Un méta-métamodèle pour la gestion de 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

*** *Escuela Superior de Informática, UCLM*
Paseo de la Universidad, 4.13071 Ciudad Real, España

RÉSUMÉ. *Les nombreux travaux de recherche autour de IDM (Ingénierie Dirigée par les Modèles) montrent que la gestion de modèles intéresse de nombreuses communautés de recherche. En effet, dans les domaines de la gestion des connaissances, la gestion de méta-données, les ontologies, la qualité de service et le génie logiciel, les chercheurs travaillent beaucoup sur la modélisation et la métamodélisation. La représentation de modèles et de métamodèles est donc un axe de recherche essentiel pour la gestion de modèles. Dans ce contexte, nous proposons ici un méta-métamodèle réflexif qui permet de décrire divers modèles et métamodèles.*

ABSTRACT. *Research work in the fields of Model-Driven Engineering (MDE) and Model-Driven Development (MDD) shows that model management attracts special interest from different research communities. The communities of knowledge management, metadata management (databases), ontologies, quality of service, software engineering, etc. are particularly focusing on the definition and the use of models. Therefore, representing models is an essential research axis for model management. In this context, we propose a reflexive meta-metamodel allowing to describe diverse metamodels.*

MOTS-CLÉS : *Modélisation, Méta-modélisation, Modèle, Métamodèle, Méta-métamodèle.*

KEYWORDS: *Modeling, Meta-modeling, Model, Metamodel, Meta-metamodel.*

1. Introduction

La notion de modèles fait l'objet de discussions depuis l'antiquité. Cependant, dans la communauté informatique, l'émergence d'une vision de développement du logiciel appelé *Model Driven Engineering* - Ingénierie Dirigée par les Modèles ou IDM - a relancé l'intérêt en mettant l'accent sur 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. Elle intervient dans des domaines aussi divers que le génie logiciel, les bases de données, la qualité de service et la gestion des connaissances.

En génie logiciel, l'approche Ingénierie Dirigée par les Modèles (Bézivin *et al.*, 2005; 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.

Dans le domaine des bases de données, la gestion des méta-données touche la manipulation de la structure des données plutôt que les données elles-mêmes. L'approche basée sur la gestion de modèles est une solution possible aux problèmes d'intégration et transformation de données (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 des systèmes multimédias répartis, l'intégration de l'information de gestion devient fondamental (Kerhervé *et al.*, 2001). Une solution possible est de développer des modèles génériques pour les systèmes, les applications et les utilisateurs et d'ensuite développer des mécanismes d'intégration (Gerbé *et al.*, 2003a).

Dans le domaine de la gestion des connaissances, l'intérêt sémantique de la modélisation 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 (Dinh *et al.*, 2004). Ceci implique un véritable défi pour la représentation et la modélisation de ces mémoires corporatives.

Dans sa thèse (Schneider, 1994), Daniel K. Schneider discute la notion de modèle scientifique avec une approche sciences sociales qui apporte un éclairage intéressant à ceux qui s'intéresse comme nous à la modélisation dans une perspective gestion de connaissances. Schneider reprend à son compte les trois fonctions d'un modèle développé par Stachowiak (Stachowiak, 1965). Un modèle a une fonction de représentation; un modèle représente un original naturel ou artificiel que l'on peut décrire comme un ensemble d'éléments et leurs interrelations. Un modèle a une fonction de réduction; un modèle ne représente que les caractéristiques pertinentes au but de la modélisation. Enfin, un modèle a une fonction « subjectivisante »; un modèle n'a pas de relation « naturelle » avec son original, l'interprétation d'un modèle tient compte du but et de l'usage.

La représentation de modèles est un axe de recherche essentiel dans la gestion des modèles de connaissances. Selon nos études (Dinh *et al.*, 2005), les formalismes couramment utilisés en modélisation (graphes conceptuels, sNets, CDIF, MOF, OWL) ne permettent pas de satisfaire tous les besoins essentiels pour la représentation et la gestion de modèles dans le cadre de la gestion de connaissances. Le lecteur trouvera à la fin de l'article, après une mise en contexte (section 3 et section 5), le résumé de ces études.

