Rapport de Synthèse de l'AS CNRS sur le MDA (Model Driven Architecture)

Par Jean Bézivin, Mireille Blay, Mokrane Bouzhegoub, Jacky Estublier, Jean-Marie Favre, Sébastien Gérard, Jean Marc Jézéquel.

1. Introduction

En novembre 2000, l'Object Management Group (OMG) a proposé une approche nommée Model Driven Architecture (MDA™) pour le développement et la maintenance des systèmes à prépondérance logicielle.

En 2003, le CNRS crée l'AS « MDA ». Après deux ans de veille technologique et de travaux de recherche, les membres de l'AS défendent alors l'idée que ce vaste mouvement mondial initié par l'industrie est une révolution culturelle dans le contexte du développement de logiciels. De plus, la portée de ce mouvement ne doit pas se limiter à des avancées technologiques guidées par l'OMG, mais au contraire s'élargir pour tenter d'établir une synergie entre les travaux de recherche présents et passés manipulant eux aussi des modèles.

Les chercheurs français de différents domaines afférant à l'informatique peuvent et doivent largement contribuer à ce vaste mouvement. Il s'agit de mettre à profit les compétences et résultats obtenus par exemple dans les domaines de la compilation des langages, des méthodes formelles, de la modélisation par objets, de la programmation par composants distribuée, des technologies du web, etc. Pour cette raison, nous ne parlerons plus de MDA dans ce document mais d'*Ingénierie Dirigée par les Modèles* (IDM, ou MDE pour "Model Driven Engineering" dans la langue de Shakespeare).

Nous affirmons ainsi que cette évolution rapide et importante des pratiques industrielles de production de logiciels pour être durable doit s'appuyer sur la convergence des travaux issus de différents domaines de l'ingénierie logicielle. Dans ce cadre, la recherche doit jouer un rôle très important en établissant des *ponts interdisciplinaires* pour profiter au mieux des avancées technologiques de chacun en proposant un *cadre intégrateur solide*.

Pour faire face à la complexité et à l'évolution croissante des applications, l'IDM ouvre de nouvelles voies d'investigation. En autorisant une appréhension des applications selon différents points de vues tout en intégrant comme fondamental la composition et mise en cohérence de ces perspectives, elle ne peut s'inscrire dans la pérennité que si elle prend ces racines dans des bases bien fondées établies par la théorie.

En conséquence dans ce rapport, nous avons choisi à la fois de présenter l'IDM de manière

synthétique au travers des premières pages, puis d'étayer nos propos par des chapitres annexes qui font état d'investigations menées pendant ces deux ans. Certains des chapitres annexes sont co-signés par des chercheurs extérieurs à l'AS qui par leur contribution renforcent l'idée que cette révolution ne restera pas une utopie. L'IDM devrait faciliter le transfert entre recherche et développement, mais également entre équipes de recherche.

Le reste de cette partie se décompose comme suit. Après un tracé rapide des principes fondateurs d'une démarche bien fondée suivant l'IDM, nous proposons de dégager les abstractions fondamentales à l'IDM, puis d'identifier les problèmes ouverts et verrous technologiques. Cette partie se termine alors par une série de perspectives envisagées par le groupe pour faire de l'Ingénierie Dirigée par les Modèles un succès.

2. Evolutions récentes en Ingénierie Dirigée par les Modèles

2.1 Le MDA et l'IDM

L'idée initiale de l'OMG consistait à s'appuyer sur le standard UML pour décrire séparément les parties des systèmes indépendantes des plates-formes spécifique (PIM ou Platform Independant Models) et les parties liées aux plates-formes (PSM ou Platform Specific Models). Dans les années qui ont suivi, ce projet est devenu plus ambitieux et a évolué de façon interne et de façon externe.

De façon interne d'abord, l'OMG met surtout en avant actuellement une architecture dont le socle est le MOF (Meta-Object Facility) [5], sur lequel s'appuie d'une part une collection de métamodèles dont UML n'est qu'un élément parmi d'autres et d'autre part une technologie émergente de transformation de modèles nommée MOF/QVT qui est en cours de standardisation [8].

De façon externe ensuite, l'approche MDA devient une variante particulière de l'Ingénierie Dirigée par les Modèles. L'IDM peut être vue comme une famille d'approches qui se développent à la fois dans les laboratoires de recherche et chez les industriels impliqués dans les grands projets de développement logiciels. Deux de ces industriels (IBM et Microsoft) ont récemment défini leur stratégie MDE et le moins que l'on puisse dire c'est qu'elles semblent converger.

2.2 UML et l'IDM

Il faut donc séparer clairement les approches IDM du formalisme UML. Non seulement la portée de l'IDM est beaucoup plus large que celle d'UML, mais la vision IDM est aussi très différente de celle d'UML et parfois même en contradiction. UML est, dans ses versions 1.5 et 2.0, un standard assez monolithique obtenu par consensus à maxima, dont on doit réduire la portée à l'aide de mécanismes comme les profils. Ces mécanismes n'ont pas toute la précision souhaitable et mènent parfois à des contorsions dangereuses pour se rapprocher d'UML.

Dans certains outils UML de première génération, le support des profils UML a été présenté comme une fonctionnalité importante. De plus certains de ces outils ont proposé des langages de décoration propriétaires. Le résultat de ces choix est bien souvent de créer des modèles "partimoniaux" (legacy model) qui ont parfois coûté cher à produire et qui vont être difficiles à réintégrer dans des approches IDM. Ces modèles patrimoniaux restent liés à l'outil qui les a

produits, ce qui est en contradiction avec les objectifs de l'approche de l'OMG prônant l'indépendance de la plate-forme et donc à fortiori des outils de développement.

Dans l'article [1] Microsoft ne cache pas que son utilisation d'UML restera probablement essentiellement « contemplative » ou encore documentaire, c'est à dire utilisable essentiellement pour produire et communiquer des esquisses. La déclinaison plus solide de l'IDM chez Microsoft, celle qui est progressivement intégrée dans l'outillage Visual Studio, s'appelle "Software Factories". Elle est fondée essentiellement sur des langages de domaines (DSL ou Domain Specific Languages) de petite taille, facilement manipulables, transformables, combinables, etc. En un mot ces DSL sont la base de l'automatisation de l'IDM chez Microsoft.

