives à la manipulation de ces éléments de paramétrage a été effectué. Ce travail couvre les besoins de manipulation des différents concepts de paramétrage présents dans le progiciel pour la fonctionnalité de recopie sur une base de données cible. Ce travail a représenté une charge de 30 jours-hommes dans le projet. Pour information, l'annexe 1 de ce document correspond à la synthèse de ces éléments tandis que l'annexe 2 correspond à la spécification complète de la recopie d'un type d'action.

Ce travail de spécification sera aussi nécessaire pour la seconde fonctionnalité souhaitée dans l'application. La charge est estimée à 15 jours-hommes et doit cependant être prise en compte pour la suite du projet.

6.2.1.2 Modélisation

Comme évoqué précédemment, l'utilisation de la démarche MDA nécessite la modélisation complète de la couche de persistance de données. Le générateur de code commandé au CSN de Lille exploite en effet les modèles UML déployés. Le travail associé représente un volume de 16 jours-hommes et est commun aux deux fonctionnalités principales souhaitées au sein de l'application.

6.2.1.3 Gestion des dépendances

Dans l'application de recopie, la gestion des dépendances entre les différents concepts de paramétrage est stockée au sein d'une table spécifique (RECDEP). Ce fonctionnement permet d'absorber la complexité fonctionnelle liée aux différentes imbrications d'éléments de paramétrage. Un algorithme permettant de lire cette table spécifique permet ainsi de garantir le bon séquencement des opérations de recopie. Ces travaux, dont la charge est estimée à 7 jours-hommes, restent à finaliser et couvrent l'ensemble des fonctionnalités souhaitées dans l'application.

6.2.2 Priorités de développement

Le prototype fonctionnel développé a permis d'avancer sur la logique de recopie d'un paramétrage d'une base de données source à une base de données cible. La priorité de développement se situe par conséquent sur la finalisation de cette fonctionnalité particulière. A l'intérieur de ce chantier, les priorités de développement sont détaillées dans les paragraphes suivants.

Dans un second temps, la deuxième fonctionnalité souhaitée pour le produit (la génération de scripts) devra être mise en œuvre. Le développement correspondant consistera en l'enrichissement de l'algorithme de manipulation des objets métiers afin qu'il puisse créer des scripts au format PL/SQL en fonction des données à recopier.

6.2.2.1 Détachement du progiciel

Le prototype fonctionnel a été implémenté en partant d'une version existante du progiciel IODAS. Le premier chantier confié au centre de services de Casablanca, consiste à rendre cette application de recopie de paramétrage totalement indépendante du progiciel. Ce nettoyage nous permettra à terme :

- d'éviter d'éventuels conflits d'évolution du code entre les deux applications,
- de pouvoir installer l'application de recopie sans la présence de la version Web du progiciel. Certains utilisateurs restent en effet pour le moment sur une version client/serveur du produit.

6.2.2.2 Modification du générateur de code IODAS

Le générateur de code IODAS (templates au format javajet) permet de générer les classes relatives à la couche de persistance des données mais nécessite encore quelques ajustements. Afin de faciliter la maintenance de l'application et de rendre les modèles plus pertinents, deux stéréotypes spécifiques au progiciel IODAS ont été créés au sein du profil UML spécifique « PIM » :

- iodas_existence, champs de la table pertinents pour trouver l'enregistrement correspondant à la source :
    - champ : nom du champ à tester
    - table : table du champ à tester
- iodas_comparaison, champs de la table pertinents pour vérifier la conformité avec l'enregistrement trouvé sur la source :
    - champ : nom du champ à tester
    - table : table du champ à tester

Ces éléments doivent être pris en compte afin de générer différentes méthodes essentielles pour la manipulation de ces éléments de paramétrage. La logique associée consiste à garder la logique de manipulation de ces objets au sein des classes de la couche de persistance des données. La maintenance de cette application au fil des versions du progiciel sera de cette manière facilitée.

6.2.2.3 Codage des algorithmes spécifiés

Le séquencement de la manipulation des objets métiers nécessite le codage d'un algorithme récursif permettant de gérer l'ensemble des cas de figure présents. Cette logique permettra d'envisager plus sereinement les éventuelles évolutions dans les interactions entre les éléments de paramétrage du progiciel car il suffira alors de faire évoluer les éléments dans cette table plutôt que dans le code applicatif.

En gardant toujours à l'esprit cette simplification de la maintenance de l'application, il est préférable de prévoir une manipulation des objets métiers par une méthode totalement générique. Celle-ci permettrait de ne pas rentrer dans le détail de la gestion de chaque objet métier considéré comme trop chronophage et difficilement maintenable.

