

---

## Interaction support à la métacognition dans un EIAH pour la modélisation orientée objet

**Mathilde Alonso, Dominique Py, Thierry Lemeunier**

*LIUM*

*Avenue Laënnec*

*72085 Le Mans cedex 9*

*Mathilde.Alonso@lium.univ-lemans.fr*

*Dominique .Py@lium.univ-lemans.fr*

*Thierry.Lemeunier@lium.univ-lemans.fr*

---

*RÉSUMÉ. Cet article décrit la conception de l'interaction dans Diagram, un EIAH dédié à l'apprentissage de la modélisation orientée objet. L'aspect « ouvert » de la tâche de modélisation conduit à mettre l'accent sur les aptitudes métacognitives (planification, contrôle et évaluation) que l'environnement doit favoriser chez l'apprenant. Le modèle d'interaction repose sur une organisation de la tâche en trois étapes, l'intégration de l'énoncé dans l'interface, des outils de modélisation graphique spécifiques et des aides contextuelles pour la création et la vérification des éléments du diagramme. Le logiciel a été développé en Java sous Eclipse, utilise XML et JGraph. Il a fait l'objet d'expérimentations en contexte écologique.*

*MOTS-CLÉS: interaction, guidage, rétroactions, métacognition, modélisation orientée objet, diagramme de classe, UML.*

---

## 1. Introduction

En génie logiciel, la modélisation est une activité cruciale de l'étape d'analyse dans le processus de développement d'un produit logiciel. Elle a pris ces dernières années une part importante des cursus informatiques universitaires à finalité professionnelle, surtout depuis l'avènement du langage de modélisation orienté objet UML (*Unified Modeling Language*). L'apprentissage de la modélisation commence également à susciter des travaux en EIAH, comme [KOMIS et al. 03], [THOLANDER & KARLGREN 02], [SURaweera & MITROVIC 04].

Dans cet article, nous nous intéressons à la conception d'un EIAH dédié à l'apprentissage de la modélisation orientée objet. Il s'agit de définir les modes d'interaction pertinents et les connaissances à embarquer dans un environnement visant à faciliter l'acquisition des concepts de la modélisation UML et l'émergence de schèmes d'action instrumentés chez des utilisateurs novices.

## 2. Cadre théorique et méthodologique

### 2.1. Aspects métacognitifs de la tâche de modélisation

La tâche de modélisation UML est une activité de conversion du registre langagier au registre symbolique, sans action de transformation sur les représentations elles-mêmes, contrairement aux activités de preuve ou de résolution. La nature de cette tâche, et notamment l'absence de méthodes formelles pour construire un diagramme ou vérifier son adéquation à l'énoncé, nécessitent de mettre l'accent sur l'acquisition par l'apprenant de procédures de contrôle, de correction et de validation de ses productions. L'environnement doit donc favoriser l'activité réflexive et métacognitive de l'apprenant par rapport à son travail, en lui offrant des moyens explicites d'analyse et de justification à l'interface.

La notion de métacognition proposé par [FLAVELL 77] se décline en deux dimensions : les connaissances métacognitives d'un individu et les régulations métacognitives qui se réfèrent aux activités supportant le contrôle individuel de la pensée ou de l'apprentissage. [SHRAW & MOSHAM 95] distinguent trois fonctions de régulation : la planification d'activités (*planning*), le contrôle et la surveillance d'activités (*monitoring*) et l'évaluation des résultats de l'activité (*evaluation*). Une aide efficace à la métacognition requiert de prendre en compte ces trois dimensions.

### 2.2. Modélisation UML et enseignement

Le formalisme UML permet de produire à l'aide de diagrammes une représentation simplifiée d'un domaine réel. Il comprend plusieurs types de

diagrammes, dont le diagramme de classes qui est le plus employé et le mieux connu, et auquel nous limitons notre étude.

