U_VBOOM : une méthode d'analyse et conception unifiée orientée objet basée sur le concept de point de vue

Hair Abdellatif
Faculté des Sciences et Techniques
B.P. 523 Beni-Mellal, Maroc
E-Mail : a.hair@fstbm.ac.ma
Sbihi Boubker*, *belangour Abdessamad ***ENSIAS***
B.P. 713, Agdal Rabat Maroc
* *E-Mail : Sbihi7@Hotmail.com* ** *E-Mail : belangour@Hotmail.com*

1- Introduction

De nos jours, chacun s'accorde à reconnaître l'intérêt de prendre en compte l'utilisateur - au sens large du terme - le plus tôt possible dans le développement des systèmes complexes. Le concept de point de vue est un moyen pertinent pour mettre en œuvre cette approche. Parmi les bénéfices de l'approche, on peut citer l'utilisation d'un modèle unifié avec points de vue multiples au lieu de plusieurs sous-modèles interdépendants et souvent incohérents, l'instanciation par point de vue, le changement dynamique de point de vue, la gestion de la cohérence entre sous-modèles, etc. [3][6].

Les concepts de vue et de point de vue ont été étudiés dans plusieurs domaines liés au traitement de l'information : bases de données, représentation des connaissances, analyse et conception, langages de programmation, outils de Génie logiciel, etc. Dans l'approche objet, ces concepts ont été employés sous différents noms : vue, point de vue, rôle, perspective, aspect, sujet, morceau, critère, etc. [6][11][1]. Dans ce cadre, plusieurs travaux de recherche ont été menés pour introduire le concept de vue et point de vue dans la modélisation par objets dans un contexte de Génie Logiciel afin d'apporter une plus grande souplesse, flexibilité et rigueur. Au niveau des langages de programmation, l'un des résultats les plus aboutis est celui de VBOOL (View Based Object-Oriented Language) [14] qui a intégré une nouvelle relation appelée «relation de visibilité» au sein d'une extension d'Eiffel [15]. Dans le même contexte, une méthode d'analyse et de conception par objet VBOOM (View Based Object Oriented

Method) [12] supportant les concepts de vue et point de vue a été élaborée et cible le langage VBOOL.
Par ailleurs, dans les dernières années le langage de modélisation unifié UML (Unified Modeling Language) connaît une forte utilisation dans la communauté qui s'intéresse à la modélisation orientée objet et s'impose comme un standard de fait dans le domaine de l'analyse/conception par objet [10][17][18].

A cet effet, l'objectif de ce travail, est de proposer une méthode d'analyse/conception unifiée orientée point de vue dans le but d'avoir un processus de génie logiciel orienté point de vue complet d'UML. La base d'élaboration de la nouvelle méthode (appelée U_VBOOM) s'inspire essentiellement de la méthode VBOOM et offre un formalisme étendant celui d'UML. Et pour que la nouvelle méthode U_VBOOM cible les langages orientés objet les plus utilisés dans le marché comme C++, Java, Eiffel, etc. nous proposons une approche d'implémentation multi-cibles de la relation de visibilité.

Cet article s'organise de la manière suivante : dans la section 2, nous présentons la relation de visibilité et la nouvelle approche d'implémentation. La section 3 présente les principes de la démarche U_VBOOM. Nous concluons par un bilan du travail effectué et une présentation de ses perspectives.

Durant tout ce papier nous allons illustrer nos propos à l'aide de l'exemple de gestion d'une médiathèque. Le cahier des charges de cet exemple est détaillé dans [13]. Le système MEDIATHEQUE doit proposer à ses adhérents de consulter et emprunter les différents types de support : livres, cassettes vidéo et audio, CD audio, etc. Seul un membre de la bibliothèque peut emprunter. Le prêt est limité dans le temps et au maximum 3 exemplaires peuvent être empruntés par un adhérent. Les différents utilisateurs potentiels du système MEDIATHEQUE sont: le bibliothécaire qui gère les prêts, le responsable adhérent qui va ajouter et retirer des adhérents, le responsable exemplaire qui va saisir les nouveaux exemplaires et retirer ceux qui sont abîmés. Selon le type d'utilisation, le système MEDIATHEQUE pourra être considéré, comme un ensemble de Comptes d'adhérents, ou d'Exemplaires, ou un moyen pour faciliter les Prêts.

De plus, nous allons illustrer le code d'implémentation en langage C++ [20], mais ceci reste valable pour d'autre langage orienté objet.

2- Relation de visibilité et les concepts liés

L'expression de représentations multiples ou du concept de vue/point de vue dans les méthodes de conception orientées objet a suscité de nombreuses recherches. 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. La méthode VBOOM cible le langage VBOOL extension du langage orienté objet Eiffel. Le principal apport de VBOOL est l'ajout au langage Eiffel d'une nouvelle relation appelée "relation de visibilité" et d'un nouveau concept "classe flexible".

Le principe de la relation de visibilité est lié à sa sémantique que l'on peut exprimer par "est vu comme"[13]. Elle permet d'exprimer le fait qu'une "entité" peut être considérée sous des "angles" multiples correspondant aux points de vue des utilisateurs concernés. Pour l'exemple de la médiathèque, selon le type d'utilisation, le système MEDIATHEQUE pourra peut "être vu comme" un ensemble de Comptes d'adhérents, des prêts, des exemplaires d'œuvres, etc. Ainsi, nous identifions 4 classes de base du système : Médiathèque, Comptes_Adhérents, Exemplaires et Prêts. La classe Comptes_Adhérents devient une *vue* de la classe Médiathèque via la relation de visibilité. Cette classe Médiathèque est appelée "flexible" au sens où son utilisation peut varier selon le contexte. Une classe flexible (classe multi-vues) est une classe dans laquelle on déclare un ensemble d'au moins deux liens de visibilité avec d'autres classes appelées ses "vues" qui seront sélectionnées au moment du choix d'un point de vue de l'utilisateur (Figure 1).

