Une méthode de diagnostic reposant sur l appariement structurel de diagrammes de classes dans un EIAH pour la modélisation orientée objet

Ludovic Auxepaules, Dominique Py, Thierry Lemeunier

Une méthode de diagnostic reposant sur l'appariement structurel de diagrammes de classes dans un EIAH pour la modélisation orientée objet

Ludovic Auxepaules, Dominique Py, Thierry Lemeunier

LIUM
Université du Maine
Avenue Laënnec
72085 Le Mans Cedex 9
prenom.nom@lium.univ-lemans.fr

RÉSUMÉ. *Nous décrivons dans cet article la méthode de diagnostic des productions des apprenants développée dans Diagram, un EIAH pour la modélisation orientée objet. Après avoir précisé le contexte et les objectifs du diagnostic, nous présentons notre méthode de diagnostic, qui repose sur la comparaison entre deux diagrammes (le diagramme de l'apprenant et un diagramme solution) au moyen d'un algorithme d'appariement structurel. Enfin, nous discutons les intérêts et les limites de cette approche et évoquons des perspectives.*

MOTS-CLÉS : *diagnostic, appariement structurel, modélisation orientée objet, diagrammes de classes, UML.*

1. Introduction

En EIAH, la faculté pour un système de valider ou d'invalider les réponses de l'apprenant est un point crucial qui conditionne la pertinence des rétroactions pédagogiques. C'est pourquoi la conception de résolveurs à caractère pédagogique, capables de produire des solutions et d'analyser celles de l'élève, a donné lieu à de nombreux travaux dans les domaines bien formalisés (algèbre, géométrie). Dans les domaines plus ouverts, où la mise en œuvre d'un résolveur n'est pas envisageable, il demeure néanmoins important que le système soit apte à diagnostiquer et critiquer les productions des apprenants. La modélisation, entendue ici comme une tâche de conversion d'un registre langagier à un registre semi-formel, fait partie de ces tâches ouvertes qui ont suscité le développement d'EIAH spécifiques ([KOMIS et al. 03], [SURAWEERA & MITROVIC 04], [BAGHAEI & MITROVIC 06]). On distingue deux approches du diagnostic dans le contexte de l'apprentissage de la modélisation, l'une basée sur le curriculum et l'autre sur la notion de contrainte. Ces deux approches ont en commun de recourir à un diagramme de référence, créé par un expert du domaine, qui représente la « solution idéale » au problème.

Diagram est un EIAH dédié à la modélisation orientée objet, et plus particulièrement à la réalisation de diagrammes de classes UML [ALONSO et al 07]. La nature des connaissances visées par l'apprentissage, ainsi que l'absence de méthodes formelles permettant de construire un diagramme ou de vérifier son adéquation à un énoncé, nous ont conduits à mettre l'accent sur l'acquisition par l'apprenant de procédures de contrôle, de correction et de validation de ses productions, et à viser la négociation et la co-validation des diagrammes entre l'utilisateur et l'EIAH. Ceci requiert du système l'aptitude à critiquer un diagramme de classes en y repérant des erreurs ou des incohérences. Actuellement, les aides proposées par Diagram sont générales et ne tiennent pas compte de la validité des diagrammes élaborés. L'objet du diagnostic est de repérer les erreurs ou les incohérences du diagramme de l'apprenant, en vue de le guider par des rétroactions plus spécifiques.

Nous décrivons dans cet article la méthode de diagnostic des productions des apprenants développée dans Diagram. Après avoir précisé le contexte et les objectifs du diagnostic, nous présentons notre méthode de diagnostic, qui repose sur la comparaison entre diagrammes (le diagramme de l'apprenant et un diagramme solution) au moyen d'un algorithme d'appariement structurel. Enfin, nous discutons les intérêts et les limites de cette approche et évoquons des perspectives.

