Quelques patrons de raffinement pour le développement de diagrammes de classes UML

Boulbaba Ben Ammar, Mahamed Tahar Bhiri, Jeanine Souquières

Quelques patrons de raffinement pour le développement de diagrammes de classes UML

Boulbaba Ben Ammar [1,2], Mohamed Tahar Bhiri [2] et Jeanine Souquières [1]

[1] *LORIA - Université Nancy 2*
BP 239, F-54506 Vandœuvre lès Nancy
{Boulbaba.Ben-Ammar, Jeanine.Souquières}@loria.fr

[2] *Laboratoire MIRACL - Faculté des Sciences de Sfax*
B. P. 802 - 3018, Sfax - Tunisie
tahar_bhiri@yahoo.fr

RÉSUMÉ. *Le concept de raffinement est un élément clé dans les approches formelles pour le développement de logiciels et a donné lieu à de nombreux travaux dans lesquels la preuve de la correction entre les différents états de spécifications joue un rôle important. Nous proposons un cadre formel de définition des patrons de raffinement de diagrammes de classes UML. L'utilisation de la méthode B et de son mécanisme de raffinement permet la vérification de la correction des patrons à l'aide des outils supports. Nous illustrons notre proposition par la définition de quelques exemples de patrons et leur application au développement d'une étude de cas.*

ABSTRACT. *The concept of refinement is a key element in formal approaches for the development of software and gave place to many studies in which the correctness proof between the various states of a specification plays an important part. We propose a formal framework to define refinement patterns for UML class diagrams. The use of the B formal method and its refinement mechanism allows the verification of the correction of the patterns by the B support tools. We illustrate our proposition by the definition of some pattern exemples and their application to a case study development.*

MOTS-CLÉS : *Patron, raffinement, développement progressif, diagramme de classes, UML, B.*

KEYWORDS: *Patterns, refinement, progressive development, class diagram, UML, B.*

1. Introduction

Le modèle d'un système complexe ne peut être construit de façon monolithique, c'est-à-dire en une seule étape et son développement s'effectue par approximations successives [ABR 03]. Ces différentes approximations établissent entre elles une relation dite "de raffinement", largement étudiée dans les approches formelles. Habituellement, deux types de raffinements sont distingués, les raffinements horizontaux et les raffinements verticaux. Le raffinement horizontal est l'occasion d'introduire de nouvelles vues, d'introduire plus de détails sur les objets modélisés. Chaque ajout de détails à un modèle entraîne un nouveau modèle qui doit rester cohérent avec le précédent. Si c'est bien le cas, on dit que le deuxième modèle raffine le premier. Quant au raffinement vertical, il permet de passer d'un niveau abstrait vers un niveau plus concret, avec en particulier réduction de l'indéterminisme et affaiblissement des préconditions des opérations. Le modèle concret traduit la réalisation finale du modèle abstrait.

Le concept de raffinement est un élément central dans les méthodes formelles, comme la méthode B [ABR 96, POT 03] ou l'approche orientée objets Perfect Developer [CRO 03], avec vérification de la correction : un raffinement est correct si ce que l'on attend du modèle abstrait est vérifié par le modèle plus concret. Dans ces approches, le raffinement entre deux modèles (abstrait et raffiné) est défini formellement. Cette relation peut être prouvée mathématiquement par des outils associés à ces méthodes. Les travaux méthodologiques liés à l'utilisation du raffinement dans un processus incrémental de développement de systèmes complexes sont rares [GUY 02]. Les études de cas en B proposant un processus incrémental de développement basé sur le raffinement sont peu nombreuses et portent essentiellement sur des algorithmes distribués et des systèmes réactifs [CAN 03]. Il est difficile, à partir de ces études de cas, de tirer des règles universelles relatives à l'utilisation du raffinement.