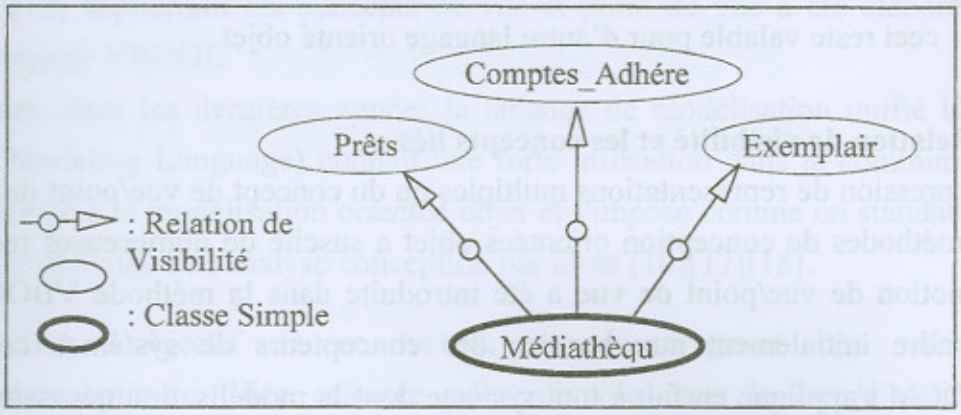

Figure 1 : *Représentation de la relation de visibilité et de la classe flexible en VBOOM*

Pour réaliser le compilateur VBOOL, M. Nassar [16] et S. Marcaillou [14] ont proposés une implémentation particulière de la relation de visibilité. Cette implémentation consiste à simuler la relation de visibilité à un héritage multiple sélectif. Ainsi, la compilation du code VBOOL, se fait par la génération intermédiaire du code Eiffel.

Mais, ces travaux souffrent encore d'un certain nombre de limites, à savoir :

VBOOL est fondé sur l'héritage multiple (inspiré du langage Eiffel) qui supporte mal l'évolutivité du modèle ; en effet, l'ajout d'une vue remet en cause la classe flexible et les classes qui en dérivent ;

la dissociation des notions de vue et point de vue complexifie la modélisation ;

la formalisation et l'implémentation du polymorphisme est délicate en VBOOL ;

la méthode VBOOM doit être mise au standard UML.

Afin de surmonter ces problèmes, profiter des avantages des langages orientés objet du marché comme C++, Java, Eiffel, etc. et faire des modélisations orientées objet par points de vue basées sur UML, nous proposons une nouvelle approche d'implémentation multi-cibles de la relation de visibilité et une nouvelle méthode orientée point de vue U_VBOOM qui s'inspire essentiellement de la méthode VBOOM et intègre le maximum possible les artefacts proposés par UML.

2.1- Implémentation de la relation de visibilité

A la différence de l'approche utilisée par M. Nassar et S. Marcaillou pour implémenter la relation de visibilité, notre approche consiste à utiliser la relation d'agrégation à la place de l'héritage. L'héritage est un mécanisme qui permet à une sous-classe de déléguer à une super-classe le traitement de méthodes héritées. Le mécanisme de délégation utilisé conjointement à la relation d'agrégation permet lui aussi à une classe de déléguer le traitement d'une requête à une autre classe. Mais, avec le mécanisme d'agrégation, la représentation de l'évolution d'un objet donné dans un tel processus en fonction du type d'utilisation joué dans ces processus devient une tâche facile à réaliser. C'est ainsi, pour notre exemple de la gestion de la médiathèque, un objet de type Médiathèque peut être utilisé et subit une évolution différente dans le processus de gestion de prêt, dans celui de gestion des exemplaires et dans celui de gestion des comptes adhérents. C'est à dire que le système MEDIATHEQUE n'est perçu et utilisé par un client qu'au travers un type d'utilisation. La classe multi-vues Médiathèque délègue alors l'utilisation du système MEDIATHEQUE comme un moyen de prêts à la classe Prêts, comme un moyen pour gérer les exemplaires à la classe Exemplaires et enfin comme un moyen pour gérer les comptes adhérents à la classe Comptes_Adhérents. La classe multi-vues Médiathèque a donc une relation d'agrégation avec les vues Prêts, Comptes_Adhérents et Exemplaires (Figure 2).

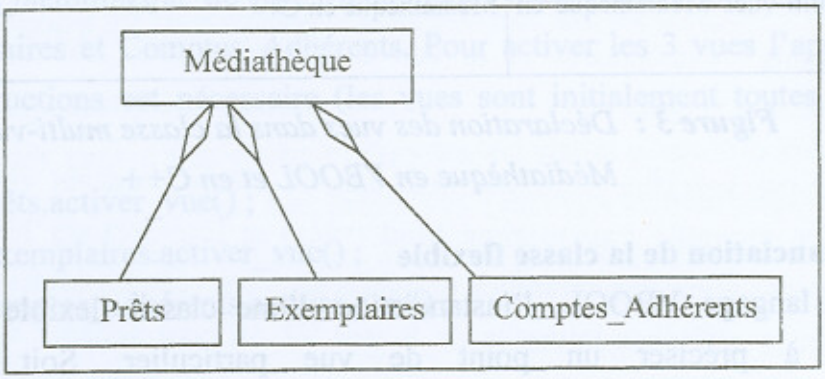

Figure 2 : *Simulation de la relation de visibilité en UML*

2.2- Déclaration de la classe flexible

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 3(a)). Dans la nouvelle implémentation de la relation de visibilité, la déclaration des vues au sein de la classe multi-vues devient des attributs de la classe multi-vues via la relation d'agrégation sur l'ensemble des vues du système (il y a autant d'attributs que des vues du système) (Figure 3(b)). Par conséquent, la déclaration d'une classe flexible (multi-vues) devient comme une déclaration d'une classe simple dans un langage objet donné.

```
flexible class Médiathèque
        -- déclaration des vues
.....
seen_as Prêts
seen_as Exemplaires
seen_as Comptes_Adherents
.......
End; - - class Médiathèque
```