6.2.2.4 Gestion de la double connexion

Cette application nécessite la gestion de deux connexions à des bases de données Oracle. La connexion à la source est intrinsèque au fonctionnement du progiciel tandis que la connexion à la cible reste à améliorer. En effet, elle a bien été réalisée au sein du prototype fonctionnel mais le fonctionnement utilisé ne peut cependant pas être conservé pour des raisons de compatibilité et de conformité au framework.

6.2.2.5 Amélioration de l'IHM

La présentation du prototype fonctionnel à différents interlocuteurs a mis en exergue la nécessité d'améliorer quelques éléments de l'IHM. Ces travaux consistant essentiellement en la gestion de fenêtre secondaire et l'ajout de boutons sur certaines pages de l'application sont ainsi confiés au Centre de Services de Casablanca.

6.2.3 Prévisions de charge

Les prévisions de charge sont exprimées en jours-hommes et différencient les deux fonctionnalités distinctes :

- la recopie sur une base de données cible initiée au sein du prototype fonctionnel,
- la génération de scripts non abordée dans le prototype fonctionnel.

6.2.3.1 Fonctionnalité de recopie sur une base de données cible

En ce qui concerne la fonctionnalité de recopie sur une base de données, certains éléments restent des éléments à finaliser au niveau de la couverture fonctionnelle et de l'industrialisation de l'application.

6.2.3.1.1 Charge de développement

L'estimation de la charge de développement a été réalisée en étroite collaboration avec des experts de l'entité Progiciels (Tableau 10).

	Spécifications technique	**Codage**	**Tests**	**Total**
Indépendance de l'application	1	3	1	**5**
Modification du générateur de code	2	3	1	**6**
Codage de l'algorithme récursif	3	6	1,5	**10.5**
Codage de l'algorithme de manipulation des objets métier	3	9	1,5	**13,5**
Gestion de la double connexion	1	3	1	**5**
Améliorations de l'IHM	0,2	0,4	0,2	**1**
Total	**10,2**	**24,4**	**6,2**	**41**

Tableau 10 : Récapitulatif des charges de développement

Les éléments transmis par l'expert ont été majorés de 40% pour obtenir ces éléments. Cette majoration reste mesurée mais doit permettre de remplir les objectifs annoncés en parant à tous les imprévus. Il ne s'agit en aucun cas de dégrader la performance globale des équipes de développeurs car d'après la loi de Parkinson : « le travail se dilate jusqu'à remplir le temps disponible ».

6.2.3.1.2 Transfert de compétences

En ce qui concerne le transfert de compétences, nous proposons d'organiser des sessions de formation au sein du centre de services de Casablanca :

- 0,5 jour de présentation du projet,
- 1 jour de présentation de l'application,
- 1,5 jour de formation au paramétrage du progiciel.

L'objectif d'un transfert de compétences physique est de transmettre le projet dans les meilleures conditions et aussi de diffuser de l'information concernant la démarche MDA au sein des différentes entités du groupe.

6.2.3.1.3 Tests de qualification

Les tests de qualification permettront de s'assurer de la conformité de l'application avant la mise à disposition du produit. Une charge estimée à 10 jours sera nécessaire pour ces opérations sollicitant différents types de consultants au regard des lotissements fonctionnels mis en œuvre :

- 4 jours pour un consultant fonctionnel pour la recopie d'un type de procédure,
- 2 jours pour un consultant fonctionnel pour la recopie d'un profil,
- 2 jours pour un consultant support client pour la recopie des modules et éditions bureautiques,
- 2 jours pour un consultant fonctionnel pour la recopie du paramétrage des extranets et des domaines fonctionnels spécifiques,

6.2.3.1.4 Bilan

La charge totale associée à la mise à disposition de l'ensemble de la première fonctionnalité se chiffre à 41 jours de développement, 4 jours de transferts de compétence et 10 jours de tests de qualification. Les ressources dans ce projet se résument à :

- deux développeurs au centre de services de Casablanca,
- un consultant au sein de l'entité Progiciels (codé JTA).

Numéro de tâche	Identifiant tâche	Tâche	Charge estimé JH	Date de début	Date de fin	Ressources
1	**FOR**	Formation	3	07/11/12	09/11/12	JTA
2	**IND**	Indépendance de l'application	5	12/11/12	16/11/12	Développeur 1
3	**GEN**	Modification du générateur de code	6	12/11/12	19/11/12	Développeur 2
4	**REC**	Codage de l'algorithme récursif	10,5	19/11/12	03/11/12	Développeur 1
5	**OBJ**	Manipulation des objets métier	13,5	20/11/12	07/12/12	Développeur 2
6	**DOU**	Gestion de la double connexion	5	03/12/12	10/11/12	Développeur 1
7	**IHM**	Améliorations de l'IHM	1	07/12/12	10/12/12	Développeur 2
8	**TES**	Test de qualifications	10	11/12/12	07/01/13	JTA

Tableau 11 : Liste des tâches restantes dans le projet

D'après l'estimation de charges et les ressources disponibles sur ce projet, la première fonctionnalité souhaitée dans l'application devrait être disponible à partir de la semaine 2 de l'année 2013.

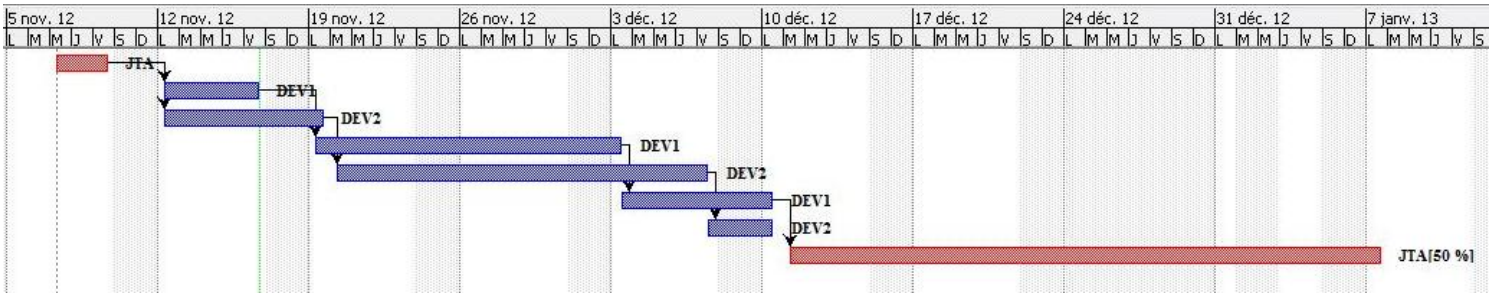

Figure 71 : Diagramme de Gantt du projet

6.2.3.2 Génération de scripts

L'absence de recul par rapport à cette fonctionnalité rend l'estimation de charges assez difficile. L'algorithme de recopie du paramétrage nécessite en effet d'être intégré dans les scripts PL/SQL générés. Ainsi, l'écriture de scripts assez complexes en plus de la programmation Java pourrait s'avérer gourmande en temps de développement. Cette partie du projet, non prioritaire dans un premier temps, nécessitera vraisemblablement au moins autant de développement que la première fonctionnalité. Les développeurs devront avoir des connaissances en SQL et PL/SQL en plus de Java.

Cette partie du projet nécessitera en outre l'écriture des spécifications fonctionnelles. Ce travail, estimé à 15 jours pour l'ensemble des lots, devra être intégré dans les plannings préalablement au déclenchement de ce chantier.

6.2.4 Maintenance du produit

La maintenance de cette application nécessitera la prise en compte des nouveaux éléments de paramétrage apportés par les versions de Iodas. Les opérations suivantes seront donc nécessaires pour maintenir l'applicatif

- maintenance des modèles UML représentant le modèle de données IODAS et nouvelle génération de la couche de persistance.
- maintenance des dépendances fonctionnelles entre les différents concepts de paramétrage.

La maintenance de l'application apparaît comme fortement facilitée par l'utilisation de la démarche MDA dans le projet. L'évolution des modèles accompagnée d'une nouvelle génération du code de l'application suffira en effet pour la prise en compte des évolutions du progiciel.

6.3 Les perspectives de l'application de cette démarche

La mise en œuvre de la démarche MDA dans cette application laisse déjà entrevoir de belles promesses pour l'avenir. En effet, en utilisant une modélisation ne couvrant que la couche de persistance des données, on devrait arriver à l'obtention d'une application maintenable à l'aide d'opérations simples. Il est ainsi facile d'imaginer les possibilités associées à une modélisation plus complète de l'application et les gains réalisés en termes de productivité, de qualité et de maintenabilité.

6.3.1 Enrichissement du générateur de code IODAS

En l'état actuel, le générateur de code IODAS et par extension, la démarche MDA, est limité au niveau de la couche de persistance des données. Il n'apporte que peu d'améliorations par rapport à un outil déjà existant qui génère les classes Java à partir du modèle de données. Le principe de modélisation permet tout de même de sensiblement améliorer la maintenabilité de ces éléments. De plus, la prise en compte des fonctionnalités du round trip permet de conserver l'ensemble des développements manuels lors d'une nouvelle génération de code.

