# Sur les principes de base de l'ingénierie des modèles

**Jean Bézivin**

*Université de Nantes*
*Equipe ATLAS, (INRIA & LINA)*
*2, rue de la Houssinière, BP 92208*
*F-44322 Nantes cedex 3*

*Jean.Bezivin@lina.univ-nantes.fr*

RÉSUMÉ. *En novembre 2000, l'OMG a rendu publique son initiative MDA™, une variante particulière d'une tendance plus générale nommée ingénierie des modèles (MDE pour Model Driven Engineering). Les idées de base de l'ingénierie des modèles sont voisines de celles de nombreuses autres approches comme la programmation générative, les langages spécifiques de domaines, le MIC (Model Integrated Computing), les usines à logiciel (software factories), etc. Le MDA 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, SPEM, QVT, etc. Tout comme le principe de base du « tout objet » fut essentiel dans les années 1980 pour établir la programmation par objets, le principe du « tout modèle » peut aussi être essentiel pour identifier les caractéristiques majeures de cette nouvelle tendance. Les domaines d'applicabilité sont multiples mais concernent essentiellement l'ingénierie des logiciels, l'ingénierie des systèmes et l'ingénierie des données.*

ABSTRACT. *In November 2000, the OMG made public the MDA™ initiative, a particular variant of a new global trend called Model Driven Engineering (MDE). The basic ideas of model engineering are germane to many other approaches such as generative programming, domain specific languages, model-integrated computing, software factories, etc. MDA may be defined as the realization of model engineering principles around a set of OMG standards like MOF, XMI, OCL, UML, CWM, SPEM, QVT, etc. Similarly to the basic principle "Everything is an Object" that was important in the 80's to set up the object-oriented technology, we suggest here, in model driven engineering, that the basic principle "Everything is a model" may be central to identify the essential characteristics of this new trend. Applicability concerns various domains but more specially software engineering, system engineering and data engineering.*

MOTS-CLÉS : *MDA, ingénierie des modèles, MOF, QVT, ATL.*

KEYWORDS: *MDA, model engineering, MOF, QVT, ATL.*

RSTI - L'objet – 10/2004. Où en sont les objets ?, pages 145 à 156

## 1 Introduction

Il y a actuellement un changement de paradigme en ingénierie du logiciel, des systèmes et des données qui pourrait avoir des conséquences importantes sur la façon dont les systèmes à prépondérance logicielle sont construits et maintenus. Comme le notent J. Greenfield et K. Short dans (Greenfield *et al*., 2003) :

> "The software industry remains reliant on the craftsmanship of skilled individuals engaged in labor intensive manual tasks. However, growing pressure to reduce cost and time to market and to improve software quality may catalyze a transition to more automated methods. We look at how the software industry may be industrialized, and we describe technologies that might be used to support this vision. We suggest that the current software development paradigm, based on object orientation, may have reached the point of exhaustion, and we propose a model for its successor."

L'idée centrale de la composition d'objet est de plus en plus remplacée par la notion de transformation de modèle. On peut considérer ceci en continuité ou en rupture. L'idée des systèmes logiciels composés d'objets communicants n'est pas antinomique à l'idée du cycle de vie du logiciel considéré comme une chaîne de transformations de modèles. Les apports mutuels de la technologie des objets et de l'ingénierie des modèles devraient se cumuler mais les principes en sont fondamentalement différents. Cet article commence par présenter rapidement un exemple de solution d'ingénierie des modèles, celui de l'OMG baptisé MDA™. Il poursuit ensuite en prenant un peu de recul et en esquissant ce que pourrait être une base conceptuelle de l'ingénierie des modèles.

