Un formalisme pour la gestion des connaissances

Approche ingénierie dirigée par les modèles

Thi-Lan-Anh Dinh*, ** — Olivier Gerbé — Houari Sahraoui***

** Département d'informatique et de recherche opérationnelle*
Université de Montréal
CP 6128 succ. Centre Ville, Montréal, Québec, Canada, H3C 3J7
{dinhthil, sahraouh}@iro.umontreal.ca

***HEC Montréal*
3000 Chemin de la Côte-Sainte-Catherine, Montréal, Québec, Canada H3T 2A7
{lan-anh.dinh-thi, olivier.gerbe}@hec.ca

RÉSUMÉ. *Les nombreux travaux de recherche autour de IDM (ingénierie dirigée par les modèles) montrent que la gestion de modèles prend une importance grandissante et intervient dans divers domaines comme la gestion des connaissances, la gestion de métadonnées, les ontologies, la qualité de service et le génie logiciel. La représentation de modèles est fondamentale pour la gestion de modèles. Comme les formalismes de modélisation couramment utilisés ne rencontrent pas toutes les exigences pour la représentation et la gestion de modèles de connaissances, nous proposons dans cet article un nouveau formalisme qui rencontre toutes ces exigences. Notre solution supporte IDM. Elle cible le domaine de la gestion des connaissances en particulier mais pourrait se servir des applications d'autres domaines où le besoin de modéliser des choses et/ou représenter des modèles se présente.*

ABSTRACT. *Research work in the fields of Model-Driven Engineering (MDE) and Model-Driven Development (MDD) shows that model management is becoming increasingly important and is implicated in a wide range of areas such as knowledge management, metadata management (databases), ontologies, quality of service, and software engineering. Model representation is fundamental in model management. Existing formalisms do not meet all essential requirements for representing and managing knowledge models. In this context, we introduce in this paper a new formalism which meets all these requirements. Our solution supports IDM. It targets the field of the knowledge management in particular but could be used for applications in other fields where the need to model things and/or to represent models arises.*

MOTS-CLÉS : *modélisation, métamodélisation, modèle, métamodèle, méta-métamodèle, représentation de connaissance, ingénierie de modèles.*

KEYWORDS: *modeling, meta-modeling, model, metamodel, meta-metamodel, knowledge representation, model engineering.*

DOI:10.3166/ISI.12.5.109-132

1. Introduction

L'approche ingénierie dirigée par les modèles ou IDM (Bézivin *et al.,* 2005a ; Girard *et al.,* 2005 ; Jézéquel *et al.,* 2005) qui a succédé au *Model Driven Architecture* (MDA) (OMG, 2003a) a pour objectif de définir un cadre pour la génération de code par des transformations successives de modèles. L'émergence de l'IDM a suscité beaucoup d'intérêt pour les activités de modélisation et de gestion de modèles. La gestion de modèles traite des mécanismes qui permettent de représenter, créer, stocker et manipuler les modèles. La gestion de modèles intervient dans des domaines aussi divers que les bases de données, la qualité de service, les ontologies, et la gestion des connaissances.

Dans le domaine des bases de données, la gestion des métadonnées exige la manipulation de structure de données plutôt que des données elles-mêmes. L'approche basée sur la gestion des modèles devient une piste de solution pour plusieurs problèmes du domaine tels que l'intégration et la traduction de données, l'intégration et la transformation des schémas (Alagic *et al.,* 2001 ; Bernstein *et al.,* 2000 ; Melnik, 2004).

Dans le domaine de la qualité de service, afin d'assurer le fonctionnement des applications dans un environnement complexe tel que celui des systèmes multimédias distribués, l'intégration de l'information de gestion et l'exécution distribuée des activités de la qualité de service sont actuellement un défi (Kerhervé *et al.,* 2001). La gestion des modèles d'information de la qualité de service (Gerbé *et al.,* 2003a) s'inscrit aussi dans ce défi.

Dans le domaine des ontologies pour le besoin de partage des connaissances entre différents domaines (Corcho *et al.,* 2001), la représentation, la modélisation et la gestion des ontologies sont des questions primordiales.

Enfin dans le domaine de la gestion des connaissances, les informations doivent être représentées sous forme des modèles pour être gérées. L'intérêt sémantique de la modélisation ici prime sur l'intérêt d'opérationnalisation. Ce qui est recherché avant tout dans ce domaine, c'est la compréhension du monde à modéliser et une représentation ou documentation la plus précise mais aussi la plus souple possible. La gestion des connaissances dans les entreprises, c'est-à-dire le développement de mémoires corporatives, pose principalement un problème de quantité, de complexité et de diversité (Gerbé, 2000 ; Dinh *et al.,* 2004). Ceci implique un véritable défi pour la représentation et la modélisation de cette mémoire ainsi que pour la gestion de modèles s'y appliquant.

Cet article propose un formalisme pour la représentation des modèles de connaissances et met l'accent sur le support IDM de la solution. Ce formalisme, bien que ciblant le domaine de la gestion des connaissances, pourrait être utilisé dans d'autres domaines. Le reste de l'article est organisé comme suit. La section 2 décrit l'architecture de modélisation. La section 3 présente les notions de base. La section 4 porte sur les besoins concernant la représentation de modèles de connaissances.

Nous présentons notre formalisme dans la section 5 et son pouvoir d'ingénierie de modèles dans la section 6. La section 7 présente d'autres formalismes de modélisation. La section 8 conclut et présente nos travaux futurs.

2. Architecture de modélisation

Notre architecture de modélisation (Dinh *et al.,* 2006) est conforme à celle à quatre niveaux (M0, M1, M2 et M3) largement acceptée aujourd'hui (Sahraoui, 1995 ; Lemesle, 2000 ; Bézivin *et al.,* 2001 ; Dinh *et al.,* 2004 ; Girard *et al.,* 2005). Nous décrivons brièvement ci-après le rôle de chacun de ces niveaux :

– M3 (méta-métamodèle) est le niveau le plus abstrait. Il est réflexif et définit les notions de base permettant la représentation des niveaux inférieurs ainsi que lui-même ;

– M2 est le niveau métamodèle. En utilisant la grammaire spécifiée au niveau M3, ce niveau définit le langage et la grammaire pour représenter des modèles au niveau M1 ;

– M1 est le niveau modèle. Ce niveau définit des représentations concrètes du monde réel (modèles) ainsi que des descriptions de ces représentations. Chaque modèle au niveau M1 est défini conformément à son métamodèle au niveau M2 ;

– M0 est le monde réel décrit par le niveau M1.

On remarque une évolution dans les architectures de modélisation. Si le nombre de niveaux est inchangé, le rôle des niveaux a été modifié. Pour VODAK (Dahchour, 2001), IRDS (ISO., 1990), Telos (Mylopoulos *et al.*, 1990), MENINGE (Rieu *et al.*, 1997), Modèle uniforme (Gerbé, 2000), UML1.x, etc., le niveau M0 représentait les instances, et le monde réel n'était pas pris en compte. La clarification des activités de modélisation a permis de préciser : (i) seuls les trois derniers niveaux (M3, M2, M1) appartiennent au monde de la modélisation tandis que le niveau M0 est le monde réel ; (ii) ce que l'on appelle couramment les types et les instances sont au même niveau M1 (Lemesle, 2000 ; Bézivin *et al.*, 2001), contrairement à ce qui est souvent perçu.

De plus, afin d'éviter des problèmes dus à la représentation de la nature des concepts (y compris des modèles), comme par exemple le problème de l'instanciation double discuté dans (Bézivin *et al.*, 2001 ; Dinh *et al.,* 2004), il est important de bien distinguer les notions suivantes (Dinh *et al.,* 2006) : l'*instanciation* (le rapport « types-instances » entre éléments du niveau M1), la *conformité entre éléments* (le rapport de conformité entre un élément et son méta-élément), et la *conformité entre modèles* (le respect sémantique entre un modèle et son métamodèle).

La figure 1 illustre ces différentes notions. Dans cette figure, l'élément *Marie* au niveau M1 représente la vraie personne Marie (du niveau M0). Cet élément est d'une part, conforme à *Object* dans le contexte global (ce qui est indiqué par le lien de type

meta sur l'axe vertical) et d'autre part, une instance de *Personne* dans le contexte local (ce qui est indiqué par le lien de type *instOf* sur l'axe horizontal). Les éléments de M2 tels que *Objetc*, *instOf* sont conformes à leurs méta-éléments correspondants de M3 (ce qui est indiqué par les liens de type *Mmeta* sur l'axe vertical). Sur l'axe vertical, la figure 1 montre également, par les liens de type *Msem*, que M3 est à la fois son propre métamodèle et celui de M2. M2 est quant à lui le métamodèle de M1 par le lien de type *sem*.