De plus, il est à noter que l'utilisation d'un ensemble de modèles avec différents formalismes pose souvent des problèmes de traduction et de pertes d'information lors des échanges entre modèles bien que ces modèles soient compatibles. Notre objectif est donc de spécifier un nouveau formalisme qui sera susceptible de satisfaire tous les besoins de la gestion de modèles dans le cadre de la gestion de connaissances. Ce formalisme est principalement basé sur les réseaux sémantiques (Sowa *et al.*, 1991) en souhaitant de maintenir

la flexibilité des réseaux sémantiques dans l'expression ainsi que la simplicité dans la représentation pour la facilité de mise en œuvre.

Cet article présente une première ébauche de ce formalisme et traite principalement de la représentation des métamodèles. Les métamodèles sont des modèles particuliers permettant de décrire les modèles eux-mêmes. Le reste de l'article est organisé comme suit. La section 2 décrit l'architecture de modélisation et métamodélisation et les différents niveaux de modélisation. Dans la section 3, les notions de base et les besoins essentiels pour un méta-métamodèle sont présentés. Nous présentons notre méta-métamodèle dans la section 4 et l'évaluons en le comparant avec d'autres formalismes dans la section 5. Finalement, la section 6 conclut et présente nos travaux futurs.

2. Architecture de modélisation

L'architecture de modélisation est basée sur quatre niveaux. Cette architecture est largement acceptée aujourd'hui (Sahraoui, 1995; Revault *et al.*, 1995; Lemesle, 2000; Bézivin *et al.*, 2001; Dinh, 2003; Girard *et al.*, 2005). Nous présentons brièvement ci-dessous chacun des quatre niveaux :

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

– M2 est le niveau métamodèle. 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 respecte la grammaire spécifiée par son métamodèle au niveau M2.

– M0 est le monde réel décrit au niveau M1.

Dans cette architecture, seuls les trois premiers niveaux (M3, M2, M1) appartiennent au monde de la modélisation, le niveau M0 est le monde réel. Ce que

l'on appelle couramment les types et les instances sont au même niveau M1 (Lemesle, 2000; Bézivin *et al.*, 2001). Afin d'éviter des problèmes dus à la représentation de la nature des concepts (y compris des modèles), par exemple le problème de l'instanciation double discuté dans (Bézivin *et al.*, 2001; Dinh *et al.*, 2004), une architecture de modélisation devrait bien distinguer la notion d'instanciation, la notion de conformité entre éléments et la notion de conformité entre modèles. Ces notions sont en effet de nature très différente, chacune possède son propre comportement vis-à-vis des contextes (global ou local) dans lesquels elle se situe. L'instanciation indique le rapport «types-instances» entre éléments du niveau M1. La conformité entre éléments indique le rapport de conformité entre un élément et son méta-élément. Enfin, la conformité sémantique entre modèles indique le rapport de conformité sémantique entre un modèle et son métamodèle. La Figure 1 illustre un exemple de ces différentes notions. Dans cette figure, *Marie* au niveau M1, représente la vraie personne *Marie* (au niveau M0). *Marie* au niveau M1 est d'une part conforme à *Object* dans le contexte global (ce qui est spécifié par la relation de type $meta_{M2}$ sur l'axe vertical), et d'autre part une instance de *Personne* dans le contexte local (ce qui est spécifié par la relation de type *instOf* sur l'axe horizontal). La Figure 1 montre aussi sur l'axe vertical par la relation de type *sem* que M3 est conforme sémantiquement à lui-même et que M2 est conforme sémantiquement à M3 et par la relation de type sem_{M2} que M1 est conforme sémantiquement au métamodèle M2. Notons que la réflexivité du niveau le plus haut (M3 dans ce cas) possède des avantages multiples. Elle 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.

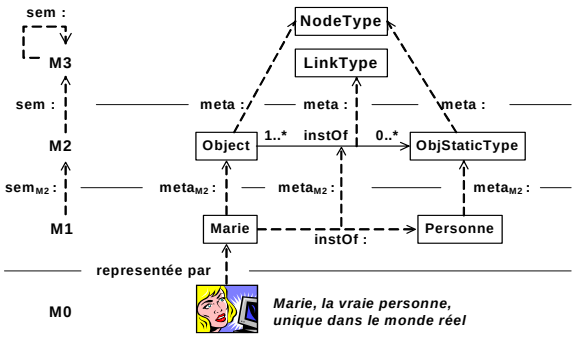

Figure 1. *Notre architecture de modélisation*

3. Besoins pour M3

Cette section recense les besoins essentiels pour un M3. En pratique, les choses à modéliser sont représentées par des éléments et des relations entre eux. Nous considérons ici qu'une relation n-aire peut toujours être décomposée en relations binaires.