L'enseignement de la modélisation orientée objet dans les cursus universitaires est composé de séances théoriques et pratiques. Ces travaux pratiques ont recours à des logiciels professionnels (tels que des éditeurs UML) qui ne sont pas adaptés aux finalités pédagogiques : ils sont généralement complexes, longs à maîtriser et présentent des fonctionnalités inutiles pour des séances d'initiation, de plus les aides ne sont pas adaptées à des utilisateurs novices. Nous avons donc choisi de concevoir l'environnement Diagram comme un éditeur de diagrammes de classes, qui comporte un sous-ensemble des fonctionnalités des éditeurs classiques et qui intègre des modalités d'interaction et des aides spécifiques aux novices.

### 3. L'interaction dans Diagram

Diagram offre la possibilité de travailler simultanément avec l'énoncé et le diagramme de classes, ce qui facilite le contrôle visuel de la modélisation. Cette fonctionnalité, absente des éditeurs UML classiques, est présente dans les environnements Kermit [SURAWEERA & MITROVIC 04] et COLLECT-UML [BAGHAEI & MITROVIC 05]. Elle donne de plus grandes possibilités d'interaction car même si le texte n'est pas modifiable, il est possible d'agir sur l'énoncé et de modifier son aspect visuel. De plus, Diagram intègre une organisation de la tâche et offre des aides contextuelles qui encouragent l'activité métacognitive chez l'étudiant.

#### 3.1. Méthode de modélisation en trois phases

Le troisième auteur, enseignant le langage UML à l'université du Maine, a mis au point une méthode pédagogique de modélisation qu'il utilise avec ses étudiants. Cette méthode comporte trois étapes : la première consiste à lire entièrement l'énoncé pour se l'approprier, la deuxième consiste à élaborer le diagramme et la troisième consiste à relire une dernière fois l'énoncé en vérifiant l'exactitude du diagramme. Cette organisation de l'activité de modélisation exerce une fonction de planification et d'évaluation au sens de la régulation métacognitive. Nous avons choisi de la transposer dans l'environnement Diagram et de l'enrichir avec des outils graphiques soit inspirés des outils papier-crayon, soit permis par l'environnement informatique.

Durant l'étape de lecture, l'étudiant découvre le sujet général de l'énoncé. Certains étudiants utilisent un crayon ou un surligneur pour marquer les mots qui leur semblent pertinents pour la modélisation. Pour qu'ils retrouvent cette fonctionnalité dans l'EIAH, nous avons intégré à Diagram la possibilité de souligner des mots de l'énoncé.

La seconde étape consiste à créer le diagramme de classes correspondant à l'énoncé donné. Diagram permet de créer directement un élément UML (classe, attribut ou relation) à partir d'une expression de l'énoncé. L'expression sélectionnée

et l'élément graphique sont affichés dans une couleur identique, afin de faciliter le contrôle visuel. Cette modalité de création est une originalité de Diagram qui vise à renforcer le lien entre énoncé et modèle.

Il est également possible de créer un élément « libre » dans le diagramme, sans lien avec l'énoncé (par exemple, une relation qui serait implicite dans le texte). Tout élément libre du diagramme peut être relié ultérieurement à l'énoncé, et toute expression libre de l'énoncé peut être reliée à une autre expression (synonyme) de l'énoncé. L'élément ou l'expression libre prend alors la même couleur que l'expression à laquelle il est rattaché, matérialisant ainsi le lien logique entre eux.

Durant la dernière étape, l'élève doit relire l'énoncé de l'exercice et le comparer à son diagramme de classes afin de vérifier que le modèle est complet et correct par rapport à l'énoncé. Au début de cette phase, l'interface ne présente plus que l'énoncé, en noir et blanc. Au fur et à mesure que l'étudiant passe la souris sur le texte, les expressions modélisées redeviennent en couleur ; simultanément, les éléments correspondants dans le diagramme réapparaissent à l'interface.

### **3.2. Aides contextuelles**

Une première expérimentation du logiciel, menée en 2005 avec un petit groupe d'étudiants, nous a conduit à proposer des aides supplémentaires pour renforcer les fonctions de monitoring et d'évaluation de la régulation métacognitive.

