Une démarche outillée pour spécifier formellement des patrons de conception réutilisables

Sandrine Blazy, Frédéric Gervais, Régine Laleau

Institut d'Informatique d'Entreprise
Laboratoire CEDRIC
18 allée Jean Rostand
91 025 Évry
{blazy,gervais,laleau}@iie.cnam.fr

RÉSUMÉ. *Cet article introduit une démarche outillée pour réutiliser des patrons de conception formellement spécifiés. Réutiliser un tel patron consiste soit à l'instancier soit à le composer avec d'autres patrons, soit à l'étendre. Cet article définit trois niveaux de composition : la juxtaposition, la composition à l'aide de liens entre patrons et l'unification.*

ABSTRACT. *This paper describes an approach for reusing design patterns that have been formally specified. Reusing such a pattern means instantiating it or composing it with other patterns or extending it. Three levels of composition are defined : juxtaposition, composition with inter-patterns links and unification. This paper shows through examples how to define specification patterns in B, how to reuse them directly in B, and also how to reuse the proofs associated to specification patterns.*

MOTS-CLÉS :*patron de conception, réutilisation, spécification formelle, B.*

KEY WORDS: *design pattern, reuse, formal specification, B.*

La réutilisation de composants pour développer un logiciel a d'abord été employée lors de la phase d'implémentation du logiciel. Elle a également été adaptée à la phase de conception. Ainsi, il est aujourd'hui courant de réutiliser des patrons de conception exprimés en UML pour concevoir un nouveau système d'information. De nombreux patrons de conception applicables à divers domaines ont été définis. Bien que ces patrons soient connus de la plupart des concepteurs, ils n'ont pas de définition formelle précise.

D'un autre côté, les spécifications formelles sont de plus en plus utilisées dans l'industrie et il devient intéressant de réutiliser en partie ces spécifications dans de nouveaux projets. Réutiliser une spécification formelle signifie d'abord définir une notion de composant de spécification formelle (que nous appelons patron de spécification formelle) et aussi la façon de combiner ces composants lors de la construction d'une nouvelle application.

Nous avons utilisé le langage B pour spécifier formellement la notion de patron de conception [FOW 97, GAM 95] que nous avons appelé patron de spécification. Nous avons également spécifié en B diverses façons de réutiliser des patrons de spécification. Réutiliser un patron de spécification consiste soit à l'instancier, soit à le composer avec d'autres patrons de spécification, soit à l'étendre. L'instanciation est simplement réalisée en B par des inclusions entre spécifications et des redéfinitions de noms de variables et d'opérations. Nous avons défini trois niveaux de composition, en fonction des liens qui existent entre les patrons composés : juxtaposition, composition à l'aide de liens entre patrons et unification. Deux patrons sont juxtaposés pour former un troisième patron lorsqu'il n'y a aucun lien entre eux dans le troisième patron. Il est également possible de composer deux patrons en définissant des liens entre eux (par exemple en définissant une bijection entre classes des deux patrons composés). Enfin, une troisième façon de composer des patrons est d'unifier des éléments partagés par les différents patrons. Étendre un patron permet de lui ajouter de nouveaux éléments ou de nouvelles opérations, ou encore de modifier ou supprimer certains éléments ou opérations. Nous avons utilisé la notion de raffinement de B pour étendre des patrons.

Nous avons choisi le langage B d'une part pour que des utilisateurs connaissant déjà la méthode B n'aient pas besoin d'apprendre un nouveau langage s'ils veulent construire une application en réutilisant des patrons de spécification, et d'autre part pour bénéficier des outils de vérification des spécifications afin de valider notre démarche. Ainsi, un concepteur pourrait non seulement réutiliser des patrons de spécification, mais également des preuves concernant ces spécifications.

Dans un état de l'art sur la réutilisation de patrons à l'aide de méthodes formelles [GER 02], nous avons comparé différentes approches fondées soit sur la définition d'un langage de spécification dédié à la réutilisation [EDE 98, LAU 99], soit sur un langage existant [FLO 00, MAR 00]. Les approches étudiées proposent toutes d'instantier les patrons de spécification pour les réutiliser. Par contre, la composition de patrons de spécification n'est jamais formellement définie et la plupart des approches ne la considèrent pas. Nous n'avons trouvé aucune approche qui permette de spécifier formellement des patrons de spécifications (et de vérifier de telles spécifications à l'aide d'outils), et qui propose ensuite différentes façons de combiner automatiquement les patrons (et d'obtenir ainsi des combinaisons valides).

Nous avons donc proposé une démarche qui repose sur la méthode B afin d'automatiser le plus possible la réutilisation de patrons de spécification [BLA 03]. Cette démarche a été validée sur différents exemples de spécifications par réutilisation de patrons, les spécifications de nos exemples ayant été entièrement prouvées à l'aide de l'Atelier B [BLA 02].