Nous détaillons ci-dessous les principaux besoins. Le besoin 1 présente les notions de base. Le besoin 2 traite du passage d'un niveau de modélisation à un autre. Le besoin 3 a pour fin de classer les types non relationnels et les types relationnels. Le besoin 4 représente l'exigence de représentation des contraintes de cardinalités qui fait partie de la représentation sémantique des relations. De plus, pour la représentation et la gestion de modèles, nous avons les besoins 5 et 6. Le besoin 5 exprime la nécessité d'expliciter différents types de modèles. Le besoin 6 exprime le besoin de représentation des interactions possibles entre modèles.

Besoin 1. *Notions de base.* Un *type* représente un ensemble de choses ayant les mêmes propriétés. Une *instance* d'un type représente une chose qui se conforme à la définition du type. Parmi les types, il convient de distinguer les *types relationnels* (associations) et les types non relationnels (classe, entité). Parmi les instances, il convient de distinguer les *liens* – instances des types relationnels – et les occurrences – instances des types non relationnels.

Besoin 2. *« Type » ou « Instance »* (Gerbé, 2000; Dinh *et al.*, 2004). Un élément peut être traité comme « instance » à un niveau mais comme « type » à un niveau inférieur. Par exemple, *ObjStaticType* est vu comme une « instance » de *NodeType* et comme le « type » de *Personne*.

Besoin 3. *Hiérarchisation entre types non relationnels et hiérarchisation entre types relationnels*. Les types relationnels et les typers non relationnels sont organisées en hiérachie.

Besoin 4. *Contraintes de cardinalité maximale/minimale pour les types relationnels*. Ceci capture le nombre des relations possibles par rapport aux éléments impliqués dans une relation.

Besoin 5. *Distinction entre modèles de différentes natures*. Exemples : *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).

Besoin 6. *Relations avec les modèles :* relations de *contenant* entre un modèle et ses éléments; relations *d'accès* entre modèles pour réutiliser dans un modèle des éléments d'un autre modèle; relation de *respect sémantique* entre un modèle et son métamodèle; et d'autres relations comme *l'intersection, la différence, l'inférence, la restriction.*

4. Méta-métamodèle (M3)

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 dans les réseaux sémantiques (Lemesle, 2000; Sowa *et al.*, 1991), la représentation est basée sur les nœuds et les arcs. Un nœud représente une occurrence alors qu'un arc représente un lien. Un nœud est représenté graphiquement par un rectangle alors qu'un arc est représenté graphiquement par un arc orienté, en pointillé.

4.1. *Éléments de base de M3*

Les éléments de base du niveau M3 sont les types : *KO, T, R, Type, NodeType, LinkType* et les relations : *sub, meta, srce* et *dest.* La Figure 2 présente la hiérarchie de généralisation de ces éléments.

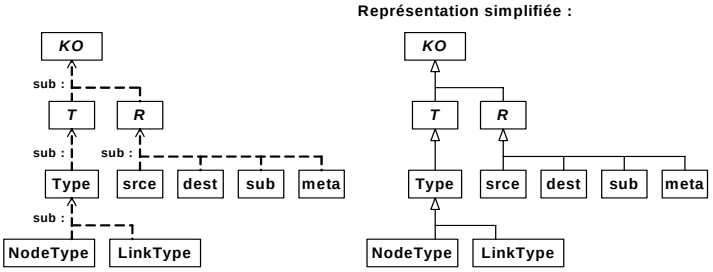

Figure 2. *Hiérarchie des éléments de base de M3*

KO, T, R. *KO* (*Knowledge Object*) est à la racine de la hiérarchie de tous les types définis au niveau M3. Il est l'élément universel et représente l'ensemble des tous les objets. Cet ensemble est subdivisé en deux sous-ensembles: les noeuds (*T*), et les liens (*R*). *KO, T, R* sont des types abstraits, ils n'ont pas directement d'instances et servent à éclaircir sémantiquement la hiérarchie de spécialisation des types.