Il est à souligner que la réflexivité du niveau M3 possède des avantages multiples. La réflexivité permet de limiter le nombre de niveaux d'abstraction. Le niveau réflexif se valide lui-même ; les outils et algorithmes applicables au niveau métamodèle sont donc aussi applicables à ce niveau (Lemesle, 2000). Ceci facilite l'adaptation aux extensions ou modifications futures.

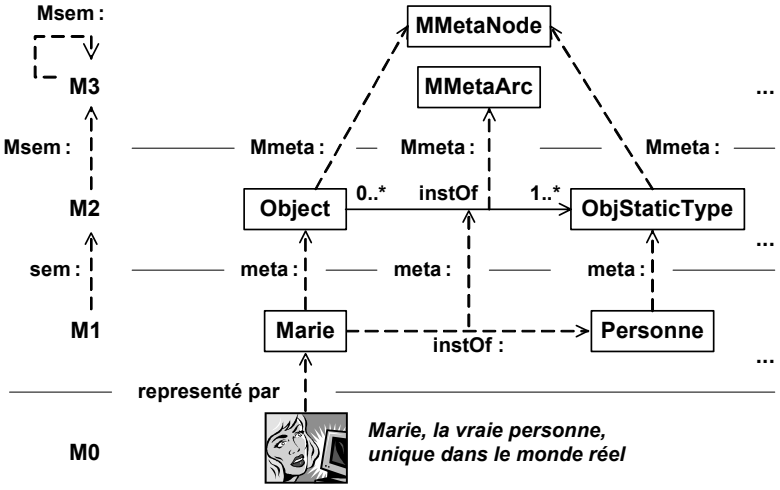

Figure 1. *Notre architecture de modélisation*

3. Notions de base

Cette section présente les définitions des notions de base que nous utilisons dans le reste de cet article. Ces notions sont couramment utilisées dans les techniques de modélisation; certaines d'elles peuvent être nommées autrement à travers différents formalismes. Ces définitions permettront de clarifier les notions utilisées afin de réduire tout malentendu et ambiguïté.

D'abord, nous précisons la notion de modèle utilisée dans notre contexte. Nous appellerons *modèles* tous ce que nous pouvons retrouver dans les différents niveaux de modélisation. Parmi les modèles, nous distinguons les métamodèles et méta-métamodèles en utilisant les règles suivantes (Lemesle, 2000) : (i) un modèle

appartient à M3 s'il est aussi son propre métamodèle ; (ii) un modèle appartient à M2 si son métamodèle est un modèle du niveau M3 ; et (iii) un modèle appartient à M1 si son métamodèle est modèle du niveau M2.

Définition 1. *Modèle*. Un *modèle*, est un codage d'une représentation de quelques aspects de la réalité ou de l'univers du discours, et est représenté par un ensemble d'éléments et de relations entre eux (Dinh *et al.,* 2005).

Le mécanisme de typage nous permet de définir et d'organiser d'une manière générique les informations à modéliser. La définition 2 précise les éléments d'un modèle et les relations qu'ils entretiennent. La définition 3 décrit les relations entre niveaux. Les définitions 4 et 5 spécifient les relations de sous-typage.

Définition 2. *Type/Instance ; Type non relationnel/Occurrence ; Type relationnel/Relation*.

Un *type* représente un ensemble d'éléments ayant les mêmes propriétés. Une *instance d'un type* représente un élément qui se conforme à la définition du type. Parmi les types, nous distinguons les *types relationnels* (types de relations; associations) et les *types non relationnels* (type d'entités ; classe; catégorie). Parmi les instances, nous distinguons les *relations* – instances des types relationnels (unaires/binaires/n-aires) – et les *occurrences* (entités) – instances des types non relationnels (types d'entités).

Définition 3. *Elément/Méta-élément ; Nœud/Métanœud ; Lien/Métalien ; Arc/Méta-arc*.

Un *élément* (entité, relation, etc.) défini à un niveau de modélisation est conforme à un type défini au niveau méta appelé *méta-élément* (méta-entité, métarelation, etc.). Un *élément* se conforme à un et un seul *méta-élément*. Parmi les *éléments* (*instances*) définis à un niveau de modélisation, il convient de distinguer les *éléments-nœuds* (*occurrences*) et les *éléments-liens* (*relations*). Un *nœud* (terme abrégé de « élément-nœud »), défini à un niveau de modélisation, est conforme à un type non relationnel défini au niveau méta ; ce type est dit *métanœud* du premier. Un *lien* (terme abrégé de « élément-lien ») n-aire, défini à un niveau de modélisation, représente une relation n-aire reliant *n* éléments (*n* est un entier positif), son type relationnel est défini au niveau méta et appelé *métalien* n-aire du premier. Un *arc* (terme abrégé de « élément-arc ») représente un lien binaire et unidirectionnel, liant un élément source à un élément destination ; et son métalien, binaire et unidirectionnel, appelé *méta-arc*, est défini pour unir le méta-élément source au méta-élément destination.

Définition 4. *Relation de sous-typage*. Si un type *B* est sous-type de *A*, chaque instance de *B* peut se comporter comme instance de *A* et elle compte parmi les instances de *A*. Egalement, si *B* est un type relationnel, les éléments unis par *B* peuvent aussi être unis par *A*. La relation de sous-typage est transitive et acyclique.

Définition 5. *Type abstrait*. Un type dit *abstrait* n'a pas directement d'instances. Il sert à organiser et à éclaircir sémantiquement une hiérarchie de spécialisation de types.

4. Besoins

Cette section recense les besoins essentiels qu'un formalisme doit supporter pour représenter toutes les sortes de métamodèles.

Besoin 1. *Typage entre éléments de deux niveaux de modélisation consécutifs*.

Etant donné que parmi deux niveaux de modélisation consécutifs le niveau inférieur respecte la grammaire spécifiée par son niveau supérieur, les éléments du niveau inférieur sont vus comme des instances dont leurs types sont définis au niveau supérieur.

Pour représenter ce typage, les notions suivantes sont nécessaires : méta-élément, métanœud, et métalien ; et pour représenter les éléments d'un niveau inférieur, les notions suivantes sont également nécessaires : élément/nœud/lien. Il est à noter qu'un lien binaire peut exister sous toutes les formes, c'est-à-dire, un lien entre deux nœuds, ou entre un nœud et un lien, ou entre deux liens ; de même pour le niveau méta.

Besoin 2. *Passage entre niveaux types et instances*.

Il est nécessaire de pouvoir se déplacer entre les différents niveaux de modélisation.

Un élément doit donc pouvoir être traité comme *instance* par rapport à un niveau supérieur mais aussi comme *type* par rapport à un niveau inférieur (Gerbé, 2000). Par exemple, *ObjStaticType* est vu comme une « instance » de *MMetaNode* et comme le « type » de *Personne*.

Besoin 3. *Distinction entre différentes notions de conformité au niveau méta*

Nous pouvons distinguer deux niveaux de conformité : conformité entre éléments et conformité entre modèles et métamodèles.

Le formalisme doit donc supporter ces deux niveaux de conformité. La relation conformité entre éléments et méta-éléments permet de définir la nature d'un élément et la relation de conformité entre modèles et métamodèles permet d'indiquer que le modèle respecte les règles définies dans le métamodèle.

Besoin 4. *Hiérarchisations entre types*

La plupart des formalismes de modélisation permettent la hiérarchisation entre types non relationnels. Cependant un type relationnel peut lui aussi avoir des attributs propres, des comportements et des relations avec d'autres éléments.

Le formalisme doit donc supporter non seulement la hiérarchisation entre types non relationnels mais aussi entre types relationnels.

Besoin 5. *Contraintes de cardinalités min/max pour un métalien*

La notion de cardinalité permet de représenter les contraintes sur les types relationnels en précisant le nombre des éléments impliqués. Par exemple, nous voulons pouvoir spécifier la contrainte suivante : *un* élément est conforme à *un et un seul* méta-élément.

Le formalisme doit donc supporter cette notion de cardinalité qui permet de capturer le nombre minimal/maximal des liens possibles (conformément au métalien défini au niveau méta) par rapport aux éléments impliqués dans un lien.