## 2. L'architecture MDA de l'OMG

En novembre 2000, l'OMG annonçait son initiative MDA (Soley *et al*., 2000). Le consensus sur UML a été essentiel dans cette transition des techniques de production de logiciel basées sur le code vers des techniques de production basées sur les modèles. Un rôle clef est maintenant joué par le concept de métamodèle. Mais ceci n'est pas suffisant. Le MOF (Object Management Group, 1997) est issu de la reconnaissance que UML était un métamodèle possible dans le domaine du développement logiciel, mais n'était pas le seul. Devant le danger de voir se développer et évoluer indépendamment une grande variété de métamodèles différents et incompatibles (data warehouse, workflow, software process, etc.), il y avait un besoin urgent de fournir un cadre global d'intégration pour tous les métamodèles dans le domaine de l'ingénierie du logiciel, des systèmes et des données. La réponse logique était donc d'offrir un langage de définition de métamodèles, c'est-à-dire un métamétamodèle, chaque métamodèle définissant lui-même un langage pour décrire un domaine spécifique d'intérêt. Par exemple UML permet de décrire les artefacts d'un logiciel à objets. D'autres métamodèles adressent des domaines différents comme le « legacy » (existant logiciel), les processus logiciels, l'organisation, les tests, la qualité de ser-

vice, etc. Leur nombre est important et croissant. Ils sont définis comme des composants séparés et de nombreuses relations existent entre eux.

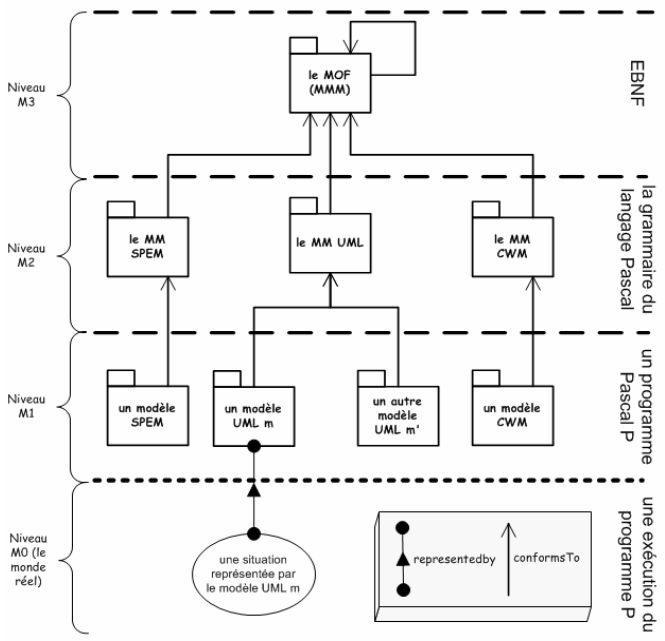


**Figure 1.** *Interprétation de la pile de modélisation multi-niveau de l'OMG*

Le changement réel en ingénierie des modèles est intervenu lorsque ces modèles ont commencé à être utilisés directement dans les chaînes de production de logiciel. Bien que cette possibilité ait souvent été considérée et partiellement appliquée, il est maintenant possible d'envisager son déploiement industriel à grande échelle (Greenfield *et al*., 2003). Jusqu'à présent les modèles d'analyse et de conception ont principalement été utilisés pour documenter les systèmes logiciels. Les analystes et les concepteurs ont produit des modèles souvent fournis aux programmeurs comme du matériau d'inspiration, pour faciliter la production de logiciel. Le passage de cette période « contemplative » à une nouvelle situation où les outils de production seront dirigés par les modèles a été facilité par l'introduction de standards du MDA comme la recommandation XMI (Object Management Group, 1998).

La question de la transformation de modèles se situe aussi au centre de l'approche MDA (figure 2). Une suggestion initiale de recherche avait initialement été faite en (Lemesle, 1998) et un appel à proposition industriel (RFP MOF/QVT) est actuellement en cours (Object Management Group, 2002) pour définir une sorte de langage unifié de transformation ou plutôt une famille coordonnée de tels langa-

ges. Ceci permettra de transformer un modèle *Ma* en un autre modèle *Mb*, indépendamment du fait que les métamodèles *MMa* et *MMb* de *Ma* et *Mb* soient identiques ou différents. De plus le programme de transformation, par exemple écrit en langage ATL (Bézivin *et al*., 2004) - un langage de la famille QVT - doit lui-même être considéré comme un modèle *Mt*. En conséquence il sera conforme à un métamodèle *MMt*, une définition abstraite de ce langage de transformation.