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

```
# include <Prêts.h>  // inclure la classe Prêts
# include <Exemplaires.h> // inclure la classe Exemplaires
# include <Comptes_Adhérents.h>
                    // inclure la classe Comptes_Adhérents
 ....
Class Médiathèque
public :
Vue_prêts    *Prêts
Vue_ Exemplaires   *Exemplaires
Vue_Comptes_adhérents   *Comptes_adhérents
 ....
End ; // fin classe
```

(b) : Déclaration des vues dans la classe multi-vues Médiathèque en C++

Figure 3 : *Déclaration des vues dans la classe multi-vues Médiathèque en VBOOL et en C++*

2.3- Instanciation de la classe flexible

Dans le langage VBOOL, l'instanciation d'une classe flexible (multi-vues), consiste à préciser un point de vue particulier. Soit l'exemple : *Bibliothécaire_mediatheque : Médiathèque(*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 multi-vues est devenue comme la déclaration d'une classe simple, son instanciation selon un point de vue dans la nouvelle approche devient aussi une instanciation simple avec l'activation des vues composant ce point de vue. Pour réaliser ceci, nous proposons d'ajouter une classe mère, nommé Vue, de toutes les classes représentant les vues du système. Cette classe possède l'attribut booléen **etat_vue** qui exprime l'état courante d'une vue et les 3 méthodes **etat_vue()**, **activer_vue()** et **desactiver_vue()**. Ces méthodes permettent respectivement de retourner l'état active ou désactive d'une vue, d'activer une vue et en fin de désactiver une vue (un point de vue est une combinaison de vues actives) (Figure 4).

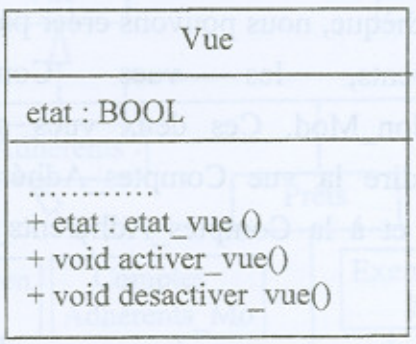

Figure 4 : *Interface de la classe Vue en C++*

L'instanciation de la classe multi-vues Médiathèque selon le point de vue *bibliothecaire_mediatheque* se ramène donc à l'instanciation de l'objet *bibliothecaire_mediatheque* de type Médiathèque avec l'activation des trois vues Prêts, Exemplaires et Comptes_Adhérents. Pour activer les 3 vues l'appellation des trois instructions est nécessaire (les vues sont initialement toutes crées et désactivées) :

```
Vue_prêts.activer_vue() ;
Vue_exemplaires.activer_vue() ;
Vue_comptes_adhérents.activer_vue() ;
```

2.4- Exclusion mutuelle de vues

L'un des intérêts de l'utilisation des points de vue est la possibilité de définir des droits d'accès au modèle. Si nous considérons par exemple l'accès de bibliothécaire à notre modèle, les vues Prêts, Exemplaires et Comptes_Adhérents le concernent. Contrairement au responsable adhérents, le bibliothécaire ne doit pas avoir le droit de changer l'attribut "bloquer_compte_adherent" d'un adhérent, donc d'appeler la primitive "bloquer_compte_adherent" de la vue Comptes_Adhérents. Pour résoudre ce problème de droit d'accès, le concept de "vues en exclusion mutuelle" a été introduit, c'est-à-dire spécifier les vues qui ne peuvent pas être simultanément dans le même point de vue. Ainsi, dans l'exemple de gestion de la médiathèque, nous pouvons créer par héritage deux vues héritant de Comptes_Adhérents, les vues Comptes_Adhérents_Mod et Comptes_Adhérents_Non_Mod. Ces deux vues sont en exclusion mutuelle (Figure 5) : c'est à dire la vue Comptes_Adhérents_Mod qu'aura accès le responsable adhérents et à la Comptes_Adhérents_Non_Mod qu'aura accès le bibliothécaire.

De la même façon, le bibliothécaire ne doit pas avoir le droit de changer l'attribut "nombre_exemplaire_disponible" d'un exemplaire, contrairement au responsable exemplaires qui peut ajouter de nouvelles exemplaires ou retirer les exemplaires abîmés par l'invocation des deux primitives "ajouter_exemplaire" ou "retirer_exemplaire" de la vue Exemplaires. Ainsi, nous pouvons créer aussi deux vues héritant de Exemplaires, la vue Exemplaires_Mod qu'aura accès le responsable exemplaires et la vue Exemplaires_Non_Mod qu'aura accès le bibliothécaire (Figure 5).

Pour ne pas spécifier simultanément dans le même point de vue deux vues en exclusion mutuelle Exemplaires_Mod (=V1) et Exemplaires_Non_Mod (=V2), nous devons avoir une référence mutuelle dans les vues en exclusion mutuelle. A l'aide de la référence mutuelle de la vue V1 on peut accéder à la vue V2 (qui réalise avec V1 l'exclusion mutuelle) et savoir son état active ou désactive et inversement. Pour cela, nous ajoutons la classe mère, nommée Vue_Exclusion, à toutes les classes représentant les vues en exclusion mutuelle du système. Cette

classe a un attribut nommé Reference_Vue de type Vue. L'attribut Reference_Vue est alors un attribut hérité uniquement par les vues en exclusion mutuelle et qui va indiquer la référence de la vue qui réalise l'exclusion mutuelle avec la vue en cours.