Type, NodeType, LinkType. *Type* représente l'ensemble de tous les types qui sont subdivisés en deux sous-ensembles : les *NodeType,* et les *LinkType.* Les types du niveau M3 permettent de définir les éléments (y compris les types) du niveau M2. Un *NodeType* représente un type non relationnel, un type de noeud dit *méta-nœud.* Un *LinkType* représente un type relationnel, un type de realtion binaire et directionnelle dit *méta-lien* entre deux types. Un *LinkType* est défini par un *Type* source et un *Type* destination (Figure 3).

srce, dest. *srce* et *dest* permettent de spécifier les sources et destinations de *LinkType.* Comme le montre la Figure 3, le méta-lien *srce* (rectangle) lie *Type* et *LinkType.* Le méta-lien *srce* a pour source *Type* et destination *LinkType* à travers un lien de type *srce* et un lien de type *dest* (flèches pointillées). Il s'agit de la définition de

tous les méta-liens. Tous les éléments conformes à *LinkType*, c'est-à-dire, *R* et ses sous-types (y compris *srce, dest, meta, sub*) se conforment à cette définition.

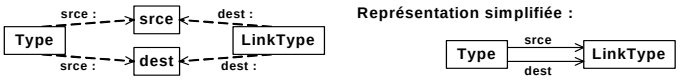

Figure 3. *Structure initiale de srce, dest*

sub. *sub* sert à classifier les types. Il implémente la relation de sous-typage entre types (Besoin 3). L'effet de cette relation (cf. les Règle 1 et Règle 2 ci-dessous) nous permet, en particulier, de déduire toutes les structures nouvelles possibles pour un méta-lien à partir de sa structure initiale en remplaçant dans cette structure l'élément source ou destination par un de ses inférieurs. Comme *NodeType* et *LinkType* sont des inférieurs de *Type*, la structure initiale de *sub* (Figure 4) autorise l'existence des liens de type *sub* entre les *Type*, y compris les *NodeType* et les *LinkType*. De plus, les définitions des types *srce* et *dest* (Figure 3) nous autorisent à définir d'autres structures de *LinkType*. Un *LinkType* peut, à travers un *srce* et un *dest*, unir deux *NodeType* ou un *Type* avec un *NodeType*, ou deux *LinkType*, ou *NodeType* avec un *LinkType*. Ceci implique que nous pouvons définir toutes les sortes de liens : un lien entre deux noeuds, ou entre deux liens, ou entre un noeud et un lien. Le pouvoir d'expression du formalisme augmente alors considérablement. Le méta-lien *sub* est cependant restreint par la Règle 3, présentée un peu plus loin.

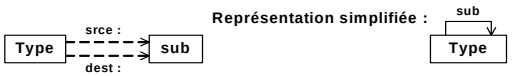

Figure 4. *Définition de sub*

meta. Les liens de type *meta* représentent les relations de conformité qui associent des éléments à leurs méta-éléments, soit du même niveau M3 ou du niveau M2 au niveau M3 dans l'architecture de modélisation. Dans le deuxième cas, ces liens jouent le rôle de transition

entre les niveaux M2 et M3. Un élément (noeud ou lien) au niveau M3 ou M2 a un et un seul méta-élément (méta-noeud ou méta-lien) auquel il est rattaché, une fois défini, par un lien de type *meta*. Les liens de type *meta* permettent alors d'indiquer la nature de chaque élément représenté au niveau M3 ou M2. La définition de *meta* est présentée à la Figure 5. Chaque méta-noeud (ou méta-lien) doit être conforme à *NodeType* (ou *LinkType*). Donc, les structures possibles pour *meta* déduites de sa structure initiale (Figure 5) sont présentées dans la Figure 6. Un *T* (ou un *R*) doit être associé par un lien de type *meta* à un *NodeType* (ou un *LinkType*). La règle sémantique Règle 4 est à retenir. Alors, les rapports de conformité entre les éléments de base de notre M3 décrits au dessus peuvent être illustrés par la Figure 7.

Figure 5. *Définition de meta*

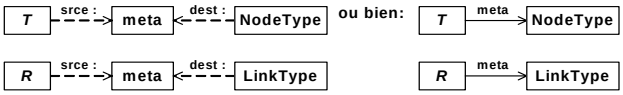

Figure 6. *Structures possibles de meta*

Nous présentons maintenant trois règles qui régissent les éléments de base vus ci-dessus.

Règle 1 : Relation de sous-typage. Si un concept B est lié à A par un lien de sous-typage, chaque instance de B peut se comporter comme instance de A et elle compte parmi les instances de A. Également, si B est un type d'associations, les types unis par B peuvent aussi être unis par A. La relation de sous-typage est transitive et acyclique.

Règle 2 : Restriction de type. Dans un modèle, un type peut être remplacé par un sous-type pour en résulter un autre modèle.

