Vers une démarche pour le développement de modèles à base de Composants Multivue

Mustapha HAIN*, Abdelaziz MARZAK*
Bernard COULETTE**, Mahmoud NASSAR***

*Laboratoire TIM, Equipe IGSI
Université Hassan II-Mohammedia, Faculté des Sciences Ben M'sik,
PB 7955 Casablanca. Maroc
infohain@yahoo.fr, marzak@hotmail.com
**Laboratoire IRIT,
Université de Toulouse 2-le Mirail
5, allées Antonio Machado F-31058 TOULOUSE CEDEX 9, France
bernard.coulette@irit.fr
***Laboratoire SIME, Equipe IMS, ENSIAS
PB 713 Agdal, Rabat, Maroc
nassar@ensias.ma

Résumé. VUML (*View based UML*) est un langage de modélisation objet qui a introduit les concepts de Classe Multivue et de Composant Multivue dans UML. Cependant, tout comme UML, VUML ne propose pas de méthode pour élaborer le diagramme de Composants Multivue. Pour combler ce manque, nous nous sommes appuyés sur les méthodes UP (*Unified Process*) et CUP (*Component Unified Process*) pour proposer une démarche pour le développement d'un PIM (*Platform Independent Model*) à base de Composants Multivue. Dans cet article, nous décrivons les étapes de cette démarche permettant d'aboutir au diagramme de Composants Multivue.

1 Introduction

L'élaboration des logiciels nécessite souvent de multiples compétences qui ne peuvent pas être mises en oeuvre par un processus linéaire. De ce fait, l'introduction de la notion de préoccupation dans la modélisation des systèmes complexes a donné lieu aux approches par multi-modélisation. L'intérêt de ces approches est de proposer des moyens de réduction de la complexité de l'analyse/conception de ces systèmes. On peut citer dans cette catégorie les approches traditionnelles telles que la modélisation par points de vue (Finkelstein et al., 1990) (Coulette et al., 1996), la modélisation par sujets (Ossher et al., 1996), la modélisation par aspects (Kiczales et al., 1997), ou plus récemment l'ingénierie multi-modèle (Muller et al., 2007).

C'est également dans cette optique qu'a été élaboré dans notre équipe le langage VUML (Nassar et al., 2003) (Nassar, 2005), un profil qui étend UML en proposant le concept de classe

multivue. Dans VUML, à chaque acteur interagissant avec le système est associé un point de vue dit fonctionnel. Le principe de la modélisation avec VUML consiste à établir des modèles UML séparés correspondant aux points de vue fonctionnels, puis à composer ces modèles pour obtenir un modèle VUML global. Une classe multivue est composée d'une base partagée par les différents acteurs, et de vues qui étendent la base en décrivant les spécificités de chaque point de vue. La composition (fusion) des modèles a été implémentée au sein de l'approche VUML par l'intermédiaire d'un processus à base de transformations ATL (*ATLAS Transformation Language*) (Anwar et al., 2010). Cependant, nous ne traitons pas dans notre approche la phase amont de résolution des conflits dus à des problèmes d'homonymie/synonymie. En effet, cette phase, indispensable, est difficilement automatisable et suppose par exemple un traitement à base d'ontologie qui sort du champ de nos travaux.

Par ailleurs, VUML propose également le concept de Composant Multivue (El Asri et al., 2005). De façon similaire à la classe multivue, un Composant Multivue est composé d'une interface base (*partagée par tous les acteurs*) et d'un ensemble d'interfaces vue (*extensions de la base*), chaque vue correspondant aux besoins spécifiques d'un acteur. L'intérêt des composants multivue est d'une part de permettre une conception décentralisée faite points de vue - ce qui est conforme à la pratique industrielle -, d'autre part de faciliter la prise en compte d'un nouveau point de vue. En effet, pour intégrer a posteriori un nouveau point de vue, il faut développer un modèle de conception selon ce point de vue - ce qui ne remet pas en cause le modèle global existant - puis fusionner le modèle obtenu avec le modèle global. Le profil VUML est accompagné d'une démarche inspirée de la méthode VBOOM (*View Based Object Oriented Methodology*) (Kriouile, 1995) et permettant de produire un diagramme de classes. Par contre, cette démarche ne couvre pas à ce jour le processus d'élaboration du diagramme de Composants Multivue.

L'objectif de cet article est de proposer une démarche pour le développement à base de Composants Multivue. Plus précisément, nous visons la phase d'analyse/conception en permettant l'élaboration de PIMs (*Platform Independent Models*) (OMG, 2003a) à base de Composants VUML. Le passage d'un PIM à un PSM (*Platform Specific Model*), sur lequel nous travaillons également via des transformations ATL, n'est pas considéré dans cet article.

Le reste de cet article est structuré comme suit : la deuxième section présente la notion de Composant Multivue qui a été développée dans notre équipe et que nous reprenons comme principe de base (Hain et al., 2007), et un aperçu de méthodes représentatives pour le développement à base de composants. Dans la troisième section, nous proposons notre démarche. Pour illustrer notre approche, nous présentons dans la quatrième section une étude de cas développée dans le cadre d'une convention de recherche CNRS-CNRST (Marzak et al., 2010). En conclusion, nous faisons le point sur le bilan du travail effectué et nous présentons nos perspectives de recherche.

2 Contexte

2.1 Concept de Composant Multivue

Dans cet article, nous nous situons dans la continuité des travaux réalisés autour de VUML sur la notion de composant Multivue (El Asri et al., 2005) en focalisant notre approche sur l'aspect méthodologique. La figure 1 présente un court extrait du méta-modèle VUML développé

dans notre équipe, correspondant à la description des composants. Les éléments représentant un Composant Multivue sont colorés en gris, les autres étant issus du méta-modèle UML2.0 (OMG, 2003b).

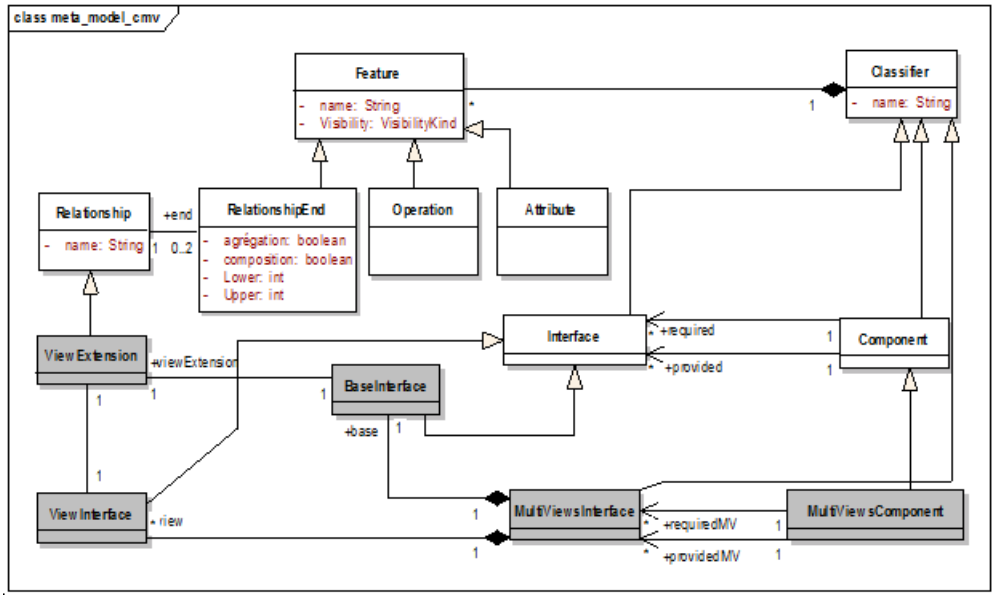

FIG. 1 – *Extrait du méta-modèle concernant le concept de Composant Multivue (Nassar, 2005).*