Exemplaires_Mod.Reference_Vue = = Exemplaires_Non_Mod;
Exemplaires_Non_Mod.Reference_Vue = = Exemplaires_Mod;

Comptes_Adhérents_Mod.Reference_Vue = = Comptes_Adhérents_Non_Mod;
Comptes_Adhérents_Non_Mod.Reference_Vue = = Comptes_Adhérents_Mod;

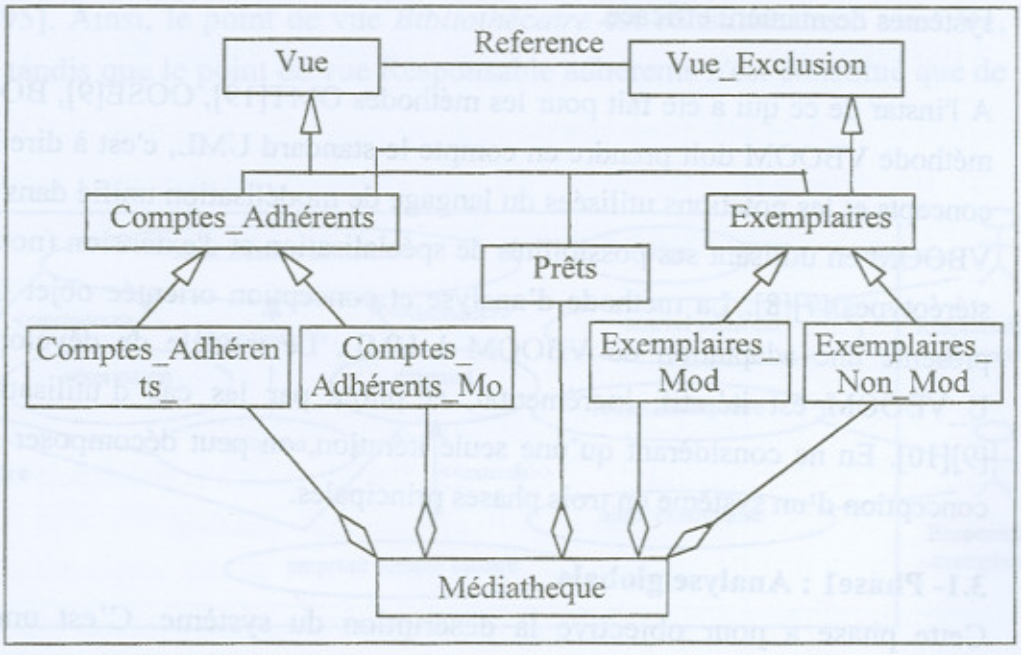

Figure 5 : *Diagramme orienté point de vue en UML*

2.5- Evolution dynamique des points de vue

La déclaration des vues d'une classe multi-vues est statique, mais le mécanisme de point de vue est dynamique. Les primitives (attributs et méthodes) définies dans la section 2.3 permettent l'évolution de points de vue applicables aux objets. Ces primitives sont évidemment soumises à des règles strictes. L'utilisation de telle primitive permet à l'utilisateur de faire évoluer dynamiquement les points de vue des objets et lui donne donc la possibilité de faire évoluer ses droits d'accès sur le modèle. Ainsi, il sera possible d'activer et de désactiver les vues en changeant le type dynamique d'un objet multi-vues.

3- Méthode U_VBOOM

VBOOM est une méthode d'analyse et conception orientée objet intégrant le concept de vue/point de vue dans la modélisation orientée objet [4][12]. VBOOM permet de mener des conceptions partielles spécifiques aux différents types d'utilisation du système. Chaque type d'utilisation du système est associé à un sous-système du système global. Ces sous-systèmes (appelés aussi modèles partiels) sont par la suite fusionnés sous forme d'un modèle global multi-vues, accessible selon plusieurs points de vue. L'originalité de VBOOM est de fournir aux concepteurs une démarche pour fusionner les résultats issus de la conception des différents sous-systèmes de manière efficace.

A l'instar de ce qui a été fait pour les méthodes OMT[19], OOSE[9], BOOCH[2], la méthode VBOOM doit prendre en compte le standard UML, c'est à dire intégrer les concepts et les notations utilisées du langage de modélisation unifié dans la méthode VBOOM en utilisant ses possibilités de spécialisation et d'extension (notamment les stéréotypes)[7][8]. La méthode d'analyse et conception orientée objet U_VBOOM présente une adaptation de VBOOM à UML. Le modèle de développement de U_VBOOM est itératif, incrémental, et piloté par les cas d'utilisation d'UML [9][10]. En ne considérant qu'une seule itération, on peut décomposer l'analyse et conception d'un système en trois phases principales.

3.1- Phase1 : Analyse globale

Cette phase a pour objective la description du système. C'est une phase de spécification globale de développement par U_VBOOM. Elle consiste à fournir une description précise des différents besoins des utilisateurs du système. Pour exprimer les besoins des utilisateurs au travers le système, U_VBOOM propose les cas d'utilisation d'UML (Figure 6). L'élaboration des vues du système est supportée par la technique de description de cas d'utilisation en actions [7][8]. Cette technique consiste à décrire les cas d'utilisation par un ensemble de séquences d'actions. Une action est un effet produit par un acteur agissant de manière déterminée sur le système ou par le système lui-même pour atteindre son objectif [5]. Chaque action peut ainsi être identifiée par un numéro et un libellé comme représenté sur la Figure 7.

