HAL open science

Ingénierie multi-modèles : Projection flexible d'assemblages de modèles

Alexis Muller, Olivier Caron, Bernard Carré, Gilles Vanwormhoudt, Salim Bouzitouna

To cite this version:

Alexis Muller, Olivier Caron, Bernard Carré, Gilles Vanwormhoudt, Salim Bouzitouna. Ingénierie multi-modèles : Projection flexible d'assemblages de modèles. Journées Langages, Modèles, Objets (LMO'07), Mar 2007, Toulouse, France. pp.167-182. hal-00714150

HAL Id: hal-00714150

https://hal.science/hal-00714150v1

Submitted on 13 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Authors' manuscript. To cite this paper :
« Ingénierie multi-modèles : Projection flexible d'assemblages de modèles », Alexis Muller, Olivier Caron, Bernard Carré, Gilles Vanwormhoudt, Salim Bouzitouna. Actes de LMO (Langages et Modèles à Objets), 2007, Toulouse, France. Hermès Lavoisier Ed., pp. 167–182.

Ingénierie multi-modèles : Projection flexible d'assemblages de modèles

Alexis Muller[**] — Olivier Caron[*] — Bernard Carré[*] — Gilles Vanwormhoudt[*,*] — Salim Bouzitouna[**]**

[*] *Laboratoire d'Informatique Fondamentale de Lille, UPRESA CNRS 8022*
Université des Sciences et Technologies de Lille
[**] *Laboratoire d'Informatique de Paris 6, CNRS UMR7606*
Université Pierre et Marie Curie
[***] *GET/ENIC Telecom Lille I*
{Olivier.Caron, Bernard.Carre, Gilles.Vanwormhoudt}@lifl.fr
{Alexis.Muller, Salim.Bouzitouna}@lip6.fr

RÉSUMÉ. *L'ingénierie dirigée par les modèles place les modèles au cœur des processus de génie logiciel. Outre leur dimension originelle de spécification de systèmes ou de parties de systèmes, les modèles deviennent des éléments manipulables, vérifiables, capitalisables, exécutables, transformables…*
Le modèle devient l'élément principal de vérification, de réalisation, d'évolution du système. Cependant, la conception d'un modèle de système n'est pas une tâche facile. En effet, les systèmes étant complexes, leurs modèles le sont également. Pour maîtriser cette complexité, il est possible d'utiliser les stratégies habituelles de structuration et de réutilisation. La conception d'un modèle de système peut ainsi être réalisée par la composition d'un ensemble de modèles issus de différentes sources. C'est ce que nous appelons l'ingénierie multi-modèles. Nous présentons notre approche à base de composants de modèles permettant de concevoir un système par assemblage d'un ensemble de modèles. Et nous proposons différents modes d'expression du modèle résultat et différentes stratégies de projections permettant de préserver ou non la structuration introduite par cet assemblage.

ABSTRACT. *The model driven engineering put models on the heart of software engineering. Beyond their original dimension of systems or system parts specification, models are gaining the status of verifiable, capitalizable, executable and transformable full software artifacts. Model becoming the main element of verification, realization, evolution. Indeed, systems being complex, their models are complex too. To control this complexity, usual structuring and reusing strategies can be used. A system model can be constructed by composing a set of models from different sources. This is what we call multi-models engineering. We expose here our model components based approach. And we propose different expressing modes of the resulting model and different targeting strategies allowing to preserve or not the introduced structuring by this assembly.*

MOTS-CLÉS : *modèles objets paramétrés, composition de modèles, fusion, vues, IDM, transformation, projection*

KEYWORDS: *parameterized models, model composition, merging, views, MDE, transformation, targeting*

1. Introduction

L'ingénierie dirigée par les modèles (IDM ou Model Driven Engineering MDE) [KEN 02] constitue l'étape la plus récente dans l'évolution constante de l'importance des modèles dans les processus de génie logiciel.

Les modèles ont été dans un premier temps employés pour concevoir, représenter et documenter les futurs systèmes informatiques [BOO 94, JAC 92, RUM 91]. Durant le processus de génie logiciel, les modèles n'intervenaient que dans les premières étapes d'analyse et de conception. Ces modèles formaient un support papier à la production de code et n'étaient plus exploités par la suite.

L'approche MDA [MIL 01, POO 01, MUL 02] a significativement renforcé l'importance et l'exploitation des modèles. D'une part, en capitalisant les spécifications fonctionnelles des systèmes permettant ainsi de les réutiliser pour leur mise en œuvre sur différentes plates-formes. D'autre part, en proposant une structuration verticale du système selon des niveaux d'abstraction représentés par des modèles. Dans MDA, tout est modèle : les modèles métiers capitalisables indépendants des plates-formes et les modèles spécifiques aux plates-formes. MDA a accentué les efforts de transformation de modèles selon une démarche de raffinement entre niveaux d'abstraction [BLA 04].

L'approche IDM généralise l'exploitation des modèles et place véritablement ceux-ci au cœur des processus de génie logiciel. Outre leur dimension originelle de spécification de systèmes ou de parties de systèmes, les modèles deviennent :

– *des éléments manipulables* grâce notamment à l'emploi de techniques de méta-modélisation (MOF ou Eclipse EMF par exemple) ou de sérialisation de modèles (XMI par exemple).

– *des élements vérifiables* à l'aide, par exemple, de logiques de description [MEN 06], de spécifications formelles (B, OCL, ...) [KLE 03].

– *des éléments capitalisables* où un modèle peut être réutilisé pour différentes plates-formes mais également pour différentes applications. Le concepteur dispose alors de bibliothèques de modèles [ANC 05].

– *des éléments exécutables* [BAL 02, SUN 02].