Selon le méta-modèle, un Composant Multivue est une extension d'un composant UML 2.0 (El Asri et al., 2005). Il combine les critères de réutilisation des composants standards et la flexibilité via la notion de point de vue. Le langage VUML offre un ensemble de stéréotypes pour spécifier le Composant Multivue et ses Interfaces Multivues. A titre d'exemple, le stéréotype «multiViewComponent» désigne un Composant Multivue. Chaque Composant Multivue dispose d'au moins une interface Multivue (requise ou fournie) composée d'une interface de base stéréotypée «baseInterface», et un ensemble d'interfaces vues stéréotypées «ViewInterface». Les interfaces vue sont reliées à l'interface base par une relation stéréotypée «ViewExtenstion». La spécificité d'un Composant Multivue est d'offrir ses services à travers différentes interfaces Multivue dont la définition change selon la vue active. De plus, un Composant Multivue est doté d'un mécanisme de gestion de la cohérence entre ses vues qui peuvent, éventuellement, être dépendantes les unes des autres (El Asri et al., 2005).

Pour illustrer le concept de Composant Multivue de VUML, nous choisissons un exemple simple de diagramme de Composants Multivue dans le domaine de la santé. Par souci de simplicité, nous nous concentrons sur deux acteurs : l'administrateur du ministère de la santé (MS) et le responsable d'un Centre Hospitalier Universitaire (CHU).

Le diagramme présenté dans la figure 2 comporte deux Composants Multivue : Hopital et Personnel stéréotypés «MultiViewComponent». Le composant Hopital publie ses méthodes

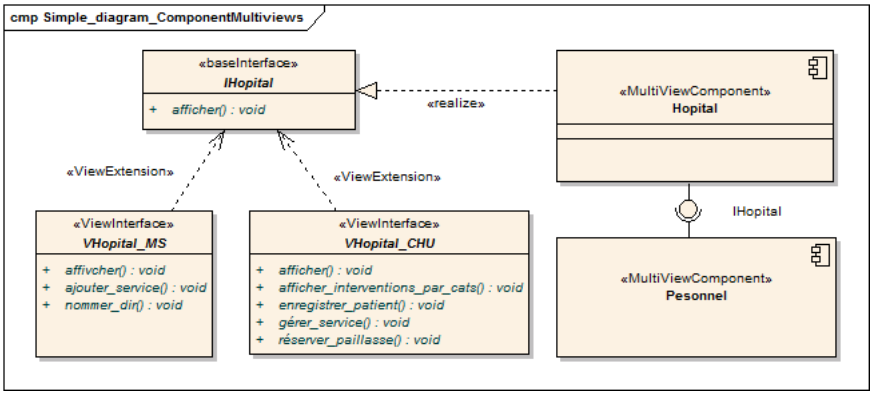

FIG. 2 – *Exemple de diagramme de Composants Multivue.*

dans l'interface IHopital stéréotypée «baseInterface» partagée par les différents acteurs, ainsi que deux interfaces stéréotypées «viewInterface» (VHopital_CHU, VHopital_MS) qui correspondent, respectivement, à la spécificité des acteurs CHU et MS. La méthode afficher() figurant dans les trois interfaces, a trois définitions différentes : une méthode afficher() dans l'interface commune IHopital qui affiche les informations communes à toutes les vues, et une méthode afficher() dans chaque interface vue pour l'affichage des informations spécifiques à chaque vue.

2.2 Méthodes de développement à base de composants

L'ingénierie des méthodes à base de composants est en pleine émergence. Dans cette section, nous synthétisons une étude de l'existant. Nous avons choisi quatre méthodes pouvant s'appliquer à notre contexte et reconnues comme les plus représentatives du domaine : Processus Unifié, Catalysis, UML Component et CUP.

Processus Unifié : le Processus Unifié est piloté par les cas d'utilisation, centré sur l'architecture, itératif et incrémental (Jacobson et al., 1999) (Kruchten, 2004). La modélisation dans le Processus Unifié est organisée en phases, chacune étant constituée d'activités. Le Processus Unifié a l'avantage de recourir aux notations d'UML, toute activité se termine par un livrable. Cependant, il n'utilise les composants que dans les phases finales du cycle de vie d'un logiciel.

Méthode Catalysis : cette méthode est basée sur les notations d'UML (D'Souza et al., 1998). Mais, pour répondre aux besoins de spécification, les auteurs ont ajouté des extensions au langage UML. La méthode est conçue à la fois pour le développement des systèmes orientés objet et pour les systèmes à base de composants. Concernant le paradigme composant, la méthode propose une séparation entre les niveaux *spécification* et *implémentation* des composants. La méthode Catalysis s'articule sur cinq activités : *préparation du modèle de domaine, spécification du système, sélection d'une architecture technique, implémentation et test*.

UML Component : cette méthode est inspirée de plusieurs méthodes, dont UP (*Unified Process*) et Catalysis (Cheesman et al., 2000). Son objectif est la spécification de composants en utilisant la notation UML. Son apport principal est la définition du concept de contrat d'assemblage entre les composants. En plus, elle a étendu la notation UML avec des stéréotypes pour désigner la spécification de composants, par exemple : « Component specification », « SubComponent », « Data Type », « Interface Type » et « Information Type ». En outre, la méthode UML Component définit six workflows, tels que *Capture de besoin, Spécification, Approvisionnement, Assemblage, Test et déploiement*. A titre d'exemple, le sous-workflow « Spécification » contient les trois activités suivantes : *Identification de composants, Interaction entre les composants et Spécification des composants*.

Méthode CUP (*Component Unified Process*) : la méthode CUP est une adaptation du Processus Unifié orientée vers les systèmes à base de composants (Renaux et al., 2004). Elle ajoute notamment à UML des stéréotypes pour la spécification de composants, à savoir :

- « LogicalComponent » : les cas d'utilisation sont organisés, selon le domaine d'intérêt, dans des paquetages considérés comme des composants logiques.
- « ExternalUse » : désigne les relations entre les composants logiques.
- « ExternalControl » : concerne le diagramme de séquence ; il permet d'identifier les interactions entre les objets de différents composants logiques.

Après l'étude de ces quatre méthodes de développement à base de composants, nous tirons le bilan suivant :

- La méthode Catalysis est complexe à mettre en oeuvre - beaucoup de connaissances à maîtriser - et elle ne supporte pas de notion similaire à celle de Composant Multivue.
- La méthode *UML Component* est limitée au niveau de la conception, car elle n'indique pas comment interpréter la spécification du composant dans l'implémentation, ni comment vérifier les spécifications du contrat au niveau du déploiement.
- L'identification des composants n'est généralement pas guidée, elle est basée sur l'expertise du concepteur. Seule la méthode CUP, qui inclut les activités d'identification de composants logiques, répond partiellement à notre objectif.