L'étude des actions associées à chaque acteur permet d'associer des vues aux points de vue. Une vue correspond à une liste d'actions propres à un acteur donné ou résulte d'une intersection d'actions communes à un groupe d'acteurs [Kriouile 95]. Pour le système MEDIATHEQUE, nous obtenons ainsi le schéma de points de vue décrit par la Figure 8. Ce schéma met tout d'abord en évidence 3 vues nommées respectivement V1, V2 et V3 : Comptes_Adhérents, Exemplaires, Prêts. La vue Exemplaires est inférée à partir des actions a3, a8 et a9 relatives naturellement à la gestion des exemplaires. Chaque point de vue se voit associé une combinaison de vues conformément à la démarche proposée par VBOOM [Kriouile 95]. Ainsi, le point de vue *Bibliothécaire* est constitué des 3 vues V1, V2 et V3, tandis que le point de vue Responsable adhérents n'est constitué que de la vue V1.

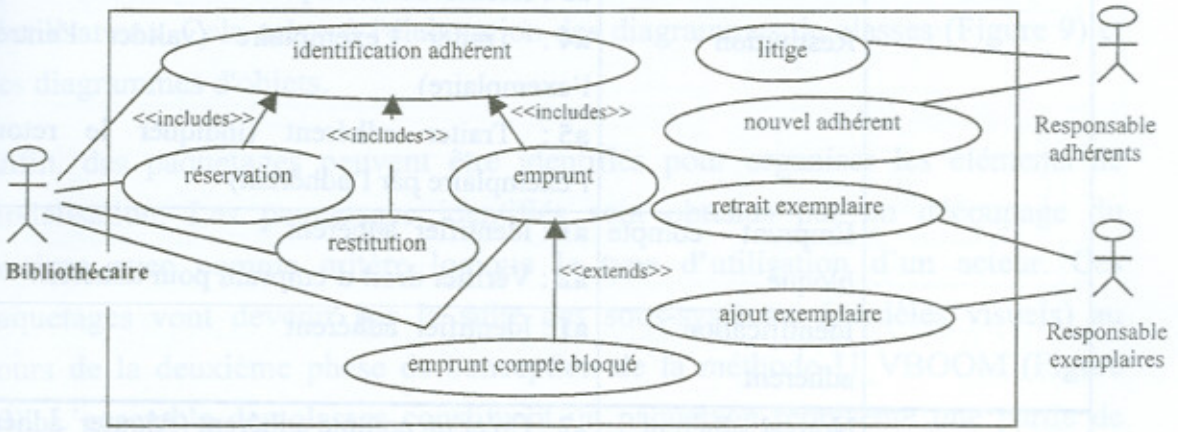

Figure 6 : *Diagramme de cas d'utilisation de la MEDIATHEQUE*

Acteurs	Cas d'utilisation	Décomposition en actions
Bibliothécaire	Emprunt	**a1** : Identifier adhérent **a2** : Vérifier droit d'emprunt pour adhérent **a3** : Rechercher exemplaire **a4** : Traiter l'exemplaire (valider la sortie de l'exemplaire) **a5** : Traiter adhérent (indiquer l'emprunt par l'adhérent)
	Réservation	**a1** : Identifier adhérent **a2** : Vérifier droit d'emprunt pour adhérent **a3** : Rechercher exemplaire **a6** : Réserver exemplaire
	Restitution	**a1** : Identifier adhérent **a3** : Rechercher exemplaire **a4** : Traiter l'exemplaire (valider l'entrée de l'exemplaire) **a5** : Traiter adhérent (indiquer le retour de l'exemplaire par l'adhérent)
	Emprunt compte bloqué	**a1** : Identifier adhérent **a2** : Vérifier droit d'emprunt pour adhérent
	Identification adhérent	**a1** : Identifier adhérent
Responsable adhérents	Nouvel adhérent	**a2** : Créer un compte adhérent (Ajouter adhérent)
	Litige	**a1** : Identifier adhérent **a2** : Bloquer compte adhérent **a7** : Avertir adhérent
Responsable exemplaires	Ajout exemplaire	**a3** : Rechercher exemplaire **a8** : Ajouter exemplaire
	Retrait exemplaire	**a3** : Rechercher exemplaire **a9** : Retirer exemplaire abîmé

Figure 7 : *Décomposition des différents cas d'utilisation en actions élémentaires*

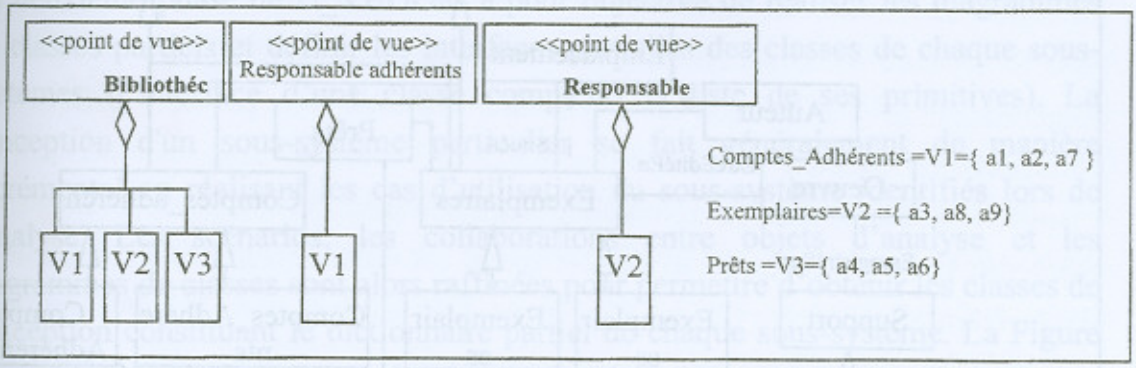

Figure 8 : *Schémas de points de vue du système MEDIATHEQUE*

La phase d'analyse globale se poursuit pour identifier les objets et les classes qui appartiennent au domaine du problème. Les classes sont découvertes scénario après scénario, au moyen d'objets qui, en collaborant, réalisent des cas d'utilisations. Cela mène à l'élaboration des diagrammes de classes (Figure 9) et des diagrammes d'objets.