– *des éléments transformables* permettant, par exemple, d'obtenir les modèles spécifiques à une plate-forme à partir d'un modèle métier.

L'approche IDM, place donc le modèle du système au cœur de sa conception. Celui-ci devient l'élément principal de vérification, de réalisation, d'évolution du système. Cependant, la conception d'un modèle de système n'est pas une tâche facile. En effet, les systèmes étant complexes, leurs modèles le sont également. Pour maîtriser cette complexité, il est possible d'utiliser les stratégies habituelles de structuration et de réutilisation. La conception d'un modèle de système peut ainsi être réalisée par la composition d'un ensemble de modèles issus de différentes sources. C'est ce que nous appelons l'ingénierie multi-modèles et que nous présentons dans la section suivante. Nous décrivons ensuite dans la section 3 notre approche permettant l'expression de

composants de modèle génériques et de leur assemblage pour la conception de modèles de système. Nous verrons section 4 que le résultat d'un tel assemblage peut s'exprimer selon différents modes. Le caractère multi-modèles des systèmes ainsi conçus peut avoir un sens qu'il est intéressant de conserver jusqu'à sa mise en œuvre, c'est ce que nous proposons section 5. Enfin la section 6 présente notre prototype d'outil supportant l'ensemble de notre approche.

2. Ingénierie Multi-Modèles

Plusieurs sources de modèles peuvent être utilisées pour la conception d'un système. Différentes parties du même système peuvent, par exemple, être modélisées séparément voire par des équipes différentes.

La décomposition fonctionnelle ou horizontale [D'S 99, VAN 99, CLA 01] est une approche utilisée afin de permettre une meilleure maîtrise de la complexité d'un système et de son évolution. Elle consiste à structurer un système selon ses différentes fonctionnalités (gestion des ventes, gestion des stocks, marketing,...par exemple). Cette démarche utilisée dès la phase de conception du système constitue une source de multi-modèles, chaque modèle représentant une fonctionnalité différente. Les approches de modélisation par vues ou par aspects [STR 04, MUL 03] peuvent également être utilisées pour la conception de modèles de système. Ces derniers sont alors conçus par composition d'un ensemble de vues ou aspects, chacun étant exprimé à l'aide d'un modèle.

Les besoins d'une ingénierie multi-modèles se révèlent également dans le domaine de la fédération de systèmes [SHE 90, DEB 98] ou plus généralement de l'intégration de données [BER 03]. En effet l'architecture type d'une fédération repose sur l'exportation "ascendante" de modèles issus des composants existants (patrimoniaux) qu'il s'agit d'aligner, d'intégrer et de transformer selon un (méta)modèle pivot de la fédération. Il est possible également ensuite d'étendre la fédération par enrichissement du modèle fédéré ou intégration de nouveaux composants par leur modèle, de façon "descendante".

Enfin, plusieurs travaux visent à constituer des bibliothèques de modèles réutilisables telles que les paquetages template UML 2, les modèles de patrons [SUN 00], les modèles de frameworks[WIL 96] ou encore la réutilisation de PIM (Platform Independent Model) et PSM (Platform Specific Model) dans le cadre du MDA [BOU 05]. Ces bibliothèques offrent des modèles génériques qu'il s'agit ensuite d'appliquer ou d'adapter au contexte applicatif du système à concevoir.

Toutes ces approches constituent autant de sources de modèles exploitables pour la conception d'un système. Se pose alors le problème de la combinaison des modèles retenus afin d'obtenir un tout cohérent. Ces combinaisons peuvent prendre des formes différentes suivant que l'on souhaite ou non préserver la structuration introduite par les différents modèles. Il est par exemple imaginable de fusionner les différents modèles fonctionnels ou d'exprimer la composition de deux modèles de système en vue de

leur fédération. Ceci fait apparaître la nécessité de spécifier un assemblage logique des modèles constituants du système indépendant des mises en œuvres ultérieures.

Nous allons nous concentrer ici sur le ciblage d'un tel assemblage. Nous proposons une démarche illustrée figure 1 qui repose sur une décomposition en plusieurs niveaux. Elle s'appuie sur la notion de composants de modèles [MUL 05] que nous rappelons en section 3. Le premier niveau permet l'expression de l'assemblage logique et la vérification de sa cohérence. Le niveau suivant (cf section 4) permet d'obtenir le modèle résultat du système en sélectionnant différents modes de combinaison (fusion, structuration en vues). On obtient une démarche de transformation paramétrée guidée par des critères de projection (préservation de l'existant, répartition, modularité, efficacité,. . .). Les derniers niveaux (cf section 5) sont dédiés aux ciblages technologiques en ayant recours à un ensemble de patrons [CAR 03, CAR 05].

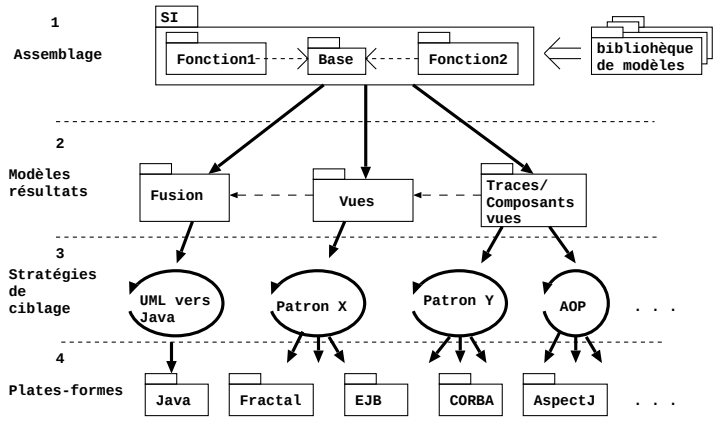

Figure 1 – Chaînes de production

3. Composants de modèle

L'objectif de notre approche est de permettre la construction de systèmes par assemblage de fonctionnalités, chacune décrite à l'aide d'un modèle générique appelé composant de modèle [MUL 06]. Dans notre approche, ces modèles sont eux-mêmes paramétrés par un modèle afin de permettre l'expression d'une partie requise complexe. On dépasse ainsi la notion de contrat d'assemblage de composants souvent réduite à une interface de services unitaires.