Les démarches de développement incrémentales sont naturelles et pertinentes mais elles sont actuellement difficiles à mener avec UML. En effet, une définition rigoureuse du concept raffinement se heurte au cadre semi-formel induit par l'utilisation d'UML. La vérification des propriétés des modèles UML pose des problèmes. Certes, OCL [OMG 03] couplé à UML a permis le développement d'une approche de validation de tels modèles [GOG 03, ZIE 03]. Une telle approche est basée sur la génération automatique de plusieurs micro-univers d'objets (diagrammes d'objets) à partir des scénarios modélisés par des diagrammes de séquences. Ces scénarios traduisent des cas de tests issus des fonctionnalités souhaitées du système étudié. Dans le cas où le scénario ne satisfait pas une contrainte OCL (précondition, postcondition ou invariant) alors le test échoue et le diagramme de classes n'est pas validé. Cette technique de validation des modèles UML/OCL est basée sur l'exécution : elle est assimilée à une activité de test de modèles UML. Par conséquent, elle ne peut pas apporter la preuve de la correction des modèles UML.

Dans le cadre UML, le concept de raffinement a fait l'objet d'un certain nombre d'études. Les travaux effectués par et autour de Claudia Pons [PON 03, PON 04, PON 05, PON 06] ont donné naissance à des patrons de raffinement UML et au dé-

veloppement d'un outil appelé E-Platero supportant une activité de raffinement des diagrammes de classes UML. L'outil E-Platero propose une relation de raffinement équivalente à celle d'Object-Z [GRA 00]. Une telle relation de raffinement est implantée par la technique d'évaluation des modèles UML/OCL présentée précédemment. Shen et Low [LOW 05] proposent des règles de raffinement des relations UML (généralisation et association). Ces règles sont décrites au niveau méta-modèle - le niveau M2 de l'architecture de méta-modélisation à quatre niveaux- en proposant des stéréotypes.

Dans cet article, nous proposons un cadre de définition de patrons de raffinement de diagrammes UML. Un patron est décrit à l'aide d'un diagramme UML présentant deux niveaux d'abstraction, le niveau abstrait et le niveau raffiné, incluant les conditions d'application et les propriétés du modèle raffiné. Ce patron est ensuite formalisé en B en utilisant les règles de transformation systématique d'UML vers B [MEY 99, LED 01], permettant ainsi de vérifier la correction du raffinement à l'aide des outils supports [B4f]. Nous illustrons notre propos par la définition de quelques exemples de patrons et leur application pour le développement des premières étapes d'un système de contrôle d'accès à un ensemble de bâtiments [ABR 98]. L'accès s'effectue sur la base d'une carte que chaque personne autorisée est censée posséder. Des contrôleurs sont installés aux différents points d'accès permettant d'identifier les personnes munies de cartes et de gérer l'ouverture et la fermeture des points d'accès.

L'article est constitué comme suit. La section 2 décrit quelques patrons de raffinement. La section 3 décrit les premières étapes de développement de l'étude de cas du contrôle d'accès à un bâtiment par instanciation de ces patrons. La section 4 propose une discussion sur ces patrons et différentes extensions possibles.

2. Patrons de raffinement

La construction incrémentale d'un diagramme de classes repose sur l'introduction progressive de ses différentes éléments, voir leur décomposition. Nous proposons quelques patrons permettant de décrire différents raffinement d'associations et de classes : par décomposition chaînée, par décomposition d'agrégat et par spécialisation.

Seul le patron de raffinement par décomposition chaînée est décrit complètement avec la spécification B correspondante. Pour les autres patrons, nous donnerons uniquement les schémas UML et les propriétés du niveau raffiné.

2.1. *Patron de raffinement par décomposition chaînée*

(a) Description UML

Le raffinement par décomposition chaînée d'une association [LOW 05] correspond au cas où une association est décomposée en deux associations avec introduction d'une classe intermédiaire. Sa représentation graphique est donnée Figure 1.

