DEVELOPPEMENT ORIENTE POINT DE VUE : IMPLANTATION DE LA RELATION DE VISIBILITÉ, APPROCHE PAR LE PATRON D'ANALYSE ROLE

Hair Abdellatif

Faculté des Sciences et Techniques
B.P. 523 BENI MELLAL,
MAROC
a.hair@fstbm.ac.ma

RÉSUMÉ. Le patron d'analyse Rôle a été largement utilisé et abordé pour proposer des solutions génériques conceptuelles indépendantes d'un langage donné et illustrer des solutions astuces de programmation dans des langages cibles. A cet effet, l'objectif de ce travail est de proposer une nouvelle implantation de la relation de visibilité de la méthode orientée point de vue VBOOM (View Based Object Oriented Method) à l'aide de patron d'analyse Rôle. L'implantation proposée a bien facilité la génération du code multi-cibles à partir d'un développement orienté point de vue.

ABSTRACT. The Role pattern has been used extensively to propose independent conceptual generic solutions of a given language and to illustrate the solutions tricks of programming in targets languages. For that, the aims of this work is to propose a new implantation of the visibility relationship of the viewpoint-oriented method VBOOM (View Based Object Oriented Method) in using the Role pattern. The proposed implantation encouraged the multi-targets generation code from a viewpoint-oriented development.

MOTS-CLÉS : Implantation, Relation de visibilité, Patron Rôle, Méthode orientée objet, Vue et Point de vue.

KEY WORDS: Implantation, visibility relationship, Role pattern, Object-oriented method, View and Viewpoint.

1. Introduction

Les concepts de vue/point de vue, rôle, état, perspective... ont été étudiés et utilisés avec des approches variées dans la conception de systèmes complexes. On retrouve ainsi ces concepts dans les bases de données orientées objet (OO), les méthodes de conception OO, les systèmes de représentation de connaissances, etc. Au niveau de l'analyse/conception, l'un des résultats les plus aboutis est celui de VBOOM (View Based Object Oriented Method) [3,8]. Le principal apport de VBOOM est l'ajout au modèle objet d'une relation appelée "relation de visibilité". Cette relation permet d'exprimer le fait qu'une "entité" peut être considérée sous des "angles" multiples correspondant aux points de vue des utilisateurs concernés. Dans le cas d'exemple de gestion d'une médiathèque présenté ci-dessous, la médiathèque peut être appréhendée sous l'angle des prêts, des exemplaires d'ouvres, des comptes adhérents, etc.

La prise en compte le standard UML (Unifed Modeling Langage) [9,13] dans la méthode VBOOM en utilisant ses possibilités de spécialisation et d'extension [5,6] a donné lieu à l'élaboration de la méthode U_VBOOM qui intègre les concepts orientés points de vue issus de VBOOM et les artefacts d'UML. Pour compléter ce travail de standardisation et pour que U_VBOOM cible les langages objets du marché comme C++, Java, etc. et non uniquement le langage VBOOL (View Based Object Oriented Langage) [11], nous proposons une implantation de la relation de visibilité et les concepts liés à l'aide du patron d'analyse rôle [4]. Ce patron a été largement abordé pour proposer des solutions génériques conceptuelles indépendantes d'un langage donné et illustrer des solutions et des astuces de programmation dans des langages cibles.

Dans cet article, nous rappelons en premier lieu les principes de la modélisation orientée point de vue et les différentes phases de développement de la démarche U_VBOOM (section 2). Dans la section 3, nous présentons notre approche pour implanter la relation de visibilité et les concepts liés, avant de conclure sur les bénéfices de l'approche et les travaux en cours (section 4).

Durant tout ce papier nous illustrons notre propos à l'aide de l'exemple simplifié de gestion d'une médiathèque. Le cahier des charges de cet exemple est détaillé dans [10]. Les différents utilisateurs concernés du système Médiathèque sont : le *bibliothécaire* qui gère les prêts, le *responsable adhérent* qui ajoute et retire des adhérents, et enfin le *responsable exemplaires* qui saisit les nouveaux exemplaires et retire ceux qui sont abîmés.