La fonctionnalité exprimée par un composant peut alors être ajoutée à un modèle par un mécanisme d'application. Ce mécanisme consiste à mettre en relation le modèle requis par le composant avec un modèle conforme auquel la fonctionnalité doit être ajoutée. Ce mécanisme fonctionne aussi bien pour l'application à un modèle de système qu'à un autre composant de modèle, permettant ainsi la construction de fonctionnalités complexes à partir de fonctionnalités plus simples. La conception d'un

système peut alors être réalisée par l'assemblage d'un ensemble de composants de modèle.

Les figures 2, 3 et 4 illustrent un ensemble de composants de modèle, exprimés sous la forme de paquetages. Le modèle requis de chacun d'eux est explicité dans le coin supérieur droit. Le schéma figuré à l'intérieur du paquetage de chaque composant correspond à son modèle fourni. Celui-ci est paramétré par le modèle requis qui y est ici représenté à l'aide des éléments en pointillés et italique.

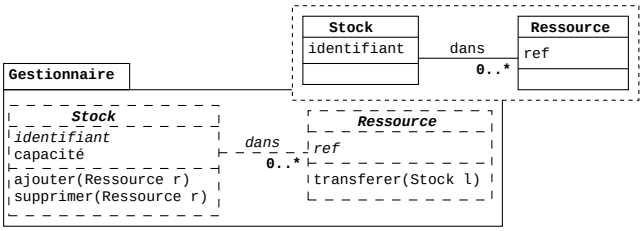

Figure 2 – Composant de gestion de ressources

La figure 2 illustre un composant pour une fonctionnalité de gestion de ressources. Celui-ci est défini à partir des concepts de stock et de ressource exprimés dans son modèle requis. Le concept de stock doit disposer d'un attribut matérialisant son identifiant et celui de ressource d'un attribut matérialisant sa référence (*ref*). Ces deux concepts doivent également être reliés par une association (*dans*). La fonctionnalité de gestion des stocks est donc exprimée à partir de ce modèle et définit des opérations d'ajout, de suppression et de transfert de ressources. Un attribut permettant de définir la capacité dans stock est également ajouté.

Le composant d'allocation de ressources de la figure 3 illustre la possibilité d'enrichir la structure d'un modèle par l'ajout de nouvelles classes ou associations. Pour appliquer ce composant, il est nécessaire de disposer de produits et de clients, les classes correspondantes à ces concepts pouvant être totalement dissociées. Ce composant a pour but d'allouer des produits à des clients et installe pour cela une relation entre ces entités. Ainsi, la classe *Allocation* est ajoutée entre les produits et clients à l'aide de deux associations. Cette classe dispose d'attributs pour la date d'allocation, la date de retour prévue et la date de retour effective, et d'une opération permettant le calcul du coût de l'allocation. Des opérations de gestion des allocations sont également ajoutées aux classes *Produit* et *Client*.

Enfin, notre dernier exemple de composant de modèle illustre leur application aux patrons de conception. La figure 4 définit ainsi un composant de modèle exploitant le patron observateur. Celui-ci définit la structure de ce patron par rapport à son modèle requis, composé de la classe *Sujet* disposant d'un état et d'une classe *Observateur* notifiée du changement de ce dernier.

Partant d'un ensemble de composants de modèle généraux issus de sources diverses, la construction d'un système peut alors être réalisée par leur assemblage. Cet assemblage doit se baser sur un modèle initial du système. Celui-ci, appelé modèle

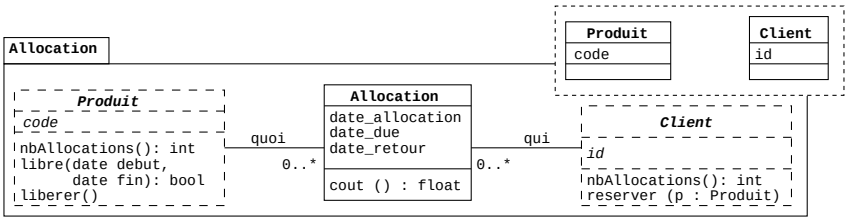

Figure 3 – Composant d'allocation de ressources

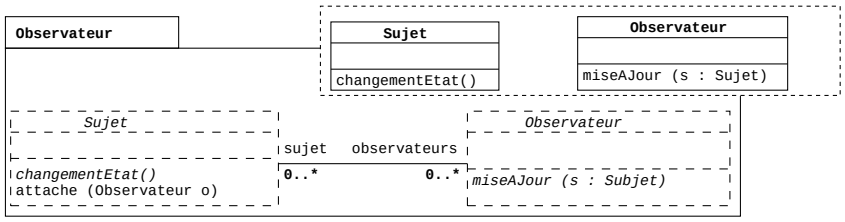

Figure 4 – Composant observateur

de base (ou "modèle primaire" dans [STR 04]), définit les concepts et les associations propres au domaine d'application.

Pour spécifier cet assemblage, nous avons défini un opérateur d'application de modèles paramétrés [MUL 05]. Cet opérateur permet de mettre en correspondance le modèle requis par un composant avec le modèle fourni par un autre (ou par la base). Le modèle fourni est conforme au modèle requis s'il présente un sous-modèle qui respecte la structure du modèle requis. C'est cette notion de conformité entre modèle fourni et modèle requis qui permet de garantir la validité d'une composition. Ceci est garanti par un jeu de contraintes qui est détaillé dans [MUL 06].