2. Le diagnostic des modèles produits par l'apprenant

Notre objectif est de concevoir une méthode de diagnostic robuste et générique des modèles créés par l'apprenant. En l'absence de résolveur pour les problèmes de modélisation, nous effectuons le diagnostic en comparant le diagramme de

l'apprenant à un diagramme de référence (fourni par l'enseignant) représentés dans le même formalisme. Nous faisons l'hypothèse que les différences de structure ou de contenu, si elles ne traduisent pas nécessairement une « erreur » au sens strict, peuvent suggérer que l'étudiant a effectué un mauvais choix de modélisation, ou qu'il a omis de représenter un élément du modèle par exemple.

Pour concevoir l'algorithme de diagnostic, nous sommes partis, d'une part, d'une étude du métamodèle objet [UML 07], et d'autre part, de l'analyse d'un corpus de diagrammes d'étudiants comparés aux diagrammes attendus par l'enseignant, les diagrammes des étudiants ayant été produits en situation réelle d'apprentissage.

Nous avons ainsi défini le diagnostic en combinant deux méthodes. La première est une méthode de validation syntaxique du modèle de l'apprenant axée sur la syntaxe et les conventions de représentation intrinsèques au formalisme du modèle construit. La transgression de règles de cohérence transcrites par [SEUMA VIDAL et al. 05] pour les diagrammes UML, permet d'identifier des erreurs syntaxiques telles qu'un cycle d'héritage par exemple.

La seconde est une méthode de comparaison et d'appariement structurel paramétrable : elle schématise tout d'abord les modèles sous forme de motifs structurels caractéristiques puis compare et apparie ces structures en utilisant des fonctions de similarité. Cette méthode est détaillée dans les parties suivantes.

2.1. La schématisation des modèles en motifs structurels caractéristiques

Afin de spécifier des niveaux de granularité d'appariement, nous avons repéré des structures caractéristiques que nous nommons « motifs ». Nous avons isolé d'une part des motifs simples correspondant aux éléments du méta-modèle UML et d'autre part des motifs complexes correspondant aux structures des motifs simples. Pour les diagrammes de classes, nous avons identifié les motifs structurels suivants.

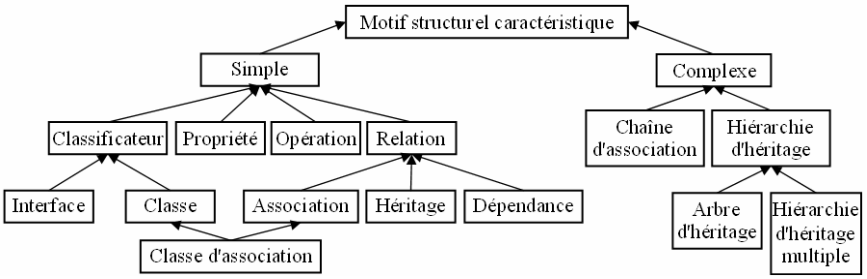

Figure 1. *Motifs structurels caractéristiques d'un diagramme de classes UML 2.0*

La figure 2 montre les motifs complexes (schéma de droite) d'un diagramme : deux chaînes d'associations (les rectangles) et une hiérarchie d'héritage multiple constituée de deux arbres d'héritage (les triangles tronqués).

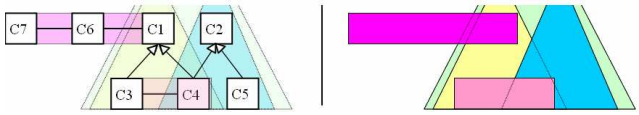

Figure 2. *Schématisation d'un diagramme de classes en motifs complexes*

2.2. La comparaison et l'appariement des modèles

Pour automatiser la comparaison, nous proposons de considérer un diagramme UML comme un graphe dans lequel les sommets sont les classificateurs et les arcs les relations. Pour l'appariement, nous nous sommes inspirés des travaux de [SORLIN et al. 06] qui proposent une mesure générique de similarité paramétrée par des fonctions exprimant des connaissances propres au domaine, à l'application considérée ainsi qu'au type d'appariement recherché (univoque ou multivoque[1]).