Règle 3 : Sous-typage entre méta-noeuds et sous-typage entre méta-liens. Un méta-noeud (méta-lien) ne peut pas être sous-type d'un méta-lien (méta-noeud).

Règle 4 : Rapport existentiel entre un nœud (lien) et son méta-nœud (méta-lien). Un nœud (respectivement lien) peut exister si et seulement si son méta-nœud (respectivement méta-lien) est défini auparavant au niveau méta. La Figure 7 montre les relations de conformité existant entre les éléments de base de M3

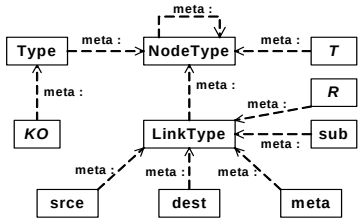

Figure 7. *Éléments de base du méta-métamodèle M3*

4.2. *Autres éléments de M3*

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

globalCard. Les liens de type *globalCard* permettent de représenter des contraintes de cardinalités sur un méta-lien

Language, Value, valueOf, depend. Ces quatre éléments permettent de traiter des valeurs. *Language* et *Value* sont utilisés pour représenter respectivement les langues et les valeurs de l'univers du discours qui servent à représenter les interprétations des éléments. Ces langues et valeurs sont spécifiées et attachées par les liens de type *depend*/ *valueOf*.

Model, Metamodel, Structure, IfThenModel, sem, defAs. Parmi les modèles (*Model*), nous distinguons les métamodèles (*MetaModel*), les structures (*Structure*), et les modèles de conditions (*IfThenModel*). *MetaModel* représente l'ensemble des métamodèles. Un modèle se conforme (*sem*) à un et un seul métamodèle. *Structure* vise à représenter, pour les

méta-liens, les structures ou bien les modèles de définition qui spécifient comment un méta-lien relie ses éléments. Cela est indiqué par des liens de type *defAs*.

defIn, contain, extend, restrict, infer, result, join, diff, intersection. Ces méta-liens permettent aussi de gérer les modèles et leur contenu. Les éléments sont définis (*defIn*) dans un et un seul modèle. Et un modèle peut contenir (*contain*) plusieurs éléments. Un modèle peut étendre (*extend*) d'autres modèles pour fin de réutilisation des éléments du modèle. Il peut aussi permettre de restreindre (*restrict*) ou inférer (*infer*) d'autres modèles. Les modèles peuvent être le résultat (*result*) d'opérations entre modèles telles que des jointures (*join*), des différences (*diff*) ou des intersections (*intersection*).

Nous détaillons ici l'élément *IfThenModel* car il permet de représenter les règles sémantiques des métamodèles.

IfThenModel sert à contextualiser les préconditions et postconditions dans la représentation des règles sémantiques. Relativement au besoin de représenter les règles qu'un élément doit observer, une règle (*Rule*) est attachée à un modèle de préconditions (*IfThenModel*) et à un modèle de postconditions (*IfThenModel*) respectivement par les liens de type *if* / *then*. Pour le traitement des variables dans les règles, nous avons défini les types *Ref* et *EveryRef*. *Ref* correspond à l'interprétation du quantificateur existentiel alors que *EveryRef* correspond à celle du quantificateur universel. La Figure 8 montre un exemple de représentation de la Règle 4 pour le cas de conformité entre liens et méta-liens. Pour tous les éléments possibles représentés par les variables *Rule3-v1*, *Rule3-v2* et *Rule3-v3* (ce qui sont de type *EveryRef*), s'il y a un lien de type *Rule3-v1* reliant *Rule3-v2* à *Rule3-v3*, il doit exister les éléments représentés par les variables *Rule3-v4*, *Rule3-v5* (ce qui sont de type *Ref*) pour que ces conditions soient satisfaites : (i) *Rule3-v4* et *Rule3-v5* sont respectivement le méta-élément de *Rule3-v2* et celui de *Rule3-v3* (ce qui est représenté par un lien de type *meta* de *Rule3-v2* à *Rule3-v4* et par un autre de *Rule3-v3* à *Rule3-v5*); et (ii) *Rule3-v1* est un méta-lien défini pour relier *Rule3-v4* à *Rule3-v5*.