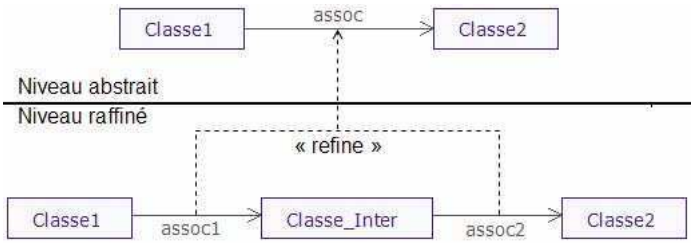

Figure 1. *Patron de raffinement par décomposition chaînée*

(b) Condition d'application

$$assoc \in classe1 \leftrightarrow classe2 \quad [1]$$

(c) Propriétés du raffinement

Les propriétés identifiées ci-dessous concernent les associations unidirectionnelles :

– Les classes qui représentent les extrémités de l'association abstraite doivent correspondre aux classes des extrémités de la chaîne crée par le raffinement, tout en respectant le sens de l'association. Sachant qu'une association UML est modélisée par une relation B [MEY 99], cette contrainte est traduite en B par l'expression [1] :

$$dom(assoc) = dom(assoc1) \wedge ran(assoc) = ran(assoc2) \quad [2]$$

– La classe intermédiaire doit correspondre à l'extrémité entrante d'une association et à l'extrémité sortante d'une autre, pour compléter la chaîne [2] :

$$ran(assoc1) = dom(assoc2) \quad [3]$$

(d) Correction du patron

La modélisation B de ce patron est présentée Figure 2. La machine abstraite *System*, correspond à la transformation en B du niveau abstrait du diagramme de classes, et la machine *System_ref* à celle du niveau raffiné. Les propriétés du raffinement sont introduites dans l'invariant de la machine *System_ref* appelé invariant de collage, permettant d'établir le lien entre machine abstraite et machine raffinée. Ces propriétés sont visualisées dans l'encadré.

La vérification de la correction du raffinement est effectuée à l'aide de B4free. Les obligations de preuves générées ont été vérifiées automatiquement.

(e) Discussion

Une association bidirectionnelle *AssocBi* [OMG 05] est sémantiquement équivalente à deux associations unidirectionnelles *AssocUni1 et AssocUni2* de sens inverses :

$$\boxed{AssocBi = AssocUni1 \cup AssocUni2 \wedge AssocUni1 = AssocUni2^{-1}}$$

MODEL
System
INCLUDES
Classe1, Classe2
VARIABLES
assoc
INVARIANT
$assoc \in classe1 \leftrightarrow classe2$
INITIALISATION
$assoc := \emptyset$
END

REFINEMENT
System_ref
REFINES
System
INCLUDES
Classe1, Classe2, Classe_Inter
VARIABLES
assoc1, assoc2
INVARIANT
$assoc1 \in classe1 \leftrightarrow classe\_Inter \wedge$
$assoc2 \in classe\_Inter \leftrightarrow classe2 \wedge$
$dom(assoc) = dom(assoc1) \wedge$
$ran(assoc) = ran(assoc2) \wedge$
$ran(assoc1) = dom(assoc2)$
INITIALISATION
$assoc1 := \emptyset || assoc2 := \emptyset$
END

Figure 2. *Modélisation en B*

Il en résulte que les propriétés du raffinement relatives à une association bidirectionnelle peuvent être définies par la conjonction des propriétés du raffinement relatives à deux associations unidirectionnelles de sens opposé.

Remarque. La définition et la vérification du patron de raffinement par décomposition chaînée d'une association proposée dans ce papier contient une seule classe intermédiaire appelée *Classe_Inter*. Nous avons généralisé ce patron à l'introduction de *n* classes intermédiaires.

2.2. *Patron de raffinement par décomposition d'agrégat*

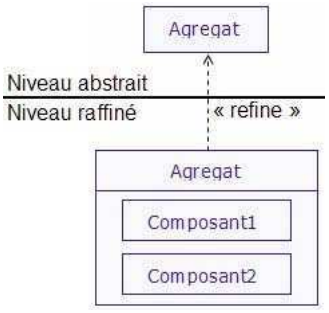

Figure 3. *Patron de raffinement par décomposition d'agrégat*

Ce patron permet de décomposer une structure composite appelée agrégat en plusieurs composants comme modélisé Figure 3. La relation entre l'agrégat et ses composants peut être matérialisée par une relation : de composition, d'agrégation ou d'hé-

ritage en fonction des exigences du système étudié et du contexte d'application de ce patron.