Ces éléments constituent donc les briques de base de ce que l'on appelle l'architecture MDA de l'OMG. Ils évoluent rapidement, donnant lieu à la réalisation d'outils industriels applicables et parfois appliqués à certains domaines spécifiques. Cependant, la communauté de recherche est elle-même impliquée dans la compréhension en profondeur des concepts et des principes régissant cette approche industrielle du MDA. Dans la section suivante, nous tentons d'identifier et de caractériser certains des concepts primitifs de l'ingénierie de modèles.

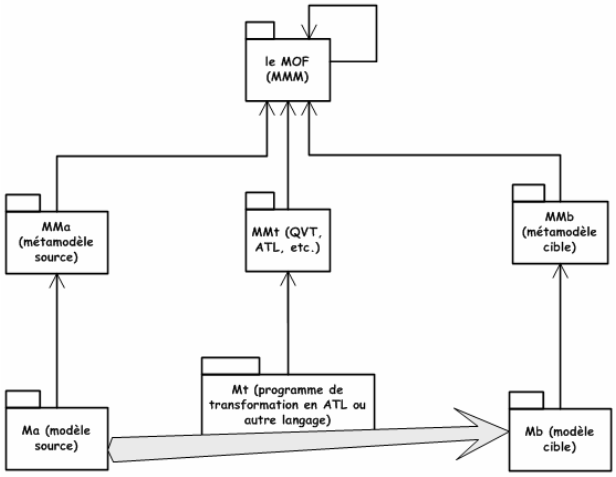


**Figure 2.** *La transformation de modèles basée sur les métamodèles*

## 3. Principes de base

En technologie des objets, un principe de base (« Tout est objet », [P1]) a été très utile lors de l'apparition de la technologie sur la scène industrielle dans les années 1980 pour la pousser dans la direction de la simplicité, de la généralité et de la puissance d'intégration. De façon similaire, en ingénierie des modèles, le principe de base « Tout est modèle » [P2] possède plusieurs propriétés intéressantes. Nous suggérons dans cet article que l'étude approfondie de ces propriétés peut se révéler fort utile dans la compréhension de nombreuses questions sur l'ingénierie des modèles en général et sur l'approche MDA en particulier.

| | |
|---|---|
| ***Tout est objet*** | **[P1]** |
| ***Tout est modèle*** | **[P2]** |

L'approche MDA n'est pas basée sur une idée unique. Parmi les objectifs poursuivis on peut mentionner la séparation de descriptions métier neutres d'avec les aspects liés à la plateforme, l'expression séparée des aspects d'un système en cours de développement par des langages spécifiques de domaines, l'établissement de relations précises entre ces différents langages dans un cadre global et en particulier la possibilité d'exprimer des transformations précises entre eux, etc.

Comme suggéré par les figures 3 et 4, les outils conceptuels qui étaient les plus utilisés dans les années 1980 sont en cours de renouvellement. Au début de la période de déploiement industriel de la technologie des objets, ce qui était important était qu'un objet soit instance d'une classe et qu'une classe puisse hériter d'une autre classe. Ceci peut être vu comme une définition minimale correspondant au principe [P1]. Nous appelons ces deux relations *instanceOf* et *inheritsFrom*. De façon très différente, ce qui semble être important aujourd'hui est qu'une vue particulière (un aspect) d'un système soit capturé par un modèle et que chaque modèle soit écrit dans le langage de son métamodèle. Ceci peut être vu comme une définition minimale correspondant au principe [P2]. Nous appelons les deux relations de base *representedBy* et *conformstTo*. Il est très probable que les interprétations possibles associées au principe [P2] vont générer des discussions vives sur la signification exacte et les propriétés de ces deux relations centrales et au moins aussi animées que celles qui ont conduit au consensus industriel actuel[1] sur les relations associées à [P1].