IBM ne dit pas autre chose dans son manifeste [7]. Les trois axes de l'Ingénierie Dirigée par les Modèles sont d'après IBM (1) les standards ouverts, (2) l'automatisation et (3) la représentation directe. Parmi les standards ouverts UML peut bien sûr avoir sa place, mais ce sera aux cotés de XML et d'autres standards. Par contre, ce qui est essentiel, c'est la possibilité de traitement automatique de modèles (par exemple tissage, vérification, transformation, etc.) s'appuyant sur des standards précis, limités en taille et spécialisés. On retrouve encore ici cette idée des DSL. Les DSL sont ici de petits langages spécifiques de domaines adaptés à des corporations particulières ou à des besoins particuliers. C'est ce que le manifeste IBM appelle la "représentation directe", c'est-à-dire la mise à disposition de métiers particuliers ou de tâches spécifiques de langages précis et outillés (éditeurs, générateurs, vérificateurs, etc.).

La vision de Microsoft est progressivement mise en œuvre dans Visual Studio. La vision d'IBM est mise en œuvre par exemple dans l'outillage EMF (Eclipse Modeling Framework). Ces visions correspondent à l'architecture multi-niveau de l'OMG fondée sur la pyramide de métamodélisation dominée par le MOF ainsi que sur la possibilité de définir une grande variété de métamodèles spécialisés pour les DSL.

2.3 Les objets et l'IDM

Il faut clairement séparer l'approche orienté objet de l'approche orientée modèle et de l'IDM. S'il est vrai que le standard MDA de l'OMG est directement fondé sur une technologie orientée objet, ce n'est là qu'un choix technologique. Dans bien des cas cependant la confusion est faite entre les concepts que l'on trouve dans l'IDM et leur incarnation dans le monde objet. Par exemple la vision selon laquelle un modèle serait une "instance d'un" métamodèle a été largement relayée par l'OMG, le promoteur d'UML et de l'OMA (Object Management Architecture). La relation *InstanceDe* est trop souvent utilisée de manière informelle. Elle fait référence à l'hypothèse implicite que modèles et métamodèles sont exprimés dans une technologie orientée-objet. Or il n'existe aucune contrainte de cette sorte dans l'IDM.

L'objectif de l'IDM est de définir une approche pouvant intégrer différents *espaces technologiques* [2]. Presque 20 ans après les débuts de l'orienté objet dans l'industrie, le "tout-objet" n'est plus de mise. Par exemple l'espace technologique des grammaires, des documents structurés XML, ou des bases de données relationnelles, ne sont pas fondés sur le concept d'objet. Les deux relations de base *InstanceDe* et *HériteDe* ont fait couler beaucoup d'encre et elles sont assurément au cœur de l'approche orienté objet. Elles ne sont cependant pas adaptées aux autres espaces technologiques [4]. La question se pose donc de définir quels sont les concepts et relations essentielles à l'IDM. Ce thème est abordé dans la section 3.

2.4 Les méthodes de modélisation et l'IDM

La terminologie « Ingénierie Dirigée par les Modèles » laisse supposer que la nouveauté de cette approche réside dans l'utilisation systématique de modèles. Il n'en est pourtant rien et cette caractéristique est loin d'être originale. Il existe depuis longtemps en informatique des méthodes de modélisation, la plus connue en France étant Merise, SSADM au Royaume Uni et Information Engineering aux Etats-Unis. Dans ces méthodes, les modèles sont vus comme étant la base de toute activité humaine d'ingénierie. Ces méthodes ont largement contribué à la diffusion du concept de modèle en informatique, mais leur impact sur la production effective du logiciel reste néanmoins mitigé. Plus de 50 ans après les débuts de l'informatique, les pratiques industrielles restent centrées sur le code, considéré comme étant le seul représentant fiable du logiciel.

Les méthodes de modélisation sont parfois décriées pour leur lourdeur et leur manque de souplesse face à l'évolution rapide du logiciel. Elles ont en fait mené à la notion de "**modèle contemplatif**", dans laquelle, un modèle peut certes être utile pour la communication et la compréhension, mais reste éloigné de la notion de production ; le codage reste, pour beaucoup d'informaticiens, le cœur de leur métier. Les modèles contemplatifs doivent être interprétés par l'homme alors que la préoccupation première de l'informaticien est de produire des artéfacts interprétables par la machine.

Les méthodes de modélisation se sont concentrées surtout sur les phases amont d'analyse et de conception, la liaison avec le code source et les phases aval comme le déploiement des logiciels et leur maintenance n'ont que peu ou pas été prises en compte. L'espace non couvert par ces méthodes de modélisation a été peu a peu rempli par des techniques et des outils plus ou moins ad-hoc comme la génération de code via des assistants (Wizards) ou l'utilisation de fichiers XML pour exprimer des modèles de déploiement. Les assistants et l'utilisation de fichiers XML sont deux exemples de montée en abstraction par rapport au code tout en restant productif. Ces techniques sont complémentaires des modèles contemplatifs proposés par les méthodes de modélisation ; l'objectif de l'IDM est d’englober ces différentes approches pour couvrir tout le cycle de vie du logiciel.

Pour qu'un modèle soit "**productif**" il doit pouvoir être interprétable et manipulable par une machine. Il est considéré comme essentiel dans le cadre de l'IDM de pouvoir exprimer formellement le contenu d'un modèle et "quoi faire" avec ce modèle et ceci quel quesoit le niveau d'abstraction. Ceci se traduit par le besoin d'exprimer formellement les transformations entre modèles, les rendant ainsi productifs. Pour cela il est indispensable de formaliser non seulement les modèles, mais aussi les langages dans lesquels ceux-ci sont décrits, et finalement les méta-modèles décrivant ces langages.

La caractéristique originale de l'Ingénierie Dirigée par les Modèles est plus dans l'utilisation systématique de méta-modèles, que dans l'utilisation systématique de modèles. L'IDM se distingue des méthodes de modélisations traditionnelles comme Merise par la préoccupation constante de rendre les (meta) modèles productifs plutôt que contemplatifs.

2.5 La programmation par aspects et L'IDM

Dans des approches comme la programmation par aspects on essaye de réaliser la synthèse des préoccupations logicielles sur la base du code, par exemple avec des extensions du langage Java comme AspectJ ou HyperJ. Dans l'approche MDA on essaye de réaliser la séparation des

aspects liés ou indépendants de la plate-forme par des modèles différents (PIM et PSM). Dans les approches plus générales IDM, chaque aspect ou point de vue particulier d'un projet de développement ou de maintenance est pris en compte par un modèle particulier conforme à un métamodèle précis.

