Une nouvelle méthode graphique pour interroger et vérifier des diagrammes de classes UML

Thomas Raimbault

LERIA, Université d'Angers, 2 boulevard Lavoisier 49045 ANGERS Cedex 01
thomas.raimbault@info.univ-angers.fr

Résumé. UML est le langage graphique de référence dans l'industrie pour la modélisation objet. Cependant UML reste un langage, et ne fournit aucun moyen de vérification ou d'interrogation de ses schémas. Il existe aujourd'hui des outils de vérification, mais ils se comportent comme des boîtes noires où l'utilisateur ne peut accéder. Nous proposons une méthode graphique de vérification et d'interrogation de diagrammes de classes UML. L'aspect intuitif et dessinable de notre méthode offre à l'utilisateur la possibilité d'interroger le contenu de diagrammes de classes, ainsi que de définir et d'adapter ses propres critères de vérification. Le modèle calculatoire de notre approche est celui des graphes conceptuels.

1 Introduction

UML, Unified Modeling Language (Booch et al. 1998), est le langage graphique de référence dans l'industrie pour la modélisation objet. Cependant UML reste un langage, et ne fournit aucun moyen de vérification ou d'interrogation de ses schémas. Il existe des outils commerciaux de vérification, tels que Rational Software Rose (IBM 2004) ou Borland Together (Borland 2004). Mais les vérifications proposées sont uniquement *standards*, vérifiant la cohérence des diagrammes par rapport aux spécifications objet. De plus, la méthode de vérification est dans une boîte noire : les traitements sont de bas niveau et non accessibles à l'utilisateur. Enfin, l'interrogation de diagrammes n'est pas totalement libre mais limité à un cadre pré-formaté de questions.

Pour répondre aux exigences de qualité et d'interaction en modélisation, nous proposons une méthode graphique d'interrogation et de vérification de diagrammes de classes UML. L'aspect intuitif et dessinable de cette méthode offre à l'utilisateur la possibilité de définir et d'adapter ses propres critères de vérification, ainsi que d'interroger librement le contenu de diagrammes de classes UML. Le travail présenté dans cet article est issu de (Raimbault 2004), et est traité pour l'atelier EGC 2005 "*Modélisation des connaissances*" de façon intuitive au travers un exemple. Concrètement, notre méthode utilise pour les calculs le modèle des graphes conceptuels (Sowa 1984).

Cet article est structuré comme suit : la section 2 traite de notre méthode graphique d'interrogation et de vérification de diagrammes de classes UML. En section 3, nous abordons l'aspect calculatoire de notre méthode qui utilise la modèle des graphes conceptuels. La section 4 discute des résultats et des perspectives de notre méthode.

2 Interroger et Vérifier un diagramme de classes

Nous indiquons d'une part comment formuler une *requête* pour interroger le contenu d'un diagramme de classes UML, d'autre part, comment définir les *critères de validité*

de diagrammes de classes. Ces vérifications peuvent être des standards que l'on retrouve en modélisation objets ou des vérifications dites de métier qui sont spécifiques aux besoins de l'utilisateur. L'aspect dessinable et intuitif des requêtes ou des critères de validité les rend accessibles et manipulables par l'utilisateur. De plus, contrairement à des outils commerciaux, les vérifications des diagrammes de classes ne se comportent pas comme des boîtes noires.

2.1 Interrogation

Pour interroger un diagramme de classes, il suffit de formuler la requête sous forme d'un diagramme de classes. La réponse à une requête, si elle existe, est une sous partie du diagramme de classes à interroger dans laquelle le diagramme requête se projette. Supposons que nous voulons interroger le diagramme de classes Figure 1 par la requête suivante : "la classe '4x4' a-t-elle une classe mère ?". Cette requête est formulée par le diagramme requête présenté à gauche de la Figure 2 où la classe '4x4' a pour généralisation une classe générique, notée *. Cette requête admet une réponse, via une projection du diagramme requête sur le diagramme Figure 1, qui est : "Oui, la classe '4x4' possède bien une classe mère ; de plus, cette classe mère est la classe 'Voiture"'.