2. Modélisation orientée point de vue

2.1. Présentation du processus de développement orientée point de vue

La notion de vue/point de vue a été introduite dans la méthode VBOOM pour répondre initialement aux besoins des concepteurs de systèmes complexes. VBOOM s'applique en fait à tout système dont la modélisation nécessite la prise en compte des besoins de plusieurs types d'utilisateurs.

U_VBOOM est une méthode d'analyse/conception par objets basée sur les vues. Cette méthode offre un formalisme étendant celui d'UML et une démarche centrée utilisateur inspirée de la méthode VBOOM. U_VBOOM s'intègre dans un processus de développement de logiciels. Ce processus permet au modélisateur de passer progressivement et semi-automatiquement d'une analyse/conception orientée point de vue à un modèle global VBOOL. Ce modèle doit être ensuite traduit pour obtenir un modèle d'implémentation dans un langage OO (C++, JAVA, etc.). Le modèle obtenu, sera finalisé par le programmeur pour obtenir le code final du système qui sera compilé pour générer du code exécutable (Figure 1(a)).

2.2. La démarche de U_VBOOM

Le modèle de développement de la méthode U_VBOOM, repose sur la prise en compte d'un certain degré de décentralisation au niveau élaboration des modèles visuels (sous-systèmes associés aux points de vue). U_VBOOM suppose ensuite une phase centralisée permettant la synthèse de ces modèles pour produire le modèle global. Ce modèle peut évoluer pour intégrer de nouveaux sous-systèmes si des points de vue sont ajoutés au système. La démarche de U_VBOOM détaillée dans [5,6], peut être résumée par un processus d'analyse/conception de trois phases (Figure 1(b)).

2.2.1. Phase1 : Analyse global

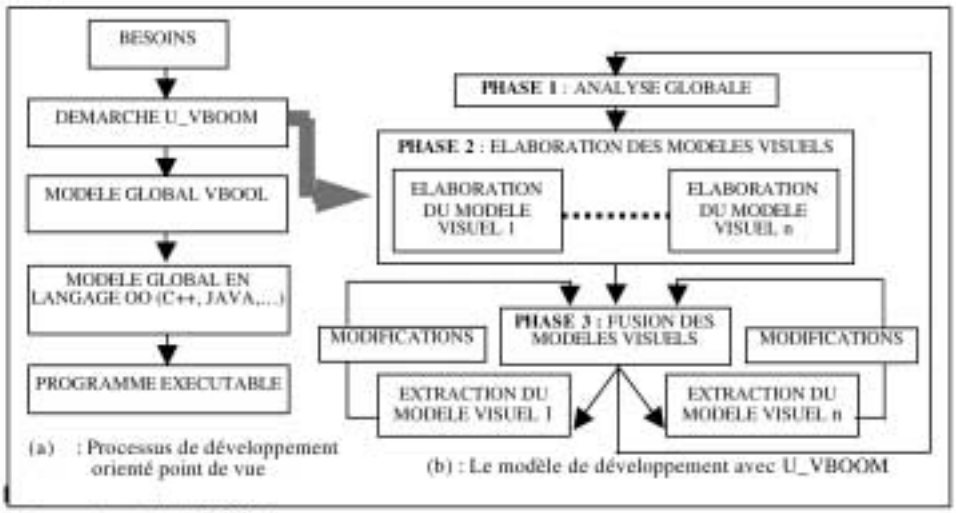
La phase d'analyse consiste à identifier les différents constituants du domaine du problème. Elle doit principalement permettre à fournir une description précise des différents besoins des utilisateurs du système. L'identification des vues du système est effectuée par une technique de description de cas d'utilisation [2,7]. Les classes et les relations entre ces classes sont identifiées scénario après scénario, au moyen d'objets qui, en collaborant, réalisent les cas d'utilisations.