Enfin, des paquetages peuvent être identifiés pour organiser les éléments de modélisation. Les paquetages identifiés sont obtenus par un découpage du système avec comme critère logique le type d'utilisation d'un acteur. Ces paquetages vont devenir par la suite des sous-systèmes (modèles visuels) au cours de la deuxième phase de conception de la méthode U_VBOOM (Figure 10). L'ensemble des classes constituant un paquetage représente une partie de diagramme de classe du système. Cette partie est définie en respectant les règles suivantes :

- Les seules vues du paquetage sont celles de son point de vue.
- Les classes reliées par la relation de clientélisme à une classe du paquetage.
- Les classes ancêtres par héritage à une classe du paquetage.

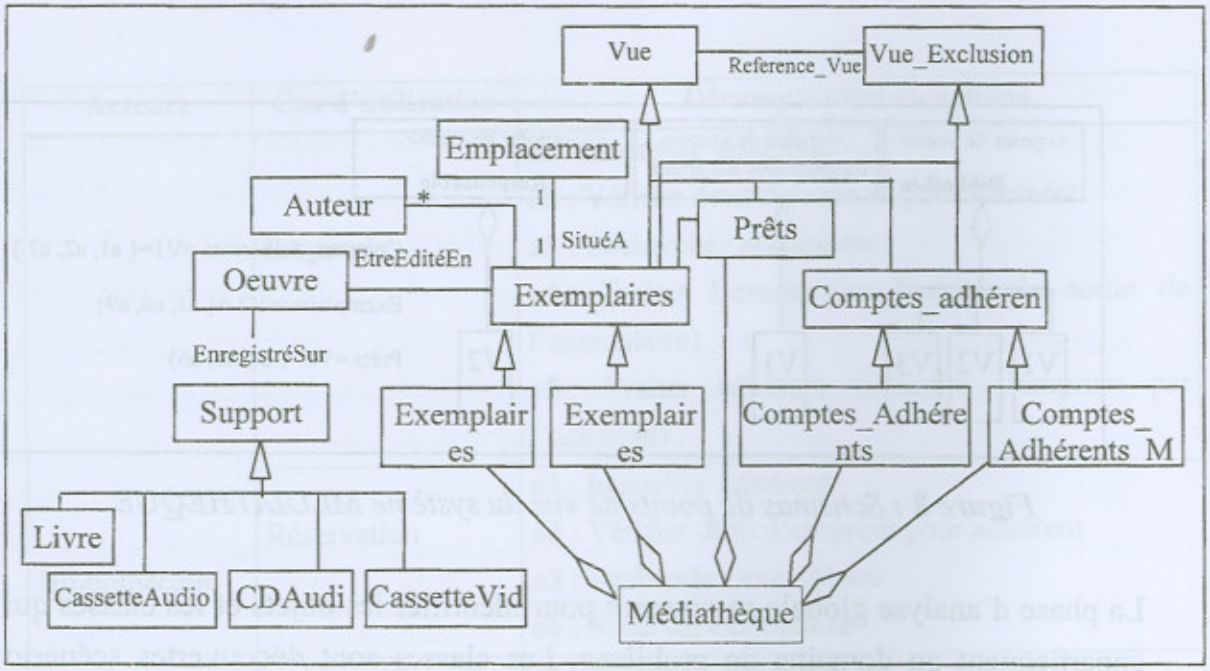

***Figure 9** : Diagramme de classes initial du système MEDIATHEQUE*

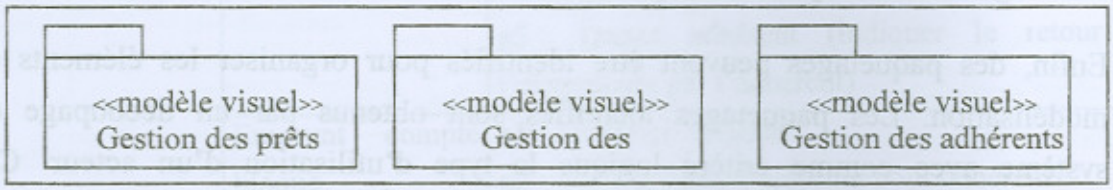

***Figure 10** : Les 3 trois paquetages du système MEDIATHEQUE*

3.2- Phase2 : Conception des sous-systèmes

L'analyse globale de la méthode U_VBOOM est élaborée et elle sera enrichie en deuxième phase de conception de la méthode. La conception des sous-systèmes permet alors la traduction du modèle d'analyse. Ces sous-systèmes constituent un artefact essentiel de la deuxième phase de la méthode U_VBOOM. En effet, le découpage de l'espace de solution du problème en sous-systèmes permet permet de simplifier considérablement la conception du système global en des conceptions partielles des sous-systèmes. La conception des sous-systèmes peut être menée en parallèle ou séquentiellement comme décrit dans [7].