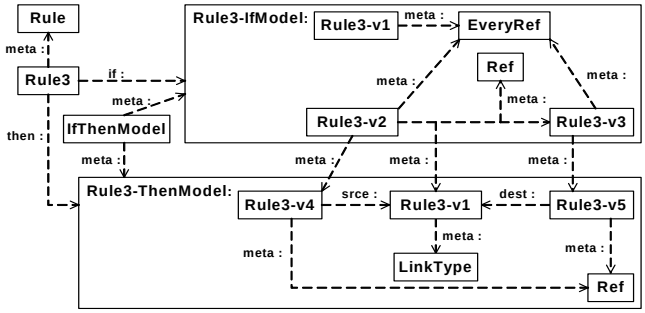

Figure 8. *Exemple de règles*

5. Évaluation du méta-métamodèle

Notre méta-métamodèle (M3) a été défini en réponse aux besoins exprimés dans la section 3. Ce méta-métamodèle permet de représenter des métamodèles de types différents au niveau M2. Pour illustrer cette capacité, les Figure 9, Figure 10, et Figure 11 montrent la représentation de *Marie* dans différents formalismes dont on trouve les éléments nécessaires au niveau M2. Dans ces figures, les liens entre des éléments de niveaux différents sont des liens de conformité. Dans la suite de cette section, nous expliquons chacune des mises en œuvre du méta-métamodèle.

La Figure 9 montre la représentation de *Marie* dans sNets. *Marie* définie dans le modèle *UnModèleM1sNets* au niveau M1 est conforme à *sNetsObject* dans le contexte global, et est également une instance de *Personne* dans le contexte local (ce qui est spécifié par un lien d'instanciation *sNetstype* sur l'axe horizontal). *Personne* est conforme à *sNetsClass*. Dans sNets, *sNetsObject* et *sNetsClass* représentent respectivement l'ensemble de tous les objets et celui de tous les types d'objets; ils sont conformes donc à *NodeType*. Le méta-lien *sNetstype* entre *sNetsObject* et *sNetsClass* est conforme à *LinkType*.

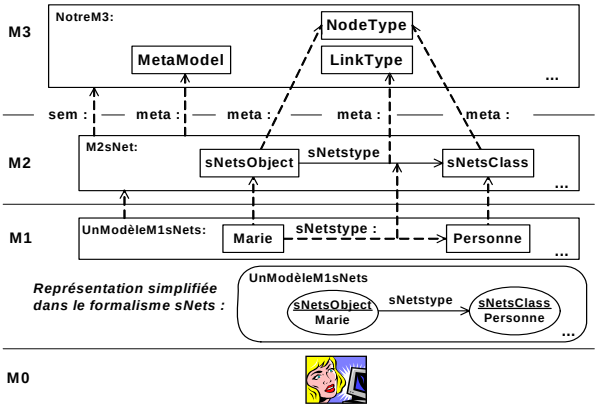

Figure 9. *Marie et sNets*

La représentation de *Marie* dans un formalisme orienté-objet, par exemple UML, est illustrée par la Figure 10. Dans cette figure, *Marie* définie dans le modèle *UnModèleUml* au niveau M1 est conforme à *Instance* dans le contexte global, et est aussi une instance de *Personne* dans le contexte local (ce qui est spécifié par un lien d'instanciation *umltype* sur l'axe horizontal). *Personne* est conforme à *Class*. Dans UML, *Instance* et *Class* représentent respectivement l'ensemble des instances et celui des classes; ils sont conformes donc à *NodeType*. Le méta-lien *umltype* entre *Instance* et *Class* est conforme à *LinkType*.

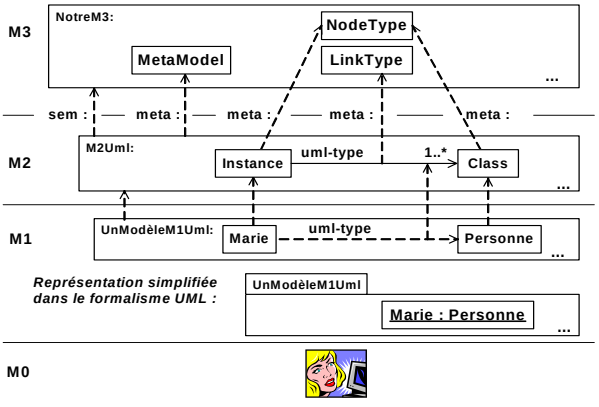

Figure 10. *Marie et UML*

La représentation de *Marie* dans un formalisme utilisé dans le contexte du web sémantique, par exemple RDF/RDFS/OWL, est montrée par la Figure 11. Dans cette figure, *Marie* défini dans une ontologie *UneOntologie* au niveau M1 est conforme à *owlThing* dans le contexte global, et est en outre une instance de *Personne* dans le contexte local (ce qui est spécifié par un lien d'instanciation *rdfstype* sur l'axe horizontal). *Personne* est conforme à *owlClass*. *owlThing* et *owlClass* représentent vis-à-vis l'ensemble des individus et celui des classes; ils sont conformes donc à *NodeType*. Le méta-lien *rdfstype* entre *owlThing* et *owlClass* est conforme à *LinkType*.