L'intégration de l'ensemble des spécificités associées à l'architecture du framework Astre permettrait d'aller jusqu'à la génération de l'IHM et de la couche controlleur. L'ensemble du cadre de l'application serait ainsi développé et il resterait alors simplement à coder les fonctionnalités non modélisées. Celle-ci est actuellement fournie par un concepteur de l'entité Progiciels et nécessite un temps non négligeable de travail évitable par l'introduction de la démarche MDA.

6.3.2 Gestion plus efficace de l'Offshore

Lé démarche MDA apporte un gain conséquent dans la gestion des développements Offshore. En donnant un rôle plus important aux spécifications et à la génération automatique, on réduit les temps de développement globaux en Offshore. La transmission normalisée de spécifications permet en outre de réduire les temps d'analyse technique nécessaires pour la bonne compréhension des opérations à mener.

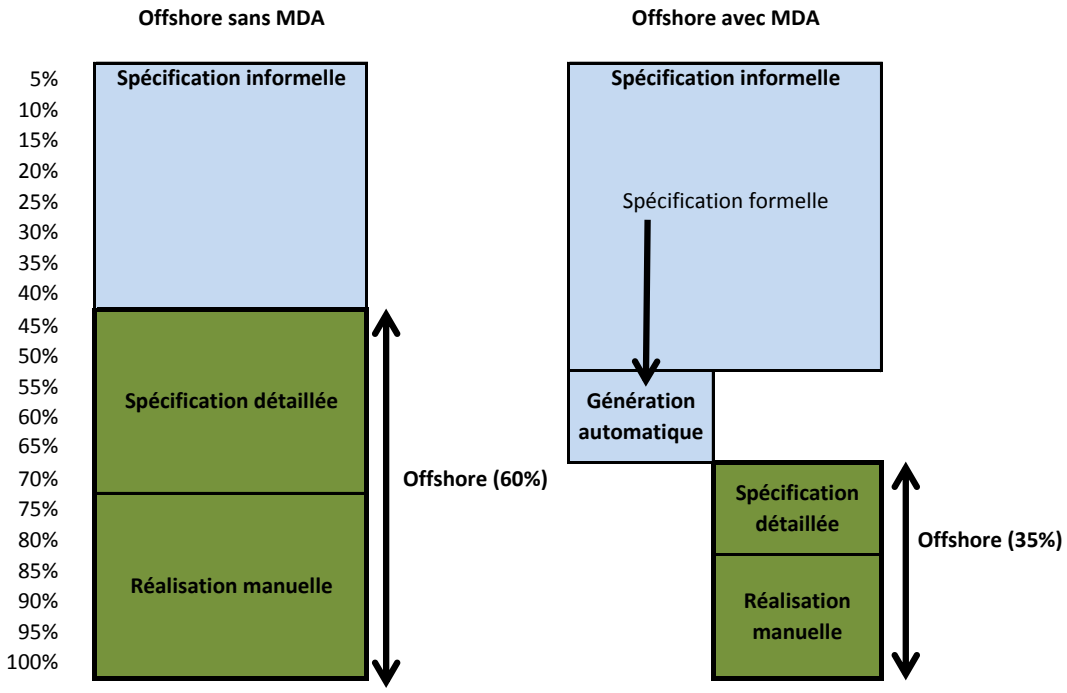

Figure 72 : Impact de la démarche MDA dans les projets Offshore

Ce nouveau mode de fonctionnement implique la modification des méthodes de travail tant au niveau de la conception que du développement.

6.3.3 Modification des méthodes de travail

La mise en œuvre de la démarche MDA dans les futures applications de GFI Progiciels devra nécessairement s'accompagner d'une révolution culturelle des méthodes de travail des concepteurs produit. Une réelle conduite du changement sera ainsi essentielle dans la réussite du déploiement de cette nouvelle méthodologie :

- obtenir l'adhésion reposant sur la mobilisation de toutes les énergies autour d'un même objectif : transformer les processus de développement,
- être "psy" : les démarches psychosociologiques privilégient le brainstorming et l'action collective et misent donc sur l'influence mutuelle entre les personnes,
- communiquer de manière collective et individuelle sur le changement,
- coordonner les équipes : toute conduite du changement se décompose en une multitude d'initiatives à mener au sein des différents services de l'entreprise,
- savoir gérer dans le temps : il est vital de formaliser et de planifier des tâches afin de suivre leur exécution et de veiller au respect des coûts et des délais impartis.

La révolution culturelle sera de toute façon compliquée à mettre en œuvre car elle nécessite de modifier les habitudes de travail de nombreuses personnes. Celle-ci nécessite de surcroît de prévoir la formation des collaborateurs à la modélisation UML et de s'assurer de sa bonne compréhension au niveau de l'Offshore. Le ressenti actuel au niveau de la conception logiciel est que la modélisation UML des applications est une perte de temps dans un projet de développement d'applications.