La propriété du raffinement est :

$$\begin{array}{l} Composant1 \subset Agregat \wedge \\ Composant2 \subset Agregat \wedge \\ Composant1 \cap Composant2 = \emptyset \end{array} \quad [4]$$

La matérialisation de la relation entre agrégat et ses composants correspond à un raffinement vertical. Ainsi, on peut proposer trois patrons d'implantation d'agrégat : par composition, par agrégation et par héritage. La Figure 4 illustre le patron d'implantation d'un agrégat par composition.

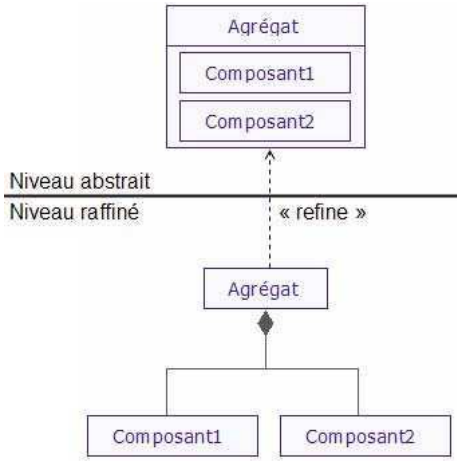

Figure 4. *Patron d'implantation d'agrégat par composition*

2.3. *Patron de raffinement par spécialisation*

Le raffinement par spécialisation d'une association consiste à ajouter des informations à une association existante : l'une des extrémités de l'association raffinée correspond à l'une des sous-classes de l'extrémité de la structure composite, voir Figure 5.

La condition d'application de ce patron est :

$$\begin{array}{l} assoc \in classe1 \leftrightarrow classe2 \wedge \\ Composant1 \subset Classe1 \wedge \\ Composant2 \subset Classe1 \wedge \\ Composant1 \cap Composant2 = \emptyset \end{array} \quad [5]$$

La propriété du raffinement de ce patron est :

$$assoc \in Composant1 \leftrightarrow Classe2 \quad [6]$$

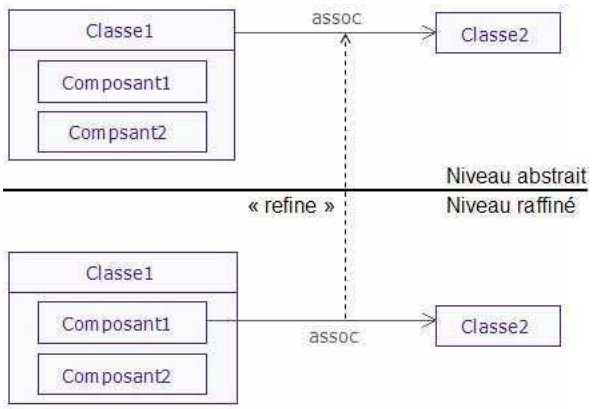

Figure 5. *Patron de raffinement par spécialisation*

[PON 03] a défini un patron de raffinement par spécialisation traitant à la fois le raffinement des classes et des associations.

3. Étude de cas

Nous décrivons le développement progressif d'un diagramme de classes de l'étude de cas considérée par application de patrons proposés. Pour chaque étape, nous décrivons informellement les propriétés prises en compte et la décision prise en termes de patron avec les paramètres requis. L'application du patron choisi donnera un nouvel état du diagramme de classes, état généré automatiquement par application du patron.

3.1. *Première spécification*

Nous nous intéressons à la spécification initiale du système en construisant un premier modèle très abstrait [ABR 98].

P1	Le modèle comprend des bâtiments ouverts à des personnes.

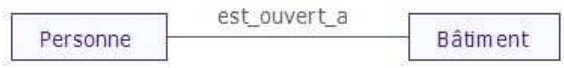

Figure 6. *Un premier diagramme de classes*

Nous initialisons le processus de développement à partir de la propriété *P1* du premier diagramme de classes proposé Figure 6. Il est constitué des deux classes *Bâtiment* et *Personne*, qui représentent respectivement l'ensemble de bâtiments et les groupes de personnes autorisées. Ces classes sont reliées par une association *est_ouvert_a*.

3.2. *Premier raffinement*

P2 L'entrée dans un bâtiment s'effectue à l'aide d'une carte magnétique
- chaque personne autorisée dispose d'une carte magnétique,
- une carte magnétique est identifiée par un lecteur de carte.