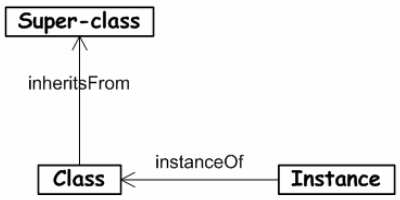


**Figure 3.** *Notions de base en technologie des objets [P1]*

1. On supposera que ce consensus industriel s'est actuellement établi autour des deux langages de programmation dominants que sont *Java* et *C#*. Il est clair que ceci ne couvre qu'une classe particulière de langages à classes et ne prend pas en compte d'autres familles de langages à objets comme le langage à prototypes *Self* par exemple.

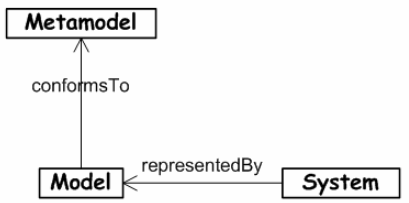


**Figure 4.** *Notions de base en ingénierie des modèles [P2]*

Le danger consisterait à utiliser les anciennes relations associées à [P1] dans le nouveau contexte de l'ingénierie des modèles, en affirmant par exemple qu'un modèle est une instance d'un métamodèle. Une telle affirmation n'apporterait pas plus d'information que celle qui consiste à dire, dans un contexte voisin, qu'un programme Java est une instance de la grammaire Java. Ce type de point de vue mène souvent à de nombreuses confusions et n'aide pas en général à clarifier une situation complexe.

Le fait de considérer différemment les ensembles de relations [P1] et [P2] permet aussi de bien illustrer que l'ingénierie des modèles et la technologie des objets offrent des vues plus complémentaires qu'opposées. Ces solutions se situent à des niveaux de représentation différents et le fait de bien les distinguer permet de mieux comprendre leurs apports respectifs et surtout leurs multiples complémentarités.

L'intérêt essentiel d'un métamodèle est de faciliter la séparation des préoccupations. Lorsque l'on considère un système donné, on peut travailler avec différentes vues de ce système, chacune de celles-ci étant caractérisée de façon précise par un métamodèle donné. Quand plusieurs modèles différents ont été extraits du même système à l'aide de métamodèles différents, ces modèles restent liés et pourront être recomposés par la suite. Pour que ceci puisse être largement appliqué, il est nécessaire de pouvoir disposer d'une organisation régulière des modèles composites.

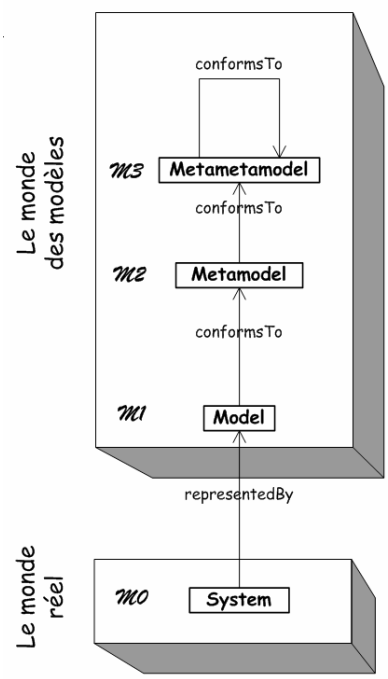


**Figure 5.** *L'organisation 3+1 du MDA*

L'organisation de la pile classique à quatre niveaux de l'OMG devrait plus précisément se nommer architecture 3+1 comme indiqué en figure 5. Au niveau le plus bas, la strate $M_0$ correspond au *système* réel. Un modèle *représente* ce système au niveau $M_1$. Ce modèle est *conforme* à son métamodèle défini au niveau $M_2$ et le métamodèle lui-même est *conforme* au métamétamodèle au niveau $M_3$. Le métamétamodèle est *conforme* à lui-même. Il s'agit d'une organisation pragmatique similaire à l'organisation des langages de programmation déjà présentée dans la partie droite de la figure 1. Une autoreprésentation de la notation EBNF prend quelques dizaines de lignes au plus. Cette notation permet de définir une infinité de grammaires bien formées. Une grammaire donnée, par exemple celle du langage Pascal, permet de définir une infinité de programmes Pascal syntaxiquement corrects.