Après l'étude de ces méthodes, nous pensons que Processus Unifié et CUP sont deux méthodes complémentaires pour le développement à base de composants UML. C'est dans cette optique que nous nous sommes appuyés sur les points forts de ces méthodes. En effet, le Processus Unifié favorise le développement guidé par les cas d'utilisation et les itérations, il a l'avantage de recourir à la notation UML. D'autre part, la méthode CUP offre des étapes systématiques pour l'identification des composants logiques. De plus, dans la méthode CUP, un composant est un élément de conception qui doit être présent dans tout le cycle de développement du système. Au vu du résultat de ce travail comparatif qui met en évidence d'importantes lacunes au niveau méthodologique, nous visons l'établissement d'une démarche pour produire des modèles à base de Composants Multivue, en nous appuyant sur les points forts du Processus Unifié et de la méthode CUP.

3 Une démarche méthodologique pour l'identification et la définition de Composants Multivue

Dans cette section, nous proposons le cadre général de notre démarche qui étend le noyau existant dans le profil VUML. Pour ce faire, nous avons décomposé le processus de développement associé à cette démarche en trois phases structurées en étapes (cf. figure 3) qui sont détaillées dans les sections suivantes.

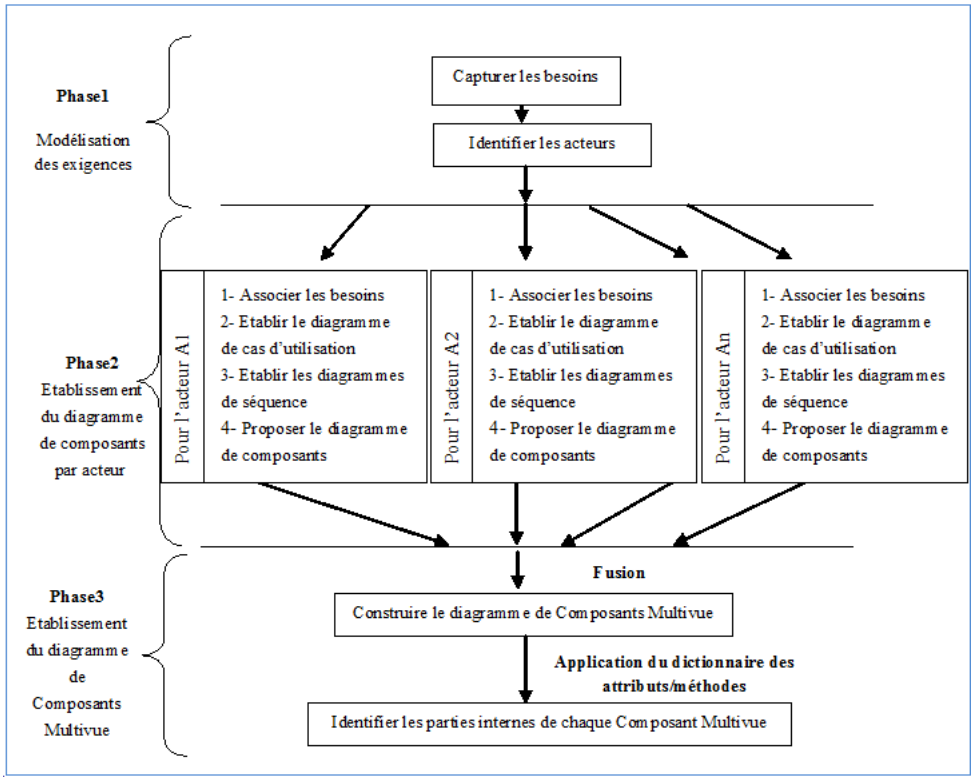

FIG. 3 – *Processus de développement d'un diagramme à base de Composants Multivue.*

3.1 Phase 1 : Modélisation des exigences

L'objectif principal de la modélisation des exigences est de spécifier le contexte métier pour bien répondre aux besoins des utilisateurs. Dans cette phase, nous nous concentrons donc sur les processus métiers. Cette phase est composée de deux étapes : la première est la capture des besoins, la deuxième étant l'identification des acteurs.

Capturer les besoins : l'objectif de cette étape est d'extraire les exigences et les contraintes du projet, en faisant la distinction entre les exigences fonctionnelles et non fonctionnelles. Les exigences fonctionnelles constituent la matière fondamentale pour la modélisation du système. En revanche, les exigences non fonctionnelles sont des propriétés que les fonctionnalités doivent garantir, comme la sécurité, la persistance, les performances, etc. Cette étape produit un document de capture des besoins dont le moyen d'expression est souvent le langage naturel (par exemple : texte, tableau, procédures, glossaire, etc.).

Identifier les acteurs : il s'agit d'identifier les acteurs qui interagissent avec le système. Chaque acteur aura un point de vue sur le système. Pour cet objectif, nous commençons par une analyse du document de capture des besoins dans le but d'identifier les acteurs. Une classification des acteurs est ensuite proposée pour faire la distinction entre les acteurs selon certains critères. Par exemple, acteur principal ou secondaire, acteur interne ou externe, etc.

3.2 Phase 2 : Établissement du diagramme de composants par acteur

L'établissement du diagramme de composants par acteur est focalisé d'abord sur l'identification de composants comme des éléments de conception. Cette phase se révèle souvent une tâche difficile mais déterminante. Elle est réalisée suivant la démarche CUP pour l'identification de composants logiques. Cette phase, qui consiste à élaborer un diagramme de composants pour chaque acteur (*point de vue*), est composée de quatre étapes :

Associer les besoins aux acteurs : nous associons les besoins fonctionnels - dégagés dans la phase 1- aux acteurs selon la spécificité de ces besoins.

Établir le diagramme de cas d'utilisation : il s'agit de formuler chaque besoin fonctionnel de chaque acteur avec un diagramme de cas d'utilisation utilisant les deux stéréotypes « LogicalComponent » et « ExternalUse ». Le stéréotype «LogicalComponent » concerne les paquetages de cas d'utilisation. Les cas d'utilisation sont organisés en paquetages considérés comme des composants logiques. Le stéréotype « ExternalUse » désigne les relations entre les différents composants logiques ; il permet d'exprimer le fait qu'un cas d'utilisation d'un composant nécessite un service d'un autre composant logique.

Établir les diagrammes de séquence : une fois les cas d'utilisation identifiés, cette étape consiste à spécifier les diagrammes de séquence des cas d'utilisation constituant chaque composant logique. Si un objet d'un composant source a besoin d'un service fourni par un objet d'un composant destination, on stéréotype l'objet qui réalise le service par «ExternalControl». Ce stéréotype permet d'identifier les interactions entre les objets des différents composants logiques.

Proposer le diagramme de composants : en utilisant les relations figurant dans les diagrammes de séquence établis dans l'étape précédente, il s'agit de réaliser une première version du diagramme de composants relatif au point de vue courant.

3.3 Phase 3 : Etablissement du diagramme de Composants Multivue

Cette phase consiste à établir un diagramme de Composants Multivue. Pour cet objectif, nous décomposons la phase en deux étapes.