Besoin 6. *Distinction entre modèles de différentes natures*

Nous pouvons distinguer différents types de modèles comme, par exemple, *métamodèle* d'un modèle ; *modèle structurel* d'un type relationnel (qui spécifie comment ce type relationnel relie ses éléments) ; *modèle de conditions* (pour contextualiser des conditions ou bien des hypothèses).

Le formalisme doit donc permettre de spécifier ces différentes natures afin de les représenter, de les classer et de les gérer efficacement.

Besoin 7. *Liens avec les modèles*

Il est nécessaire de distinguer les différents types de liens entre les modèles et leurs éléments ou entre modèles. Nous donnons ici quelques exemples : les liens de *contenant* entre un modèle et ses éléments (pour indiquer clairement le rapport de type « être défini dans », ou « être contenu dans », etc., entre un modèle et ses éléments) ; les liens d'*accès* entre modèles (pour réutiliser dans un modèle des éléments d'un autre modèle) ; le lien de *respect sémantique* entre un modèle et son métamodèle ; et des liens d'autres types (pour représenter explicitement des opérations entre modèles) comme *jointure*, *intersection*, *différence*, *inférence*, *restriction* entre modèles.

Le formalisme doit donc permettre de spécifier les différents types de liens afin de permettre de développer des mécanismes de validation et de gestion.

5. Méta-métamodèle pour la gestion de modèles

Le méta-métamodèle (M3) fournit le langage et la grammaire pour décrire les formalismes de modélisation. Notre méta-métamodèle est inspiré de celui des graphes conceptuels (Gerbé, 2000 ; Sowa, 1984) et des sNets (Lemesle, 2000).

Comme pour les réseaux sémantiques (Sowa *et al.,* 1991), la représentation dans notre formalisme est basée sur les *nœuds* et les *arcs*. Un modèle est représenté par un ensemble d'éléments et de relations entre ces éléments. Un élément est représenté par un nœud ou un arc. Une relation unaire, binaire ou n-aire qui unit un ensemble d'éléments est représentée par un nœud et un ensemble d'arcs. Un nœud est

représenté graphiquement par un rectangle et un arc par une flèche pointillée, allant de l'élément source à l'élément destination. Un élément *A* peut s'écrire sous la forme « *MetaA* : *A* » pour indiquer que *A* est conforme à son méta-élément *MetaA*.

5.1. *Eléments de base de M3*

Les éléments de base du niveau M3 sont : *MElement*, *MNode*, *MArc*, *MMetaElement*, *MMetaNode*, *MMetaArc*, *MsubType*, *Mmeta*, *Msrce*, et *Mdest*.

MElement, MNode, MArc. Ces métanœuds sont des types abstraits. *MElement* est à la racine de la hiérarchie de tous les méta-éléments de M3. Cette hiérarchie se divise en deux branches : l'une pour tous les métanœuds de M3 dont *MNode* est à la racine, et l'autre pour tous les méta-arcs de M3 dont *MArc* est à la racine. Chaque élément défini à un niveau méta (M2 ou M3) est conforme à un et un seul méta-élément de M3.

MMetaElement, MMetaNode, MMetaArc. Un méta-élément est aussi un élément au niveau méta, il est donc conforme à un méta-élément de M3. Pour cette raison, *MMetaElement* représente l'ensemble de tous les méta-éléments. Ces méta-éléments, exceptés ceux conformes à *MMetaElement*, sont classifiés en deux groupes : les métanœuds, et les méta-arcs. Chaque élément considéré comme un métanœud (ou un méta-arc) doit être conforme à *MMetaNode* (ou à *MMetaArc*). Puisque les méta-éléments sont aussi des nœuds, les éléments *MMetaElement*, *MMetaNode*, *MMetaArc*, étant métanœuds, sont donc sous-types de *MMetaNode*.

Msrce, Mdest. Les méta-arcs *Msrce* et *Mdest* permettent de spécifier les sources et destinations des méta-arcs. Un méta-arc unit deux méta-éléments *via* un arc de type *Msrce* et un arc de type *Mdest*. La figure 2 illustre un exemple de structures de méta-arcs. *Msem*, un méta-arc défini au niveau M3 pour représenter au niveau méta l'ensemble des liens de *respect sémantique* entre les modèles (*MModel*) et leurs métamodèles (*MMetaModel*), lie *MModel* à *MMetaModel via* un arc de type *Msrce* et un arc de type *Mdest*.

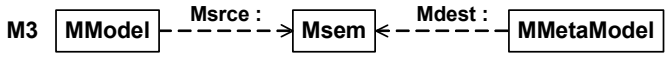

Figure 2. *Structure d'un méta-arc*

MsubType. Le méta-arc *MsubType* implémente la relation de sous-typage entre méta-éléments. L'effet de cette relation nous permet, en particulier, de déduire de nouvelles structures pour un méta-arc à partir de sa structure initiale, en remplaçant l'élément source ou destination dans cette structure par un de ses inférieurs (cf. Règle 1 ci-après).

Mmeta. Le méta-arc *Mmeta* représente l'ensemble des liens de conformité qui associent les éléments à leurs méta-éléments, soit du même niveau M3 ou du niveau M2 au niveau M3. Dans le deuxième cas, ces liens jouent le rôle de transition entre les niveaux M2 et M3. Un élément (nœud ou arc) de M3 ou M2 a un et un seul méta-élément (respectivement métanœud ou méta-arc) auquel il est rattaché, une fois défini, par un arc de type *Mmeta*. La règle existentielle entre un élément et son méta-élément doit être respectée (cf. Règle 3 ci-après). La figure 3 montre un exemple. Dans cette figure, au niveau M3, le méta-arc *MdefIn* (rectangle) lie *MMetaNode* à *MMetamodel via* un arc de type *Msrce* et un arc de type *Mdest* (flèches pointillées). Ceci spécifie que les métanœuds sont définis dans les métamodèles. Afin de représenter que le métanœud *Class* est défini dans le métamodèle *M2Uml*, l'arc qui associe *Class* à *M2Uml* est conforme à son méta-arc *MdefIn*.

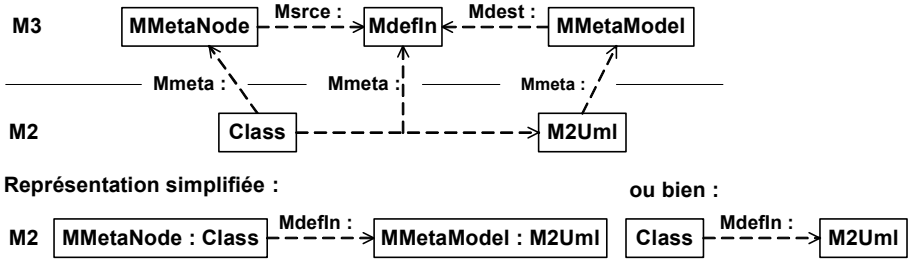

Figure 3. *Rapport existentiel entre un arc et son méta-arc*

Nous présentons ici trois règles qui régissent les éléments de base vus précédemment.

– Règle 1 : *restriction de type*. Dans un modèle, un type peut être remplacé par un sous-type. Il en résulte un autre modèle ;

– Règle 2 : *sous-typage entre méta-éléments*. Un métanœud (méta-arc) ne peut pas être sous-type d'un méta-arc (métanœud) ;

– Règle 3 : *rapport existentiel entre un élément et son méta-élément*. Un nœud (arc) peut exister si et seulement si son métanœud (méta-arc) est défini auparavant au niveau méta.

5.2. *Autres éléments de M3*

Nous présentons succinctement ci-après quelques uns des autres éléments du méta-métamodèle.

5.2.1. *Contraintes de cardinalités sur méta-arcs*

Le méta-arc *Mcard* sert à spécifier les contraintes de cardinalités sur les sources et destinations des méta-arcs. Afin de simplifier la représentation graphique d'un méta-arc, un méta-arc qui sert à unir un méta-élément source à un méta-élément destination peut être représenté par une flèche, en ligne solide, du méta-élément source au méta-élément destination. Si le méta-arc est représenté d'une manière simplifiée par une flèche en ligne solide, les contraintes (sur la source ou destination) s'écriront sur l'extrémité correspondante (source ou destination) de la flèche. La notation retenue est semblable à celle d'UML pour en faciliter la lecture. Comme l'illustre la figure 4, une instance de type *X* peut être associée par des arcs de type *R* avec de *c* à *d* instances de type *Y*, et une instance de type *Y* peut être associée par des arcs de type *R* avec de *a* à *b* instances de type *X*.

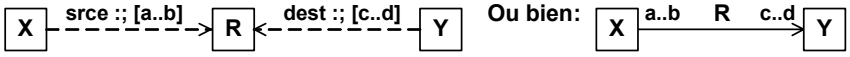

Figure 4. *Notations graphiques d'un méta-arc*

5.2.2. *Nommage et interprétations d'éléments*

Les éléments *MLabel* et *Mname* permettent le nommage d'éléments. Un libellé (*MLabel*) peut nommer (*Mname*) au maximum un *MElement*, et un *MElement* peut être nommé par au maximum un *MLabel*. *MLanguage* et *MValue* sont utilisés pour représenter respectivement les langues et les valeurs de l'univers du discours qui servent à représenter les interprétations des éléments nommés. Ces langues et valeurs sont spécifiées et attachées par les arcs de type *Mdepend*/*MvalueOf*. Le multilinguisme dans les interprétations d'éléments est permis. Tel qu'illustré par la figure 5, l'élément *Object* est interprété en anglais comme « *Object* » et en français comme « *Objet* ».

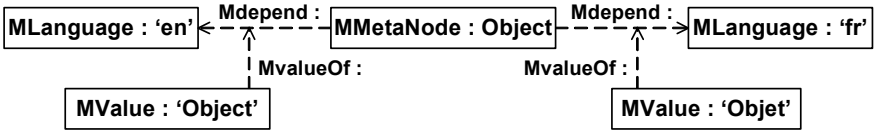

Figure 5. *Interprétation d'éléments*

5.2.3. *Représentation des modèles et de leurs liens*

MModel, MMetaModel, MStructure, MIfThenModel, Msem, MdefAs. *MModel* permet la contextualisation des éléments au niveau méta. Parmi les modèles (*MModel*) aux niveaux M2 et M3, nous distinguons les métamodèles

(*MMetaModel*), les structures (*MStructure*), et les modèles de conditions (*MIfThenModel*).

MMetaModel représente l'ensemble des métamodèles. Un modèle se conforme (*Msem*) à un et un seul métamodèle. Les éléments dans un modèle ont leurs méta-éléments spécifiés dans un métamodèle auquel ce modèle est conforme.

MStructure a pour but de contextualiser les structures des méta-arcs. Le méta-arc *MdefAs* permet de représenter les *liens de définition* entre les méta-arcs et leurs structures initiales. Un méta-arc peut être défini par une ou plusieurs structures (*MStructure*) dites *structures initiales* et une structure représente la définition d'un seul méta-arc. La figure 6 illustre la définition d'un méta-arc. Elle décrit la définition du type d'instanciation (*instOf*) entre les objets (*Objet*) et les types statiques d'objets (*ObjStaticType*). Le méta-arc *instOf* se conforme à *MMetaArc* et est rattaché à sa structure initiale *Structure-instOf* par le lien *MdefAs*.

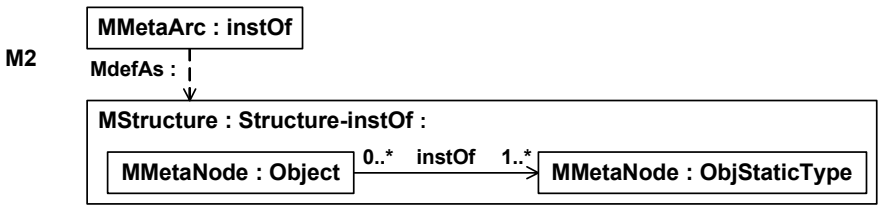

Figure 6. *Définition d'un méta-arc de M2*

L'élément *MIfThenModel* sert à contextualiser les préconditions et postconditions dans la représentation des règles sémantiques des métamodèles. Il sera détaillé en section 5.2.4.

Nous présentons maintenant les autres méta-arcs permettant de gérer les modèles de M3 et M2 ainsi que leur contenu.

MdefIn, Mcontain, Mextend. Les éléments d'un modèle sont définis (*MdefIn*) dans un et un seul modèle et un modèle peut contenir (*Mcontain*) plusieurs éléments. Un modèle peut étendre (*Mextend*) d'autres modèles afin de réutiliser les éléments de ces derniers.

Mrestrict, Minfer, Mjoin, Mdiff, Mintersection, Mresult. Ces méta-arcs visent à représenter, d'une manière explicite, au niveau méta, des opérations entre modèles telles que celles mentionnées dans le Besoin 7 – « Liens avec les modèles ».

Le méta-arc *Minfer* permet de spécifier qu'un modèle infère un autre modèle. *Mrestrict* permet de spécifier qu'un modèle est plus restrictif qu'un autre modèle. *Mjoin*, *Mdiff* et *Mintersection* représentent respectivement les opérations de *jointure*, de *différence*, et d'*intersection* entre modèles et *Mresult* permet d'indiquer qu'un

modèle est résultat de ces opérations. La figure 7 spécifie que *Modèle3* est le modèle résultant de l'union des deux modèles *Modèle1* et *Modèle2*.

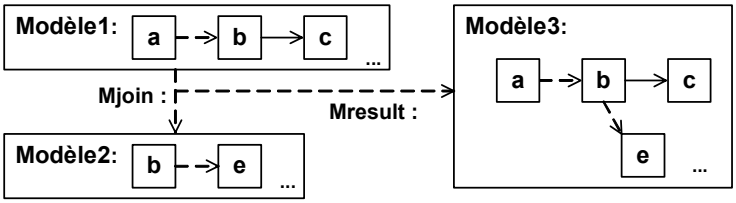

Figure 7. *Exemple de jointure entre modèles au niveau M2/M3*

5.2.4. *Représentation de règles*

***MRule*, *MIfThenModel*, *MRef*, *MEveryRef*, *Mif*, *Mthen*.** Ces types permettent la structuration des règles sémantiques de métamodèles.

Une règle (*MRule*) est attachée à sa partie de préconditions (*MIfThenModel*) et à sa partie de postconditions (*MIfThenModel*) respectivement par des arcs de type *Mif* et respectivement *Mthen*.

Pour le traitement des variables dans les règles, nous avons défini les méta-éléments *MRef* et *MEveryRef*. *MRef* correspond à l'interprétation du quantificateur existentiel alors que *MEveryRef* correspond à celle du quantificateur universel.

La figure 8 montre un exemple de représentation de règles. Afin de spécifier que pour tout métanœud, tous ses sous-types ne sont que des métanœuds, nous avons établi la règle suivante si (i) *x* est un métanœud, et *y* est un sous-type de *x*, alors (ii) *y* est un métanœud. Cette règle peut être modélisée comme suit. L'élément *RègleSousTypeDeMetaNode* (représentant la règle) est associé par un arc de type *Mif* à *RègleSousTypeDeMetaNode-ModèleIf* et par un arc de type *Mthen* à *RègleSousTypeDeMetaNode-ModèleThen*. Le modèle *RègleSousTypeDeMetaNode-ModèleIf* exprime l'expression (i) comme suit. L'élément *x* représente une variable désignant n'importe quel métanœud. *x* est déclaré comme conforme à *EveryRef*, ce qui est indiqué par un arc de type *Mmeta* de *x* à *EveryRef*. Le fait que *x* désigne un métanœud est indiqué par un arc de type *Mmeta* de *x* à *MMetaNode*. L'élément *y* représente une variable désignant n'importe quel sous-type de *x*. *y* est donc déclaré comme conforme à *EveryRef*, ce qui est indiqué par un arc de type *Mmeta* de *y* à *EveryRef*. Le fait que *y* désigne un sous-type de *x* est indiqué par un arc de type *MsubType* de *y* à *x*. Le modèle *RègleSousTypeDeMetaNode-ModèleThen* exprime l'expression (ii). Le fait que *y* désigne un métanœud est indiqué par un arc de type *Mmeta* de *y* à *MMetaNode*.

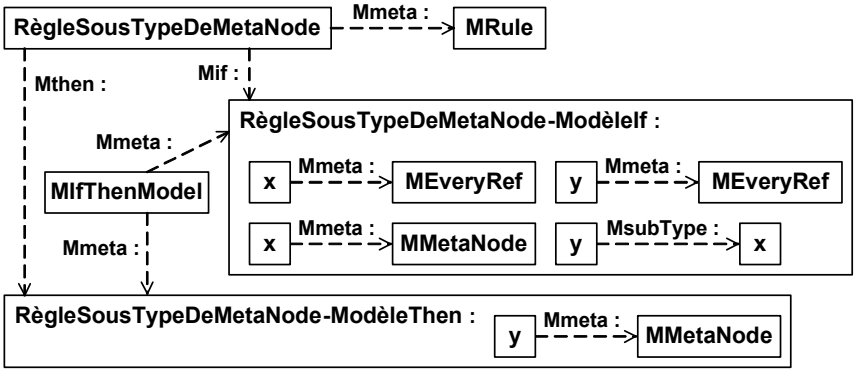

Figure 8. *Exemple de règles*

Mfulfil, Mwhere, MassignedTo. *Mfulfil* s'utilise pour indiquer les règles que les méta-éléments doivent vérifier. A l'aide du méta-arc *Mwhere*, il est possible de représenter les cas où une règle peut être personnalisée en remplaçant d'abord, une ou des variables dans cette règle par des éléments concrets, puis être appliquée à une situation.

Nous présentons ici un exemple d'utilisation. L'élément *MNode* doit vérifier la règle spécifiant que tous les sous-types de *MNode* sont des métanœuds. Remplacer *x* par *MNode*, la règle *RègleSousTypeDeMetaNode* (cf. figure 8) exprime que tous les sous-types de *MNode* sont des métanœuds. Donc, tel qu'illustré par la figure 9, *MNode* est associé par un arc de type *Mfulfil* à la règle *RègleSousTypeDeMetaNode* où (ce qui est indiqué par un arc de type *Mwhere*) *MNode* est attribué à la variable *x* (ce qui est indiqué par un arc de type *MassignedTo*).

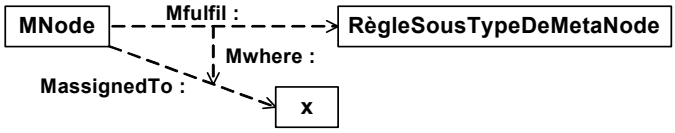

Figure 9. *Exemple de Mfulfil, MassignedTo, Mwhere*

5.3. *Réflexivité de notre méta-métamodèle*

Les éléments de base de notre M3 permettent de définir les autres éléments du niveau M3 ainsi que les éléments du niveau M2 et forment donc le cœur réflexif de notre méta-métamodèle. Nous présentons ci-après comment ces éléments de base se définissent eux-mêmes puis nous montrons comment notre méta-métamodèle se définit lui-même.

Les liens de conformité (*Mmeta*) entre les éléments de base de M3 sont illustrés dans la figure 10. *MNode* et ses sous-types (y compris *MMetaElement*, *MMetaNode*, *MMetaArc*) se conforment à *MMetaNode*. *MArc* et ses sous-types (y compris *Msrce*, *Mdest*, *Mmeta*, *MsubType*) se conforment à *MMetaArc*. Le méta-élément *MElement* n'est ni un métanœud ni un méta-arc et se conforme donc à *MMetaElement*.

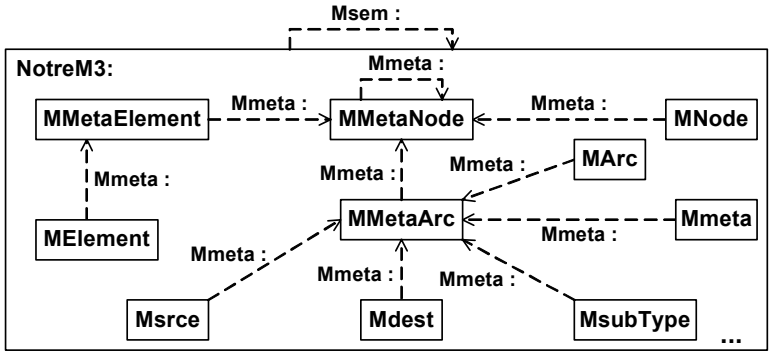

Figure 10. *Les éléments de base – le cœur de notre méta-métamodèle*

Comme les arcs de type *Mmeta* associent les éléments au niveau méta à leurs méta-éléments, le méta-arc *Mmeta* relie *MElement* à *MMetaElement* (figure 11a). Puisqu'un nœud ou un arc doit être conforme respectivement à un métanœud ou à un méta-arc, un *MNode* ou un *MArc* doit être associé respectivement à un *MMetaNode* ou à un *MMetaArc* par un arc de type *Mmeta*. Ceci peut être spécifié par les structures plus restrictives de *Mmeta* (cf. figures 11b et 11c) qui sont déduites de sa structure initiale (cf. figure 11a).

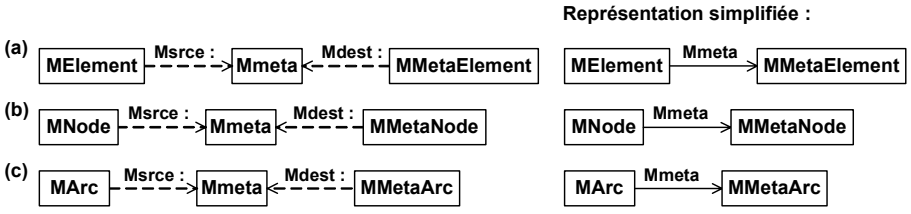

Figure 11. *Structures de Mmeta (sans contraintes de cardinalités) – (a) structure initiale ; – (b), (c) structures plus restrictives*

Etant donné qu'un méta-arc unit deux méta-éléments *via* un arc de type *Msrce* et un arc de type *Mdest*, le méta-arc *MArc* (rectangle), super-type de tous les méta-arcs de M3, doit lier le méta-élément *MElement* à *MElement* lui-même *via* un arc de type *Msrce* et un arc de type *Mdest* (flèches pointillées). Tel qu'illustré par la figure 12,

suivant la règle d'existence entre un arc et son méta-arc (cf. Règle 3), l'arc de type *Msrce* ou *Mdest* de *MElement* à *MArc* peut exister si et seulement si le méta-arc *Msrce* ou *Mdest* est prédéfini pour relier *MMetaElement* à *MMetaArc*. Ces structures de *Msrce*/*Mdest* spécifient la définition de tous les méta-arcs : un *MMetaArc* unit deux *MMetaElement via* un arc de type *Msrce* et un arc de type *Mdest*. Cette définition permet d'expliquer aussi comment la direction d'un arc/méta-arc est établie. Tous les éléments conformes à *MMetaArc* se conforment à cette définition.

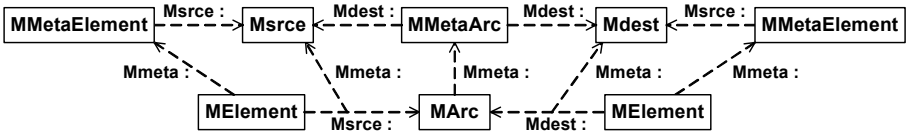

Figure 12. *Structures initiales de MArc, Msrce, Mdest (sans contraintes de cardinalités)*

Afin que les liens de sous-typage (*MsubType*) puissent exister entre les méta-éléments, le méta-arc *MsubType* doit être défini pour lier l'élément *MMetaElement* à *MMetaElement* lui-même (figure 13). Cette structure de *MsubType* autorise l'existence d'arcs de type *MsubType* entre les *MMetaElement*, y compris les *MMetaNode* et les *MMetaArc*. Un *MMetaNode* (respectivement un *MMetaArc*) ne peut cependant pas être associé à un *MMetaArc* (respectivement un *MMetaNode*) par un arc de type *MsubType* (cf. Règle 2).

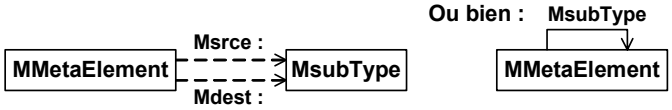

Figure 13. *Structure initiale de MsubType (sans contraintes de cardinalités)*

Chacun des éléments de base de M3 est défini en se basant sur les éléments de M3, conformément à leur sémantique et définition. Ceci démontre que le noyau constitué de ces éléments de base est réflexif.

6. Ingénierie de modèles

Nous montrons dans cette section comment le formalisme que nous avons présenté ci-avant s'intègre dans une démarche de gestion de connaissances et d'ingénierie des modèles.

6.1. *Représentation de métamodèles de types différents*

Le méta-métamodèle (M3) permet de décrire des « frameworks » de modélisation complexes comme celui présenté dans la figure 14. Dans cette figure, la transformation d'un modèle source (*Modèle1*) en un modèle destination (*Modèle2*) respecte les règles de transformation entre les éléments du métamodèle source (*Métamodèle1*) et les éléments du métamodèle destination (*Métamodèle2*) correspondants. Nous pouvons considérer que *Modèle1* est associé à *Modèle2* par un lien « transformation », et celui-ci est associé à une ou des règles de transformation par des liens « applique les règles ». Le métamodèle *Métamodèle2* étend le métamodèle *MétamodèleA* (ce qui est indiqué par l'arc de type *Mextend*) pour réutiliser les éléments de ce dernier.

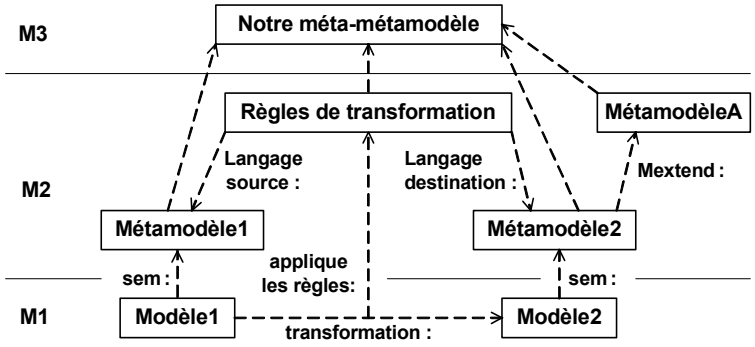

Figure 14. *Modèles de différents formalismes et liens de transformation*

M3 permet de représenter des métamodèles de types différents au niveau M2 pour représenter le monde réel. Pour illustrer cette capacité de représentation de métamodèles de types différents au niveau M2, nous présentons comment notre M3 met en œuvre les familles de formalismes suivantes : métamodèles sNets, formalismes à objets et métamodèles RDF/RDFS/OWL.

6.1.1. *sNets*

M3 permet d'implémenter tous les métamodèles de la famille des métamodèles sNets. Dans cette famille, toute méta-entité, toute métarelation, tout univers sémantique se conforment respectivement à nos éléments *MMetaNode*, *MMetaArc*, et *MMetaModel*. La figure 15 montre un exemple de la représentation de *Marie* dans sNets dont on a représenté les éléments nécessaires au niveau M2.

Dans cette figure, *Marie*, définie dans le modèle *UnModèleM1sNets* au niveau M1, est conforme à *sNetsObject* de M2. *Marie* est également une instance de *Personne* de M1 (ce qui est spécifié par un lien d'instanciation *sNetstype* sur l'axe horizontal). L'élément *Personne* est conforme à *sNetsClass* de M2. Dans la famille

des métamodèles sNets, *sNetsObject* et *sNetsClass* représentent respectivement l'ensemble de tous les objets et celui de tous les types d'objets ; ils sont donc conformes à *MMetaNode*. La métarelation *sNetstype* entre *sNetsObject* et *sNetsClass* est conforme à *MMetaArc*. Représentant un univers sémantique dans sNets, *M2sNet* est conforme à *MMetaModel* (ce qui est indiqué par un arc de type *Mmeta*), et le contenu de *M2sNet* est conforme au contenu de *NotreM3*, c'est-à-dire à notre M3 (ce qui est indiqué par un arc de type *Msem*).

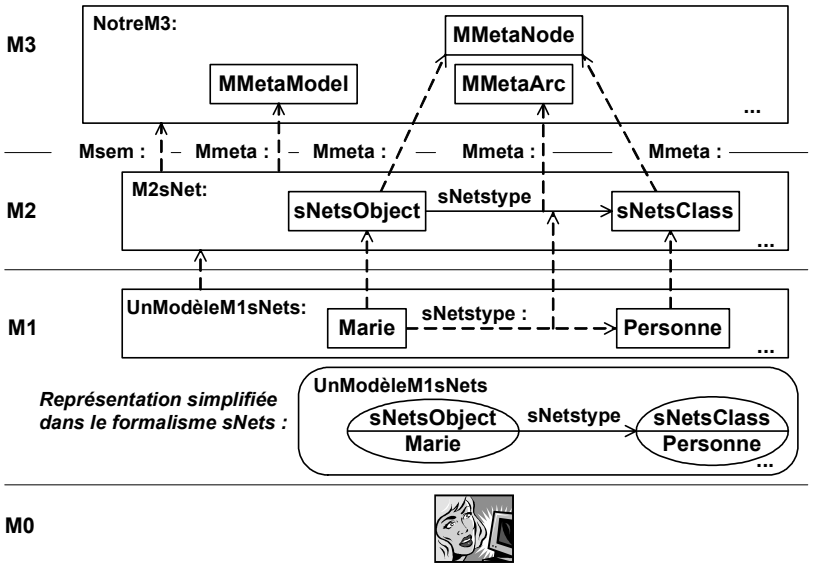

Figure 15. *Marie et sNets*

6.1.2. *Les formalismes à objets*

M3 permet aussi d'implémenter les formalismes à objets, par exemple UML. Dans le métamodèle UML, toute métaclasse, toute association entre métaclasses, et tout package sont conformes respectivement à nos éléments *MMetaNode*, *MMetaArc*, et *MMetaModel*. La figure 16 décrit partiellement le diagramme de classes du package *Core::Constructs* du métamodèle UML (OMG, 2005a). Dans ce diagramme, les métaclasses *Classifier*, *Class*, *Relationship*, *Association*, *Property*, *Operation* sont conformes à *MMetaNode*, et elles représentent respectivement l'ensemble des classificateurs, des classes, des types relationnels, des associations, des attributs, et des opérations au niveau M1. *Association* est un sous-type de *Classifier* et de *Relationship* ; *Class* est un sous-type de *Classifier* (ce qui est indiqué par des arcs de type *MsubType*). Le méta-arc *memberEnd* est une association entre *Association* et *Property*, spécifiant qu'une *Association* détient au moins deux *Property* comme ses extrémités (*AssociationEnd*) et qu'un *Property* peut être une

extrémité d'une seule *Association*. Le méta-arc *attribute* est une association entre *Classifier* et *Property*, spécifiant qu'un *Classifier* peut prendre plusieurs *Property* pour ses attributs et qu'un *Property* peut être un attribut d'au plus un *Classifier*. Le méta-arc *ownedAttribute* est une association entre *Class* et *Property*, spécifiant qu'une *Class* peut posséder plusieurs *Property* comme ses attributs propres et qu'un *Property* peut être un attribut d'au plus une *Class*. Le méta-arc *ownedOperation* est une association entre *Class* et *Operation*, indiquant qu'une *Class* peut posséder plusieurs *Operation* comme ses opérations/méthodes et qu'une *Operation* peut être une opération/méthode d'au plus une *Class*.

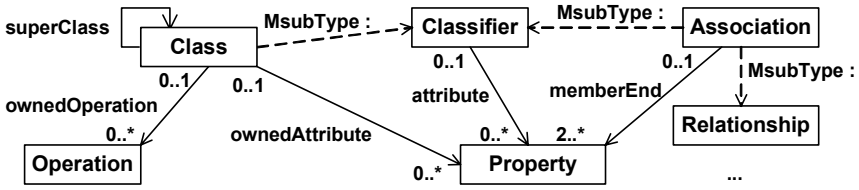

Figure 16. *Diagramme de classes selon le package Core::Constructs de UML2.0*

La figure 17 illustre la représentation de *Marie* dans UML.

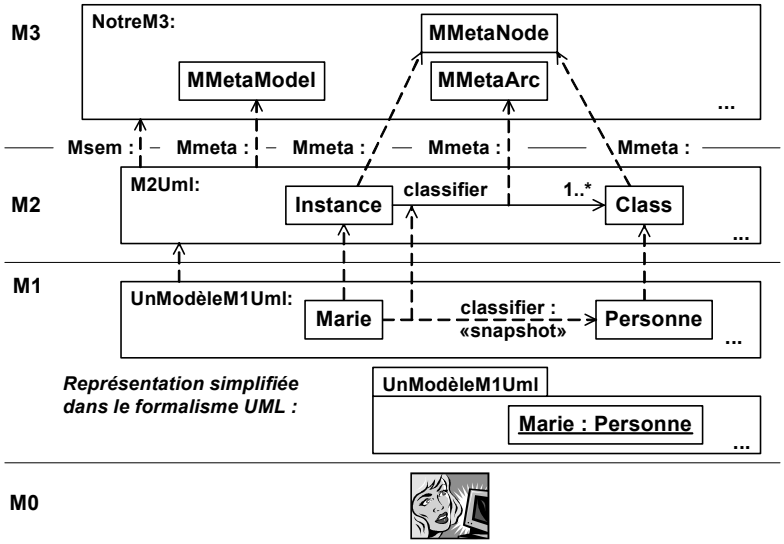

Figure 17. *Marie et UML*

Dans cette figure, *Marie*, définie dans le modèle *UnModèleUml* au niveau M1, est conforme à *Instance* de M2. Elle est aussi une instance de *Personne* de M1

(ce qui est spécifié par un lien d'instanciation (« snapshot ») de type *classifier* sur l'axe horizontal). *Personne* est conforme à *Class*. *Instance* et *Class* représentent respectivement l'ensemble des instances et des classes ; ils sont donc conformes à *MMetaNode*. *classifier* (associant *Instance* à *Class*) est conforme à *MMetaArc*.

6.1.3. *RDF/RDFS/OWL*

Enfin M3 permet également de représenter les formalismes utilisés dans le contexte du web sémantique comme RDF/RDFS/OWL. Dans la famille de métamodèle RDF/RDFS/OWL, les éléments conformes à notre élément *MMetaNode* sont les métaclasses de ressources, exemples : *xs:schema* / *rdf:RDF* / *owl:Ontology* ; *xs:element* ; *rdfs:Class* / *owl:Classe* ; *rdf:resource* / *owl:Thing* ; *rdf:Property* / *owl:ObjectProperty* / *owl:DatatypeProperty* ; etc. Les éléments conformes à notre élément *MMetaArc* sont les méta-propriétés, exemples : *rdf:type* ; *rdfs:subClassOf* / *owl:subClassOf* ; *rdfs:subPropertyOf* ; *rdfs:domain* ; *rdfs:range* ; *owl:onProperty* ; *owl:hasValue* ; *owl:imports* ; *owl:priorVersion* ; etc. Egalement, les éléments conformes à notre élément *MMetaModel* sont les schémas, les espaces de noms, les ontologies qui contiennent la définition des vocabulaires XML/RDF/RDFS/OWL, exemples : *xmlns:xs* ; *xmlns:rdfs* ; *xmlns:rdf* ; *xmlns:owl*.

La représentation de *Marie* dans RDF/RDFS/OWL est illustrée par la figure 18.

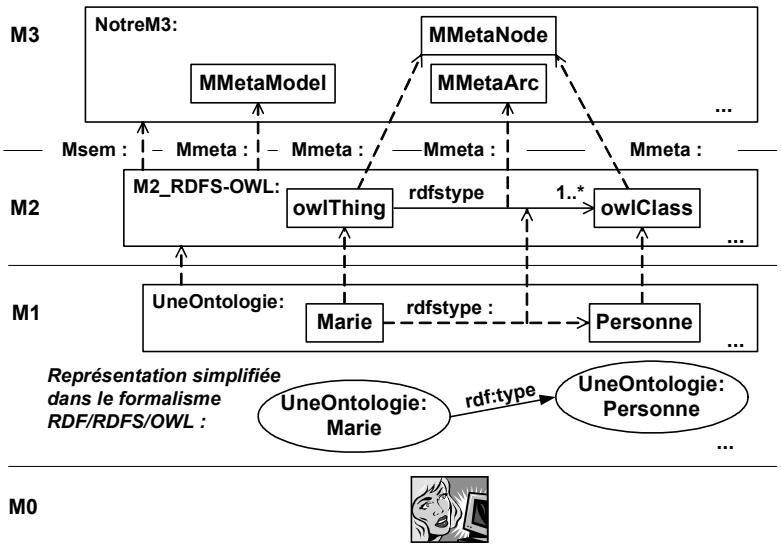

Figure 18. *Marie et RDF/RDFS/OWL*

Dans cette figure, *Marie*, définie dans une ontologie *UneOntologie* au niveau M1, est conforme à *owlThing* de M2, et est aussi une instance de *Personne* de M2

(ce qui est spécifié par un lien d'instanciation de type *rdfstype* sur l'axe horizontal). *Personne* est conforme à *owlClass*. *owlThing* et *owlClass* représentent respectivement l'ensemble des individus et celui des classes ; ils sont donc conformes à *MMetaNode*. *rdfstype* (associant *owlThing* à *owlClass*) est conforme à *MMetaArc*.

7. Autres formalismes de modélisation

Pourquoi développer un nouveau formalisme alors qu'il existe déjà de nombreux formalismes de modélisation ? Car, malheureusement, ces formalismes ne permettent pas de modéliser toutes les situations que l'on peut rencontrer en gestion de connaissances. En effet, considérons le problème ci-après.

Problème : comment peut-on modéliser la situation suivante ? Au niveau M1, *travailler* représente un type de relations qui implique un seul type d'objets *Personne* en tant qu'acteur de l'action travailler. Il peut exister une structure de *travailler* avec les contraintes suivantes. (i) Il y a de une à 7 personnes (*Personne*) au maximum pouvant travailler ensemble, c'est-à-dire celles-ci peuvent s'associer ensemble à une relation de type *travailler*. L'arité de *travailler* est donc variable. (ii) Une personne peut ne participer à aucune, ou participer à une ou plusieurs relations de type *travailler*. (iii) On peut trouver au maximum vingt personnes pouvant chacune s'associer à une relation de type *travailler* avec un même groupe de personnes. Et (iv) les mêmes personnes peuvent participer ensemble à une relation de type *travailler* plusieurs fois.

Aucun des formalismes que nous avons étudiés ne prend en compte toutes les contraintes illustrées dans ce problème. Pour remplir ces besoins de représentation, nous avons défini en complément au méta-métamodèle (M3) présenté ici un métamodèle pour la gestion de connaissances. Faute de place, nous ne détaillerons pas ce métamodèle ni les besoins essentiels concernant la représentation de modèles de connaissances, mais le lecteur intéressé en trouvera la spécification complète dans (Dinh, 2007). Nous présentons maintenant un récapitulatif des formalismes étudiés et leurs possibilités.

Nous avons étudié les formalismes suivants : les réseaux sémantiques (Sowa *et al.,* 1991), le modèle uniforme des GCs (Gerbé, 2000), sNets (Lemesle, 2000) basé sur les réseaux sémantiques ; CDIF (Flatscher, 2002) ; MOF (OMG, 2003b ; OMG, 2006) ; Eclipse Eclipse Modeling Framework (EMF, 2007 ; Bézivin *et al.,* 2005b ; Smolik, 2006 ; Karlsch, 2007) ; Microsoft Domain-Specific Language (MS/DSL) (ATLAS group, 2005 ; Bézivin *et al.,* 2005b ; Smolik, 2006 ; MSDN, 2007) ; XML/XML Schema, RDF/RDF Schema et OWL[1]. Excepté le méta-métamodèle Ecore de EMF, aucun des noyaux ne supporte complètement la notion de *lien*

1. Pour XML-XML Schema, RDF-RDF Schema et OWL, notre analyse est basée sur les documents de spécifications datés de 2004, 2005, 2006 par W3C (http://www.w3.org/).

binaire (cf. Besoin 1). Ces noyaux permettent de représenter un *lien* entre *nœuds* mais non entre *liens* ni entre un *lien* et un *nœud*, et donc ne permettent pas de représenter les liens de type *meta* comme illustrés dans la figure 1, ni les liens de jointure entre modèles comme illustrés dans la figure 7, ni les liens « transformation » et « applique les règles » comme illustrés dans la figure 14. Concernant la représentation de modèles et de leurs liens, excepté le modèle uniforme des GCs, les formalismes ne tiennent pas compte des notions de modèles structurels et de modèles de conditions (cf. Besoin 6). Les formalismes étudiés ne permettent pas de représenter différents types de liens entre modèles comme la jointure, l'intersection, la différence, l'inférence, la restriction (cf. Besoin 7). Le tableau 1 présente une synthèse de la comparaison des noyaux réflexifs listés en réponse aux besoins essentiels pour M3.