La figure 1 montre un exemple de conception par réutilisation de patrons. Deux patrons (*Composite* et *Resource Allocation*) ont été réutilisés pour produire le

schéma UML présenté dans la figure 1. Ce schéma a été obtenu en suivant le processus suivant :

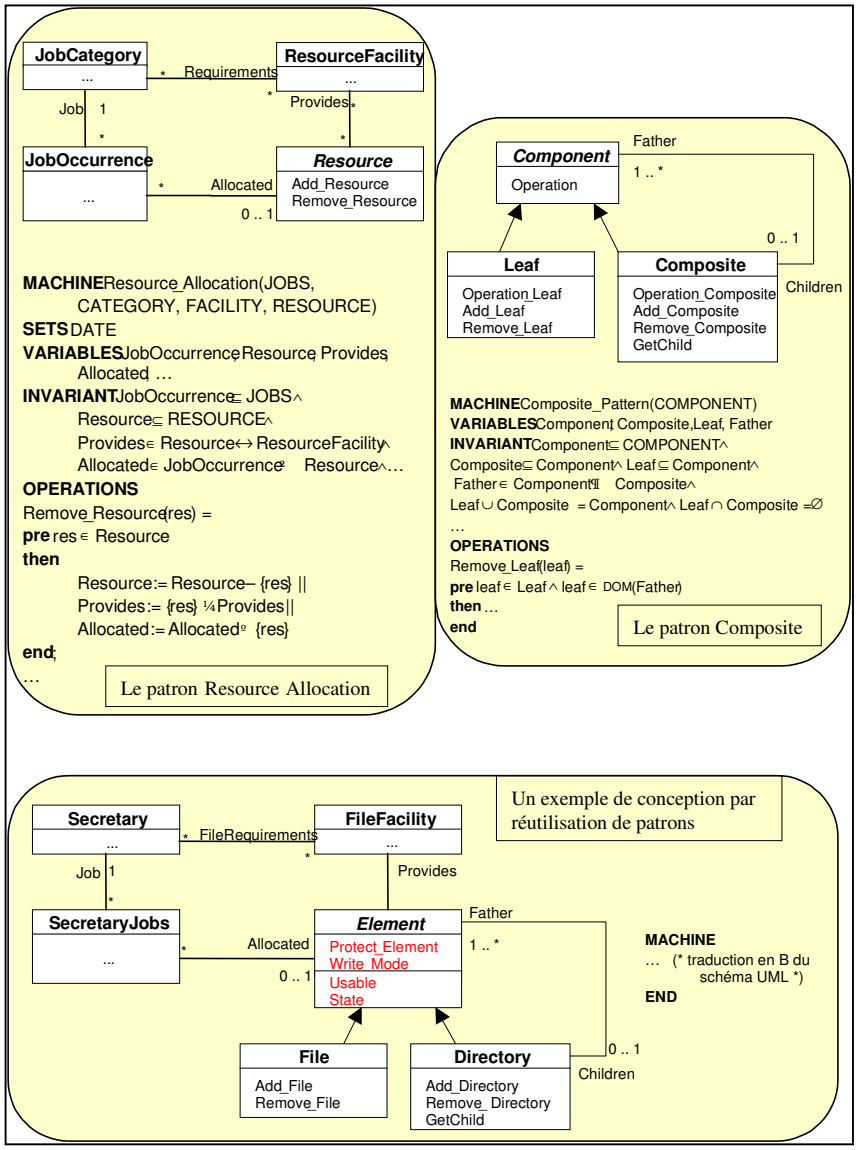

Figure 1. *Un exemple de réutilisation de patrons de spécification*

- Instanciation de chacun des deux patrons. Par exemple, dans le patron *Composite* qui sert à décrire dans des structures arborescentes des objets composés d'autres objets, *Component* a été instancié en *Element*, *Leaf* a été instancié en *File* et *Composite* a été instancié en *Directory*. Des opérations ont également été instanciées. Par exemple, *Add_Leaf* a été instanciée en *Add_File*. On génère ainsi deux nouveaux patrons instanciés que nous avons omis dans la figure pour ne pas la surcharger.
- Composition des deux patterns instanciés. Dans notre exemple, les deux patrons sont composés par unification : aucun lien n'a été ajouté entre les deux patrons et dans les patrons instanciés, l'instance de *Resource* a été unifiée avec celle d'*Element*. Les opérations du résultat de la composition ont elles-mêmes été redéfinies. Par exemple l' opération *Remove_File* est le résultat de la composition de *Remove_Resource* et *Remove_Leaf.*
- Extension. Dans l'exemple, des caractéristiques sont rajoutées à *Element* afin de prendre en compte les différents modes de protection d'un fichier ou d'un répertoire. L'extension permet également de modifier des opérations ou d'en définir de nouvelles.