La conséquence de l'application du principe [P2] au domaine du développement des systèmes à prépondérance logicielle est que nous pouvons actuellement observer l'apparition d'un grand nombre de métamodèles simples et de fort pouvoir d'abstraction, chacun définissant un langage spécifique de domaine. Il y a un besoin d'organiser de façon régulière cette énorme collection naissante de métamodèles, du point de vue conceptuel (Bézivin *et al*., 2001) ou pragmatique (Bézivin *et al*., 2003). Bien sûr l'approche MDA de l'OMG apparaît comme l'une des déclinaisons possibles de ces principes. Elle est basée sur quelques spécificités.

1) L'existence du MOF comme métamétamodèle unique ; le MOF est une ontologie de représentation basée sur des graphes non orientés avec étiquetage des extrémités des arêtes.

2) L'existence, dans la collection des métamodèles, du formalisme UML 2.0 sur lequel l'OMG a beaucoup investi dans la dernière période. Ce métamodèle UML possède pour le moment un statut privilégié à l'OMG car plusieurs outils AGL (comme Rational ROSE ou Poseidon) ont fait le choix d'en faire leur socle unique. Ces outils ne sont pas paramétrables par les métamodèles et il leur faut donc utiliser la technique encore assez peu formalisée des profils UML pour offrir un minimum d'extensibilité. Pour des raisons pratiques qui en découlent, le MOF est aligné sur une partie du formalisme UML.

3) L'existence au niveau $M_2$ d'une collection de métamodèles standard (ou de profils UML) qui permettent de prendre en charge des aspects liés aux données (CWM), aux systèmes (SysML), aux processus logiciels (SPEM), aux modèles d'entreprise (EDOC), etc.

4) La définition en cours d'un standard de transformation de modèles (MOF/QVT) indépendant des métamodèles.

5) Une vision encore embryonnaire de la façon de réaliser automatiquement la projection des modèles métier neutres (PIM) vers des plateformes spécifiques de façon à obtenir des modèles liés à la plateforme (PSM). Hormis le fait que cette projection s'appuiera sur des techniques de transformation de modèles, il n'y a pas encore de guide méthodologique précis correspondant. Deux possibilités sont essentiellement explorées : l'utilisation de décorations informelles de modèles (utilisant par exemple les profils UML) et la prise en compte de modèles explicites de plateformes. La première approche est pragmatique et informelle et la seconde est plus ambitieuse mais sa faisabilité n'a pas encore été démontrée sur des cas réels de grande taille.

Mais il y a aussi en ingénierie des modèles des approches divergentes et des variantes de l'approche MDA. Récemment des sociétés comme Microsoft ou IBM ont défini leurs positions à ce sujet. Dans le « manifeste MDA » (Booch *et al*., 2004) d'IBM les principes de base sont au nombre de trois :

1) la **représentation directe**, c'est-à-dire la nécessité de disposer de familles de langages de domaines (DSL) permettant de prendre en compte chacune des situations et des communautés corporatives ;

2) l'**automatisation** permettant les traitements de mise en correspondance automatique des modèles conformes aux différents langages de domaines ;

3) les **standards ouverts** permettant l'émergence rapide d'écosystèmes d'industriels utilisateurs et fournisseurs d'outils et de chercheurs appliqués autour de plateformes de logiciel libre utilisant ces standards, par exemple ceux basés sur MOF ou sur XML.

Les visions de l'OMG (MDA), de Microsoft et d'IBM sont des variantes de l'ingénierie des modèles. Elles partagent les concepts mais pas forcément tous les

standards. Dans les différentes approches proposées pour l'ingénierie des modèles, on retrouve les deux principes de base associés à [P2]. La façon dont ils sont appliqués est assez variable. La période dans laquelle nous entrons sera riche en expérimentations et réalisations diverses dans ces domaines. Il est probable que le résultat de ce travail sera une formalisation des deux relations de base de *conformité* et de *représentation*. Dans les deux cas, une interprétation purement ensembliste est insuffisante de la même façon que la caractérisation d'une classe comme l'ensemble de ses instances en programmation par objets et beaucoup trop réductrice. Dire que la relation de représentation est une sélection (filtrage basé sur un métamodèle) d'un ensemble d'éléments d'un système pour construire l'ensemble des éléments d'un modèle ne prend en compte qu'une partie de cette opération. Dire que la relation de conformité consiste à associer tout élément d'un modèle à son métaélément de définition dans le métamodèle est tout aussi simpliste et ne capture également qu'une partie de la complexité de cette relation.