La séparation et le tissage des aspects sont donc au centre des approches IDM. Pour l'OMG, ce qui est prioritaire dans le MDA, c'est la séparation des aspects liés ou indépendants de la plate-forme mais pour l'IDM, de façon plus générale, les aspects à prendre en compte couvrent toutes les préoccupations qui peuvent apparaître lors du développement, de la maintenance et de l'évolution des systèmes à prépondérance logicielle (aspects fonctionnels mais aussi aspects non-fonctionnels comme la qualité de service, la sécurité, etc.)

2.6 Vers des plates-formes IDM

De nombreux projets français, européens ou internationaux visent actuellement à construire des plates-formes collaboratives supportant des outils IDM de nouvelle génération. Bien sûr ces plates-formes pourront aussi accepter des outils anciens de première génération comme des outils de capture/édition de modèles UML, voire même des outils de support ou de récupération de profils UML pour raisons de compatibilité. Mais la caractéristique importante des nouveaux outils est d'être à métamodèle variables et non plus à métamodèle fixe (UML) comme précédemment. Ceci est réalisable d'un point de vue technologique comme l'ont montré plusieurs projets déjà anciens (MetaEdit+ de Metacase ou OpenTool de TNI par exemple). Il suffit de placer ces outils dans le contexte normatif adéquat (MOF et XML par exemple) pour disposer par exemple d'éditeurs de DSL puissants et paramétrables.

Sur une plate-forme IDM on va donc trouver ces éditeurs de DSL, mais aussi des référentiels de modèles et de métamodèles ("repositories"), des outils supportant la transformation de modèles ainsi que de nombreuses autres opérations. La plate-forme supportera un vaste écosystème de fonctionnalités allant des outils anciens encore maintenus aux outils plus récents de traitement automatique de DSL. La puissance de cette plate-forme ne proviendra pas de sa complexité interne, mais par ce qu'elle traitera, de façon homogène, des modèles conformes à des DSL concrétisés par des métamodèles précis.

3. Concepts essentiels de l'IDM

Au cours des deux dernières décennies l'approche orientée objet est peu a peu devenue l'approche de choix pour le développement de logiciels dans l'industrie. Or, l'objectif de l'IDM est de proposer une approche intégrant de manière homogène différents espaces technologiques dans laquelle l'approche orientée-objet et la technologie correspondante ne représentent qu'un cas particulier.

Se pose alors la question des concepts essentiels. L'approche orientée objet est basée sur deux relations fondamentales: la relation *InstanceDe* qui permet d'introduire la notion de classe et la relation *HériteDe* qui permet d'introduire la notion de superclasse. Il s'agit de déterminer quels sont les relations et les concepts essentiels de l'IDM. Bien qu'aucune réponse à cette question ne puisse être définitive à ce stade des recherches, il apparaît de plus en plus consensuel que deux relations sont fondamentales. La première relation, appelée *ReprésentationDe* est liée à la notion de *modèle*, alors que la relation *EstConformeA* permet de définir la notion de modèle par rapport à celle de *métamodèle*. Il est essentiel de comprendre

que ces relations peuvent avoir des incarnations différentes dans tel où tel espace technologique, même si des concepts similaires peuvent s'y retrouver. C'est là le point fort de l'IDM, que de faire ressortir des concepts essentiels et fédérateurs de technologies développées de manière disjointe.

3.1 Modèles et *ReprésentationDe* (μ)

Il n'existe pas à ce jour de définition universelle de ce qu'est un modèle (tout comme il n'existe pas de définition universelle de ce qu'est un objet, un aspect, etc.). Une étude plus approfondie de la littérature montre cependant un relatif consensus sur une certaine compréhension. Les trois définitions ci-dessous sont représentatives de cette tendance.

La définition ci-dessous provient du standard UML qui, bien que critiquable sur bien des points, est néanmoins un standard de facto:

> *"A **model** is an abstraction of a physical **system**, with a certain purpose."*

La mention système "physique" est clairement de trop, car cela signifierait qu'il n'est pas possible de définir des concepts abstraits comme la notion de cours, de session ou de transaction dans de tels modèles.

Seidwitz rapporte dans[10], des travaux récents concernant l'étude des concepts fondamentaux de l'IDM. La définition suivante est proposée:

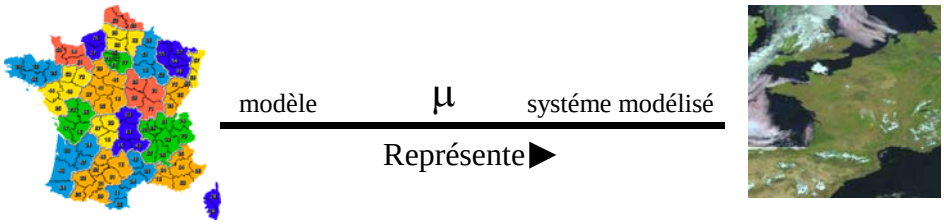
> *"A **model** is a set of statements about some **system under study** (SUS)."*

Finalement, une définition plus complète est proposée par Bézivin et Gérbé dans [13]:

> *"A **model** is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual **system**."*

Comme le suggèrent ces définitions provenant de sources différentes, il existe un consensus sur le fait que ***modèle*** et ***système étudié*** sont deux rôles complémentaires basés sur une unique relation liant un modèle au système qu'il modélise. Cette relation est appelée ***ReprésentationDe*** dans[4][10] et [11]. Notée μ pour éviter toute connotation et confusion, cette relation est écrite plus en détail dans l'annexe « Concepts et relations de base pour l'Ingénierie Dirigée par les Modèles ».

Notons que rien n'est dit dans les définitions ci-dessus sur la nature exacte des modèles et des systèmes considérés. Ces définitions ne sont donc pas restreintes aux modèles et systèmes informatiques, bien que ce soit le cadre de cette étude. Par exemple une carte géographique peut jouer le rôle de *modèle*, alors que la région étudiée jouera celui de *système modélisé*.



Notons que la relation μ ne peut généralement pas être formalisée, car cela impliquerait déjà d'avoir formalisé les deux extrémités de la relation. Ce n'est généralement pas le cas lorsque les systèmes modélisés font partie "du monde réél". Par exemple la région considérée dans

l'exemple précédent n'est pas en elle-même formalisée. Par contre le modèle que l'on fait d'un tel système peut lui être décrit dans un langage précis. C'est par exemple le cas d'une carte numérique dont on connaît parfaitement la structure.