Dans Diagram, la comparaison est réalisée en calculant un score de similarité pour les motifs comparés les uns par rapport aux autres. Les scores sont calculés par le biais de fonctions paramétrables pondérant des critères aussi bien qualitatifs que quantitatifs que nous ne pouvons décrire ici faute de place.

Nous utilisons un algorithme « glouton » descendant car la comparaison des motifs complexes dirige la comparaison des motifs simples. Il conduit l'appariement de manière univoque et ensuite de manière multivoque s'il n'est pas satisfaisant. Il peut également repérer des constructions alternatives proches grâce à des métaconnaissances sur la sémantique de la généralisation[2], la relaxation des associations[3] et la restructuration des classes d'association, ce qui permet une comparaison plus large et pas seulement basée sur une stricte égalité.

3. La taxonomie des différences repérées dans les modèles

L'appariement des modèles donne en sortie des motifs appariés strictement et des motifs appariés moyennant des différences organisées dans la taxonomie suivante.

---

[1] Un appariement des sommets de graphes est multivoque lorsque chaque sommet d'un graphe peut être apparié à un ensemble de sommets de l'autre graphe. Lorsque chaque sommet est apparié à un autre somment, l'appariement est univoque [SORLIN et al. 06].

[2] La généralisation a une sémantique plus large que le simple fait de relier deux classes car elle implique notamment la propagation des propriétés (relations, attributs, opérations) de la classe la plus générale vers les classes spécialisées [KHAYATI 05].

[3] La relaxation peut s'appliquer sur une association (association ↔ agrégation ↔ composition) de manière descendante ou ascendante, sur le type d'un paramètre (par généralisation ou spécialisation), sur une cardinalité monovaluée (1 ↔ 0..1), ou sur une cardinalité multivaluée (1..* ↔ 0..*) [KHAYATI 05].

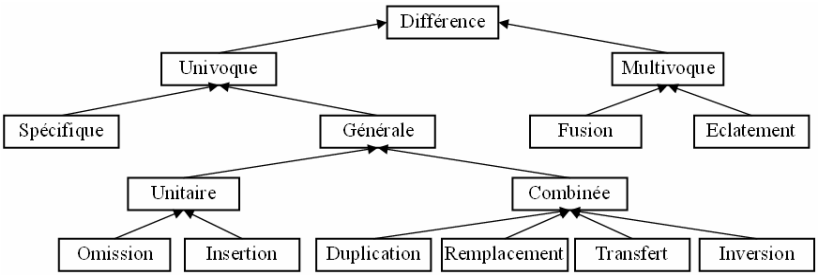

Figure 3. *Taxonomie des différences*

Les différences spécifiques concernent les propriétés et la sémantique de certains motifs. Pour les diagrammes de classes, elles sont relatives au nom, à la visibilité, à l'abstraction, aux multiplicités, à la nature, au conteneur, et aux éléments contenus.

Les autres différences concernent tous les motifs. L'insertion d'un motif est due à son absence dans le modèle attendu (c'est une sur-spécification de l'énoncé), et l'omission à sa présence dans le modèle attendu (c'est une sous-spécification de l'énoncé). En combinant les différences unitaires, nous pouvons déduire des différences plus complexes comme le remplacement, le transfert ou l'inversion. Un même concept de l'énoncé peut être modélisé avec un ou plusieurs motifs (différence multivoque d'éclatement) et inversement (différence multivoque de fusion).

Par exemple, la figure 4 représente à gauche un diagramme erroné et à droite un diagramme correct. Ces deux diagrammes sont appariés avec des différences locales relevées par le diagnostic : d'une part, les attributs « nom » et « adresse » n'ont pas été représentées dans la classe « Personne » (deux différences d'omission) mais ont été représentés dans la classe « Animateur » (deux différences d'insertion). A partir de ces différences unitaires, une différence combinée de transfert est déduite : les attributs « nom » et « adresse » ont été transférés de la classe « Personne » à la classe « Animateur ». D'autre part, la relation d'héritage entre la classe « Personne » et la classe « Animateur » a été inversée (inversion de la classe mère et de la classe fille).