Construire le diagramme général de Composants Multivue : dans cette étape, on établit une première version du diagramme de Composants Multivue. Pour ce faire, on se base sur une comparaison des différents diagrammes de composants issus de la phase précédente. Dans un premier temps, on cherche les composants figurant dans plus d'un diagramme de composants. Si un composant commun offre un service (interface fournie/requise) à plusieurs acteurs mais avec des méthodes de signatures différentes, on l'identifie comme un Composant Multivue avec le stéréotype «MultiviewComponent». L'interface qui présente le service est alors marquée avec le stéréotype «MultiviewInterface».

Identifier les parties internes de chaque Composant Multivue : Cette étape consiste à identifier les constituants internes de chaque Composant Multivue ; elle se déroule en considérant deux volets : le volet des attributs et celui des méthodes. Pour chaque volet on collecte les informations dans un dictionnaire. Ce dernier est un tableau dont les colonnes représentent les attributs ou les méthodes et dont les lignes représentent les acteurs.

Volet des attributs

On dresse tout d'abord le dictionnaire des attributs pour chaque Composant Multivue. Le tableau 1 présente le format du dictionnaire des attributs.

	Attribut 1	Attribut 2	attribut 3	...
Acteur 1	+	+		
Acteur 2		+		
Acteur 3		+	+	
...				

Légende : + signifie visible pour acteur

TAB. 1 – *Format du dictionnaire des attributs.*

	Attribut 1	Attribut 2	attribut 3	...
Acteur 1	+	+		
Acteur 2		+		
Acteur 3		+	+	
...				
View_base		*		
View_Acteur1	-			
View_Acteur2				
View_Acteur3			-	
...				

Légende : + visible par l'acteur,- visible par la vue acteur, * visible par la vue commune

TAB. 2 – *Format du tableau de marquage des attributs.*

Par la suite, nous procédons à une spécification des attributs communs ou spécifiques. Pour éviter les problèmes de polysémie, nous supposons que les attributs qui ont le même nom représentent la même information. De même nous supposons qu'il n'y a pas de problème d'homonymie. Cette hypothèse suppose qu'un travail préalable d'alignement ait été réalisé comme cela a été mentionné dans (Anwar et al., 2010) ; ce travail qui fait l'objet de plusieurs études mais qui n'est pas automatisable à l'heure actuelle - sort du cadre de cet article. Après le remplissage du dictionnaire des attributs, nous appliquons un algorithme de "marquage" sur ce dictionnaire dont le principe est donné ci-après :

- Les attributs qui portent le signe + dans plus d'une ligne d'acteur (Acteur1, Acteur2, Acteur3,...) sont des attributs communs, ils caractérisent la vue commune « View_base ». Ces attributs sont marqués par le signe "*".
- Par contre, les attributs qui caractérisent un acteur particulier se trouvent exclusivement dans sa vue. Par exemple, un attribut noté (Attribut1) visible par un seul acteur noté (ActeurK), se trouve uniquement dans la vue View_ActeurK. Ces attributs spécifiques sont marqués par le signe "-".
  Le résultat de cet algorithme est présenté dans un tableau de marquage. Le tableau 2 illustre le format du tableau de marquage des attributs.

Volet des méthodes

Pour les méthodes, on applique le même algorithme pour élaborer le dictionnaire, sauf que dans ce cas, les méthodes qui portent le même nom peuvent offrir plusieurs comportements. Autrement dit, une méthode peut exister dans la vue commune et être surchargée dans une ou plusieurs vues spécifiques. L'étape de comparaison des méthodes permettant d'affirmer qu'il s'agit bien d'une surcharge n'a pas été traitée dans notre étude. Le tableau 3 présente le format du tableau de marquage des méthodes.

	Attribut 1	Attribut 2	attribut 3	...
Acteur 1	+	+		
Acteur 2		+		
Acteur 3		+	+	
...				
View_base		*		
View_Acteur1	-	-		
View_Acteur2		-		
View_Acteur3		-	-	
...				

Légende : + visible par l'acteur, - visible par la vue acteur, * visible par la vue commune

TAB. 3 – *Format du tableau de marquage des méthodes.*

Nous précisons que la constitution des dictionnaires des attributs et des méthodes, ainsi que le remplissage du tableau de marquage des attributs et des méthodes se font manuellement.

4 Étude de cas

Notre démarche a été appliquée dans le projet DOME++ (Marzak et al., 2010), dans le cadre d'une convention CNRS-CNRST franco-marocaine. Dans ce projet, nous avons mis en oeuvre notre approche afin de développer des composants d'un observatoire multivue pour le Maroc, destiné en particulier à aider les décideurs du ministère de la santé publique en favorisant :

- l'accès équitable aux services de santé dans toutes les zones du Maroc,
- la prévention et la maîtrise des problèmes de santé identifiés,
- la prévention et la maîtrise de problèmes de santé dans des zones difficilement accessibles.

La présentation qui suit a un objectif de validation de notre approche ; elle est bien sûr un reflet très simplifié de la politique de santé considérée.

4.1 Phase 1 : Modélisation des exigences

Capturer les besoins : notre observatoire de la santé est organisé de telle façon que chaque utilisateur soit muni de droits d'accès aux différentes parties du système. Ces parties sont structurées par domaines ou catégories selon l'objectif ou le besoin du décideur. Par exemple, les données concernant les hôpitaux, celles qui intéressent les malades et ainsi de suite. L'administrateur a bien sûr des droits particuliers tels que la modification, l'affectation et la configuration du système. Les données du système seront exploitées à des fins diverses, notamment pour générer des rapports périodiques et produire des statistiques.

Identifier les acteurs : voici les acteurs principaux qui vont utiliser le système :

- L'administrateur du Ministère (MS) ;
- Le responsable d'un Centre Hospitalier Universitaire (CHU) ;
- Le médecin ;
- Le patient.

4.2 Phase 2 : Établissement du diagramme de composants par acteur

En raison de la limite d'espace, nous focalisons notre présentation dans la suite sur les deux acteurs suivants : l'« Administrateur du ministère » (acteur MS) et le « Responsable d'un CHU » (acteur CHU).

4.2.1 Établir le diagramme de composants pour l'acteur MS

Associer les besoins aux acteurs : l'acteur MS doit satisfaire les besoins suivants :

- Gérer les ressources humaines ;
- Consulter les hôpitaux ;
- Consulter la liste des médecins ;
- Consulter les fichiers des maladies.

Etablir le diagramme de cas d'utilisation associé à l'acteur MS (figure 4).

Le diagramme de cas d'utilisation contient six cas d'utilisation et trois composants logiques à savoir : Personnel, Hopital, Maladie. Nous relevons sur ce diagramme la relation stéréotypée «ExternalUse» entre Personnel et Hopital, car le cas d'utilisation GérerRH nécessite le cas Consulter_les_hôpitaux du composant logique Hopital.