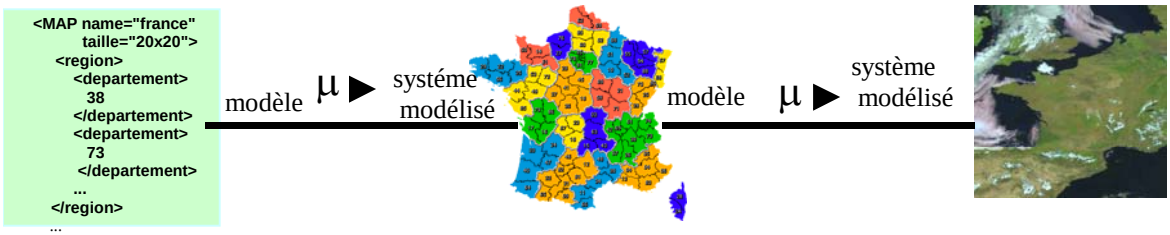
Un modèle informel peut trouver son utilité dans le cadre d'activités humaines. Par exemple la cartographie a montré l'utilité de tels modèles aux cours des siècles. L'utilisation de modèles informels en informatique est aussi une pratique courante. Les limites des modèles contemplatifs sont néanmoins connues.

Dans le cadre de l'Ingénierie Dirigée par les Modèles on s'intéresse aux modèles formalisés pour les rendre productifs en automatisant leur traitement et leur exploitation informatique. Pour mettre en avant ce caractère fondamental de l'approche, certains auteurs intègrent d'ailleurs cette restriction directement dans la définition de la notion de modèle. Ainsi dans le contexte de l'IDM, Warmer et ses collègues donnent la définition suivante:

> *"A **model** is a description of (part of) a **system** written in a **well-defined language**" [15]*

Comme le montre cette définition la notion de modèle, telle qu'elle est considérée dans ce rapport, fait explicitement référence à la notion de langage bien-défini et donc, comme nous allons le voir dans la section suivante, à la notion de "méta-modèle" qui permet justement de définir un langage.

Il est important de souligner tout de suite que ce n'est pas en modélisant un modèle que l'on rend ce modèle plus formel. Par exemple ce n'est pas en modélisant une région que la région s'en voit mieux connue. Par exemple une description (formelle ou informelle) d'une carte géographique permet d'obtenir un modèle de cette carte (qui joue ainsi le rôle de système modélisé); la carte jouant à son tour le rôle de modèle par rapport à la région.



La relation μ a été ici combinée deux fois, la carte est un modèle de la France, et le fichier XML à gauche est un modèle de la carte, pourtant on n'a pas fait intervenir la notion de langage ou de métamodèle. Le modèle en XML **n'est pas** un métamodèle de la France. C'est un modèle d'un autre modèle. Contrairement à une idée parfois véhiculée, *un métamodèle n'est pas un modèle d'un modèle*.

3.2 Métamodèles et *EstConformeA* (χ)

Pour qu'un modèle soit productif, et c'est là l'objectif premier de l'IDM, il faut qu'il puisse être manipulé par une machine et donc que le langage dans lequel ce modèle est exprimé soit clairement défini. La notion de métamodèle est au cœur même de l'IDM. Bien que certains aspects relatifs à cette notion restent sujets à controverses, on peut noter néanmoins une convergence dans les définitions que l'on trouve dans la littérature:

*"A **meta-model** is a model that defines the language for expressing a **model**"*[5].

*"A **metamodel** is a specification model for a class of SUS where each SUS in the class is itself a valid **model** expressed in a certain modelling language"*[15].

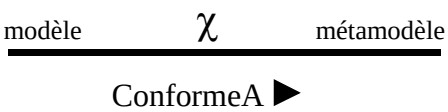
Il ressort d'une étude plus approfondie des définitions que l'on trouve dans la littérature qu'un meta-modèle est un modèle d'un langage de modélisation. La notion de métamodèle mène à l'identification d'une seconde relation nommée ***ConformeA*** dans [4] et [12]. Cette relation, notée χ, pour éviter tout problème terminologique et toute connotation erronée, est décrite plus en détail dans l'annexe « Concepts et relations de base pour l'Ingénierie Dirigée par les Modèles ».

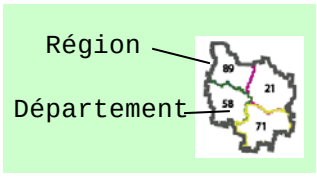
L'originalité de l'IDM *n'est pas* de mettre l'accent sur la relation μ liant un modèle au système modélisé. La paternité de la relation μ, ou tout au moins la reconnaissance de son importance en informatique, doit assurément être attribuée aux méthodes de modélisation. Par contre le point clé dans l'IDM est de mettre l'accent sur l'importance de la relation χ liant un modèle au métamodèle auquel il est conforme. Ceci permet d'assurer d'un point de vue théorique mais surtout opérationnel qu'un modèle est correctement construit et donc qu'il est envisageable de lui appliquer des transformations automatisées.

La force de l'IDM est de reconnaître par ailleurs que la relation de conformité a plusieurs incarnations selon les Espaces Technologiques considérés. Par exemple dans l'espace technologique XML, on dira qu'un document XML est conforme à un schéma ou à une DTD; dans l'espace technologique des grammaires on dira qu'une phrase ou un programme est conforme à une grammaire donnée; dans l'espace technologique des bases de données on dira que le contenu d'une base de données est conforme au schéma de cette base de données, dans l'espace technologique orienté-objet on dira qu'un objet est conforme à sa classe. La relation *InstanceDe* correspond donc à l'incarnation de la notion de conformité dans le cadre orientée objet mais ce n'est là qu'un cas particulier.

En réalité, dans un même espace technologique, on peut considérer plusieurs variantes de cette relation de conformité, par exemple l'annexe « Espace Technologique Language et IDM » décrit différents types de relations de conformité comme la conformité syntaxique, la conformité sémantique, etc.

La longue histoire de la cartographie, représentative s'il en est de l'évolution de la notion de modèle au cours des ages, montre qu'il est désormais jugé indispensable d'associer à chaque carte la description du "langage" utilisé pour réaliser cette carte. Ceci se fait notamment sous la forme d'une légende explicite. La carte doit être conforme (χ) à cette légende, sinon elle n'est pas utilisable. Bien évidemment plusieurs cartes peuvent être conformes à une même légende.