La figure 1 montre également des extraits de traduction en B qui ont été générées automatiquement à partir des différents schémas UML obtenus durant le processus de conception.

Du point de vue de l'utilisateur, pour concevoir son application l'utilisateur a sélectionné les deux patrons initiaux dans une bibliothèque de patrons exprimés à l'aide de schémas UML. Il a ensuite décrit les étapes de son processus de conception et a finalement obtenu une spécification B du schéma UML final. Les traductions en B des schémas intermédiaires peuvent également être consultées, mais cela n'intéresse pas forcément l'utilisateur.

Concernant la réutilisation des preuves faites lors de la définition d'un patron, les opérateurs d'instanciation et de juxtaposition engendrent des spécifications B pour lesquelles aucun travail de preuve supplémentaire n'est nécessaire. Les deux autres opérateurs de composition génèrent des obligations de preuve que l'Atelier B sait prouver automatiquement, c'est-à-dire sans interaction avec l'utilisateur. L'extension d'un composant génère de nouvelles obligations de preuve relatives aux opérations ayant été ajoutées ainsi qu'au raffinement des opérations ayant été modifiées. Ces preuves sont à faire car elles n'ont pas été faites lors de la définition des patrons puisqu'elles concernent uniquement des opérations qui n'existaient pas dans le patron d'origine. Notre démarche de réutilisation de patrons permet donc de réutiliser également les preuves associées aux patrons réutilisés.

En conclusion, les contraintes liées à l'utilisation de B sont finalement faibles comparées aux avantages que nous apporte cette méthode outillée. La contrainte la plus embarrassante est la nécessité de produire la spécification en un seul module B

quel que soit le patron défini ou réutilisé (et non pas un ensemble de modules de taille plus raisonnable).

Nous développons actuellement un outil automatisant la réutilisation des patrons de spécification. Nous étudions également la formalisation des différents moyens de combinaison de patrons afin de définir précisément les contraintes portant sur les opérateurs de composition et d'extension de patrons. En particulier, la redéfinition des opérations résultat de ces opérateurs n' est possible que sous certaines conditions. Nous souhaitons spécifier formellement en B les différentes façons de réutiliser les patrons que nous avons définies. Nous comptons de plus prendre en compte dans notre démarche la réutilisation de plusieurs instances d'un même patron. À plus long terme, nous souhaitons nous intéresser à la notion de réutilisation de preuve associée à une spécification formelle, qui n'a pour l'instant jamais été étudiée dans le contexte de la méthode B. Enfin, nous souhaitons appliquer notre démarche sur de nouveaux exemples, et enrichir ainsi notre bibliothèque de patrons.

Bibliographie

[BLA 03] BLAZY S., GERVAIS F., LALEAU R.“ Reuse of specification patterns with the B method ”, *Conférence internationale ZB'2003*, Lecture Notes in Computer Science 2651, Springer-Verlag, juin 2003, Turku, Finlande.

[BLA 02] BLAZY S., GERVAIS F., LALEAU R., “ Un exemple de réutilisation de patterns de spécification avec la méthode B ”, rapport de recherche CEDRIC 395, Paris, 2002.

[EDE 98] EDEN A., HIRSHFELD Y., YEHUDAI A. “ LePUS – a declarative pattern specification language ” rapport technique 326/98, Université de Tel Aviv, 1998.

[FLO 00] FLORES A., REYNOSO L., MOORE R. “ A formal model of object-oriented design and GoF design patterns ” rapport technique 200, UNU/IIST, Macau, 2000.

[FOW 97] M.FOWLER “ Analysis pattern : reusable object models ” Addison-Wesley, 1997

[GAM 95] GAMMA E., HELM R., JOHNSON R, VLISSIDES J. . “ Design patterns : elements of reusable object-oriented software ” Addison-Wesley, 1995.

[GER 02] GERVAIS F. “ Réutilisation de composants de spécification en B ”, rapport de stage de DEA, Université d'Évry, 2002, http://cedric.cnam.fr/PUBLIS/RC394.ps.gz

[LAU 99] LAU Y., ORNAGHI M. “ OOD frameworks in component-based software development in computational logic ” Conférence internationale LOPSTR'98, Lecture Notes in Computer Science 1559, Springer-Verlag, 1999, pp.101-123.

[MAR 00] MARCANO-KAMENOFF R., LEVY N., LOSAVIO F. “ Spécification et spécialisation de patterns en UML et B ” *Langages et modèles à objets, LMO'2000*.