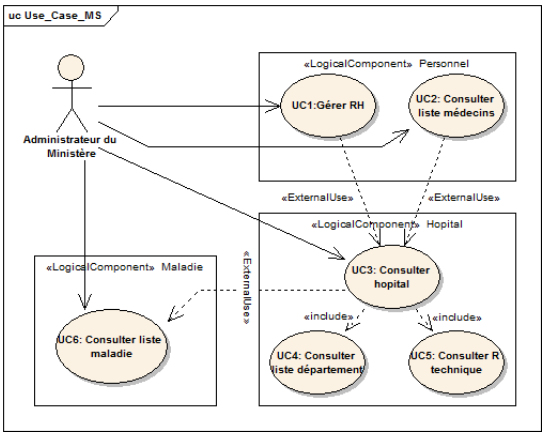

FIG. 4 – *Diagramme de cas d'utilisation associé à l'« administrateur du ministère »*

Établir les diagrammes de séquence :

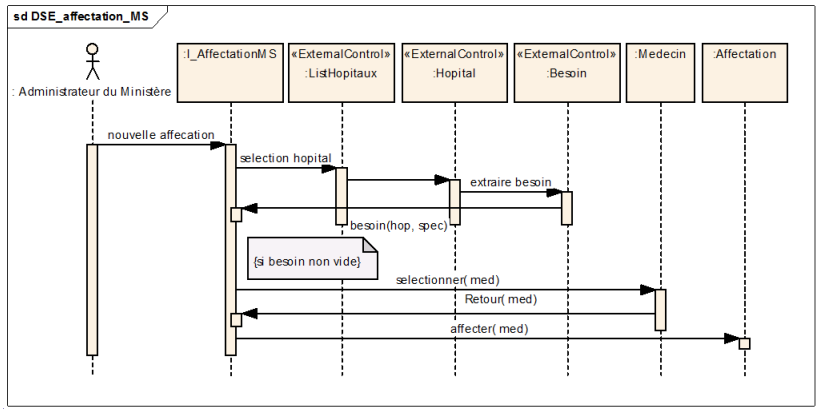

FIG. 5 – *Diagramme de séquence pour une affectation ministérielle.*

Nous proposons dans la figure 5 un diagramme de séquence pour le cas d'utilisation GérerRH (cf. figure 4).

A partir du diagramme de séquence dans la figure 5, nous identifions les interactions entre les objets de différents composants logiques (Renaux et al., 2004). Lors de l'affectation d'un nouveau médecin, l'administrateur du ministère appelle l'interface I_Affectation. Cette dernière fonctionne comme un gestionnaire de personnel. L'interface I_Affectation permet de sélectionner un hôpital parmi les hôpitaux pour charger les différentes demandes de CHU, puis de sélectionner le médecin à affecter. Si le médecin dispose des critères recherchés, le système enregistre l'affectation et la confirme dans une base d'affectation persistante intitulée Affectation.

Proposer un diagramme de composants associé à l'acteur MS :

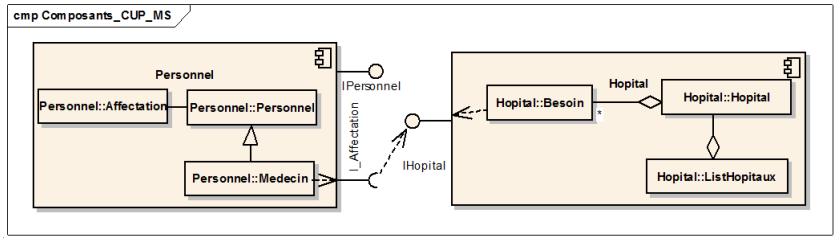

FIG. 6 – *Diagramme de composants du point de vue de l'acteur MS.*

Après avoir identifié les principaux messages circulant entre les objets des composants, nous pouvons proposer une première version du diagramme de composants du point de vue « administrateur du ministère ». Pour cet objectif, on se base sur le diagramme de cas d'utilisation (cf. figure 4) et le diagramme de séquence (cf. figure 5). Le diagramme de cas d'utilisations permet d'identifier les composants logiques Personnel et Hopital. Le diagramme de séquence précise les interactions entre les objets de différents composants logiques. A titre d'exemple, sur la figure 5, nous relevons que l'objet Besoin fournit un service Besoin (hop, spec) à l'objet I_AffectationMS du composant Personnel. Ceci permet de stéréotyper les objets Besoin, Hopital et ListeHopitaux par le stéréotype «ExternalControl ». Par conséquent, nous établissons une relation entre l'interface requise I_AffectationMS de composant Personnel et l'interface fournie IHopital du composant Hopital. La figure 6 présente le diagramme de composants associé à l'« administrateur du ministère ».

Le diagramme de composants du point de vue de l'acteur MS décrit deux composants : un composant Hopital qui expose une interface fournie IHopital et un composant Personnel avec une interface fournie IPesonnel et une interface requise I_Affectation.

4.2.2 Etablir le diagramme de composants pour l'acteur CHU

Pour le diagramme de composants associé au «responsable du CHU» on suit la même démarche.

Associer les besoins de «responsable du CHU»

Le «responsable du CHU» doit gérer les besoins suivants :

– Gérer les ressources humaines du CHU ;
– Consulter la liste des médecins ;

– Gérer les services ;
– Gérer les ressources matérielles ;
– Consulter les fichiers des maladies.

Établir le diagramme de cas d'utilisation associé à l'acteur CHU :

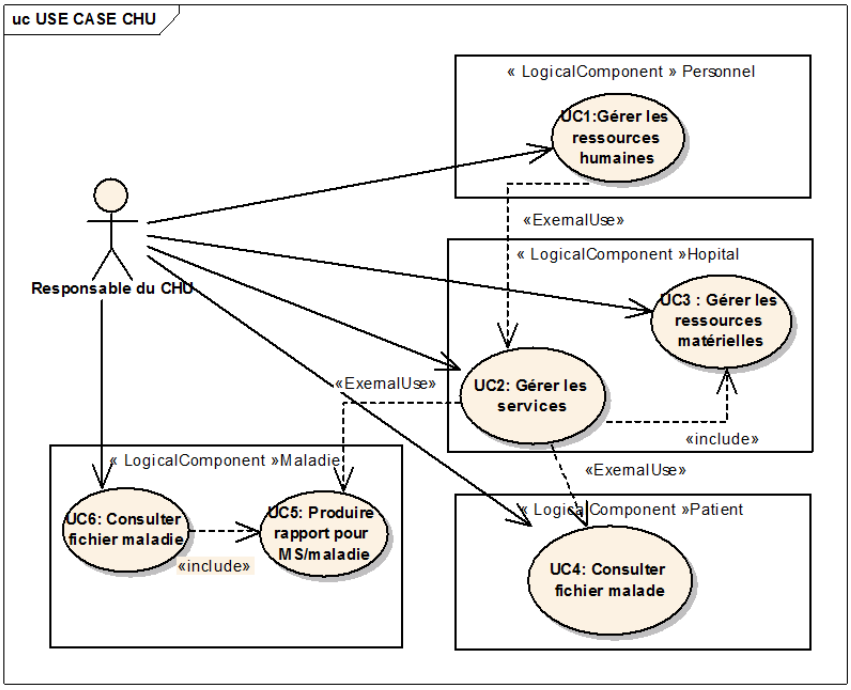

FIG. 7 – *Diagramme de cas d'utilisation associé au responsable du « CHU ».*

Après l'extraction des besoins de cet acteur, nous obtenons le diagramme de cas d'utilisation présenté sur la figure 7 décrivant au diagramme de cas d'utilisation. Ce diagramme de cas d'utilisation contient six cas d'utilisation qui sont répartis dans quatre composants logiques à savoir : Personnel, Hopital, Maladie et Patient.