#### *3.2.1. Aide à la création d'éléments*

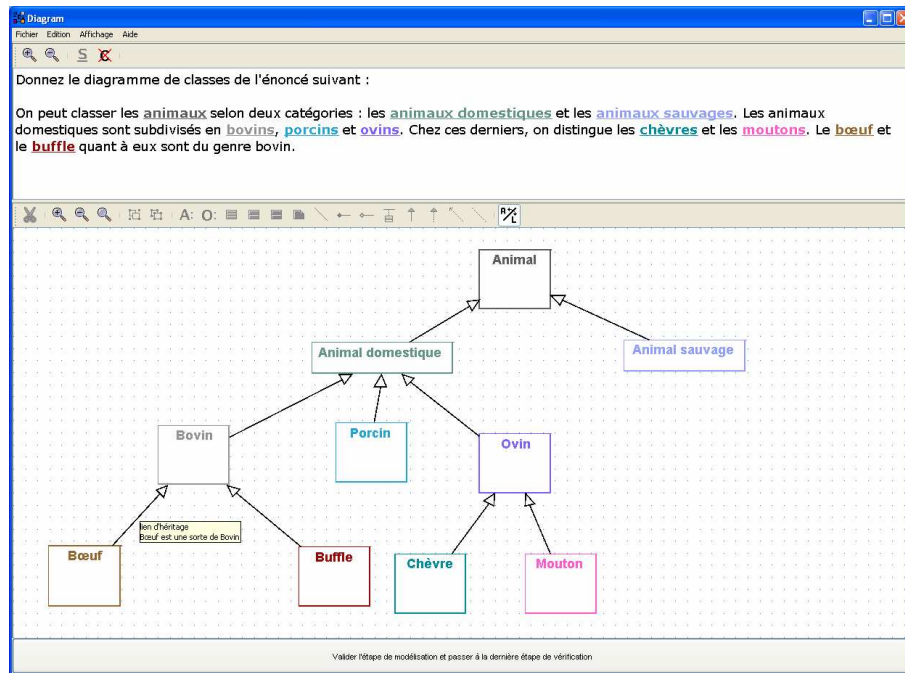
L'expérimentation a montré que les apprenants avaient parfois des difficultés à utiliser les fonctionnalités du logiciel et n'associaient pas de sémantique à leurs actions. Pour les aider à surmonter ces difficultés, nous avons intégré à Diagram un guidage contextuel lors de la création d'éléments du diagramme. Il se présente sous la forme d'un message indiquant l'action en cours de réalisation. Ce dispositif s'apparente aux rétroactions produites dans le logiciel Cabri-Géomètre lors d'opérations élémentaires [LABORDE & LABORDE 06]. D'après les auteurs, « ces messages soutiennent l'utilisateur dans l'usage du logiciel » et « restreignent la difficulté de la tâche ». Il s'agit ici d'une fonction facilitant le contrôle de son activité par l'apprenant, qui vise à éviter les erreurs de construction en cours d'action.

Un cas particulièrement intéressant est celui de la création d'une relation d'agrégation, de composition ou d'héritage. En effet, en environnement informatique, l'ordre dans lequel sont sélectionnées les deux classes à relier détermine l'orientation de leur relation. Les apprenants ne sont pas toujours conscients de cette contrainte, absente sur le papier, et cela est la source de nombreuses erreurs. L'affichage d'un message explicitant l'action en cours vise à réduire les erreurs de ce type.

### 3.2.2. Reformulation des éléments du diagramme

Le langage UML se base sur des éléments de modélisation et la sémantique de ces éléments. Or nous avons constaté que la sémantique véhiculée par l'orientation des relations entre les classes du diagramme n'est pas toujours perçue par les étudiants novices. Par exemple, ils ont tendance à confondre le diagramme signifiant « un bœuf est une sorte de bovin » avec celui signifiant « un bovin est une sorte de bœuf » car ces diagrammes sont visuellement identiques au sens de la flèche près.