Mais même si la réflexion sur la caractérisation précise de ces deux relations de base ne fait que débuter, les conséquences pratiques d'une application du principe [P2] apparaissent déjà. Considérer les modèles comme des entités de première classe a déjà aujourd'hui une signification précise. Dans (Bézivin *et al*., 2003) certaines conséquences en avaient déjà été tirées, notamment :

1) dans un atelier de développement, chaque outil implémente un certain nombre d'opérations précises sur les modèles. Il n'y a pas d'outil universel. Certains font de la capture et de l'édition de modèles, d'autres de la transformation, d'autres de la confrontation, de la vérification, du test, du tissage, de la mesure, etc. et en général une combinaison restreinte d'opérations. Les outils peuvent communiquer, par exemple grâce à un bus logiciel qui s'appuie sur des normes de communication ;

2) le stockage et l'accès aux modèles et aux métamodèles se fait de façon uniforme. Des fonctionnalités comme la gestion de configuration ou de versions des modèles sont donc factorisées ;

3) chaque opération prend en entrée des modèles et fournit en sortie des modèles. La signature de l'opération spécifie les métamodèles correspondants qui jouent donc le rôle de types pour ces modèles. Un outil met en œuvre un ensemble d'opérations et sa signature se compose de l'ensemble des signatures de ces opérations ;

4) une opération peut être considérée comme un modèle. C'est par exemple le cas des opérations de transformations. Dire qu'une opération est un modèle est une affirmation importante qui a plusieurs conséquences. Tout d'abord, il est possible de produire dynamiquement ou de modifier des modèles correspondant à des opérations de haut niveau. Ceci signifie aussi que chaque opération s'appuie sur un métamodèle.

Au-delà de la proposition spécifique MDA de l'OMG, on comprend mieux que l'ingénierie des modèles est essentiellement une forme particulière d'ingénierie des langages. Elle a cependant plusieurs spécificités :

1) elle s'intéresse à des langages de modélisation et non pas uniquement à des langages de programmation. Contrairement à ces derniers, un langage de modélisation n'est pas systématiquement et canoniquement exécutable ;

2) la nouvelle dimension de l'ingénierie des modèles se situe plus dans la relation de représentation que dans la relation de conformité. En effet, la relation de conformité entre un modèle et son métamodèle est très similaire à la relation entre un programme et la grammaire du langage dans lequel il a été écrit. En revanche, la question soulevée par la relation de représentation est liée à une problématique nouvelle et importante, de nature ontologique.

Un programme classique est en effet une représentation composite de plusieurs aspects que l'on n'a pas pu séparer. Le programme représente entre autre l'environnement dans lequel il va fonctionner, la plateforme sur laquelle il est destiné à s'exécuter et les besoins applicatifs auxquels il est supposé répondre. Si le programme est un modèle, de quoi exactement est-il un modèle ? Répondre à cette question signifie d'abord être capable de séparer les différentes préoccupations présentes dans ce programme.

Il est intéressant de noter que c'est la technologie des objets qui a déclenché l'évolution vers cette prise de conscience. Après avoir appris à construire des programmes homogènes constitués d'objets, on s'est posé la question de connaître l'origine de ces objets et on s'est aperçu de la multiplicité de leurs natures et de leurs rôles. On a commencé par considérer différemment les objets métier (en provenance de l'environnement) et les objets techniques (en provenance de la plateforme). On a ensuite réalisé la grande complexité du problème et l'incapacité des langages de programmation à le prendre en compte au-delà des aspects syntaxiques purs. Or l'organisation de chacun des aspects, par exemple des aspects métier, relève elle-même d'un langage spécifique de domaine qu'il importe de définir de façon précise, complète et rigoureuse.