Établir les diagrammes de séquence :

Après l'affectation ministérielle du personnel dans un CHU, le « responsable du CHU » lui assigne le poste et le service adéquat. Pour cela, il consulte les fichiers des patients et la liste du personnel dans chaque service pour voir les besoins. Selon les réponses issues de cette recherche multicritères, il examine le profil du personnel récemment recruté. Si le besoin est conforme au profil, le système enregistre le poste au niveau d'un service et le confirme. La figure 8 illustre le diagramme de séquence.

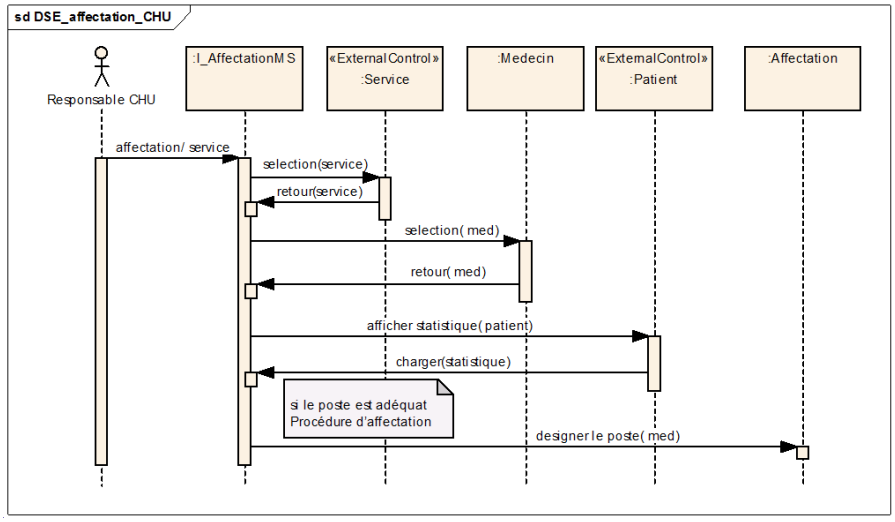

FIG. 8 – *Diagramme de séquence pour la gestion des ressources humaines.*

Proposer un diagramme de composants associé à l'acteur CHU :

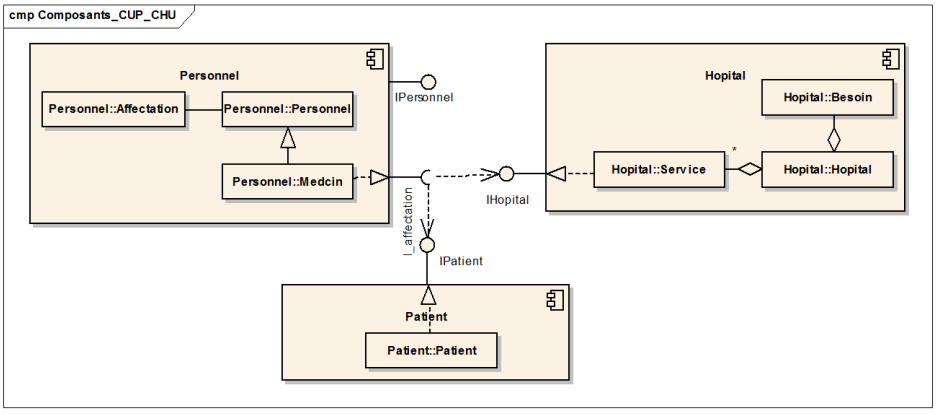

FIG. 9 – *Diagramme de composants du point de vue « responsable du CHU ».*

D'après le diagramme de cas d'utilisation et le diagramme de séquence, nous constatons que l'objet Service fournit la méthode retour (service) à l'objet I_AffectationMS du composant Personnel. Ceci permet de stéréotyper les objets Service et Patient par le «ExternalControl». Par conséquent, nous établissons une relation entre l'interface requise I_AffectationMS du composant Personnel et l'interface fournie IHopital du composant Hopital.

Le diagramme de composants de la figure 9 présente trois composants : un composant Hopital, un composant Personnel et un composant Patient. Le composant Hopital dispose d'une interface fournie IHopital. Le composant personnel offre une interface IPersonel et requiert une interface I_Affectation. Par contre, le composant Patient offre une seule interface IPatient.

4.3 Phase 3 : Etablissement du diagramme de Composants Multivue

Construire le diagramme général de Composants Multivue : pour établir le diagramme de Composants Multivue, on applique une « fusion manuelle » entre les différents diagrammes de composants issus de la phase précédente ; cela consiste à chercher les composants communs. Par exemple, le composant Personnel figure dans les deux diagrammes de composant précédents. Ce composant offre l'interface IPersonnel aux deux acteurs, sachant que chaque acteur demande des informations offertes différemment par cette interface. On en déduit que le composant Personnel est un Composant Multivue et que l'interface IPersonnel est une Interface Multivue. En revanche, le composant Patient se trouve seulement dans le diagramme de composant de l'acteur « responsable du CHU ». On peut en déduire que le composant Patient est un composant standard.

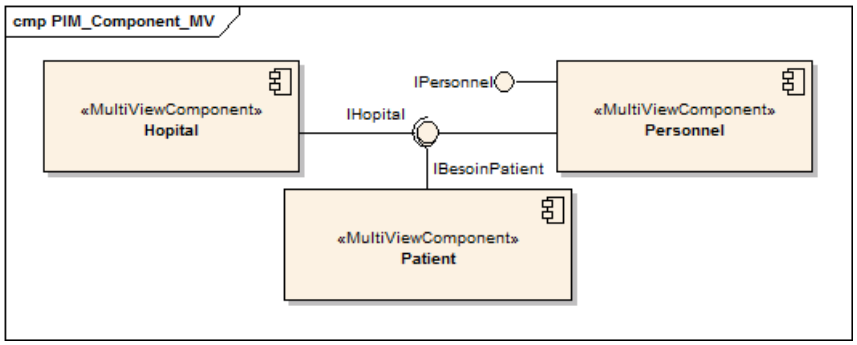

FIG. 10 – *Diagramme de Composants Multivue simplifié.*

La figure 10 illustre un diagramme de Composants Multivue dans un format compacté. Il comporte trois composants dont deux sont des Composants Multivue stéréotypés « Multiview-Component » Personnel, Hopital et un composant standard Patient.

Identifier les parties internes de chaque Composant Multivue : Un Composant Multivue publie ses méthodes dans des interfaces portant le stéréotype « baseInterface », partagées par les différents points des vues, et une spécification de point de vue acteur qui définit la spécificité de chaque acteur stéréotypé « viewInterface ». Dans ce sens, nous avons collecté les informations dans un dictionnaire de données afin de clarifier les besoins selon chaque point de vue. Pour illustrer la démarche nous avons choisi le Composant Multivue Personnel. Pour cela, nous présentons un tableau selon deux volets : attributs et méthodes.