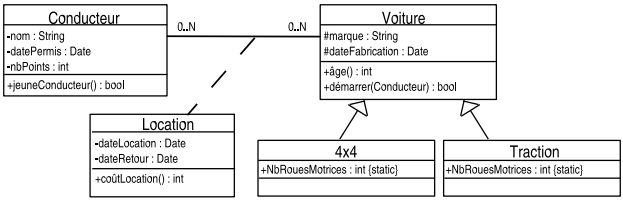

FIG. 1 – *Exemple de diagramme de classes UML.*

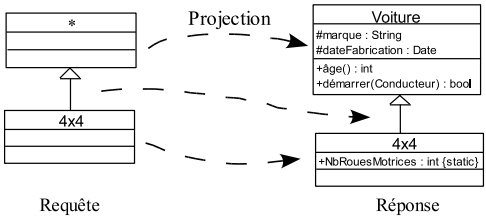

FIG. 2 – *Exemple de diagramme requête et sa réponse.*

Des requêtes plus riches peuvent être modélisées. Par exemple, "obtenir les attributs statiques des sous-classes d'une classe donnée" ou bien "quelles sont les classes publiques associées avec une multiplicité de 1 à une classe donnée".

2.2 Vérification

Nous disposons de deux types de critères de validité de diagrammes de classes UML. Un critère positif est de la forme "si *condition*, il faut que *obligation*", et un critère négatif de la forme "si *condition*, il ne faut pas *interdiction*".

Un critère de validité, positif ou négatif, est modélisé sous la forme d'un diagramme de classes bicolore où la condition est sur fond blanc et l'obligation ou l'interdiction sur

fond noir. Un critère de validité positif est repéré par le symbole $\boxed{+}$, et un négatif par $\boxed{-}$. La Figure 3 présente trois critères de validité positifs, C_1, C_2 et C_3, et deux critères de validité négatifs, C_4 et C_5. Le critère C_1 a pour condition une classe, notée * car non spécifiée, qui possède une opération abstraite non spécifiée elle aussi. L'obligation de ce critère est que la classe doit être abstraite. En d'autres termes, C_1 exprime le fait que "Si une classe possède une opération abstraite, alors cette classe doit être déclarée abstraite". C_2 stipule que "Si une classe A possède des opérations abstraites, alors une sous-classe B de A ne peut être non abstraite qu'à la condition que les opérations abstraites de A soient redéfinies comme non abstraites dans B". C_3 indique que "toute classe doit être associée à une autre classe". C_4 interdit tout cycle d'héritage. La condition de C_5 est une classe possédant une opération finale notée $\$x$, et l'interdiction est que cette même opération $\$x$ soit présente dans une sous-classe de cette classe. Donc, "une opération finale, d'une classe A, ne peut être redéfinie dans une classe B qui est sous-classe de A".

Remarquons d'une part que pour les critères de validité présentés Figure 3, le lien de généralisation est considéré comme transitif. D'autre part, ces mêmes critères de validité, à l'exception de C_3, sont des critères *standard* de validité en modélisation objet. C_3 est un exemple de critère de validité qui peut être spécifique aux besoins d'un utilisateur, nous le nommons *critère métier*.