Les approches des patterns et des aspects sont des tentatives pour étendre les capacités des langages de programmation à objets dans cette direction. L'ingénierie des modèles est une tentative plus radicale visant à fournir un grand nombre de langages spécifiques de domaines (DSL) pour exprimer séparément chacune des multiples préoccupations. Le problème du tissage des aspects se traduit alors par une intégration de langages, la transformation de modèles ne représentant qu'une solution très partielle à ce problème.

Expliciter de façon distincte les principes de base de la technologie des objets et de l'ingénierie des modèles permet de mieux comprendre les raisons des évolutions actuelles (aspects et modèles par exemple) et en particulier de mieux apprécier leurs possibles complémentarités.

## 4. Conclusion

Le passage de la technologie procédurale à la technologie des objets a déclenché un changement radical dans notre façon de considérer les systèmes d'information et de conduire les opérations de construction et de maintenance en ingénierie du logiciel, des systèmes et des données. Mais, contrairement à ce que l'on avait pu penser initialement, cette technologie des objets ne constitue pas un aboutissement mais au contraire un point de départ vers de nouvelle migrations technologiques.

L'un de ces chemins possibles de migration se nomme ingénierie des modèles. Il consiste à donner un statut de première classe aux modèles et aux éléments de modèles de la même façon que le statut d'entités de première classe fut donné aux objets et aux classes d'objets dans les années 1980, au début de l'ère de la programmation par objets. Le changement essentiel provient du fait que les modèles ne sont plus perçus comme uniquement documentaires, à des fins de guidage d'une activité humaine de programmation, mais qu'ils peuvent maintenant être directement utilisés pour alimenter les outils de production automatique de logiciel.

Le passage des objets aux modèles peut donc être vu en continuité ou en rupture. De la même façon que l'arrivée de la programmation par objets n'a pas invalidé les apports de la programmation structurée, le développement dirigé par les modèles ne contredit pas du tout les apports de la technologie des objets. Il ne faut pas considérer ces solutions comme antagonistes mais complémentaires. Des ponts de plus en plus précis existent, comme le standard JMI permettant de passer du monde des modèles au monde de la programmation Java. Une telle passerelle est complémentaire des autres passerelles vers le monde des données semi-structurées (XMI) ou encore du middleware Corba (CMI), etc. Les rapides développements actuels sur la base du système EMF (Eclipse Modeling Framework) sont une démonstration éclatante de la complémentarité opérationnelle entre le monde des objets et celui des modèles. Des analogies entre les approches par objets et composants et les approches par modèles permettent de mieux comprendre leurs apports respectifs (Bézivin *et al*., 2004).

Le seul point de friction entre les deux propositions est celui de l'intégration de paradigmes. Initialement, la technologie des objets se voulait aussi une technologie d'intégration car il était théoriquement possible de prendre en compte de façon uniforme les processus, les règles, les fonctions, etc. par des objets. Aujourd'hui on en revient à une vision moins hégémonique où les différents paradigmes de programmation coexistent paritairement, sans donner plus d'importance à l'un ou à l'autre. Une vision intégratrice pourra se développer dans le cadre du MDE par prise en compte de DSL basés sur des paradigmes différents et génération vers les supports correspondants. La technologie des objets a conquis ses lettres de noblesse dans la mise en œuvre des grands systèmes exécutables. Elle sera donc une cible privilégiée de génération pour le MDE, mais ce ne sera pas la seule car il y en aura aussi d'autres comme la technologie des données semi-structurées (XML), les règles, le fonctionnel, etc.