2.2.2. Phase2 : Conception des modèles visuels (sous-systèmes)

La phase de conception de U_VBOOM doit permettre de concevoir les modèles visuels. Le découpage de l'espace de la solution du problème en modèles visuels permet de simplifier considérablement la conception. La conception des modèles visuels peut être menée en parallèle ou séquentiellement comme décrit dans [8]. Le résultat est un modèle UML par acteur.

2.2.3. Phase3 : Fusion des modèles visuels

Dans le cas d'une conception des modèles visuels menée en parallèle, l'élaboration de la solution générale du système nécessite une démarche pour fusionner les résultats partiels obtenus à l'issue de la conception des modèles visuels. Ainsi, U_VBOOM doit fournir aux concepteurs une démarche pour fusionner les différents diagrammes de classes partiels et les différentes interfaces de classes partielles des modèles visuels, en leur proposant des heuristiques pour gérer les conflits éventuels qui peuvent apparaître au cours de la fusion [5,6].



démarche de U_VBOOM

La Figure 2 représente le diagramme de classes global du système Médiathèque. Ce diagramme est obtenu après fusion des diagrammes de classes des modèles visuels Gestion des prêts, Gestion des adhérents et Gestion des exemplaires. Bien entendu, pour des raisons de simplicité, ce diagramme ne correspond pas à une solution complète et réaliste du problème.

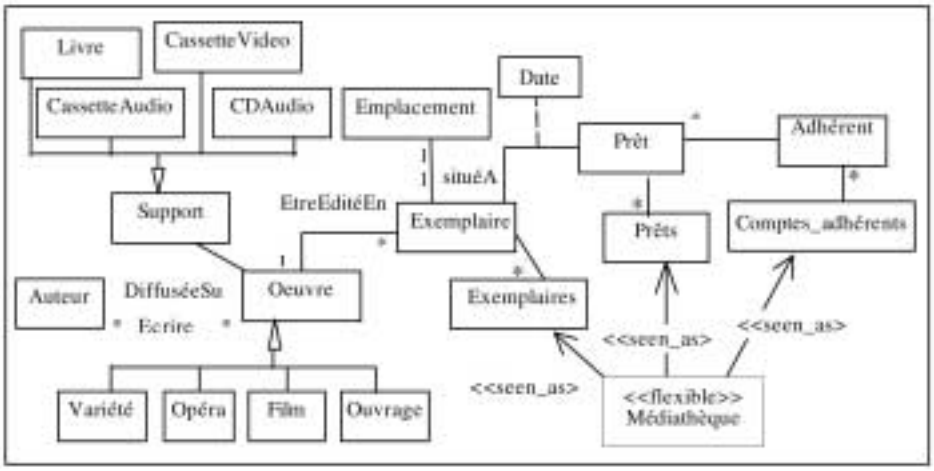


Figure 2. *Diagramme de classes de l'exemple de la MEDIATHEQUE (extrait)*

Le diagramme de classes généré par U_VBOOM comprend les classes : Comptes_Adhérents, Exemplaires et Prêts. Ces classes représentent les vues de la classe *Médiathèque* via la relation de visibilité (Figure 2). La classe *Médiathèque* est appelée "flexible" au sens où son utilisation peut varier selon le point de vue de l'utilisateur. Une classe flexible est une classe reliée par la relation de visibilité à un ensemble de vues. La représentation graphique de la classe flexible et de la relation de visibilité est illustrée respectivement par le symbole d'une classe UML avec le stéréotype <<flexible>>, et le symbole de la relation d'association unidirectionnelle d'UML avec le stéréotype <<seen_as>>.