Désormais la cartographie n'est plus liée uniquement à la géographique. Il existe une foule de cartes spécialisées décrivant différents domaines: cartes géopolitiques, économiques,

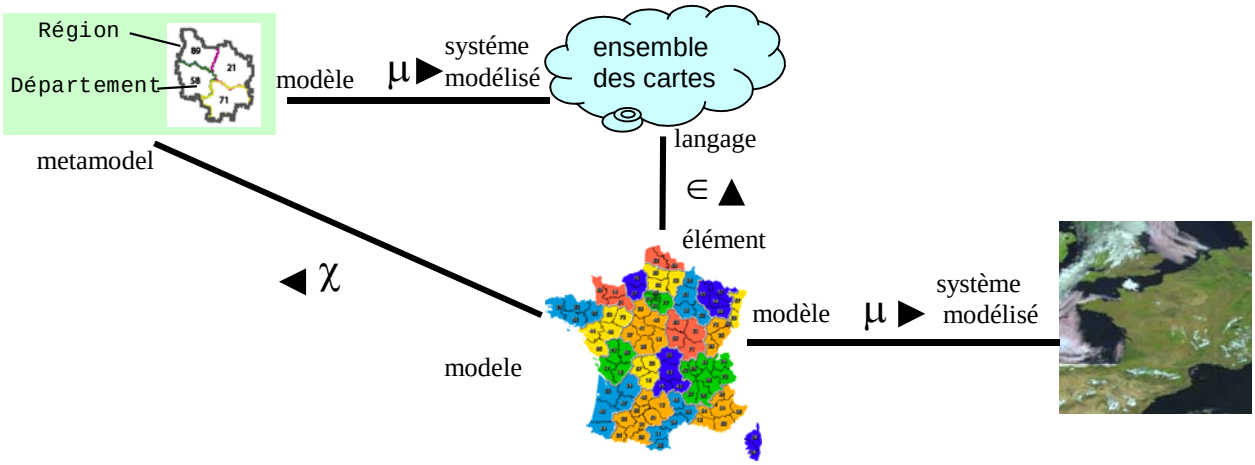
minières, archéologiques, vinicoles, etc. A chaque fois il s'agit de définir un langage spécialisé de domaine et de définir une légende appropriée. L'importance de définir et de développer ces formalismes est reconnu comme étant une activité à part entière, tout comme le développement et l'évolution de modèles représentant des régions particulières.

Il en est de même pour l'Ingénierie Dirigée par les Modèles. Les métamodèles deviennent des entités de première classe. Il doit entre autre être possible de définir des langages spécifiques de domaine (DSL) pour adresser les spécificités liées à tels secteurs d'activités, à tel métier, à tel point de vue, à tel niveau d'abstraction, ou à telles plates-formes technologiques.

3.3 Métamodèles vs. Langages

Bien souvent la notion de langage et de métamodèle sont confondus. Bien que ces deux concepts soit proches, ils sont néanmoins différents. Revenons sur la définition donnée auparavant: *"A **meta-model** is a model that defines the **language** for expressing a **model**"* [5]. Un langage est un système abstrait alors qu'un métamodèle est une définition explicite de ce langage. Il faut donc distinguer le langage, qui joue le rôle de système, du (ou des) métamodèle(s) qui jouent le rôle de modèle(s) de ce langage.

Si l'on reprend l'exemple des cartes, on peut considérer par exemple le langage "des cartes IGN". Ce langage *est* l'ensemble de toutes les cartes décrites selon des conventions définies par l'Institut Géographique National. Comme le suggère la figure suivante, une légende *n'est pas* un langage (un ensemble) mais un outil concret pour définir et appréhender un langage de cartes. Une légende est une *ReprésentationDe* (μ) de ce langage. Elle joue le rôle de *modèle* par rapport au langage et de *métamodèle* par rapport à une carte particulière qui doit être *ConformeA* cette légende. La carte, quant à elle, est un élément du langage.



La vision ci-dessus n'est pas neuve. Ce n'est qu'une explicitation et reformulation de résultats connus dans la théorie des langages. En effet un langage est un ensemble (au sens mathématique du terme) de phrases, alors qu'une grammaire est un modèle de ce langage (de cet ensemble). La notion de grammaire correspond, dans l'Espace Technologique Langage, à la notion de métamodèle en utilisant la terminologie unificatrice IDM. D'un point de vue pratique c'est la relation de conformité qui est utilisée et qui est implantée par exemple par un analyseur. Dans la langue parlée, remarquons qu'on assimile souvent langage et grammaire. Un métamodèle est un moyen concret de *définir* un langage, ce n'est pas un langage.

3.4 Synthèse

L'approche orientée objet est fondé sur deux relations essentielles, "*InstanceDe*" et "*HériteDe*" qui donne lieu à la notion d'objet, de classe et de super-classe. La nature exacte de ces relations et de ces notions est encore source de débat plus de 30 ans après leur introduction, mais il existe néanmoins des solutions « suffisamment bonnes » pour que ces concepts soient jugés pertinents et utilisables.

L'ingénierie Dirigée par les Modèles est basée sur un autre jeu de concepts et de relations, qui n'ont pas de rapport direct avec l'approche orientée objet. Un consensus est en train de se former autour de deux relations: *ReprésentationDe* (μ) menant à la notion de *modèle*, et *ConformeA* (χ) menant à la notion de *métamodèle*. De plus, le fait qu'un métamodèle soit un modèle d'un langage de modélisation, est également un fait admis.

Les différentes relations se retrouvent sous différentes incarnations selon l'espace technologique considéré. Il ne s'agit pas ici de définir de nouveaux concepts mais au contraire de donner une vision unificatrice à des technologies et des concepts trop souvent étudiés de manière séparée. La relation *InstanceDe* est une incarnation particulière de la relation *ConformeA* dans la technologie orientée objet, mais une fois de plus il est important de distinguer l'IDM du MDA et de l'approche objet.

La notion de transformation est un autre concept central pour l'IDM. Décrire comment transformer un modèle est une condition indispensable pour le rendre productif. L'annexe « Concepts et relations de base pour l'Ingénierie Dirigée par les Modèles » décrit la notion de transformation ainsi que les concepts associés, en faisant l'hypothèse de l'existence d'une troisième relation fondamentale appellée *TransforméEn* (τ) [12]. Il n'existe cependant pas de définition ou de terminologie consensuelle et les termes modèles de transformations, transformations, langages de transformation, correspondances, etc. sont souvent utilisés de manière peu cohérente. Les transformations font néanmoins l'objet de recherches actives.

4. Verrous scientifiques et technologiques

Cette section résume les principaux verrous scientifiques et technologiques de l'IDM. Nous distinguons principalement quatre catégories de verrous à lever dans les prochaines années.