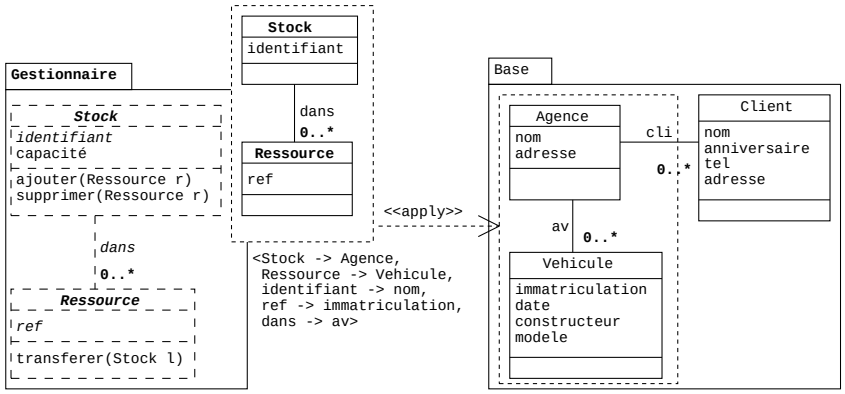

Figure 5 – Application de gestion de ressources au système de location de véhicules

La figure 5 donne un exemple d'application du composant de modèle "gestionnaire" à un système de location de véhicules pour lui ajouter la fonctionnalité de gestion de stocks. Cette application est spécifiée à l'aide de la relation stéréotypée `«apply»` qui établit les relations de correspondance entre le modèle requis et le modèle fourni. Les classes *Stock* et *Ressource* sont respectivement mises en correspondance avec les classes *Agence* et *Vehicule*. De même, les attributs *identifiant* et *ref* sont mis en relation avec les attributs *nom* et *immatriculation*.

Comme nous le présentons section 4 le modèle résultat d'un tel assemblage peut s'exprimer selon différents modes, en fusionnant, par exemple, le composant de modèle et la base ou en utilisant une représentation à l'aide de vues.

Pour supporter l'évolution incrémentale des systèmes, l'approche permet l'application d'un composant de modèle à un autre. Cette possibilité permet la construction de nouveaux composants de modèle complexes à partir de composants plus simples. Elle permet également l'expression de chaînes d'applications. Une telle chaîne est illustrée dans le modèle d'assemblage de la figure 6 par l'application du composant *Observateur* au composant *Allocation*, lui-même appliqué à la base. Cette première application permettant d'ajouter une fonctionnalité de réservation automatique dès qu'un véhicule devient disponible.

L'approche autorise également l'utilisation d'un même composant pour différentes parties du même système. Cette possibilité est ici illustrée par l'application du composant *Gestionnaire* pour la gestion des véhicules et pour la gestion des clients.

Finalement, la figure 6 présente un assemblage de modèles composés autour d'un modèle initial du système à construire. Un ensemble de propriétés définies pour l'opérateur *apply* permet de garantir la cohérence de l'assemblage quelque soit l'ordre de composition [MUL 05].

4. Modèles résultats

Nous présentons dans cette section deux modes d'obtention d'un modèle de système, et la façon de les obtenir à partir d'un assemblage. Le premier est basé sur la fusion des différentes fonctionnalités avec le modèle initial et permet d'obtenir un modèle à objets classique. Le second mode permet d'obtenir une structuration par vues.

Afin d'illustrer ces modes d'obtention, nous utiliserons une partie de l'assemblage illustré figure 6 : l'application du modèle *Gestionnaire* pour la gestion des stocks de véhicules et du modèle *Allocation* au modèle de base.

4.1. *Fusion*

La représentation la plus directe pour obtenir le système issu d'un assemblage, consiste à fusionner les éléments définis par les différents composants dans un modèle unique. Tous les éléments propres à la fonctionnalité, c'est-à-dire ne faisant pas partie