6.4 Bilan personnel

Ce stage de fin de cursus d'ingénieur m'a fourni l'opportunité de travailler dans un contexte différent de ce que j'ai pu connaître au fil de mes expériences professionnelles. Il est toujours intéressant de participer à l'activité aux seins d'autres organisations, de partager d'autres visions du monde professionnel et d'autres contraintes. Le travail aux contacts de chercheurs m'a permis aussi de mieux appréhender le principe de fonctionnement de la recherche dans le domaine informatique.

Les différents travaux menés m'ont permis de toucher à de nombreux métiers de l'informatique et m'ont permis de mettre en pratique les différents enseignements reçus au sein du CNAM. La programmation réalisée dans le cadre du prototype fonctionnel m'a par exemple permis de réaliser un premier réel travail dans ce domaine et ainsi de mieux appréhender les différentes contraintes industrielles liées.

C'est aussi le premier projet au cours duquel j'ai été tour à tour le chef de projet, l'installateur logiciel, le concepteur, l'architecte, le développeur, le testeur et l'utilisateur. Intervenir dans toutes les phases du cycle de vie du projet laisse une liberté d'action et de décision tout à fait appréciable mais nécessite aussi de nombreuses compétences difficiles à acquérir. La découverte de ces différentes facettes sera un bonus très intéressant pour la suite de ma carrière professionnelle et m'a conforté dans l'utilité d'un travail en équipe. Une personne seule ne peut pas disposer de l'ensemble des compétences requises pour la finalisation d'un projet tel que celui présenté dans ce document.

Table des figures

Table des illustrations

Bibliographie

17 experts du développement logiciel. (2001). *Manifeste Agile*.

Initiative MDA de l'OMG. (2001). Récupéré sur OMG: http://www.omg.org/mda/

Eclipse Modeling Framework Project (EMF). (2012). Récupéré sur Eclipse: www.eclipse.org/modeling/emf/

Unified Modeling Language. (2012). Récupéré sur Wikipedia: http://fr.wikipedia.org/wiki/Unified_Modeling_Language

Bézivin, J. (2004). Sur les principes de base de l'ingénierie des modèles. *L'Objet*, 145-156.

Bézivin, J., & Gerbé, O. (2001). Towards a Precise Definition of the OMG/MDA Framework. *Automated Software Engineering - ASE'01*. San Diego.

Bézivin, J., & Jouault, F. (2009). Acquis et Défis de l'Ingénierie Dirigée par les Modèles. *Génie Logiciel n°90*, 55-60.

Blanc, X. (2005). *MDA en action*. Eyrolles.

Bondé, L. (2006). *Transformations de Modèles et Interopérabilité dans la Conception de Systèmes Hétérogènes sur Puce à Base d'IP*. Lille.

Bouara, M., Barbouche, A., & Kouider El Ouahed, A. (2008). La démarche MDA (Model Driven Architecture). *INFØDays*. Chlef, Algérie.

Canals, A., Millan, T., & Rault, J.-C. (2011). L'ingénierie Dirigée par les Modèles : Bilan et perspectives. *Génie Logiciel n°97*, 1-3.

Cariou, E., & Barbier, F. (2008). Ingénierie des modèles : panorama et tendances. *Génie Logiciel n°85*, 2-6.

Desfray, P. (2009). MDE, DSL et UML : Où en est-on ? *Génie Logiciel n°90*, 2-5.

Eveillard, S., Youbi, K., & Henry, A. (2009). Rétro-ingénierie dirigée par les modèles d'applications COBOL. *Génie Logiciel n°90*, 31-39.

Favre, J.-M., & Musset, J. (2005). Rétro-ingénierie dirigée par les métamodèles. *Journées sur l'Ingénierie Dirigée par les Modèles*. Lille.

Favre, J.-M., Estublier, J., & Blay-Fornarino, M. (2006). *L'ingénierie dirigée par les modèles : au-delà du MDA*. Lavoisier.

IBM. (2004). *An introduction to Model Driven Architecture*.

Jézéquel, J.-M., Combenale, B., & Vojtisek, D. (2012). *Ingénierie dirigée par les modèles - des concepts à la pratique*. Ellipses 2012.

Kadima, H. (2005). *Conception orientée objet guidée par les modèles*. Dunod.

Kherraf, S. (2011). *Méthodologie de transformation du CIM en PIM dans l'approche MDA*. Québec.

Kordon, F. (2010). Ingénierie basée sur les modèles : quelques défis actuels. *Génie Logiciel n°93*, 2-4.

Roques, P. (2005). *UML 2 par la pratique*. Eyrolles.