4.1 Verrous liés à la définition et la validation des modèles

Les modèles servent à abstraire un système étudié en faisant émerger des points de vue particuliers et en en donnant une compréhension synthétique facilitant la communication et le raisonnement. L'une des premières difficultés pour atteindre cet objectif est la construction des modèles : Quelle est la finalité des modèles visés ? Comment appréhender le système observé ? Comment abstraire cette réalité et faire ressortir les artefacts essentiels ? Quelle granularité de modélisation en fonction de quel contexte d'utilisation? La modélisation du système étudié doit faire appel à plusieurs sciences complémentaires pour proposer des méthodologies selon la complexité du système et la finalité de cette étude. L'intégration de techniques provenant de l'ingénierie des besoins, de l'acquisition des connaissances et des sciences cognitives, entre autre, n'est pas une tâche aisée ; elle nécessite une approche pluridisciplinaire qui n'est pas sans risque.

L'autre problème important concerne la validation des modèles. Un modèle est un point de

départ à une prise de décision ou à la production d'un logiciel ; il doit donc constituer une base saine tant dans sa représentation syntaxique que dans sa sémantique. Identifier les critères de qualité d'un modèle, qualifier un modèle selon son niveau d'abstraction et sa finalité, vérifier la cohérence interne d'un modèle, valider la pertinence d'un modèle vis-à-vis à la fois de la réalité modélisée et de l'objectif de modélisation sont autant de tâches difficiles mais indispensables. Elles dépendent souvent du langage (formalisme) de modélisation utilisé (voir plus loin) mais aussi de l'expertise de modélisation et de la complexité des modèles produits.

4.2 Verrous liés à la définition des méta modèles

Un modèle peut prendre différentes formes comme une expression mathématique, un diagramme, une spécification dans un langage formel… Les modèles réalisés ne sont pas neutres par rapport au méta modèle utilisé pour les décrire. En effet, les méta modèles et les méthodes les manipulant ont une grande influence sur la perception que le modeleur a du système qu'il doit décrire. Ainsi, les objets de ce système sont appréhendés à travers les concepts proposés par le meta modèle, quitte, quelquefois, à tordre cette réalité pour l'exprimer dans le langage. Le choix du méta modèle est donc très important et doit être en bonne adéquation avec la finalité de la modélisation et la complexité du système à modéliser. D'autres aspects de la modélisation sont liés au choix du langage : la validation de la conformité des modèles (cohérence, complétude…), le raisonnement sur les modèles (inférence de types, inférence de connaissances…) et «l'exécutabilité» des modèles (i.e. l'étude de leurs comportements opérationnels). Selon les bases théoriques du langage adopté, ces tâches seront plus ou moins faciles à réaliser, voire impossibles si le langage manque de rigueur dans sa définition ou n'est pas en adéquation avec la finalité de la modélisation. L'utilisation de diagrammes est très courante en génie logiciel et en conception de systèmes d'information. Les diagrammes, très pratiques comme support de communication, souffrent souvent de bases théoriques insuffisantes. Par ailleurs, la forme graphique ne permet souvent de visualiser qu'une partie de la sémantique d'un modèle, il faut donc compléter les diagrammes par des annotations textuelles (formelles ou semi-formelles) rendant les tâches de validation, de raisonnement et d'exécution plus difficiles à réaliser.

Le challenge des prochaines années dans ce domaine est la définition de meta modèles riches et rigoureux, séparant la sémantique du langage de sa syntaxe et offrant des mécanismes de validation et de raisonnement en adéquation avec la nature et la complexité des systèmes à modéliser. Une grande variété de langages, vus comme des DSL, est ainsi nécessaire pour couvrir à la fois plusieurs domaines d'application et plusieurs niveaux d'abstraction. Cette hétérogénéité des langages n'est pas sans risque mais elle est nécessaire car imposée par la diversités des points de vue et des systèmes à modéliser, ainsi que par la diversité des acteurs de la modélisation (domaines et niveaux d'expertise différents…). Ce problème fait ainsi l'objet du troisième verrou identifiés par l'AS.

4.3 Verrous liés à l'hétérogénéité des modèles et méta modèles

Dans le monde informatique, on constate aujourd'hui une prolifération de métamodèles pour décrire des modèles soit pour décrire différentes facettes d'un même système, soit pour dénoter différents niveaux d'abstraction (modèles conceptuels, modèles logiques, modèles physiques…).

Cette prolifération s'accroît d'autant plus que les systèmes sont complexes et représentés par

une agrégation de composants faisant eux-mêmes l'objet de modélisation. La diversité technologique impose aussi sa propre complexité en modélisation (foisonnement de plateformes support, de services, de standards de représentation…). Cette hétérogénéité des métamodèles et des modèles ne peut être maîtrisée que si l'on dispose de techniques rigoureuses de

- comparaison de métamodèles (et de modèles) pour prouver l'équivalence des métamodèles (ou de modèles),
- transformation de métamodèles (et de modèles) pour passer ou d'un métamodèle (d'un modèle) à l'autre ou raffiner un même modèle,
- intégration de modèles pour créer de nouveaux modèles par composition ou par tissage.

Le challenge des prochaines années se situe dans la formalisation des interactions entre modèles via la formalisation de leur métamodèles, dans l'interopérabilité des modèles et en général dans la gestion des métamodèles et des modèles. Les approches de méta modélisation, vues comme une science d'analyse et de description de langages, constituent une piste ouverte. Cependant, les résultats dépendront de la capacité des chercheurs à définir des langages à sémantique claire et à prouver le bénéfice des méthodes sous-jacentes en termes de productivité et de flexibilité (voir ci-dessous).

4.4 Verrous liés aux coûts de production et d'exploitation des modèles et méta modèles

S'il faut développer aujourd'hui peu d'effort pour convaincre de l'utilité des modèles, il n'en est pas de même quant au prix à payer pour leur définition et leur exploitation. En effet, au delà des bases formelles saines et des méthodes de conception rigoureuses nécessaires, l'autre défi de l'Ingénierie Dirigée par les Modèles est de prouver sa compétitivité par rapport à d'autres méthodes d'ingénierie. En quoi l'IDM facilite-t-elle le développement logiciel ? Jusqu'à quel degré l'IDM simplifie-t-elle la conception et l'analyse des systèmes complexes ? Quel est le gain de productivité que l'IDM induit au niveau du développement et de la maintenance de logiciels ? Quel est l'impact de l'IDM sur la gestion du changement et de l'évolution ? Quel est l'impact de l'IDM sur les technologies du marché ? Comment se répartissent les coûts de l'IDM entre fournisseurs de plateformes et utilisateurs de ses plateformes ? Comment intégrer l'IDM dans un contexte où le poids des applications patrimoniales (legacy application) est si important? Comment faciliter la rétro-ingénierie de modèles à partir de logiciels existants et ainsi prendre en compte les capitaux investis? Comment assurer une migration des technologies actuelles vers le cadre unificateur de l'Ingénierie Dirigée par les Modèles? Comment développer le marché de la vente des outils logiciels pour faire du logiciel?