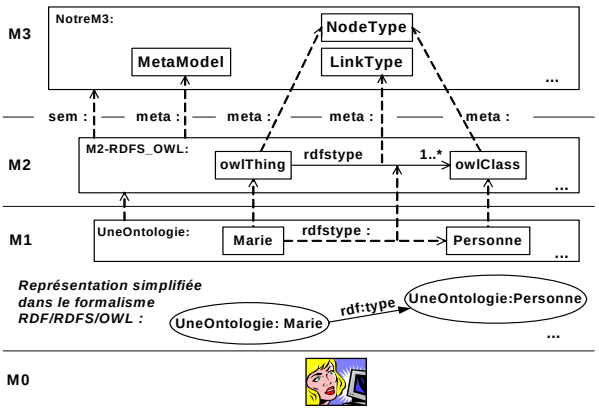

Figure 11. *Marie et RDF/RDFS/OWL*

Nous avons présenté ci-dessus un exemple simple qui illustre la puissance de représentation de notre méta-métamodèle. Pour continuer l'évaluation, nous présentons maintenant une comparaison avec d'autres formalismes

Comme mentionné dans l'introduction, il existe évidemment d'autres candidats pour la représentation de métamodèles : les graphes conceptuels (GCs) (Sowa, 1984; American National Standard, 1999; Gerbé, 2000; Gerbé *et al.*, 2003b), sNets (Lemesle, 2000) basé sur les réseaux sémantiques (Sowa *et al.*, 1991); CDIF (Flatscher, 2002); MOF (OMG, 2003b; OMG, 2004a); XML-XML Schema, RDF-RDFS et OWL (W3C, 2004). Cependant aucun de ces formalismes ne remplit tous les besoins présentés dans la Section 3.

Nous résumons ci-dessous les différents manques identifiés dans (Dinh et al.. 2005).

Les graphes conceptuels et le modèle uniforme (Gerbé, 2000) ne traitent pas explicitement des modèles et de leurs relations. De plus, l'implémentation des GCs est très difficile, plusieurs problèmes restent encore ouverts dont l'interprétation sémantique, les projections et l'opérateur de «matching» dans les GCs.

Le métamodèle de sNets proposé dans (Lemesle, 2000) ne supporte pas l'héritage entre méta-relations (méta-liens) ni les contraintes de cardinalité minimale d'une méta-relation (méta-lien).

CDIF ne traite pas non plus explicitement des modèles et de leurs relations. Concernant la représentation des relations, il permet des relations binaires entre entités mais ne permet pas lier une relation et une autre relation.

MOF est le plus complet et remplit les besoins 2, 3, 4 mais pas entièrement les Besoin 1, Besoin 5 et Besoin 6. Concernant le Besoin 1, MOF supporte différentes notions de base mais pas entièrement la notion de lien. En fait, MOF dans (OMG, 2003b) définit un lien entre deux objets mais un lien n'est pas vu comme un objet ; MOF dans (OMG, 2004a; OMG, 2006) définit un lien entre deux éléments mais un lien n'est pas vu comme un élément. Ceci indique que MOF ne couvre pas toutes les structures possibles pour un lien. Quant aux Besoin 5 et Besoin 6, MOF ne fait pas de distinction entre la relation de conformité d'un élément à un méta-élément et la relation de conformité d'un modèle à un métamodèle et les représente par la même relation « instanceOf » bien que ces deux relations soient de natures très différentes; il n'explicite pas notre notion de modèles de conditions; et il ne représente pas différents types de liens entre modèles comme *l'intersection, la différence, l'inférence, la restriction.*

Enfin, les formalismes XML-XML Schema, RDF-RDFS et OWL sont utilisés dans le contexte du Web sémantique (Berners-Lee, 2001). XML et XML Schema sont reconnus pour leur flexibilité d'expression mais aussi pour

l'ambiguïté sémantique dans la représentation qui rend difficile le traitement sémantique des informations. RDF, RDFS et OWL sont des langages développés explicitement pour la représentation sémantique de l'information sur le Web. Cependant, ils ne supportent pas tous nos besoins pour MB. Par exemple, ils n'explicitent pas différents types de modèles cités dans le Besoin 5 tels que les notions de métamodèles, modèles structurels, modèles de conditions. Ils ne font pas de distinctions entre les notions : conformité entre éléments de différents niveaux, conformité sémantique entre modèles, et instanciation locale « types-instances ». Ils ne considèrent pas non plus les différents liens entre modèles pour la gestion de modèles tels que l'intersection, la différence, l'inférence, la restriction, etc.