3. Implantation de la relation de visibilité et les concepts liés

Pour rendre la méthode U_VBOOM facilement utilisable et pour qu'elle ne cible pas uniquement le langage VBOOL, nous proposons d'utiliser le patron d'analyse rôle [1,4] pour implanter la relation de visibilité afin de traduire le modèle global VBOOL issu de la méthode U_VBOOM en un modèle d'implémentation objet multi-cibles.

3.1. Implantation de la relation de visibilité

De nombreuses méthodes ont étendu leur notation pour intégrer le concept de rôle [12] de faite qu'il est un concept intuitif pour les utilisateurs. Le concept rôle est utilisé pour créer et représenter des collaborations d'objets. Le concept rôle spécifie alors les

différents comportements d'un objet S1 de type Médiathèque qui peuvent être identifiés sont : comptes des adhérents, exemplaires des œuvres et prêts des adhérents (Figure 3 (a)). Ces différents comportements représentent les différents rôles de S1 dans le système Médiathèque. Par ailleurs, le patron d'analyse rôle est utilisé lorsqu'un objet a des comportements variables selon le rôle joué. Si nous appliquons ce patron, la solution consiste à associer une classe Rôle abstraite à la classe Médiathèque et de lier chaque objet de type Rôle_Médiathèque à des instances des classes Rôles concrètes (Rôle1 : Exemplaires, Rôle2 : Prêts, Rôle3 : Comptes _Adhérents) (Figure 3 (b)).

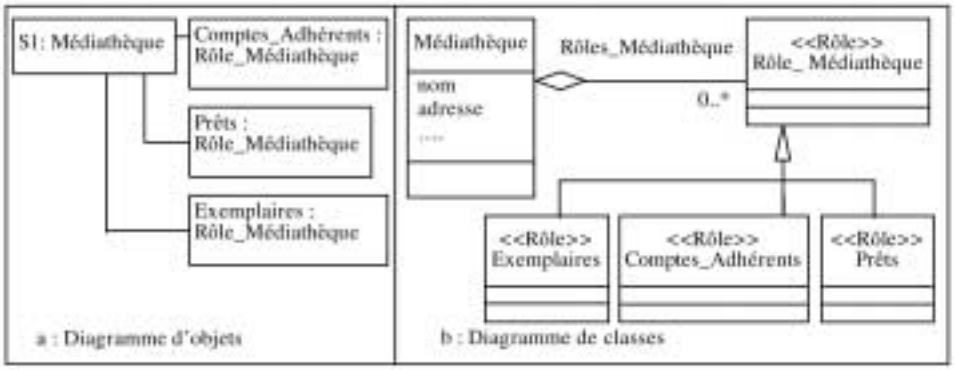


Figure 3. *Différents rôles identifiés dans le système Médiathèque*

Nous déduisons que l'application de la méthode U_VBOOM au système Médiathèque a permis d'identifier les 3 vues (Comptes_Adhérents, Exemplaires, Prêts) reliées à la classe flexible Médiathèque via la relation de visibilité. L'application du patron d'analyse Rôle au même système a permis aussi d'identifier les mêmes classes dont les 3 vues identifiées par U_VBOOM représentent les différents rôles joués par un objet de type Médiathèque. Nous pouvons conclure d'une part que les rôles au sens du patron d'analyse Rôle ne sont que des vues au sens de U_VBOOM et d'autre part que la relation de visibilité de VBOOM a été implantée par une relation d'agrégation, un ensemble de relation d'héritages et enfin une classe abstraite Rôle.