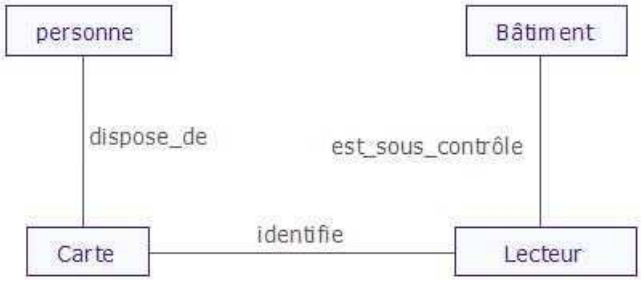

Figure 7. *Premier raffinement*

La propriété P2 apporte des informations supplémentaires sur l'association *est_ouvert_a*. Deux classes intermédiaires peuvent être introduites :

– la classe *Carte* associée à chaque personne,

– la classe *Lecteur* associée à chaque lecteur de cartes d'un bâtiment.

Ces classes sont reliées de la manière suivante : *Carte* est reliée à *Personne*, *Carte* est reliée à *Lecteur* et *Lecteur* est reliée à *Bâtiment*.

L'application du patron de raffinement par décomposition chaînée à l'association bidirectionnelle *est_ouvert_a* du diagramme de classes de la Figure 6 avec comme paramètres :

– les classes *Carte* et *Lecteur*,

– les associations :

- *dispose_de* entre *Personne* et *Carte*,
- *identifie* entre *Carte* et *Lecteur*,
- *est_sous_contrôle* entre *Lecteur* et *Bâtiment*,

génère le diagramme de classes de la Figure 7.

3.3. *Deuxième raffinement*

P3 Chaque lecteur de cartes est accompagné d'un voyant lumineux pour la visualisation de l'autorisation.

La propriété P3 fait apparaître la nécessité d'introduire une nouvelle classe, *Voyant* et de regrouper cette classe avec la classe *Lecteur* dans une super classe *Contrôleur*.

L'application du patron de raffinement par regroupement de sous-classes, non présenté dans ce papier, avec :

– la classe *Lecteur* comme paramètre d'entrée,
– les deux classes *Contrôleur* et *Voyant* comme paramètres de sortie,
– l'association *identifie* entre *Carte* et *Lecteur* comme paramètre de sortie

génère un nouvel état du diagramme de classes présenté Figure 8.

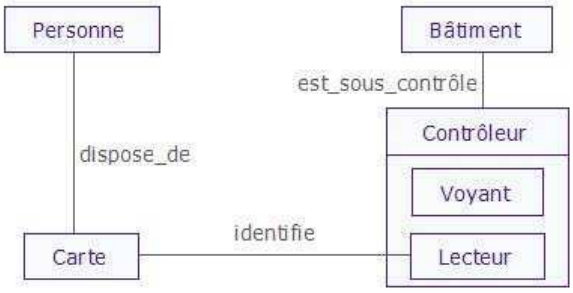

Figure 8. *Deuxième raffinement*

3.4. *Troisième raffinement*

P4	Chaque bâtiment est muni – de capteurs d'incendie, – de points d'accès qui sont sous le contrôle du contrôleur.

Par une instanciation des paramètres analogue à celle de la section 3.2, l'application du patron de raffinement par décomposition d'agrégat suivi de spécialisation, nous obtenons le diagramme de classes présenté Figure 9.

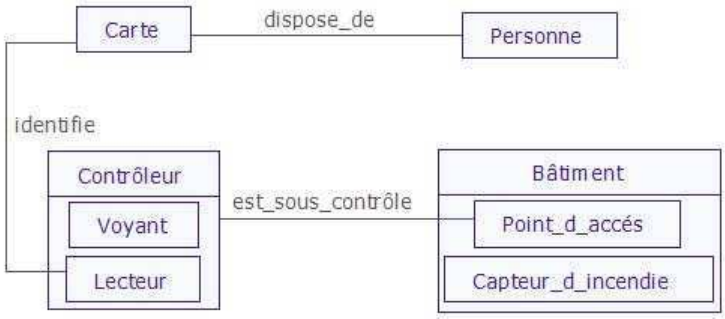

Figure 9. *Troisième raffinement*

3.5. *Autres raffinements*

Les deux classes *Lecteur* et *Voyant* de la structure composite *Contrôleur* font référence à des composants physiques qui n'ont d'existence que relativement au contrôleur. Ceci est également vrai pour les deux classes *Point_d_acces* et *Capteur_d_incendie* de la structure composite *Bâtiment*. En appliquant successivement sur ces deux structures composites le patron d'implantation d'agrégat par composition, nous obtenons le diagramme de classes présenté Figure 10.