Volet des attributs

Pour élaborer le tableau des attributs, nous appliquons l'algorithme de marquage présenté dans la section 3.3. Le tableau 4 présente le tableau obtenu.

	code	nom	prénom	adresse	échelle	service	spécialité	tache	dateVisa	visaDeDossier
Acteur/CHU	+	+	+	+	+	+	+	+		
Acteur/MS	+	+	+	+	+		+		+	+
View_base	*	*	*	*	*		*			
View_CHU						-	-			
View_MS									-	-

TAB. 4 – *Tableau de marquage des attributs.*

Selon la vue de l'acteur MS (respectivement CHU), un composant Personnel est caractérisé par les attributs qui portent le signe + selon la ligne Vue Acteur/MS (respectivement CHU). Donc, les attributs qui portent le signe + dans les deux lignes sont des attributs communs marqués par le signe *, par conséquent ils caractérisent le comportement d'une vue commune. Par contre, les attributs qui se trouvent dans une ligne d'un acteur exclusif caractérisent uniquement la vue de cet acteur. Ces attributs sont marqués par le signe - ; par exemple l'attribut service pour la vue de l'acteur CHU ou dateVisa pour la vue de l'acteur MS.

Volet des méthodes

	afficher()	mvt_national()	demander_att_sal()	demander_conge()	changer_service()
Acteur/MS	+	+	+		
Acteur/CHU	+			+	+
View_base	*				
View_MS	-	-	-		
View_CHU	-			-	-

TAB. 5 – *Tableau de marquage des méthodes.*

Concernant le volet des méthodes, nous appliquons la même démarche que celle des attributs, sauf que les méthodes qui portent le même nom peuvent offrir plusieurs comportements : un comportement commun et des comportements spécifiques à chaque acteur. Le tableau 5 présente le tableau de marquage des méthodes.

Par exemple, la méthode afficher() de l'interface commune «base» permet d'afficher les informations communes et elle change de traitement selon la vue active. En effet, pour la vue MS, cette méthode permet d'afficher en plus des informations de base, la dateVisa et le visaDeDossier. Pour la vue CHU, elle s'occupe de l'affichage du service, de la spécialité et de la tâche. Le diagramme de Composants Multivue est schématisé dans la figure 11 dans un format expansé (*montrant la base et les vues*).

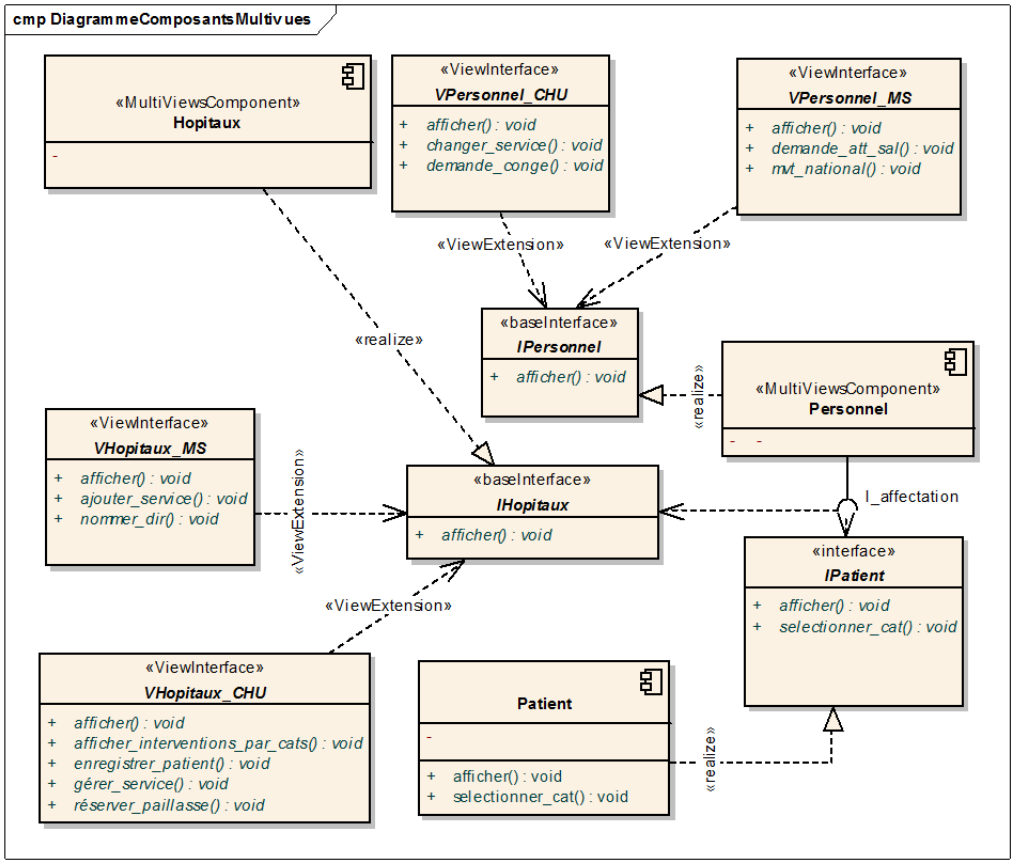

FIG. 11 – *Diagramme de Composants Multivue.*

Ce diagramme de Composants Multivue contient deux Composants Multivue (Hopital et Personnel) et un composant standard (Patient). Chaque Composant Multivue dispose au moins d'une interface Multivue (fournie/requise). Par exemple, le Composant Multivue Hopital offre l'interface Multivue IHopital. Cette interface est composée d'une base intitulée IHopital stéréotypée «baseInterface» et deux interfaces vues stéréotypées «ViewInterface» VHopital_MS et VHopital_CHU qui sont associés respectivement à l'«administrateur du ministère» et au «responsable du CHU».

Le diagramme de Composants Multivue obtenu à l'issue de cette étude de cas est un modèle conforme au profil VUML ; ce modèle est exploitable de plusieurs manières. Le processus de conception proposé supporte l'évolutivité (*ajout/suppression de nouveaux points de vue*). Ce modèle peut être considéré comme un PIM (*Modèle indépendant de Plate-forme*) pour des éventuelles transformations vers des plates-formes spécifiques.

4.4 Discussion

Cette étude de cas n'étant pas complètement terminée, nous ne pouvons pas fournir d'évaluation chiffrée concernant le nombre définitif de composants, de diagrammes de séquence, etc. En effet, nous sommes en train de prendre en compte d'autres points de vue, notamment ceux du médecin et du patient. Cependant, notre démarche de conception par points de vue étant incrémentale, nous pouvons d'ores et déjà tirer des enseignements significatifs de notre étude de cas. Nous pouvons ainsi apporter des éléments de discussion relativement à deux points importants : le passage à l'échelle et l'alignement sémantique des composants.