3.2. Concepts liés à la relation de visibilité

3.2.1. Déclaration des vues

Dans le langage VBOOL, les vues Exemplaires, Prêts et Comptes_Adhérents de la classe flexible Médiathèque sont déclarées dans une clause spécifiée par le mot clé "seen_as" (Figure 4 (a)). Dans la nouvelle implantation de la relation de visibilité, la déclaration des vues au sein de la classe multi-vues devient une liste de références

(*Vues_Médiathèques) sur l'ensemble de ses vues (Figure 4(b)). Par conséquent, la déclaration d'une classe flexible devient comme une déclaration d'une classe simple.

```
flexible class Médiathèque
        → déclaration des vues
....
seen_as Prets
seen_as Exemplaires
seen_as Comptes_Adherents
.....
end; - - class Médiathèque
```

(a) : Déclaration des vues dans une classe multi-vues Médiathèque en VBOOL

```
# include <Vues_Médiathèques.h>
# include <Prets.h>    // inclure la classe Prets
# include <Exemplaires.h>    // inclure la classe
Exemplaires
# include <Comptes_Adherents.h>
 ....                    // inclure la classe Comptes_Adherents
Class Médiathèque
public :
*Vues_Médiathèques     * VuesMédiathèques
  // La déclaration des vues devient une liste de références
 ....
End ; // fin classe
```

(b) : Déclaration des vues dans une classe multi-vues Médiathèque

Figure 4. *Déclaration des vues dans la classe multi-vues Médiathèque*

3.2.2. Instanciation de la classe flexible

Dans le langage VBOOL, l'instanciation d'une classe flexible consiste à préciser un point de vue particulier. Soit l'exemple *Bibliothécaire_mediatheque:Médiathèques*(Prêts, Exemplaires, Comptes_Adhérents), l'instance *Bibliothécaire_mediatheque* a le point de vue (Prêts, Exemplaires, Comptes_Adhérents). De faite que la déclaration d'une classe flexible devient comme la déclaration d'une classe simple, son instanciation selon un point de vue donné dans la nouvelle approche consiste à instancier *Bibliothécaire_mediatheque* comme un objet de type Médiathèque et à activer les vues composant ce point de vue. Pour cela, nous proposons d'ajouter une classe mère, nommé Vue, de toutes les classes représentant les vues du système. La classe Vue possède l'attribut booléen **etat_vue** qui exprime son état courant et les 3 méthodes **etat_vue()**, **activer_vue()** et **desactiver_vue()**. Ces méthodes permettent respectivement de retourner l'état active ou désactive de la vue, d'activer la vue et en fin de désactiver la vue (Figure 5).

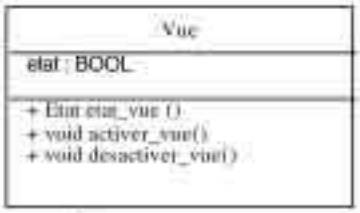
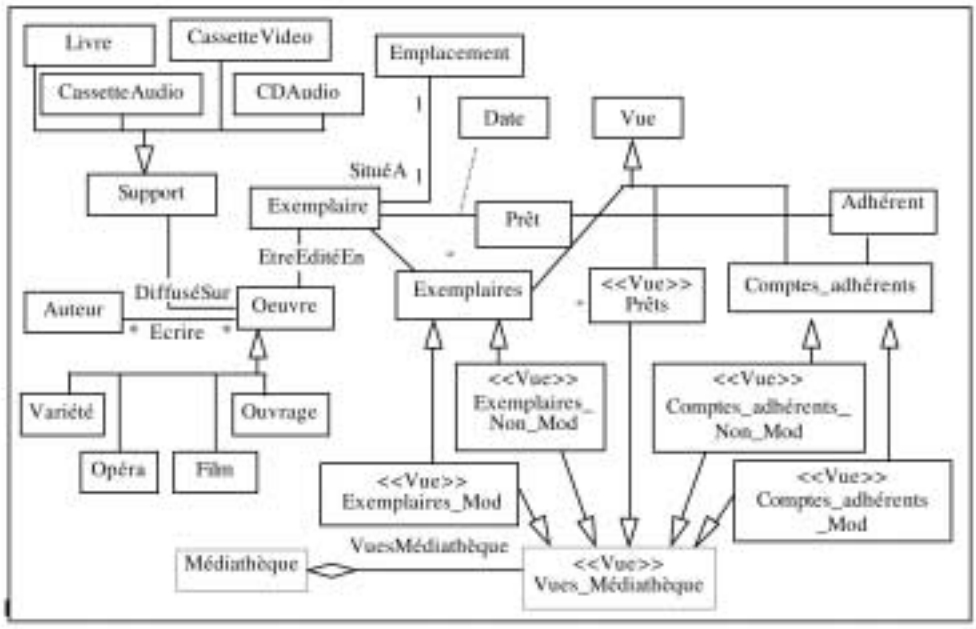