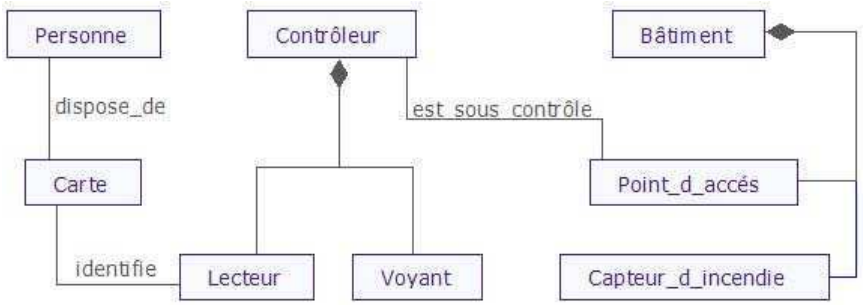

Figure 10. *Diagramme de classes du contrôle d'accès à un bâtiment*

4. Conclusion et perspectives

Nous avons défini un ensemble de patrons de raffinement de base relatifs aux différents concepts des diagrammes de classes UML. Ces patrons ont été définis formellement avec leur expression en B, permettant entre autre de vérifier ses conditions d'application (par l'intermédiaire d'un invariant sur l'état abstrait). La vérification de la correction de chaque patron est ainsi effectuée à l'aide des outils support. Nous avons montré comment les patrons de raffinement proposés peuvent être appliqués sur un exemple concret.

La question de la découverte des classes est centrale et difficile pour les concepteurs de logiciels orientés objet. La méthodologie objet apporte plusieurs réponses basées sur des approches différentes [MEY 00] : linguistique, cas d'utilisation, type de données abstrait (TDA) et patron (patron de conception et analyse). De notre point de vue, les patrons de raffinement sont un guide intéressant permettant au concepteur de déterminer les classes. Mais la pertinence des classes trouvées demeure liée à la compétence du concepteur. Le gain apporté par l'application des patrons de raffinement vis-à-vis des approches existantes concerne essentiellement la vérification de la cohérence entre les différentes approximations du système modélisé.

Le premier modèle UML conditionne le processus incrémental de développement basé sur le raffinement. Le premier raffinement effectué par l'application d'un patron de raffinement fait la supposition que le modèle initial est cohérent. Celui-ci peut être obtenu en ajoutant des contraintes OCL. Le modèle initial UML/OCL peut être traduit et prouvé en B.

De manière générale, la composition des patrons, qu'ils soient de conception, d'analyse ou de raffinement, est un problème ouvert. En partant d'un contexte donné, il faut choisir le "meilleur" patron applicable, et on répète le même processus sur le nouveau contexte obtenu. Cependant, pour les patrons de raffinement présentés dans ce travail, nous proposons l'ordre d'application suivant : patron de raffinement par décomposition chaînée, patron de raffinement par décomposition d'agrégat et patron de raffinement par spécialisation. Nous travaillons actuellement à la définition de patrons dits stratégiques permettant la composition de patrons de base.

L'application des patrons de raffinement proposés dans ce travail permet l'obtention d'une structure macroscopique (voir figure 10) du système étudié : les notions jugées comme classes et les relations entre ces classes. Mais ces classes sont "vides" dans le sens où elles ne contiennent ni partie statique, ni partie dynamique. Pour faire face à cette situation, il est nécessaire de compléter ces patrons de raffinement orientés relations (association, agrégation, composition et héritage) par de nouveaux patrons de raffinement orientés classes (raffinement des méthodes, des attributs). De même, nous définissons des patrons permettant de faire évoluer en parallèle plusieurs diagrammes UML : diagramme de classes et diagramme de machines d'état-transitions, diagrammes de classes et diagrammes de séquences.