Roques, P., & Vallée, F. (2004). *UML 2 en action : de l'analyse des besoins à la conception J2EE*. Eyrolles.

Rousse, N. (2007, 11). *Ingénierie Dirigée par les Modèles*. Récupéré sur Modelia: http://www.modelia.org/html/9_fichesTechniques/8000_IDMcahier.pdf

Seignard, X. (2012). *Model driven tools*. Récupéré sur About model driven engineering: http://mdwhatever.free.fr/index.php/2010/06/model-driven-tools-the-big-list/

Sottet, J.-S., Calvary, G., & Jean-Marie, F. (2005). Ingénierie de l'Interaction Homme-Machine Dirigée par les Modèles. *Premières journées sur l'Ingénierie Dirigée par les Modèles*. Paris.

Staron, M. (s.d.). *Model Driven Architecture in Industry – A Bigger Picture*. Goteborg.

TECHNOLOGIES CLÉS 2010. (s.d.). Consulté le 10 09, 2012, sur Ministère de l'Économie, des Finances et de l'Industrie: http://www.industrie.gouv.fr/techno_cles_2010/html/tech_07.html

Tolbert, D. (2000). CWM : A Model-Based Architecture for DataWarehouse Interchange. *Workshop on Evaluating Software Architectural Solution*. Irvine.

Topcased, G. d. (2007). Un panorama des techniques de transformation de modèles. *Workshop Topcased*. Toulouse.

Vojtisek, D. (2009). Introduction à la construction d'un DSL sous Eclipse. *Programmez n°120*, 70-72.

Annexe 1 : synthèse des spécifications concernant les interactions dans le paramétrage

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Catégories de prestataire	Paramètres associés	Produit	
Catégories de tiers	Paramètres associés		
Types d'acte	Paramètres associés Motif d'interruption/annulation Catégorie de l'agenda		
Types de dossier			
Types de formation	Paramètres associés		
Types de scolarisation	Paramètres associés		
Types d'activité professionnelle	Paramètres associés		
Types de non activité professionnelle	Paramètres associés		
Editions bureautiques	Type de pièce bureautique Données bureautique	Armoire GED Sous-dossier GED Sous-sous-dossier GED	
Données de l'engagement			
Enveloppes IODAS	Enveloppe territoriale		Type d'unité territoriale Exercice budgétaire Acteur
Groupes de rubriques			Rubrique
Groupes d'opérations de liquidation hors contexte			Type de traitement Rubrique
Groupes d'opérations de valorisation/évaluation/contrôle			Type de traitement
Groupes de type d'action			Type d'action
Groupes de type de relevé d'information			Type de relevé d'information
Groupes de type d'étape de procédure			Type d'étape de procédure
Groupes de type de procédure			Type de procédure
Indicateurs de pilotage IODAS			
Lots de produit			Produit

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Modules de traitement			
Paramètres	Attribut de tiers Nature de paramètre Valeur possible		
Paramètres généraux	Valeur du paramètre		
Paramètres généraux DUW			Produit Enveloppe GFD Opinion
Paramètres généraux RSA			Type de dossier Type d'unité territoriale Produit Motif d'arrêt de procédure Motif d'interruption d'intervention Type d'action Type de procédure
Paramètres généraux de l'extraction CNSA			Produit Type de procédure Type d'étape de procédure Type d'instance Opinion Type de réponse Type de relevé d'information Paramètre Valeur possible

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Périmètres d'apurement			Produit Type de procédure Type de dossier Type d'instance Type d'action collective Type de traitement Type de relevé de mouvements
Produits	Nature de produit Propriété extranet Produit lié (service et sous-produits) Type de mouvement Motif d'interruption Modalité de sortie/fin d'attente Opinion Type de réponse Paramètre Duplication de paramètre	Produit de consolidation Type de procédure Type d'évènement fondateur Type d'étape de procédure Rubrique Type d'instance Type de mouvement	
Profils		Typologie sensible Outils Type de dossier Type de procédure Type d'étape de procédure Type de traitement Type d'action collective Editions bureautiques Type de relevé de mouvements Catégorie de prestataire Type de relevé d'information Catégorie de tiers Armoire GED Sous-dossier GED Type de service Enveloppe GFD	

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
		Domaine de contact Groupe de type de procédure Groupe de type d'étape de procédure Groupe de type de relevé d'information Groupe de type d'action Type de parcours Groupe de moyens Type de situation financière Groupe d'opérations de liquidation hors contexte Groupe d'opérations d'évaluation/contrôle hors contexte Groupe de rubriques Type de relevé d'information sur le logement	
Profils OSW	Item OSW de niveau 2		
Rôles			
Rubriques	Paramètres associés Mode de paiement	Enveloppe IODAS Rubrique de compensation	Type d'unité territoriale Module de traitement Enveloppe GFD Sous-rubrique
Rubriques d'évaluation PH	Thème d'évaluation PH Paramètres associés Valeur possible		
Champs du module d'évaluation PH		Type de procédure Type d'action	
Types de parcours			Produit