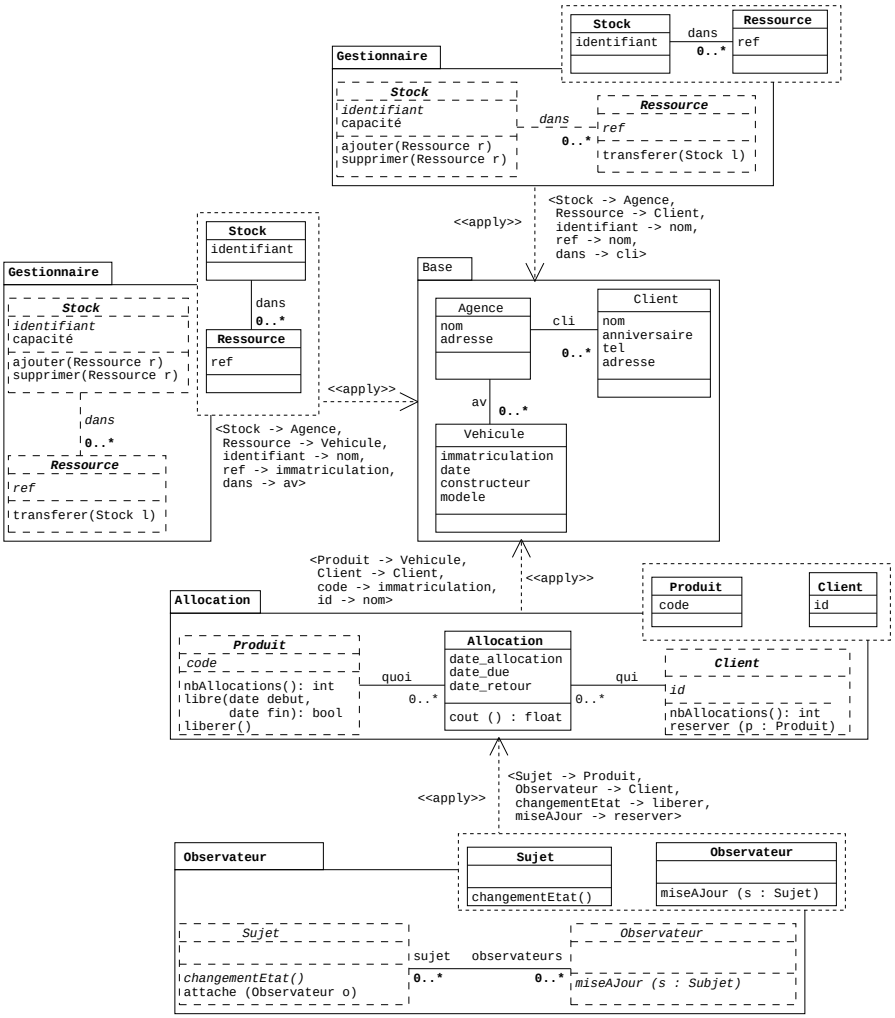

Figure 6 – Assemblage du système de locations de véhicules

du modèle paramètre, sont ajoutés au modèle auquel le composant est appliqué. Les attributs et opérations introduits par les composants sont ajoutés aux classes correspondantes. De même, les classes et associations introduites par les composants sont ajoutées au modèle. Dans un composant, une opération peut être définie à l'aide de paramètres ou une valeur de retour typés par des éléments appartenant au modèle requis. Dans ce cas, ceux-ci doivent être typés dans le modèle fusionné par les éléments du modèle initial correspondant.

Le modèle ainsi obtenu est un modèle classes-associations qui pourra être directement mis en œuvre avec tout langage ou plate-forme à objets. Ce mode de représen-

tation par fusion est illustré par la figure 7 pour le système de location de véhicules conçu à partir de notre assemblage. L'attribut *capacite* et les opérations *ajouter* et *supprimer* définis pour un stock sont ajoutés à l'entité *Agence*. La classe *Vehicule* est quant à elle enrichie de l'opération *transferer* issue du composant *Gestionnaire* et des opérations (*nbAllocations*, *libre* et *liberer*) issues du composant *Allocation*.

Cet exemple permet également d'illustrer le typage cohérent des paramètres et valeurs de retour des opérations ajoutées aux entités. L'opération permettant l'ajout d'une ressource dans un stock (*ajouter (r : Ressource)*) est bien définie dans le modèle résultat comme une opération permettant l'ajout d'un véhicule à une agence (*ajouter (r : Vehicule)*).

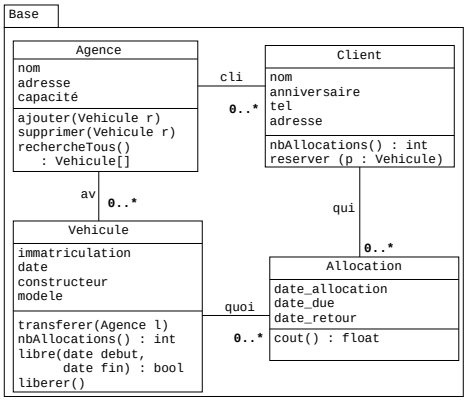

Figure 7 – Fusion des différentes fonctionnalités

Des éléments issus de différents composants étant fusionnés dans un modèle unique, des conflits apparaissent si des éléments de même nom doivent être ajoutés au même espace de nommage. En effet, dans un modèle tel que présenté figure 7, l'identification des éléments se fait par leur nom et l'élément qui les contient. Il peut alors être nécessaire de mettre en place une stratégie de renommage des éléments à fusionner comme dans [STR 04] ou d'utiliser des mécanismes de priorités comme dans [BOU 04].

Cette représentation permet d'exprimer le système selon un modèle objet conventionnel, mais elle ne permet pas de préserver la structuration de ce système selon ses différentes fonctionnalités. C'est cette possibilité que nous proposons d'étudier dans la section suivante.

4.2. *Structuration par vues*

La préservation de la structuration du système introduite lors de sa conception a pour but de répondre aux besoins de traçabilité, de maîtrise de la complexité ou encore d'évolution. Nous présentons ici un moyen d'obtention du modèle résultat d'un assemblage selon une structuration par vues [MUL 03]. Dans une représentation à base de vues, les entités des différentes vues sont mises en correspondance avec les

entités du modèle de base. Pour déduire la vue correspondant à un composant de modèle, il est nécessaire de substituer les éléments paramètres par les éléments du modèle de base correspondants. Tous les éléments nécessaires à la vue qui n'y sont pas définis le sont dans le modèle de base.