Nous faisons l'hypothèse que confronter l'étudiant à une reformulation textuelle de son diagramme peut l'aider à comprendre sa signification et donc lui permettre de valider ou d'invalider ses constructions : la reformulation exerce une fonction de contrôle de l'activité (*monitoring*). Nous avons donc ajouté à Diagram un dispositif de reformulation textuelle des éléments du diagramme. Lorsque la souris pointe sur un élément du diagramme (classe ou relation), un message indiquant la sémantique de cet élément est affiché dans une bulle.



**Figure 1.** Affichage d'une reformulation durant l'élaboration du diagramme

### 3.3. Expérimentations

Nous avons mené des expérimentations à l'automne 2006 pour évaluer la manière dont les apprenants utilisent les fonctionnalités pédagogiques de Diagram. Elles ont porté sur quatre séances de TP de trois heures chacune, avec des étudiants de deuxième année de DEUST Informatique, Systèmes et Réseaux, la moitié

utilisant Diagram et l'autre un éditeur classique. Les actions réalisées à l'interface ont été enregistrées et les vidéos sont en cours d'analyse. Les premiers résultats indiquent que le mode d'interaction de Diagram favorise une plus grande attention à l'énoncé et un meilleur auto-contrôle chez ses utilisateurs. L'aide à la création d'éléments est peu utilisée, mais le dispositif de reformulation est bien exploité par les étudiants pour contrôler leurs productions. Les analyses doivent être poursuivies pour déterminer l'influence des outils et des aides sur les stratégies de modélisation.

#### 4. Conclusion

Nous avons présenté le modèle d'interaction de Diagram, un EIAH dédié aux diagrammes de classe UML qui organise la tâche en trois étapes, offre des outils de modélisation graphique originaux, propose des aides contextuelles, et nous avons montré comment ce modèle favorise l'activité métacognitive des apprenants. Les aides proposées sont génériques : elles ne tiennent pas compte de la validité du diagramme de l'apprenant. Nous envisageons de les compléter par des rétroactions plus spécifiques, basées sur un outil de diagnostic en cours de développement, qui signalerait à l'étudiant les erreurs ou les incohérences de son diagramme.

#### 5. Bibliographie

- [BAGHAEI & MITROVIC 05] Baghaei, N., Mitrovic, A., « COLLECT-UML: Supporting individual and collaborative learning of UML class diagrams in a constraint-based tutor », *Proceedings of Knowledge-Based and Intelligent Engineering Systems*, Khosla, R., Howlett, R. and Jain, L. (eds), 2005, p. 458-464.
- [FLAVELL 77] Flavell J. H., *Cognitive Development*, Engelwood Cliffs, N. J.:Prentice Hall Inc., 1997.
- [KOMIS et al. 03] Komis V., Avouris N., Dimitracopoulo A., Margarities M., « Aspects de la conception d'un environnement collaboratif de modélisation à distance », *Environnements Informatiques pour l'Apprentissage Humain*, Strasbourg, 15-17 avril 2003, Desmoulins, C., Marquet, P. et Bouhineau, D. (eds), 2003, p. 271-282.
- [LABORDE & LABORDE 06] Laborde, C., Laborde, J.-M., « Genèse et développement de Cabri-géomètre, environnement de géométrie dynamique », Grand Bastien, M., Labat, J.-M., Derycke, A. et Desmoulins, C. (eds) *Environnements Informatiques pour l'Apprentissage Humain*, London, UK: Hermès Science Publishing Ltd, 2006, p.345-374.
- [SHRAW & MOSHAM 95] Shraw G., Mosham D., « Metacognitive Theories », *Educational Psychology Review*, vol. 7, n°4, 1995.
- [SURaweera & MITROVIC 04] Suraweera. P., Mitrovic A., « An intelligent Tutoring System for Entity Relationship Modelling », *Int. J. Artificial Intelligent in Education*, vol. 14, n°3-4, 2004, p. 375-417.
- [THOLANDER & KARLGREN 02] Tholander J., Karlgren K., « Support for Cognitive Apprenticeship in Objet-Oriented Model Construction », *Proceedings of Computer Support for Collaborative Learning*, G. Stahl (ed), Boulder, Colorado, 2002.