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Types de procédure PDA			Type de procédure Type d'étape Type d'acte Type d'action Type d'évènement Type de relevé d'information
Types d'évènement calendrier			Produit de service Type d'évènement calendrier compatible Type de mouvement
Types d'intervention PDA			Produit Paramètre Valeur possible Opinion Paramètre général
Types d'action	Paramètres associés Motif d'interruption/annulation Qualité dans l'action		Type d'acte Produit
Types d'action collective	Type d'acte collectif Paramètres associés		Type d'unité territoriale Type de relevé d'information collectif
Types de découpage UT			Type d'unité territoriale
Types de dossier papier	Paramètres associés		Type d'unité territoriale Armoire GED
Types de mouvement	Paramètres associés		
Types de périmètre de contrôle d'effectivité			Type d'unité territoriale Rubrique

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Types de procédure	Groupe de type de procédure (Client/serveur) Groupe de type d'étape de procédure (Client/serveur) Type d'étape Evènement fondateur Qualité dans la procédure Type d'étape de procédure Type d'évènement sortant Produit Type d'action Type de relevé d'information sur le logement Type de relevé d'information Type de situation financière Type d'instance Type de réponse Type de pièce bureautique Type d'acte Opinion	Editions bureautiques	Type d'instance Type de tiers Type de dossier papier Sous-dossier GED Sous-sous-dossier GED
Types de relevé de mouvements	Paramètres associés		Produit de consolidation Type de mouvement Produit d'intervention Paramètre du type de mouvement
Types de relevé d'information	Paramètres associés		
Type de relevé d'information collectif	Paramètres associés		
Types de retour GFD	Spécificités du logement actuel Spécificités du logement futur		
Types de relevé d'information sur le logement	Types de ressources/charges/dettes/crédits/biens Types de valeur		

Bloc fonctionnel principal	Bloc fonctionnel secondaire	Avertissement	Blocage
Types de situation financière			Rubrique Type de relevé de mouvements Catégorie de prestataire Produit
Types de traitement de calcul	Paramètres associés		
Types d'instance		Editions bureautiques	Type de réponse Opinion
Types d'unité territoriale			
Type de cadre d'accueil	Mode de contact Domaine de contact		
Type d'objet de contact	Paramètres associés		

Annexe 2 : Spécifications fonctionnelles de la recopie d'un type d'action

A dérouler pour chaque type d'action sélectionné par l'utilisateur.

- Recherche sur la BDD cible d'un type d'action dont le code (NOMENC.NOME) et la typologie (NOMENC.TYPO) correspondent aux éléments de la BDD source
    - Si aucun élément trouvé :
        - Intégration des données sur la BDD cible (tables NOMENC et TYPACT)
        - Ecriture d'une consignation
    - Sinon :
        - Utilisation de l'élément trouvé sur la BDD cible
            - Si des éléments des tables NOMENC et TYPACT diffèrent entre les deux bases (comparaison des champs détaillés ci-après) :
                - Mise à jour des données de la BDD cible avec les données de la BDD source
                - Ecriture d'une consignation

- Pour chaque paramètre associé sur la BDD source (enregistrement correspondant à TPATAC.ID_TYPEACTN=NOMENC.ID)
    - Recherche sur la BDD cible d'un paramètre associé dont le code (TYPPAR.TYPEPARA) correspond à celui de la BDD source
        - Si aucun élément trouvé :
            - Déclenchement du bloc fonctionnel concernant les paramètres pour TPATAC.ID_TYPEPARA=TYPPAR.ID.
                - Si le champ « ID_PARAGRAP » de la table TPATAC de la BDD source est non nul
                    - Déclenchement du bloc fonctionnel concernant les typologies standards pour TPATAC.ID_PARAGRAP=NOMENC.ID.
            - Intégration des données sur la BDD cible (table TPATAC)
            - Ecriture d'une consignation
        - Sinon :
            - Utilisation de l'élément trouvé sur la BDD cible
                - Si des éléments diffèrent entre les deux bases (comparaison des champs détaillés ci-après) :
                    - Si le champ « ID_PARAGRAP » de la table TPATAC de la BDD source est non nul
                        - Déclenchement du bloc fonctionnel concernant les typologies standards pour TPATAC.ID_PARAGRAP=NOMENC.ID.
                    - Mise à jour des données sur la BDD cible (table TPATAC) avec les données de la BDD source
                    - Ecriture d'une consignation