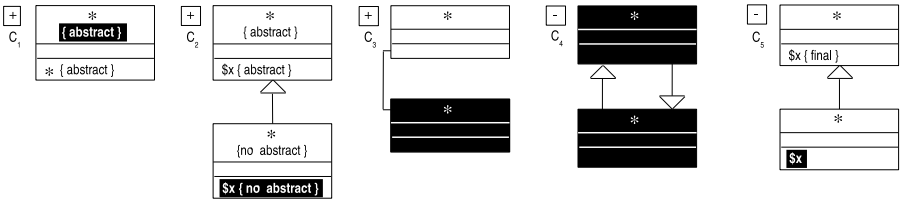

FIG. 3 – *Exemple de critères de validité positifs et négatifs.*

3 Graphes conceptuels, Règles et Contraintes

Les *graphes conceptuels* (GCs) constituent un modèle formel de représentation des connaissances, de la famille des réseaux sémantiques (Lehmann 1992). (Sowa 1984) introduit un modèle simple des GCs muni d'une sémantique logique du premier ordre. Les GCs permettent une interprétation du dessin le représentant en termes de relations entre des concepts.

Dans la pratique, une connaissance donnée est modélisée par un GC. Il s'agit d'un graphe biparti étiqueté, où les étiquettes d'une catégorie de sommets correspondent à des noms de *concepts* et celles de l'autre catégorie à des noms de *relations*. Ce GC est défini sur un *support* qui spécifie le vocabulaire de base pour représenter les connaissances sur le domaine modélisé. Chaque élément du modèle des GCs est interprétable en logique du premier ordre par un opérateur qui traduit un GC en une formule logique.

Notre méthode graphique d'interrogation et de vérification de diagrammes de classes UML est développée en utilisant le modèle des GCs (Mugnier et Chein 1996) emboîtés typés (Chein et Mugnier 1997), avec liens de coréférence (Chein et Mugnier 2004), règles

(Salvat 1998) et contraintes (Baget et Mugnier 2002). L'idée clé de cette implémentation est de modéliser un diagramme de classes UML dans le formalisme des GCs, où d'un coté le support définit les notations UML et d'un autre côté l'agencement des sommets entre eux d'un GC ainsi que les marqueurs individuels des sommets concepts définissent un diagramme de classes donné. Ensuite, nous utilisons les possibilités de raisonnement logique qu'offre le modèle des GCs avec l'utilisation d'opérateurs graphiques. L'application de règles permet de faire ressortir de façon explicite des informations implicites, comme par exemple la transitivité du lien de généralisation entre classes. Les critères de validité positifs et négatifs sont respectivement modélisés par des contraintes positives et négatives. La vérification de ces contraintes détermine si un diagramme de classes est valide ou non selon les critères choisis par l'utilisateur.

La Figure 4 représente la modélisation, en un GC, des classes 'Conducteur' et 'Voiture' de la Figure 1 et leur association. La classe 'Voiture' par exemple est modélisée par un sommet concept de type *class* (type défini dans le support) qui a pour marqueur individuel *Voiture*. Cette classe possède des attributs et des opérations qui eux aussi sont modélisés par des sommets concepts. Ces derniers décrivent la classe 'Voiture' et sont donc emboîtés au sein du sommet concept qui la représente. Tout comme la classe 'Voiture', ses attributs et opérations possèdent leurs propres descriptions qui sont respectivement emboîtés dans les sommets concepts les représentant. Ces emboîtements ne sont pas visibles sur la Figure 4 mais sous-entendus avec les caractères '. . . '. L'association entre les classes 'Conducteur' et 'Voiture' est centralisée autour de la classe d'association 'Location' par l'intermédiaire de sommets relations de type *association*. Remarquons que les emboîtements d'un sommet concept constituent des niveaux de lecture plus internes de l'information le représentant. L'utilisateur peut ainsi plus ou moins zoomer sur la description d'une partie du diagramme de classes.