- Passage à l'échelle : d'un point de vue théorique, l'approche développée dans cet article passe à l'échelle car la conception des composants multivue peut se faire itérativement en fusionnant initialement deux composants par points de vue, puis en intégrant progressivement les composants deux à deux. Dans (Dkaki et al., 2011), on peut trouver une formalisation de la fusion de diagrammes de classes comme un opérateur binaire sur les modèles, doté de la propriété d'associativité. Il resterait à transposer cette formalisation dans le contexte de fusion des composants en sachant que la problématique théorique est la même et que la complexité d'élaboration des composants multivue est du même ordre que celle des diagrammes de classe multivue. En outre, d'un point de vue pratique, les diverses expérimentations que nous avons effectuées sur la composition de diagrammes de classes avec VUML ont montré que l'approche passe à l'échelle.
- Alignement des composants : cette question est bien sûr importante et nous ne l'ignorons pas mais notre approche ne traite pas à proprement parler l'alignement sémantique des composants. En effet, nous considérons que ce travail relève en partie de l'ingénierie des exigences en faisant appel à des résultats sur l'alignement d'ontologies. En pratique, nous supposons l'existence d'une étape amont éliminant les polysémies et homonymie et produisant un dictionnaire (glossaire) partagé par les différents concepteurs. Nous pensons que cette étape qui fait largement appel à l'intelligence d'experts des domaines fonctionnels associés aux différents points de vue - n'est pas complètement automatisable à ce jour, surtout dans un contexte applicatif réel et de grande envergure, même si elle peut assister utilement les concepteurs. En cas d'ajout d'un point de vue a posteriori, il faut de ce fait que la terminologie employée respecte ce glossaire partagé.

5 Conclusion

Dans cet article, nous avons présenté une démarche méthodologique pour le développement par Composants Multivue, en ciblant plus particulièrement la phase de conception. Cette démarche s'appuie sur le langage VUML qui est un profil UML intégrant la notion de Composant Multivue. D'un point de vue méthodologique, cette démarche s'appuie sur le Processus

Unifié qui est une référence pour les méthodes orientées UML et sur la méthode CUP (*Component Unified Process*) qui est une méthode pour le développement à base de composants.

Après avoir rappelé le concept de Composant Multivue et présenté brièvement un travail comparatif sur les méthodes de développement de systèmes d'information à base de composants, nous avons décrit une démarche méthodologique pour l'identification de Composants Multivue. La démarche gère les Composants Multivue tout au long du cycle de développement, elle offre également un enchainement d'étapes clair et concis. Pour valider notre travail, nous avons appliqué notre approche à une étude de cas issue de besoins réels concernant le domaine de la santé au Maroc.

La démarche présentée dans cet article cible essentiellement l'aspect statique (structurel) du développement par Composants Multivue. On peut ainsi identifier certaines limites de cette approche qui constituent autant de perspectives pour nos travaux :

- il faudrait inclure l'aspect comportemental associé à la construction du diagramme de Composants Multivue. Ce travail a déjà été mené dans notre équipe dans VUML (Ober et al., 2008) à travers la composition de machines à états, mais sans aborder spécifiquement la notion de composant multivue.
- il faudrait développer les outils d'aide graphiques pour supporter la démarche (la représentation et l'assemblage de Composants Multivue).
- enfin, il faudrait intégrer la reconstruction du contrat d'assemblage de Composants Multivue pour assurer la cohérence du système.

Actuellement, nous travaillons sur le passage de la notion de PIM à celle de PSM par l'intermédiaire d'un processus IDM. Une génération de code J2EE a déjà été réalisée sous forme d'une transformation opérationnelle écrite en ATL.

Une autre perspective de ce travail est l'automatisation partielle de l'étape de fusion proposée dans notre démarche. Cette automatisation nous paraît similaire dans son principe à celle que nous avons réalisée pour fusionner des diagrammes de classes (Anwar et al., 2010).

Remerciements : Ce travail a été partiellement financé par la convention de recherche DOME++ 21589 entre le CNRST marocain et le CNRS français de 2008 à 2010.

Références

Anwar, A., S. Ebersold, B. Coulette, M. Nassar, et A. Kriouile (2010). A rule-driven approach for composing viewpoint-oriented models. *ETH Swiss Federal Institute of Technolog. Journal of Object Technology 9*(2), 89–114.

Cheesman, J. et J. Daniels (2000). *UML components : A Simple Process for Specifying Component-Based Software*. Addison-Wesley.

Coulette, B., A. Kriouile, et S. Marcaillou (1996). L'approche par points de vue dans le développement orienté objet des systèmes complexes. *Revue l'Objet 2*(4), 13–20.

Dkaki, T., A. Anwar, S. Ebersold, B. Coulette, et M. Nassar (2011). A formal approach to model composition applied to vuml. In *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on*, pp. 188–197. IEEE Computer Society.

D'Souza, D. et A. Wills (1998). Objects, components, and frameworks with uml : The catalysis (sm) approach, edition addison-wesley.

El Asri, B., M. Nassar, A. Kriouile, et B. Coulette (2005). Multiviews component for development. *Proceedings ICEIS'05 Miami-USA*, 124–28.

Finkelstein, A., J. Kramer, et M. Goedicke (1990). Viewpoint oriented software development. In *Proc. Conf" Le Génie Logiciel et ses Applications", Toulouse*, pp. 337–351.

Hain, M., A. Marzak, et T. Ouahmane (2007). Vers un pim à base de composants multivue.

Jacobson, I., G. Booch, et J. Rumbaugh (1999). The unified software development process. *Reading : Addison Wesley*.

Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, et J. Irwin (1997). Aspect-oriented programming. *ECOOP Object-Oriented Programming*, 220–242.

Kriouile, A. (1995). Vboom, une méthode orientée objet d'analyse et de conception par points de vue. *thèse d'Etat de l'université Mohammed V de Rabat, Maroc*.

Kruchten, P. (2004). *The rational unified process : an introduction*. Addison-Wesley Professional.

Marzak, A., M. Hain, et B. Coulette (2010). Développement d'observatoires multivue, rapport de recherche.

Muller, A., O. Caron, B. Carré, G. Vanwormhoudt, et S. Bouzitouna (2007). Ingénierie multi-modèles : Projection flexible d'assemblages de modèles. *LMO'07*, 167–82.

Nassar, M. (2005). Analyse/conception par points de vue : le profil vuml. *Thèse de doctorat à l'Institut National Polytechnique de Toulouse, Numéro d'ordre : 2252*.

Nassar, M., B. Coulette, X. Crégut, S. Marcaillou, et A. Kriouile (2003). Towards a view based unified modeling language. In *Proceedings of 5th International Conference on Enterprise Information Systems ICEIS'03*, Volume 3, pp. 257–265.

Ober, I., B. Coulette, et Y. Lakhrissi (2008). Behavioral modelling and composition of object slices using event observation. *Proceedings ACM/IEEE international conference MODELS, Toulouse, October 1-3*, 219–233.

OMG (2003a). Mda guide version 1.0.1, document number : omg/2003-06-01 edition. object management group, 2003.

OMG (2003b). Uml 2.0 : Superstructure specification. http ://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/.

Ossher, H., M. Kaplan, A. Katz, W. Harrison, et V. Kruskal (1996). Specifying subject-oriented composition. *Theory and Practice of Object Systems 2*(3), 179–202.

Renaux, E., O. Caron, et J. Geib (2004). Chaîne de production de systèmes à base de composants logiques. *Numéro spécial de la revue l'Objet 10*(2-3), 147–160.

Summary

VUML (*View based UML*) is an object-oriented modeling language that introduces the concepts of Multiview class and Multiview component. However, like UML, VUML does not provide any process to build the Multiview components diagram. To address this issue, we have reused UP (*Unified Process*) and CUP (C*omponent Unified Process*) to propose a process for

developing Multiview components based PIMs (*Platform Independent Models*). In this article, we describe the process' stages allowing to produce the Multiview Components diagram.