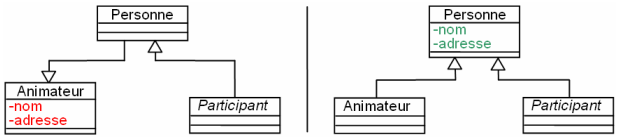

Figure 4. *Exemples de différences combinées*

4. Conclusions et perspectives

Nous proposons ici une méthode de diagnostic des modèles graphiques pour lesquels il n'existe pas de résolveur automatique. Le diagnostic exploite des aspects structurels et sémantiques intrinsèques à ces modèles. Leur comparaison, reposant sur des fonctions de similarité paramétrables, peut s'adapter aux modèles et au type d'appariement recherché. Il serait envisageable d'appliquer ces résultats à d'autres domaines proches, comme la modélisation entité-relation des bases de données.

Une limite de notre approche est qu'elle repose sur un seul diagramme solution. Néanmoins, l'algorithme mis en place doit permettre de prendre en compte d'autres sources de connaissances pendant la comparaison, comme des extraits de modèles corrects ou erronés définis par l'enseignant par exemple.

Actuellement, nous achevons l'implantation en JAVA du diagnostic dans Diagram. Dans un but de généricité et de réutilisabilité, nous prenons en compte les normes et standards actuels de la modélisation orientée objet avec notamment une représentation des diagrammes en adéquation avec le métamodèle UML 2.x et une possibilité d'échanger les diagrammes sous forme de fichiers XMI 2.1.

Nous avons pour perspectives de tester tout d'abord le diagnostic *a posteriori* sur des modèles construits par des apprenants avec Diagram, afin de vérifier la validité de notre proposition. Ensuite, nous expérimenterons le module de diagnostic couplé au module Interaction [ALONSO et al. 07] interagissant avec l'apprenant en lui fournissant des rétroactions pendant l'activité de modélisation.

5. Bibliographie

[ALONSO et al. 07] Alonso, M., Py, D., Lemeunier, T., « Interaction support à la métacognition dans un EIAH pour la modélisation orientée objet », *3ème conférence en Environnements Informatiques pour l'Apprentissage Humain 2007*, Lausanne, juin 2007.

[BAGHAEI & MITROVIC 06] Baghaei, N., Mitrovic, A., « A Constraint-Based Collaborative Environment for Learning UML Class Diagrams », *8th International Conference on Intelligent Tutoring Systems 2006*, Taiwan, juin 2006, p. 176-186.

[KHAYATI 05] Khayati, O., Modèles formels et outils génériques pour la gestion et la recherche de composants, Thèse de doctorat, INPG Institut National, Polytechnique de Grenoble, décembre 2005, 231p.

[KOMIS et al. 03] Komis, V., Avouris, N., Dimitracopoulou, A., Margaritis, M., « Aspects de la conception d'un environnement collaboratif de modélisation à distance », *Actes de la 1ère conférence EIAH 2003*, Strasbourg, avril 2003, p. 271-282.

[SEUMA VIDAL et al. 05] Seuma Vidal, J-P., Malgouyres, H., Motet, G., Règles de Cohérence UML 2.0, rapport interne du LESIA, juillet 2005, 194 p.

[SORLIN et al. 06] Sorlin, S., Sammound, O., Solnon, C., Jolion, J.-M., « Mesurer la Similarité de Graphes », *Actes des 6e journées francophones Extraction et Gestion des Connaissances 2006, Atelier ECOI 2006*, janvier 2006, p. 21-30.

[SURAWEERA & MITROVIC 04] Suraweera, P., Mitrovic, A., « An Intelligent Tutoring System for Entity Relationship Modeling », *International Journal of Artificial Intelligence in Education*, vol. 14, n° 3-4, 2004, p. 375-417.

6. Références sur le WEB (dernières consultations le 1er mai 2007)

[UML 07] Unified Modeling Language, Object Management Group, http://www.uml.org/.