5. Bibliographie

[ABR 96] ABRIAL J., *The B Book - Assigning Programs to Meanings*, Cambridge University Press, 1996, ISBN 0521496195.

[ABR 98] ABRIAL J., « Étude Système : méthode et exemple », rapport, Octobre 1998, ClearSy SYSTEM ENGINEERING.

[ABR 03] ABRIAL J.-R., « B : passé, présent, futur. », *Technique et Science Informatiques*, vol. 22, n° 1, 2003, p. 89-118.

[B4f] B4free, http ://www.b4free.com/.

[CAN 03] CANSELL D., « Assistance au développement incrémental et à sa preuve », PhD thesis, Université Henri Poincaré (Nancy I), Apr 2003.

[CRO 03] CROCKER D., « Perfect Developer : A tool for Object-Oriented Formal Specification and Refinement », *FM 2003 : the 12th International FME Symposium*, , 2003.

[GOG 03] GOGOLLA M., BOHLING J., RICHTERS M., « Validation of UML and OCL Models by Automatic Snapshot Generation », STEVENS P., WHITTLE J., BOOCH G., Eds., *UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference, San Francisco, CA, USA, October 2003, Proceedings*, vol. 2863 de *LNCS*, Springer, 2003, p. 265–279.

[GRA 00] GRAEME S., *The Object-Z specification language*, Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[GUY 02] GUYOMARD M., « Spécification et raffinement en B : deux exemples pédagogiques », *ZB2002 4th International B Conference, Education Session Proceedings*, , 2002.

[LED 01] LEDANG H., « Automatic Translation from UML Specifications to B », *ASE '01 : Proceedings of the 16th IEEE international conference on Automated software engineering*, IEEE Computer Society, 2001, page 436.

[LOW 05] LOW W. L., « Using the Metamodel Mechanism to Support Class Refinement », *ICECCS '05 : Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems*, IEEE Computer Society, 2005, p. 421–430.

[MEY 99] MEYER E., SOUQUIÈRES J., « A Systematic Approach to Transform OMT Diagrams to a B Specification », *FM '99 : Proceedings of the Wold Congress on Formal Methods in the Development of Computing Systems-Volume I*, Springer-Verlag, 1999, p. 875–895.

[MEY 00] MEYER B., *Conception et programmation orientées objet*, Éditions Eyrolles, 2000, ISBN 2-212-09111-7.

[OMG 03] OMG, « OCL 2.0 OMG Final Adopted Specification », Octobre 2003.

[OMG 05] OMG, « UML 2.0 Superstructure Specification », August 2005.

[PON 03] PONS C., PEREZ G. A., GIANDINI R., KUTSCHE R., « Understanding Refinement and Specialization in the UML », *2nd International Workshop on MAnaging SPEcialization/Generalization HIerarchies (MASPEGHI 2003)*, October 2003.

[PON 04] PONS C., GIANDINI R. S., PÉREZ G., PESCE P., BECKER V., LONGINOTTI J., CENGIA J., « PAMPERO : Precise Assistant for the Modeling Process in an Environment with Refinement Orientation. », *UML Satellite Activities*, 2004, p. 246-249.

[PON 05] PONS C., « On the Definition of UML Refinement Patterns », *2nd MoDeVa workshop, Model design and Validation, Montego Bay, Jamaica*, October 2005.

[PON 06] PONS C., GARCIA D., « An OCL-Based Technique for Specifying and Verifying Refinement-Oriented Transformations in MDE. », *MoDELS*, 2006, p. 646-660.

[POT 03] POTET M. L., « Spécifications et développements structurés dans la méthode B », *TSI, Technique et science informatique*, vol. 22, n° 1, 2003, p. 61–88.

[ZIE 03] ZIEMANN P., GOGOLLA M., « Validating OCL Specifications with the USE Tool - An Example Based on the BART Case Study », ARTS T., FOKKINK W., Eds., *Proc. 8th Int. Workshop Formal Methods for Industrial Critical Systems (FMICS'2003)*, vol. 80, ENTCS, Elsevier, 2003.