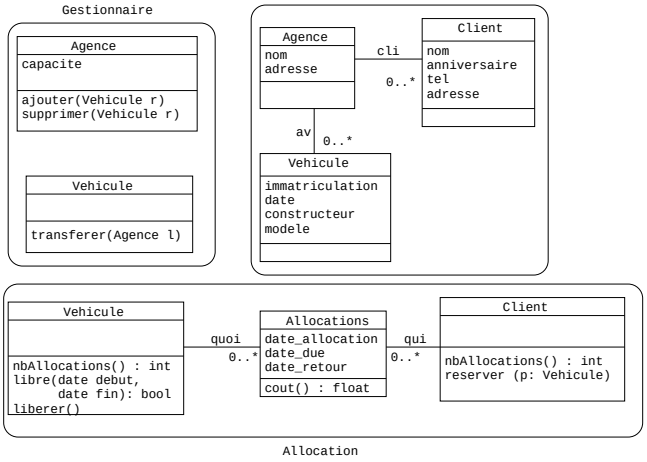

Figure 8 – Représentation à l'aide de vues

La figure 8 illustre la représentation du système résultat à l'aide d'un modèle à vues. On y retrouve deux vues, l'une correspondant au composant *Gestionnaire*, l'autre au composant *Allocation*. Les entités *Stock* et *Ressource* du composant *Gestionnaire* par exemple, sont respectivement renommées *Agence* et *Vehicule* dans la vue correspondante. Leurs attributs requis *identifiant* et *ref*, hérités de la base, n'apparaissent pas dans la vue. Au contraire, l'attribut *capacite* étant un attribut ajouté par le composant, il est défini dans la vue correspondante. Il en est de même pour les opérations définies par les composants.

Les deux représentations présentées ici permettent donc de formuler un modèle du système issu d'un assemblage de composants de modèles. Ces modes de représentation ne sont pas limitatifs et d'autres peuvent être utilisés. Notamment, ces modes de représentations peuvent être appliquées de manière uniforme, entièrement fusionnée ou entièrement structurée, mais ils peuvent être également hybridées. En effet, il peut être intéressant selon leur granularité, par exemple, de fusionner certains composants de modèles ou de déduire les vues correspondantes pour d'autres au sein d'un même assemblage. Il semble également intéressant de considérer d'autres modes de composition comme la substitution d'éléments du modèle de base par des éléments issus des composants [BOU 04].

Nous proposons de considérer dans la section suivante un mécanisme de transformation permettant d'hybrider nos modes de représentation fusionnée et structurée.

5. Projection flexible

Nous avons proposé dans [BLA 04] un processus de transformation paramétrée suggéré par l'OMG dans le cadre du MDA [MIL 03]. Nous présentons dans cette section une application de ce processus à notre approche afin de donner au concepteur le moyen d'adapter et d'hybrider les modes de mise en œuvre, fusionnée ou éclatée, pour des parties du système à différents niveaux de granularité. En effet, plusieurs critères peuvent également intervenir dans le choix de la fusion ou non d'un composant : distribution, sécurité, droits d'accès, réutilisation d'existant,... Ce processus est illustré figure 9 et présente différentes étapes.

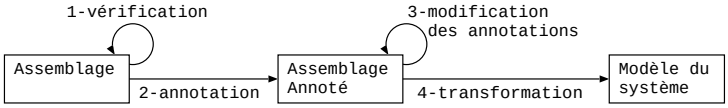

Figure 9 – Processus de transformation paramétrée

La première étape permet de vérifier la cohérence de l'assemblage, et notamment que les modèles fournis correspondent bien aux modèles requis. La seconde étape consiste à annoter chaque relation d'application par «merge» ou «view» pour indiquer le mode de projection (fusion ou vues) pour la fonctionnalité correspondante. La figure 10 illustre un assemblage muni de telles annotations. Dans cet exemple, on souhaite fusionner le composant *C1* et obtenir une vue pour le composant *C2*. Pour ne pas rendre cette phase d'annotation fastidieuse dans le cadre d'applications complexes construites à l'aide d'un grand nombre de composants de modèle, nous proposons un mécanisme d'annotation par défaut. Celui-ci permet d'annoter automatiquement toutes les relations d'application avec une valeur par défaut (par exemple «view»), l'utilisateur n'a plus alors qu'à modifier les annotations là où la stratégie par défaut n'est pas adaptée.

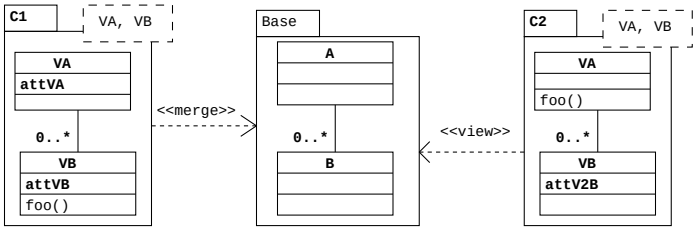

Figure 10 – Assemblage annoté

La figure 11 illustre le modèle résultant de ces choix. Le composant *C1* est fusionné dans les éléments du paquetage de base. Au contraire, le composant *C2* prend la forme d'une vue.

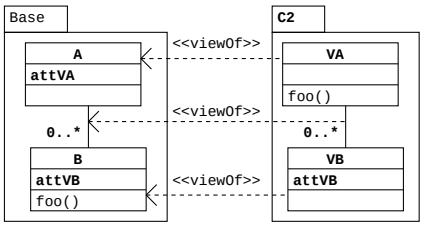

Figure 11 – Modèle résultat

Le même mécanisme de transformation paramétrée est également disponible au niveau de granularité plus fin des entités. Cela permet au concepteur de choisir pour chacune d'elles au sein d'un composant, si elle doit être fusionnée ou non [MUL 06] dans le modèle objet résultat.

Ainsi, à différents niveaux de granularité, composants et entités du système peuvent être réalisés sous forme éclatée ou fusionnée afin de permettre des stratégies de répartition, des capacités d'évolution ou encore la réutilisation d'existant [CAR 03, CAR 05].