6. Conclusion et Travaux futurs

Les travaux sur la gestion de modèles sont applicables dans divers domaines dont la gestion de connaissances qui nous tient particulièrement à cœur. La représentation de modèles y est un axe de recherche essentiel. Comme les formalismes de modélisation couramment utilisés ne rencontrent pas toutes les exigences pour la représentation de modèles, nous avons spécifié un nouveau formalisme qui devrait satisfaire tous les besoins de la gestion de connaissances. Ce formalisme est basé sur les réseaux sémantiques pour fin de facilité de mise en œuvre. Il est important de noter que le méta-métamodèle est extensible. En se basant sur le noyau du méta-métamodèle, il est possible de définir et d'ajouter de nouveaux éléments au besoin.

Conformément au méta-métamodèle, un métamodèle (niveau M2) est défini pour répondre aux besoins concernant la représentation du monde réel. Nos travaux portent actuellement à la définition d'un métamodèle permettant la représentation de connaissances. Les travaux porteront ensuite à la validation théorique de ce métamodèle ainsi que la réalisation d'un outil de validation pour le méta-métamodèle et le métamodèle. Nous travaillerons sur des opérations de manipulation de

modèles qui constituent également un axe important dans la gestion de modèles de connaissances.

Réferences

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.

American National Standard, *Conceptual Graph Standard*, 1999.

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

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

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

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

Chen P., «The Entity-Relationship Model -- Towards a Unified View of Data», *ACM Transactions on Database systems*, Vol. 1(1), 1976, p. 9-36.

Connolly D., Harmelen F.v., Horrocks I., McGuinness D.L., Patel-Schneider P.F., Stein L.A., *DAML+OIL (March 2001) Reference Description*, W3C Note, 2001.

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 T.-L.-A., Métamodèle pour la gestion des modèles, Partie orale de l'examen pré-doctoral, Université de Montréal, 2003.

Dinh T.-L.-A., Gerbé O., «A Metamodel for Knowledge Management», *RIVF'04 - International Conference of French-Speaking or Vietnamese Computer Scientists*, Hanoi, Vietnam, 2004, p. 107-112.

Flatscher R.G., «Metamodeling in EIA/CDIF—Meta-Metamodel and Metamodels», *ACM Trans. on Modeling and Computer Simulation (TOMACS)*, Vol. 12(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», *In 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. editors, *Actes des premières journées sur l'Ingénierie Dirigée par les Modèles (IDM05)*, Paris, France, Juin-Juillet 2005.

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.

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.

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

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

OMG, Meta Object Facility (MOF) 2.0 Core Specification, OMG Adopted Specification, ptc/04-10-15, 2004a.

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

OMG., Unified Modeling Language: Infrastructure, Version 2.0, OMG Adopted Specification, ptc/04-10-14, 2004b.

OMG., Unified Modeling Language: Superstructure, Version 2.0, OMG Adopted Specification, ptc/05-07-04, 2005.

Revault N., Sahraoui H.A., Blain G., Perrot J.-F., «A Metamodeling technique: The MétaGen system», *In Tools 16: Tools Europe'95*, Prentice Hall, Versailles, France, Mar., 1995, p. 127-139 http://www-poleia.lip6.fr/~revault/papers/TOOLSEur95-RSBP.pdf.

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

Schneider D., Modélisation de la démarche du décideur politique dans la perspective de l'intelligence artificielle. Thèse de doctorat, Université de Genéve, 1994.

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.

Stachowiak, H., Gedanken zu einer allgemeinen Theorie der Modelle; Studium Generale, Springer, 1965.

W3C. http://www.w3.org/[1], 2004.

[1] Pour les spécifications des standards XML-XML Schema, RDF-RDF Schema et OWL, notre analyse est basée sur les documents de spécifications datés de 2004.