- Pour chaque paramètre associé sur la BDD cible (enregistrement correspondant à TPATAC.ID_TYPEACTN=NOMENC.ID) non présent sur la BDD source
    - Suppression de l'association avec le paramètre
    - Ecriture d'une consignation

- Pour chaque motif d'annulation/interruption associé au type d'action sur la BDD source (enregistrement correspondant à TYANMT.ID_TYPEACTN=NOMENC.ID)
    - Recherche sur la BDD cible d'un motif d'annulation/interruption associé dont le code (NOMENC.NOME) et la typologie (NOMENC.TYPO) correspondent aux éléments de la BDD source :
        - Si aucun élément trouvé :
            - Déclenchement du bloc fonctionnel concernant les typologies standards pour TYANMT.ID_MOTI=NOMENC.ID.
            - Intégration des données sur la BDD cible (table TYANMT)
            - Ecriture d'une consignation
        - Sinon :
            - Utilisation de l'élément trouvé sur la BDD cible
                - Si des éléments diffèrent entre les deux bases (comparaison des champs détaillés ci-après) :
                    - Mise à jour des données sur la BDD cible (table TYANMT) avec les données de la BDD source
                    - Ecriture d'une consignation

- Pour chaque motif d'annulation/interruption associé sur la BDD cible (enregistrement correspondant à TYANMT.ID_TYPEACTN=NOMENC.ID) non présent sur la BDD source
    - Suppression de l'association avec le motif d'annulation/interruption
    - Ecriture d'une consignation

- Pour chaque type d'acte associé sur la BDD source (enregistrement correspondant à TACTAC.ID_TYPEACTN=NOMENC.ID)
    - Recherche sur la BDD cible d'un type d'acte associé dont le code (NOMENC.NOME) et la typologie (NOMENC.TYPO) correspondent aux éléments de la BDD source :
        - Si aucun élément trouvé :
            - Arrêt des opérations
            - Ecriture d'un blocage
        - Sinon :
            - Utilisation de l'élément trouvé sur la BDD cible
                - Si des éléments diffèrent entre les deux bases (comparaison des champs détaillés ci-après) :
                    - Mise à jour des données sur la BDD cible (table TACTAC) avec les données de la BDD source
                    - Ecriture d'une consignation

- Pour chaque type d'acte associé sur la BDD cible (enregistrement correspondant à TACTAC.ID_TYPEACTN=NOMENC.ID) non présent sur la BDD source
    - Suppression de l'association avec le type d'acte
    - Ecriture d'une consignation

- Pour chaque qualité dans l'action associé sur la BDD source (enregistrement correspondant à TACQUA.ID_TYPEACTN=NOMENC.ID)
    - Recherche sur la BDD cible d'une qualité dans l'action associée dont le code (NOMENC.NOME) et la typologie (NOMENC.TYPO) correspondent aux éléments de la BDD source :
        - Si aucun élément trouvé :
            - Déclenchement du bloc fonctionnel concernant les typologies standards pour TACQUA.ID_QUALACTN=NOMENC.ID.
            - Intégration des données sur la BDD cible (table TACQUA)
            - Ecriture d'une consignation
        - Sinon :
            - Utilisation de l'élément trouvé sur la BDD cible
                - Si des éléments diffèrent entre les deux bases (comparaison des champs détaillés ci-après) :
                    - Mise à jour des données sur la BDD cible (table TACQUA) avec les données de la BDD source
                    - Ecriture d'une consignation

- Pour chaque qualité dans l'action associée sur la BDD cible (enregistrement correspondant à TACQUA.ID_TYPEACTN=NOMENC.ID) non présent sur la BDD source
    - Suppression de l'association avec la qualité dans l'action
    - Ecriture d'une consignation

- Pour chaque produit associé sur la BDD source (enregistrement correspondant à TACPRO.ID_TYPEACTN=NOMENC.ID)
    - Recherche sur la BDD cible d'un produit associé dont le code (PRODUI.PROD) correspond à celui de la BDD source :
        - Si aucun élément trouvé :
            - Arrêt des opérations
            - Ecriture d'un blocage
        - Sinon :
            - Utilisation de l'élément trouvé sur la BDD cible
                - Si des éléments diffèrent entre les deux bases (comparaison des champs détaillés ci-après) :
                    - Mise à jour des données sur la BDD cible (table TACPRO) avec les données de la BDD source
                    - Ecriture d'une consignation

- Pour chaque produit associé sur la BDD cible (enregistrement correspondant à TACPRO.ID_TYPEACTN=NOMENC.ID) non présent sur la BDD source
    - Suppression de l'association avec le produit
    - Ecriture d'une consignation