6. L'outil

Nous présentons dans cette section l'outil développé afin de démontrer comment nos composants de modèle peuvent être intégrés dans des ateliers de modélisation et leur pertinence pour la conception de modèles de systèmes[1]. Notre outil a été conçu pour permettre la conception de composants de modèle sous forme de paquetages templates UML 2 et la réalisation d'assemblage. L'outil repose sur une extension du méta-modèle UML et un ensemble de contraintes OCL a été implémenté afin, d'une part, de garantir que le paramétrage d'un paquetage template forme bien un modèle et, d'autre part, de garantir la conformité entre modèle requis et modèle fourni. La figure 12 illustre la fenêtre principale de cet outil. Pour définir l'application d'un composant, l'utilisateur établit une relation *apply* entre ce composant et celui auquel il souhaite l'appliquer. Il doit ensuite renseigner les paramètres de cette application. Pour cela, notre outil propose une aide à la saisie qui ne propose à l'utilisateur que les éléments de type compatible. Cette aide pourra être étendue en ne proposant que les éléments conformes au modèle requis.

Une fois l'assemblage réalisé, l'outil permet, pour chaque application, de fusionner les éléments du composant (*merge*) ou de préserver la structuration (*apply*). Une fonctionnalité de génération de code, supportant nos patrons de représentation éclatée, vers la plate-forme CORBA (fichier IDL et implantation Java) est également offerte.

1. http ://www.lifl.fr/∼mullera/cocoamodeler/

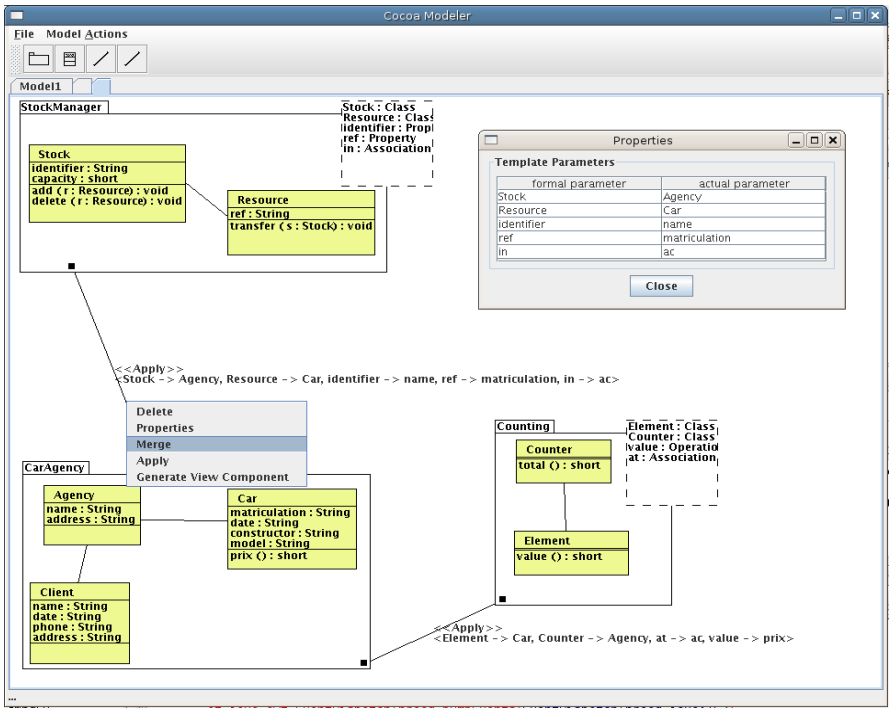

Figure 12 – Atelier de conception de composants de modèles

7. Conclusion

Nous avons vu qu'un modèle de système peut être construit à partir d'un ensemble de modèles afin notamment d'en maîtriser la complexité. Il est alors nécessaire d'intégrer cet ensemble de modèles, c'est ce que nous avons appelé l'ingénierie multi-modèles. Nous avons proposé dans ce cadre une approche à base de composants de modèle avec lesquelles il est possible de réaliser des assemblages complexes afin de concevoir des modèles de système. Pour permettre une projection flexible selon des modes de réalisation variés (fusionnée ou par vues) et hybridables nous avons proposé un processus de transformation paramétrée par annotation.

Nous avons également développé un outil de modélisation afin d'expérimenter notre approche. Bien que notre outil supporte la fusion des composants et leur mise en œuvre sous forme de vues, ce choix doit actuellement être fait composant par composant. Il semble intéressant d'ajouter à notre outil le support du mécanisme de transformation paramétrée.

A plus long terme, l'extension de nos travaux aux modèles dynamiques tels que les diagrammes d'états semble une perspective intéressante. Nous envisageons également la possibilité d'étendre les mécanismes d'enrichissement de notre approche avec des

mécanismes de substitution et de fusion tels que ceux décrits dans [BOU 05]. En effet, dans notre approche, il ne peut exister qu'un unique modèle de base avec lequel les composants sont assemblés (directement ou indirectement). L'approche [BOU 05] est au contraire conçue pour fédérer des modèles de systèmes indépendants et propose des mécanismes de substitution d'éléments.

Enfin, imposée par l'ingénierie multi-modèles, la composition de modèles est un besoin de plus en plus étudié, prenant des formes différentes, mais pour lequel il est possible d'identifier des idées communes [BÉZ 06]. Il semble intéressant d'identifier à partir de ces approches les concepts généraux de la composition de modèles et de classifier ses différentes formes : fusion, tissage, assemblage, substitution...

8. Bibliographie

[ANC 05] ANCA DANIELA IONITA JACKY ESTUBLIER G. V., « Domaines Réutilisables Dirigés par les Modèles », *Ingénierie dirigée par les modèles IDM'05*, 2005.

[BAL 02] ANDY BALCER M.J. M. S., *Executable UML, A Foundation For Model-Driven Architecture*, Addison-Wesley, 2002.

[BER 03] BERNSTEIN, P.A., « Applying Model Management to Classical Meta Data Problems », *Proceedings of CIDR 2003, Conference of Innovative Data Systems Research*, 2003.