Figure 5. *Interface de la classe Vue en C++*

Ainsi, le diagramme de classes global traduit du système Médiathèque peut être représenté comme suit (Figure 6) :



4. Conclusion

L'approche présentée dans cet article a pour intérêt de générer le code multi-cibles à partir d'une analyse/conception orientée points de vue à l'aide de la méthode U_VBOOM. L'implantation de la relation de visibilité dans un langage OO comme C++, JAVA, etc., en utilisant le patron d'analyse Rôle, a bien facilité l'opération de génération de code.

Le travail décrit ici fait partie d'un projet plus vaste dont l'objectif est de définir une méthodologie de développement de composants objets multi-vues. Parmi les tâches qu'il reste à réaliser dans ce projet, nous pouvons citer :

- la définition de la notion de composant multi-vues, comme regroupement pertinent de classes éventuellement multi-vues,
- l'élaboration d'une base de patrons de conception supportant l'approche par points de vue,
- la réalisation d'un environnement support à U_VBOOM.

Bibliographie

[1] P. COAD : Object-oriented patterns ; *Communications of the ACM*, Vol. 35 - n°9, Septembre 1992.

[2] B. DANO : Spécifications dynamiques des besoins orientés objet : une démarche fondée sur les scénarios ; *Actes de INFORSID*, juin 1996.

[3] A. FINKELSTEIN, D. GABBAY, A. HUNTER, J. KRAMER, B. NUSEIBEH : Inconsistency Handling in Multi-Perspective Specifications ; *Actes ESEC'93*, Garmish-Paternkirchen (D), pp. 84-99, 1993.

[4] M. FOWLER : *Analysis Patterns – Reusable Object Models* ; Addison-Wesley, 1997.

[5] A. HAIR, A. KRIOUILE , B. COULETTE : Une méthode d'analyse et de conception orientée objet intégrant UML et le concept de point de vue. *Actes Conf. internationale ICSSEA'2001*, vol 3, Paris, France, 4-6 décembre, 2001.

[6] A. HAIR, A. KRIOUILE , B. COULETTE : Un processus d'analyse et de conception unifié basé sur le concept de point de vue. *Actes de CARI'02*, Yaoundé, Cameroun, octobre, 2002.

[7] I. JACOBSON, M. CHRISTERSON, P. JONHSON, G. OVERGAARD : *Object Oriented Software Engineering, A Use Case Driven Approach* ; Addison Wesley, 1992.

[8] A. KRIOUILE : VBOOM, une méthode d'analyse et de conception par objet fondée sur les points de vue ; Thèse d'état, faculté des sciences de Rabat, 1995.

[9] P. KRUTCHEN : *The Rational Unified Process - An Introduction* ; Addison Wesley, 2000.

[10] N. LOPEZ, J. MIGUEIS, E. PICHON : *Intégrer UML dans vos projets* ; Edition Eyrolles, 1998.

[11] S. MARCAILLOU : Intégration de la notion de points de vue dans la modélisation par objets ; Le langage VBOOL ; Thèse de l'université Paul Sabatier de Toulouse, 1995.

[12] G. MAUGHAN, B. DURNOTA MON : *An Object Relationship Model Incorporating Roles, Classification*, Publicity And Assertions, 1995.

[13] OMG : Unified Modeling Language (UML), version 1.4 ; OMG Document formal/2001-09-07, septembre, http://www.omg.org/cgi-bin/doc?formal/01-09-67.