La deuxième phase de U_VBOOM a pour objective de réaliser les diagrammes de classes partiels et définir les interfaces partielles des classes de chaque sous-systèmes (l'interface d'une classe comprend la liste de ses primitives). La conception d'un sous-système particulier se fait généralement de manière incrémental en réalisant les cas d'utilisation du sous-système identifiés lors de l'analyse. Les scénarios, les collaborations entre objets d'analyse et les diagrammes de classes sont alors raffinées pour permettre d'obtenir les classes de conception constituant le dictionnaire partiel de chaque sous-système. La Figure 11 représente le diagramme de classes (partiel) du sous-système **Gestion des prêts**.

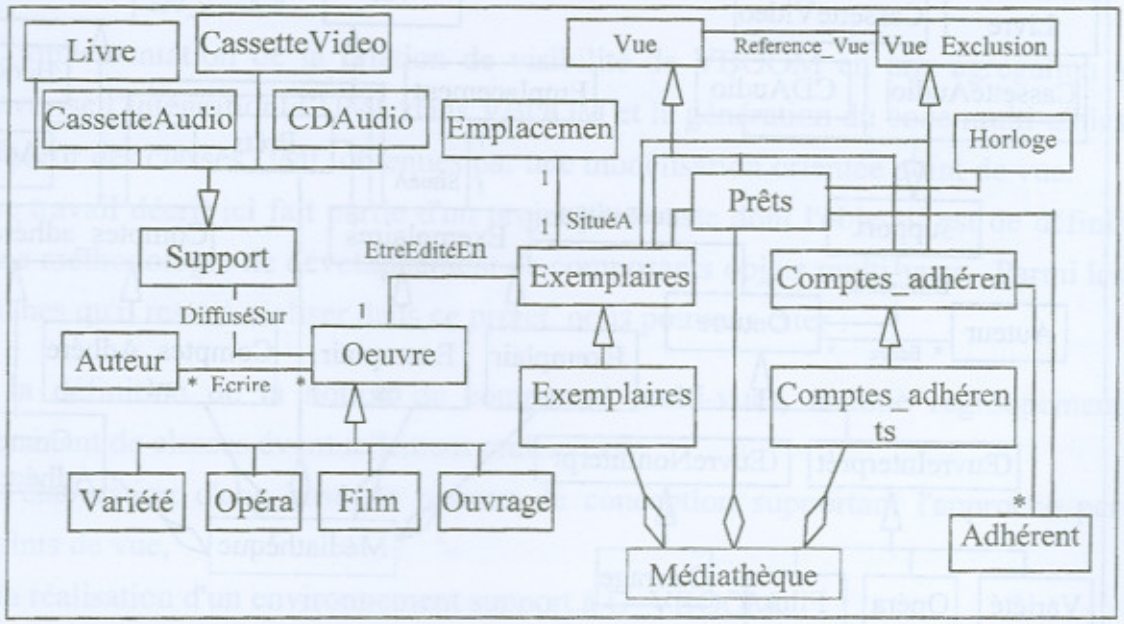

Figure 11 : *Diagramme de classes du sous-système Gestion des prêts*

3.3- Conception du modèle global

La troisième phase de la méthode U_VBOOM fournit aux concepteurs une démarche pour fusionner les résultats partiels obtenus à l'issue de la conception des sous-systèmes. Ainsi, la méthode U_VBOOM doit fournir aux concepteurs une démarche pour fusionner les différents diagrammes de classes partiels et les différentes interfaces de classes partielles des sous-systèmes, en leur proposant des heuristiques pour gérer les conflits éventuels qui peuvent apparaître au cours de la fusion : la synonymie et la polysémie des primitives, l'adaptation des paramètres des méthodes et éventuellement le typage, etc.. .

La figure 12 ci-dessous représente le diagramme de classes global du système pour le système MEDIATHEQUE. Ce diagramme est obtenu après fusion des diagrammes de classes des sous-systèmes **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. Dans cet exemple, le modèle global est relativement proche du modèle visuel *Gestion des prêts* car ce dernier est prédominant dans le système, mais cette situation n'est évidemment pas une généralité.