[BÉZ 06] BÉZIVIN J., BOUZITOUNA S., FABRO M. D. D., GERVAIS M.-P., JOUAULT F., KOLOVOS D. S., KURTEV I., PAIGE R. F., « A Canonical Scheme for Model Composition. », *ECMDA-FA*, Lecture Notes in Computer Science, Springer, 2006, p. 346-360.

[BLA 04] BLANC X., CARON O., GEORGIN A., MULLER A., « Transformation de modèles : d'un modèle abstrait aux modèles CCM et EJB », *Proceedings of Langages, Modèles, Objets (LMO'04)*, Hermès Sciences, Mars 2004.

[BOO 94] BOOCH G., *Object-oriented analysis and design with applications (2nd ed.)*, Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994.

[BOU 04] BOUZITOUNA S., GERVAIS M. P., « Composition rules for PIM Reuse », *Proceedings of the 2nd European Workshop on MDA with Emphasis on Methodologies and Transformations*, Canterbury, UK, September 2004.

[BOU 05] BOUZITOUNA S., GERVAIS M.-P., BLANC X., « Model Reuse in MDA. », *Software Engineering Research and Practice*, 2005, p. 354-360.

[CAR 03] CARON O., CARRÉ B., MULLER A., VANWORMHOUDT G., « A Framework for Supporting Views in Component Oriented Information Systems », *Proceedings of International Conference on Object Oriented Information Systems (OOIS'03)*, vol. 2817 de *LNCS*, Springer, septembre 2003, p. 164-178.

[CAR 05] CARON O., CARRÉ B., MULLER A., VANWORMHOUDT G., « Mise en oeuvre d'aspects fonctionnels réutilisables par adaptation », *L'objet, Programmation par Aspects*, vol. 11, n° 3, 2005, p. 105–118, Hermes.

[CLA 01] CLARKE S., « Composition of Object-Oriented Software Design Models », PhD thesis, Dublin City University, janvier 2001.

[DEB 98] DEBRAUWER L., « Des vues aux contextes pour la structuration fonctionnelle de bases de données à objets en CROME », PhD thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, décembre 1998.

[D'S 99] D'SOUZA D., WILLS A., *Objects, Components and Frameworks With UML : The Catalysis Approach*, Addison-Wesley, 1999.

[JAC 92] JACOBSON I., *Object-Oriented Software Engineering : A Use Case Driven Approach*, Addison-Wesley, 1992.

[KEN 02] KENT S., « Model Driven Engineering », *Proceedings of IFM 2002*, LNCS 2335, Springer-Verlag, 2002, p. 286-298.

[KLE 03] KLEPPE A., WARMER J., *The Object Constraint Language, Getting your Model Ready for MDA*, Addison-Wesley, 2003.

[MEN 06] MENS T., STRAETEN R. V. D., D'HONDT M., « Detecting and Resolving of Model Inconsistency Using Transformation Dependency Analysis », *MoDELS/UML, 9th International Conference*, 2006.

[MIL 01] MILLER J., MUKERJI J., « Model Driven Architecture (MDA) », July 2001, http ://www.omg.org/cgi-bin/apps/doc ?ormsc/01-07-01.pdf.

[MIL 03] MILLER J., MUKERJI J., « MDA Guide Version 1.0.1 », 2003, http ://www.omg.org/cgi-bin/apps/doc ?omg/03-06-01.pdf.

[MUL 02] MULLER A., « La démarche MDA », rapport, 2002, Projet RNTL Accord http ://www.infres.enst.fr/projets/accord/.

[MUL 03] MULLER A., CARON O., CARRÉ B., VANWORMHOUDT G., « Réutilisation d'aspects fonctionnels : des vues aux composants », *Langages et Modèles à Objets (LMO'03)*, Hermès Sciences, January 2003, p. 241-255.

[MUL 05] MULLER A., CARON O., CARRÉ B., VANWORMHOUDT G., « On Some Properties of Parameterized Model Application », *First European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'05)*, vol. 3748 de *LNCS*, Springer, November 2005, p. 130–144.

[MUL 06] MULLER A., « Construction de systèmes par application de modèles paramétrés », PhD thesis, University of Lille, 2006.

[POO 01] POOLE J. D., « Model-Driven Architecture : Vision, Standards And Emerging Technologies », *ECOOP 2001, Workshop on Metamodeling and Adaptive Object Models*, , 2001.

[RUM 91] RUMBAUGH J., BLAHA M., PREMERLANI W., EDDY F., LORENSEN W., *Object-oriented modeling and design*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[SHE 90] SHETH A., LARSON J., « Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases », vol. 22(3), ACM Computing Surveys, 1990, p. 183–236.

[STR 04] STRAW G., GEORG G., SONG E., GHOSH S., FRANCE R., BIEMAN J. M., « Model Composition Directives », *Proceedings of 7th International Conference on The Unified Modeling Language. Model Languages and Applications (UML 2004)*, vol. 3273 de *LNCS*, Springer, 2004, p. 84–97.

[SUN 00] SUNYÉ G., GUENNEC A. L., JÉZÉQUEL J.-M., « Design Patterns Application in UML », BERTINO E., Ed., *Proceedings of ECOOP 2000*, vol. 1850 de *LNCS*, Springer, 2000, p. 44–62.

[SUN 02] SUNYÉ G., GUENNEC A. L., JÉZÉQUEL J.-M., « Using UML action semantics for model execution and transformation », *Information Systems*, vol. 27(6), 2002, p. 445-457.

[VAN 99] VANWORMHOUDT G., « CROME : un cadre de programmation par objets structurés en contextes », PhD thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, 1999.

[WIL 96] WILLS A., « Frameworks and Component-Based Development », *Proceedings of International Conference on Object Oriented Information Systems (OOIS'96)*, 1996, p. 413-431.