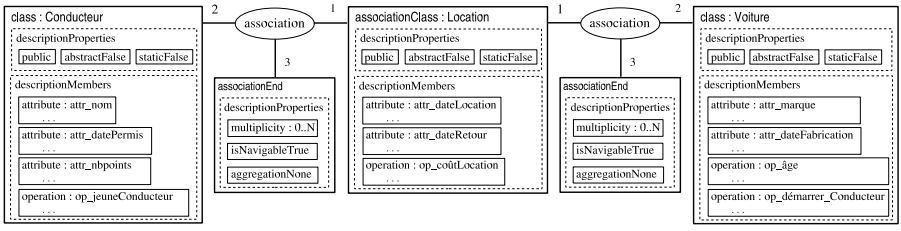

FIG. 4 – *Modélisation en un GC des classes 'Conducteur', 'Voiture' et leur association.*

4 Résultats et perspectives

Un prototype a été développé pour d'une part réécrire un diagramme de classes UML au format XMI (OMG 2002) dans le formalisme des GCs emboîtés typés au format BCGCT (Haemmerlé O. 1995). D'autre part, règles et contraintes ont été écrites au format BCGCT pour vérifier des diagrammes de classes et satisfaire au concept objet. La vérification tout comme l'interrogation d'un diagramme de classes, sous forme d'un GC, a été testé sur la plateforme CoGITaNT (Genest 2004). Les résultats obtenus sur des exemples simples de diagrammes de classes sont tout à fait satisfaisants. Ils

fournissent les exigences souhaitées et ce de façon quasi instantanée.

Nous souhaitons étendre notre méthode graphique de modélisation, d'interrogation et de vérification aux différents diagrammes UML. Cette extension offrira ainsi un formalisme unique et visuel pour modéliser tous les diagrammes UML ainsi que des requêtes ou des vérifications à effectuer sur et entre ces derniers. Actuellement, une interface est en cours de développement pour permettre à l'utilisateur de formuler ses requêtes et vérifications dans un formalisme UML (cf. Figure 3). Ainsi, le modèle des GCs peut être transparent pour l'utilisateur qui le désire.

Références

Baget J.F. et MugnierM.-L (2002), Extensions of Simple Conceptual Graph : the complexity of Rules and Contraints. JAIR, vol. 16, pp. 425-465, 2002.

Booch G., Jacobson C. et Rumbaugh J. (1998), The Unified Modeling Language - a reference manual. Addison Wesley.

Borland (2004), Together, http://www.borland.com/together/.

Chein M. et Mugnier M.-L. (1997), Positive Nested Conceptuals Graphs. In Proc. of ICCS'97, vol. 1257 of LNAI, pages 95-109, Springer Verlag.

Chein M. et Mugnier M.-L. (2004), Concept types and coreference in simple conceptual graphs. In Proc. of ICCS'2004, vol. 3127 of LNAI, pages 303-318, Springer Verlag.

Genest D. (2004), CoGITaNT 5.1.5, LIRMM-LERIA, http://cogitant.sourceforge.net

Haemmerlé O. (1995), Plate-forme CoGITo, Rapport technique 95012, LIRMM.

IBM (2004), Rational Software Rose, http://www-306.ibm.com/software/rational/

Lehmann F. (1992), Semantic Networks in Artificial Intelligence, Pergamon Press.

Mugnier M.-L. et Chein M. (1996), Représenter des connaissances et raisonner avec des graphes, vol.10 dans RIA, n°1, 7-56.

OMG (2002), Object Management Group Documentation, XML Metadata Interchange (XMI), http://www.omg.org/technology/documents/formal/xmi.htm

Raimbault T. (2004), Une nouvelle méthode graphique pour vérifier et interroger des diagrammes de classes UML, Mémoire de DEA, LERIA.

Salvat E. (1998), Theorem Proving Using Graph Operations in the Conceptual Graph Formalism. In Proc. of ECAI'98.

Sowa J.F. (1984), Conceptual Structures - Information Processing in Mind and Machine, Addison-Wesley.

Summary

UML is an industrial reference graphic modeling language to express object systems. However UML is just a language, it does not provide any means to check or interrogate diagrams. There are commercial tools today, but they behave as black boxes where the user cannot accede into. We propose a graphic method to check and interrogate UML class diagrams. The intuitive and drawing aspect of our method make it possible to the user to interrogate contents of UML class diagrams, and to define and adapt its own criteria of checks. The calculative model of our approach is that of conceptual graphs.