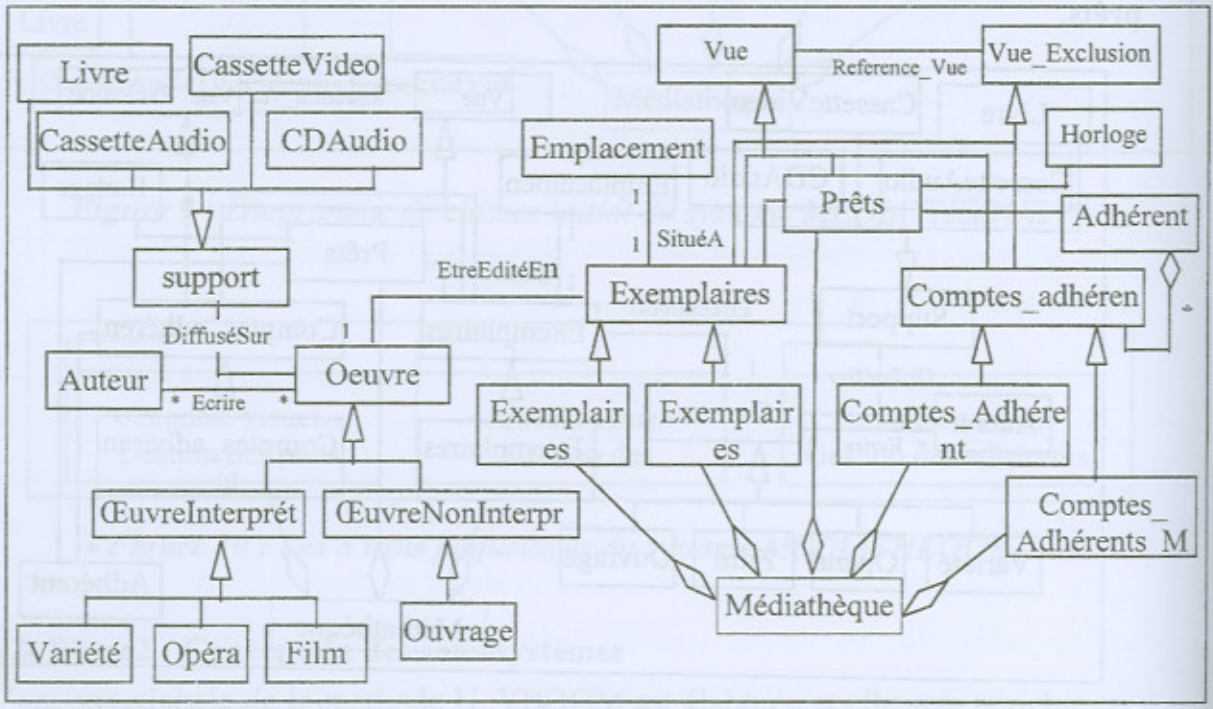

Figure 12 : *Diagramme de classes de l'exemple de la gestion de la Médiathèque*

4- Conclusion

L'approche présentée dans cet article a pour intérêt de faire des modélisations orientées points de vue sous le standard UML. Le résultat est donc une standardisation du travail effectué dans VBOOM. La méthode U_VBOOM représente le résultat de l'adaptation de VBOOM sous le standard UML. Elle préserve le développement en continu offert par la méthodologie objet. U_VBOOM améliore le processus de développement proposé par la méthode VBOOM par la réutilisation de concepts majeurs d'UML. La démarche d'analyse/conception selon U_VBOOM est un processus incrémental et itératif, piloté par les cas d'utilisation d'UML.

L'implémentation de la relation de visibilité de VBOOM en une agrégation a favorisé l'intégration d'UML dans VBOOM et la génération du code multi-cibles à partir des classes UML obtenues par une modélisation orientée point de vue.
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] Bardou, D. -Etude de langages à prototypes, du mécanisme de délégation et de son rapport à la notion de point de vues.-Thèse de Doctorat en Informatique, LIRMM, Université de Montpellier 2, avril 1998.

[2] Booch, G.- Object-Oriented Analysis and Design with Applications.- Benjamin/Cummings, Redwood City, 2ème édition, 1994.

[3] Carre, B. ; Geib, J.M. -The Point of View notion for Multiple Inheritance.- In: Actes de ECOOP/OOPSLA, 1991.

[4] Coulette, B. ; Kriouile, A. ; Marcaillou, S. -L'approche par points de vue dans le développement orientée objet de systèmes complexes. -In Revue l'Objet, vol.2, n°4, 1996. -pp.13-20.

[5] Dano, B. - Spécifications dynamiques des besoins orientés-objet : une démarche fondée sur les scénarios. -In : Actes de INFORSID, juin 1996.

[6] Finkelstein, A. ; Gabbay, D. ; Hunter, A. ; Kramer, J. ; Nuseibeh, B. - Inconsistency Handling in Multi-Perspective Specifications. -In : Actes de conférence ESEC'93, 1993, Garmish- Paternkirchen (D), pp.84-99.

[7] Hair, A. ; Kriouile, A. ; Coulette, B. -VUML : Une méthode d'analyse et de conception orientée objet, intégrant UML et le concept de point de vue.- In : Actes de conférence ICSSEA'2001 : International Conference on Systems, Software Engineering and their applications, 4-6 décembre 2001, vol.3, -Paris.

[8] Hair, A. ; Kriouile, A. ; Coulette, B. -Un processus d'analyse et de conception unifié basé sur le concept de point de vue. –In : Actes de conférence CARI'02 : 6ème colloque africain sur la recherche en informatique, 14-17 octobre 2002,- Yaoundé-pp.229-237.

[9] Jacobson, I. ; Christerson, M. ; Jonsson, P. ; Overgaard, G. -Object-Oriented Software Engineering, A Use Case Driven Approach. -Addison-Wesley, Inc., 1992.

[10] Jacobson, I. ; Booch, G. ; Rumbaugh, J. -The Unified Software Development Process. Addison-Wesley, Inc., 1999.

[11] Kaplan, H. M. ; Harrison, W. ; Katz A. ; Kruskal, V. -Subject-oriented composition rules. -In Acte de Coférence OOPSLA'95, Octobre 1995, -Austin, TX, -pp.235-250.

[12] Kriouile, A. -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.

[13] Lopez, N. ; Migueis, J. ; Pichon E. -Intégrer UML dans vos projets. -Edition Eyrolles, 1998.

[14] Marcaillou, S. - 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.

[15] Meyer, B. -Object success - A managers's guide. -Prentice Hall - The Object-Oriented Series, 1995.

[16] Nassar, M. -Vers une programmation orientée objet par point de vue - Conception et réalisation d'un compilateur pour le langage VBOOL-. -Thèse pour l'obtention du diplôme de spécialité de 3ème cycle de l'universite Mohamed V, ENSIAS, -Rabat, 1999.

[17] 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.

[18] Rumbaugh, J. ; Jacobson, I. ; Booch, G. -The Unified Modeling Language Reference Manual. -Addison-Wesley, 1999.

[19] Rumbaugh, J. ; Blaha, M. ; Premerlani, W. ; Eddy, F. ; Lorensen, W. -OMT: Modélisation et conception orientées objet. Prentice Hall, 1995.

[20] Stroustrup, B. -The C++ Programming Language. -Addison-Wesley, 3ème Edition, 1997.