Négliger les réponses à ces questions et ne pas considérer l'existant, c'est donner peu de chances et de perspectives à l'IDM. L'automatisation des procédés de construction, de validation, de gestion et d'évolution de modèles est un préalable pour constituer une base aux mesures précédentes. L'IDM doit être validée par confrontation par rapport aux autres approches d'ingénierie mais aussi de façon réflexive en exploitant ses propres artefacts.

En conclusion, il faut ajouter que ces défis ne peuvent être abordés que si préalablement, il y a un assainissement de la terminologie dans le domaine et une compréhension commune des notions de modèle, méta modèle, transformations, correspondances, etc. Il faut aussi une conviction partagée que l'IDM ne peut réussir qu'au prix d'une certaine rigueur dans la définition des langages et des méthodes sous-jacentes.

5. Perspectives

Pour répondre aux problématiques de complexité, d'évolutivité et d'hétérogénéité des systèmes, la recherche de plates-formes intergicielles (middleware) pour l'intégration de systèmes distribués a constitué un objectif important. Il semble nécessaire aujourd'hui de recourir à des méthodes assez nouvelles, fondées sur des techniques de nature générative organisées autour de nombreux DSL s'appuyant sur des métamodèles précis.

5.1 Du MDA à l'IDM

La proposition MDA de l'OMG fut, en 2000, la première proposition marquante allant dans ce sens. Elle faisait entrevoir des perspectives alléchantes, mais proposait des concepts ambigus et des solutions floues. Au cours des années suivantes un certain nombre d'outils se réclamant du «MDA» ont vu le jour, et un certain nombre de projets, tant industriels que de recherche, ont été lancés. Cet ensemble d'expériences a permis de montrer que l'idée selon laquelle il suffirait d'utiliser UML pour pouvoir exprimer des modèles métiers neutres (PIM, Platform Independent Model) et ensuite les lier à différentes plates-formes d'exécution (PSM, Platform Specific Model) est simpliste, voire naïve. Mais ces expériences ont également permis, au moins dans des contextes particuliers, de valider le principe même d'une approche dirigée par les modèles, c'est-à-dire plaçant les modèles au centre de l'activité de développement, les autres artefacts (documentation, code, tests, etc.) devenant périphériques et contingents. Pendant toute cette période, une évolution rapide et profonde de l'approche s'est opérée :

- Généralisation à tous types de modèles (par opposition à des modèles MOF/UML uniquement i.e. liés aux standards produits par l'OMG) sous l'appellation Ingénierie Dirigée par les Modèles (IDM, ou MDE "Model Driven Engineering ", MDD " Model Driven Development", ou encore MDSE "Model Driven Software Engineering").
- Prise de conscience de la part des praticiens du Génie Logiciel, et des autres Génies, que l'usage des modèles est aussi ancien que l'ingénierie elle-même.
- Prise de conscience de la part de la communauté académique qu'une partie importante du travail a toujours été d'établir des modèles, des méta-modèles et d'en dériver des outils. D'ailleurs, une importante base de connaissances existe à ce sujet.
- Prise de conscience de la part des grands producteurs de logiciels, comme Microsoft et IBM, des potentialités de l'approche. Ces derniers affichent que leur stratégie est aujourd'hui basée sur l'IDM.

La validité de l'approche est actuellement admise par les industriels, et on constate ainsi une accélération impressionnante des ralliements et des efforts dans ce domaine. A tort ou à raison, la prochaine décennie au moins sera marquée par l'utilisation de l'IDM dans l'industrie. Cette utilisation ne sera certainement pas limitée à UML. En effet, la complexité du standard UML qui ne cesse de croître, son manque de définition rigoureuse pour plusieurs concepts et les fortes limitations de son formalisme d'extension par profils ont conduit plusieurs équipes de recherche et de nombreux industriels à s'appuyer sur des DSL bien délimités pour prendre en compte les différents aspects du développement logiciel. L'approche de l'IDM et du MDA sont bien différentes de celle d'UML.

De leurs côtés, les milieux académiques constatent un manque criant de bases solides à cette

« nouvelle » approche, une incohérence et une confusion inquiétante dans la définition des concepts à la base de l'IDM. Dans le même temps ils constatent que la démarche scientifique, pas seulement présente dans le domaine du génie logiciel, consiste souvent à établir des modèles, leurs méta-modèles, puis à les formaliser de façon à générer automatiquement des outils tels que des analyseurs et générateur de code… Parmi les plus matures, des espaces technologiques, tels que les langages et les bases de données, donnent de bons exemples de cette démarche.

De ce fait, de nombreux domaines scientifiques ont accumulé depuis des décennies des masses de connaissances, de résultats, d'outils, d'expériences qui pourraient/devraient contribuer à l'essor de cette approche intégratrice qu'est l'IDM. En effet, on se rend compte graduellement que l'IDM pourrait devenir **le** domaine dans lequel les résultats de recherche relatifs à la modélisation seraient collectés, mis en forme, homogénéisés, standardisés avant d'être mis à la disposition des diverses communautés scientifiques et industrielles. L'IDM remplirait alors une fonction d'intégration tant des pratiques industrielles que des concepts, théories, techniques et outils provenant de domaines scientifiques de recherche jusque là cloisonnés. Nul doute que de cette confrontation surgiraient nombre de généralisations fécondes.

5.2 Les évolutions prévisibles

Les travaux liés à l'IDM lancés par les industriels sont comme d'habitude essentiellement tournés vers le court terme. La communauté scientifique se doit alors d'accompagner cet effort et de fournir les vues à moyen et à long terme, nécessaires pour assurer le succès et la pérennité de cette nouvelle approche du développement du logiciel.

Sur le moyen terme, il est raisonnable d'attendre une clarification et une simplification de la définition des concepts de base. Il faut toutefois garder à l'esprit que cette tâche pourrait bien ne jamais obtenir un consensus universel, compte tenu des nombreuses implications et des habitudes prises dans certains domaines. A l'image de la communauté des langages à objets, même si un consensus complet n'est pas établi, on devrait obtenir une compréhension des concepts « suffisamment bonne » pour rendre l'approche pertinente et utilisable à très large échelle.

A plus long terme de nombreuses inconnues demeurent. Ces inconnues sont d'ordre pratiques comme théoriques.

L'IDM, par nature, manipule de nombreux formalismes, appartenant à divers espaces technologiques et situés à divers niveaux d'abstraction. Comment les concepteurs s'adapteront-ils à cette logique ?, Comment la complexité qui en résulte pourra-t-elle être maîtrisée et masquée par les outils ? L'expérience dans le domaine des langages a montré qu'il est très long, mais pas impossible, de maîtriser des systèmes s'appuyant sur des traductions en cascades entre des langages différents ; mais l'IDM pousse très loin cette logique.

5.2.1 Perspectives industrielles

Du point de vue pratique, les processus de développement et de maintenance seront profondément modifiés ; les outils et les environnements devront être repensés en conséquence. Nul ne peut prédire aujourd'hui ce que seront ces environnements, ni quels problèmes se poseront lorsque de nombreux systèmes de grande taille, développés avec l'IDM, seront à

maintenir. L'expérience montre que les effets pervers de toute technique nouvelle sont extrêmement difficiles à prédire. D'autres problèmes pratiques seront à résoudre pour faire interopérer les approches classiques et l'IDM, pour réutiliser les logiciels patrimoines, pour gérer l'hétérogénéité, pour faire évoluer les applications pour gérer les lignes de produits etc. Des pans entiers de l'informatique industrielle devront être adaptés à l'IDM. Parmi les problèmes pratiques, il faut aussi mentionner la standardisation des langages de modélisation, des méta-modèles spécifiques, des standards d'échange etc. Ces standards sont nécessaires pour que l'IDM se développe ; à ce sujet les obstacles sont principalement liés à la stratégie des acteurs industriels dominants du marché.

5.2.2 Perspectives scientifiques

Du point de vue théorique, il est clair que nous allons vers des travaux de fond sur la sémantique des modèles, sur les techniques de composition, de tissage et de transformation de modèles. Un volant de travail considérable est à réaliser sur les propriétés qu'il sera possible d'attacher aux modèles, leur préservation ou modification lors des opérations de composition/transformation avec d'autres modèles ayant d'autres propriétés. Les vérifications et preuves, les divers outillages (test, migration, optimisation, etc.) restent à imaginer et à réaliser. Ce dernier point est capital au succès de cette technologie.

Ces travaux semblent généraliser les travaux déjà effectués dans plusieurs domaines de l'informatique. Cela n'a rien de surprenant si l'on considère qu'un modèle est un « programme » dans le langage de son méta modèle, que le concept de modèle a été profondément étudié dans d'autres domaines comme les bases de donnés (schéma = modèle), les systèmes d'informations, et quasiment toutes les disciplines. Les ontologies, les approches formelles, l'AOP, les langages spécifiques de domaine, la programmation générative et bien d'autres domaines vont être sollicités pour contribuer au développement de l'IDM.

5.3 Synthèse

Sur le long terme, il est difficile de savoir ce qui émergera de ces travaux. Il est toutefois raisonnable de penser que les fondements de l'approche seront bien maîtrisés. Il est présomptueux de dire que l'IDM sera la technologie dominante de demain, et cette question n'a peut-être pas de sens, mais il est raisonnable de penser que l'on va disposer avant 2007 de plates-formes industrielles opérationnelles IDM qui permettront aux idées exprimées ci-dessus de se déployer de façon pratique. Par exemple l'un de ces projets est le projet intégré européen ModelWare rassemblant une communauté européenne d'industriels et d'institutions de recherche pour la création d'une telle plate-forme ouverte. Mais il y a également de nombreux projets similaires en cours de développement en Amérique du Nord, qui bénéficient de financements importants des organismes de soutien, par exemple aux USA. Enfin, et surtout, les acteurs industriels majeurs comme Microsoft et IBM ont lancés leurs équipes de recherches et annoncent déjà des produits et un support fort aux approches de l'IDM (avec une vision plus DSL que MDA).

Il est donc acquis que les techniques issues de l'IDM feront partie intégrante de la panoplie d'outils et des approches à la disposition des développeurs de logiciels, et qu'elles vont profondément influencer les pratiques du Génie Logiciel. Il est probable également que les travaux sur l'IDM auront une influence profonde sur le contour des disciplines scientifiques telles qu'elles sont perçues aujourd'hui et, espérons le, favoriseront une unification de

domaines actuellement cloisonnés.

6. Références

[1] Greenfield, J. & Short, K. "Moving to Software factories", Software development, http://www.sdmagazine.com, Juillet 2004.

[2] I. Kurtev, J. Bezivin, and M. Aksit. "Technological spaces: an initial appraisal.", In CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002

[3] Groupe OFTA Ingénierie de Modèles logiciels et Systèmes, Arago #30

[4] Bézivin, J. In search of a Basic Principle for Model Driven Engineering, Novatica/Upgrade, Vol. V, N°2, (April 2004), pp. 21-24, http://www.upgrade-cepis.org/issues/2004/2/upgrade-vol-V-2.html

[5] OMG, "Meta Object Facility (MOF) Specification" Version 1.4, April 2002

[6] Bézivin, J., Gérard, S. Muller, P.A., Rioux, L. MDA Components: Challenges and Opportunities, Metamodelling for MDA, First International Workshop, York, UK, (November 2003), http://www.cs.york.ac.uk/metamodel4mda/onlineProceedingsFinal.pdf

[7] Booch G., Brown A., Iyengar S., Rumbaugh J., Selic B. The IBM MDA Manifesto The MDA Journal, May 2004, http://www.bptrends.com

[8] OMG The MOF/QVT Queries, Views, Transformations request for proposal

[9] Soley, R. & the OMG staff MDA, Model-Driven Architecture, November 20000, http://www.omg.org/mda/presentations.htm

[10] E. Seidwitz, "What Models Mean", IEEE Software, September 2003

[11] C. Atkinson, T. Kühne, "Model-Driven Development: A Metamodeling Foundation", IEEE Software, September 2003

[12] J.M. Favre, "Towards a Basic Theory to Model Model Driven Engineering", 3rd Workshop in Software Model Engineering, WiSME 2004, http://www-adele.imag.fr/~jmfavre

[13] J. Bézivin, O. Gerbé, "Towards a Precise Definition of the OMG/MDA Framework", ASE'01, Novembre 2001

[14] S.J. Mellor, K. Scott, A.Uhl, D.Weise, "MDA Distilled: Principles of Model-Driven Architecture", Addison Wesley, March 2004

[15] A. Kleppe, S. Warmer, W. Bast, "MDA Explained. The Model Driven Architecture: Practice and Promise", Addison-Wesley, April 2003