Noyaux réflexifs (au M3)	**Besoins pour M3 (*Besoin*)**						
	1	2	3	4	5	6	7
Notre M3	+	+	+	+	+	+	+
Réseaux sémantiques (RSs)	-	-	-	-	-	-	-
sNets	+-	+	+	+-	+-	+-	+-
Graphes conceptuels (GCs)	+-	+	-	+	-	+-	+-
Modèle uniforme des GCs	+-	+	-	+	+	+-	+-
CDIF	+-	+	-	+	+	-	+-
MOF	+-	+	-	+	+	-	+-
Méta-métamodèle Ecore (de Eclipse EMF)	+-	+	(*)	+	+	-	+-
Méta-métamodèle de MS/DSL	+	+	(*)	+	+	-	+-
XML – XML Schema	+-	+	-	+	-	+-	+-
RDF – RDF Schema	+-	+	-	+	-	+-	+-
OWL	+-	+	-	+	+	+-	+-

Notations :

«+» : remplir le besoin correspondant;

«+-» : remplir partiellement et à développer pour remplir le besoin correspondant;

«-» : ne pas remplir et à développer pour remplir le besoin correspondant

(*) : les spécifications disponibles ne nous permettent pas de documenter ce point

Tableau 1. *Synthèse des formalismes à représenter et gérer les métamodèles*

8. Conclusion et travaux futurs

La gestion de modèles prend une importance grandissante et suscite de nombreux travaux de recherche dans divers domaines comme la gestion des connaissances, la gestion de métadonnées, les ontologies, la qualité de service et le génie logiciel. La représentation de modèles y est fondamentale. Nous avons identifié les besoins essentiels concernant la représentation des métamodèles et la représentation de modèles de connaissances du monde réel. Ces besoins nous permettent de mieux comprendre la complexité du monde à modéliser, d'identifier plus facilement les éléments pour chaque niveau de modélisation, et d'aboutir à une représentation la plus précise mais aussi la plus souple possible. Dans cet article, nous avons présenté en détail les besoins pour un M3. Ces besoins nous ont permis d'identifier les éléments essentiels pour un M3. Comme les formalismes de modélisation couramment utilisés que nous avons étudiés ne rencontraient pas toutes les exigences pour la représentation de modèles, nous avons défini notre propre formalisme. Ce dernier est un noyau réflexif composé d'un nombre minimal d'éléments qui permettent de définir tous les éléments de niveau méta-métamodèle et aussi ceux de niveau métamodèle. Ce formalisme est basé sur les réseaux sémantiques pour faciliter sa mise en œuvre. Il est à souligner que notre formalisme permet l'extensibilité. Le méta-métamodèle proposé est extensible car il offre la possibilité de définir et d'ajouter de nouveaux éléments au besoin, en se basant sur le noyau réflexif. Nos travaux portent actuellement sur la réalisation d'un outil de validation pour ce formalisme avec des opérations de manipulation de modèles. Quoique nos études de cas se situent plutôt dans la gestion des connaissances, nos résultats de recherche sont certainement applicables à d'autres domaines comme dans celui de qualité de service où le besoin de représentation et de gestion de modèles se présente.

9. Bibliographie

Alagic S., Bernstein P.A., "A Model Theory for Generic Schema Management", *Proc. DBPL 2001*, Italy, vol. 2397/2002, Springer-Verlag Heidelberg, 2001, p. 228-246.

ATLAS group, LINA & INRIA Nantes, ATL Transformation Examples – The Microsoft DSL to EMF, ATL transformation, version 0.1, October 2005.

Bernstein P.A., Levy A.Y., Pottinger R.A., "A Vision for Management of Complex Models", *SIGMOD*, Record 29, (4), 2000, p. 55-63.

Berners-Lee T., Hendler J., Lassila O., *The Semantic Web*, Scientific American, May 2001.

Bézivin J., Blay M., Bouzeghoub M., Estublier J., Favre J.-M., Action spécifique CNRS sur l'Ingénierie Dirigée par les Modèles, Rapport de synthèse, janvier, 2005a.

Bézivin J., Gerbé O., "Towards a Precise Definition of the OMG/MDA framework", *Proc. of the 16th International Conference on Automated Software Engineering*, 2001.

Bézivin J., Hillairet G., Jouault F., Kurtev I., Piers W., "Bridging the MS/DSL Tools and the Eclipse Modeling Framework", *Proceedings of the International Workshop on Software Factories at OOPSLA'05*, San Diego, California, USA, 2005b.

Chen P., "The Entity-Relationship Model – Towards a Unified View of Data", *ACM Transactions on Database systems*, vol. 1, n° 1, 1976, p. 9-36.

Corcho O., Fernández-López M., Pérez A.G., OntoWeb. D1.1, Technical Roadmap, 2001.

Dahchour M., Integrating Generic Relationships into Object Models Using Metaclasses, PhD Thesis, Université catholique de Louvain, Belgium, Mar. 2001.

Dinh T.-L.-A., Gerbé O., "A Metamodel for Knowledge Management", *RIVF'04 – Int. Conf. of French-Speaking or Vietnamese Computer Scientists*, Vietnam, 2004, p. 107-112.

Dinh L.-A., Gerbé O., Houari S., « Gestion de modèles : définitions, besoins et revue de littérature », *Actes des premières journées sur l'Ingénierie Dirigée par les Modèles (IDM05)*, Paris, France, Juin-Juillet 2005, p. 1-15.

Dinh L.-A., Gerbé O., Houari S., « Un méta-métamodèle pour la gestion de modèles », *Deuxièmes journées sur l'Ingénierie Dirigée par les Modèles (IDM'06)*, France, 2006.

Dinh T.-L.-A., Modélisation pour la gestion de modèles, Thèse de doctorat, Université de Montréal, 2007.

EMF., Eclipse Modeling Framework Project, www.eclipse.org/emf/, 2007.

Flatscher R.G., "Metamodeling in EIA/CDIF – Meta-Metamodel and Metamodels", *ACM Trans. on Modeling and Computer Simulation (TOMACS)*, vol. 12, n° 4, 2002, p. 322-342.

Gerbé O., Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise, Thèse de doctorat, Université de Montréal, 2000.

Gerbé O., Kerhervé B., Srinivasan U., "Model Operations for Quality-Driven Multimedia Delivery", *Contributions to ICCS 2003*, 2003.

Gerbé O., Mineau G., Keller R., La métamodélisation et les graphes conceptuels, Cahier du GReSI no 03-01, HEC Montréal, 2003.

Girard S., Favre J.-M., Muller P.-A., Blanc X. (eds), *Actes des premières journées sur l'Ingénierie Dirigée par les Modèles (IDM05)*, Paris, France, Juin-Juillet 2005.

ISO., ISO-IEC 10027: Information technology – Information Resource Dictionary System (IRDS) – Framework, ISO/IEC International standard, 1990.

Jézéquel J.-M., Gérard S., Mraidha C., Baudry B., Approche unificatrice par les modèles, Action spécifique CNRS sur l'Ingénierie Dirigée par les Modèles, janvier 2005.

Karlsch K., A model-driven framework for domain specific languages demonstrated on a test automation language, Master Thesis, Hasso-Plattner-Institute of Software Systems Engineering Potsdam, Germany, 2007.

Kerhervé B., Gerbé O., "Model Management for Quality of Service Support", *Proc. of the 14th Int. Conf. on Soft. & Syst. Engineering and their Applications*, vol. 1, France, 2001.

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

Melnik S., Generic Model Management, Ph. D Thesis, Lecture Notes in Computer Science, Springer, 2004.

MSDN, Domain-Specific Language Tools, http://msdn2.microsoft.com/fr-ca /library/ bb126235(*vs*.80).aspx, 2007.

Mylopoulos J., Borgida A., Jarke M., Koubarakis M., "Telos: Representing knowledge about information systems", *ACM Trans. on Inform. Systems*, vol. 8, n° 4, 1990, p. 325-362.

OMG, MDA Guide Version 1.0.1. Joaquin Miller & Jishnu Mukerji (ed.), 2003a.

OMG, Meta Object Facility (MOF) 2.0 Core Specification, ptc/03-10-04, 2003b.

OMG, Meta Object Facility (MOF) 2.0 Core Specification, formal/06-01-01, 2006.

OMG., UML 2.0 Infrastructure Specification, formal/05-07-05, 2005.

OMG., UML 2.1.1 Infrastructure Specification, formal/07-02-06, 2007.

Rieu D., Bounaas F., Morat P., Tamzalit D., « La méta-circularité au service de la méta-modélisation », *Inforsid'97*, Toulouse, 1997, p. 577-599.

Sahraoui H.A, Application de la méta-modélisation à la génération d'outils de conception et de mise en œuvre de bases de données, Thèse de doctorat, Université P. et M. Curie (Paris 6), Paris, France, 1995.

Smolik P. C., Mambo Metamodeling Environment, A dissertation submitted in partial fulfillment of the requirements for the degree of doctor of philosophy, Brno University of Technology, Czech Republic, 2006.

Sowa J.F., *Conceptual Structures – Information processing in mind and machine*, Addison Wesley 14472, 1984.

Sowa J.F., Borgida A., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann Publishers, San Mateo, CA, 1991.