Malgré toutes ces convergences, il y a cependant un intérêt pédagogique certain à considérer le passage du monde des objets et de celui des modèles comme une réelle rupture. En effet, si l'on ne prend pas conscience qu'il s'agit bien de perspectives différentes, on risque de créer beaucoup de confusion. Certains peuvent par exemple considérer le langage UML 2.0 comme une syntaxe graphique pour Java, C++ ou C# et l'on sait les dangers d'une telle vision simpliste. D'autres vont considérer un modèle comme une instance d'un métamodèle comme on l'a précédemment noté. L'ingénierie des modèles n'est pas basée sur les objets de façon intrinsèque et elle permet de prendre en compte des métamodèles à objets mais aussi bien d'autres comme des métamodèles relationnels, de règles, de workflow, etc. Il est donc nécessaire de bien montrer que l'on travaille dans des dimensions différentes et que les relations de base de la technologie des objets (héritage et instanciation essentiellement) ne sont pas les mêmes que celles de l'ingénierie des modèles (représentation et conformité essentiellement). Après tout, il a bien fallu créer une rupture pour passer du monde procédural au monde des objets et la situation était plus simple puisque l'on restait dans des perspectives de programmation.

Dire que l'ingénierie des modèles s'appuie essentiellement sur les relations de représentation et de conformité, c'est montrer que ses sources d'inspiration sont l'ingénierie des langages (conformité) et l'ingénierie ontologique (représentation). On pourrait même aller plus loin en suggérant qu'elle tente une possible synthèse entre ces deux courants.

Le fait de tout considérer comme des modèles dans une approche de production de logiciel a plusieurs conséquences positives qui apparaissent progressivement. Le MDA est l'une des premières applications industrielles significatives de ces principes. On verra probablement d'autres propositions similaires se développer dans un futur proche, par exemple celle des « software factories » de Microsoft (Greenfield *et al*., 2003). Ce qui est important c'est de bien comprendre la portée de ces évolutions et les bases conceptuelles sur lesquelles elles s'appuient.

**Remerciements**

Je voudrais remercier les membres du séminaire Dagstuhl #04101 et du groupe AS CNRS sur l'ingénierie des modèles pour les nombreuses discussions sur les sujets évoqués dans cet article. La responsabilité de la façon dont leurs suggestions ont été interprétées dans cette version reste entièrement la mienne. Je voudrais aussi remercier plus particulièrement Jacky Estublier, Jean-Marie Favre, Jean-Marc Jézéquel, Frédéric Jouault et Bernard Rumpe pour de nombreuses discussions éclairantes sur ce sujet.

Au cours des vingt dernières années, Jean-François Perrot a été, dans notre communauté française, l'accompagnateur fidèle, l'observateur critique et bien souvent l'inspirateur modeste mais efficace de ces évolutions profondes des procédures aux objets et des objets aux modèles. Je lui dédie les quelques éléments de réflexion présentés dans cet article.

## 5. Bibliographie

Bézivin J., Gérard S., Muller P.A., Rioux L., "MDA Components: Challenges and Opportunities", *Metamodelling for MDA Workshop*, York, 2003. http://www.sciences.univ-nantes.fr/info/lrsg/Pages_perso/JB/Jean.Bezivin.html

Bézivin J., Gerbé O., "Towards a Precise Definition of the OMG/MDA Framework", *ASE'01, Automated Software Engineering*, San Diego, USA, November 26-29, 2001. http://www.sciences.univ.nantes.fr/info/perso/permanents/atl/publications/ASE01.OG.JB.pdf

Bézivin J., Jouault F., Rosenthal P., Valduriez P., "Modeling in the Small and Modeling in the Large", article soumis à publication, septembre 2004.

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

Greenfield J., Short K., "Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools", *OOPSLA'03*, Anaheim, California, 2003.

Lemesle R., Transformation rules based on meta-modeling, *EDOC'98*, San Diego, November 3-5, 1998, http://www.sciences.univ-nantes.fr/info/lrsg/Pages_perso/RL/Publications/EDOC98-lemesle.pdf

Object Management Group, OMG/MOF Meta Object Facility (MOF) Specification, September 1997, http://www.omg.org/docs/ad/97-08-14.pdf

Object Management Group, OMG/RFP/QVT MOF 2.0 Query/Views/Transformations RFP, October 2002, http://www.omg.org/docs/ad/02-04-10.pdf

Object Management Group, XML Model Interchange (XMI), October 1998, http://www.omg.org/docs/ad/98-10-05.pdf

Soley R., and the OMG staff: Model-Driven Architecture, November 2000, ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf