Conception et usage des composants métier processus pour les systèmes d'information

Rajaa Saidi

INSTITUT POLYTECHNIQUE DE GRENOBLE

THESE EN COTUTELLE INTERNATIONALE

pour obtenir le grade de

DOCTEUR DE L'Institut polytechnique de Grenoble
et
de l'Université Mohammed V-Agdal Rabat

Spécialité : Informatique

préparée au **Laboratoire d'Informatique de Grenoble**
dans le cadre de **l'École Doctorale** « Mathématiques, Sciences et technologies de l'information, Informatique»
et au **Laboratoire de Recherche en Informatique et Télécommunications**

présentée et soutenue publiquement
par

Rajaa SAIDI

le 26 Septembre 2009

Titre :

CONCEPTION ET USAGE DES COMPOSANTS METIER PROCESSUS POUR LES SYSTEMES D'INFORMATION

DIRECTEURS DE THESE : Mme. Dominique Rieu et M. Driss Aboutajdine
CO-DIRECTEUR(S) DE THESE : Mmes. Agnès Front, Mounia Fredj et Salma Mouline

JURY

M. Mohammed Abbad	, Président
Mme. Corine Cauvet	, Rapporteur
M. Bouchaib Bounabat	, Rapporteur
Mme. Dominique Rieu	, Directeur de thèse
M. Driss Aboutajdine	, Directeur de thèse
Mme. Selmin Nurcan	, Examinateur

À mes parents

À Mes frères : Hicham, Adil, Mohammed Amine et Zakaria

À mes sœurs : Bouchra et Fadwa

À ma belle sœur Zakia et beau frère Khalid

À mes adorables nièces et neveux : Najlae, Aya, Saber et Mohammed Reda

C'est avec beaucoup d'émotions que je vous remercie pour vos nombreux sacrifices ainsi que pour le soutien et la confiance que vous m'avez toujours accordés. Il m'est difficile de traduire par les mots l'affection et la gratitude que je vous réserve. Aussi, est-ce à vous que je dédie ces heures de travail, de joies et de souffrances que vous avez partagées. J'espère que vous vous réjouissez de la réussite qui couronne mes efforts. Merci pour votre amour.

Remerciements

Les travaux présentés dans ce mémoire ont été effectués à l'équipe SIGMA (Système d'Information : InGénierie et Modélisation Adaptable) du Laboratoire d'Informatique de Grenoble (LIG), à l'Institut polytechnique de Grenoble (INPG), en collaboration avec le Laboratoire de Recherche en Informatique et Télécommunications (LRIT), à la Faculté des Sciences de Rabat (FSR) dans le cadre d'une thèse en cotutelle internationale.

Mes premières pensées vont à mes cinq directeurs de thèse. Je vais les évoquer par ordre chronologique d'apparition dans le déroulement de ma thèse.

Janvier 2005

Je tiens à remercier M. Driss Aboutajdine, Professeur à la FSR et directeur du LRIT. Sans l'environnement de recherche qu'il a su créer, je n'aurais pas pu me lancer dans la préparation de cette thèse. J'exprime ici ma profonde gratitude à son égard et l'estime respectueuse que je lui porte.

Merci aussi à mes co-directrices de thèse, Mme. Salma Mouline, Professeur à la FSR, et Mme. Mounia Fredj, Professeur à l'École Nationale Supérieure d'Informatique et d'Analyse des Systèmes (ENSIAS). Grâce à elles, j'ai pu entreprendre une thèse dans mon domaine favori : les systèmes d'information. Je tiens à les remercier toutes les deux pour ces années de soutien, pour leurs précieux conseils, et pour leur manière très simple de toujours trouver les mots d'encouragement qui ne manquaient pas de raviver ma motivation.

Avril 2006

Je tiens également à remercier mes directrices de thèse au sein du LIG, Mmes. Dominique Rieu et Agnès Front, Professeurs à l'Université Pierre Mendès France (UPMF) à Grenoble. Qu'elles trouvent d'abord mes remerciements pour la confiance qu'elles m'ont toujours témoignée. L'intérêt constant qu'elles ont pris pour ce travail, m'a permis de ne pas m'égarer dans des problèmes flous et de ne jamais perdre de vue l'essentiel, en donnant de leur temps et en acceptant de partager leurs expériences.

Je ne saurais omettre bien sûr M. Jean Pierre Giraudin, Professeur à l'UPMF et responsable de l'équipe SIGMA. Je le remercie de m'avoir intégrée au sein de l'équipe et d'avoir accepté de gérer le projet de collaboration. Le projet dans lequel il a toujours agi en faveur du doctorant. Sa gentillesse, ses encouragements et ses conseils éclairants m'ont permis de mener ce travail à son terme. Qu'il trouve ici un témoignage sincère de ma profonde reconnaissance. Je tiens aussi à remercier Mme. Laila Kjiri, Professeur à ENSIAS, d'avoir accepté de gérer le projet de collaboration du côté marocain.

Je tiens aussi à remercier les membres du jury :

- M. Mohammed Abbad, *Professeur à la Faculté des Sciences de Rabat*, qui m'a fait l'honneur de présider ce jury.
- Mme. Corine Cauvet, *Professeur à l'Université Aix-Marseille*, de m'avoir fait l'honneur de rapporter ma thèse.
- M. Bouchaib Bounabat, *Professeur à l'École Nationale Supérieure d'Informatique et d'Analyse des Systèmes*, de m'avoir fait l'honneur de rapporter ma thèse.

- Mme. Dominique Rieu, *Professeur à l'Université Pierre Mendès France à Grenoble*, qui m'a fait l'honneur de co-diriger ma thèse.
- M. Driss Aboutajdine, *Professeur à la Faculté des Sciences de Rabat*, qui m'a fait l'honneur de co-diriger ma thèse.
- Mme. Selmin Nurcan, *Maître de conférences à l'Université Paris 1 Panthéon Sorbonne*, d'avoir accepté d'être examinatrice de ma thèse.

Dans le but de valider les propositions de ma thèse, j'ai réalisé des expérimentations utilisateurs, cette activité ne pourrait être entreprise sans l'aide de plusieurs personnes. Mes pensées vont premièrement à Mme. Nadine Mandran, ingénieur de méthodes et de qualité au sein du Marvelig. Je la remercie pour son aide, sa rigueur et ses conseils précieux. Je remercie également les sujets participants aux expérimentations : Sophie, Charlotte, Yassine H, Diallo, Jorge, Amin, Ansem, Luz Maria, Hassan, Youssef, Yassine, Ali, Marco, Mikael et Latifa. Je les remercie tous d'avoir accepté ma demande et de m'avoir aidé à évaluer mes travaux.

Je tiens à remercier aussi mes amies côtoyées à Grenoble : Jihane, Latifa, Saadia, Raja, Asma, Salma et Hayet. Un remerciement particulier est adressé à mes amies de l'AJCIT : Sanaa G, Sanaa E, Najlae, Siham, Samira et Rajae. Merci de m'avoir fait confiance hormis la distance, j'espère que mes préoccupations n'ont pas entravé les activités de l'association et que cette association reste vivante.

Je remercie bien sûr mes amies de toujours : Amina, Ouafae, Asmaa et Aouatif, qui, malgré la distance, continuent toujours à me soutenir et à m'encourager.

Enfin, je désire remercier tout particulièrement des amis, grâce auxquels j'ai pu puiser toute la force dont j'avais besoin pour arriver au terme de ce travail. Dans les conseils desquels j'ai pu rechercher à chaque fois que le besoin s'en faisait ressentir, le courage de ne pas abandonner et la foi dans l'ouvrage entrepris. À Marie Paule, Marc Olivier et d'autres personnes qui se reconnaîtront.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1 : Contexte général et problématique

Ce chapitre présente en premier lieu le contexte général de cette thèse (§1) ainsi que la problématique abordée (§2). Les principales contributions réalisées sont ensuite introduites (§3) avant de préciser le plan de ce mémoire (§4).

1 La réutilisation dans les systèmes d'information

1.1 Ingénierie des systèmes d'information

Un Système d'Information (SI) doit être considéré comme un artefact, greffé sur l'objet réel qu'est l'entreprise, volontairement conçu et mis en place pour remplir des fonctions de représentation, de mémorisation et de communication, en vue d'atteindre des objectifs informationnels [Rolland *et al.*, 1988].

Sous la pression concurrentielle, un nombre croissant d'entreprises ne se préoccupent plus uniquement de la qualité de leurs produits, mais également de celle de leurs SI. La certification basée sur la norme ISO9000 et l'évaluation de la maturité des entreprises ont renforcé ce mouvement [Morley *et al.*, 2000]. Pour cette raison, les SI des entreprises doivent être suffisamment flexibles pour pouvoir apporter rapidement une solution adaptée à tout changement dynamique. C'est dans ce sens que l'ingénierie des SI par réutilisation est devenue largement adoptée et utilisée, et a fait émerger des méthodes et outils innovants. Une telle approche doit permettre de réduire le temps de conception des SI, d'en améliorer la qualité et d'en faciliter la maintenance.

1.2 De l'orienté objet vers l'orienté composant

À la fin des années 90, une nouvelle mutation dans les approches de développement des SI s'est opérée, puisque l'on est passé des approches orientées objets aux approches à base de composants. Ces évolutions ont nécessité la mise en œuvre de SI flexibles, adaptables à l'évolution des produits et des processus métier, ce qui a motivé un intérêt accru pour les modèles et les méthodes de réutilisation. Avec l'effervescence autour de la technologie objet, on a ressenti le besoin de réutiliser des connaissances nouvelles, différentes de celles associées aux simples objets techniques. Cette mutation était motivée par les limites de l'approche objet, en particulier en terme de granularité. Ces limites ont été incontestablement confirmées, quand les expériences et les pratiques dans la communauté de l'ingénierie des SI ont conclu que la réutilisation des composants, et plus particulièrement sur les connaissances de domaine, constitue une approche puissante pour le développement des SI [Barbier *et al.*, 2002].

1.3 Réutilisation des composants métier

La réutilisation des connaissances de domaine, et plus particulièrement de celles issues d'un métier donné et qui constituent un Domaine Métier, est actuellement largement adoptée. Elle a fait l'objet de nombreux travaux de recherche [Herzum *et al.*, 2000] [Barbier *et al.*, 2002] [Cauvet *et al.*, 2005] [Hassine , 2005]. Une telle approche est intuitive, à partir du moment où l'on se rend compte que certains domaines d'activité conduisent à la manipulation fréquente de concepts similaires. On pense alors immédiatement à analyser ces concepts et à créer pour eux des abstractions informatiques pouvant être réutilisées lors de nouveaux développements.

Ainsi, des composants réutilisables, qui ne sont pas seulement destinés à la solution de problèmes techniques purement informatiques, mais qui répondent à des besoins propres d'un domaine d'activité particulier, pourront aussi rendre plus facile, plus rapide et moins coûteuse la mise en place de systèmes d'information dans le domaine auquel ils se rapportent.

C'est dans ce sens que l'approche « Composant Métier » (CM) a été proposée [Herzum *et al.*, 2000] [Bass *et al.*, 2000] [Heineman *et al.*, 2001]. De tels composants sont ainsi utilisés par exemple dans les domaines de la médecine, de la finance ou de la comptabilité [Andro *et al.*, 1998].

2 Problématique

2.1 Nature de l'information réutilisable

Dans le développement de SI à base de composants, il est souvent difficile d'expliciter des critères clairs de réutilisation, en particulier la manière dont on peut identifier et spécifier des composants candidats à la réutilisation. Autrement dit, il est important de savoir comment les connaissances d'un SI ou d'un domaine peuvent être identifiées pour être ensuite formalisées, structurées et mises à la disposition des concepteurs de SI à base de composants.

Décrire le SI de l'entreprise revient à représenter son fonctionnement et son organisation sous l'angle des informations que l'on a choisies de gérer et sur lesquelles on s'appuie. Cependant, quand le SI est vaste, avec un nombre élevé de types d'informations et un nombre élevé d'acteurs ayant des rôles différents, il est nécessaire de le concevoir de façon modulaire pour mieux le comprendre, pour maîtriser son développement et pour permettre des évolutions partielles sans impact sur l'ensemble.

Dans ce sens, un SI a été souvent considéré comme un ensemble de sous-SI centrés sur un Processus Métier (PM) [Morley *et al.*, 2004] [Dumas *et al.*, 2008]. Il s'agit d'une structure modulaire spécifique à un SI donné. C'est un processus au cœur du système, représentatif d'une partie du métier de l'entreprise et qui constitue une solution particulière à une classe de problèmes définie pour un domaine métier donné.

De ce fait, il est devenu nécessaire de relier la définition d'un CM réutilisable à la façon dont un SI est organisé. Le fait qu'un SI soit constitué par un ensemble de PM matérialisant la finalité globale de ce SI nous amène à bien situer le besoin d'un SI en terme de réutilisation.

C'est dans cette perspective qu'un CM ne peut être candidat à une éventuelle réutilisation que s'il répond aux besoins d'un SI. Dans ce sens, il parait indispensable de spécifier des CM à partir de l'abstraction des PM menés par différents intervenants dans différents SI. Deux types principaux de CM sont distingués : les CM de type entité (CME), qui permettent la manipulation d'abstractions d'entités du monde réel, et les CM de type processus (CMP), destinés à soutenir directement les processus métier spécifiques aux différents intervenants, et qui mettent à disposition pour cela les fonctionnalités adéquates.

Dans cette thèse, l'intérêt doit être mis plus particulièrement sur la réutilisation de CM de nature **processus** qui s'avère plus pertinente qu'une simple réutilisation de CME. En effet, un CM ne doit pas représenter de simples concepts métier, mais des concepts qui sont relativement autonomes dans l'espace du problème. Par exemple, l'allocation de ressources est un bon candidat, tandis qu'une ressource ne l'est pas, car l'information y est trop réduite.

2.2 Portée de l'information réutilisable

Dans le développement ou la maintenance des SI, la réutilisation ne se restreint pas à la simple utilisation de fragments de code existants, mais inclut aussi tous les produits de

développement, de l'ingénierie des besoins jusqu'à la conception. Tout fragment produit à partir d'un effort de développement d'un système peut être potentiellement réutilisé.

Cependant, malgré la diversité des travaux de spécification des CM, la tendance qui s'impose est de considérer un composant comme une brique logicielle [Szyperski, 1998] [Ewald, 2001] [Belvins, 2001]. En conséquence, la spécification des composants est relativement technique et son usage reste limité à la phase de conception d'architecture et de production logicielles [Cauvet *et al.*, 2005].

La réutilisation dans un SI ne peut être réduite à la réutilisation de spécifications logicielles mais elle doit aussi concerner celles conceptuelles produites dans les premières phases du développement. Cela présente un intérêt accru quand apparaissent des difficultés à gérer la traçabilité entre un besoin d'entreprise et sa réalisation dans le système logiciel.

Ce constat se retrouve également dans les approches qui séparent les besoins fonctionnels des besoins techniques d'une organisation [Rocques *et al.*, 2004] [Hassine, 2005]. Pour ce qui est de l'organisation, son évolution est relativement lente, sauf incidents économiques ou politiques venant de l'intérieur ou de l'environnement. C'est pour cette raison que les besoins fonctionnels du SI varient légèrement au fil du temps. Par contre, les besoins techniques changent dans le temps d'une manière beaucoup plus sensible. Par exemple, le changement de matériel ou encore l'évolution dans les solutions informatiques.

La politique d'étude qui doit être menée dans les organisations doit prendre en compte ces différences d'évolution entre ce qui est fonctionnel et ce qui est technique. À l'extrême, si un SI est bâti sans cette séparation, s'il est construit à l'aide d'éléments susceptibles de changements brutaux, c'est l'ensemble du SI qui devra être revisité.

De ce fait, un modèle de CM de portée **conceptuelle** incluant des besoins **fonctionnels** permet de procurer des bénéfices importants lors de sa réutilisation ; il est technologiquement neutre et n'évolue qu'en fonction de l'évolution des besoins auxquels il répond.

2.3 Propriétés de l'information réutilisable

Un CM doit se suffire à lui-même, de façon à pouvoir être documenté efficacement. Aussi, sa spécification doit bénéficier de certaines propriétés qui le rendent à la fois flexible et adaptable.

2.3.1 Complétude

La modélisation d'un CM a souvent été limitée à l'aspect structurel du composant, en particulier dans les modèles de données de [Mineau *et al.*, 1995], les composants de [Castano *et al.*, 1994], le modèle de domaine de [Snoeck *et al.*, 2000], le modèle Symphony de [Hassine, 2005] et le Business Component de [Herzum *et al.*, 2000]. Cependant, une approche à vues multiples permet d'exprimer plus complètement la solution offerte par un composant. En effet, la réutilisation d'une telle solution permettrait une meilleure qualité de conception, grâce à une spécification plus complète matérialisée par plusieurs vues.

Définir la complétude n'est pas une activité facile, car cette notion varie selon le contexte de recherche. En se situant dans un contexte métier, doter le CM par une vue à caractère métier

est certainement nécessaire. Cette vue matérialise l'aspect organisationnel du composant. Une deuxième vue qui traduit la partie informatisée du composant est aussi intuitive pour illustrer ses fonctionnalités. Ces fonctionnalités doivent inévitablement être détaillées dans une vue explicitant le comportement du CM et une vue détaillant sa structure générale.

De ce fait, la solution d'un CM doit fournir une spécification complète en intégrant plusieurs vues tout en gérant leur cohérence.

2.3.2 Variabilité

Pour augmenter sa réutilisation, un composant doit offrir plusieurs réalisations possibles. Sa solution doit donc être flexible afin de pouvoir être adaptée aux spécificités du système en cours de développement.

Ceci n'est, en fait, qu'une des conséquences prévisibles du développement informatique des organisations. En effet, après avoir consenti des investissements importants pour améliorer leur gestion, les entreprises et les administrations manifestent leur volonté d'adapter les solutions informatiques à leurs propres besoins au lieu d'ajuster leurs méthodes de travailler aux applications utilisées. C'est notamment pour cette raison qu'il est devenu nécessaire de spécifier des CM supportant la variabilité [VanGurp *et al.*, 2000] [Kang *et al.*, 1990].

Plusieurs formes de variabilité existent, parmi lesquelles certaines peuvent être adoptées pour représenter la variabilité dans les CM. Citons la variabilité des processus, correspondant au fait que des PM similaires peuvent être déclinés de plusieurs façons selon les besoins de chaque SI. De même, la variabilité du degré d'informatisation d'un PM varie d'un SI à un autre. Enfin, pour favoriser la réutilisation par analogie entre domaines différents, mais qui manipulent des informations similaires, la variabilité se doit de discriminer les spécificités de chaque domaine métier.

La représentation de la variabilité dans un CM s'avère donc nécessaire. Son identification est réalisée en analysant des PM admettant des propriétés communes et d'autres variables. Ce constat nous amène à distinguer dans la spécification d'un CMP des propriétés fixes et des propriétés variables facilitant sa réutilisation.

2.4 Guide méthodologique pour la réutilisation

La spécification des composants réutilisables ne doit pas se restreindre seulement à la description des informations effectivement réutilisables, mais aussi à celles requises pour assurer une spécification cohérente et une réutilisation appropriée de ces composants.

Aujourd'hui, des applications permettent de gérer les composants mais ne proposent pas d'outils puissants pour assister le concepteur des composants dans le processus « conception pour la réutilisation » ainsi que le concepteur des SI à base de composants dans le processus « conception par la réutilisation ».

Dans cette perspective, la recherche en matière de CM a fait émerger des méthodes de développement basées composants. Le degré de prise en charge de la réutilisation dans le cycle de vie de ces méthodes varie d'une approche à une autre, mais reste toujours incomplète.

Par conséquent, un processus d'ingénierie de CM se doit d'une part de fournir les directives requises pour représenter la structure fondamentale du composant, en offrant des mécanismes facilitant sa description, et d'autre part de proposer une organisation du modèle de composants afin d'en faciliter l'accès. Enfin, un processus de réutilisation doit fournir l'ensemble des activités liées à la manipulation des composants pour la conception d'un SI, à savoir la recherche, l'adaptation et la composition.

3 Contributions de la thèse

Les contributions de cette thèse s'articulent autour de trois principaux résultats.

La première contribution consiste en la proposition d'un modèle de CM de nature **processus** appelé « CMP ». La particularité de ce modèle est qu'il est de portée **conceptuelle incluant les besoins fonctionnels**. Ce choix est motivé par le fait que seuls les modèles conceptuels, produits lors des étapes de spécification fonctionnelle, constituent de véritables productions capitalisables et réutilisables dans le cycle de conception d'un SI à base de composants. Pour modéliser un CMP, nous nous basons sur des modèles UML, d'une part parce que UML propose un ensemble de diagrammes standards supportés par la plupart des outils existant, d'autre part, parce que son extensibilité rend possible la modélisation des concepts liés aux CMP. Dans le but de maximiser la réutilisabilité d'un CMP, le modèle proposé est centré sur la technique de la **variabilité** [Saidi *et al.*, 2007a] [Saidi *et al.*, 2007b]. Dans ce sens, et pour spécifier un CMP supportant la variabilité, nous proposons des **extensions d'UML** supportant les concepts de variabilité. Ces extensions ont mené à la proposition d'un **profil UML** pour les CM supportant la variabilité. Une autre particularité de ce modèle concerne la **complétude** de sa solution spécifiée sur quatre vues de développement : une vue métier représentant l'aspect organisationnel du CMP, une vue fonctionnelle représentant l'ensemble des fonctionnalités informatisées du CMP, une vue dynamique représentant le comportement de chaque fonctionnalité incluse dans le CMP et une vue structurelle représentant les données manipulées par le CMP [Saidi *et al.*, 2009a], [Saidi *et al.*, 2008a], [Saidi *et al.*, 2008c].

Une deuxième contribution s'inscrit dans le cadre de la proposition d'un **processus d'ingénierie de CMP** permettant sa spécification selon le modèle proposé. Dans ce processus, nous proposons un ensemble de **règles de construction**, de **traduction** et de **cohérence** qui assurent la traçabilité des artefacts produits tout au long du cycle de développement du CMP.

Une troisième contribution concerne la proposition de directives qui assistent l'ingénieur de SI à base de composants lors de la réutilisation des CMP. Ces directives s'intéressent premièrement à la proposition de mécanismes d'organisation des CMP en proposant un **formalisme** de documentation et de classification de CMP validé dans le cadre d'un environnement de stockage de composants. Nous proposons également un **processus d'imitation** permettant de tirer partie de la spécification d'un CMP, que nous intégrons dans le cycle de vie d'une méthode de développement en utilisant la méthode Symphony comme référence [Saidi *et al.*, 2009b]. Afin de faciliter l'utilisation du processus d'imitation, nous proposons un prototype qui automatise une phase importante du processus. Finalement, et afin de valider notre modèle de CMP, nous présentons les résultats d'expérimentations utilisateurs qui nous ont aidés d'un point de vue qualitatif à : 1) vérifier que notre modèle répondait bien aux objectifs de réutilisation auxquels nous prétendions répondre et 2) vérifier

que la solution offerte par notre modèle de CMP correspondait bien à la propriété de complétude que nous visions.

4 Plan de la thèse

Ce rapport de thèse est organisé en six chapitres (Cf. Figure 1-1) :

Le second chapitre est dédié à un état de l'art sur les composants métier, les méthodes de développement basées composants et la variabilité. Nous définissons un cadre de référence pour chacun des axes, qui permet par la suite de décrire chaque approche étudiée selon ce cadre. Une approche comparative nous permet de positionner notre approche par rapport aux travaux existants.

Les constatations du deuxième chapitre conduisent dans le troisième chapitre à une première partie consacrée à la description des concepts de base liés à la notion de CM ainsi que des besoins qui ont guidé vers le modèle de CMP proposé dans cette thèse. La deuxième partie détaille la vue métier, pilier de la spécification d'un CMP, à travers les techniques d'intégration de variabilité ainsi que les règles utilisées pour sa réduction.

Nous proposons dans le quatrième chapitre un processus d'ingénierie des CMP intégrant à la fois les techniques utilisées pour spécifier les trois autres vues et maintenir la cohérence inter-vues. Ce processus est destiné principalement à assister les ingénieurs de CMP dans leur activité de spécification d'un CMP.

Les spécifications proposées pour un CMP ne sont utiles que si elles sont organisées dans un environnement pour des futures réutilisations. C'est dans ce but que nous présentons, dans le cinquième chapitre des mécanismes d'organisation en proposant un formalisme de CMP, un processus d'imitation qui assiste l'ingénieur des SI pour imiter un CMP et finalement une extension d'une méthode de développement par les concepts de réutilisation. Ces propositions sont accompagnées d'outils et d'expérimentations utilisateurs servant de support de validation et de mise en œuvre des travaux réalisés.

Le dernier chapitre clôture ce document en rappelant les propositions de cette thèse, et en donnant des perspectives possibles sur un plan d'approfondissement des travaux réalisés ainsi que sur un plan d'élargissement du domaine de la recherche.

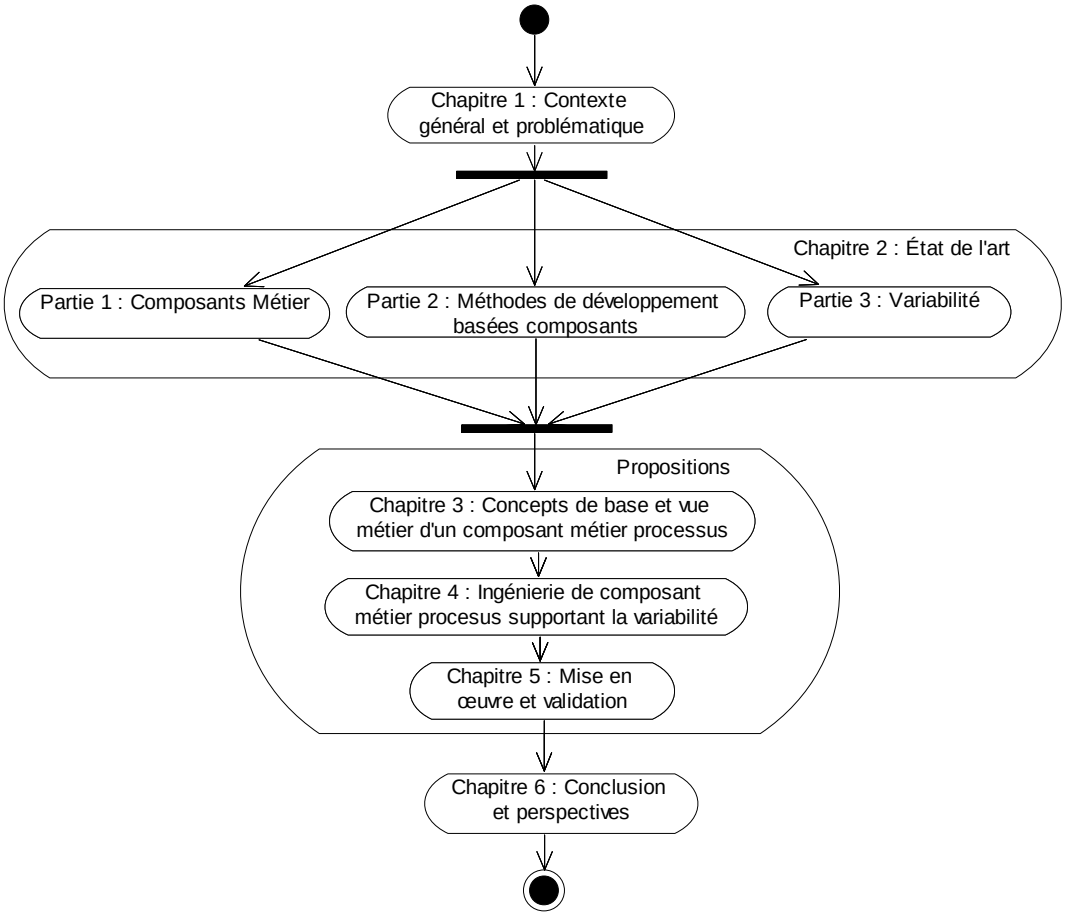


Figure 1-1 : Plan de la thèse

Chapitre 2 : État de l'art

Ce chapitre expose un état de l'art sur les composants métier (§ 1), les méthodes de développement basées composants (§ 2) ainsi que les concepts inhérents à l'expression et la gestion de la variabilité (§ 3). Cet état de l'art constitue une vue d'ensemble des travaux existants permettant ainsi de positionner (§ 4) les propositions de cette thèse.

1 Composants métier

La réutilisation des connaissances de domaine, et plus particulièrement de celles issues d'un métier donné et qui constituent un Domaine Métier (DM), connaît un essor important, en faisant l'objet de nombreux travaux de recherche [Herzum *et al.*, 2000] [Barbier *et al.*, 2002] [Cauvet *et al.*, 2005] [Hassine , 2005]. Une telle approche est intuitive, à partir du moment où l'on se rend compte que certains domaines d'activité conduisent à la manipulation fréquente de concepts identiques. On pense alors immédiatement à analyser ces concepts et à en créer des abstractions informatiques pouvant être réutilisées lors de nouveaux développements. Ainsi, des composants réutilisables, qui ne sont pas seulement destinés à la solution de problèmes techniques purement informatiques, mais qui répondent à des besoins propres à un domaine d'activité particulier, pourront aussi rendre plus facile, plus rapide et moins coûteuse la mise en place de systèmes d'information dans le domaine auquel ils se rapportent. C'est dans ce sens que l'approche « Composant Métier » (CM) a été proposée [Herzum *et al.*, 2000] [Bass *et al.*, 2000] [Heineman *et al.*, 2001]. De tels composants sont ainsi utilisés par exemple dans les domaines de la médecine, de la finance ou de la comptabilité [Andro *et al.*, 1998].

La recherche dans le domaine des CM a donné naissance à de nombreuses technologies à base de composants, qui sont aujourd'hui proposées et utilisées dans tous les secteurs d'activité. Ces méthodes de développement ont connu une grande évolution pour bien répondre aux besoins des systèmes d'information.

L'objectif ici est de présenter plus précisément le concept de CM. Pour ce faire, différentes définitions attribuées aux CM sont citées (§1.1). Nous présentons ensuite des critères de classification des CM (§1.2), ainsi qu'un ensemble de modèles et approches (§1.3) que nous comparons selon ces critères (§1.4), avant un paragraphe de synthèse (§1.5).

1.1 Définitions

Différentes définitions existent dans la littérature lorsqu'il est question de composants métier. Ces définitions ne fournissent pas toutes la même description d'un CM.

Selon l'Object Management Group (OMG) "a **business component** represents a software implementation of an "autonomous" business concept or business process. It is composed of all software artefacts necessary to express, implement, and deploy a business component as an autonomous, reusable element of an information system" [OMG, 1998].

D'autre part, [Herzum *et al.*, 2000] rajoute des informations sur le cycle de vie et la granularité d'un CM par la définition suivante : "a **Business Component** is a ***software implementation of a concept*** and itself is a ***composition of software artifacts*** *(*including ***distributed components****)*. It is a unifying concept through the development **lifecycle** and the architectural tiers".

D'autres définitions citées dans [Barbier *et al.*, 2002] mettent en évidence les notions de couverture et portée des CM :

« A **business component** is a software component that provides functions in a business domain. (…) for example, an order management application is part of the Enterprise Resource Planning (ERP) business domain » [Heineman *et al.*, 2001].

«A **Business Component** models and implements business logic, rules and constraints that are typical, recurrent and comprehensive notions characterizing a domain or business area. Within software engineering, business components are key abstractions that are captured during the domain engineering activity. They are software artefacts in the sense that they are *not* part of reality but embody and represent recurrent invariants relevant to requirements, particularly at the earliest phases of development» [Ambler, 1998].

«A **Business Component** is a part of an enterprise that has the potential to operate independently, in the extreme as a separate company, or as part of another company» [Cherbakov *et al.*, 2005].

Par ailleurs, la structure d'un CM peut être caractérisée par les éléments suivants [Cummins, 1999] [Casanave, 1996] : nom (terme employé par les experts pour caractériser le CM), définition (signification du CM), attributs (propriétés appropriées concernant le CM), comportement (actions ou services que le CM peut réaliser), relations (connections représentant les interactions avec d'autres CM) et contraintes métier (contraintes liées aux comportements, relations et attributs du CM).

À partir de toutes ces définitions qui varient selon les auteurs, nous considérons qu'un CM est une représentation d'un **concept** actif dans un **domaine métier**. Ainsi, les CM sont utilisés pour définir des concepts de type « **produit** » (par exemple, bibliothèque, compte, abonné…), ou pour définir des « **processus** » pour les concepts qu'ils représentent (par exemple, processus d'emprunt, processus de réservation d'un ouvrage, processus d'inscription d'un abonné…).

Un CM peut être une **composition d'artefacts** qui assurent la complétude de sa solution. Ces artefacts peuvent représenter différents **niveaux d'abstraction** (analyse, conception…) ou différentes **vues de développement** (fonctionnelle, dynamique, statique…). Finalement, un CM doit avoir une structure qui facilite sa réutilisation sous forme d'un modèle de composants.

1.2 Critères de classification des CM

Plusieurs critères et caractéristiques permettent de comparer et de classifier les modèles de CM [Conte *et al.*, 2001] [Klement *et al.*, 2000] [Redolfi *et al.*, 2005] [Wartik *et al.*, 1992]. L'ensemble de ces critères est donné dans cette partie.

1.2.1 Définition des critères de classification

a) Visibilité

La visibilité caractérise le niveau de transparence du CM. Selon que sa réutilisation entraîne ou non une modification de sa structure interne, il est possible d'en distinguer plusieurs types.

- *Boîte noire* : la structure interne du composant n'est ni visible, ni modifiable, seule l'interface est accessible ;
- *Boîte blanche* : le composant est complètement transparent et modifiable ;
- *Boîte en verre* : la structure interne du composant est visible mais non modifiable.

b) Granularité

La notion de granularité définit la taille du plus petit élément d'un composant. Pour les composants orientés objets, la granularité est souvent mesurée en nombre de classes. De

manière plus générale, elle peut être mesurée en nombre d'entités (des modules, des activités, etc.). Elle peut être *faible* (< 10), *moyenne* (< 100) ou *forte*.

c) Unité de composition

Il s'agit du plus petit élément d'un composant. Les *classes* et les *objets* restent dans la plupart des approches des unités de composition.

d) Variabilité

La variabilité est une propriété essentielle qui peut permettre à un utilisateur (concepteur d'applications ou utilisateur final) d'adapter un composant à son besoin [Kang *et al.*, 1990], la variabilité peut être *conceptuelle* ou *technique*.

- *Variabilité conceptuelle* : concerne la variabilité introduite dans les modèles conceptuels du composant, ex. variabilité des fonctionnalités, des modes d'interaction ou de la structure d'un CM.
- *Variabilité technique :* consiste principalement en des variations dans la plate-forme ; systèmes d'exploitation, matériel, interface utilisateur, langage de programmation, etc.

Le concept de variabilité sera traité en profondeur dans une partie suivante (§ 3).

e) Type de connaissance

Selon différents auteurs, une classification des CM en trois types de connaissance est proposée [Herzum *et al.*, 2000] [Schmid, 1999] :

- *Composants métier de type entité* : ils représentent les éléments "statiques" et réels du domaine d'activité considéré, et correspondent généralement aux éléments identifiés lors de la conception des modèles de données.
- *Composants métier de type processus* : ils représentent des activités (processus métier) du domaine considéré. Les utilisateurs s'appuient dans la réalisation de leurs tâches sur ce type de composants métier. De ce fait, les processus manipulent et utilisent des composants métier de type entité.
- *Composants métier de type utilitaire* : il s'agit de composants métier qui peuvent être mis en œuvre dans des SI se rapportant à des contextes différents. Ils sont utilisés par les composants métier de type processus et de type entité. Ils sont de granularité plus faible que les deux catégories de composants métier précédemment citées. Une mesure, une adresse, une valeur monétaire, ainsi que les composants à référence spatiale comme les points, les lignes ou les polygones, sont des exemples de composants métier de type utilitaire.

On peut noter que les CM de type utilitaire sont très stables et n'évoluent quasiment pas dans le temps ; une adresse reste une adresse. Ils sont de plus aisément standardisables. Les CM de type entité présentent également une grande stabilité, mais dans un domaine beaucoup plus spécifique. Toutefois, ils restent pauvres en terme de connaissances s'ils sont réutilisés indépendamment de leur processus d'origine. Les CM de type processus sont généralement présentés, dans le contexte des entreprises, comme des éléments peu stables, car appelés à évoluer pour permettre à l'entreprise de rester concurrentielle. Le Tableau 2-1 synthétise le critère *Type de connaissance* des CM.

Type de connaissance	Type de connaissance utilisé	Stabilité	Exemples
Utilitaire	--------	Très stable	Adresse, nom
Entité	Utilitaire	Stable	Personne, Client
Processus	Entité, utilitaire	Peu stable	Abonnement, facturation

Tableau 2-1 : Critère Type de connaissance d'un CM

f) Nature de solution

Les solutions offertes par un CM peuvent être de nature conceptuelle ou logicielle.

- *Conceptuelle :* une solution d'un CM est conceptuelle si elle résout un problème conceptuel sous la forme d'un modèle (ou une partie d'un modèle) destiné à être réutilisé. Selon l'approche de modélisation utilisée, la solution peut être spécifiée avec le langage UML ou tout autre langage de modélisation.
- *Logicielle :* une solution d'un CM est logicielle si le CM est un paquetage cohérent d'implantations logicielles qui peut être indépendamment développé et délivré. Il possède des interfaces explicites spécifiant les services offerts et les services requis par le composant. Il peut être composé avec d'autres composants et être éventuellement paramétrable sans pour autant modifier son implantation [D'Souza *et al.*, 1998].

g) Langage de représentation

Pour représenter un CM, l'utilisation d'un langage de représentation est inévitable. Le langage le plus utilisé étant le langage UML parce qu'il est supporté par la plupart des outils de développement. À côté du langage de représentation, une description en langage naturel est souvent utilisée.

h) Langage d'implantation

Un CM peut fournir le processus et les produits de développement qui sont nécessaires pour implémenter et déployer un concept de domaine dans un système logiciel. Dans cette approche, un CM est le plus souvent implanté au moyen des technologies composants distribués de type COM/DCOM, CORBA ou EJB.

i) Complétude

Il s'agit du degré d'articulation des fragments de solutions offerts par le CM, permettant d'aboutir à sa réutilisation dans différents contextes. Nous considérons que la complétude d'un CM concerne sa spécification selon plusieurs vues de développement.

- *La vue métier :* illustre le processus métier qui organise le déroulement des activités d'un CM dans un système donné ;
- *La vue fonctionnelle* : représente les fonctionnalités du CM, les dépendances qui relient ces fonctionnalités et les acteurs qui les déclenchent ;
- *La vue dynamique* : modélise les différentes interactions entre classes du CM ainsi qu'une description détaillée de ses fonctionnalités ;

- *La vue structurelle* : identifie les concepts du domaine sous forme de classes, leurs associations avec multiplicité et leurs attributs ;
- *La vue logicielle* : représente la forme d'un CM quand il est implanté par une technologie composant (ex. EJB, CORBA, etc.).

j) Couverture

Il s'agit du domaine d'application du CM. Il peut s'agir d'un CM *générique*, de *domaine* ou d'une *entreprise*.

- *Générique* : il s'agit de CM indépendants d'un domaine d'application. Un CM de réservation est, par exemple, un composant générique car il résout un problème fréquent dans de nombreux domaines d'applications.
- *Domaine* : il s'agit de CM dépendants d'un domaine d'application. Par exemple, un CM de réservation de train est un CM qui résout un problème fréquent dans un domaine d'application précis.
- *Entreprise* : il s'agit de CM dépendants de l'entreprise où ils sont modélisés.

k) Portée

La portée d'un composant est évaluée en fonction de l'étape d'ingénierie à laquelle le composant s'adresse. Elle peut être de trois types.

- *Analyse :* les CM de cette portée ont des caractéristiques et des comportements qui leur permettent d'être utilisés naturellement dans la modélisation de produits et de processus dès le niveau d'analyse des besoins.
- *Conception* : les CM de cette portée vise à répondre à des problèmes d'architecture et de conception des logiciels.
- *Implantation :* les CM de cette portée fournissent le type de technologie logicielle avec lequel ils sont implantés.

1.2.2 Synthèse

Le Tableau 2-2 résume les critères de classification des CM. Notons que les valeurs données à ces critères sont les plus rencontrées dans la littérature et peuvent être enrichies par d'autres valeurs selon la spécificité de l'approche utilisée pour la conception d'un CM.

Critère	Valeurs				
Visibilité	Noire	Blanche	Verre		
Granularité	Faible	Moyenne	Forte		
Unité de composition	Classe	Objet			
Variabilité	Conceptuelle	Technique			
Type de connaissance	Entité	Processus	Utilitaire		
Nature de solution	Conceptuelle	Logicielle			
Langage de représentation	UML / langage naturel				
Langage d'implantation	COM/DCOM, CORBA, EJB, etc.				
Complétude	Métier	Fonctionnelle	Dynamique	Structurelle	Logicielle

Couverture	Générique	Domaine	Entreprise
Portée	Analyse	Conception	Implantation

Tableau 2-2 : Critères de classification des CM

1.3 Approches à base de composants métier

L'étude des différentes approches de conception de composants métier fait apparaître trois grandes tendances.

Dans la première tendance, un CM est implanté au moyen des **technologies composants distribués** de type COM/DCOM (Distributed Component Object Model) [Wang *et al.*, 2001], CORBA (CORBA Components Model) [Ewald, 2001] ou EJB (Enterprise JavaBeans) [Blevins, 2001]. Cette forme de CM est adoptée par l'OMG [Herzum *et al.*, 2000] et par IBM dans son architecture SanFrancisco [Bohrer, 1998].

Une deuxième tendance opérée récemment, considère un composant comme un **service** à partir du moment où il offre des fonctionnalités sous forme de services. De telles approches sont les SCA (Service Component Architecture) d'IBM [IBM, 2006] et les SOA (Service Oriented Architecture) [Rouillard *et al.*, 2007]. Il est généralement admis qu'une approche orientée services permet la réutilisation. Cependant, ces concepts sont souvent conçus pour répondre à des besoins métier sans être forcément intégrés dans des applications, alors que les composants sont typiquement conçus pour être redéployés ou intégrés dans différentes applications.

Dans la troisième tendance, un CM est une unité de réutilisation des informations d'un **domaine**. Cette approche met l'accent sur l'information d'un domaine et sur sa représentation. Elle est surtout adoptée par les méthodes d'ingénierie de domaine [Cauvet *et al.*, 2005].

Cette thèse s'inscrit dans la troisième tendance sur laquelle nous nous sommes basés pour choisir, pour leur clarté et le degré de maturité qu'elles ont atteint, trois approches de modélisation de CM, que nous présentons dans cette partie. Il s'agit de l'approche Business Component [Herzum *et al.*, 2000], l'approche composant de domaine [Cauvet *et al.*, 2005] et l'approche composant métier Symphony [Hassine, 2005].

1.3.1 Approche Business Component [Herzum *et al.*, 2000]

Un composant métier selon l'OMG [Herzum *et al.*, 2000] est l'implémentation logicielle d'un concept ou d'un processus métier autonome. Il est constitué de l'ensemble des artefacts nécessaires pour représenter, implémenter et déployer un élément métier autonome et réutilisable d'un système d'information large et distribué. [Herzum *et al.*, 2000] a donné cinq dimensions d'un CM.

- *Niveau de granularité* : faible, moyen ou fort.
- *Unité de composition* : classe, composant distribué, composant métier, système de composants métier ou fédération de systèmes de composants métier.
- *Type de connaissance* : ce modèle utilise 3 types de connaissances (processus, entité et utilitaire) ainsi qu'un type supplémentaire appelé auxiliaire pour désigner les CM sur lesquels s'appuient les trois types précédents.

- Niveau de distribution : utilisateur (présentation), entreprise (métier) et ressource (données persistantes).
- *Cycle de vie* : un CM est composé d'un ensemble d'artefacts permettant sa spécification. Ils couvrent à la fois les niveaux de distribution et l'ensemble du cycle de développement.

La Figure 2-1 présente un exemple de CM selon cette approche. Cet exemple illustre le CM *Facture* modélisé par la composition de trois CM : *Bon de commande*, *Client* et *Produit*. La composition est réalisée à travers les interfaces de chacun de ces CM. Par exemple, le composant *Bon de commande* a besoin d'un composant qui offre l'interface *Article.*

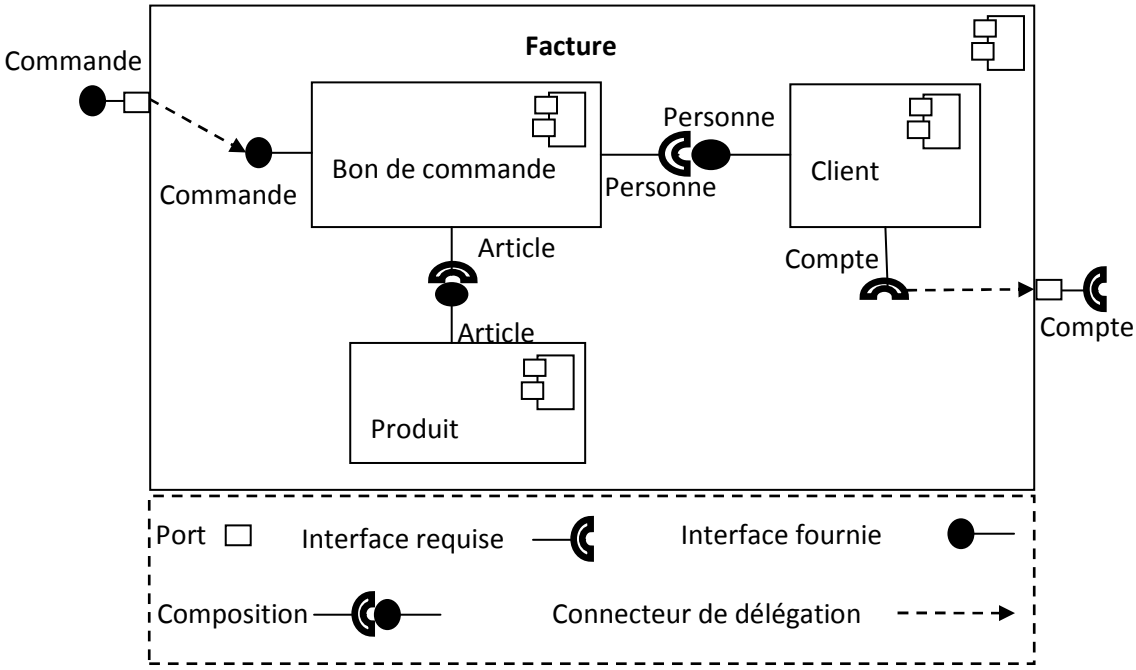


Figure 2-1 : Exemple d'un CM [Herzum *et al.*, 2000]

1.3.2 Approche composant de domaine [Cauvet *et al.*, 2005]

Dans cette approche, un composant est considéré comme « une solution à un problème d'ingénierie. Il définit également le contexte pour lequel cette solution est adaptée ».

Un composant est décrit sous forme d'un triplet de la forme <nom, descripteur, réalisation>, où descripteur = <intention, contexte >. Le descripteur exprime un problème de modélisation conceptuelle à résoudre dans un cadre particulier. La réalisation fournit une solution à ce problème. Elle prend la forme d'un fragment de schéma conceptuel ou d'un fragment de méthode d'ingénierie. La réalisation est la partie réutilisable du composant. Au contraire, le descripteur guide la réutilisation du composant.

En outre, ce modèle de composant de domaine distingue trois niveaux de connaissances : le niveau tâche (les actions à réaliser), le niveau inférence (les manières de faire les actions) et le niveau connaissances du domaine (les objets sur lesquels portent les actions ainsi que les moyens utilisés pour leur mise en œuvre). Un exemple de composant de domaine de [Cauvet *et al.*, 2005] est présenté dans la Figure 2-2.

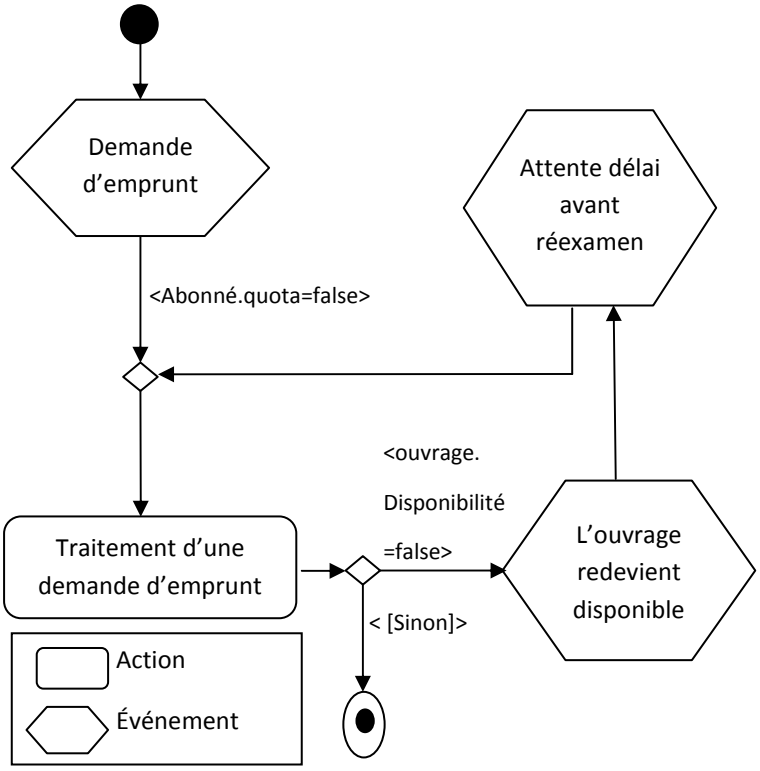
<table>
<tr><td>Nom</td><td colspan="3">Traitement d'une demande d'emprunt avec enregistrement du rejet de la demande en cas de dépassement de quota et réexamen en cas d'indisponibilité de l'ouvrage</td></tr>
<tr><td rowspan="4">Descripteur</td><td>Intention</td><td colspan="2">(organiser)_{Action} (l'activité « traitement d'une demande d'emprunt ») _{Cible}</td></tr>
<tr><td rowspan="3">Contexte</td><td>Domaine</td><td>(Gérer)_{Action} (une bibliothèque) _{Cible}</td></tr>
<tr><td>Processus</td><td>(Gérer)_{Action} (les prêts de livres) _{Cible}</td></tr>
<tr><td>Règle (s)</td><td>(accepter)_{Action} (la demande) _{Cible} (si le livre est disponible et si l'abonné n'a pas dépassé son quota) _{Précision}
(enregistrer)_{Action} (la demande rejetée) _{Cible} (si l'abonné a dépassé son quota d'emprunts simultanés) _{Précision}
(réexaminer) _{Action} (la demande) _{Cible} (si le livre est indisponible, dès disponibilité) _{Précision}</td></tr>
<tr><td rowspan="2">Réalisation</td><td>Solution</td><td colspan="2"> 
</td></tr>
<tr><td>Points d'adaptation</td><td colspan="2">Il est nécessaire de fixer le délai de réexamen de la demande dans le cas où celle-ci serait rejetée pour cause d'indisponibilité de l'ouvrage. Ce délai peut être nul (réexamen immédiat dès disponibilité de l'ouvrage) ou compté en jours ou encore être fonction du nombre de demandes actuellement en attente.</td></tr>
</table>

Figure 2-2 : Exemple de composant de domaine [Cauvet *et al.*, 2005]

1.3.3 Approche composant métier Symphony [Hassine, 2005]

Le modèle de CM Symphony [Hassine, 2005] spécifie trois types de CM. Aux deux catégories fondamentales de CM processus et entité, Symphony rajoute celle des CM « Données » (appelé utilitaire au § 1.2.1) qui représentent les données de références des CM processus et entité.

Les CM Symphony sont décrits à chaque niveau d'abstraction par un ensemble d'artefacts de modélisation. Chaque niveau ou phase de développement correspond à une vue du CM. Chacune de ces vues, à l'exception de la vue d'analyse, est composée de diagrammes UML standard.

C'est ainsi que par exemple, un CM Entité ou Données au niveau de la phase de spécification des besoins, est modélisé par un paquetage regroupant les cas d'utilisation qui représentent les services qu'il propose. Un CM « Processus » est décrit, à ce niveau, par un diagramme d'activités mettant en évidence les différentes tâches nécessaires à sa réalisation. Tous les artefacts sont complétés par des descriptions textuelles.

L'architecture des CM Symphony est une structuration inspirée de la technique CRC (Classe-Responsabilité-Collaboration) [Wirfs-Brock *et al.*, 1990]. Un CM est modélisé par un paquetage composé de trois parties :

- « *Ce que je sais faire* » : une partie *contractuelle avec l'extérieur* représentée par une classe *interface* qui joue le rôle de la porte d'entrée du CM,
- « *Ce que je suis* » : une partie *structurelle* représentée par deux classes : la classe *maître* et la classe *partie* encapsulant la structure interne du CM,
- « *Ce que j'utilise* » : une partie *collaborative* représentée par la classe *rôle* qui formalise les différentes collaborations avec les autres CM du système.

a) Classe Interface

Cette classe correspond à la porte d'entrée du CM. Elle définit les opérations réalisées par le CM dont l'exécution laisse le composant dans un état cohérent. La classe interface permet d'accéder à la structure interne du CM et d'exécuter les services qu'il offre. Elle définit la signature des opérations représentant les cas d'utilisation pour lesquels le CM a été déclaré responsable.

Cette classe supporte également les opérations qui sont déclenchées par les messages des diagrammes de séquence (définis dans la phase d'expression des besoins). Il s'agit alors d'opérations nécessaires à l'exécution d'un cas d'utilisation. La Figure 2-3 illustre un exemple de trois CM Symphony (*Abonné, Emprunt* et *Œuvre).* La partie interface du CM *Abonné* est matérialisée par la classe « Service Abonné » stéréotypée « interface ».

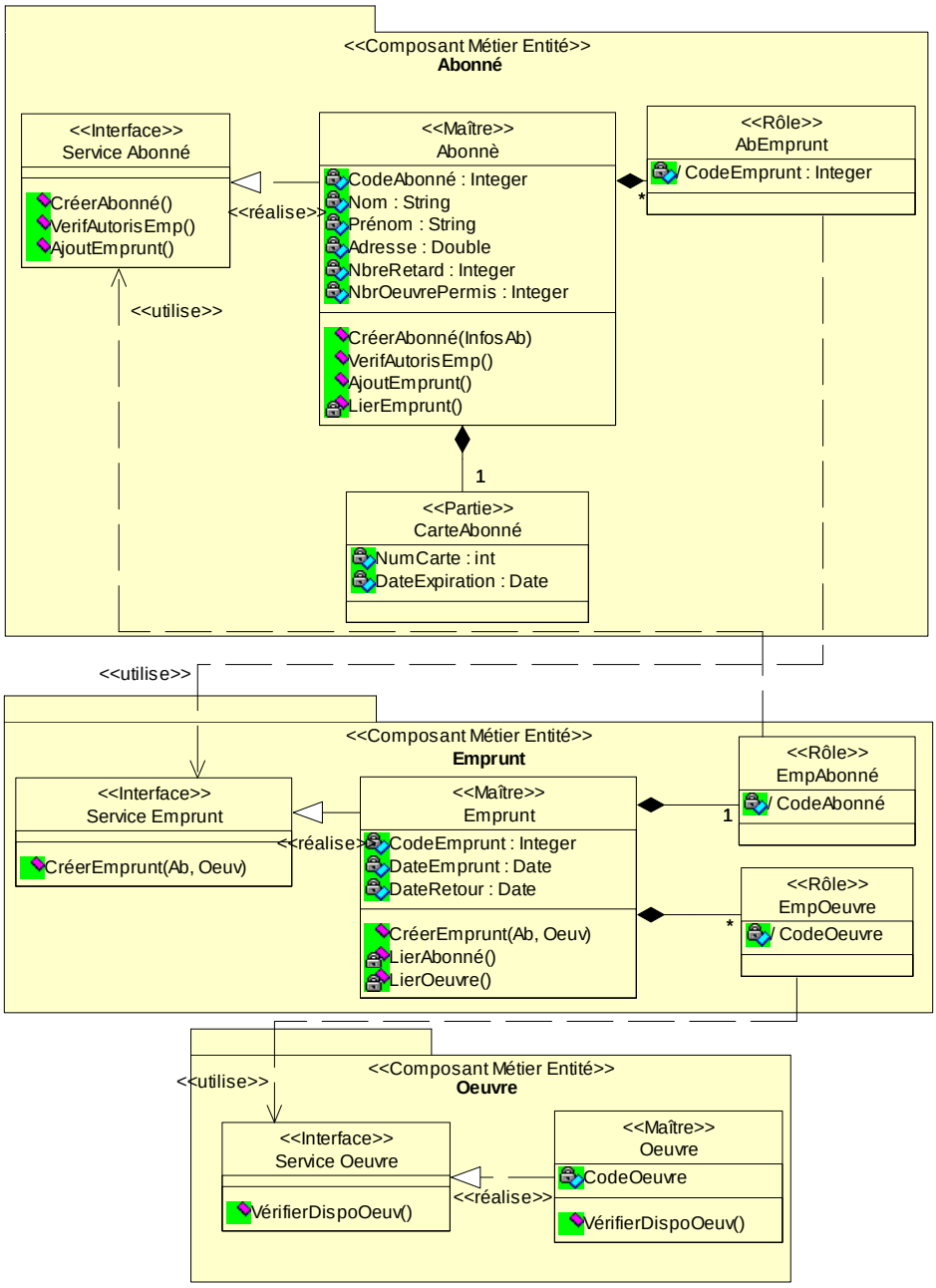


Figure 2-3 : Exemple de CM Symphony

b) Classe Maître

La classe maître est la classe principale du CM pour lequel les opérations sont réalisées. Sa connaissance est nécessaire au préalable de toutes les manipulations du composant. Cette classe est qualifiée de hiérarchiquement supérieure. Elle est autonome : une fois créée, elle ne dépend pas directement d'autres classes si ce n'est au travers de ses propres règles de gestion. Elle justifie son existence par la prise de responsabilité des finalités du système.

Pour être une classe maître, la classe candidate doit vérifier les caractéristiques suivantes : a) être le centre du problème, b) exister en un seul exemplaire lorsque l'on traite le problème **:** une classe maître est mono-instanciée.

La classe maître décrit les éléments d'identification du CM. Ces éléments facilitent en particulier la recherche du CM dans le cas où un autre CM client a besoin de l'un de ses services. Dans l'exemple de la Figure 2-3, la classe Abonné stéréotypée *«* Maître » est la classe maître du CM *Abonné.*

c) Classe Partie

C'est une classe complémentaire de la classe maître ayant un intérêt dans son contexte. Elle s'identifie par son attachement à la classe maître. Une classe partie est reliée à la classe maître par une relation de composition. La classe partie permet de structurer certains attributs de la classe maître. Dans notre exemple, un abonné peut avoir une *carte d'abonné*. Le diagramme de la Figure 2-3 introduit le concept de carte d'abonné comme une classe partie stéréotypée « Partie ».

Ce découpage fournit une structuration des données métier à partir de la classe maître pour permettre une mise en valeur : a) d'un aspect métier de l'objet métier, b) de multiples occurrences d'un concept, c) d'un aspect optionnel d'une caractéristique du CM.

d) Classe Rôle

Une classe rôle décrit ce qu'un CM utilise. Elle met en évidence les *collaborations* réalisées par un CM et représente un fournisseur de services auprès du CM client. En UML, cette dépendance est représentée par un lien de dépendance de stéréotype « utilisation ». Les attributs importés dans le rôle sont représentés par la notation d'attribut dérivé " / ". Dans notre exemple, le CM *Abonné* possède une classe rôle stéréotypée « Rôle » qui lui permet l'utilisation des services fournis par le CM *Emprunt*. De la même manière, le CM *Emprunt* possède deux rôles : un sur le CM *Abonné* et un autre sur le CM *Œuvre*.

1.4 Évaluation des approches

Le Tableau 2-3 compare les approches citées ci-dessus selon les critères de classification de CM. L'étude comparative de ces approches démontre des points de convergence au niveau de certains critères. Nous soulignons particulièrement l'analogie des valeurs assignées aux critères « Type de connaissance » et « Langage de représentation » des CM selon chaque approche. Cette analogie met l'accent sur la nécessité de manipulation de différents types de CM ainsi que le besoin de les modéliser par un langage standard supporté par la plupart des outils de développement. Cette étude dévoile aussi le besoin d'un CM dont la solution aurait un degré de complétude élevé. En effet, la conception d'un CM à plusieurs vues de développement apporterait une meilleure qualité de conception, en ayant une spécification plus complète.

		Modèles de Composants Métier		
		Business Component **[Herzum *et al.*, 2000]**	Composant de domaine **[Cauvet *et al.*, 2005]**	Composant Métier Symphony **[Hassine, 2005]**
Critères	**Visibilité**	Noire/Blanche/Verre	Blanche	Blanche
	Granularité	Faible/Moyenne/Forte	Faible/Moyenne	Faible/Moyenne
	Unité de composition	Classe/Composant distribué/Composant métier/Système de composants métier/Fédération de systèmes de composants métier	Objet/Activité/But	Classe/ Composant Métier Symphony
	Variabilité	Technique	Conceptuelle	----------
	Type de connaissance	Entité/Processus/ Utilitaire/ Auxiliaire	Entité/Processus	Entité/Processus/ Donnée
	Nature de solution	Logicielle	Conceptuelle	Conceptuelle
	Langage de représentation	Extension d'UML	Extension d'UML/ langage naturel	Extension d'UML/ langage naturel
	Complétude	Structurelle	Dynamique/Structurelle	Structurelle
	Couverture	Métier/Entreprise	Fonctionnelle/Métier/Entreprise	Métier
	Portée	Analyse/Conception/ Implantation	Analyse/Conception	Analyse/Conception/ Implantation

Tableau 2-3 : Évaluation des approches de modélisation de CM

1.5 Synthèse

Cette partie a présenté l'approche composant métier avec ses spécificités. En effet, un CM caractérise un domaine ou un secteur d'activité particulier et matérialise des invariants récurrents concernant des besoins du même métier. Dans cette partie, nous avons tenu à rassembler différentes définitions attribuées au concept CM, formulant ainsi une définition générale. Ensuite, nous avons effectué une étude comparative de différentes approches de conception de CM s'inscrivant dans la tendance actuelle de spécification d'un CM de nature conceptuelle.

La mise en œuvre des différentes approches nécessite bien évidemment la mise en place d'un processus de spécification ainsi que d'un processus de réutilisation de CM. Ces deux processus seront présentés dans la partie suivante dans le cadre des méthodes de développement basées composants.

2 Méthodes de développement basées composants

L'ingénierie de systèmes d'information à base de composants est devenue largement adoptée et utilisée, et a fait émerger des méthodes et outils innovants. Une telle approche doit permettre de réduire le temps de conception des systèmes d'information, d'en améliorer la qualité et d'en faciliter la maintenance.

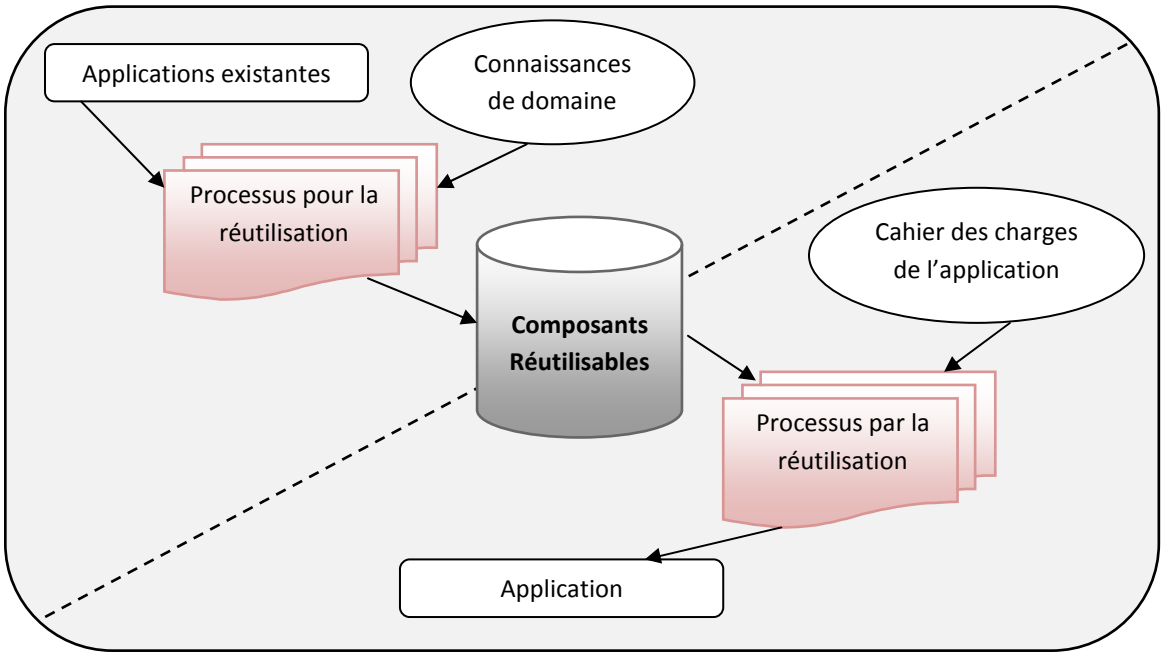


Figure 2-4 : Processus pour la réutilisation Vs Processus par la réutilisation [Oussalah *et al.*, 1999]

En effet, les méthodes de développement basées composants telles que **Catalysis** [D'Souza *et al.*, 1998], **RUP** [IBM, 2009], **Select Perspective** [Select, 2009] et **Symphony** [Hassine, 2005], ont remplacé les démarches traditionnelles, par l'introduction dans leur cycle de développement, de deux processus liés à la réutilisation (Cf. Figure 2-4) :

- Processus *pour la réutilisation de composants* (design for reuse) : activités de production d'artefacts réutilisables qui soulèvent les problèmes d'identification, de représentation et d'organisation des composants.
- Processus *par la réutilisation de composants* (design by reuse) : activités de développement d'applications par réutilisation de composants. Cette approche fait intervenir les problèmes de recherche, d'adaptation et de composition de composants.

Le degré de prise en charge de ces deux processus varie cependant d'une méthode à une autre.

Avant de définir l'ensemble des axes sur lesquels nous nous basons pour comparer un certain nombre de méthodes, nous détaillons ci-dessous la méthode Symphony qui sera utilisée plus loin dans ce rapport.

2.1 Méthode de développement Symphony

Symphony est une méthode de développement élaborée au sein de la société *Umanis*[1] et formalisée au sein de l'équipe SIGMA du laboratoire LSR dans le cadre de la thèse d'Ibtissem Hassine [Hassine, 2005]. Elle s'appuie sur les bonnes pratiques de développement orienté objet dont l'objectif majeur est de satisfaire le client en répondant aux exigences fixées par le cahier des charges et au développement rapide des applications. Cette méthode s'appuie sur le langage unifié UML et est construite autour d'un certain nombre de pratiques. Symphony est itérative, orientée utilisateur et pilotée par les cas d'utilisation, orientée objet métier, et adopte un modèle de cycle de vie en Y.

a) Cycle de vie en Y : séparation des aspects fonctionnels et des aspects techniques

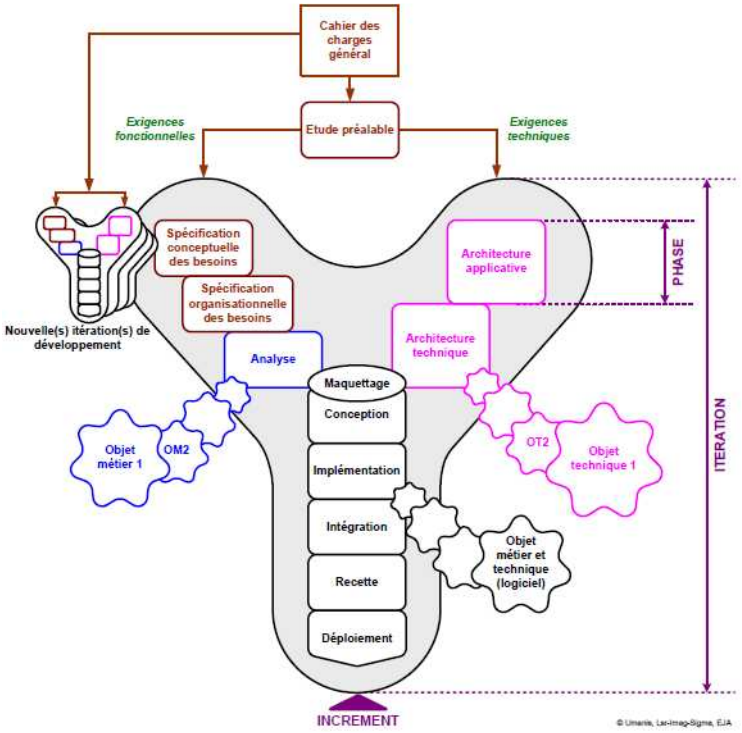


Figure 2-5 : Cycle de vie en Y de la démarche Symphony

Symphony sépare l'étude des besoins fonctionnels de celle des besoins techniques et ceci dès le début du cycle de vie des applications. Les deux aspects fonctionnels et techniques sont mis en évidence en adoptant un modèle de cycle de vie en Y [André, 1994] [Larvet, 1994] [Rocques *et al.*, 2004] (Cf. Figure 2-5).

Ce modèle de cycle de vie s'articule autour de trois branches : la branche gauche capitalise la connaissance du métier, la branche droite capitalise un savoir-faire technique et la branche commune consiste à fusionner les résultats des deux branches gauche et droite afin d'assurer la réalisation du système.

[1] Umanis (site Internet : http://www.umanis.com/fr).

b) Une démarche basée sur la modélisation UML

De nombreux formalismes ont été proposés dans le contexte de la conception des systèmes d'information, comme les formalismes orientés objet, en particulier UML (Unified Modelling Langage) [Muller, 1998].

Symphony est une méthode basée sur la modélisation UML, motivée par le fait que ce langage propose un ensemble de modèles adaptés aux différentes phases de développement. Ses différentes représentations graphiques standardisées et synthétiques facilitent les échanges au sein de l'équipe de développement ainsi que l'élaboration de la documentation des systèmes.

c) Une démarche itérative

La démarche Symphony préconise un développement itératif (Cf. Figure 2-6) dans lequel le développement s'organise à travers un ensemble de cycles de développement en Y (premiers concepts apparus sous le nom de développement en spirale et de développement évolutif par B.W. Boehm [Boehm, 1988]).

Chaque cycle de développement, appelé aussi itération, est centré sur un processus métier ou le raffinement d'un processus métier identifié, décrit et pondéré lors de la phase d'étude préalable. Cette phase, précédant le processus de développement, consiste à faire une première reformulation et expression des besoins fonctionnels sous forme de processus métier.

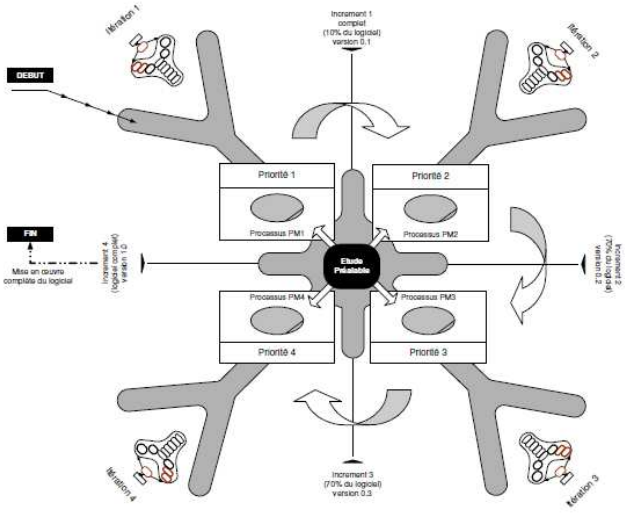


Figure 2-6 : Processus itératif de Symphony : modèle en flocons

d) Une démarche pilotée par les cas d'utilisation

La démarche est orientée utilisateur et pilotée par les cas d'utilisation. En effet, elle intègre les besoins et les usages réels des utilisateurs dès les phases amont du développement. Ces besoins sont alors modélisés par des cas d'utilisation qui assurent la cohésion des activités et guident le processus de développement dans son ensemble. Les cas d'utilisation constituent un mécanisme essentiel de traçabilité entre modèles (Cf. Figure 2-7).

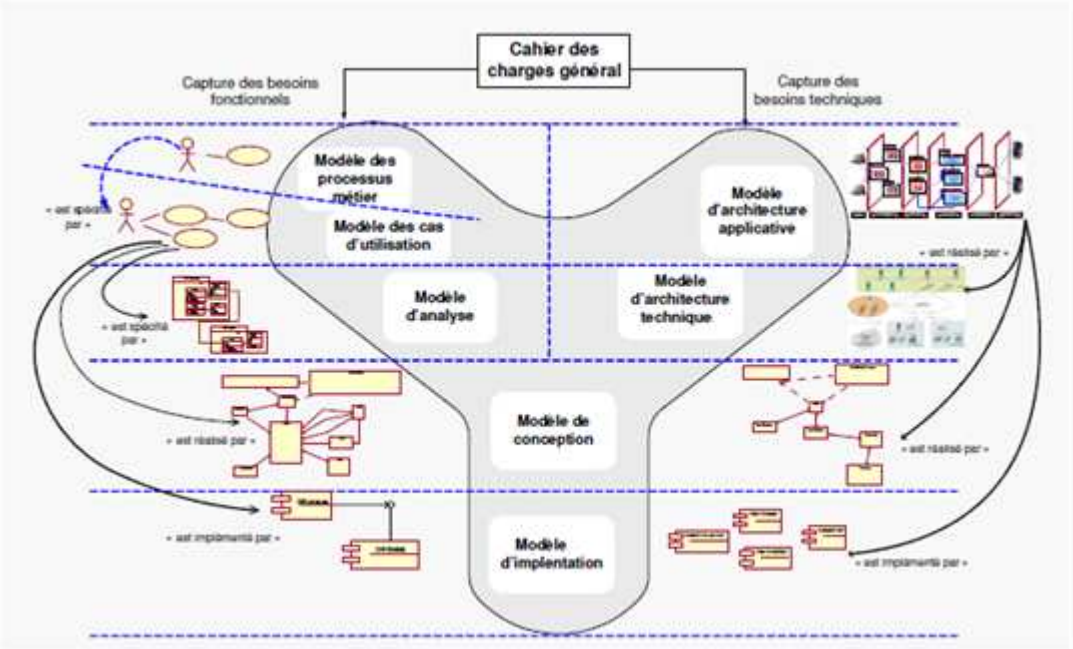


Figure 2-7 : Mécanisme de traçabilité basé sur les cas d'utilisation

e) Une démarche orientée objet métier

La démarche est orientée objet métier car l'application est vue, tant au niveau conceptuel que logiciel, comme un assemblage d'objets métier indépendants et interconnectés. Cette pratique garantit une bonne modularité des spécifications et facilite la réutilisation.

La Figure 2-8 met en évidence que les objets métier (resp. objets techniques) sont identifiés et analysés dans la branche fonctionnelle (resp. technique). Dans la branche centrale du Y, les objets métier conceptuels sont progressivement transformés en objets métier logiciels par intégration des choix techniques spécifiés dans les objets techniques.

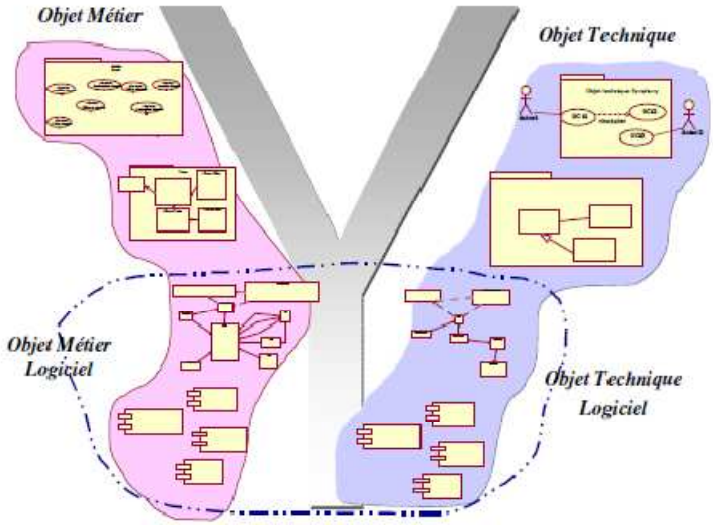


Figure 2-8 : Typologie des objets Symphony

2.2 Cadre de référence

Nous proposons un cadre de comparaison des méthodes basées composants divisé en deux axes : l'ingénierie des composants et la réutilisation des composants.

2.2.1 Axe ingénierie des composants

- **Analyse de domaine** : L'analyse de domaine est définie dans [SEI, 2007] comme l'activité d'identification et de représentation d'informations pertinentes pour un domaine donné (une classe de systèmes similaires partageant les mêmes fonctionnalités). L'analyse de domaine a pour but d'acquérir les connaissances sur un champ d'applications et de les structurer dans un *modèle de domaine* suffisamment générique pour constituer un cadre de référence pour les développements de tous les systèmes d'un domaine donné. Ce cadre de référence exprime en termes de solutions les besoins souvent pris en compte dans les différents systèmes existants.
- **Identification des composants** : L'identification des composants doit, à partir du modèle du domaine, permettre de comprendre les finalités attendues du composant. Ces finalités sont exprimées sous forme de classes qui collaborent entre elles et qui sont regroupées dans des packages (modules), en assurant un degré élevé de cohésion entre les classes du même package et un degré de couplage faible avec les classes des autres packages.
- **Conception des composants** : Une fois le composant identifié, chaque package doit être spécifié en tenant compte de la cohérence et de la traçabilité entre les différentes phases de spécification.

2.2.2 Axe réutilisation des composants

- **Processus d'imitation :** Il s'agit de l'ensemble des activités liées à la manipulation des composants pour la conception d'un SI à base de composants, à savoir la recherche, l'adaptation et la composition.
- **Niveau de réutilisation :** Le niveau de réutilisation détermine la phase de développement où le composant est introduit (analyse, conception, ou code).

2.3 Méthodes basées composants : comparaison

2.3.1 Axe ingénierie des composants

a) Analyse de domaine

Chacune des méthodes **RUP** [IBM, 2009], **Select perspective** [Select, 2009] et **Catalysis** [D'Souza *et al.*, 1998] construit un modèle de domaine en utilisant des concepts et notations différentes.

La méthode **RUP** utilise des extensions d'UML modélisées par le profil de conception métier « UML profil for business modelling ». Ce profil définit un certain nombre d'éléments comme l'acteur métier « business actor », l'entité métier « business entity », les cas d'utilisation métier « business use case »…. Les diagrammes ainsi utilisés sont le diagramme de cas d'utilisation métier, le diagramme d'objets métier, le diagramme d'interaction métier et le diagramme d'activités.

Au niveau de son processus de développement « *Consume* », **Select Perspective** utilise la notation CSC Catalyst pour la modélisation des processus métier (BPM, *Business Process Modeling*). Lors de la définition des besoins métiers, la méthode utilise les diagrammes BPM suivants : PHD (*Process Hierarchy Diagram*) pour la hiérarchie des processus et PTD (*Process Thread Diagram*) pour l'enchaînement des processus.

La méthode **Catalysis** utilise les mêmes concepts et notations tout au long de son cycle de développement. En plus des concepts et notations UML, elle utilise deux concepts spécifiques : type d'objets (utilisé à la place de la classe dans les premières phases de développement) et type de collaboration (pour modéliser les interactions entre les objets).

b) Identification des composants

Dans **RUP** [IBM, 2009], l'identification d'un composant se base sur la génération d'un modèle de besoin par génération de cas d'utilisation à partir du modèle métier ; un cas d'utilisation est créé à partir d'une activité [Molina *et al.*, 2000]. Les règles métier sont représentées sous forme de pré et post conditions.

Dans **Select perspective** [Select, 2009], la notion de composant métier est introduite au niveau de l'étude des besoins. De ce point de vue, la définition d'un composant consiste en la cohésion de fonctionnalités connexes, accessibles par une interface et encapsulées par une implémentation. Dans **Symphony** [Hassine, 2005], l'identification des composants repose sur la modélisation du modèle métier par des diagrammes de cas d'utilisation et des diagrammes de séquence qui représentent leurs dynamiques respectives. Ces cas d'utilisation sont regroupés à la fin de cette phase dans des entités capables de les assumer, et qualifiés d'objets métier.

c) Conception des composants

En terme de conception de composants, **RUP** [IBM, 2009] et **Select perspective** [Select, 2009] formalisent les composants en langage UML, en se basant sur les notions de classes ou de sous-systèmes assortis d'une ou plusieurs interfaces permettant de préciser ou non les services proposés. En terme d'implémentation, les deux méthodes ont recours aux diagrammes d'implémentation et de déploiement d'UML [OMG, 2007]. Les composants sont représentés en tant que type dans les diagrammes de composants et en tant qu'instances d'exécution dans les diagrammes de déploiement. Dans **Catalysis** [D'Souza *et al.*, 1998], la phase de conception apparaît de manière différente. En effet, il est question de spécification de comportement externe d'un composant. Pour ce faire, la méthode utilise la notion de type. Quant à **Symphony** [Hassine, 2005], la conception des composants repose sur le modèle Symphony (§1.3.3) consistant en un diagramme de classes étendu.

2.3.2 Axe réutilisation des composants

a) Processus d'imitation

Le processus d'imitation est souvent défini comme un processus parallèle à un processus de développement basé composants. Ce processus peut être développé et amélioré séparément. À l'heure actuelle, seule la méthode **Select perspective** prend en compte clairement le double processus « Pour » et « Par » la réutilisation. La Figure 2-9 extraite de [Hassine, 2005] présente

le modèle *SMaC* (Supply Manage Consume) qui illustre la prise en charge de ces deux processus par cette méthode.

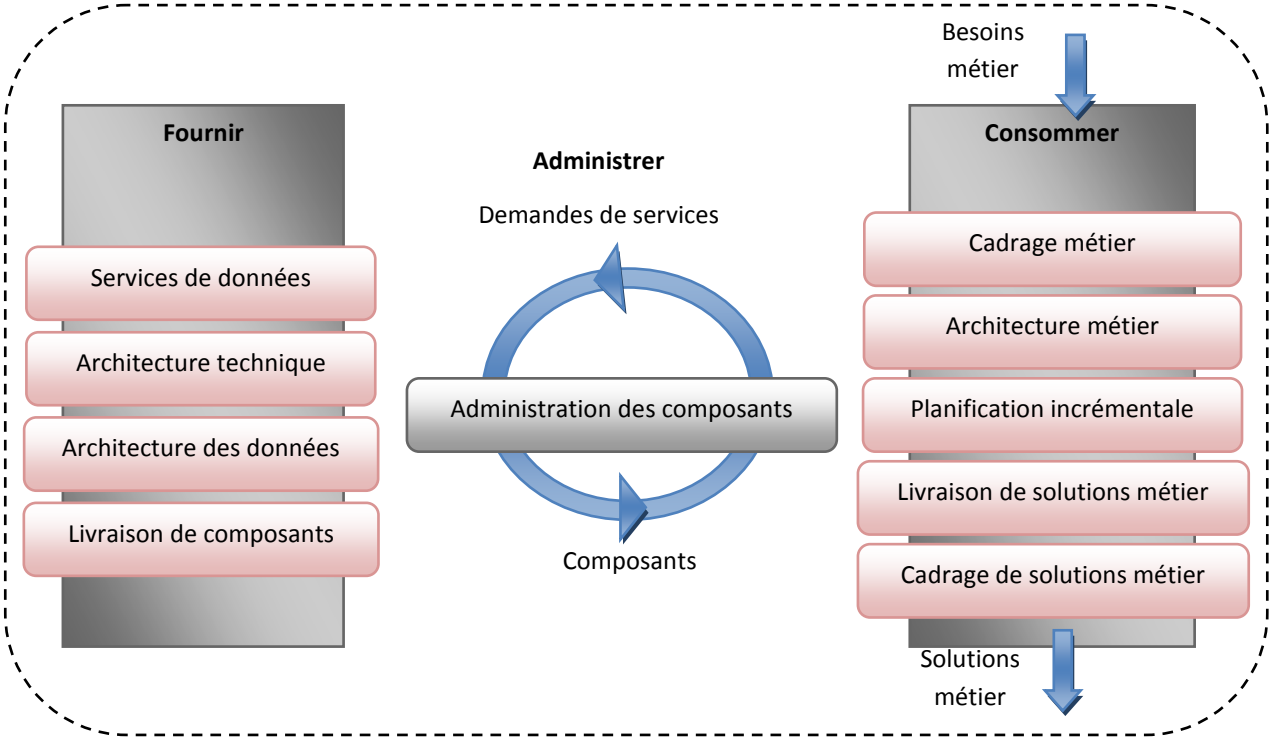


Figure 2-9 : Framework de processus de développement SMaC

b) Niveau de réutilisation

Le niveau d'introduction de la notion de composant varie d'une méthode à une autre. **Symphony** et **Select Perspective** se distinguent des autres méthodes par l'identification des composants métier et de leurs responsabilités dans le système d'information cible, dès l'expression des besoins. Les autres méthodes introduisent les composants à partir de la phase d'analyse.

2.4 Évaluation des approches

Le Tableau 2-4 compare les méthodes citées ci-dessus selon le cadre de référence.

	Ingénierie des composants			**Réutilisation des composants**	
	Analyse du domaine	Identification des composants	Conception des composants	Processus d'imitation	Niveau de réutilisation
RUP [IBM, 2009]	Diagrammes de cas d'utilisation métier, d'objets métier, d'interaction métier et d'activités	Diagramme de cas d'utilisation	Diagramme de classes étendu	---	Étude des besoins

Catalysis [D'Souza *et al.*, 1998]	Type objet et type collaboration	---	Collaboration de classes avec la notion type	---	Analyse
Symphony [Hassine, 2005]	---	Diagramme de cas d'utilisation et de séquence	Modèle Symphony (Diagramme de classes étendu)	---	Analyse
Select perspective [Select, 2009]	CSC Catalyst Process Hierarchy, Process Thread	---	Diagramme de classes étendu	Modèle SMaC	Étude des Besoins

Tableau 2-4 : Comparaison des méthodes de développement basées composants

2.5 Synthèse

Dans cette partie, nous avons présenté une description générale des méthodes basées composants, en nous intéressant en particulier à la prise en charge ou non du double processus pour et par la réutilisation.

Selon cette étude, la plupart des méthodes de développement à base de composants souffrent d'un certain nombre de faiblesses. Le principal problème qui freine la réutilisation est que le processus de développement est principalement axé « développement de systèmes d'information» (la prise en charge, par exemple, des techniques d'abstraction des connaissances de domaine reste absente). Généralement, ces méthodes n'utilisent pas clairement d'infrastructure de réutilisation, et la conception des composants passe par l'encapsulation des objets métier liés à un SI dans des entités appelées composants.

En outre, parmi les critères mis en avant lors de la conception des CM, nous avons distingué le critère variabilité défini comme une propriété essentielle qui maximise la réutilisabilité du composant. Ce critère est traité dans la partie suivante.

3 Variabilité

La mise en œuvre de l'ingénierie des CM pose un certain nombre d'exigences. Le principal besoin rencontré est d'assurer la production de composants dont la réutilisation est efficace en prévoyant une activité d'adaptation qui ajuste ces composants selon le contexte de son utilisation. Cela signifie que le modèle de composants doit prendre en considération la capacité de son adaptation ainsi que les effets produits par des éventuels changements.

De cette exigence, nous avons tenu à expliciter la notion de variabilité, définie comme la capacité d'un système à s'adapter, se spécialiser et se configurer en fonction du contexte de son utilisation [VanGurp *et al.*, 2000]. La variabilité doit être prise en compte d'autant plus que les besoins des utilisateurs ne sont généralement pas clairs, mais flous ou évolutifs.

La variabilité a été introduite dans différents domaines, plus particulièrement dans le développement des lignes de produits [Ziadi, 2004], [Pohl *et al.*, 2005] et l'ingénierie de domaines [Kang *et al.*, 1990] [Kang *et al.*, 1998] [Bashroush *et al.*, 2008]. Par contre, elle a été

peu exprimée au niveau de la conception des composants réutilisables, même s'il y a eu certaines tentatives comme celle de [Ramadour, 2001].

D'autre part, la variabilité a fait l'objet d'études au niveau des spécifications techniques et de l'implémentation d'un système [Bachmann *et al.*, 2003], [Bosch, 2007] mais, paradoxalement n'a que rarement été abordée dans les plus hauts niveaux d'abstraction.

Cette partie a comme objectif la présentation du concept de variabilité. Elle s'articule de la manière suivante : dans un premier temps, nous présentons différentes définitions attribuées à ce concept (§3.1). Nous illustrons ensuite (§3.2) les concepts de base de la variabilité qui permettent par la suite de comparer un ensemble d'approches de modélisation (§3.3) de la variabilité dans un paragraphe d'évaluation (§3.4), avant un dernier paragraphe de synthèse (§3.5).

3.1 Définitions

L'expansion de la recherche dans le domaine de la variabilité a donné lieu à plusieurs définitions pour ce concept. La plupart des auteurs ont adopté la définition citée dans [VanGurp *et al.*, 2000] : « Variability is the ability to change or customise a software system », traduit en « la capacité d'un système ou d'un artefact à être changé, personnalisé ou configuré selon un contexte d'utilisation particulier ». Cette définition est une définition de base qui a été personnalisée selon le domaine d'étude.

Selon l'ingénierie des domaines, la variabilité est considérée comme la représentation des caractéristiques qui sont définies en fonction des aspects essentiels et visibles du domaine, eux-mêmes identifiés à partir des connaissances de domaine [Kang *et al*., 1990].

Dans l'approche lignes de produits, la variabilité permet la conception d'une architecture permettant de définir plusieurs produits [Ziadi, 2004] [Pohl *et al.*, 2005]. Les membres d'une ligne de produits sont caractérisés par leurs points communs et leurs points variables appelés « points de variation ».

Dans l'ingénierie des composants orientés problème [Ramadour, 2001], la variabilité est exprimée à travers les différentes solutions possibles d'un même problème, ce qui permet de maximiser la réutilisation des connaissances du domaine.

Dans l'approche Web sémantique, la variabilité est décrite comme un principe qui permet d'associer à un même concept de plusieurs points de vue. Ainsi, la variabilité vise à associer aux systèmes une connaissance qui guide leur usage [Guzélian *et al*., 2005]. [Sipka, 2005] rejoint cette dernière définition en décrivant la variabilité comme la représentation des instances d'un concept et la relation entre ses caractéristiques variables. Une caractéristique représente l'intention d'un concept et ses variantes représentent l'extension de ce concept.

En résumé de cette partie, la variabilité est une technique qui sert à contrôler les parties communes et variables des artefacts d'un système dans le but de faciliter sa réutilisation dans un contexte donné.

3.2 Cadre de référence

La variabilité a fait l'objet de plusieurs travaux de recherche dans plusieurs contextes. Pour comparer ces travaux, nous définissons un cadre de référence composé de six axes : a)

Contexte de l'approche, b) Finalité de l'approche, c) Complétude, d) Identification de la variabilité, e) Représentation de la variabilité et f) Réduction de la variabilité. Ces six axes sont présentés dans la suite.

3.2.1 Définition du cadre de référence

a) Contexte de l'approche

Cet axe porte sur la description des principes et objectifs d'une approche donnée, en analysant la nature des artefacts produits par le processus de représentation de la variabilité. Cet axe précise aussi le domaine dans lequel chacune des approches a fait ses preuves.

b) Finalité de l'approche

Cet axe vise à décrire la portée des solutions développées selon le concept de la variabilité. Ainsi, nous considérons que la variabilité peut être appliquée selon deux aspects : i) variabilité pour la ***personnalisation*** et ii) variabilité pour la ***réutilisation***.

- ***La variabilité pour la personnalisation***

Ce type de variabilité est conçu pour des besoins d'adaptation des solutions offertes par un système : soit aux besoins de l'utilisateur final, dans ce cas le système ainsi utilisé est appelé un système *adaptable,* soit à son environnement d'exécution, le système utilisé est appelé un système *adaptatif*. Ces deux notions sont appelées respectivement *adaptation statique* et *adaptation dynamique* selon [Frasincar *et al.*, 2002] et [Kappel *et al.*, 2000].

– *Adaptation statique* : les développeurs d'un composant permettent à l'utilisateur final de configurer et paramétrer ce composant par rapport à ses propres besoins. L'adaptation du composant implique une évolution du contexte d'exécution du composant et peut être explicitée par des interactions entre le composant et son contexte. Il est alors nécessaire de prendre en charge ces interactions au niveau du composant et, par voie de conséquence, leur impact sur l'adaptation de la structure et du comportement du composant. Un exemple de composant statiquement adaptable est présenté dans [Occello *et al.*, 2006] ; il concerne une application de gestion d'agendas adaptables. L'application est constituée de composants dont le rôle est de simuler le fonctionnement d'un agenda de base, c'est-à-dire de permettre d'ajouter, supprimer et consulter des rendez-vous. À l'exécution, les agendas peuvent être adaptés par l'utilisateur final pour modifier leur comportement dans le but d'enrichir dynamiquement l'application. Par exemple, un agenda peut être adapté pour accepter un rendez-vous uniquement sous certaines conditions (disponibilité par exemple). Cette adaptation modifie le comportement de la méthode addRdv() de l'agenda adapté. D'autres formes d'adaptation vont faire intervenir des entités tierces, par exemple, « visualiser les rendez-vous » requiert une IHM, « mémoriser les RDV » requiert une base de données, etc.

– *Adaptation dynamique :* les développeurs d'applications sont aujourd'hui confrontés à des contextes d'exécution de plus en plus variables, qui nécessitent la création d'applications capables de s'adapter de façon autonome aux évolutions de ces contextes. Avec l'émergence de l'informatique nomade, les nouvelles applications doivent être capables de s'adapter elles-mêmes de façon autonome [Kephart, 2002] aux divers contextes d'exécution (ex. position géographique, bruit externe, etc.) auxquels elles sont confrontées,

et aux évolutions dynamiques de ceux-ci. Il devient ainsi nécessaire d'embarquer le programme chargé de l'adaptation dans les applications elles-mêmes. [David *et al.*, 2006] propose que l'adaptation d'un système soit représentée par un programme chargé de (i) l'observation de l'environnement dans lequel évolue le système, (ii) la prise de décision des ajustements à apporter au système suite à la détection d'un changement significatif dans l'environnement, et enfin (iii) l'application des modifications choisies pour adapter le système.

- ***La variabilité pour la réutilisation***

Ce type de variabilité consiste à rendre la variabilité explicite pour la conception d'un artefact réutilisable ; ceci signifie que la spécification de cet artefact définit une partie variable et une partie générique afin d'augmenter son applicabilité dans différentes applications. Ce type de variabilité est orienté concepteur d'applications qui adapte l'artefact selon les besoins de l'application en cours de construction. L'artefact ainsi utilisé est appelé un artefact *adaptable* avant de le réutiliser dans une application. Par exemple, un artefact réutilisable de gestion d'inscription peut contenir les deux fonctionnalités « inscription avec gestion de carte d'abonné » et « inscription sans gestion de carte d'abonné ». Lors de la conception du système, le concepteur doit adapter cet artefact selon les besoins de l'application à concevoir.

c) Complétude

Il s'agit des différents types de variabilité représentés sur un artefact réutilisable. Nous distinguons cinq types de variabilité : variabilité métier, variabilité fonctionnelle, variabilité dynamique, variabilité structurelle et variabilité technique.

- *Variabilité métier* : représente la variabilité des activités représentées par l'artefact. Un artefact peut être décliné de différentes façons selon le SI auquel il se rapporte ;
- *Variabilité fonctionnelle* : représente la variabilité des fonctionnalités offertes par un artefact. Une fonctionnalité peut être choisie ou non par le concepteur lors de la réutilisation de cet artefact ;
- *Variabilité dynamique* : modélise la variabilité des interactions entre les classes du système ainsi qu'une description détaillée de la variabilité fonctionnelle ;
- *Variabilité structurelle* : représente la variabilité des concepts du système ; une structure de données peut varier d'un système à un autre. Par exemple, la structure de données d'une voiture dans une application chez un concessionnaire est différente de celle dans une agence de location de voitures ;
- *Variabilité technique* : représente la variabilité des artefacts de l'implantation d'un système donné.

d) Identification de la variabilité

L'identification de la variabilité est liée à la définition des caractéristiques qui différencient des applications de même nature. Cette phase se base aussi sur l'association de chaque variation identifiée à un ensemble d'alternatives ou variantes.

Par ailleurs, l'identification de la variabilité s'applique en fonction de la distribution de la variabilité dans l'*espace* et dans le *temps* [Bosch *et al.*, 2001].

- *La variabilité dans l'espace* est mise en évidence quand un artefact capture différentes formes en même temps. Par exemple, un artefact de réservation peut se rapporter à plusieurs formes en même temps, selon qu'il s'agisse d'une réservation d'une chambre d'hôtel, d'un ouvrage d'une bibliothèque, d'un billet d'avion, etc.
- *La variabilité dans le temps* concerne le fait qu'il y ait différentes versions d'un artefact à différents moments. Il s'agit donc de l'*évolution* de l'artefact en fonction de l'évolution de son domaine d'application.

e) Représentation de la variabilité

Il est nécessaire de définir un langage de modélisation de la variabilité, afin d'assister le concepteur d'applications à bien prendre en compte ce concept. Nous présentons ici des exigences importantes qui doivent être prises en considération lors de la représentation de la variabilité.

- *Représenter des parties fixes et des parties variables*

Pour représenter la variabilité sur un artefact réutilisable, nous distinguons deux parties essentielles.

– *Parties fixes* : ce sont des éléments récurrents dans des SI de même nature. Ils représentent les structures de l'artefact obligatoirement réutilisables dans chaque système d'information. Ils sont directement intégrés dans le SI en cours de construction.

– *Parties variables* : elles représentent les structures de l'artefact pour lesquelles la réutilisation exige un processus de sélection adapté aux exigences d'un SI particulier. Ces éléments variables peuvent être communs à tous les SI tout en prenant différentes formes, ou ils peuvent être spécifiques à des SI particuliers.

Par ailleurs, un élément variable dans un artefact est représenté par une *unité de variabilité*, il s'agit d'un endroit spécifique auquel une décision est reliée. En effet, l'unité de la variabilité consiste à identifier avec précision la propriété variable d'un artefact. Cela mène à la définition, en premier lieu, du *sujet* et de *l'objet* de la variabilité [Pohl *et al.*, 2005].

– *Le sujet de la variabilité* représente une propriété variable du monde réel. Par exemple, la réservation est un sujet de variabilité.

– *L'objet de la variabilité* représente un exemple particulier du sujet de variabilité. Par exemple, la réservation de train est un objet de la variabilité lié au sujet réservation.

En outre, pour représenter les différences entre des artefacts de la même famille, les concepts de *Points de Variation* (PV) et *de Variantes* (V) sont employés [Jacobson *et al.*, 1997]. Nous définissons ces deux concepts comme suit :

– *Un point de variation* localise un endroit spécifique dans un artefact auquel une décision, prise lors de la conception d'un SI, est attachée.

– *Une variante* représente une réalisation spécifique d'un point de variation. Elle correspond aux solutions alternatives de conception.

Par exemple, lors de la conception du processus de réservation, la fonctionnalité réservation peut être réalisée de différentes manières. Elle est identifiée comme un PV, tandis que la réservation d'une chambre d'hôtel et la réservation d'un billet d'avion constituent, par

exemple, un ensemble de solutions alternatives pour ce PV et sont identifiées comme des variantes.

- *Différencier les types de variation*

La variabilité est définie par l'identification des points de variation et de leurs variantes. Un point de variation doit fournir des contraintes sur le choix de ses variantes, et ces contraintes sont matérialisées par les quatre types de variation suivants [Bachmann *et al.*, 2001] [VanDerMaβen *et al.*, 2002] :

– *Option :* un point de variation de type « *option* » fournit des variantes qui peuvent être sélectionnées ou non : à partir d'un ensemble de n options, k options (k compris entre 0 et n) peuvent être choisies, y compris toutes (n), ou aucune (0). Par exemple, un système de gestion d'une bibliothèque peut proposer ou non l'attribution d'une carte d'abonné. La création de carte d'abonné est identifiée comme optionnelle dans ce système.

– *Alternative :* un point de variation de type « *alternative* » permet le choix d'une seule variante parmi n choix possibles. Ce type de variation est connu sous le nom de la relation XOR entre les variantes de ce point de variation. Par exemple, pour réutiliser un système de gestion d'une bibliothèque, le concepteur doit choisir entre un système qui gère la réservation d'œuvre ou bien un système sans réservation. Ce système propose donc un choix entre deux alternatives : système avec ou sans réservation.

– *Alternative Optionnelle :* un point de variation de type « *alternative optionnelle* » est un point de variation de type option qui offre des variantes alternatives. Ce type de variation permet le choix de 0 ou 1 variante parmi n choix possibles. Par exemple, l'emprunt d'une œuvre dans une bibliothèque peut contenir un point de variation optionnel sur la gestion de garantie. Ce point de variation offre deux alternatives possibles : emprunt avec dépôt de garantie et emprunt sans dépôt de garantie. Le concepteur peut choisir une ou aucune variante parmi ces deux alternatives.

– *Ensemble d'alternatives :* un point de variation de type « *ensemble d'alternatives* » signifie qu'il existe de multiples réalisations de cette variation et qu'au moins une doit être sélectionnée. Par exemple, un système de gestion d'une bibliothèque peut offrir différents types de carte d'abonné : une carte magnétique, une carte à code barre, etc. Au moins un type de carte doit être choisi par le concepteur.

- *Représenter les contraintes de dépendance de la variabilité*

Les artefacts réutilisables supportant la variabilité doivent être caractérisés par des contraintes de dépendance entre variantes et points de variations. En effet, le choix d'un point de variation ou d'une variante peut influencer le choix d'autres points de variation ou variantes. Nous distinguons ici deux types de contraintes inspirés des règles de compositions de FODA [Kang *et al.*, 1990] : l'inclusion et l'exclusion mutuelle.

– La règle *d'inclusion* spécifie que le choix d'un élément variable exige la présence d'un autre élément variable lors de la réutilisation de l'artefact comportant ces éléments.

– La règle *d'exclusion mutuelle* entre deux éléments variables spécifie que la présence de l'un de ces éléments interdit la présence de l'autre lors de la réutilisation de l'artefact comportant ces éléments.

- *Fournir des mécanismes de représentation de la variabilité faciles à exploiter*

Pour communiquer les modèles aux concepteurs d'applications, les notations utilisées pour représenter la variabilité doivent être faciles à lire et à comprendre. La représentation de la variabilité peut être graphique, textuelle ou un mélange des deux. L'avantage de la représentation graphique est que les variations sont facilement repérées par rapport à une pure description textuelle. Ainsi, une notation graphique peut aider à comprendre les parties variables très facilement. En outre, il est important de mentionner que la notation graphique ne doit pas remplacer les descriptions en langage naturel, mais les compléter pour fournir un point de vue différent.

f) Réduction de la variabilité

La réduction de la variabilité consiste principalement à sélectionner les variantes utiles. Cette activité est liée à la portée de la variabilité exprimée et peut se produire au moment de :

- *la conception* : un artefact réutilisable est réduit par le concepteur d'applications avant son utilisation par l'utilisateur final.
- *l'exécution* : la variabilité est résolue après que l'artefact réutilisable ait été délivré et installé dans son environnement d'exécution à l'aide des techniques des langages à objets telles que l'héritage, la liaison dynamique, la surcharge et la délégation.

3.2.2 Synthèse

Le Tableau 2-5 résume les axes de notre cadre de référence ainsi que leurs valeurs.

Axes du cadre de référence	Valeurs	
Contexte de l'approche	Domaine d'application de l'approche	
Finalité de l'approche	Réutilisation / Personnalisation	
Complétude	Métier/ Fonctionnelle/Dynamique/Structurelle/Technique	
Identification de la variabilité	Processus d'identification des connaissances variables	
Représentation de la variabilité	Partie fixe/variable	Oui / Non
	Type de variation	Option / alternative / alternative optionnelle / ensemble d'alternatives
	Contraintes de variabilité	Non/ exclusion/ inclusion
	Notation	Graphique / textuelle
Réduction de la variabilité	Processus de sélection des variantes utiles Conception / exécution	

Tableau 2-5 : Axes du cadre de référence

3.3 Approches de modélisation de la variabilité

Les approches de modélisation de la variabilité présentées dans cette partie sont les plus rencontrées dans la littérature. Il s'agit de a) l'approche **ingénierie de domaine** selon FODA [Kang *et al.*, 1990] et son extension FORM [Kang *et al.*, 1998] [Bashroush *et al.*, 2008] où la variabilité a été proposée pour le développement de logiciels par réutilisation à travers l'analyse de domaine, b) l'approche **lignes de produits** selon [Clauss, 2001], [Ziadi, 2004], [Oliveira *et al.*, 2005] et [Pohl *et al.*, 2005] où la variabilité a été proposée pour représenter les

propriétés communes et discriminantes des applications d'une même famille et enfin, c) l'approche **patrons de conception** selon [Sunyé, 1999] et [Arnaud, 2008] où la variabilité a été proposée pour définir des alternatives de conception offertes par la solution d'un patron.

3.3.1 Ingénierie de domaine

La variabilité a été initialement exprimée dans l'approche FODA (Feature-Oriented Domain Analysis) [Kang *et al.*, 1990] développée au SEI (Software Engineering Institute). Dans la méthode FODA, l'analyse du domaine consiste à étudier les points communs et discriminants entre les systèmes. Ainsi, pour chaque système étudié, des caractéristiques appelées « features » sont identifiées. Trois catégories de caractéristiques sont distinguées : obligatoires, optionnelles et alternatives. Il s'agit d'une méthode d'analyse de domaine utilisée pour développer des logiciels par réutilisation et basée sur le diagramme des caractéristiques.

Un diagramme de caractéristiques constitue une structure arborescente composée d'une racine et de nœuds. La racine représente un concept et les nœuds représentent les caractéristiques. Les caractéristiques communes sont définies en tant que caractéristiques obligatoires qui seront incluses dans tous les systèmes. Les caractéristiques variables sont celles configurables, c'est-à-dire qu'elles ne sont pas nécessairement incluses dans chaque instance du système. La Figure 2-10 illustre un système partiel d'un Guichet Automatique Bancaire (GAB) représenté par un diagramme de caractéristiques.

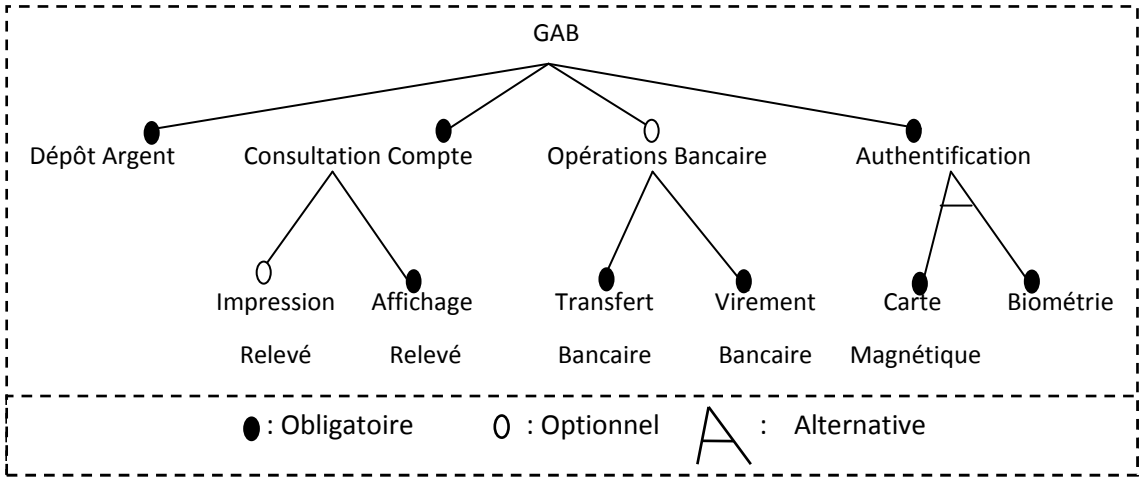


Figure 2-10 : Exemple de diagramme de caractéristiques [Kang *et al.*, 1990]

Dans [Kang *et al.*, 1998], les auteurs proposent FORM (Feature Oriented Reuse Method) comme une extension de FODA. Les extensions proposées concernent la classification des caractéristiques en quatre niveaux (capacité, environnement opérationnel, technologie de domaine, technique d'implémentation). La décomposition du diagramme de caractéristiques dans FORM inclut plus de détails à différents niveaux, mais présente également une plus grande complexité de modélisation.

Les résultats de l'ingénierie de domaine (les modèles de caractéristiques) contiennent de la variabilité. La réduction de la variabilité a donc besoin de décisions (ou de choix) associées aux points de variation modélisés. La notion de modèle de décision est utilisée pour capturer et enregistrer les décisions nécessaires à la réduction de la variabilité au processus d'application, il s'agit d'un développement par la réutilisation.

Bien que les méthodes FODA et FORM aient fourni beaucoup de techniques utiles pour la gestion de la variabilité, un certain nombre de limitations sont restées présentes. Par exemple, la capacité de présenter un grand nombre de points de variabilité avec leurs contraintes de dépendance dans une seule vue demeure un défi. Pour essayer d'alléger cette limitation, [Bashroush *et al.*, 2008] traite le problème en divisant le modèle de caractéristiques en un certain nombre de vues. Le modèle ainsi adopté est appelé « modèle à quatre vues » (the Four Views Model (4VM)), où chaque vue couvre un ensemble spécifique de préoccupations. Les vues adoptées dans le modèle 4VM sont les suivantes :

- *Vue métier* : représente les propriétés liées à la gestion du projet, telles que le coût (ou l'avantage) de l'utilisation d'une caractéristique, le temps d'implémentation d'une caractéristique, etc. Ces propriétés sont utilisées par les chefs de projet pour effectuer le choix d'une variante au moment de la conception d'un produit.

- *Vue comportementale & hiérarchique* : représente la manière dont les différentes caractéristiques sont organisées (habituellement dans une structure arborescente) à côté du comportement attaché à chaque caractéristique.

- *Vue d'interaction & de dépendance* : complémentaire à la vue comportementale et hiérarchique, elle représente les interactions et les relations de dépendance entre certaines caractéristiques.

- *Vue intermédiaire* : représente les endroits où des décisions de conception sont injectées dans le modèle de caractéristiques afin de le lier à l'architecture du système.

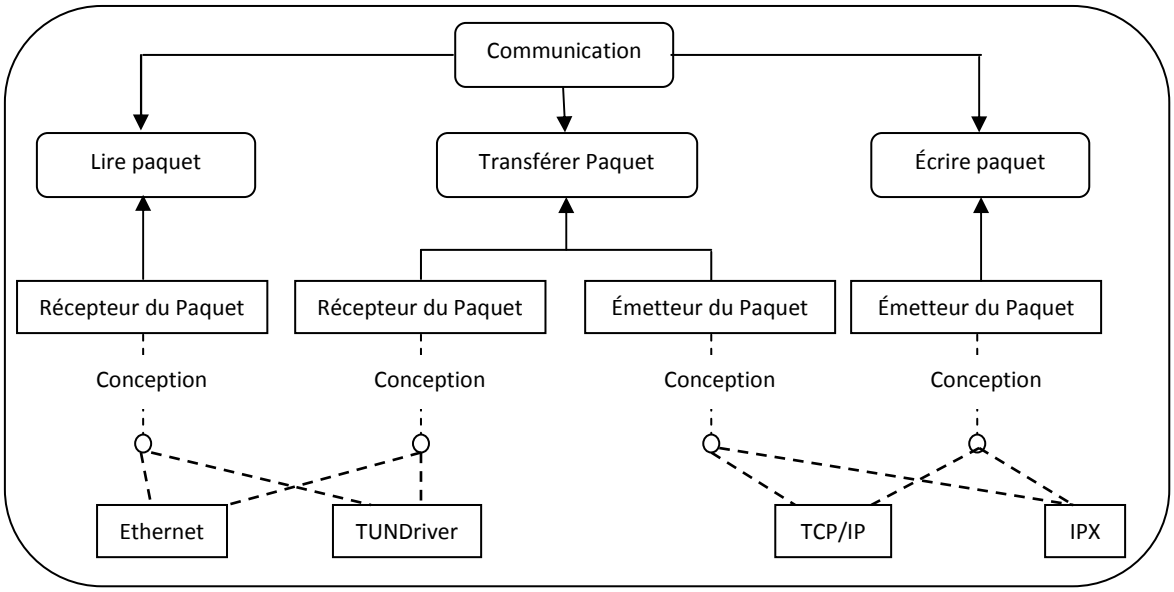


Figure 2-11 : Exemple de vue intermédiaire

La Figure 2-11 représente la vue intermédiaire d'un « émulateur d'un réseau ». Les caractéristiques sont liées de différentes manières :

– *Composition* : une caractéristique est composée de plusieurs sous-caractéristiques (flèche vers le haut).

– *Réalisation* : une caractéristique est réalisée et déployée par l'intermédiaire d'autres caractéristiques (représentée par une flèche vers le bas).

– *Environnement* : relation de liaison d'une caractéristique avec une autre caractéristique d'environnement. Par exemple, la caractéristique « Émetteur du Paquet » est liée aux caractéristiques d'environnements TCP/IP et IPX.

3.3.2 Lignes de produits

Les approches « lignes de produits » considèrent un domaine comme une famille d'applications partageant un ensemble de propriétés communes, et satisfaisant aux besoins spécifiques à ce domaine. Motivée par la diversité des facteurs de variation des logiciels dans certains domaines, cette approche a été adoptée plus particulièrement dans l'industrie. Par exemple, Nokia a opté pour l'approche ligne de produits pour gérer la diversité des logiciels de téléphones mobiles [Maccari *et al.*, 2000] [Maccari *et al.*, 2003].

La variabilité est représentée sur une ligne de produits de deux façons différentes : intégrée dans des diagrammes UML ou séparée des modèles de développement et modélisée dans un modèle de variabilité.

a) Variabilité intégrée dans les diagrammes UML

Les approches intégrant la variabilité dans les diagrammes UML proposent des extensions d'UML, principalement sur les *diagrammes d'activités* [Razavian *et al.*, 2008], les *diagrammes des cas d'utilisation* [Oliviera *et al.*, 2005], les *diagrammes de séquences* [Ziadi , 2004] et les *diagrammes de classes* [Clauss, 2001].

Dans les diagrammes d'activités de [Razavian *et al.*, 2008], la variabilité est exprimée au niveau des activités à travers les stéréotypes « alt_vp » et « variant » pour indiquer l'endroit où il y a une variabilité. Dans l'exemple de la Figure 2-12, l'activité « Check Account » est une activité de type alternative qui offre deux variantes « Check Credit Card » et « Check Contract ». Ces deux alternatives sont stéréotypées « variant ».

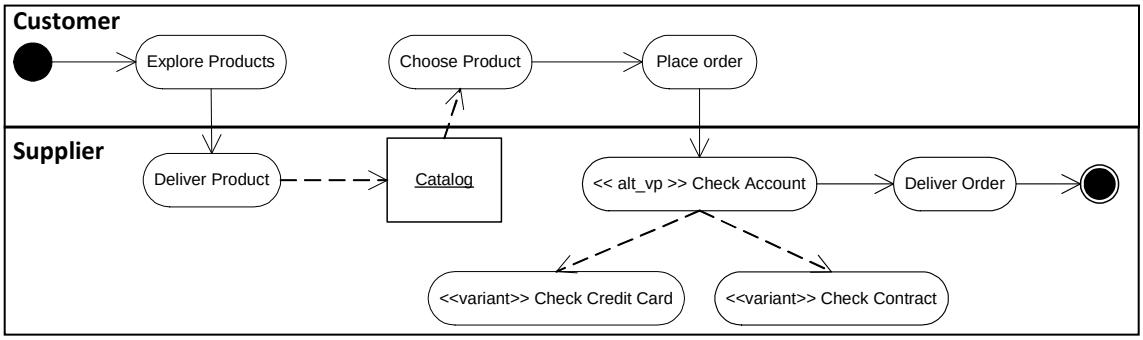


Figure 2-12 : Diagramme d'activités supportant la variabilité [Razavian *et al.*, 2008]

Dans les diagrammes de cas d'utilisation étendus de [Oliviera *et al.*, 2005], la variabilité est exprimée au niveau des cas d'utilisation à travers les stéréotypes « VariationPoint » pour indiquer l'endroit où il y a une variabilité, « mandatory » pour désigner les cas d'utilisation obligatoires, « alternative_OR » pour désigner une variante d'un point de variation de type ensemble d'alternatives, « alternative_XOR » pour désigner une variante d'un point de variation de type alternative. La Figure 2-13 présente un exemple d'expression de la variabilité au niveau du diagramme des cas d'utilisation [Oliviera *et al.*, 2005].

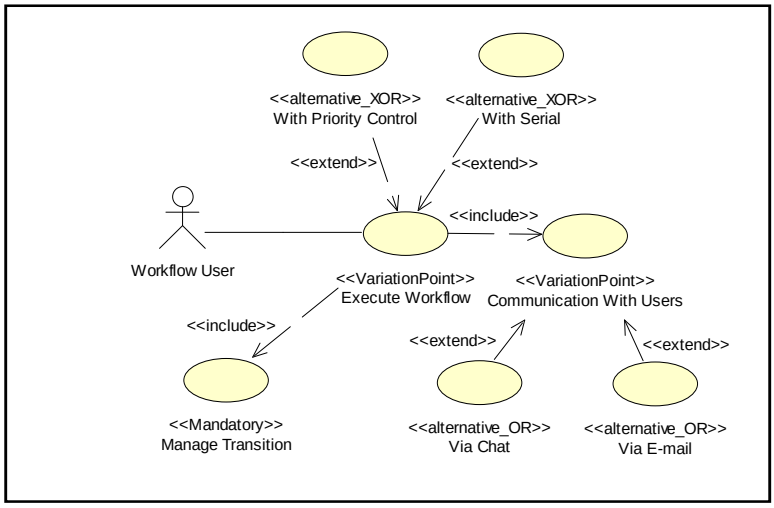


Figure 2-13 : Diagramme de cas d'utilisation supportant la variabilité [Oliviera *et al.*, 2005]

Dans les diagrammes de classes étendus de [Clauss, 2001], la variabilité est exprimée au niveau des classes à travers les stéréotypes « mandatory » pour les classes obligatoires, « alternative » pour les classes alternatives, « optionnel » pour les classes optionnelles. De plus, le stéréotype « XOR » est indiqué sur la relation liant le point de variation et ses variantes alternatives pour contraindre leur choix. Concernant les règles de dépendance entre les classes, la démarche utilise des dépendances d'inclusion et d'exclusion mutuelles matérialisées par les stéréotypes « requires » et « mutex ».

La Figure 2-14 présente un exemple dans le domaine du commerce électronique modélisé sous la forme d'un diagramme de classes supportant la variabilité [Clauss, 2001].

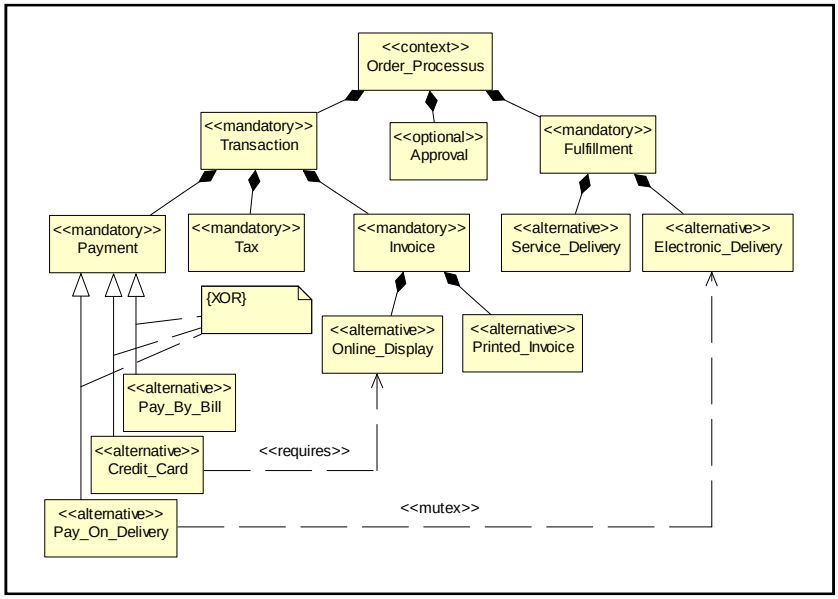


Figure 2-14 : Exemple de diagramme de classes supportant la variabilité [Clauss, 2001]

Dans les diagrammes de séquences étendus de [Ziadi, 2004], la variabilité est exprimée par le stéréotype <<Variation>> représenté sur un sous-diagramme de séquence représentant une variation de comportement. Les variantes sont exprimées à l'aide du stéréotype « Variant ».

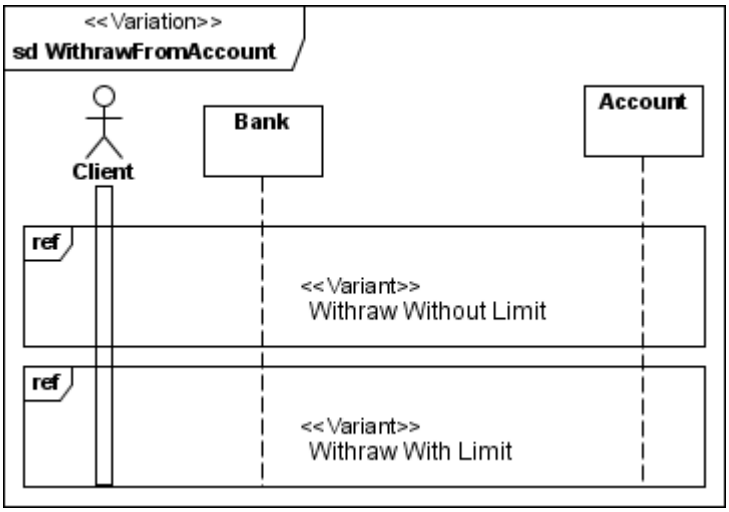


Figure 2-15 : Exemple de diagramme de séquence supportant la variabilité [Ziadi, 2004]

Dans l'exemple illustré dans la Figure 2-15, il existe deux interactions variantes pour le retrait d'argent à partir des comptes : retrait avec vérification du solde et du montant du découvert, et retrait avec seulement vérification du solde. L'interaction WithdrawFromAccount est stéréotypée <<Variation>> et les deux interactions WithdrawWithLimit et WithdrawWithoutLimit sont stéréotypées <<Variant>>.

Pour la réduction de la variabilité, appelée souvent dans le domaine des lignes de produits « dérivation de produits », [Ziadi, 2004] propose une approche pour la dérivation de l'aspect statique basée sur les transformations de modèles. Les modèles statiques des produits (des diagrammes de classes) sont dérivés par transformation à partir du modèle statique de la ligne de produits en utilisant MTL (Model Transformation Language) [INRIA, 2005] comme langage de transformation de modèles.

b) Variabilité séparée des modèles de développement

La recherche dans le domaine des lignes de produits a fait émerger la notion de modèle de variabilité séparé des modèles de développement [Bachman *et al.*, 2003] [Bûhne *et al.*, 2005] [Pohl *et al.*, 2005]. Ces modèles sont souvent appelés des modèles orthogonaux (Orthogonal Variability Model (OVM)).

Un OVM selon l'approche de [Pohl *et al.*, 2005] est un modèle composé de points de variation liés à un ensemble de variantes par des relations définissant le type du point de variation, qu'il soit optionnel, obligatoire ou alternatif avec une cardinalité pour définir le choix permis. L'approche définit aussi des contraintes de dépendance entre variantes et points de variation par les relations « requires » et « excludes ». Un OVM est lié à un modèle de développement (par exemple un diagramme UML) par des relations appelées « artefact dependency ».

La Figure 2-16 illustre un exemple simple d'un OVM selon l'approche de [Pohl *et al.*, 2005]. Le diagramme des cas d'utilisation contient un seul cas d'utilisation « Ouvrir entrée principale » (Open Front Door) de l'acteur « habitant » (Inhabitant). Ce cas d'utilisation inclut les deux cas d'utilisation « ouvrir la porte par un clavier numérique » (Unlock Door by Keybad) et « ouvrir la porte par les empreintes digitales » (Unlock Door by Fingerprint). L'OVM définit un seul point de variation « serrure de porte » (door lock) avec les deux variantes « clavier numérique » (Keypad) et « scanner d'empreintes digitales » (Fingerprint Scanner) reliées au point de

variation par la relation alternative. Chaque variante est liée au cas d'utilisation correspondant par une relation de type «artefact de dépendance » (artefact dependency).

La réduction de la variabilité est réalisée lors de l'ingénierie de l'application (dérivation de produits à partir de la ligne). Ce processus est effectué par une activité de communication en mettant à la disposition du concepteur l'OVM contenant des variantes bien documentées pour faciliter leur choix.

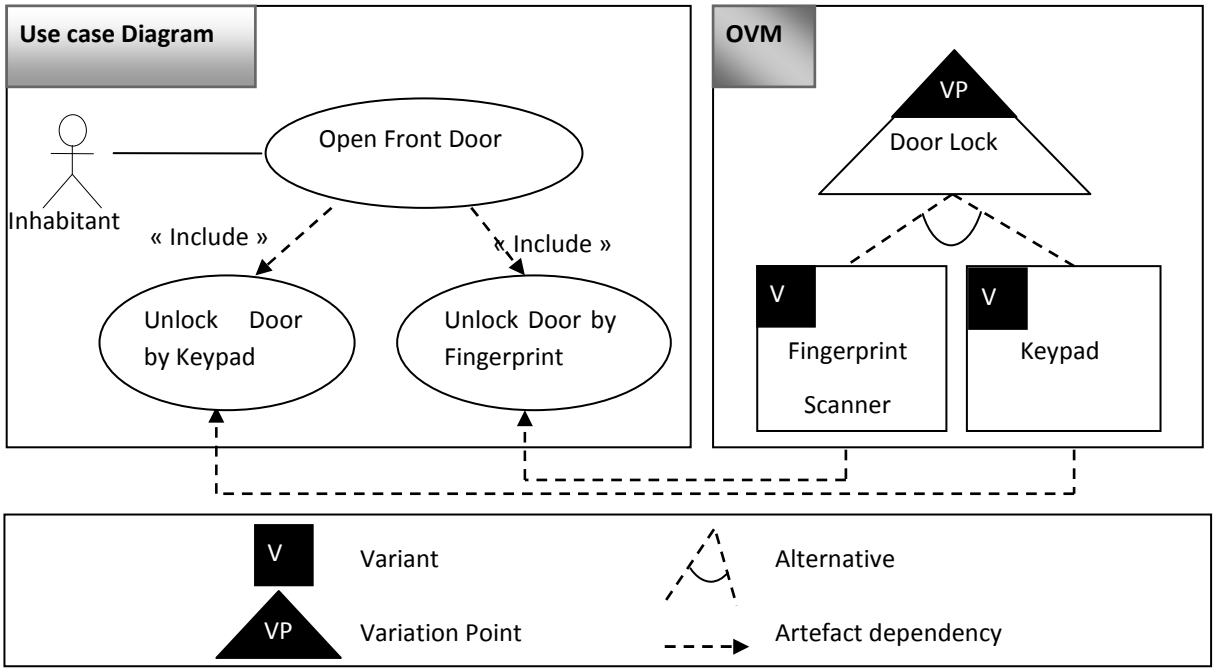


Figure 2-16 : Exemple d'un OVM [Pohl *et al.*, 2005]

3.3.3 Patrons de conception

Certains travaux se sont intéressés à l'expression de la variabilité dans les patrons de conception. Nous soulignons plus particulièrement le système PatternGen de [Sunyé, 1999] et l'approche mini-système de [Arnaud, 2008].

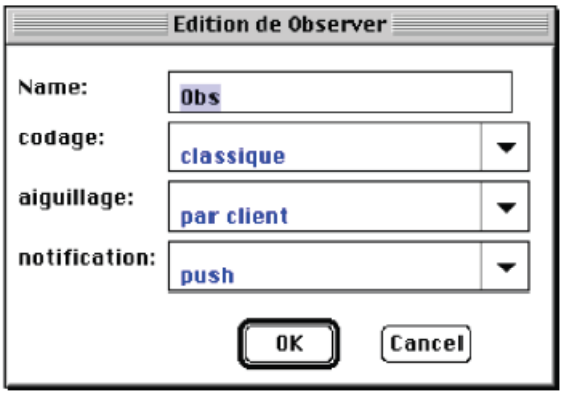


Figure 2-17 : Choix des compromis d'implémentation du patron « Observateur »

PatternGen [Sunyé, 1999] définit un prototype d'outil de génération automatique de code à l'aide de patrons de conception qui prend en compte les variantes d'implémentation. Le principe de cette approche est de fournir au concepteur des moyens de sélectionner le patron

de conception souhaité, de lui permettre de choisir sur quelles classes le patron sera appliqué et de générer automatiquement le code correspondant à cette instance spécifique du patron. La Figure 2-17 présente un exemple de choix de compromis d'implémentation appliqué au patron « Observateur ».

L'approche de [Arnaud, 2008] propose au concepteur de patrons de représenter sa solution semi-formelle comme un mini-système à variantes, dont le pilier central, exprimant la variabilité, est la vue des cas d'utilisation. L'approche de [Arnaud, 2008] propose de concevoir un patron sous forme de trois vues : fonctionnelle, dynamique et statique.

Pour représenter la variabilité sur la vue fonctionnelle, [Arnaud, 2008] définit un opérateur générique pouvant être utilisé pour les quatre types de variabilité présentés précédemment. Cet opérateur est composé d'un point de variation, matérialisé par un cas d'utilisation stéréotypé <<Variation>> et de plusieurs variantes matérialisées par des cas d'utilisation stéréotypés <<Variant>>. À l'aide de valeurs marquées sur ces stéréotypes, les cardinalités minimale et maximale sont exprimées (selon le type de variabilité). La Figure 2-18 illustre la vue fonctionnelle du patron *observateur.*

L'expression de la variabilité sur la vue dynamique est basée sur la notion de frame de référence (interaction use) et de frame de type 'sd' (Sous-Diagramme). Ces mécanismes permettent de modéliser des fragments d'interaction combinés. Par exemple, pour représenter une variation de type « Alternative », un frame de type référence stéréotypé « Variation » est utilisé pour référencer le sous-diagramme de séquence comportant les variantes alternatives. Les variantes sont modélisées par des frames de type 'sd' stéréotypés « Variant ».

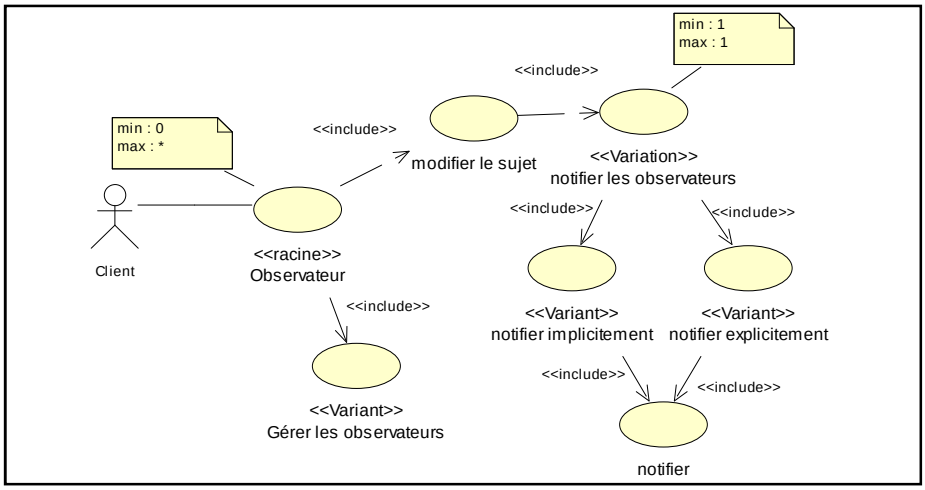


Figure 2-18 : Exemple de diagramme de cas d'utilisation supportant la variabilité

Concernant la vue statique, l'approche suggère de donner, pour chaque cas d'utilisation spécifié, ses apports statiques au modèle de la solution. La structure est disséminée dans plusieurs fragments qui s'assembleront pour former une vue statique classique lors du processus de réduction. Toute classe « impactée » par une variation ou plus généralement par une fonctionnalité sera représentée dans le fragment statique de ce cas d'utilisation avec ses propriétés apportées.

Quant au processus de réduction de la variabilité selon [Arnaud, 2008], il est constitué de deux activités : a) le choix du patron à imiter qui consiste à sélectionner un patron dans un système de patrons. La solution du patron sélectionné est appelée un modèle imitable et consiste en un mini-système à variantes composé de trois vues, b) la réduction qui permet au concepteur de choisir les variantes qu'il désire imiter à partir de la vue des cas d'utilisation. Les vues dynamiques et statiques sont alors automatiquement déduites. Un modèle imité dans l'état adaptable est ainsi obtenu.

3.4 Évaluation des approches

Le Tableau 2-6 et le Tableau 2-7 comparent ces trois types d'approches selon le cadre de référence proposé.

	Méthode	Contexte	Finalité	Complétude
Ingénierie de domaine	FODA **[Kang *et al.*, 1990]** / FORM **[Kang *et al.*, 1998]**	Développement des logiciels par réutilisation	Réutilisation	Structurelle
	4VM **[Bashroush *et al.*, 2008]**	Développement des logiciels par réutilisation	Réutilisation	Métier/ hiérarchique & comportementale/ interaction/ technique
Lignes de produits	- Extension du diagramme d'activités **[Razavian *et al.*, 2008]** - Extension du diagramme de cas d'utilisation **[Oliviera *et al.*, 2005]** - Extension du diagramme de classes **[Clauss, 2001], [Ziadi, 2004]** - Extension du diagramme de séquence **[Ziadi, 2004]**	Représentation des propriétés communes et discriminantes des applications d'une même famille	Personnalisation	Métier **[Razavian *et al.*, 2008]** Fonctionnelle **[Oliviera *et al.*, 2005]**/ Dynamique **[Ziadi, 2004]**/ Statique **[Clauss, 2001] [Ziadi, 2004]**
	OVM **[Pohl *et al.*, 2005]**		Réutilisation / personnalisation	Structurelle
Patrons de conception	PatternGen **[Sunyé, 1999]**	Génération de diverses variantes	Personnalisation orientée utilisateur final	Technique
	Mini-système **[Arnaud, 2008]**	Solutions complètes de patrons à variantes	Réutilisation	Fonctionnelle/ Dynamique/Structurelle

Tableau 2-6 : Comparaison des approches : contexte, finalité et complétude

	Identification de la variabilité	Représentation de la variabilité		Réduction de la variabilité
Analyse de domaine	Identifier pour chaque système logiciel étudié, des caractéristiques ("features") en terme de connaissances optionnelles, obligatoires ou alternatives	Partie fixe/variable	Oui	Réduction lors du processus d'application par un modèle de décision
		Type variation	Option/ alternative	
		Contraintes de variabilité	Exclusion/ inclusion	
		Notation	Graphique : - Diagramme de caractéristiques - Diagramme de caractéristiques à quatre vues **[Bashroush *et al.*, 2008]**	
Lignes de produits	- Rassembler toutes les connaissances relatives à une famille de produits - Discriminer les différentes variantes possibles d'une même connaissance pour qu'elle soit commune à plusieurs produits ou spécifique à un produit	Partie fixe/variable	Oui	- Réduction au moment de la conception par transformation de modèles avec MTL **[Ziadi, 2004]** - Communication de l'OVM au concepteur pour choisir les variantes **[Pohl *et al.*, 2005]**
		Type variation	Option/ alternative optionnelle / alternative/ ensemble d'alternatives	
		Contraintes de variabilité	Exclusion/mutex, Inclusion/requires	
		Notation	Graphique : - Diagrammes UML étendus **[Razavian *et al.*, 2008] [Oliviera *et al.*, 2005] [Clauss, 2001] [Ziadi, 2004]** - OVM **[Pohl *et al.*, 2005]**	
Patrons de conception	Définir des compromis d'implémentation spécifiques à chaque implémentation d'un patron **[Sunyé, 1999]**	Partie fixe/variable	Oui	Sélection du patron, choix des classes sur lesquelles il sera appliqué et génération automatique du code correspondant à cette instance
		Type variation	Non	
		Contraintes de variabilité	Non	
		Notation	Textuelle : Boîte de dialogue	
	Définir des variantes pour la solution d'un patron avec comme point d'entrée la vue fonctionnelle du patron **[Arnaud, 2008]**	Partie fixe/variable	Oui	Choix des variantes à imiter à partir de la vue des cas d'utilisation. Les autres vues sont automatiquement déduites
		Type variation	Option / alternative	
		Contraintes de variabilité	Non	
		Notation	Graphique : Diagrammes UML étendus	

Tableau 2-7 : Comparaison des approches : identification, représentation et réduction de la variabilité

3.5 Synthèse

Dans le Tableau 2-8, nous évaluons les différents types d'approches en soulignant leurs avantages et inconvénients.

Approche	Avantages	Inconvénients
FODA **[Kang *et al.*, 1990]** FORM **[Kang *et al.*, 1998]** 4VM **[Bashroush *et al.*, 2008]**	- Bonne hiérarchisation des connaissances de domaine - Solutions à différentes vues de développement	- Techniques non supportées par les outils standards de développement
- Diagramme d'activités **[Razavian *et al.*, 2008]** - Diagramme de cas d'utilisation **[Oliviera *et al.*, 2005]** **-** Diagramme de classes **[Clauss, 2001], [Ziadi, 2004]** **-** Diagramme de séquence **[Ziadi , 2004]**	- Expression de la variabilité sur des diagrammes UML	- Absence d'une solution complète intégrant différentes vues du développement
- OVM **[Pohl *et al.*, 2005]**	- Maintien de la traçabilité de la variabilité	- Technique non supportée par les différents outils standards de développement
PatternGen **[Sunyé, 1999]**	Définition des compromis d'implémentation spécifiques à chaque implémentation d'un patron	- Approche fortement liée aux concepts de patrons - Approche liée à la phase d'implémentation
Mini-système **[Arnaud, 2008]**	- Expression de la variabilité sur des diagrammes UML - Solutions complètes à différentes vues de développement	- Approche évaluée seulement dans le contexte des patrons - Absence de la gestion des contraintes de variabilité

Tableau 2-8 : Évaluation des approches de modélisation de la variabilité

Parmi les concepts de base de la variabilité, nous avons introduit des nouveaux concepts tels que la *complétude* de la variabilité. En effet, la variabilité était souvent représentée sur la vue statique d'un artefact de développement malgré certaines tentatives de sa représentation sur différentes vues, mais souvent sans tenir compte de la cohérence entre les vues. Nous avons aussi distingué le concept de *portée* de la variabilité pour lequel nous avons remarqué l'absence de la distinction entre une variabilité orientée exécution et une variabilité orientée conception, c'est pour cela nous avons classifié la variabilité en deux types : variabilité pour la personnalisation (orientée exécution) et variabilité pour la réutilisation (orientée conception).

4 Positionnement de notre approche

L'étude des différentes démarches de conception de composants métier fait apparaître plusieurs tendances. Notre approche s'inscrit dans la tendance où l'on perçoit un CM comme une unité de réutilisation de connaissances de domaine. En effet, nous considérons qu'un CM est une représentation d'un concept actif dans un domaine. Ainsi, les CM sont utilisés pour définir des concepts de type « Entité » (par exemple, bibliothèque, compte, abonné…), ou pour définir des « Processus » pour les concepts qu'ils représentent (par exemple, processus d'emprunt, processus de réservation d'un ouvrage, processus d'inscription d'un abonné…).

Dans ce contexte, nous nous penchons plus particulièrement vers la réutilisation de CM de type **processus** qui s'avère plus pertinente que la simple réutilisation d'un CM de type entité. En effet, un processus est un bon candidat pour la réutilisation, tandis qu'une entité ne l'est pas, car la réutilisation des entités réduit l'information réutilisable quand elle n'est pas intégrée dans un processus ou un fragment de processus. Ainsi, notre travail est basé sur la réutilisation de CM processus qui utilisent des CM entité pour leur exécution.

De plus, nous nous focalisons aussi sur une modélisation **conceptuelle** des CM. En effet, nous considérons que seuls les modèles conceptuels des CM produits lors des étapes de spécification, constituent de véritables productions capitalisables et réutilisables dans le cycle de conception d'un SI à base de composants ; ils sont technologiquement neutres et n'évoluent qu'en fonction de l'évolution des besoins auxquels ils répondent.

Dans ce sens, et pour modéliser des CM, nous nous basons sur des **modèles UML,** d'une part car UML propose un ensemble de diagrammes standards supportés par la plupart des outils qui existent, d'autre part car son extensibilité rend possible la modélisation des concepts liés aux CM.

Du point de vue modèle de composants, nous nous intéressons à une spécification d'une solution conceptuelle **complète**. La modélisation d'un CM a été souvent limitée à l'aspect statique du composant, alors qu'une approche à vues multiples permet d'exprimer plus complètement et avec une meilleure qualité de conception, la solution offerte par ce composant. Dans ce travail, nous proposons donc de spécifier quatre vues pour le composant : **vue métier**, **vue fonctionnelle**, **vue dynamique** et **vue structurelle**.

La vue métier est représentée par un diagramme d'activités, elle représente l'aspect organisationnel du CM. La vue fonctionnelle du CM est modélisée par un diagramme de cas d'utilisation, elle présente les fonctionnalités du CM en construction, les dépendances qui relient ces fonctionnalités et les acteurs qui les déclenchent. La vue dynamique du CM est modélisée par un diagramme de séquence modélisant ainsi les différentes interactions entre les classes du CM ainsi qu'une description détaillée de ses fonctionnalités. Pour la vue structurelle, nous adoptons le modèle conceptuel Symphony [Hassine, 2005]. Le choix de ce modèle est motivé par sa nature conceptuelle ainsi que sa formalisation pertinente à travers une définition tripartie d'un composant (partie structurelle, partie interface et partie collaborative). Les cases grisées dans le Tableau 2-9 soulignent les critères des CM pris en compte dans notre approche.

Critère	Attributs			
Visibilité	Noire	Blanche	Verre	
Granularité	Faible	Moyenne	Forte	
Unité de composition	Classe	Objet		
Variabilité	Conceptuelle	Technique		
Type de connaissance	Entité	Processus	Utilitaire	
Nature de solution	Conceptuelle	Logicielle		
Complétude	Métier	Fonctionnelle	Dynamique	Structurelle
Couverture	Générique	Domaine	Entreprise	
Portée	Analyse	Conception	Implantation	

Tableau 2-9 : Critères des CM considérés dans notre approche

Concernant l'axe **variabilité**, le Tableau 2-10 résume les concepts de variabilité pris en considération dans notre approche. Parmi les approches étudiées, nous distinguons plus particulièrement l'approche de [Arnaud, 2008], plus représentative de nos objectifs, et nous basons sur ses techniques d'expression de solutions complètes à variantes. Par contre, cette approche ayant été évaluée seulement pour les patrons de conception, des extensions liées à la conception des CM sont nécessaires. En outre, une démarche basée sur les principes d'analyse de domaine est indispensable pour identifier la variabilité.

Axes du cadre de référence	Attributs	
Contexte de l'approche	Variabilité conceptuelle pour la réutilisation des CM	
Finalité de l'approche	Réutilisation	
Complétude	Métier/Fonctionnelle/Dynamique /Structurelle	
Identification de la variabilité	Étude des points communs et discriminants entre différents SI de même nature	
Représentation de la variabilité	Partie fixe/variable	Oui
	Type de variation	Option/ alternative / alternative optionnelle / ensemble d'alternatives
	Contraintes de variabilité	Exclusion/ inclusion
	Notation	Graphique / textuelle
Réduction de la variabilité	Processus intégré dans le processus de conception par la réutilisation	

Tableau 2-10 : Concepts de la variabilité pris en compte dans notre approche

Par ailleurs, l'étude des différentes méthodes de développement basées composants a montré que beaucoup de travaux ont été consacrés à la formalisation des composants, mais que peu d'entre eux ont apporté de véritables solutions concernant la prise en charge du double processus « pour et par la réutilisation ». Par exemple, dans l'actuelle version de la démarche Symphony [Hassine, 2005], la prise en compte de la réutilisation n'a pas été achevée : il s'agit maintenant d'identifier, en plus des structures modulaires métier (appelées objets métier), des structures modulaires métier réutilisables (appelées composants métier).

L'objectif de nos travaux s'inscrit dans ce sens. Pour cela, nous proposons une **extension** de la méthode **Symphony**, afin de la doter des deux processus pour et par la réutilisation adaptés à son processus de développement de SI à base de composants tout en tenant compte de l'intégration de la variabilité dans ces processus.

Chapitre 3 : Concepts de base et vue métier d'un composant métier processus

Dans le développement de systèmes d'information à base de composants, il est souvent difficile d'expliciter des critères clairs de réutilisation, en particulier la manière dont on peut identifier et spécifier des composants candidats à la réutilisation. Autrement dit, une question importante concerne le fait de savoir comment les connaissances d'un SI ou d'un domaine peuvent être identifiées pour être ensuite formalisées, structurées et mises à la disposition des concepteurs de SI à base de composants.

Ce chapitre présente tout d'abord les concepts de base liés à la notion de CM (§1) ainsi que les besoins qui ont guidé vers le modèle de CM proposé dans cette thèse (§2). Ensuite, nos propositions en termes de vue métier du modèle de CM sont présentées (§3).

1 Vers un choix de concepts

1.1 Système d'information

Un système d'information est la partie du réel constituée d'informations organisées, d'événements ayant un effet sur ces informations, et d'acteurs qui agissent sur ces informations ou à partir de ces informations, selon des processus visant une finalité de gestion et utilisant les technologies de l'information [Morley *et al.*, 2000].

Décrire le SI de l'entreprise revient à représenter son fonctionnement et son organisation sous l'angle des informations que l'on a choisies de gérer et sur lesquelles on s'appuie. Cependant, quand le SI est vaste, avec un nombre élevé de types d'informations et un nombre élevé d'acteurs ayant des rôles différents, il est nécessaire de le concevoir de façon modulaire pour mieux le comprendre, pour maîtriser son développement et pour permettre des évolutions partielles sans impact sur l'ensemble.

Concevoir un SI de façon modulaire consiste donc à le décomposer en sous-SI quasi-autonomes et d'une taille raisonnable. Par exemple, le SI d'une institution d'enseignement comprend les sous-SI suivants : scolarité (programmes et planning), élèves (inscription, notes…), personnel (carrière et rémunération). Un sous-SI représente donc un métier, c'est-à-dire une mission, un savoir-faire, des compétences. C'est une partie de l'activité de l'entreprise, suffisamment cohérente du point de vue informationnel pour pouvoir être étudiée et représentée par des modèles. Cette notion de sous-SI correspond à la notion de domaine dans [Morley *et al.*, 2000].

Ainsi, nous pouvons considérer un SI comme un ensemble de sous-SI centrés sur un Processus Métier (PM) [Morley *et al.*, 2004] [Dumas *et al.*, 2008]. Un PM est un plan d'ensemble indiquant comment les acteurs collaborent au moyen des informations gérées pour traiter une catégorie d'événements. Il s'agit d'une structure modulaire spécifique à un système d'information donné. C'est un processus au cœur du système, représentatif d'une partie du métier de l'entreprise et qui constitue une solution particulière à une classe de problèmes définie pour un domaine métier donné.

1.2 Domaine

Un domaine peut être défini par un ensemble de problèmes ou de fonctions que les applications de ce domaine peuvent résoudre. Il peut donc être associé à un secteur métier, une collection de problèmes, une collection d'applications ou encore un ensemble de connaissances avec un vocabulaire commun [SEI, 2007].

Pour déterminer un domaine, il est nécessaire de définir sa frontière. Or, cette frontière n'est souvent pas facile à délimiter. Ceci peut être dû à l'inexpérience de l'analyste de domaine ou à la nature de certains domaines qui utilisent ou fournissent des services à d'autres domaines.

Une manière de considérer un domaine est de le voir comme un ensemble d'applications. Ce point de vue se concentre sur des familles d'applications. Le domaine est borné selon la similarité de ces applications [Harsu, 2002].

Dans nos travaux, nous nous intéressons à la réutilisation de concepts récurrents dans plusieurs systèmes d'information. Motivés par cet objectif, nous considérons un domaine

comme une classe de problèmes pour laquelle il existe plusieurs solutions. Un système d'information est vu comme une solution particulière à la classe de problèmes.

Un modèle de domaine, défini comme une classe de problèmes, est fortement réutilisable, du fait que des concepts orientés problème sont plutôt stables. Ils ne varient pas nécessairement, bien que les besoins des systèmes changent. Par exemple, le domaine de réservation de vols peut être représenté par plusieurs systèmes similaires modélisés selon les règles métier de chaque compagnie aérienne.

En outre, les domaines peuvent être divisés en *domaines métier (DM)* et *domaines fonctionnels (DF)*. Dans les domaines métier, les systèmes d'information sont classifiés selon le secteur d'activité. De tels systèmes sont, par exemple, des systèmes de réservation de vols, des systèmes médicaux, des systèmes de gestion des stocks, etc. Dans les domaines fonctionnels, les systèmes d'information sont classifiés selon leurs fonctionnalités. Des exemples sont des systèmes d'allocation de ressources, des systèmes d'enregistrement, des systèmes de surveillance, etc.

1.3 Identification des composants réutilisables

Des techniques basées sur l'analyse de domaine, la recherche de similarité, d'analogie et de généricité peuvent être utilisées pour identifier systématiquement des composants candidats à la réutilisation. Le résultat de cette étape est en général un ensemble de modèles d'analyse, de diagrammes, par exemple en UML, décrivant les besoins en termes de fonctionnalités d'un type de composants.

Neighbors [Neighbors, 1984] a été l'un des premiers à définir l'activité d'identification des objets et des opérations d'une classe de systèmes similaires qui offrent une solution concernant un problème particulier de domaine. Dans ce sens, l'approche de la réutilisation par analogie [Maiden *et al.*, 1992] a été proposée. Elle supporte le processus d'analyse de domaine en fournissant des produits réutilisables dans un domaine différent mais analogue. En effet, la résolution de problèmes par analogie est un processus de transfert de connaissances à partir des expériences précédentes de résolution de problèmes vers un nouveau problème qui partage des similitudes avec le problème existant. Les connaissances transférées sont ensuite structurées et encapsulées dans des unités appelées composants réutilisables et utilisées pour construire de nouvelles solutions pour de nouveaux problèmes.

En outre, selon la classification de domaines en domaines métier et domaines fonctionnels, il nous paraît intuitif de parler de Composants Métier qui modélisent des connaissances réutilisables d'un domaine métier et de Composants Fonctionnels modélisant des connaissances réutilisables d'un domaine fonctionnel.

1.4 Vers un composant fonctionnel

Il est généralement admis qu'il existe des similitudes entre les différents domaines métier. Par exemple, Finkelstein [Finkelstein, 1988] a présenté des similitudes entre le système de surveillance de patient et le système d'alarme antivol. Maiden et Sutcliffe [Maiden *et al.*, 1992] ont également montré que le système de contrôle de trafic aérien et celui de fabrication partagent des caractéristiques en commun. Généralement, les structures de connaissances similaires constituent les clés d'une analogie entre deux domaines métier différents.

Ainsi, pour structurer des connaissances réutilisables et transversales à plusieurs domaines métier, nous utilisons le concept de Composant Fonctionnel (CF). Nous définissons un CF comme un composant horizontal, qui représente des éléments récurrents appartenant à différents domaines, et fournit alors des services génériques utiles à plus d'un domaine. Un CF peut représenter des concepts communs à plusieurs domaines métier (ex. ressource, client, …), de façon standard, ou des processus utilisant ces concepts (ex. gestion d'allocation de ressources, enregistrement, …). Le Tableau 3-1 représente des exemples de composants fonctionnels qui peuvent être adaptés dans des domaines métier. Ces exemples sont extraits de [Lung *et al.*, 1995] où ils sont perçus comme des abstractions de domaines.

Composant fonctionnel	Description	Exemples de domaines métier
Contrôle de valeur	Modélise des domaines où l'on réapprovisionne un objet quand un niveau minimum est atteint. Une valeur prédéfinie est ordonnée ou ajoutée ou bien une alerte est signalée.	Contrôle de stock, contrôle de température.
Enregistrement	Modélise des domaines où l'on garde la trace de l'enregistrement des objets qui quittent et entrent dans un environnement.	Gestion du personnel, administration des étudiants.
Ordonnancement	Modélise des domaines où l'on effectue un ordonnancement d'objets dans le but de réaliser un maximum de productivité ou d'efficacité en limitant les ressources.	Systèmes de fabrication, systèmes d'ordonnancement de véhicules (ex. avion, bus, camions).
Position	Modélise des domaines où l'on surveille le mouvement d'objets dans un espace dans le but d'en assurer la position correcte.	Contrôle de trafic aérien, systèmes de fabrication.
Allocation	Modélise des domaines où l'on alloue un objet à un autre objet. Les objets alloués sont rendus après une période d'allocation.	Gestion d'une bibliothèque, location de voitures, gestion d'hôtels.
Distribution	Modélise des domaines où l'on collecte un ensemble d'objets dans un endroit central avant de les distribuer vers des destinations prédéfinies.	Système de distribution de colis, distribution de marchandise.
Coordination	Modélise des domaines où l'on surveille une classe d'objets, synchronise et coordonne d'autres objets pour maximiser l'efficacité ou la sécurité de cette classe.	Gestion de trafic, signaux de circulation.

Combinaison	Modélise des domaines où l'on collecte des objets individuels avant de les combiner pour leur mise à jour, analyse ou contrôle.	Gestion de trafic, gestion d'un terminal de point de vente.
Vérification périodique	Modélise des domaines où l'on vérifie périodiquement l'état d'un objet.	Système bancaire, maintenance de véhicules, gestion de stock.

Tableau 3-1 : Exemples de composants fonctionnels

1.5 Processus métier Vs. Composant métier Vs. Composant fonctionnel

Le méta-modèle de la Figure 3-1 met en évidence le concept de processus métier, concept central d'un système d'information, les concepts de composant métier et composant fonctionnel, concepts centraux d'un système de réutilisation et les concepts de domaine métier et domaine fonctionnel, concepts centraux d'un domaine [Saidi *et al.*, 2008b].

- Un composant fonctionnel, qui relève d'un domaine fonctionnel, peut être concrétisé par un ensemble de CM issus de différents DM ou par un ensemble de PM issus de différents systèmes d'information. Par exemple, les CM « Réservation de lit d'hôpital » et « Réservation d'une chambre d'hôtel » issus des domaines métier « Gestion d'hôpital » et « Gestion d'hôtel », concrétisent un CF de « Réservation d'un lieu de séjour» qui relève du domaine fonctionnel « Réservation ».
- Un CM, qui s'applique à un domaine métier, peut être une concrétisation d'un ou plusieurs CF ou concrétisé par un ensemble de PM. Par exemple, deux PM de gestion d'hôpital, dans 2 hôpitaux différents, peuvent concrétiser un CM « gestion d'hôpital » qui prend en charge les différentes règles de gestion de chaque hôpital et qui est générique et réutilisable dans les deux systèmes d'information.
- Un PM représente un SI ou un sous-SI, il peut être construit à partir de rien ou par concrétisation de CM ou de CF. Par exemple, un CM de « Réservation de chambre d'hôtel » peut être concrétisé par des PM différents constituant différents systèmes d'information du même domaine de l'hôtellerie. La concrétisation est alors réalisée selon les règles de gestion du SI cible.
- Chacun des concepts SI, domaine métier et domaine fonctionnel est respectivement décomposé en sous-SI, sous-domaine métier et sous-domaine fonctionnel, matérialisant ainsi une organisation par hiérarchisation de ces concepts. Ces décompositions peuvent être cohérentes l'une avec l'autre pour assurer une correspondance entre tous ces concepts.

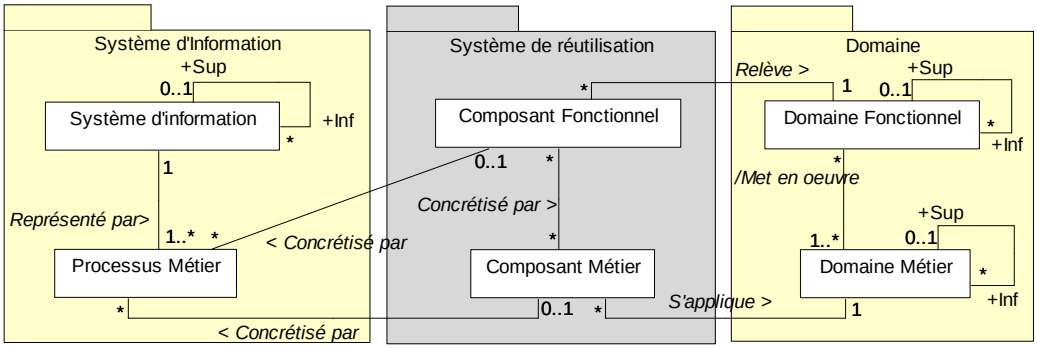


Figure 3-1 : Méta-modèle décrivant la relation entre PM, CM et CF

1.6 Vers un CMP réutilisable

À partir de la définition des différents concepts liés au paradigme CM, nous avons soulevé la nécessité de relier la définition d'un CM à la façon dont un SI est organisé. Le fait qu'un SI soit constitué par un ensemble de PM matérialisant la finalité globale de ce SI nous amène à bien situer le besoin d'un SI en terme de réutilisation. De ce fait, un CM ne peut être candidat à une éventuelle réutilisation que s'il répond aux besoins d'un SI.

Dans ce sens, nous proposons de spécifier des CM nécessaires à la construction de SI à partir de l'abstraction des processus métier menés par différents intervenants dans différents SI. Deux types principaux de CM sont distingués : les CM de type entité (CME), qui permettent la manipulation d'abstractions d'entités du monde réel, et les CM de type processus (CMP), destinés à soutenir directement les processus métier spécifiques aux différents intervenants, et qui mettent à disposition pour cela les fonctionnalités adéquates.

Le point d'intérêt est mis plus particulièrement sur la réutilisation de CMP qui s'avère plus pertinente qu'une simple réutilisation de CME. En effet, un CM ne doit pas représenter de simples concepts métier, mais des concepts qui sont relativement autonomes dans l'espace du problème. Par exemple, l'allocation de ressources est un bon candidat, tandis qu'une ressource ne l'est pas, car l'information y est trop réduite. Ainsi, notre travail est basé sur la réutilisation de CMP qui utilisent des CME pour leur exécution.

D'un autre côté, nous considérons un système de réutilisation comme un espace structuré sous forme d'une hiérarchie de CM (CMP et CME). D'une part, un CMP (ex. Réservation d'hôtel) qui s'applique à un domaine métier (ex. Hôtellerie) peut être créé par réutilisation d'un autre CMP (ex. Réservation de lieu de séjour) plus abstrait et qui relève d'un domaine fonctionnel (ex. Réservation). D'autre part, un CME (ex. Chambre) peut être créé par réutilisation d'un autre CME plus abstrait (ex. Lieu de séjour). En outre, et d'une manière générale, un CF n'étant qu'un CM avec un niveau d'abstraction plus élevé, nous préférons fusionner les deux notions CM et CF par le même concept « CM ».

Nous obtenons ainsi plusieurs types de hiérarchisation : 1) au niveau des SI qui peuvent être décomposés en des PM, 2) au niveau du système de réutilisation où les CM peuvent être une réutilisation d'autres CM plus abstraits, et 3) au niveau des domaines métier et fonctionnels hiérarchisés en sous-domaines par étude d'analogie et de similarité de domaines. Ces types de décomposition sont corrélés et favorisent la réutilisation.

Le méta-modèle de la Figure 3-2 illustre l'ensemble des concepts discutés dans ce paragraphe en utilisant un exemple illustratif présenté dans la Figure 3-3.

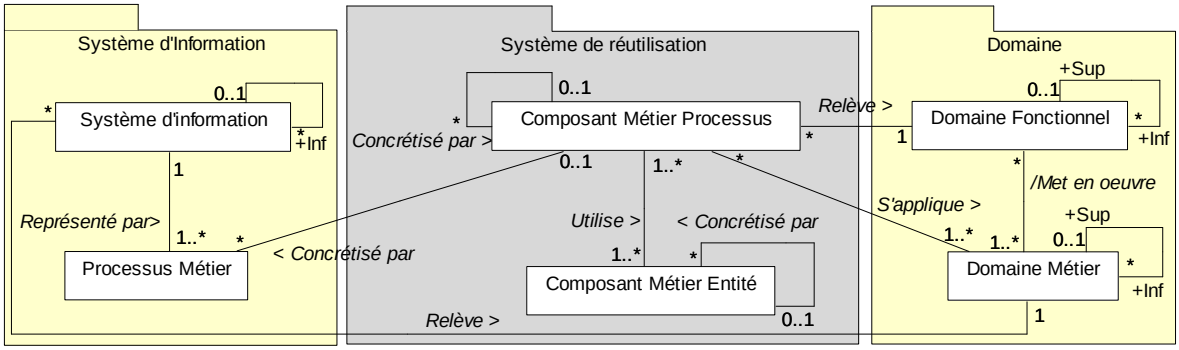


Figure 3-2 : Hiérarchisation de domaines, CM et de SI

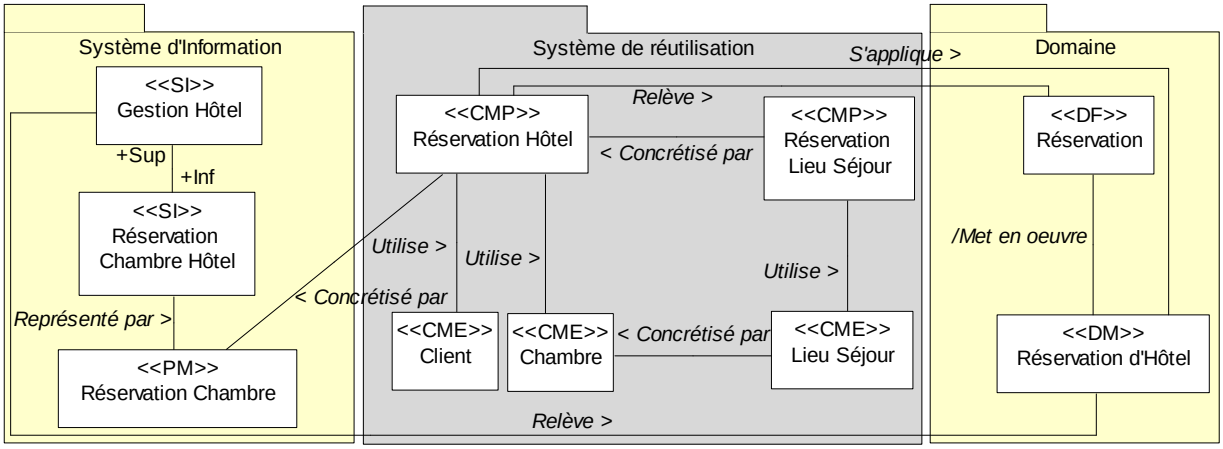


Figure 3-3 : Exemple de hiérarchisation de DM, DF, CM et de SI

La description de la solution d'un CMP doit fournir toutes les informations nécessaires pour comprendre sa fonctionnalité indépendamment de son niveau d'abstraction. L'ensemble des concepts concernant le modèle du CMP proposé est présenté dans la section suivante.

2 Vers un modèle de CMP multi-vues supportant la variabilité

2.1 Un CMP multi-vues

La modélisation d'un CM a souvent été limitée à l'aspect structurel du composant, en particulier dans les modèles de données génériques de [Mineau *et al.*, 1995], les composants génériques de [Castano *et al.*, 1994], le modèle de domaine de [Snoeck *et al.*, 2000], le modèle Symphony de [Hassine, 2005] et le Business Component de [Herzum *et al.*, 2000]. Cependant, une approche à vues multiples pourrait exprimer plus complètement la solution offerte par un composant. En effet, la réutilisation de telles solutions apporterait une meilleure qualité de conception, en ayant une spécification plus complète matérialisée par plusieurs vues de développement. Dans ce travail, nous proposons de spécifier quatre vues pour un composant : 1) une vue métier représentant l'aspect organisationnel du CMP, 2) une vue fonctionnelle, 3) une vue dynamique et 4) une vue statique. Les trois dernières vues représentent les parties

informatisées du CMP. Pour illustrer notre modèle de CMP multi-vues, nous utilisons l'exemple du CMP « Allocation de Ressource ».

a. *Vue métier*

La vue métier d'un CMP modélise l'ensemble des tâches à accomplir et les différents acteurs impliqués dans la réalisation d'un PM lié à un SI donné. Cette vue illustre les interactions sous forme d'échange d'informations entre divers acteurs et fournit en outre à chacun des acteurs les informations nécessaires pour la réalisation de sa tâche.

Pour modéliser cette vue, nous utilisons le diagramme d'activités d'UML 2. Un diagramme d'activités permet de modéliser le comportement du CMP, à travers une séquence d'activités schématisant le déroulement du processus. La distinction entre les activités informatisées et celles manuelles est réalisée par l'utilisation des deux stéréotypes « Informatisée » et « Manuelle ». À titre d'exemple, la Figure 3-4 illustre la vue métier du CMP « Allocation de Ressource » modélisée par un diagramme d'activités.

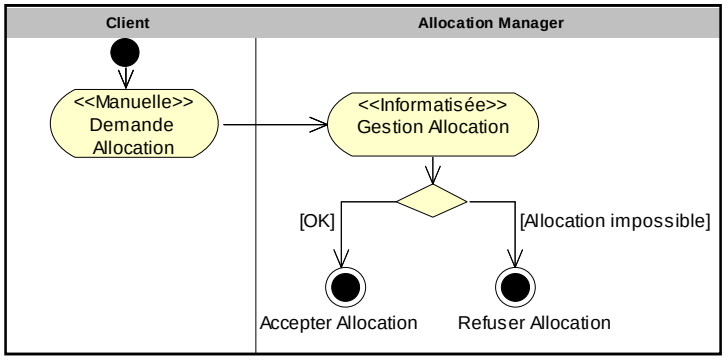


Figure 3-4 : Vue métier du CMP « Allocation de Ressource »

b. *Vue fonctionnelle*

La vue fonctionnelle représente les fonctionnalités informatisées du CMP en construction, les dépendances qui les relient et les acteurs qui les déclenchent. Nous modélisons cette vue par un diagramme de cas d'utilisation UML 2.

Un diagramme de cas d'utilisation représente les cas d'utilisation identifiés et le ou les acteurs associés à chacun. Les cas d'utilisation constituent une technique de description du système étudié privilégiant le point de vue de l'utilisateur. Il s'agit d'une façon spécifique d'utiliser le système. La Figure 3-5 illustre la vue fonctionnelle du CMP « Allocation de Ressource » modélisée par un diagramme de cas d'utilisation. Notons que dans cet exemple, la seule activité de l'acteur Client (Demande Allocation) étant manuelle, cet acteur n'est par conséquent pas conservé dans la vue fonctionnelle.

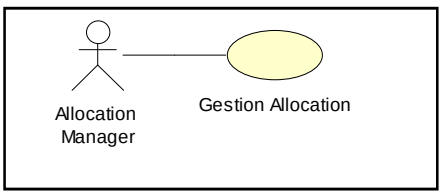


Figure 3-5 : Vue fonctionnelle du CMP « Allocation de Ressource »

La description des cas d'utilisation est libre. Cependant, cette description prend souvent une forme rédigée qui convient à la communication avec les utilisateurs. Des règles de structuration doivent être appliquées pour en faciliter l'expression, la compréhension et la cohérence. La Figure 3-6 illustre une façon de décrire le cas d'utilisation « Gestion Allocation ».

Cas d'utilisation : Gestion Allocation
Acteurs : Allocation Manager.
Résumé : ce cas traite la gestion d'allocation de ressources par un manager d'allocation.
Événement déclencheur : un client demande une allocation de ressource.
Pré-conditions : 1. Client déjà enregistré. 2. Ressource déjà enregistrée.
Description détaillée : Un client se présente au lieu de l'allocation de ressources. Le manager : 1. Identifie le client. 2. Identifie la ressource. 3. Vérifie l'autorisation du client à louer des ressources. 4. Vérifie la disponibilité de la ressource. 5. Crée une allocation.
Post-condition : Allocation créée.

Figure 3-6 : Description du cas d'utilisation « Gestion Allocation »

c. *Vue dynamique*

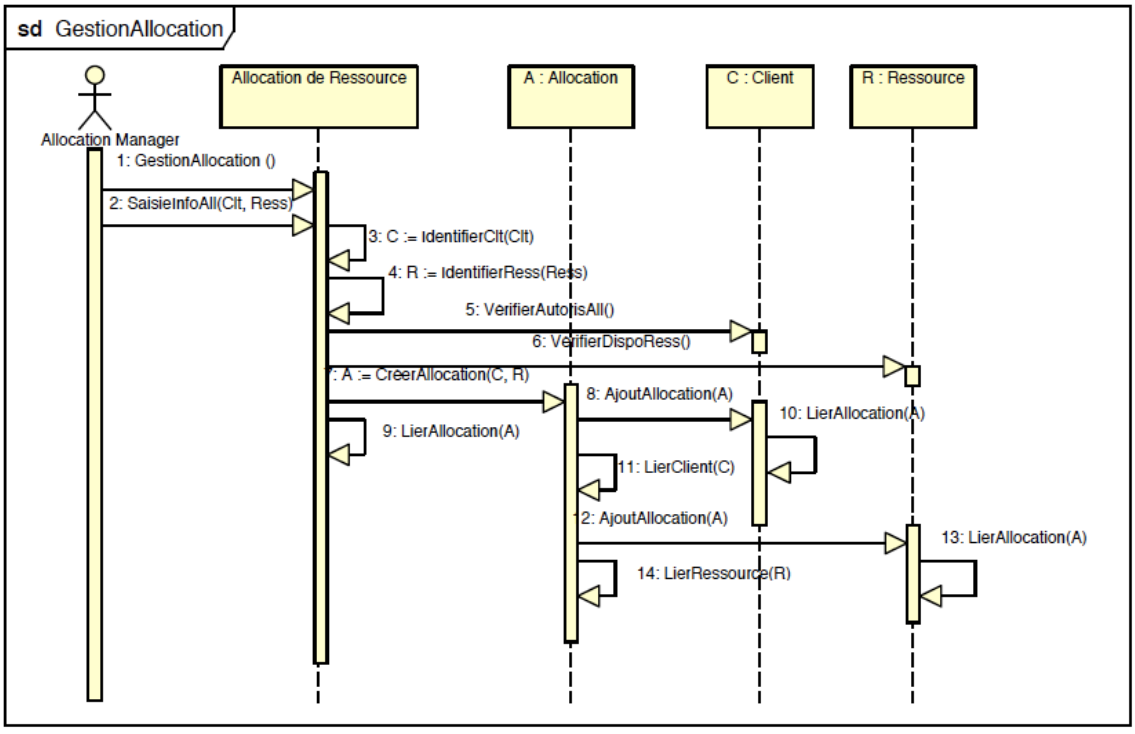


Figure 3-7 : Vue dynamique du CMP « Allocation de Ressource »

La vue dynamique d'un CMP modélise les différentes interactions entre les classes du CMP ainsi qu'une description détaillée de ses fonctionnalités.

Pour modéliser cette vue, nous utilisons un ensemble de diagrammes de séquence UML 2 (un diagramme de séquence par cas d'utilisation). Un diagramme de séquence est la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique. Ces interactions sont spécifiées dans le cadre d'un scénario d'un cas d'utilisation. La Figure 3-7 illustre la vue dynamique du CMP « Allocation de Ressource ». Dans cet exemple, cette vue est composée d'un seul diagramme de séquence puisque la vue fonctionnelle n'inclut que le cas d'utilisation « Gestion Allocation ».

d. *Vue structurelle*

La vue structurelle d'un CMP permet d'identifier les concepts encapsulés par le CMP sous forme de classes, leurs associations avec multiplicité et leurs attributs.

Pour modéliser cette vue, nous adoptons le modèle conceptuel Symphony déjà présenté dans le chapitre précédent (§1.3.3). Le modèle Symphony est un diagramme de classes avec une structuration inspirée de la technique CRC (Classe-Responsabilité-Collaboration) [Wirfs-Brock *et al.*, 1990]. Le choix de ce modèle est motivé par son avantage au niveau de la séparation des rôles, services et structure d'un composant, requise pour la spécification d'un CMP. La Figure 3-8 représente la cartographie des CM (CMP et CME). La Figure 3-9 représente la vue structurelle du CMP « Allocation de Ressource » selon le modèle Symphony. Les classes « Rôle » formalisent les liens d'utilisation entre les différents CM. Par exemple, le CMP « Allocation de Ressource » utilise les CME « Allocation », « Client » et « Ressource » par invocation de leurs classes « Interface » par l'intermédiaire des classes « Rôle » : GestionRessource, GestionClient et CréationAllocation.

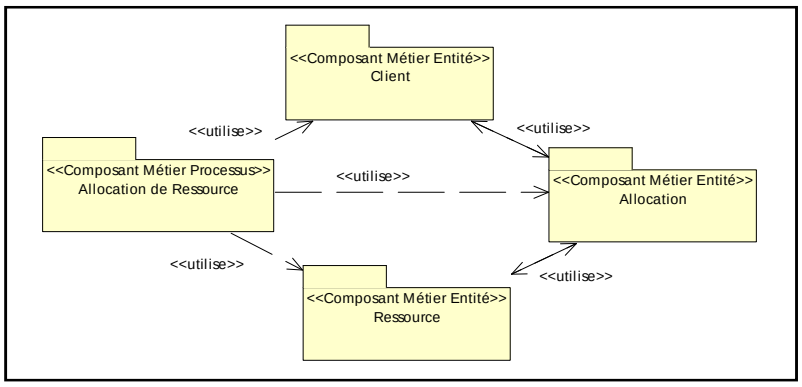


Figure 3-8 : Cartographie des CM

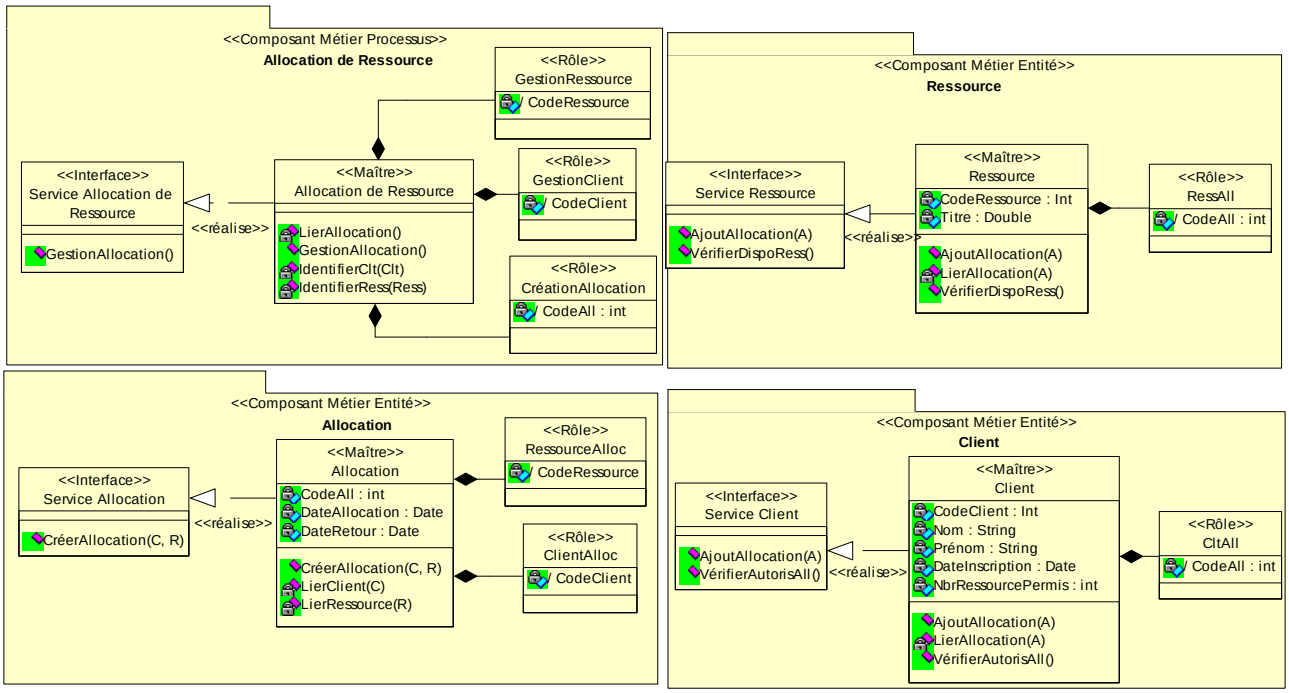


Figure 3-9 : Vue structurelle du CMP « Allocation de Ressource »

2.2 Un CMP supportant la variabilité

Un PM peut être exécuté de différentes manières en fonction de la variation des activités ou des événements à l'origine du processus. Un PM peut donc être décliné de plusieurs façons selon chaque SI. En effet, les différentes versions d'un PM doivent être analysées pour créer des CMP réutilisables dans plusieurs SI.

Le but de notre travail est la conception de CMP qui représentent des comportements abstraits récurrents de plusieurs SI d'un domaine métier donné, afin de les réutiliser dans un nouveau SI. L'idée principale consiste donc à identifier ce qui est commun et ce qui est variable entre ces SI, pour pouvoir en abstraire des informations qui partagent ces comportements.

2.2.1 Mécanisme de variabilité

Pour supporter le concept de variabilité, nous proposons que la spécification d'un CMP distingue une partie fixe et des parties variables. La partie fixe représente les propriétés du composant obligatoirement réutilisées dans chaque SI ; ces dernières sont directement intégrées dans le SI en cours de construction. Les parties variables représentent les propriétés du composant pour lesquelles la réutilisation exige un processus d'adaptation aux exigences d'un SI particulier. La variabilité permet ainsi de contrôler les parties communes et variables des artefacts de ce CMP. La réduction de la variabilité produit un ensemble de PM issus du même CMP. Ainsi, l'abstraction (conception pour la réutilisation) est un processus qui identifie les artefacts récurrents entre plusieurs SI, et la réutilisation (conception par la réutilisation) est le processus qui produit des cas de réutilisation des artefacts du CMP afin de développer des SI (Cf. Figure 3-10).

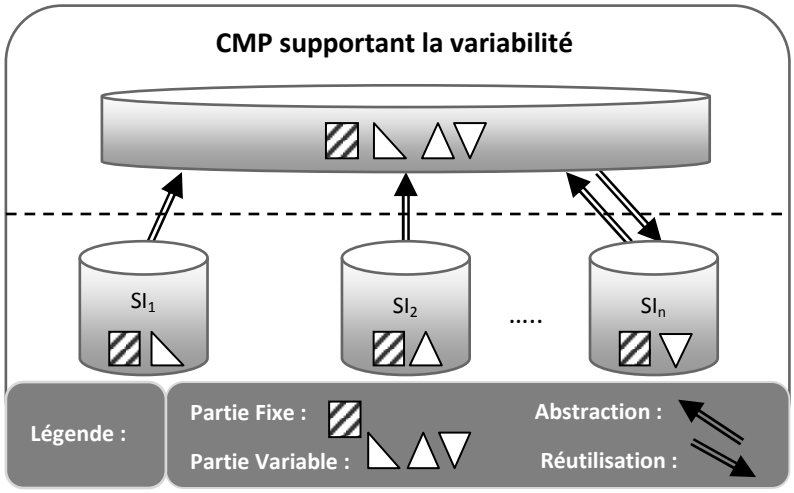


Figure 3-10 : Mécanisme d'abstraction / réutilisation des parties fixes et variables d'un CMP

Ce mécanisme est perçu dans [Cauvet *et al.*, 1999] comme des propriétés introduites au niveau de la spécification du composant. Au moment de la réutilisation, c'est en fixant la partie variable que l'on choisit une réalisation particulière et que l'on adapte le composant en fonction des spécificités du système en cours de développement.

2.2.2 Dimensions de la variabilité dans un CMP

La modélisation de la variabilité dans un CMP introduit trois nouvelles activités dans les activités classiques de l'ingénierie pour et par la réutilisation.

- L'*identification de la variabilité* : il s'agit de l'activité de définition de ce qui diffère d'un PM à un autre, pendant la modélisation d'un CMP.
- La *représentation de la variabilité* : il s'agit de la spécification explicite de la variabilité à travers l'introduction des points de variation et des variantes.
- La *réduction de la variabilité* : il s'agit de choisir les différentes variantes définies dans un CMP afin de générer des artefacts aptes à être réutilisés dans un SI.

Chacune de ces activités repose sur un ensemble de dimensions de la variabilité que nous souhaitons prendre en considération.

a) Variabilité conceptuelle

Pour la conception de CMP, nous nous focalisons sur une modélisation conceptuelle. De ce fait, la variabilité que nous représentons sur ces CMP doit être aussi conceptuelle. Il s'agit d'une variabilité liée aux modèles conceptuels des CMP, tels que les modèles UML produits lors des étapes de spécification.

b) Variabilité pour la conception

Ce type de variabilité consiste à rendre la variabilité explicite pendant la conception d'un artefact réutilisable. Il est orienté concepteur d'applications qui adapte le CMP selon les besoins du SI en cours de construction. Le CMP candidat à la réutilisation est appelé un CMP *adaptable*.

c) Origines de la variabilité

Nous distinguons deux aspects qui peuvent être à l'origine de la variabilité représentée dans des CMP.

1. Variabilité des PM : un CMP modélise des PM similaires qui peuvent être déclinés de plusieurs façons selon les besoins de chaque SI (Cf. Figure 3-11.a et Figure 3-11.b).
2. Variabilité du degré d'informatisation du PM : un CMP doit prendre en considération le degré d'informatisation d'un PM qui peut varier d'un système d'information à un autre (Cf. Figure 3-11.a et Figure 3-11.c).

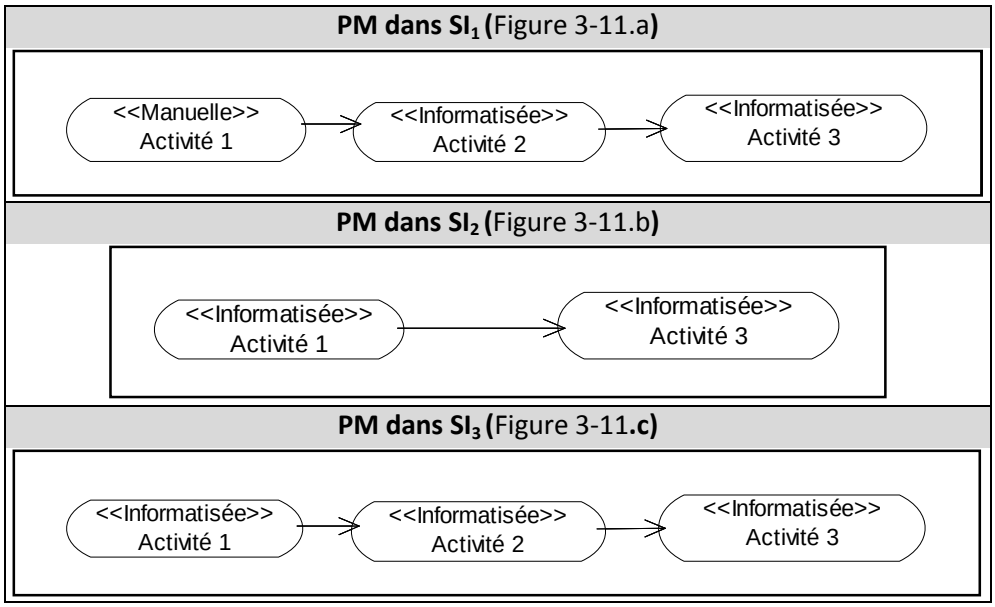


Figure 3-11 : Origines de la variabilité

d) Unités de la variabilité

Pour représenter la variabilité dans un CMP, nous utilisons les concepts de *Points de Variation* et de *Variantes*.

1. Un *point de variation* localise un endroit spécifique dans un CMP auquel une décision, prise lors de la conception d'un SI, est attachée.
2. Une *variante* représente une exécution spécifique d'un point de variation. Elle correspond aux solutions alternatives de conception.

e) Types de variabilité

Pour qu'un point de variation fournisse des contraintes sur le choix de ses variantes, nous utilisons les quatre types de variation définis dans [Bachmann *et al.*, 2001] [VanDerMaβen *et al.*, 2002], en particulier, l'*alternative* et l'*option.* Les types *ensemble d'alternatives* et *alternative optionnelle,* ne sont que des combinaisons des deux premiers types.

2.2.3 Modélisation de la variabilité

a) Variabilité multi-vues

Nous l'avons déjà dit, nous cherchons à spécifier un CMP avec une solution complète. Ainsi, un CMP doit supporter la variabilité selon ses quatre vues de développement :

- *Variabilité métier* : représente la variabilité des activités représentées par un CMP. La variabilité est modélisée par une extension des concepts du diagramme d'activités d'UML 2.
- *Variabilité fonctionnelle* : représente les différentes fonctionnalités offertes par le CMP ainsi que les variantes. La variabilité est modélisée par une extension des concepts du diagramme de cas d'utilisation d'UML 2.
- *Variabilité dynamique* : modélise la variabilité des interactions entre les classes du système ainsi qu'une description détaillée de la variabilité fonctionnelle. La variabilité est modélisée par les concepts du diagramme de séquence d'UML 2.
- *Variabilité structurelle* : représente la variabilité de la structure du système. La variabilité est modélisée par une extension du modèle conceptuel Symphony [Hassine, 2005].

L'expression de la variabilité multi-vues repose essentiellement sur la représentation des points de variation, des variantes ainsi que des contraintes de dépendance. En outre, la spécification du CMP est contrôlée par la vue métier qui représente l'organisation du CMP dans son ensemble, assure la cohésion des activités et guide le processus de modélisation et de réduction de la variabilité. Ainsi, la vue métier constitue un mécanisme essentiel de traçabilité entre les vues du CMP (Cf. Figure 3-12).

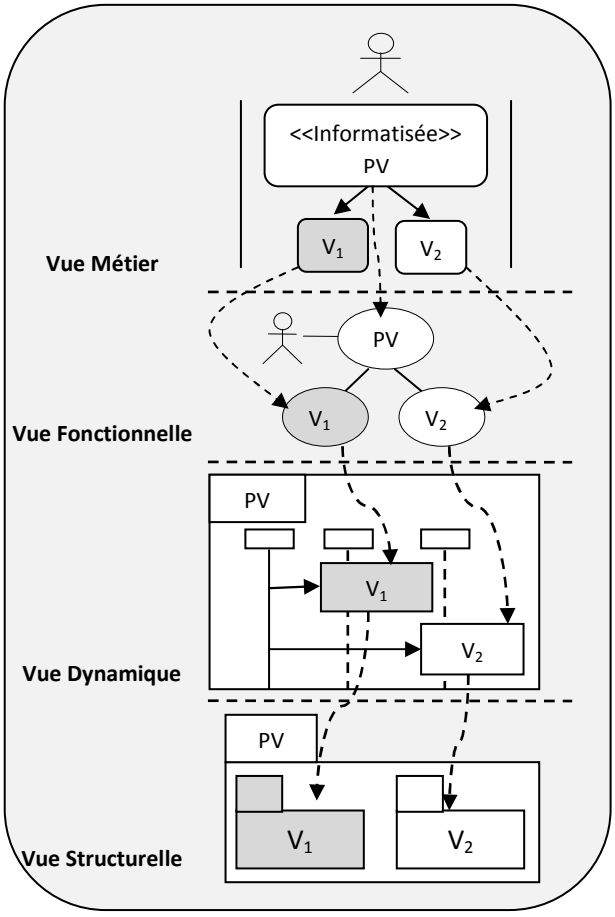


Figure 3-12 : Variabilité multi-vues

b) Extension d'UML par les concepts de la variabilité

Pour la modélisation des systèmes, UML propose un ensemble de diagrammes. Cependant, ces diagrammes restent généralement dédiés à la modélisation d'un seul système, et ne

supportent pas la modélisation de la variabilité et en particulier celle orientée conception. L'extensibilité du langage UML, grâce aux mécanismes de *stéréotypes*, d*es valeurs marquées (tagged values)* et des *contraintes* [OMG, 2007], rend possible la représentation de la variabilité. Les stéréotypes spécialisent les classes du méta-modèle, les valeurs marquées étendent les attributs des classes du méta-modèle et les contraintes sont des relations sémantiques entre éléments de modélisation qui définissent des conditions que doit vérifier le système. Ainsi, ces différents mécanismes nous permettent de représenter la variabilité sur les quatre vues de développement d'un CMP.

c) Expression des contraintes de dépendance

Nous définissons des contraintes de construction du CMP. Ce type de contraintes représente des règles que l'ensemble des éléments d'un composant doit respecter pour assurer sa cohérence, le risque étant de se retrouver devant un composant incomplet en supprimant les éléments variables.

D'autre part, les composants doivent être caractérisés par des contraintes de dépendance entre variantes. En effet, le choix d'une variante peut influencer le choix d'autres variantes. Nous distinguons deux types de contraintes inspirées des règles de composition de FODA [Kang *et al.*, 1990] :

1. La règle **d'exigence** spécifie que le choix d'une variante exige le choix d'une autre variante lors de la réutilisation du CMP.
2. La règle **d'exclusion** spécifie que le choix d'une variante interdit le choix de l'autre variante lors de la réutilisation du CMP.

d) Réduction de la variabilité

La réduction de la variabilité consiste principalement à choisir les variantes utiles spécifiées dans un CMP. Cette activité est liée à la portée de la variabilité exprimée et se produit au moment de la conception d'un SI particulier. Dans notre approche, un CMP est réduit par le concepteur du SI.

Dans la suite de ce chapitre, nous précisons les concepts de la vue métier qui guide le processus de gestion de la variabilité.

3 Vue métier d'un CMP supportant la variabilité

Cette vue propose une spécification organisationnelle d'un CMP qui peut être construite par abstraction d'un ou plusieurs PM liés à leur propre SI. L'accent est mis sur l'identification et la spécification d'une vue d'ensemble des activités internes d'un métier, dont l'objectif est de fournir un résultat observable et mesurable pour un utilisateur individuel du métier. La vue métier est modélisée par un diagramme d'activités UML étendu par les concepts de variabilité.

Dans la suite de cette section, nous présentons les mécanismes que nous adoptons pour représenter la variabilité sur la vue métier. Nous illustrons i) les endroits où la variabilité survient, ii) la représentation de la variabilité ainsi que son mécanisme de réduction en utilisant des exemples appropriés et iii) finalement les différentes extensions d'UML introduites.

3.1 Variabilité dans la vue Métier

Un diagramme d'activités est un graphe orienté qui décrit un enchaînement de traitements (flots de contrôle et de données). L'enchaînement des activités peut être soumis à des branchements conditionnels ou à des synchronisations. Les couloirs d'activités permettent de représenter la répartition de la responsabilité des activités entre les différents acteurs. Les activités sont reliées par des transitions qui sont déclenchées par des événements. Une transition peut être assortie d'une condition de garde qui bloque la transition si elle n'est pas vérifiée.

La variabilité dans la vue métier survient essentiellement sur les activités, mais peut également être portée par les flots de contrôle, les objets et les acteurs. Le Tableau 3-2 illustre la variabilité dans la vue métier.

Élément	**Définition**	**Variabilité**
Activité	Une activité modélise une étape dans l'exécution d'un processus. Elle représente une tâche effectuée par un acteur donné.	Une activité particulière peut être présente dans un PM d'un SI et absente dans le même PM d'un autre SI, ou encore, elle peut être déclinée de plusieurs façons selon les besoins de chaque SI.
Degré d'informatisation	Une activité peut être manuelle ou informatisée dans un processus donné.	Le choix du degré de l'informatisation d'un processus doit être effectué par le concepteur selon les besoins du SI en cours de construction.
Acteur/partition	Les activités peuvent être placées dans des couloirs (partitions) qui représentent des systèmes ou des acteurs.	Si toutes les activités exécutées par un acteur donné sont optionnelles, alors cet acteur devient optionnel car ses activités peuvent ne pas être choisies lors de la réutilisation. En outre, un acteur peut avoir un certain rôle dans un PM d'un SI et un rôle différent dans le même PM d'un autre SI.
Objet (flot de données)	Un objet est une donnée qui peut initier une activité, qui peut être utilisée par une activité ou qui est modifiée par une activité. L'ensemble de ces transitions constitue un flot de données.	La variabilité sur les objets découle de celle des activités. En effet, la présence ou l'absence d'une activité donnée lors de la réutilisation, peut provoquer la présence ou l'absence d'un objet.
Flot de contrôle	Le flot de contrôle modélise l'enchaînement des traitements dans un diagramme. Le passage d'une activité vers une autre est matérialisé par une transition.	La variabilité des activités influence l'enchaînement du traitement dans chaque PM incluant ces activités. Par conséquent, chaque réutilisation de cette vue engendre une variation sur le flot de contrôle produisant ainsi un

	Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre.	chemin particulier dans le diagramme d'activités pendant chaque réutilisation.

Tableau 3-2 : Variabilité dans la vue métier

À partir du Tableau 3-2, nous extrayons les différentes formes de variabilité qui peuvent être représentées dans la vue métier d'un CMP. Le Tableau 3-3 résume l'ensemble de ces formes dont les mécanismes de représentation sont donnés dans la section suivante.

Élément	**Formes de variabilité**
Activité	Alternative/ Alternative Optionnelle /Option /Ensemble d'alternatives
	Manuelle / Informatisée
Acteur	Variation de Rôle
	Optionnel
Objet	Optionnel

Tableau 3-3 : Formes de variabilité dans la vue métier

3.2 Représentation de la variabilité

La représentation de la variabilité dans un diagramme d'activités repose essentiellement sur l'introduction des concepts clés de la variabilité, à savoir : point de variation et variantes. Dans cette vue, nous considérons l'activité comme une unité de variabilité, afin d'unifier les concepts utilisés et de garantir une cohérence de représentation. Pour l'introduction de ces concepts, nous nous basons sur l'utilisation des **stéréotypes** « *Variation* » et « *Variant* » sur des activités supportant la variabilité. La transition d'une activité stéréotypée « *Variation* » vers une activité stéréotypée « *Variant* » est conditionnée par le choix du concepteur. Notons qu'une activité de type « Variation » peut être de plusieurs types : alternative, option, ensemble d'alternatives et alternative optionnelle. La différenciation entre ces quatre types de variation se fait par des **valeurs marquées** associées à l'activité « Variation » et qui déterminent les cardinalités des choix possibles. Dans la suite de cette partie, nous présentons chaque type de variation selon l'organisation suivante : sa définition, la notation associée ainsi que les règles de réduction qui assisteront le concepteur dans ses choix. Néanmoins, nous donnons d'abord quelques définitions utiles :

- *Flot entre deux activités* : le flot entre deux activités A et B est l'ensemble des flots de données et de contrôle liant A à B.
- *Flot entrant de l'activité A* : flot de contrôle et de données dont l'activité finale est l'activité A.
- *Flot sortant de l'activité A* : flot de contrôle et de données dont l'activité initiale est l'activité A.
- *Supprimer le flot entrant de l'activité A* : supprimer le flot de contrôle et de données dépendant de l'activité A et dont l'activité finale est l'activité A.
- *Supprimer le flot sortant de l'activité A* : supprimer le flot de contrôle et de données dépendant de l'activité A et dont l'activité initiale est l'activité A.

Nous représentons le flot entrant et le flot sortant d'une activité « A » (Cf. Figure 3-13.a) par des activités appelées respectivement « Flot entrant de A » et « Flot sortant de A ». Nous utilisons dans le premier cas une transition de l'activité « Flot entrant de A » vers l'activité « A » et dans le deuxième cas une transition de l'activité « A » vers l'activité « Flot sortant de A » (Cf. Figure 3-13.b).

Figure 3-13.a

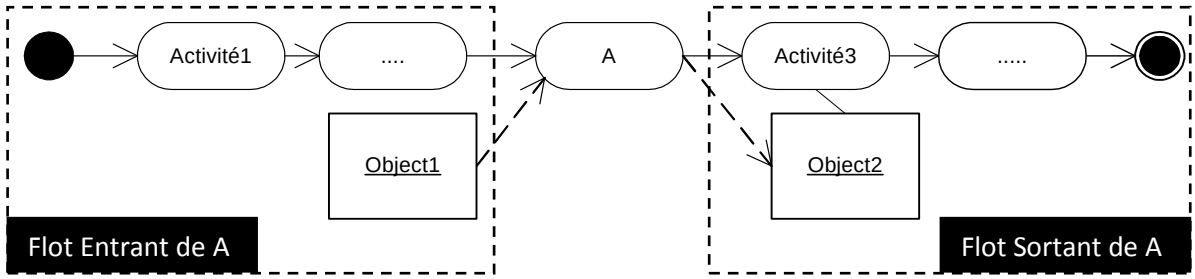


Figure 3-13.b

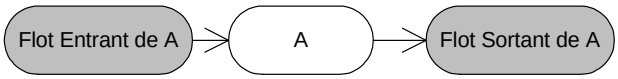


Figure 3-13 : Flots entrant et sortant d'une activité

3.2.1 Variabilité des activités

a) Activité de type « Alternative »

✓ *Définition*

Une activité PV est un point de variation de type « Alternative » si PV permet au concepteur le choix d'une seule activité parmi plusieurs variantes.

✓ *Notation*

Comme nous l'avons présenté précédemment, nous introduisons les concepts de point de variation et de variante par l'utilisation des stéréotypes. Dans ce type de variation, nous annotons l'activité variation par la valeur marquée (min : 1, max : 1) exprimant ainsi les cardinalités à respecter lors du choix des variantes de ce type de variation (Cf. Figure 3-14.a).

✓ *Règles de réduction*

Si une variante V_i est choisie par le concepteur, l'activité PV est remplacée par l'activité V_i et le flot entrant de PV devient le flot entrant de V_i. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-14.b).

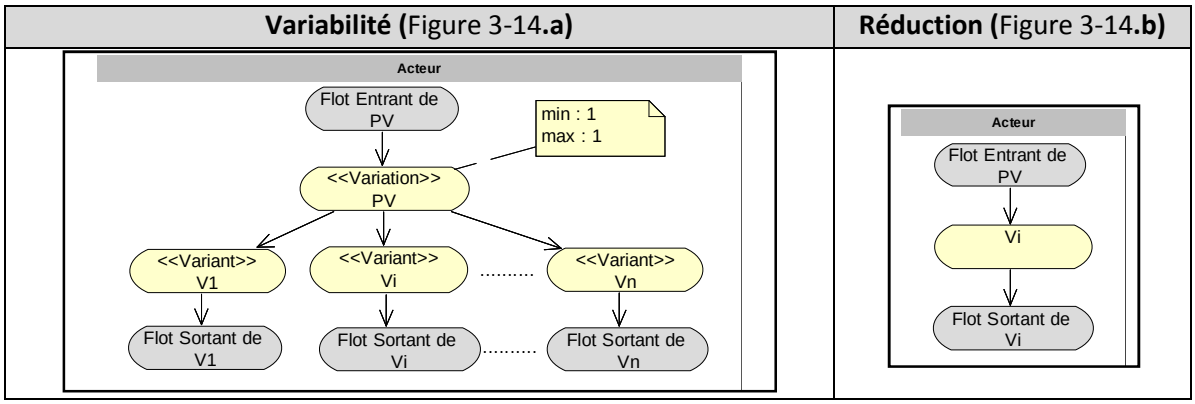


Figure 3-14 : Notations utilisées pour une variation « Alternative »

✓ *Exemple*

Pour illustrer la représentation de la variabilité dans ce type de variation, nous utilisons un fragment du processus « Allocation de ressources » comme exemple. Dans la Figure 3-15.a, nous illustrons le cas où une activité peut être exécutée par le même acteur mais de différentes façons. Dans cet exemple, le traitement d'une demande d'allocation peut se faire avec ou sans gestion de réservation. Si par exemple, le concepteur choisit la variante « Traitement sans réservation », la réduction du diagramme donne lieu au diagramme illustré dans la Figure 3-15.b.

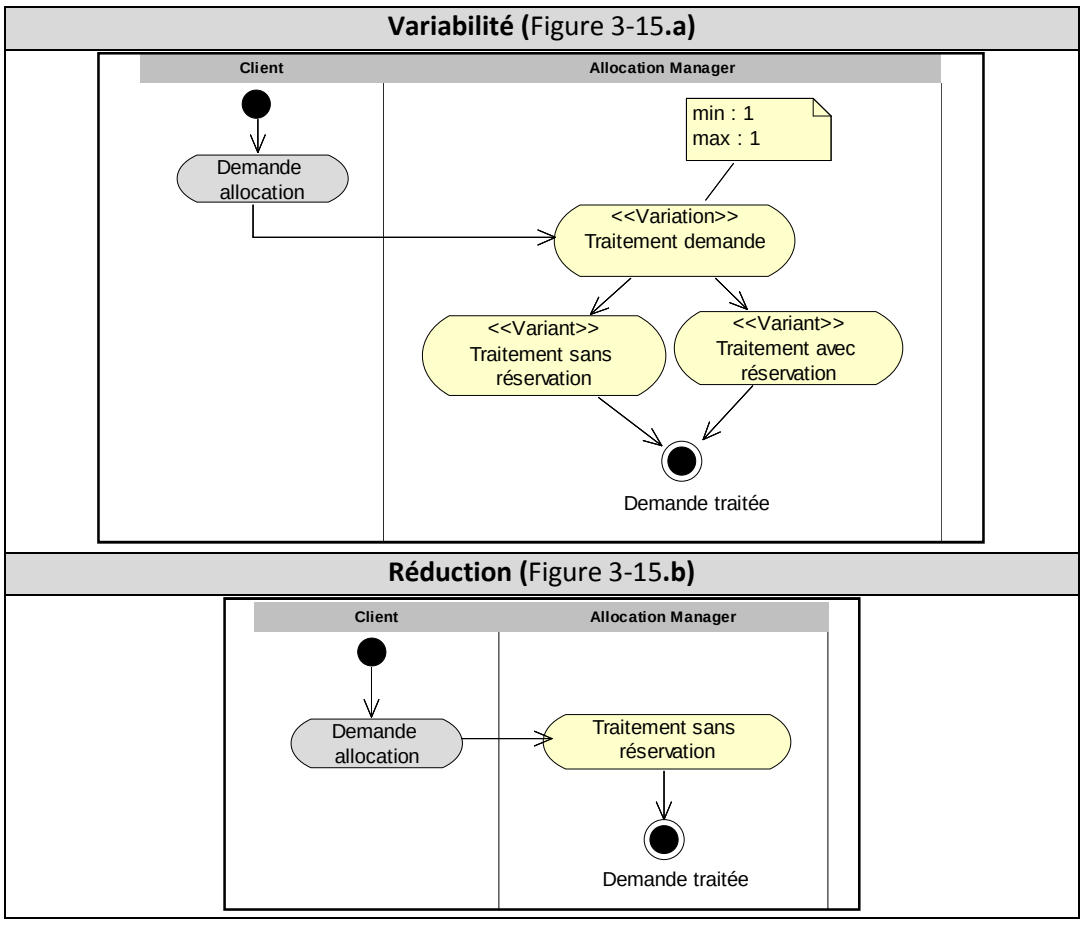


Figure 3-15 : Exemple d'une variation « Alternative »

b) Activité de type « Alternative-Optionnelle »

✓ *Définition*

Une activité PV est un point de variation de type « Alternative-optionnelle » si PV peut être remplacée ou non par une seule variante parmi un ensemble d'alternatives.

✓ *Notation*

Dans ce type de variation, nous annotons l'activité variation par la valeur marquée (min : 0, max : 1) exprimant ainsi les cardinalités à respecter lors du choix des variantes de ce type de variation (Cf. Figure 3-16.a).

✓ *Contraintes de construction*

Dans ce type de variation, il faut vérifier une contrainte que nous appelons « *contrainte de construction* », obligatoire pour maintenir la cohérence du diagramme d'activités. Cette contrainte est donnée de la manière suivante :

Le flot entrant d'une variation de type « Alternative-Optionnelle » doit contenir au moins une transition vers une autre activité non stéréotypée par « Variation » et « Variant », ou vers une activité stéréotypée « Variation » dont le min=1.

✓ *Règles de réduction*

- Si une variante V_i est choisie par le concepteur, l'activité PV est remplacée par l'activité V_i et le flot entrant de PV devient le flot entrant de V_i. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-16.b).
- Si aucune variante n'est sélectionnée, l'activité PV, et ses variantes Vi sont supprimées avec leur flot sortant (Cf. Figure 3-16.c).

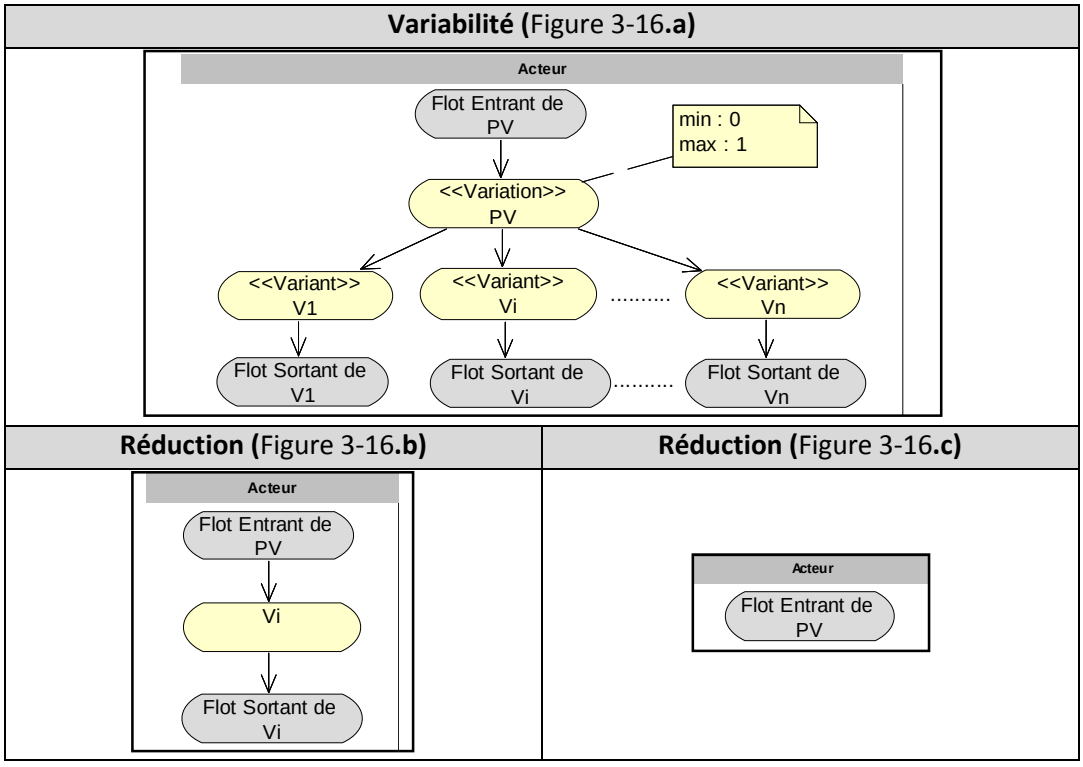


Figure 3-16 : Notations utilisées pour une variation « Alternative-Optionnelle »

✓ *Exemple*

Pour illustrer la représentation de la variabilité dans ce type de variation, nous utilisons un fragment du processus « Enregistrement client » comme exemple. Dans la Figure 3-17.a, après l'activité d'enregistrement d'un client, certains systèmes incluent une activité d'impression d'un justificatif et d'autres se restreignent à un simple enregistrement des informations du client. L'impression du justificatif peut se faire de différentes manières, par exemple par impression d'une carte ou d'un reçu. Dans ce cas, un concepteur peut choisir une des variantes proposées (Cf. Figure 3-17.b) ou ne rien choisir (Cf. Figure 3-17.c) selon les besoins du SI en cours de construction. Notons que l'activité « Enregistrer Client », flot entrant de cette variation, a une transition vers une activité fixe (nœud final) puisque la variation est de cardinalité minimale égale à 0. Cette contrainte a permet de maintenir la cohérence du diagramme obtenu après la réduction de la variabilité (Cf. Figure 3-17.c).

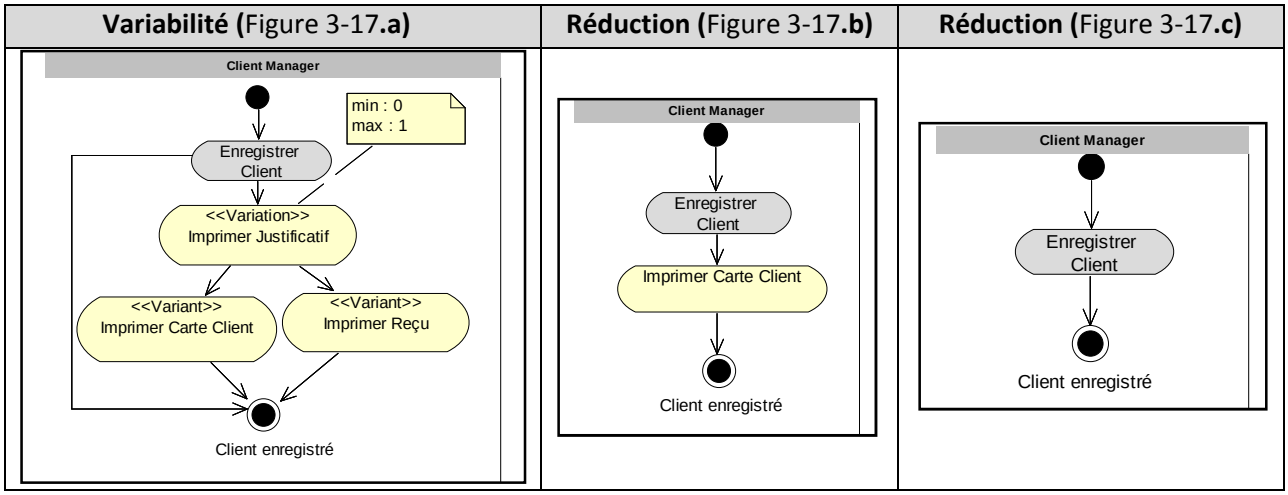


Figure 3-17 : Exemple d'une variation « Alternative-Optionnelle »

c) Activité de type « Option »

✓ *Définition*

Une activité PV est un point de variation de type « Option » si chaque variante de PV peut être choisie ou non par le concepteur.

✓ *Notation*

Dans ce type de variation, nous annotons l'activité variation par la valeur marquée (min : 0, max : n) exprimant ainsi les cardinalités à respecter lors du choix des variantes de ce type de variation (Cf. Figure 3-18.a).

✓ *Contraintes de construction*

Dans ce type de variation, il faut vérifier la même *contrainte de construction* donnée dans 3.2.1.b, car il s'agit d'une contrainte liée à la cardinalité minimale 0.

✓ *Règles de réduction*

1) Si un ensemble $\{V_i ; 1 \leq i \leq n\}$ de variantes est choisi par le concepteur, le point de variation est remplacé par une *transition composite*, qui est matérialisée par un nœud de décision où arrive l'ensemble du flot entrant de PV et d'où partent plusieurs transitions vers les variantes sélectionnées. Un deuxième nœud de décision est créé où arrive une transition de chaque V_i

sélectionnée et d'où partent une transition vers le flot sortant de V_i et une transition vers le premier nœud de décision. Cette deuxième décision permet à l'utilisateur final le choix de plusieurs options à l'exécution du processus. Toutes les décisions sont conditionnées par le choix de l'utilisateur final. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-18.b).

2) Si une seule variante V_i est choisie par le concepteur, l'activité PV est remplacée par l'activité V_i et le flot entrant de PV devient le flot entrant de V_i. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-16.b).

3) Si aucune variante n'est sélectionnée, l'activité PV et ses variantes V_i sont supprimées avec leur flot sortant (Cf. Figure 3-16.c).

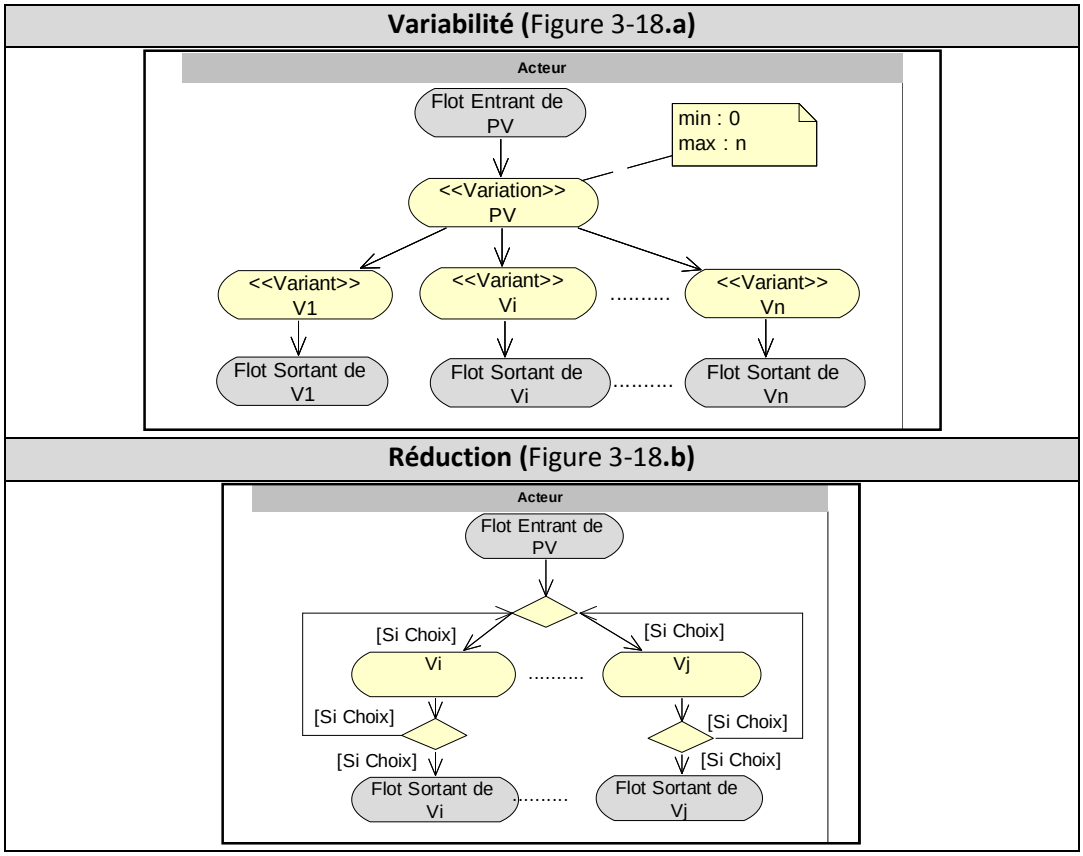


Figure 3-18 : Notations utilisées pour une variation « Option »

✓ *Exemple*

Pour illustrer la représentation de la variabilité dans ce type de variation, nous utilisons un fragment du processus « Gestion des devises » comme exemple. Certains systèmes sont destinés à des banques qui ne gèrent que des comptes en Euro et ne supportent pas la gestion des devises. L'activité « Convertir Devise » est donc représentée comme un point de variation de type option en offrant plusieurs choix possibles : convertir vers dollar, vers euro, etc. (Cf. Figure 3-19.a). Le concepteur peut choisir ou non une variante proposée. Si le concepteur choisit au moins deux variantes, le diagramme d'activités supportant la variabilité se transforme en un autre diagramme d'activités (Cf. Figure 3-19.b) en utilisant les règles de réduction précédemment présentées.

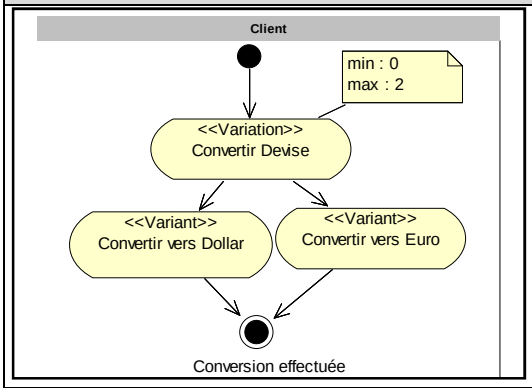
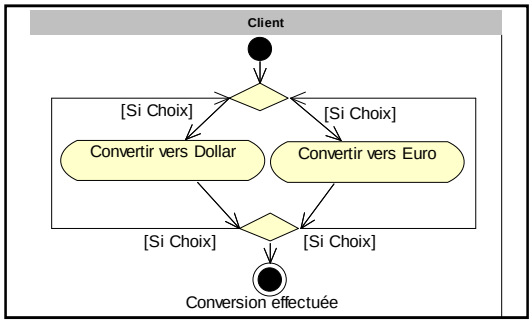
Variabilité (Figure 3-19**.a)**	**Réduction (**Figure 3-19**.b)**
	

Figure 3-19 : Exemple d'une variation « Option »

d) Activité de type « Ensemble d'alternatives »

✓ *Définition*

Une activité PV est un point de variation de type « Ensemble d'alternatives » si PV doit être remplacée par au moins une variante parmi un ensemble d'alternatives. Au moment de l'exécution, une seule variante est choisie par l'utilisateur final.

✓ *Notation*

Pour ce type de variation, nous annotons l'activité variation par la valeur marquée (min : 1, max : n) exprimant ainsi les cardinalités à respecter lors du choix des variantes de ce type de variation (Cf. Figure 3-20.a).

✓ *Règles de réduction*

1. Si un ensemble $\{V_i ; 1 \leq i \leq n\}$ de variantes est choisi par le concepteur, le point de variation est remplacé par une *transition composite*, qui est matérialisée par un nœud de décision où arrive l'ensemble du flot entrant de PV et d'où partent plusieurs transitions vers les variantes sélectionnées. Le nœud de décision est conditionné par le choix de l'utilisateur final. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-20.b).

2. Si une seule variante V_i est choisie par le concepteur, l'activité PV est remplacée par l'activité V_i et le flot entrant de PV devient le flot entrant de V_i. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-16.b).

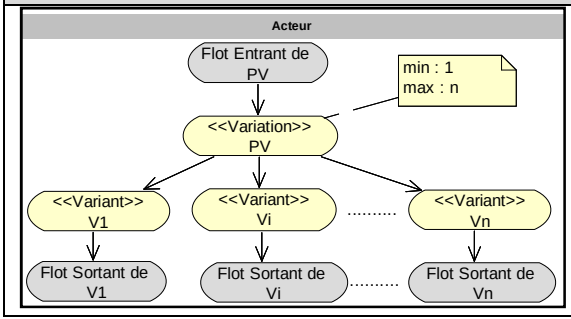
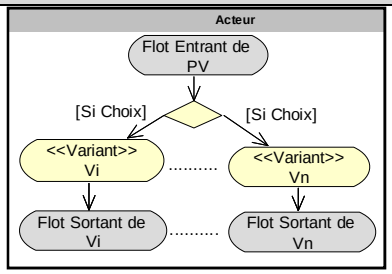
Variabilité (Figure 3-20**.a)**	**Réduction (**Figure 3-20**.b)**
	

Figure 3-20 : Notations utilisées pour une variation « Ensemble d'alternatives »

✓ *Exemple*

Pour illustrer la représentation de la variabilité dans ce type de variation, nous utilisons un fragment du processus « Gestion de paiement » comme exemple. Dans ce processus, pour valider un paiement, le caissier doit saisir le type de paiement selon qu'il s'agit d'un paiement par espèce, par carte bancaire ou par chèque. Un système incluant la gestion de paiement doit au moins supporter l'un de ces types (Cf. Figure 3-21.a). Ainsi, le concepteur doit choisir au moins une des variantes, et au moment de l'exécution une seule variante sera choisie (Cf. Figure 3-21.b), d'où la nécessité de l'utilisation d'un point de variation de type ensemble d'alternatives.

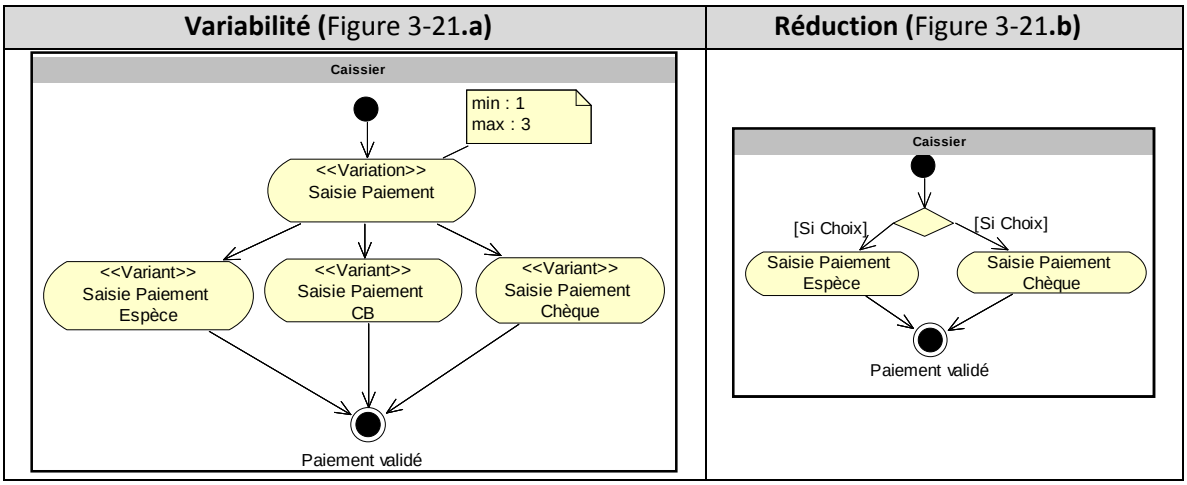


Figure 3-21 : Exemple d'une variation « Ensemble d'alternatives »

3.2.2 Variabilité du degré d'informatisation des activités

✓ *Définition*

Une activité PV est un point de variation de type « choix d'informatisation » si PV est une activité de type « alternative » et qui offre deux variantes possibles : manuelle et informatisée. Le concepteur doit choisir entre les deux variantes.

✓ *Notation*

Pour ce type de variation, nous annotons l'activité variation par la valeur marquée (min : 1, max : 1). Les deux variantes de cette activité sont stéréotypées « Manuelle » et « Informatisée » (Cf. Figure 3-22.a).

✓ *Règles de réduction*

Si une variante V_i est choisie par le concepteur, l'activité PV est remplacée par l'activité V_i et le flot entrant de PV devient le flot entrant de V_i. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-22.b).

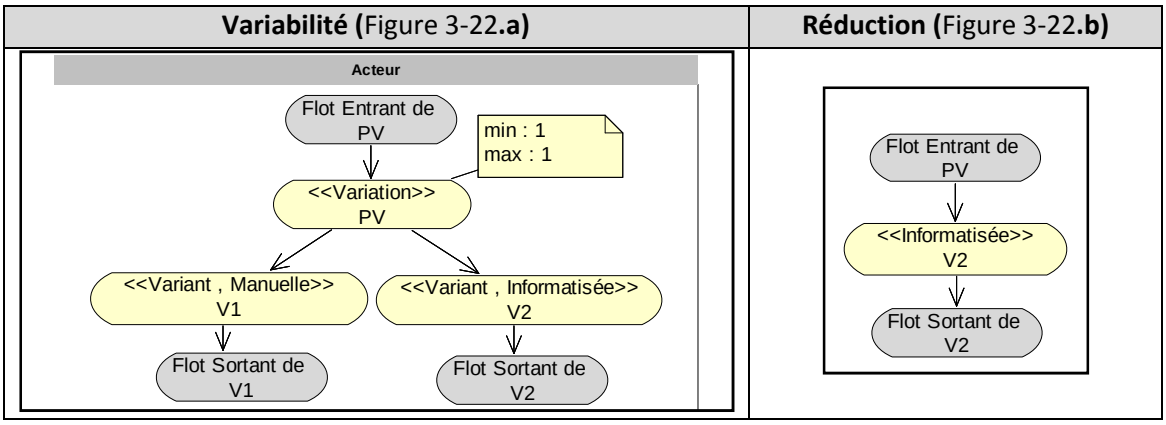


Figure 3-22 : Notations utilisées pour une variation « Choix d'informatisation »

✓ *Exemple*

Pour illustrer la représentation de la variabilité dans ce type de variation, nous utilisons un fragment du processus « Gestion d'allocation » comme exemple. Dans ce processus, la demande d'allocation peut être manuelle (dans ce cas, le client passe sa demande en remplissant une fiche papier) ou informatisée (dans ce cas, le client passe sa demande en remplissant un formulaire électronique). Par contre, l'activité « traiter Demande » est considérée informatisée dans n'importe quel SI réutilisant ce composant (Cf. Figure 3-23.a). Le concepteur doit choisir entre les variantes proposées exprimant ainsi le degré d'informatisation de son SI (Cf. Figure 3-23.b).

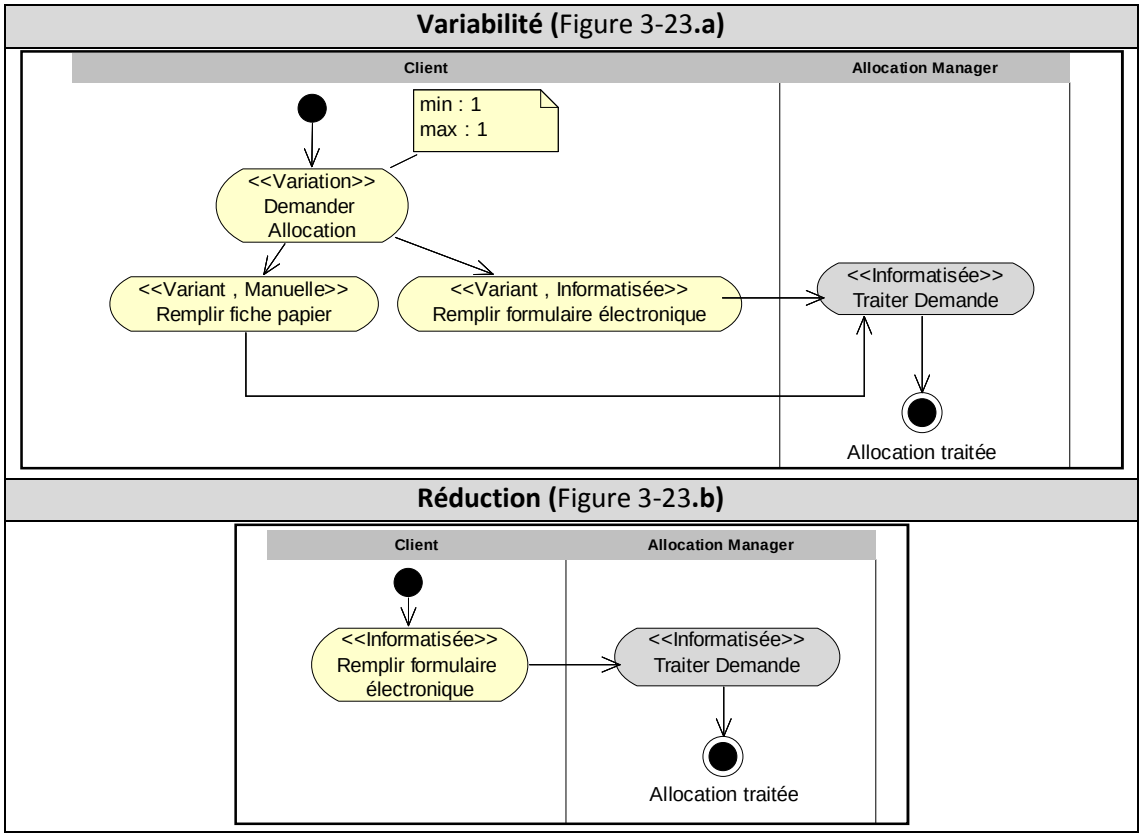


Figure 3-23 : Exemple d'une variation « Choix d'informatisation »

3.2.3 Variabilité des acteurs

✓ *Définition*

Si toutes les activités exécutées par un acteur donné sont optionnelles, alors cet acteur devient optionnel car ses activités peuvent ne pas être choisies lors de la réutilisation. En outre, un acteur peut avoir un rôle dans un PM d'un SI et un rôle différent dans le même PM d'un autre SI.

Cas 1 : Le rôle d'un acteur A est variable si au moins une activité de A peut être exécutée par d'autres acteurs. Le concepteur doit choisir l'acteur responsable de cette activité.

Cas 2 : Un acteur est optionnel si toutes ses activités peuvent ne pas être choisies lors de la réutilisation.

✓ *Notation*

Pour représenter la variabilité dans le **cas 1**, nous utilisons une activité de type « Alternative ». Chaque variante de cette activité est incluse dans une partition correspondante à l'acteur responsable de son exécution. (Cf. Figure 3-24.a).

Dans le **cas 2**, aucune nouvelle notation n'est nécessaire puisque ce type de variabilité est une conséquence de la variabilité des activités.

✓ *Règles de réduction*

Cas 1 : Si une variante V_i est choisie par le concepteur, le flot entrant de PV devient le flot entrant de V_i en supprimant l'activité PV. Les variantes non sélectionnées sont supprimées avec leur flot sortant (Cf. Figure 3-24.b).

Cas 2 : Si toutes les activités d'un acteur sont supprimées au moment de la réduction de la variabilité, cet acteur est aussi supprimé.

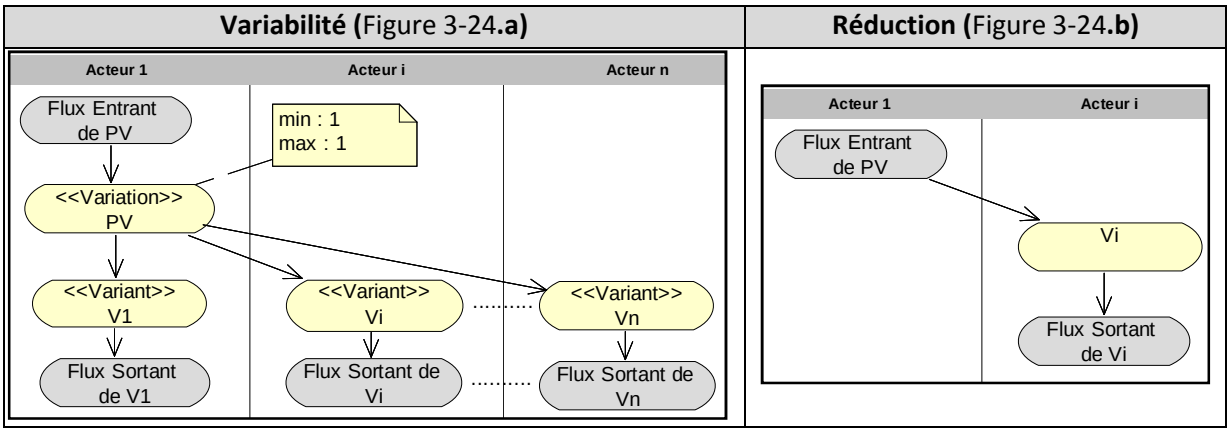


Figure 3-24 : Notations utilisées pour une variation des acteurs

✓ *Exemple*

Dans la Figure 3-25.a, nous illustrons le cas où le rôle d'un acteur donné est variable. Dans cet exemple, la saisie d'une demande d'allocation peut être faite par le « Client » ou par le « Allocation Manager », d'où la nécessité de représenter la variabilité. Si, par exemple, le concepteur choisit « Allocation Manager » comme responsable de cette activité, la réduction du diagramme donne lieu au diagramme illustré dans la Figure 3-25.b.

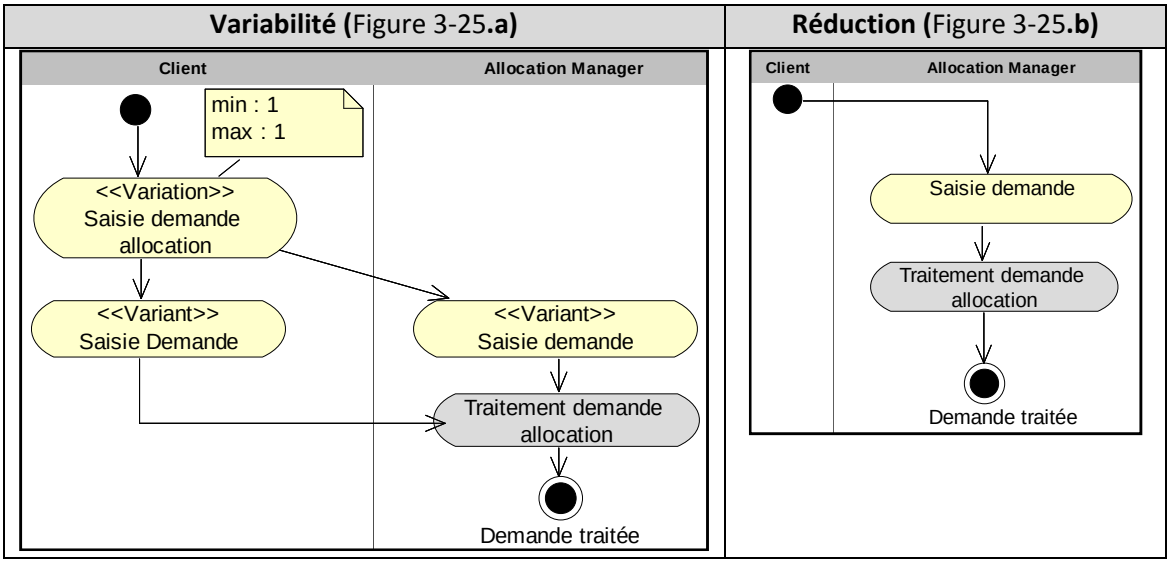


Figure 3-25 : Exemple d'une variation d'acteurs

3.2.4 Variabilité des objets

✓ *Définition*

Un objet est optionnel s'il est produit par un flot qui contient au moins une activité stéréotypée « Variant ».

✓ *Notation*

Pour représenter un objet optionnel, aucune nouvelle notation n'est nécessaire puisque la variabilité d'un objet est considérée comme une conséquence de la variabilité des activités.

✓ *Règles de réduction*

Si le flot qui produit un objet est supprimé au moment de la réduction de la variabilité, cet objet est aussi supprimé.

✓ *Exemple*

Dans un système de gestion d'une bibliothèque, l'inscription d'un client peut être gratuite ou payante. L'inscription payante provoque la création d'un objet « Facture ». Cet objet est optionnel puisque le concepteur peut choisir ou non la variante inscription payante (Cf. Figure 3-26.a). Dans le cas où le concepteur choisit une inscription gratuite, l'objet « Facture » est supprimé puisqu'il fait partie du flot supprimé (Cf. Figure 3-26.b).

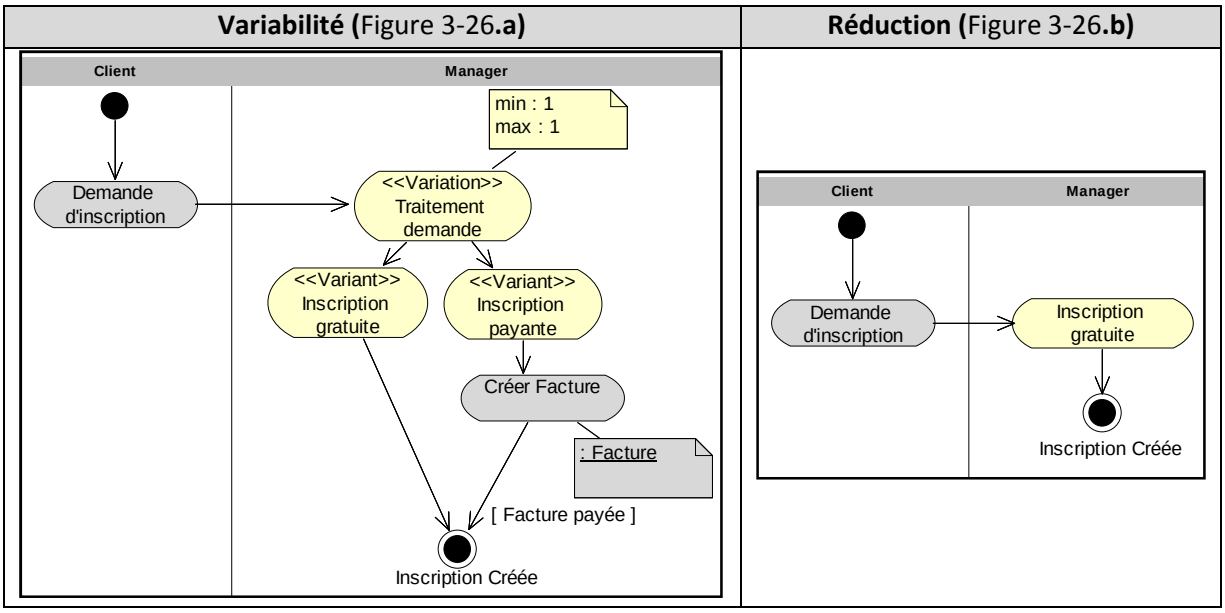


Figure 3-26 : Exemple d'une variation d'objets

3.2.5 Contraintes de dépendance

Les contraintes de dépendance dans la vue métier aident le concepteur à maintenir la cohérence du diagramme réduit après le choix des variantes. Nous définissons dans ce qui suit la notion d'exigence et d'exclusion entre deux variantes de la vue métier.

✓ *Définition*

- Si une variante A *exige* une variante B, alors le choix de A implique le choix de B.
- Si une variante A *exclut* une variante B, alors le choix de A interdit le choix de B.

Notons que la contrainte d'exigence concerne particulièrement des variantes qui ne sont pas liées par des flots. Une relation d'exigence entre variantes liées par des flots est implicite car elle est obtenue par les règles de construction du diagramme d'activités qui est un graphe orienté.

✓ *Propriétés des contraintes*

Nous donnons ici des propriétés des contraintes de dépendance entre deux variantes :

Transitivité

- Si une variante A *exige* une variante B et B *exige* une variante C, alors A *exige* C.

Symétrie

- Si une variante A *exclut* une variante B, alors B *exclut* A.

✓ *Notation*

Pour représenter les contraintes de dépendance entre deux variantes, nous utilisons une valeur marquée associée à une activité variante et qui détermine ses variantes exigées (Exige) et ses variantes exclues (Exclut) (Cf. Figure 3-27).

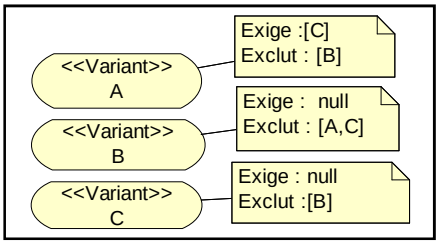


Figure 3-27 : Représentation des contraintes de dépendance

✓ *Contraintes de construction*

- Si une variante A *exclut* une variante B, alors il ne doit pas y avoir un flot entre A et B.

✓ *Règles de réduction*

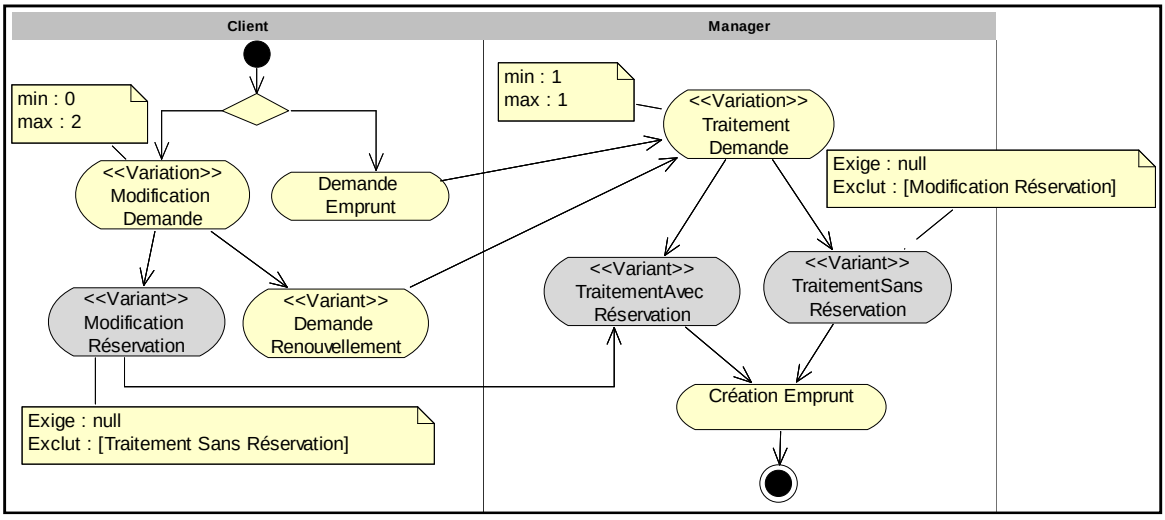
- Si une variante est choisie, alors toutes les variantes liées avec elle par une contrainte d'*exigence* doivent être conservées.

- Si une variante est choisie, alors toutes les variantes liées avec elle par une contrainte d'*exclusion* ne doivent pas être conservées.

✓ *Exemple*

Pour illustrer les contraintes de dépendance dans un diagramme d'activités, nous utilisons l'exemple « Gestion d'Emprunt ». Dans cet exemple, nous représentons deux types de points de variation : « Modification Demande » de type « Option » et « Traitement Demande » de type « Alternative » (Cf. Figure 3-28.a). Les valeurs marquées des deux variantes « Modification Réservation » et « TraitementSansRéservation » indiquent que ces deux variantes sont liées par un lien d'exclusion. En outre, les deux variantes ne sont liées par aucun flot, ce qui signifie que la contrainte de construction est validée. Par conséquent, le choix de la variante « Modification Réservation » interdit le choix de la variante « Traitement sans Réservation » (Cf. Figure 3-28.b).

Variabilité (Figure 3-28**.a)**



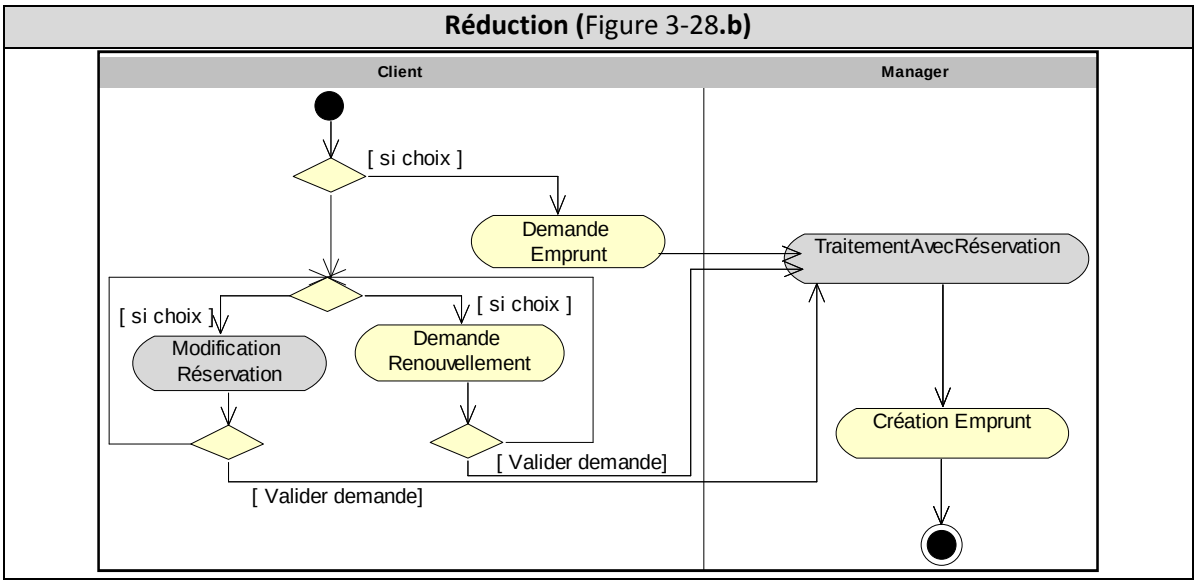


Figure 3-28 : Exemple de contraintes de dépendance

3.2.6 Cas particuliers

Pour illustrer la représentation de la variabilité, nous avons utilisé des mécanismes spécifiques pour chaque type de variation. Dans le cas où une variation inclut plusieurs types de variation, nous utilisons une seule variation avec les différentes variantes possibles. Les cardinalités minimale et maximale sont déduites de l'ensemble des types utilisés.

La Figure 3-29 illustre un exemple de représentation simultanée de deux types de variation (Rôle variable et Choix d'informatisation). Le point de variation de type « Alternative » permet le choix d'une seule variante parmi plusieurs.

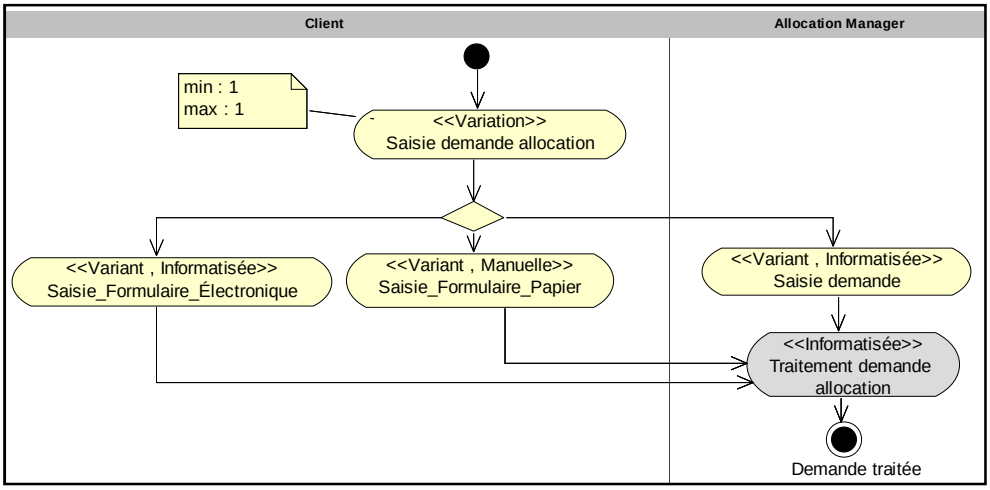


Figure 3-29 : Représentation simultanée de deux types de variation

3.3 Extensions d'UML

Les stéréotypes introduits pour la modélisation de la variabilité dans les diagrammes d'activités sont définis comme des extensions du méta-modèle d'UML 2 spécifiant les actions [OMG, 2007]. En effet, et pour rester en conformité avec UML 2, nous considérons que :

– la vue métier d'un CMP est une spécialisation de la classe « Activity » UML 2 ;

– les activités de cette vue sont représentées par des classes « Action » UML 2.

Les classes grisées illustrent les méta-classes concernées par nos extensions (Cf. Figure 3-30).

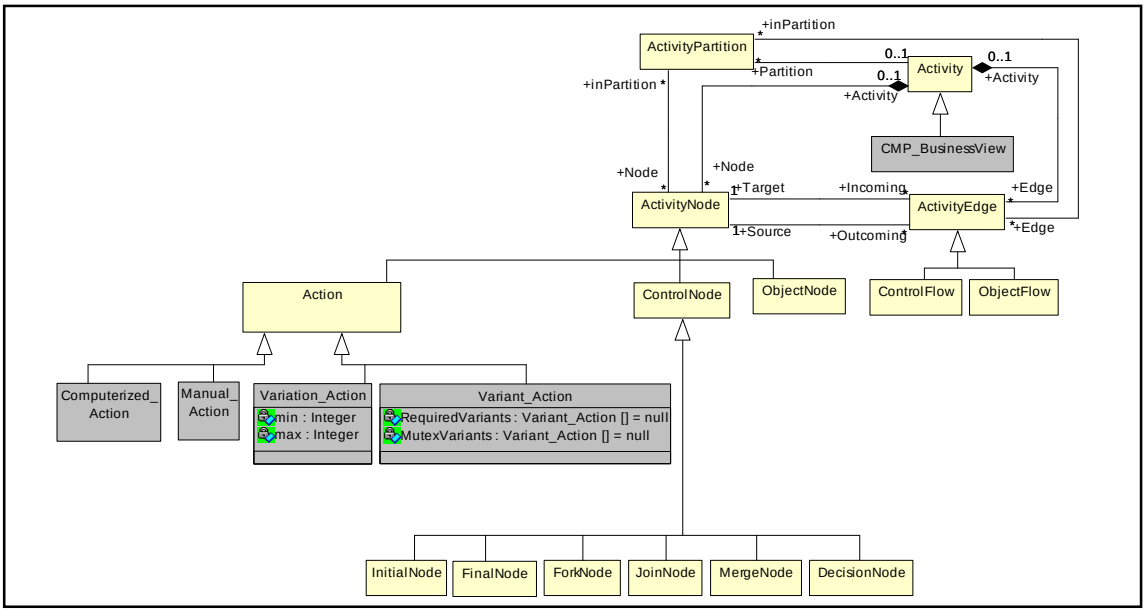


Figure 3-30 : Extension des concepts du diagramme d'activités

3.3.1 Description des extensions

Classe	**Description**
CMP_BusinessView	**Superclasse** : Activity. **Description** : cette classe modélise la vue métier d'un CMP. **Notation :** diagramme d'activités.
Variation_Action	**Superclasse** : Action. **Description** : cette classe modélise une activité de type point de variation. **Attributs** : min et max sont des entiers qui modélisent les cardinalités minimale et maximale pour le choix des variantes. **Notation :** action stéréotypée « Variation ».
Variant_Action	**Superclasse** : Action. **Description** : cette classe modélise une activité de type variante. **Attributs** : *RequiredVariants : Variant_Action []* : modélise l'ensemble des variantes exigées par le choix d'une variante. Cet attribut est initialisé par une valeur nulle. *MutexVariants : Variant_Action []* : modélise l'ensemble des variantes exclues par le choix d'une variante. Cet attribut est initialisé par une valeur nulle. **Notation :** action stéréotypée « Variant ».
Manual_Action	**Superclasse** : Action. **Description** : cette classe modélise une activité manuelle. **Notation :** action stéréotypée « Manuelle ».

Computerized_Action	**Superclasse** : Action. **Description** : cette classe modélise une activité informatisée. **Notation :** action stéréotypée « Informatisée ».

Tableau 3-4 : Description des extensions dans un diagramme d'activités

3.3.2 Expression des contraintes

Le langage OCL (Object Constraint Language) [OMG, 1997] a été créé spécialement pour répondre au besoin de formalisation des contraintes architecturales sur la structure des modèles UML. Il permet de définir des invariants sur les méta-classes du méta-modèle d'UML. Après l'extension du méta-modèle d'UML, plusieurs contraintes doivent être définies.

[1] Une action stéréotypée « Variation » doit avoir uniquement des transitions sortantes vers des actions stéréotypées « Variant ». Ceci est traduit par : tout arc dont la source est un point de variation admet comme cible une variante.

- ✓ **Formalisation en OCL**

Context ActivityEdge
inv: self.Source.isStereotyped ('Variation')
implies
self.Target.isStereotyped ('Variant')

[2] Une action stéréotypée « Variation » admet un stéréotype « Informatisée » (respectivement « Manuelle ») si toutes ses variantes sont stéréotypées « Informatisée » (respectivement « Manuelle »).

[3] Une action stéréotypée « Variation » n'admet pas de stéréotype « Informatisée » ou « Manuelle » si ses variantes n'ont pas le même degré d'informatisation.

- ✓ **Formalisation en OCL**

Context ActivityNode
inv: self.isStereotyped ('Variation') **and** self.Outcoming->**exists** (e| (e.Target.isStereotyped ('Manuelle') **and** self.Outcoming->**exists** (e| (e.Target.isStereotyped ('Informatisée')
implies
not self.isStereotyped ('Manuelle') **and not** self.isStereotyped('Informatisée')

[4] Une action fixe est une action non stéréotypée par « Variation » et « Variant » et son flot d'entrée est obligatoire.

[5] Une action fixe qui a une transition vers une action de type « Variation » dont le min=0, doit avoir au moins une transition vers une autre action fixe ou vers une action stéréotypée « Variation » dont le min=1.

- ✓ **Formalisation en OCL**

Context ActivityEdge
inv: self.Target.isStereotyped ('Variation') **and** self.Target.min=0
implies
self.Source.Outcoming->**exists** (e| (**not** e.Target.isStereotyped ('Variation') **and not** e.Target.isStereotyped ('Variant')) **or** ((e.Target.isStereotyped ('Variation') **and** e.Target.min=1)))

4 Conclusion

Dans ce chapitre, nous avons présenté en premier lieu la description des concepts de base nécessaires pour la définition d'un CM. Nous avons tout d'abord défini la manière dont les connaissances d'un SI ou d'un domaine peuvent être identifiées pour être ensuite formalisées, structurées et mises à la disposition des concepteurs de SI à base de composants. Le principal résultat concerne la nature de la solution encapsulée dans un CM. Ainsi, nous avons considéré qu'un CM potentiellement réutilisable doit être exprimé sous la forme d'un CM processus (CMP) qui utilise des CM entité (CME).

Dans la deuxième partie de ce chapitre, nous avons présenté nos propositions en terme de spécification de la solution offerte par un CMP donné. Cette dernière est composée de quatre vues de développement organisées autour de l'expression de la variabilité. Dans ce chapitre, le point d'intérêt est mis sur la vue métier qui définit l'aspect organisationnel du CMP. Les autres vues sont déduites de la vue métier matérialisant ainsi les propriétés informatisées du CMP. Il nous faut désormais proposer un processus d'ingénierie de composants capable de produire des CMP exprimés selon ce modèle. Ceci fait l'objet du chapitre suivant.

Chapitre 4 : Ingénierie de composant métier processus supportant la variabilité

L'ingénierie des composants métier nécessite un cadre permettant de représenter la structure fondamentale du composant, en offrant des mécanismes pour son identification et sa description. L'identification aide à mieux comprendre la situation pour laquelle le composant peut être une solution appropriée pour résoudre la totalité ou une partie d'un problème. La description explicite la solution du composant. Ainsi, indépendamment de son niveau d'abstraction, le composant doit fournir toutes les informations nécessaires pour comprendre sa fonctionnalité.

Ce chapitre est consacré en premier lieu à la description du processus de spécification d'un CMP (§ 1) dont le modèle a été présenté dans le chapitre précédent. Nous nous appuyons, dans la suite de ce chapitre, sur l'exemple de gestion d'allocation de ressources (présenté en totalité en annexe A) pour montrer comment la démarche nous permet de spécifier des CMP réutilisables dans plusieurs SI (§ 2).

Notre proposition se démarque des autres approches par une identification et une spécification de CM de nature processus. De plus, elle propose une solution multi-vues du composant et y exprime la variabilité dans chacune des vues (§3). La vue métier ayant été décrite dans le chapitre précédent, ce chapitre détaille principalement les trois autres vues du CMP en tenant compte de la cohérence inter-vues. En outre, l'expression de la variabilité a mené vers la proposition d'un profil UML pour les CM supportant la variabilité (§4).

1 Ingénierie de composants métier : concepts de base

1.1 Approches d'ingénierie de composants

L'ingénierie des CM consiste à spécifier de nouveaux composants. Elle recouvre les tâches d'identification, de spécification et d'organisation de composants. F. Semmak [Semmak, 1998] définit diverses techniques d'identification de composants, appartenant à deux grandes approches. La première approche est la **rétro-ingénierie**, elle vise à abstraire des éléments réutilisables à partir de l'analyse de produits existants. Cette approche permet d'évaluer les similarités et les différences entre produits par *analyse de similarités* ou par *recherche d'analogie*, afin d'identifier de manière systématique des éléments candidats à la réutilisation. La deuxième approche est **l'analyse de domaine** qui produit deux grandes tendances [Cauvet *et al.*, 2005] : 1) les méthodes *orientées familles d'application*s ou *lignes de produits* [Pohl *et al.*, 2005], [Griss, 2001], [Bachmann *et al.*, 2003], qui considèrent une famille de systèmes ayant des fonctionnalités similaires dans l'objectif de produire un modèle de domaine réutilisable qui exprime les ressemblances (parties communes) et les différences (parties spécifiques) entre les systèmes considérés. 2) les méthodes *orientées problèmes* [Kang *et al.*, 1990], [Ramadour, 2001], [Jacob *et al.*, 2000] où un domaine est défini par un ensemble de problèmes pour lesquels différentes solutions sont envisageables. Les modèles de domaine obtenus sont donc orientés problèmes et non plus solutions [Grosz *et al*., 1996], contrairement à la première démarche. L'optimisation des réservations ou la satisfaction des attentes des clients spécifiques au domaine de réservation de places d'avions sont considérées, par exemple, comme problèmes spécifiques à un domaine particulier.

D'un autre côté, la réutilisation des composants est d'autant plus efficace si ces composants sont associés à une spécification pertinente. Cela revient à distinguer deux types de connaissances : les connaissances effectivement réutilisables (la solution du composant) et celles requises pour une utilisation correcte de celui-ci (le savoir-faire de la réutilisation) [Cauvet *et al.*, 2001].

En outre, les composants doivent être stockés dans un environnement pour y faciliter l'accès de telle manière qu'ils puissent être intégrés dans un système pour accomplir leurs objectifs.

1.2 Ingénierie de CMP multi-vues supportant la variabilité

Dans le chapitre précédent, nous avons proposé un modèle de CM centré sur trois points forts : 1) spécifier des CM de type processus appelés Composant Métier Processus (CMP), 2) spécifier des CMP avec une solution complète et 3) exprimer la variabilité sur la solution des CMP. La prise en compte de ces trois points forts introduit de nouvelles préoccupations à considérer lors de l'ingénierie du CMP.

- *Identification de l'information réutilisable* : il s'agit d'identifier le processus métier pour en créer une abstraction réutilisable. Cette activité est assurée par l'analyse de PM ayant des fonctionnalités similaires pour créer un PM qui exprime les ressemblances.
- *Identification de la variabilité* : il s'agit de définir ce qui diffère dans un PM d'un SI à un autre. L'étude de la variabilité met en évidence les parties spécifiques de chaque PM liées à son SI d'origine.

- *Modélisation multi-vues* : il s'agit de concevoir le CMP sur quatre vues de développement en tenant compte de la cohérence inter-vues.
- *Représentation de la variabilité* : il s'agit de spécifier explicitement la variabilité à travers l'introduction des points de variation et des variantes sur chaque vue de développement en tenant compte de la cohérence inter-vues.

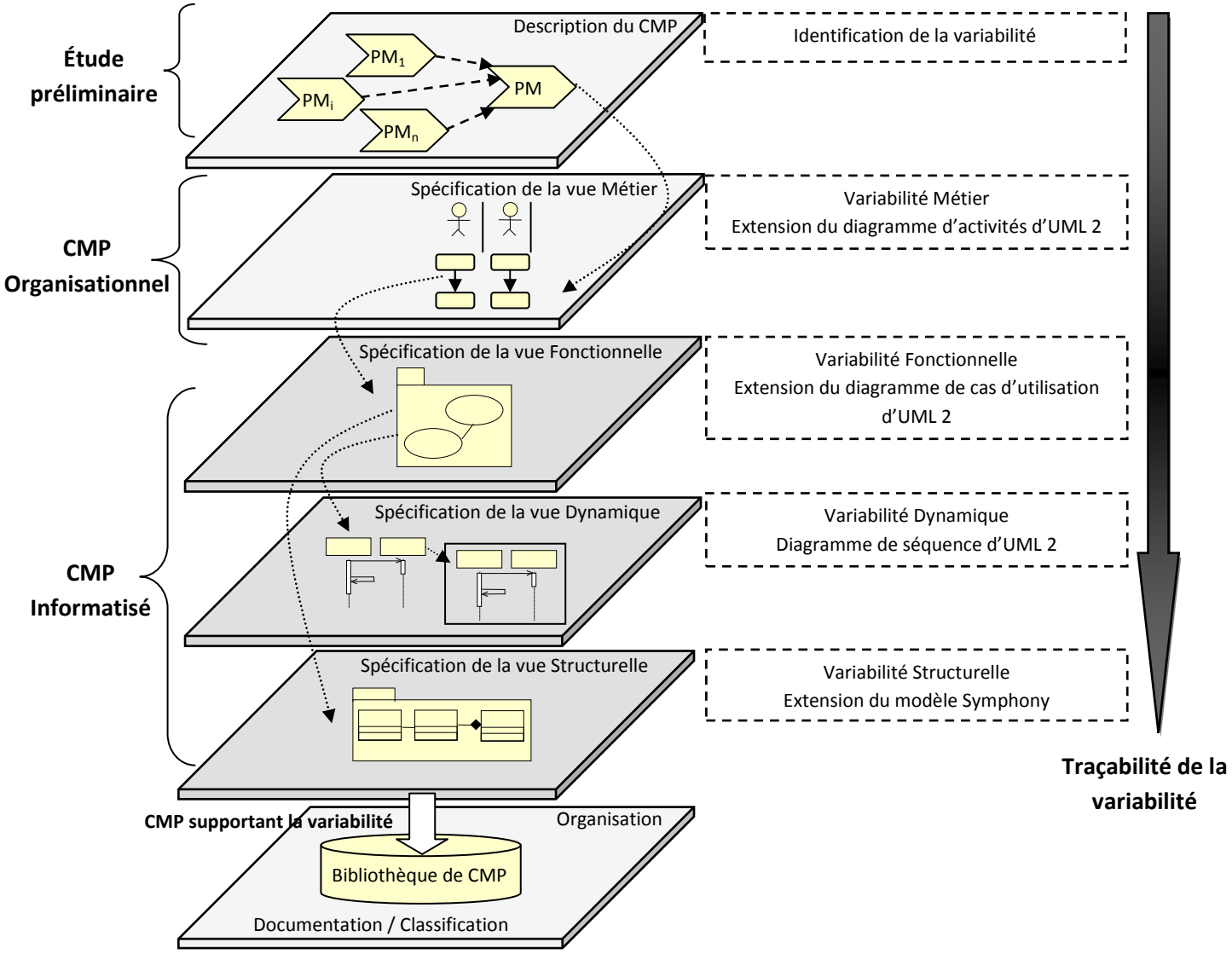


Figure 4-1 : Représentation en couches du processus de spécification des CMP

Le Tableau 4-1 illustre l'ensemble des phases constituant le processus de spécification d'un CMP représenté dans la Figure 4-1. La première phase (*Description du CMP*) constitue une étape d'étude préliminaire qui vise à décrire le CMP et à identifier la variabilité. La deuxième phase (*Spécification de la vue métier*) a pour but d'identifier et de modéliser l'aspect organisationnel du CMP. Les trois phases suivantes (*Spécification de la vue fonctionnelle, Spécification de la vue dynamique, Spécification de la vue structurelle*) identifient les vues constituant les propriétés informatisées du CMP. La dernière phase (*Organisation*) est consacrée à la documentation et à l'organisation du CMP dans une bibliothèque de composants. Durant l'ensemble de ces phases, la traçabilité des concepts de variabilité est

maintenue. Les cinq premières phases de spécification sont illustrées dans la section 3, tandis que la phase d'organisation sera traitée dans le chapitre suivant.

Phase	**Artefact produit**
Description du CMP	Fiche descriptive détaillée du CMP : diagramme de cas d'utilisation métier du CMP générique, scénario détaillé du CMP supportant la variabilité.
Spécification de la vue métier	Vue métier du CMP représentée par un diagramme d'activités supportant la variabilité.
Spécification de la vue fonctionnelle	Vue fonctionnelle du CMP représentée par un diagramme de cas d'utilisation supportant la variabilité. Une fiche descriptive par cas d'utilisation.
Spécification de la vue dynamique	Vue dynamique du CMP représentée par un ensemble de diagrammes de séquence.
Spécification de la vue structurelle	Vue structurelle du CMP représentée par des fragments structurels selon le modèle Symphony.

Tableau 4-1 : Description des phases du processus et des artefacts produits

2 Étude de cas : Gestion Allocation de Ressources

Pour illustrer le processus de spécification d'un CMP, nous utilisons l'exemple du processus de gestion d'allocation de ressources. Une ressource est une entité dont la particularité est d'être partagée, requise et disponible pour la gestion dont elle est objet. Plusieurs entités peuvent jouer le rôle d'une ressource. Par exemple, dans le contexte d'une bibliothèque, un livre est une ressource pour le processus « Emprunt ». Dans une agence de location de voitures, une voiture est une ressource pour le processus « Location ».

L'étude des différentes manières de gérer les ressources permet de définir des catégories de ressources. Certaines sont des ressources consommables puisque leur existence est liée à un événement donné (ex. un billet d'avion, un billet de spectacle…), et d'autres ne sont pas consommables car elles sont allouées pour une période précise, et exigent donc un processus de retour qui libère la ressource pour d'autres demandes (ex. livre, voiture…). Il existe aussi des ressources dont la demande passe par une réservation avant l'allocation. Dans ce chapitre, pour illustrer le processus d'ingénierie d'un CMP de gestion d'allocation de ressources, nous nous appuyons sur cinq PM simplifiés d'allocation de ressources issus de SI différents. L'utilisation de ces PM nous servira à identifier la variabilité. Le Tableau 4-2 illustre l'ensemble de ces PM.

Ressource	**PM**	**SI**
Livre	PM_1 : Gestion Emprunt Livre	Bibliothèque
Voiture	PM_2 : Gestion Allocation Voiture	Agence de voiture
Chambre	PM_3 : Gestion Allocation Chambre d'Hôtel	Hôtel
Billet d'avion	PM_4 : Gestion Allocation Billet d'Avion	Compagnie aérienne
Billet de spectacle	PM_5 : Gestion Allocation Billet Spectacle	Agence artistique

Tableau 4-2 : Liste des PM utilisés

3 Processus de spécification de CMP

3.1 Phase 1 : Description du CMP

3.1.1 Description

Les étapes de cette phase sont les suivantes (Cf. Figure 4-2) :

- *Description du processus métier* : Il s'agit en premier lieu de modéliser les éléments et mécanismes principaux de chaque PM lié à son SI d'origine. La fiche descriptive de chaque PM produite à l'issue de cette phase comprend : 1) le modèle de *cas d'utilisation métier* UML [Jacobson *et al.*, 1999] (besoins clients) de chaque PM, 2) une description en langage naturel du scénario de chaque PM et 3) un diagramme d'activités illustrant l'aspect organisationnel de chaque PM. En se basant sur le fait qu'un PM est modélisé par un cas d'utilisation métier, et qu'un cas d'utilisation concerne un ou plusieurs acteurs externes dont l'acteur principal attend un résultat, chaque PM est décomposé en sous-PM dont l'événement déclencheur provient généralement de l'acteur principal.
- *Analyse du CMP* : il s'agit premièrement de la reformulation des besoins de l'ensemble des PM utilisés en un CMP générique réutilisable en analysant la similarité sémantique entre ces PM. La fiche descriptive du CMP produite à l'issu de cette étape comprend : 1) le modèle de cas d'utilisation métier modélisant les propriétés génériques du CMP et 2) une description en langage naturel du scénario du CMP de ce modèle. Ensuite, on procède à l'identification de la variabilité entre les différents PM en définissant les points de variation et les variantes ainsi que des contraintes de dépendance liant ces variantes.

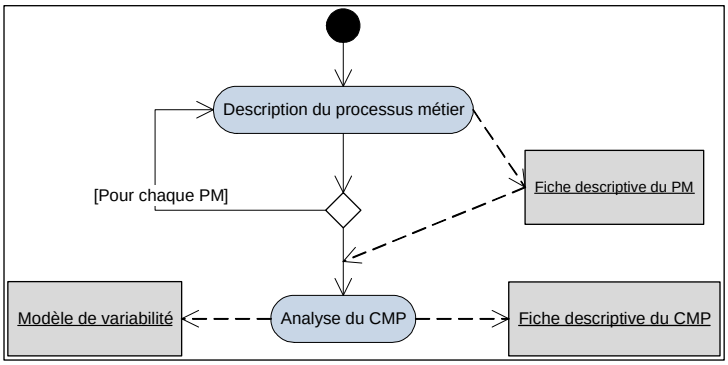
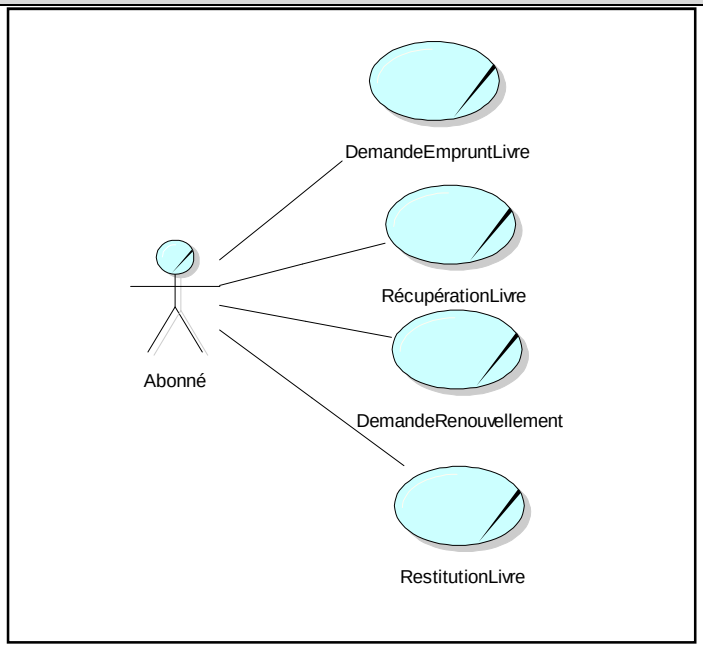


Figure 4-2 : Enchaînement des étapes de la phase description du CMP

3.1.2 Exemple d'illustration

a) Description du processus métier : fiche descriptive de chaque PM

Pour garantir une cohérence de spécification entre le diagramme de cas d'utilisation métier et le modèle organisationnel de chaque PM, le nom de chaque cas d'utilisation métier correspond au nom du sous-PM et au nom de l'activité ou de l'événement qui déclenche le sous-PM.

<table>
<tr><th colspan="2">PM $_1$: Gestion Emprunt Livre</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un abonné (acteur principal) émet une demande d’emprunt d’un livre au bibliothécaire en remplissant un formulaire électronique.

Le bibliothécaire (acteur secondaire) traite la demande :

1. Identifie l’abonné par le biais du code abonné.
2. Vérifie la disponibilité du livre.
3. Informe l’abonné de la disponibilité du livre et enregistre la demande.

- L’abonné se présente à la bibliothèque pour récupérer le livre.

Le bibliothécaire :

1. Identifie l’abonné par le biais du code demande.
2. Crée l’emprunt.
3. Remet le livre à l’abonné.

- L’abonné peut émettre une demande de renouvellement de l’emprunt.

Le bibliothécaire :

1. Identifie l’abonné par le biais du code d’emprunt.
2. Vérifie l’autorisation de l’abonné à renouveler l’emprunt.
3. Met à jour l’emprunt.

- L’abonné se présente à la bibliothèque pour rendre le livre.

Le bibliothécaire enregistre le retour du livre en mettant à jour l’état du livre.</td></tr>
</table>

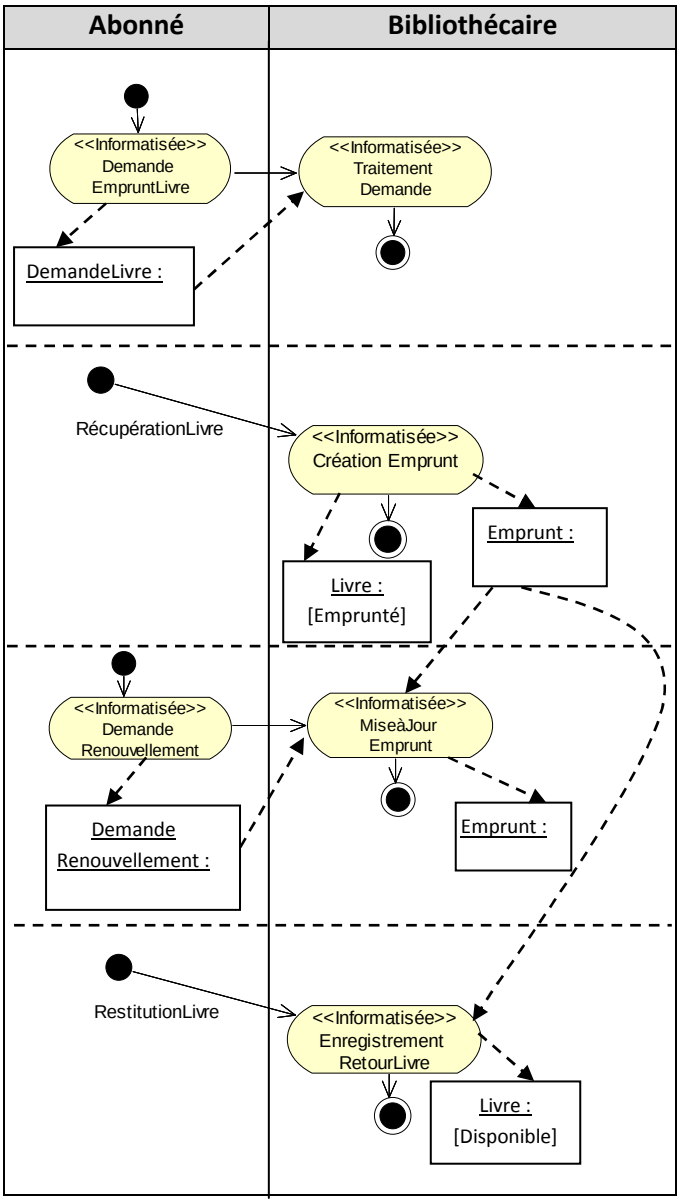
Modèle organisationnel

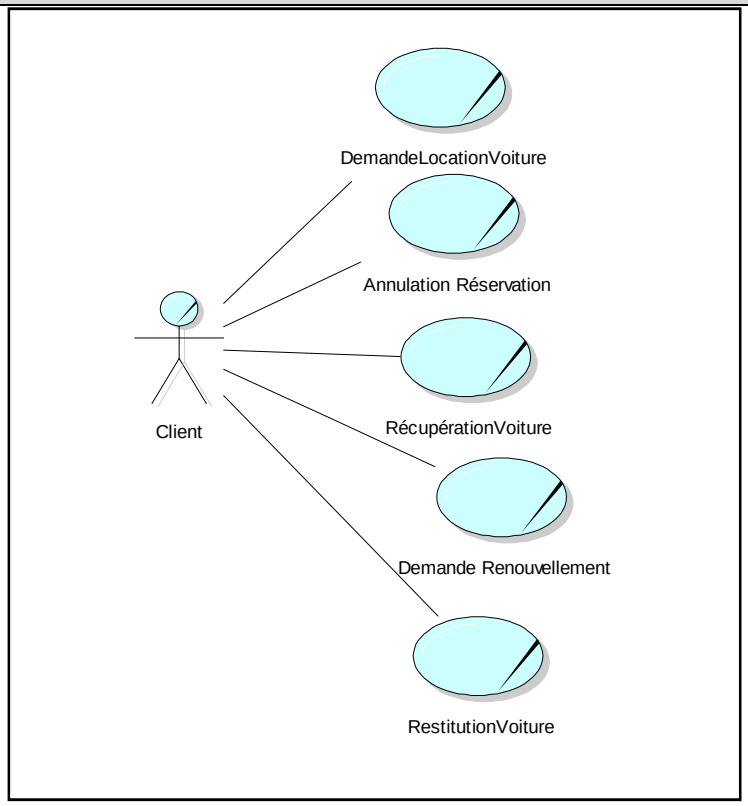
« Demande Emprunt Livre »

« Récupération Livre »

« Demande Renouvellement »

« Restitution Livre »



<table>
<tr><th colspan="2">PM ₂ : Gestion Allocation Voiture</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) émet une demande de location d’une voiture en remplissant un formulaire électronique.

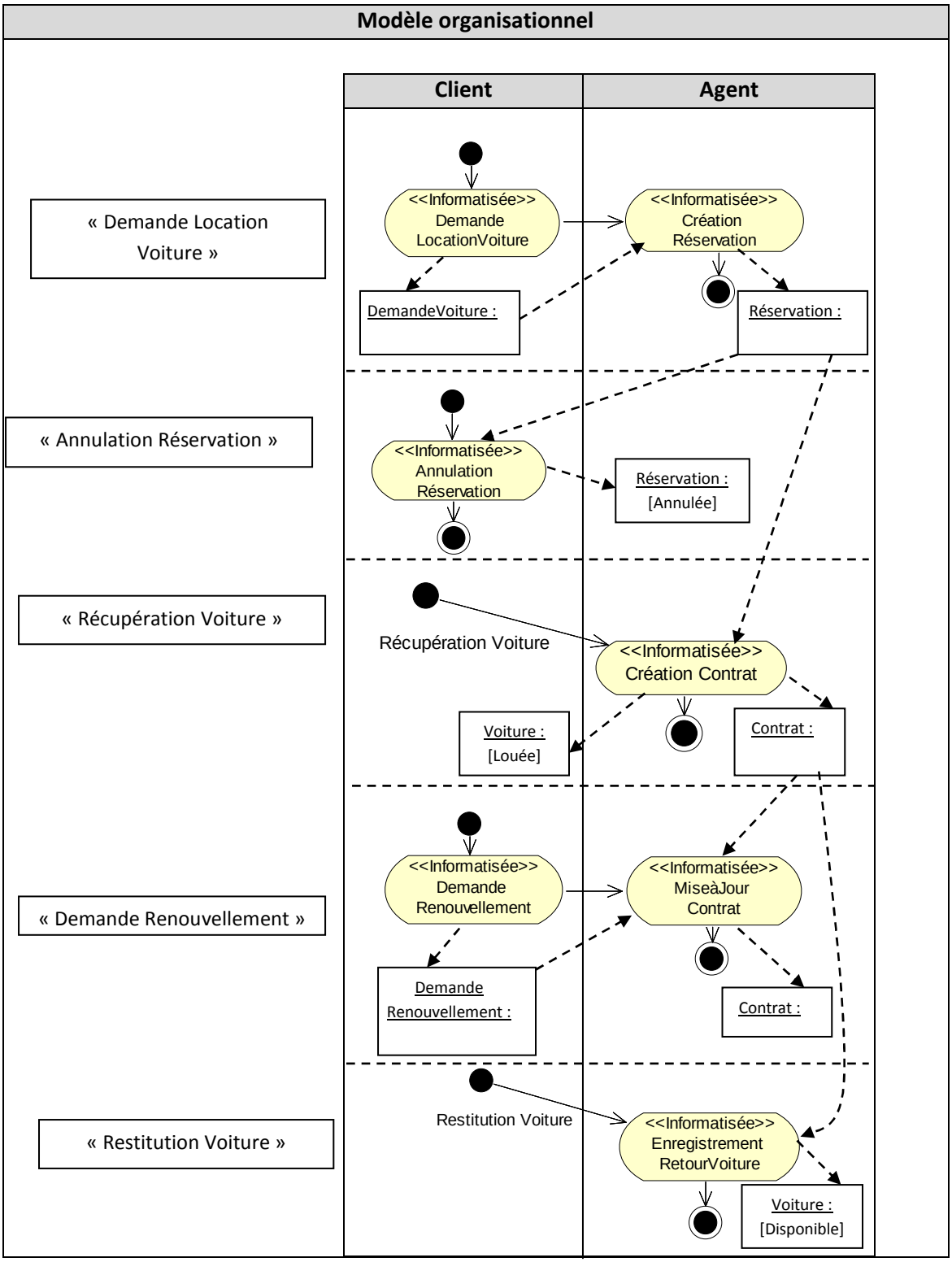
L’agent (acteur secondaire) :
1. Vérifie la disponibilité d’une voiture.
2. Crée une réservation.
3. Communique au client le code de réservation.

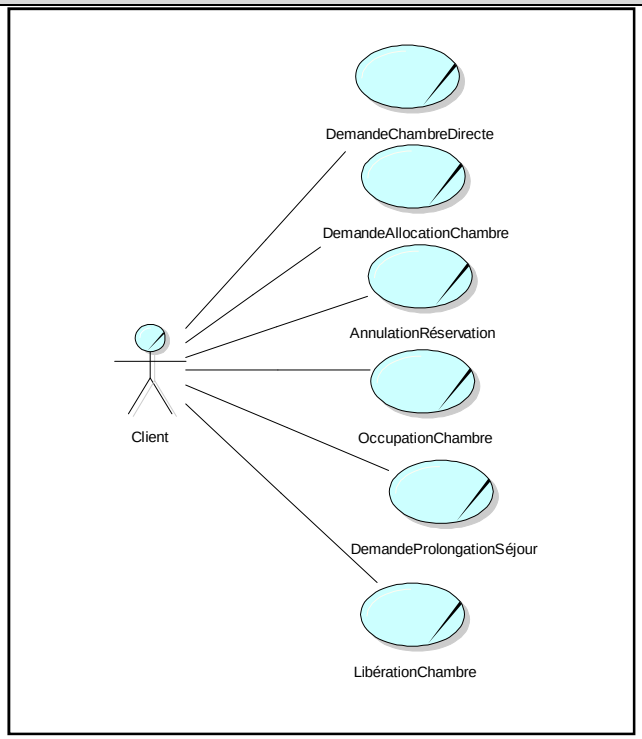
- Le client peut annuler sa réservation via un formulaire électronique.
- Le client se présente à l’agence de voiture pour récupérer la voiture.
L’agent :
1. Identifie le client par le biais du code réservation.
2. Crée le contrat de location.
3. Remet au client la voiture.

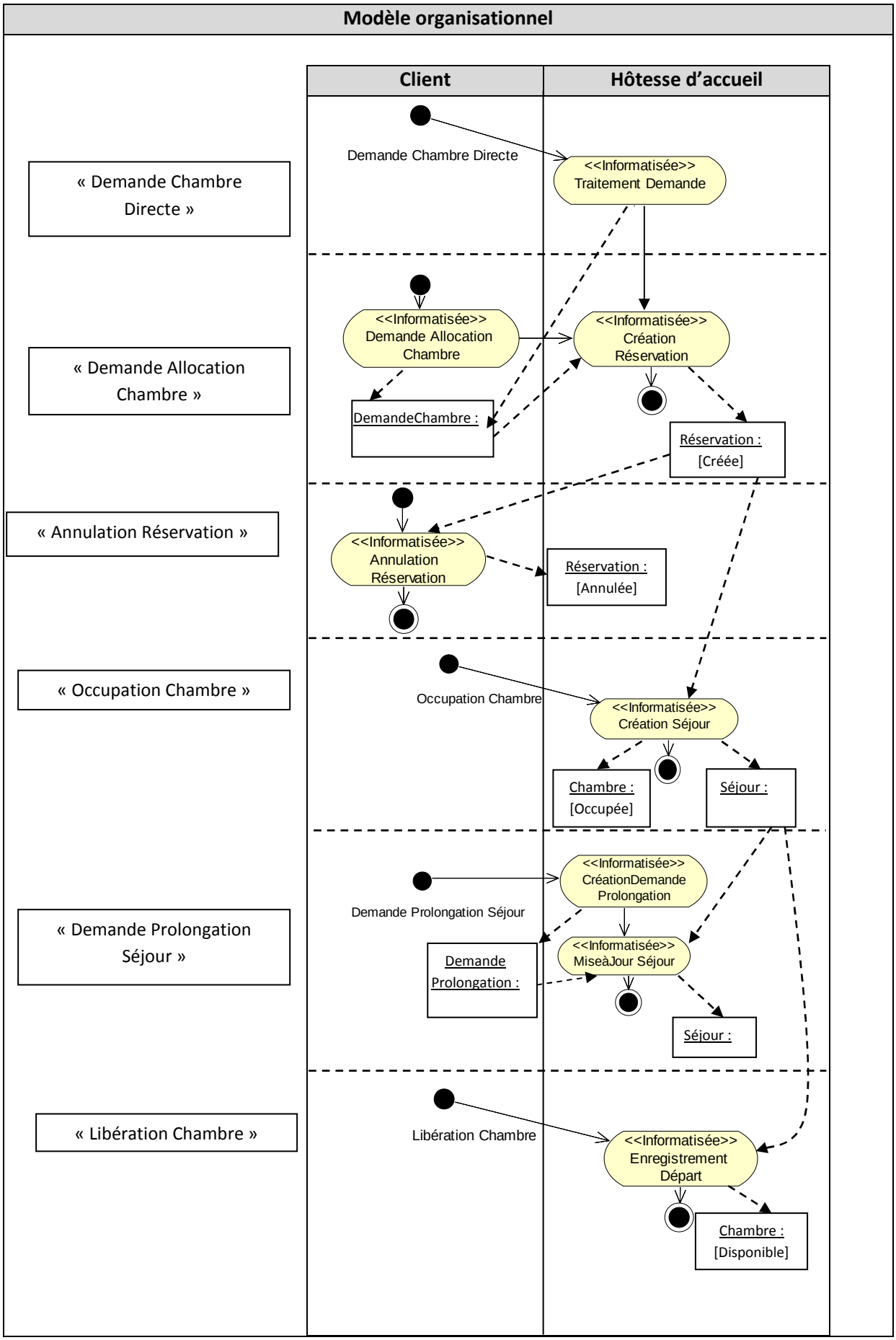
- Le client peut demander le renouvellement du contrat.
L’agent :
1. Identifie le client par le biais du code contrat.
2. Vérifie l’autorisation du client à renouveler le contrat.
3. Met à jour le contrat.

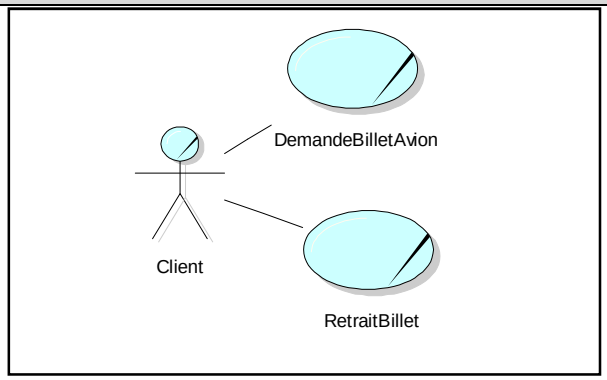
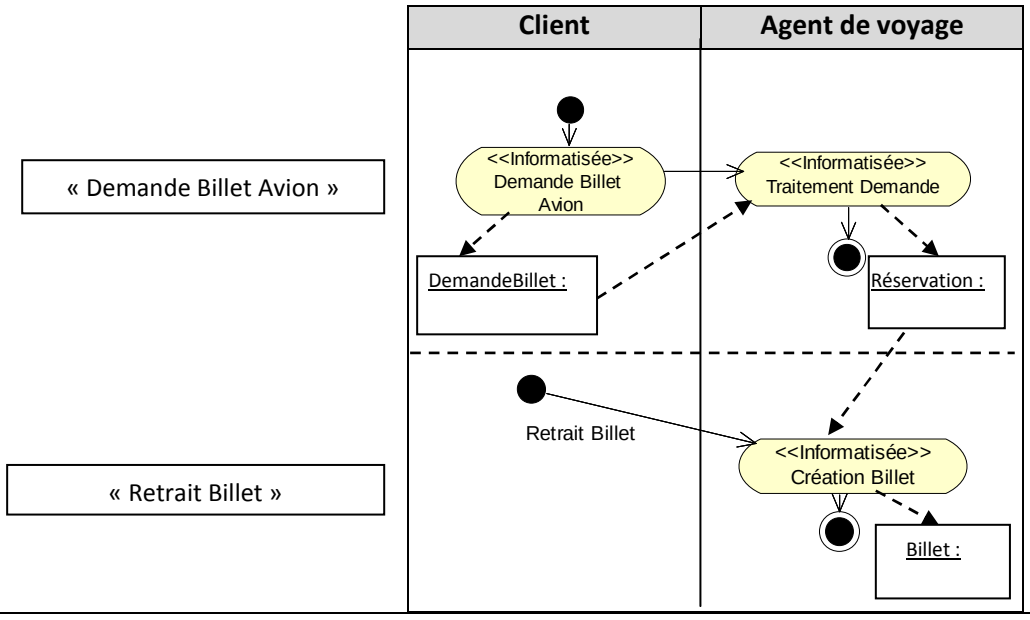
- Le client se présente à l’agence de voiture pour rendre la voiture.

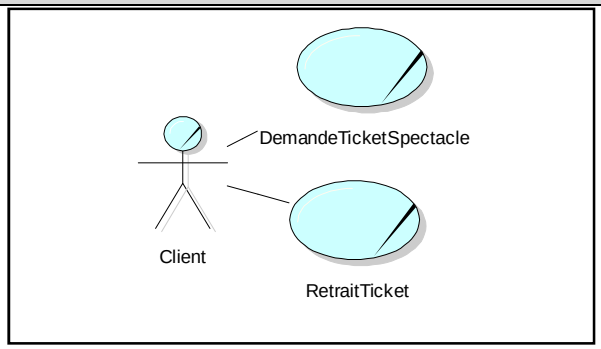
L’agent enregistre le retour de la voiture en mettant à jour l’état de la voiture.</td></tr>
</table>


Modèle organisationnel
Client
Agent
« Demande Location Voiture »
<<Informatisée>> Demande LocationVoiture
<<Informatisée>> Création Réservation
DemandeVoiture :
Réservation :
« Annulation Réservation »
<<Informatisée>> Annulation Réservation
Réservation : [Annulée]
« Récupération Voiture »
Récupération Voiture
<<Informatisée>> Création Contrat
Voiture : [Louée]
Contrat :
« Demande Renouvellement »
<<Informatisée>> Demande Renouvellement
<<Informatisée>> MiseàJour Contrat
Demande Renouvellement :
Contrat :
« Restitution Voiture »
Restitution Voiture
<<Informatisée>> Enregistrement RetourVoiture
Voiture : [Disponible]

<table>
<tr><th colspan="2">PM ₃ : Gestion Allocation Chambre d'hôtel</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) se présente à un hôtel et demande une chambre.
L'hôtesse d'accueil (acteur secondaire) :
1. Crée la demande.
2. Vérifie la disponibilité d'une chambre et enregistre la réservation.
3. Crée le séjour et remet la clé de la chambre au client.
- Un client émet une demande de réservation de chambre d'hôtel en remplissant un formulaire électronique.
L'hôtesse d'accueil :
1. Vérifie la disponibilité d'une chambre.
2. Crée une réservation.
3. Communique au client le code réservation.
- Le client peut annuler sa réservation via un formulaire électronique.
- Le client se présente à l'hôtel pour récupérer la clé de sa chambre.
L'hôtesse d'accueil :
1. Identifie le client par le biais du code de la réservation.
2. Crée le séjour et remet la clé de la chambre au client.
- Le client peut demander une prolongation du séjour.
L'hôtesse d'accueil :
1. Identifie le client par le biais du code séjour.
2. Crée la demande de prolongation du séjour.
3. Vérifie la disponibilité de la chambre ou d'une autre à la date demandée.
4. Met à jour l'ancien séjour.
- Le client libère la chambre.
L'hôtesse d'accueil met à jour l'état de la chambre.</td></tr>
</table>

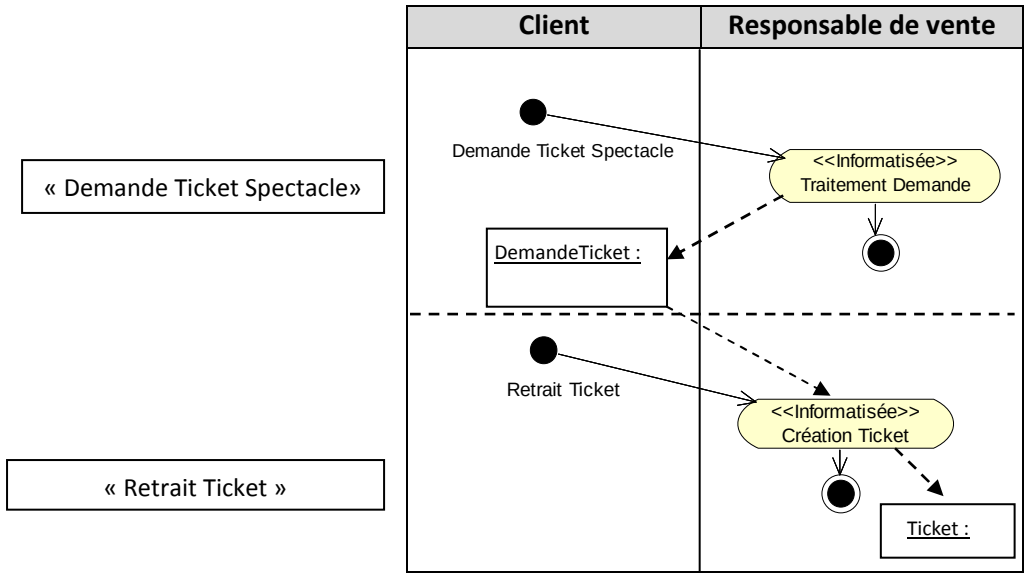

Modèle organisationnel
Client
Hôtesse d'accueil
« Demande Chambre Directe »
Demande Chambre Directe
<<Informatisée>> Traitement Demande
« Demande Allocation Chambre »
<<Informatisée>> Demande Allocation Chambre
<<Informatisée>> Création Réservation
DemandeChambre :
Réservation : [Créée]
« Annulation Réservation »
<<Informatisée>> Annulation Réservation
Réservation : [Annulée]
« Occupation Chambre »
Occupation Chambre
<<Informatisée>> Création Séjour
Chambre : [Occupée]
Séjour :
« Demande Prolongation Séjour »
Demande Prolongation Séjour
<<Informatisée>> CréationDemande Prolongation
Demande Prolongation :
<<Informatisée>> MiseàJour Séjour
Séjour :
« Libération Chambre »
Libération Chambre
<<Informatisée>> Enregistrement Départ
Chambre : [Disponible]

<table>
<tr><th colspan="2">PM ₄ : Gestion Allocation Billet Avion</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) émet une demande d’achat d’un billet d’avion à une agence de voyage Lowcost en remplissant un formulaire électronique.
L’agent de voyage (acteur secondaire) :
1. Vérifie la disponibilité d’un billet.
2. Crée une réservation.
3. Communique au client le code de sa réservation en mentionnant l’impossibilité d’annulation.
- Le client se présente à l’agence de voyage pour payer et retirer son billet.
L’agent de voyage :
1. Identifie le client par le biais du code réservation.
2. Crée le billet d’avion.
3. Imprime le billet et le remet au client.</td></tr>
<tr><th colspan="2">Modèle organisationnel</th></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

<table>
<tr><th colspan="2">PM 5 : Gestion Allocation Billet Spectacle</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) se présente à une agence artistique et demande l’achat d’un billet de spectacle au responsable de vente.

Le responsable de vente (acteur secondaire) :

1. Vérifie la disponibilité d’un ticket.
2. Informe le client de la disponibilité du ticket.
- Le client confirme sa demande en demandant le retrait du ticket.
Le responsable de vente :

1. Crée le ticket du spectacle.
2. Imprime le ticket et le remet au client.</td></tr>
<tr><th colspan="2">Modèle organisationnel</th></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

b) Analyse du CMP

✓ Fiche descriptive du CMP

Dans le Tableau 4-3, nous étudions la similarité sémantique entre les cas d’utilisation métier, les acteurs, les activités /événements ainsi que les objets de chaque PM considéré en créant des cas d’utilisation métier, des acteurs, des activités / événements et des objets abstraits utilisés dans le CMP. L’abstraction des cas d’utilisation métier étant identique à celle des activités déclencheurs de chaque sous-PM, nous ne montrons que l’abstraction de ces activités (Activités soulignées dans le tableau).

Acteurs /Activités/Objets des PM	**Acteurs/ Activités/Objets - Abstraction -**
Acteurs	
Abonné	Client
Client	
Bibliothécaire	Allocation Manager
Agent	
Hôtesse d'accueil	
Agent de voyage	
Responsable de Vente	
Activités /événements	
Demande Chambre Directe	Demande Allocation Directe
Demande Ticket Spectacle	
Demande Emprunt Livre	Demande Allocation
Demande Location Voiture	
Demande Allocation Chambre	
Demande Billet Avion	
Traitement Demande	Traitement Sans Réservation
Création Réservation	Traitement Avec Réservation
Annulation Réservation	Annulation Réservation
Récupération Livre	Récupération Ressource
Récupération Voiture	
Occupation Chambre	
Retrait Billet	
Retrait Ticket	
Création Emprunt	Création Allocation
Création Contrat	
Création Séjour	
Création Billet	
Création Ticket	
Demande Renouvellement	Demande Renouvellement
Création Demande Prolongation	
Demande Prolongation Séjour	Demande Renouvellement Direct
Mise à Jour Emprunt	Mise à Jour Allocation
Mise à Jour Contrat	
Mise à Jour Séjour	
Restitution Livre	Restitution Ressource
Restitution Voiture	
Libération Chambre	
Enregistrement Retour Livre	Enregistrement Retour Ressource
Enregistrement Retour Voiture	
Enregistrement Départ	
Objets	
Demande Livre	Demande Allocation
Demande Voiture	
Demande Chambre	
Demande Billet	

Demande Ticket	
Réservation	Réservation
Emprunt	Allocation
Contrat	
Séjour	
Demande Renouvellement	Renouvellement
Demande Prolongation	
Livre	Ressource
Voiture	
Chambre	
Billet	
Ticket	

Tableau 4-3 : Abstraction et analyse de similarité entre les PM

Après la phase d'abstraction et d'analyse de similarité des PM étudiés, nous extrayons les cas d'utilisation métier (Cf. Figure 4-3). Nous utilisons une notation qui sera détaillée dans la section 3.3. Cette notation nous permet de représenter la liste préliminaire des variations identifiées correspondant aux activités ou événements déclencheurs (soulignés dans le Tableau 4-3). Les cas d'utilisation grisés correspondant à des points de variation des différentes variantes.

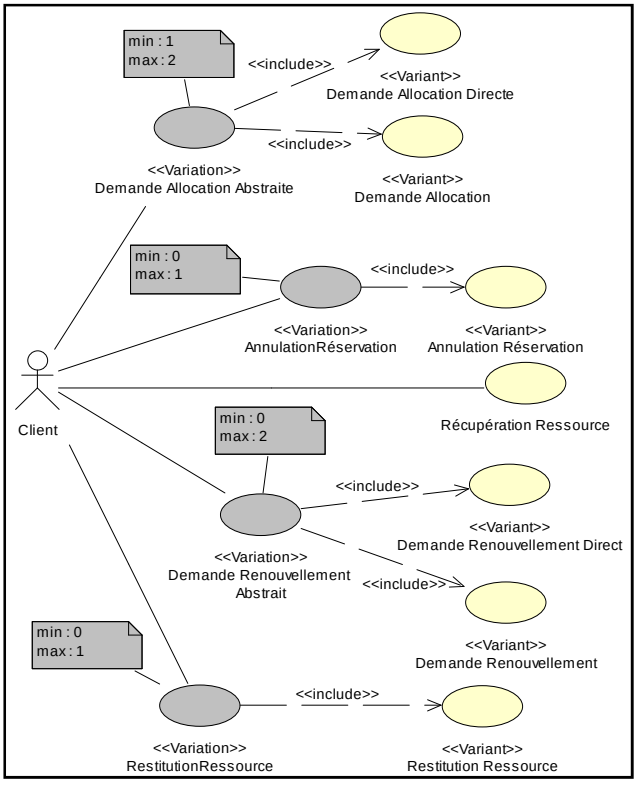


Figure 4-3 : Modèle métier du PM « Gestion Allocation Ressource »

À partir du modèle métier, nous décrivons le scénario de chaque cas d'utilisation, en mettant en évidence les points de variation discriminant l'ensemble des PM considérés. Nous distinguons deux types de variation : 1) des variations identifiées à

partir des cas d'utilisation métier, et 2) des variations déduites de ces cas d'utilisation représentées dans le scénario par des pointillés.

<table>
<tr><td rowspan="3">Scénario</td><td>
- Variation : Demande Allocation abstraite
• Variante : Demande Allocation Directe
- Le client (acteur principal) se présente directement pour demander sa ressource.
- L'Allocation Manager (acteur secondaire) crée la demande d'allocation.
• Variante : Demande Allocation
- Le client crée la demande d'allocation via un formulaire électronique.
Variation : Traitement Demande Allocation
L'Allocation Manager traite la demande :
1. Identifie le client par le biais du code demande.
2. Vérifie la disponibilité de la ressource.
• Variante : Traitement Avec Réservation
- Crée une réservation et communique au client le code de réservation.
• Variante : Traitement Sans Réservation
- Confirme au client la disponibilité de la ressource à la date demandée.
</td></tr>
<tr><td>
- Variation : Annulation Réservation
• Variante : Annulation Réservation
- Le client peut annuler sa réservation en la supprimant.
</td></tr>
<tr><td>
- Le client se présente pour récupérer la ressource.
L'Allocation Manager :
1. Variation : Allocation avec Réservation
• Variante : Vérification Réservation
- Vérifie le code de réservation.
2. Crée l'allocation.
3. Variation : Modification État Ressource
• Variante : Suppression Ressource
- Supprime la ressource (ressource consommable).
• Variante : Mise à jour Ressource
- Met à jour l'état de la ressource (ressource non consommable).
</td></tr>
</table>

	- Variation : Demande Renouvellement Abstrait • **Variante** : *Demande Renouvellement Direct* - Le *client* se présente directement pour demander le renouvellement de l'allocation. - L'*Allocation Manager* crée la demande de renouvellement. • **Variante** : *Demande Renouvellement* - Le *client* peut créer une demande de renouvellement d'une allocation. - L'*Allocation Manager* : 1. Identifie l'allocation. 2. Vérifie la disponibilité de la ressource. 3. Met à jour l'allocation.
	- Variation : Restitution Ressource • **Variante** : *Restitution Ressource* - Le *client* restitue la ressource. - L'*Allocation Manager* : 1. Récupère la ressource. 2. Enregistre le retour de la ressource.

✓ Modèle de variabilité

Le Tableau 4-4 illustre la variabilité identifiée dans le scénario. Le Tableau 4-5 représente les contraintes de dépendance liant certaines variantes.

Points de variation	**Type**	**Cardinalités**	**Variantes**
PV $_1$: Demande Allocation Abstraite	Degré d'informatisation / Ensemble d'alternatives	Min : 1 Max : 2	V $_{11}$: Demande Allocation / Informatisée
			V $_{12}$: Demande Allocation Directe / Manuelle
PV $_2$: Traitement Demande Allocation	Alternative	Min : 1 Max : 1	V $_{21}$: Traitement Avec Réservation
			V $_{22}$: Traitement Sans Réservation
PV $_3$: Annulation Réservation	Option	Min : 0 Max : 1	V $_{31}$: Annulation Réservation
PV $_4$: Allocation Avec Réservation	Option	Min : 0 Max : 1	V_{41} : Vérification Réservation
PV_5 : Modification État Ressource	Alternative	Min : 1 Max : 1	V $_{51}$: Mise à Jour Ressource
			V $_{52}$: Suppression Ressource
PV $_6$: Demande Renouvellement Abstrait	Degré d'informatisation / Option	Min : 0 Max : 2	V $_{61}$: Demande Renouvellement / Informatisée
			V $_{62}$: Demande Renouvellement Direct / Manuelle
PV $_7$: Restitution Ressource	Option	Min : 0 Max : 1	V $_{71}$: Restitution Ressource

Tableau 4-4 : Variabilité identifiée

	V_{11}	V_{12}	V_{21}	V_{22}	V_{31}	V_{41}	V_{51}	V_{52}	V_{61}	V_{62}	V_{71}
V_{11}											
V_{12}											
V_{21}						Exige					
V_{22}					Exclut	Exclut					
V_{31}			Exige	Exclut							
V_{41}			Exige	Exclut							
V_{51}											Exige
V_{52}									Exclut	Exclut	Exclut
V_{61}							Exige	Exclut			
V_{62}							Exige	Exclut			
V_{71}							Exige	Exclut			

Tableau 4-5 : Contraintes de dépendance

Nous rappelons que la contrainte d'exigence n'est pas symétrique. Par exemple, le choix de V_{31} (Annulation Réservation) exige le choix de V_{21} (Traitement Avec Réservation) n'implique pas que le choix de V_{21} exige le choix de V_{31}.

3.2 Phase 2 : Spécification de la vue métier

3.2.1 Description

La spécification de la vue métier comprend les étapes suivantes (Cf. Figure 4-4) :

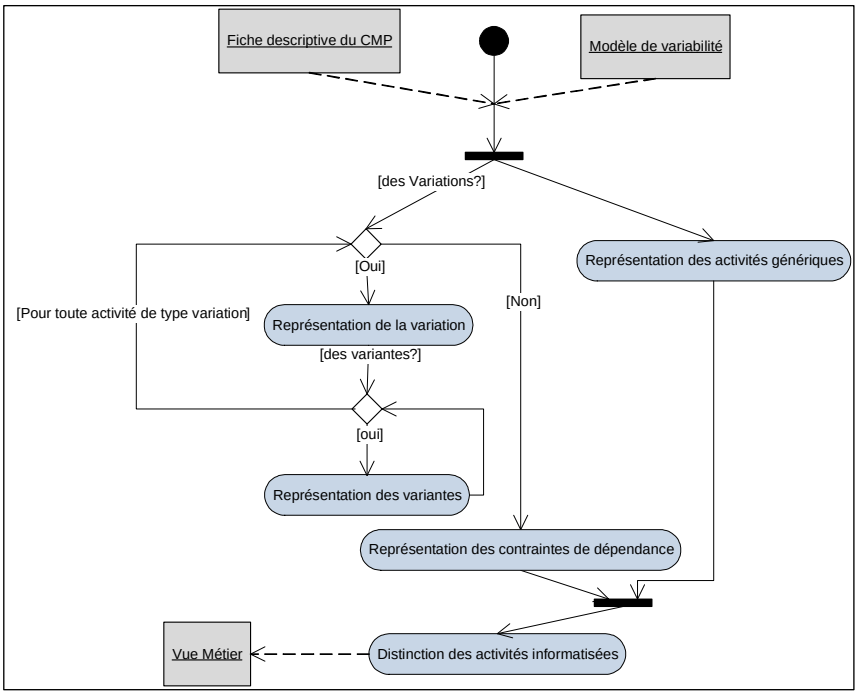


Figure 4-4 : Enchaînement des étapes de la phase spécification de la vue métier

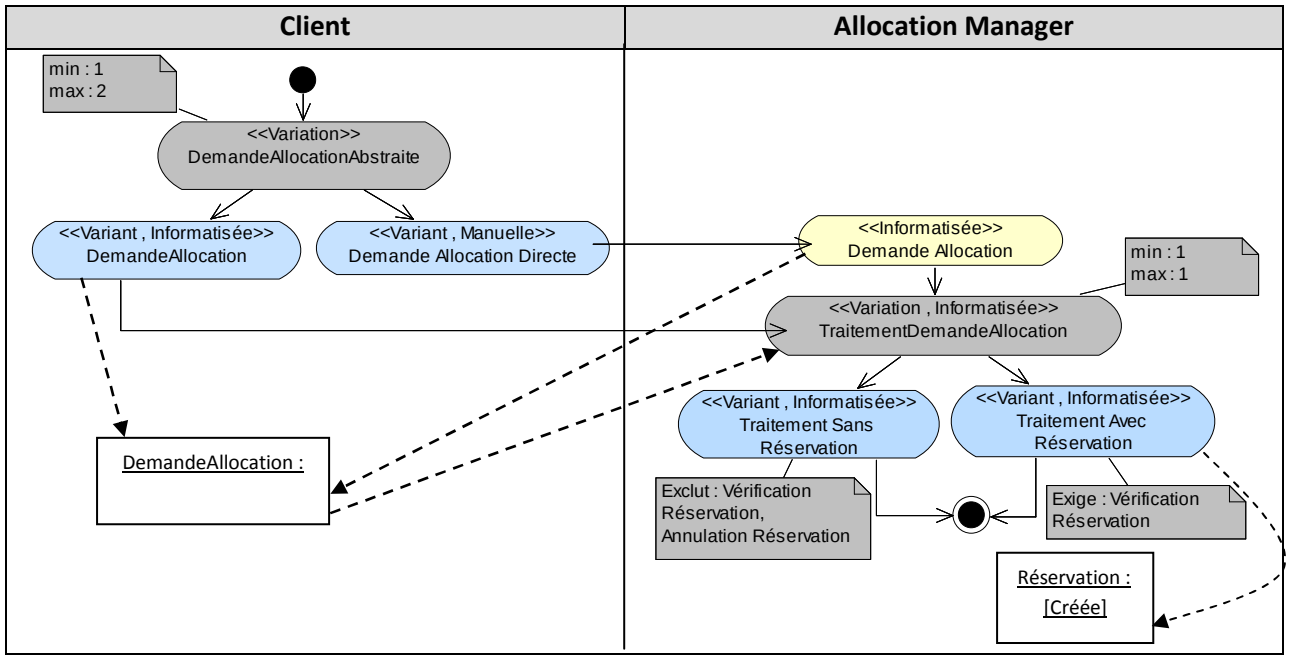
– *Représentation des activités* : il s'agit de représenter les activités génériques, variations et variantes de façon parallèle en se basant sur la fiche descriptive du CMP décrite dans la phase précédente.

– *Représentation des contraintes de dépendance* : il s'agit de représenter les contraintes de dépendances liant les variantes.
– *Distinction des activités informatisées* : il s'agit de distinguer les activités informatisées de celles manuelles.

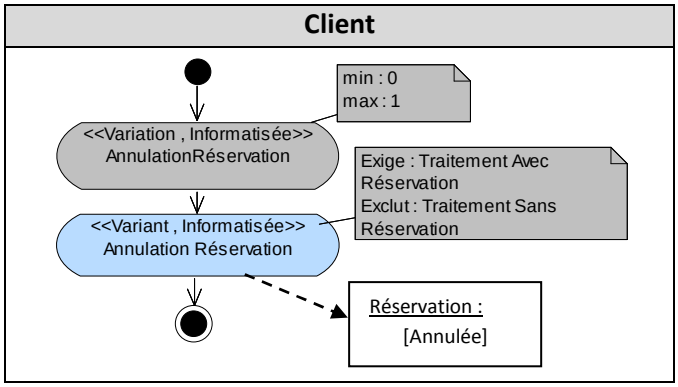
3.2.2 Exemple d'illustration

La Figure 4-5 illustre la vue métier du CMP « Gestion Allocation Ressource » modélisée par un diagramme d'activités supportant la variabilité. Nous identifions cinq sous-CMP : « Demande Allocation Abstraite», « Annulation Réservation », « Récupération Ressource », « Demande Renouvellement Abstrait » et « Restitution Ressource ». Notons que pour une raison de lisibilité de cette vue, nous donnons un diagramme d'activités par sous-CMP. En outre, pour pouvoir intégrer des variantes de type événement dans un point de variation de type activité, nous représentons un événement par une activité manuelle (ex. Demande Allocation Directe).

« Demande Allocation Abstraite »



« Annulation Réservation »



Notons que dans le sous-CMP « Récupération Ressource », nous utilisons une règle de construction déjà définie dans le troisième chapitre, pour assurer la cohérence du diagramme

d'activités. Cette règle concerne l'événement « Récupération Ressource » qui apparaît deux fois dans le modèle.

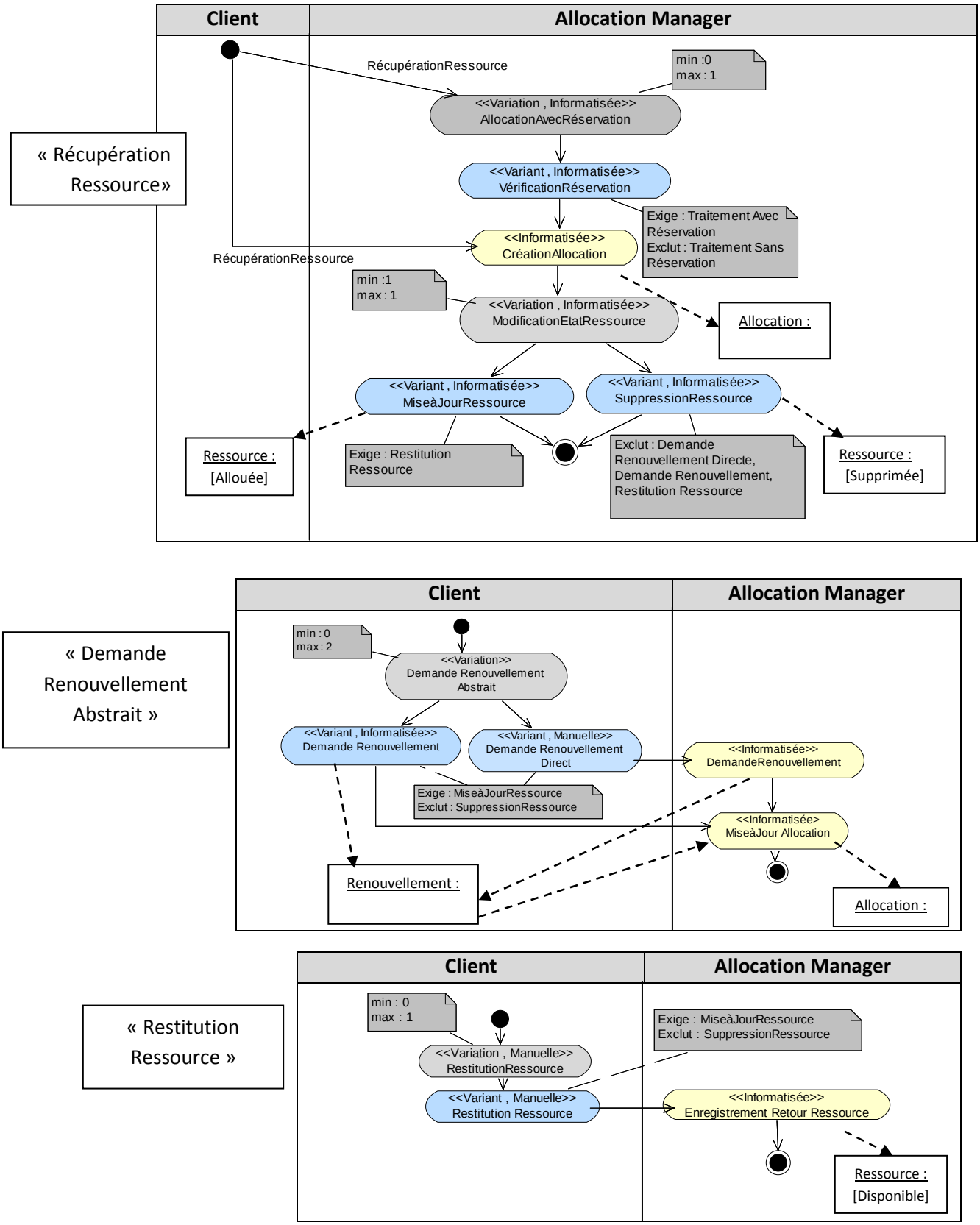


Figure 4-5 : Vue Métier du CMP « Gestion Allocation Ressource »

3.3 Phase 3 : Spécification de la vue fonctionnelle

3.3.1 Description

La vue fonctionnelle d'un CMP représente un ensemble de services rendus par le composant. Elle est modélisée par un diagramme de cas d'utilisation où chaque cas d'utilisation correspond à une fonction métier du composant et correspond à une séquence d'actions produisant un résultat observable pour un acteur particulier. L'objectif de la spécification fonctionnelle est la génération des cas d'utilisation à partir des activités informatisées. À ce niveau, on modélise les aspects informatiques du système, sans pour autant rentrer dans les détails d'implémentation. Notons que la décomposition du CMP global en sous-CMP exige la conservation de cette décomposition au niveau fonctionnel en affectant chaque cas d'utilisation au sous-CMP candidat à l'assumer. La spécification de la vue fonctionnelle comprend les étapes suivantes (Cf. Figure 4-6) :

- *Génération des cas d'utilisation fixes* : à partir de la liste des activités fixes informatisées, on génère des cas d'utilisation fixes.
- *Génération des cas d'utilisation variables* : à partir de la liste des activités variables informatisées, on génère des cas d'utilisation variables. Cette étape sera détaillée dans le paragraphe (3.4.2).
- *Mise en relation des cas d'utilisation* : il s'agit d'une part, de préciser si un cas d'utilisation appelle un autre cas d'utilisation (relation « include »), et d'autre part de préciser si un cas d'utilisation s'inspire d'un autre cas d'utilisation (relation « extend »). Il s'agit aussi de préciser s'il existe des cas d'utilisation génériques (relation « généralisation »). Concernant les cas d'utilisation variables, la mise en relation entre des cas d'utilisation variations et leurs variantes est exprimée par la relation « include ».
- *Description des cas d'utilisation* : il s'agit de définir de manière détaillée chaque cas d'utilisation et de décrire comment il assume l'obtention de sa finalité.
- *Établissement de la cartographie fonctionnelle* : il s'agit d'établir une cartographie des CME utilisés par chaque sous-CMP. Chaque CMP/CME est modélisé par un package UML, et chaque cas d'utilisation est affecté au CMP/CME candidat pour l'assumer. Une technique pour déterminer un CME est d'appliquer une analyse syntaxique sur le nom du cas d'utilisation (une des pratiques consiste à choisir le complément du verbe nommant le cas d'utilisation [Hassine, 2005]). Notons que le niveau de granularité du CMP peut agir sur la création ou non de sous-CMP qui utilisent des CME. Dans le cas où le CMP est de granularité faible, il n'est pas nécessaire de garder la décomposition établie au niveau de la vue métier pour constituer la cartographie fonctionnelle. Dans le cas contraire, il est recommandé de créer un sous-CMP pour chaque sous-PM identifié.

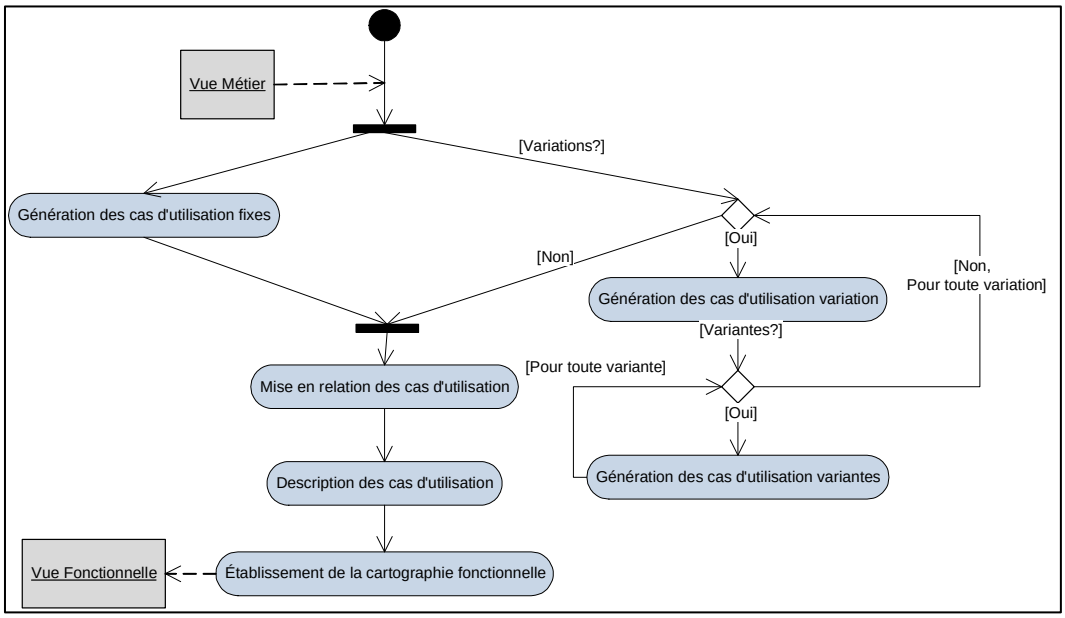


Figure 4-6 : Enchaînement des étapes de la phase spécification de la vue fonctionnelle

3.3.2 Génération des cas d'utilisation variables

a) De la variabilité métier vers la variabilité fonctionnelle

La vue métier constitue la vue organisationnelle du CMP constituée par des activités fixes et des activités supportant la variabilité. La vue fonctionnelle est réalisée par génération et mise en relation des cas d'utilisation à partir des activités **informatisées** de la vue métier sous la forme d'un diagramme de cas d'utilisation. En effet, le passage entre ces deux vues conduit à la génération d'une variabilité fonctionnelle matérialisée par des fonctionnalités variables du CMP. Le Tableau 4-6 illustre les types de variabilité fonctionnelle engendrés par la variabilité métier.

Élément	**Définition**	**Variabilité**
Cas d'utilisation	Un cas d'utilisation est une fonctionnalité du système déclenchée en réponse à la stimulation d'un acteur externe.	Un cas d'utilisation peut être présent dans un PM d'un SI particulier et absent dans le même PM dans un autre SI. Il peut être déclenché par différents acteurs ou exécuté de différentes manières selon les exigences de chaque SI. Ces deux formes de variabilité sont déduites de la variabilité d'une activité dans la vue métier.
Acteur	Un acteur représente un rôle joué par une personne qui interagit avec le système.	La variabilité de l'acteur est aussi une conséquence de la variabilité des activités (donc des acteurs) de la vue métier.

Tableau 4-6 : Variabilité dans la vue fonctionnelle

À partir du Tableau 4-6, nous extrayons les différentes formes de variabilité qui peuvent être représentées dans la vue fonctionnelle d'un CMP. Le Tableau 4-7 résume l'ensemble de ces formes dont les mécanismes de représentation sont donnés dans la section suivante.

Élément	**Forme de variabilité**
Cas d'utilisation	Alternative/ Alternative Optionnelle / Option / Ensemble d'alternatives
Acteur	Variation de Rôle
	Optionnel

Tableau 4-7 : Formes de variabilité dans la vue fonctionnelle

b) Règles de traduction

✓ Variabilité des cas d'utilisation

Pour représenter la variabilité sur la vue fonctionnelle, nous utilisons l'opérateur générique défini dans [Arnaud, 2008]. Il est composé d'un point de variation, matérialisé par un cas d'utilisation stéréotypé <<Variation>> et de plusieurs variantes matérialisées par des cas d'utilisation stéréotypés <<Variant>>. À l'aide de valeurs marquées sur ces stéréotypes, on exprime les cardinalités minimale et maximale (selon le type de variabilité).

- Si les cardinalités minimale et maximale sont égales à 1, il s'agit de variantes alternatives. Il faudra obligatoirement sélectionner l'une d'entre elles.
- Si la cardinalité minimale est égale à 0 et la cardinalité maximale est égale à n, il s'agit alors de variantes optionnelles, et la cardinalité maximale indique le nombre d'options potentiellement réutilisables.
- Si la cardinalité minimale est égale à 0 et la cardinalité maximale est égale à 1, il s'agit d'une variation de type alternative optionnelle où aucune ou une seule variante peut être sélectionnée lors de la réutilisation.
- Si la cardinalité minimale est égale à 1 et la cardinalité maximale est égale à n, il s'agit d'une variation de type ensemble d'alternatives où il faudra obligatoirement sélectionner au moins une variante.

D'une manière générale, lors de la traduction de la variabilité de la vue métier (Cf. Figure 4-7.a) vers la vue fonctionnelle (Cf. Figure 4-7.b), une activité de type « Variation » (respectivement « Variant ») donne lieu à un cas d'utilisation de type « Variation » (respectivement « Variant »), et la transition entre un point de variation et ses variantes est traduite par une relation d'inclusion entre le cas d'utilisation « Variation » et les cas d'utilisation « Variant » (Cf. Figure 4-7.b). Notons que lors du passage d'un diagramme d'activités vers un diagramme de cas d'utilisation, il ne faut conserver que les activités informatisées. En outre, pour décrire un cas d'utilisation, nous utilisons le modèle textuel présenté dans la Figure 4-8.

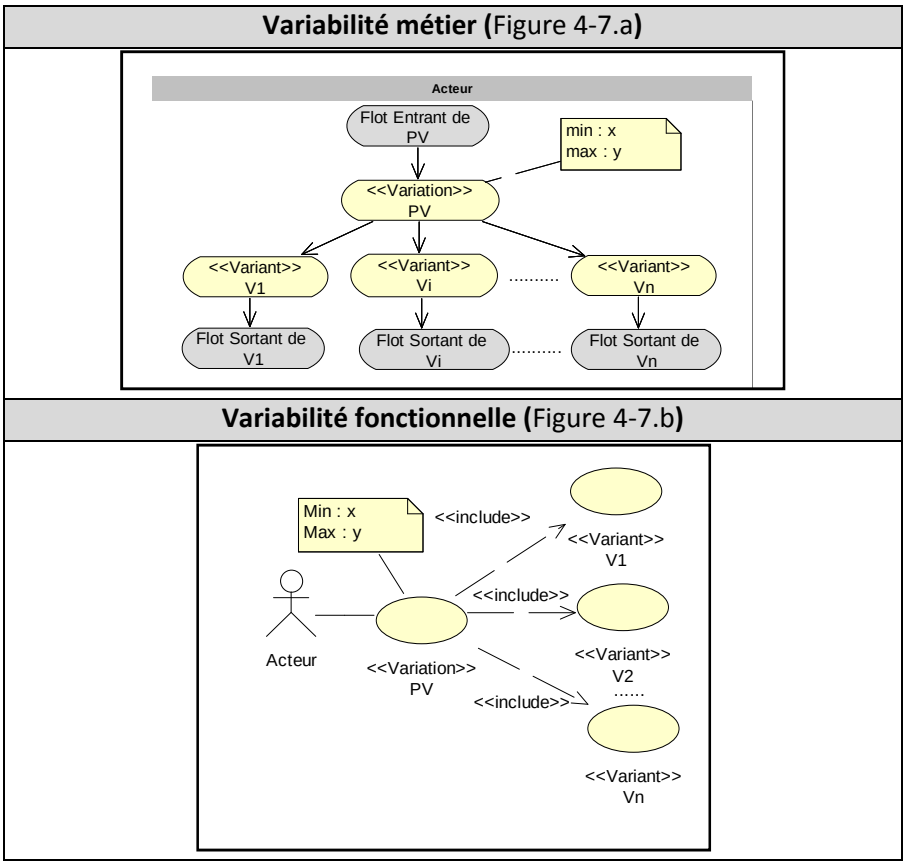


Figure 4-7 : Représentation de la variabilité des cas d'utilisation

Cas d'utilisation
Acteurs
Résumé
Événement déclencheur
Pré-conditions
Variantes **- V1 :…** **Description détaillée : scénario** **- V2 :...** **Description détaillée : scénario**
Post-conditions

Figure 4-8 : Description textuelle d'un cas d'utilisation

- ✓ **Variabilité du degré d'informatisation**

Pour traduire la variabilité du degré d'informatisation des activités de la vue métier (Cf. Figure 4-9.a) vers la vue fonctionnelle (Cf. Figure 4-9.b), nous créons un cas d'utilisation point de variation de type option incluant un seul cas d'utilisation de type variante correspondant à l'activité variante informatisée.

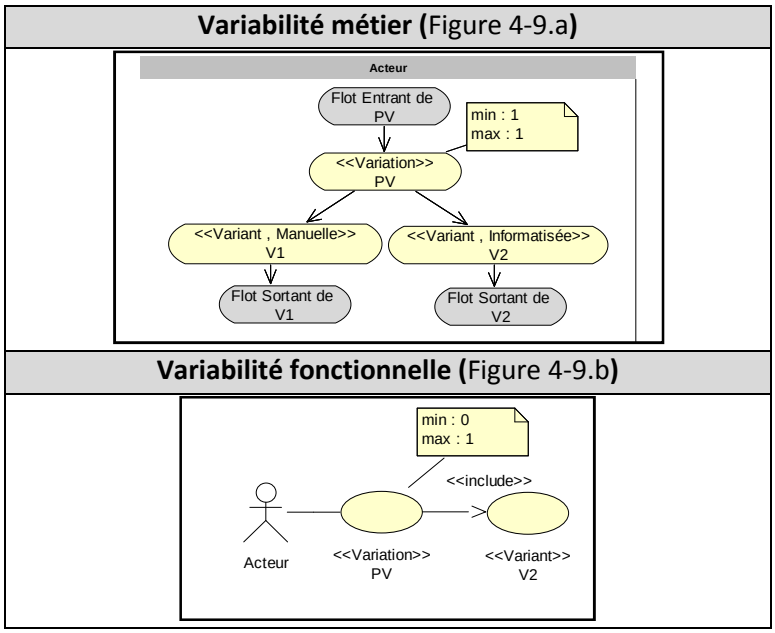


Figure 4-9 : Variabilité du degré d'informatisation

- ✓ **Variabilité des acteurs**

Pour traduire la variabilité des acteurs de la vue métier (Cf. Figure 4-10.a) vers la vue fonctionnelle (Cf. Figure 4-10.b), nous créons un cas d'utilisation point de variation de type alternative incluant des cas d'utilisation variantes. Chaque variante incluse dans cette variation est déclenchée par l'auteur responsable de l'activité correspondante à cette variante.

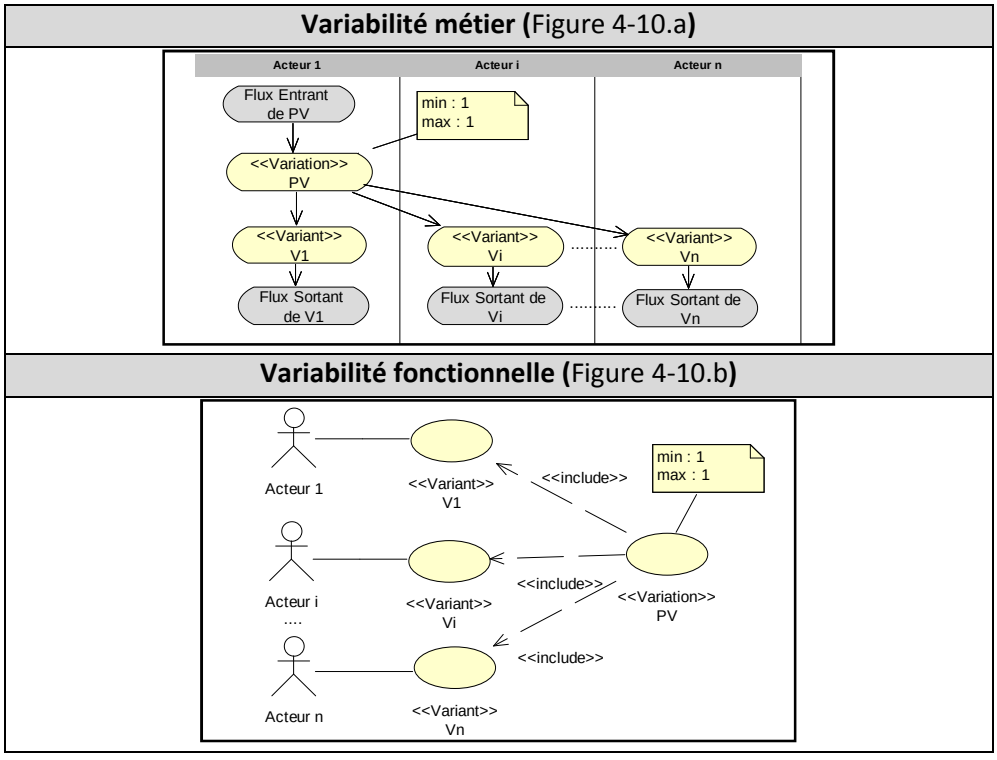


Figure 4-10 : Représentation de la variabilité des acteurs

Les règles de traduction suivantes permettent de ne conserver que les activités informatisées lors du passage au diagramme de cas d'utilisation.

- Si une activité point de variation n'admet que des variantes informatisées, le point de variation et ses variantes sont conservés dans la vue fonctionnelle. Les mêmes valeurs marquées sont conservées.

- Si une activité point de variation n'admet que des variantes manuelles, le point de variation et ses variantes ne sont pas conservés dans la vue fonctionnelle.

- Si une activité point de variation admet des variantes manuelles et informatisées, on ne conserve que les variantes informatisées, la cardinalité maximale du point de variation devient le nombre des variantes conservées :

+ Si le nombre des variantes conservées est supérieur ou égal à 2, le même type de variation est conservé en changeant la cardinalité maximale.

+ Sinon, le point de variation devient de type option de cardinalité maximale égale à 1.

Cas particuliers :

[1] Une activité informatisée, non stéréotypée « Variant » et « Variation », liée directement à une variante informatisée, donne lieu à un cas d'utilisation de type option incluant le cas d'utilisation variante correspondant à cette activité.

[2] Une activité informatisée « A » liée directement à une variante manuelle issue d'un point de variation « PV », donne lieu à un cas d'utilisation « Variant ».

2.a. Si PV admet des variantes informatisées et manuelles, PV est conservée dans la vue fonctionnelle, ses variantes manuelles sont remplacées par les activités informatisées auxquelles elles sont directement liées.

2.b. Si PV est manuelle, le même nom PV est conservé dans le cas d'utilisation de type « Variation » créé, ce cas d'utilisation inclut le cas d'utilisation correspondant à A.

[3] Un cas d'utilisation racine peut être créé incluant un ensemble de cas d'utilisation correspondant à un ensemble d'activités informatisées successives et ininterruptibles.

c) Règles de réduction

Les règles de réduction suivantes sont les mêmes que celles utilisées pour la réduction de la vue métier, mais elles sont automatiquement appliquées suite à la réduction de la vue métier.

- Si un point de variation admet des « *alternatives* », il sera remplacé par la variante sélectionnée.

- Si un point de variation est de type « *alternative optionnelle* », il sera remplacé par la variante sélectionnée. Si aucune variante n'est choisie, il sera supprimé.

- Si un point de variation admet des « *options* », il sera remplacé par la variante sélectionnée. Si aucune variante n'est choisie, il sera supprimé. Si plusieurs variantes sont sélectionnées, le point de variation est conservé et les variantes choisies restent reliées au point de variation par un *include*.

- Si un point de variation est de type « *ensemble d'alternatives* », il sera remplacé par la variante sélectionnée. Si plusieurs variantes sont sélectionnées, le point de variation est conservé et les variantes choisies restent reliées au point de variation par un *include*.

- Les variantes non sélectionnées sont supprimées.
- Un acteur dont les cas d'utilisation sont supprimés est supprimé.

3.3.3 Exemple d'illustration

a) Génération et mise en relation des cas d'utilisation

La Figure 4-11 représente la vue fonctionnelle du CMP « Gestion Allocation Ressource ».

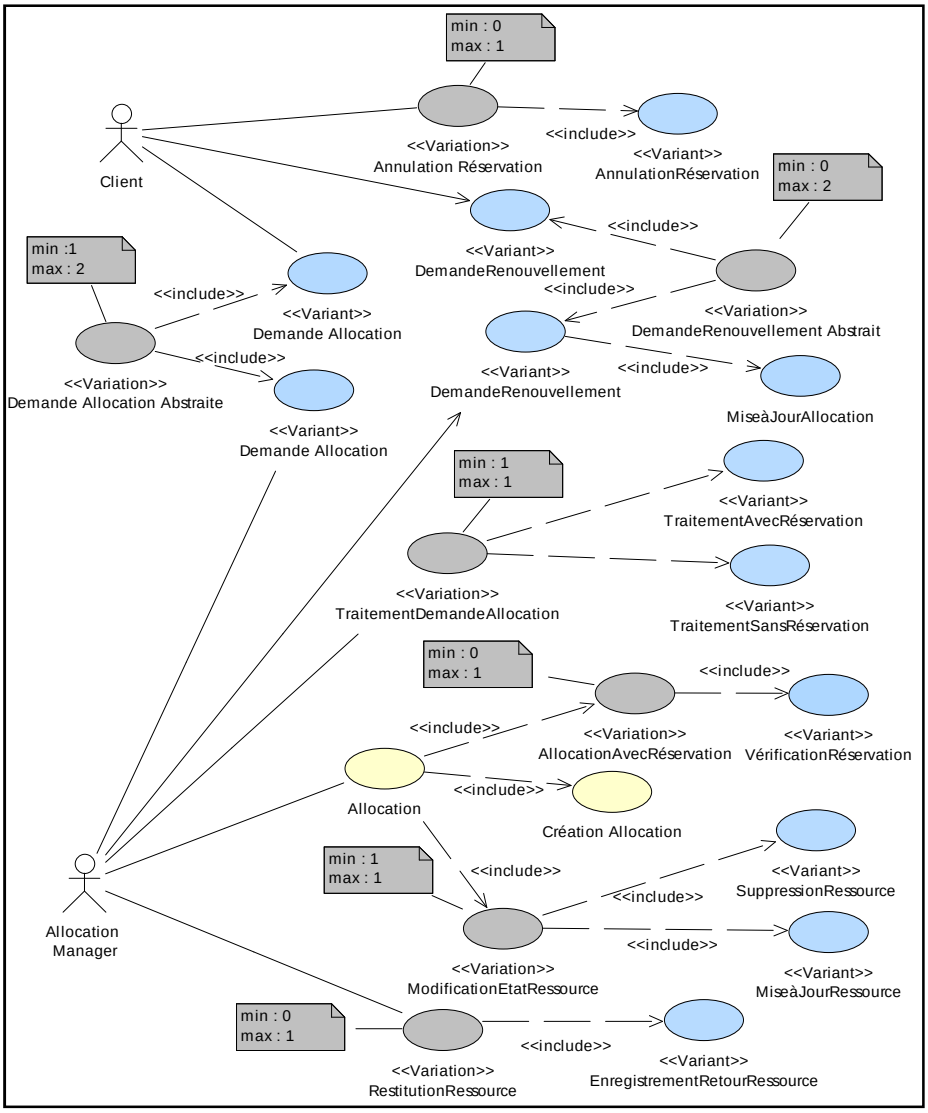


Figure 4-11 : Vue fonctionnelle du CMP « Gestion Allocation Ressource »

La génération des cas d'utilisation à partir de la vue métier donne lieu à six cas d'utilisation de type variation (*DemandeAllocationAbstraite*, *TraitementDemandeAllocation, AnnulationRéservation*, *AllocationAvecRéservation*, *ModificationEtatRessource* et *DemandeRenouvellementAbstrait*) automatiquement générés à partir des activités variation. D'un autre côté, le point de variation manuel *RestitutionRessource* est conservé en incluant la variante *EnregistrementRetourRessource* (Cas particulier [2.b]). En outre, le cas d'utilisation *Allocation* est créé comme un cas d'utilisation racine des cas d'utilisation : *AllocationAvecRéservation, CréationAllocation* et *ModificationEtatRessource* (Cas particulier

[3]). Notons aussi que les deux variantes *Demande Allocation* et *Demande Renouvellement* sont créées selon la règle [2.a]. Les variations sont conservées en appliquant les règles de traduction sur les valeurs marquées.

La Figure 4-12 illustre un exemple de réduction de la vue fonctionnelle. La variante *Demande Allocation* est assignée seulement au client, ce qui implique la suppression de la variante assignée à l'allocation manager. La variante *TraitementSansRéservation* remplace le point de variation *TraitementDemandeAllocation.* La variante *SuppressionRessource* remplace le point de variation *ModificationEtatRessource*. Les autres points de variation (A*nnulation Réservation*, *AllocationAvecRéservation*, *DemandeRenouvellementAbstrait* et *RestitutionRessource)* sont tous supprimés avec leurs variantes.

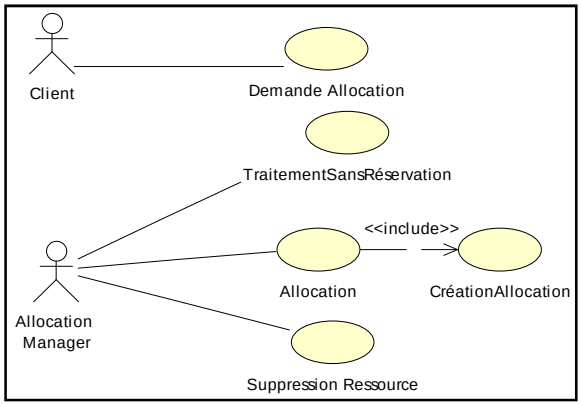


Figure 4-12 : Exemple de réduction de la vue fonctionnelle du CMP « Gestion Allocation Ressource »

b) Description des cas d'utilisation

La Figure 4-13 illustre un exemple de description textuelle du point de variation « Traitement Demande Allocation ».

Cas d'utilisation : Traitement Demande Allocation
Acteur : Allocation Manager.
Résumé : Désigne le traitement d'une demande d'allocation d'une ressource.
Événement déclencheur : Le client demande l'allocation d'une ressource.
Pré-conditions : - Une demande d'allocation est créée.
Variantes **- V_1 :** Traitement Avec Réservation. **Description détaillée** : 1. Identifie la ressource. 2. Vérifie la disponibilité de la ressource. 3. Crée la réservation. 4. Communique au client le code réservation. **- V_2 :** Traitement Sans Réservation. **Description détaillée** : 1. Identifie la ressource. 2. Vérifie la disponibilité de la ressource. 3. Informe le client de la possibilité d'allocation.
Post-conditions : La demande est traitée.

Figure 4-13 : Description du cas d'utilisation « Traitement Demande Allocation »

c) Établissement de la cartographie fonctionnelle

La Figure 4-14 propose une cartographie des CME : « DemandeAllocation », « Allocation », « Réservation », « Ressource », « Client » et « Renouvellement » utilisés par le CMP principal « Gestion Allocation Ressource ». Nous rappelons que l'identification des CME est réalisée par le choix du complément utilisé dans le nom de cas d'utilisation. Le niveau de granularité de ce CMP étant faible, nous avons choisi de réunir tous les sous-CMP dans un seul CMP global.

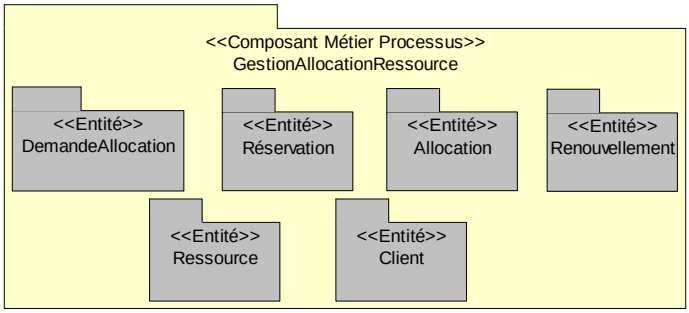


Figure 4-14 : Cartographie fonctionnelle

3.4 Phase 4 : Spécification de la vue dynamique

3.4.1 Description

La vue dynamique d'un CMP représente son aspect comportemental. Nous représentons cette vue par un ensemble de diagrammes de séquence d'UML 2. L'objectif de la spécification dynamique du CMP est d'illustrer les interactions entre CME en définissant un scénario. Cette phase inclut trois étapes (Cf. Figure 4-15) :

- *Analyse dynamique* : il s'agit d'une modélisation précise proposée par un diagramme de séquence pour chaque cas d'utilisation et une formalisation des opérations à assumer par les CME spécifiés dans la cartographie fonctionnelle.
- *Spécification des interactions fixes* : il s'agit de la formalisation des scénarios correspondants aux cas d'utilisation fixes.
- *Spécification des interactions variables* : il s'agit de la formalisation des scénarios correspondants aux cas d'utilisation variables. Cette étape est détaillée dans le paragraphe 3.4.2.

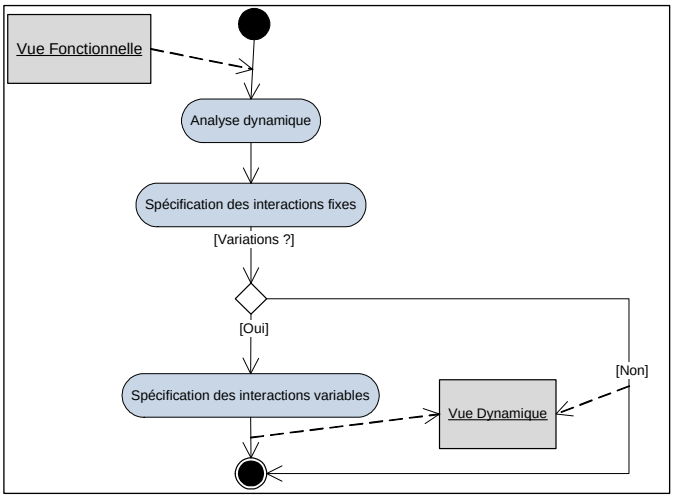


Figure 4-15 : Enchaînement des étapes de la phase spécification de la vue dynamique

3.4.2 Spécification des interactions variables

a) De la variabilité fonctionnelle vers la variabilité dynamique

La variabilité dans la vue dynamique est matérialisée essentiellement par la variabilité du comportement d'un CMP lors de sa réutilisation dans un SI donné. Elle correspond à une description détaillée de la variabilité fonctionnelle. Le Tableau 4-8 illustre la variabilité sur la vue dynamique d'un CMP.

Élément	**Définition**	**Variabilité**
Interaction	Une interaction modélise un comportement dynamique. Elle se traduit par l'envoi de messages entre objets.	La variation des fonctionnalités offertes par un CMP influence sur la variabilité des interactions contenues dans son comportement. La variation d'une interaction signifie que son comportement est optionnel, c.à.d. que tous les messages contenus dans cette interaction peuvent être absents dans certains SI.
Occurrence d'interaction	Une occurrence d'interaction est un raccourci vers une copie du contenu d'une interaction. Elle est matérialisée par une interaction qui référence une interaction combinée.	La variabilité des occurrences d'interaction découle de celle des interactions combinées. En effet, une interaction combinée permet aux CMP de définir un ensemble de comportements variants parmi lesquels le concepteur doit effectuer un choix.
Ligne de vie des objets	La ligne de vie d'un objet précise l'existence de l'objet concerné durant un certain laps de temps.	Un objet supportant la variabilité signifie que les messages envoyés ou reçus par cet objet sont variables, c.à.d. qu'ils peuvent être supprimés par le concepteur lors de la réutilisation du CMP.

Tableau 4-8 : Variabilité de la vue dynamique

À partir du Tableau 4-8, nous extrayons les différentes formes de variabilité qui peuvent être représentées dans la vue dynamique d'un CMP. Le Tableau 4-9 résume l'ensemble de ces formes dont les mécanismes de représentation sont donnés dans le paragraphe suivant.

Élément	**Forme de variabilité**
Interaction	Variation / Variant
Occurrence d'interaction	Variation
Ligne de vie des objets	Optionnel

Tableau 4-9 : Formes de variabilité dans la vue dynamique

b) Règles de traduction

Lors de la traduction d'un type de variabilité de la vue fonctionnelle (Cf. Figure 4-16.a) vers la vue dynamique (Cf. Figure 4-16.b), un cas d'utilisation de type « Variation » donne lieu à une

interaction de type « Variation ». L'ensemble de ses variantes est représenté par un sous-diagramme de séquence comportant des références sur ces variantes.

Dans cette partie, nous présentons seulement la représentation de la variabilité sur les interactions dont découle la variabilité des objets. Pour ce faire, nous utilisons le mécanisme de composition proposé dans la nouvelle génération des diagrammes de séquence d'UML 2. Il s'agit de la possibilité de composer les diagrammes de séquence pour spécifier des comportements plus complexes.

La composition est réalisée dans UML 2 par deux mécanismes principaux : 1) les frames de type référence (interaction use) ou de type 'sd' (sous-diagramme) et 2) les opérateurs d'interaction. Ces mécanismes permettent de modéliser des fragments d'interaction combinés.

Des exemples d'opérateurs d'interaction sont donnés ci-dessous.

- L'opérateur *loop* associé à un seul opérande, permet d'exprimer une boucle.
- L'opérateur *opt* permet de spécifier qu'un fragment d'interaction (opérande) est optionnel (à l'aide d'une contrainte).
- L'opérateur *alt* est destiné à la définition de structures de choix conditionnels entre plusieurs fragments (opérandes).
- L'opérateur *seq* indique que différents opérandes doivent être exécutés en séquence, mais dans n'importe quel ordre.
- L'opérateur *strict* exprime lui aussi une séquence, mais selon l'ordonnancement donné.

Pour représenter la variabilité sur la vue dynamique, nous nous focalisons sur les deux opérateurs *Alt* et *Opt* pour spécifier des choix de comportements qui doivent être effectués par un concepteur. En effet, pour représenter un point de variation, nous utilisons un frame de type référence pour référencer le sous-diagramme de séquence comportant les variantes (Cf. Figure 4-16.b). Les variantes sont modélisées aussi par des références inclues dans les opérateurs utilisés. Les variantes de type « Alternative » et « Ensemble d'alternatives » sont des opérandes de l'opérateur Alt (Cf. Figure 4-16.c). Les variantes de type « Option » sont des opérandes de l'opérateur Opt (Cf. Figure 4-16.d). Enfin, les variantes de type « Alternative optionnelle » sont des opérandes de l'opérateur Alt inclus dans l'opérateur Opt (Cf. Figure 4-16.e). Une variante donnée est représentée dans un sous-diagramme de séquence de type « sd » (Cf. Figure 4-16.f).

Dans la vue dynamique, nous n'utilisons aucun stéréotype et nous ne distinguons pas les types de variation par l'utilisation de cardinalités car cette contrainte est déjà modélisée dans les vues précédentes. Le type de variation est obtenu par application du mécanisme de traçabilité de la variabilité exprimée.

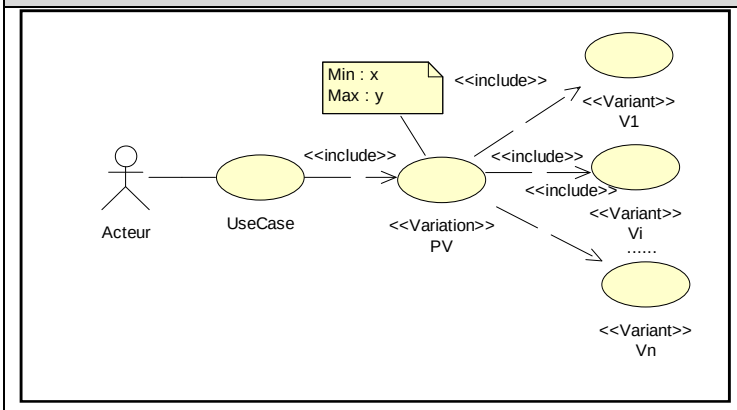
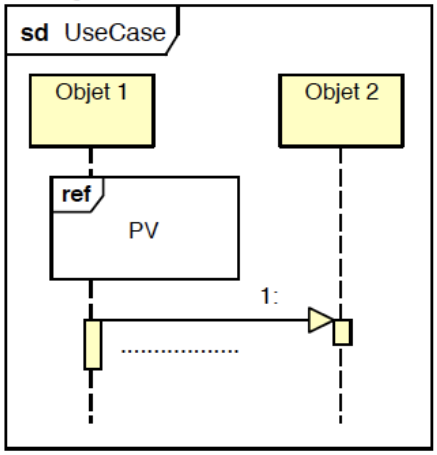
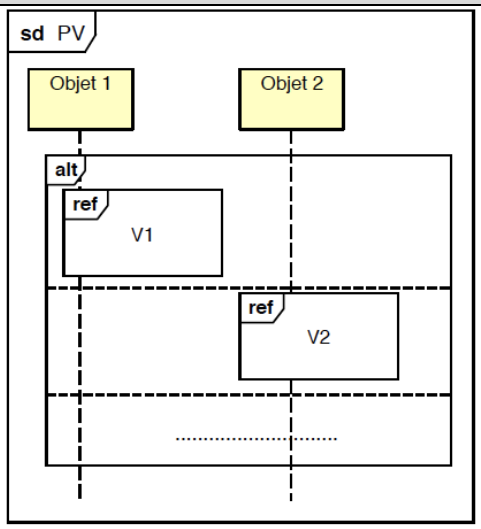
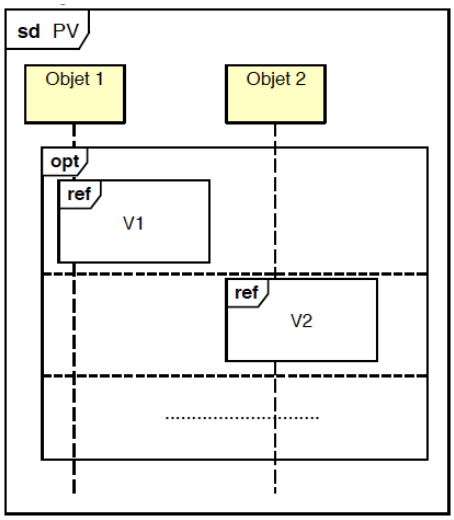
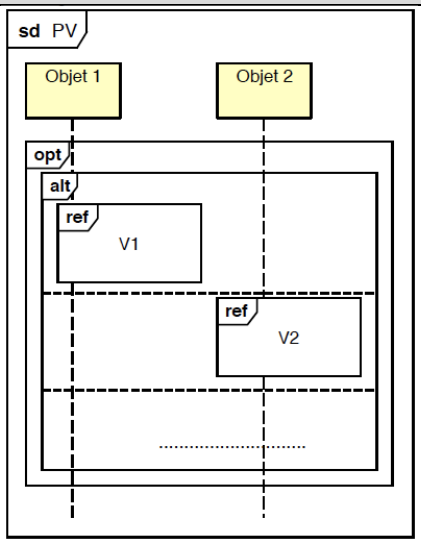
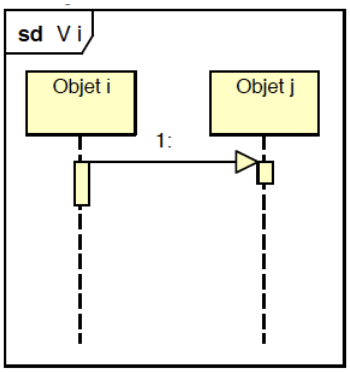
Variabilité fonctionnelle (Figure 4-16.a)	**Point de variation** (Figure 4-16.b)
	
« Alternative » (min=1, max=1) / « Ensemble d'alternatives » (min=1, max=n) (Figure 4-16.c)	**« Option » (min=0, max=n)** (Figure 4-16.d)
	
« Alternative Optionnelle » (min=0, max=1) (Figure 4-16.e)	**Variante** (Figure 4-16.f)
	

Figure 4-16 : Représentation de la variabilité sur les diagrammes de séquence

c) Règles de réduction

- Si aucune variante n'est choisie dans la vue métier, le frame de référence est supprimé ainsi que le sous-diagramme de séquence référencé par le frame.

- Si au moins une variante est choisie, l'ensemble des frames de référence des variantes non sélectionnées est supprimé. Les sous-diagrammes de séquence référencés par ces frames sont aussi supprimés.

  + Si la variation est de type « Alternative » ou « Alternative optionnelle », la variante choisie devient obligatoire en supprimant les opérateurs utilisés.

- Après la réduction des variantes, les lignes de vie qui n'envoient et qui ne reçoivent aucun message sont supprimées.

3.4.3 Exemple d'illustration

Nous rappelons par la Figure 4-17 la partie de la vue fonctionnelle correspondant au cas d'utilisation « Allocation ».

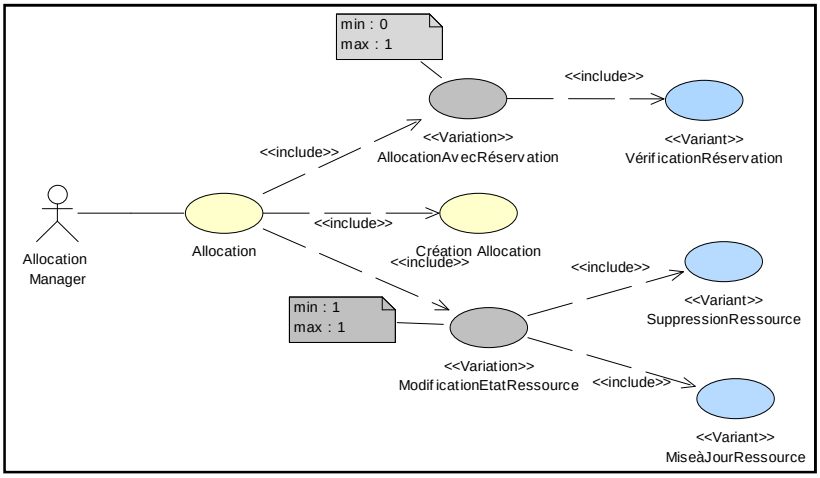
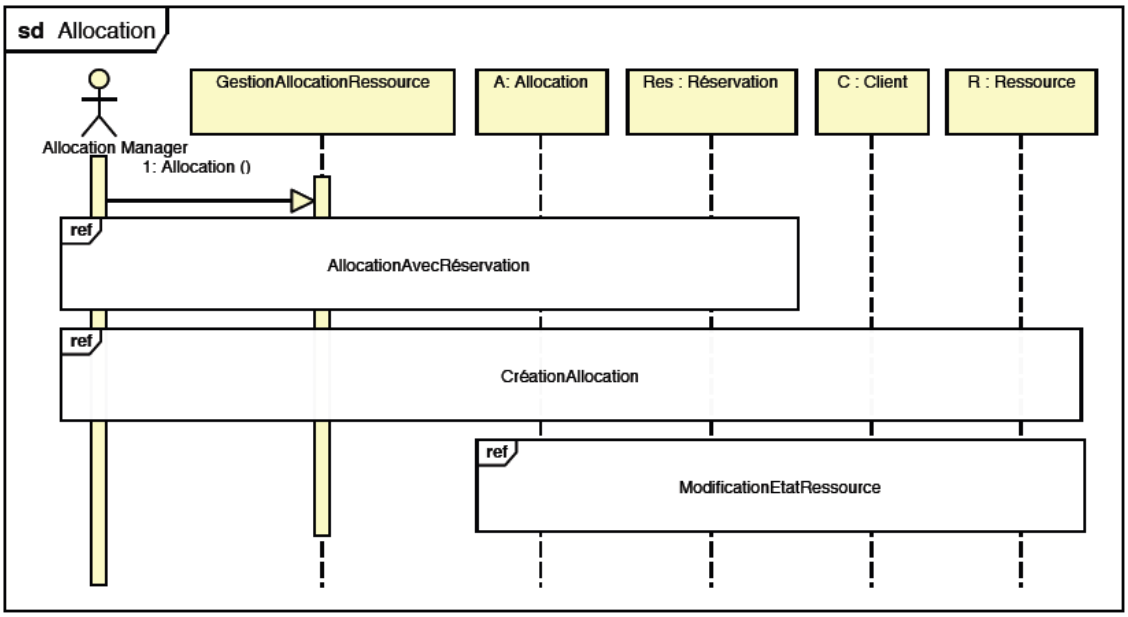


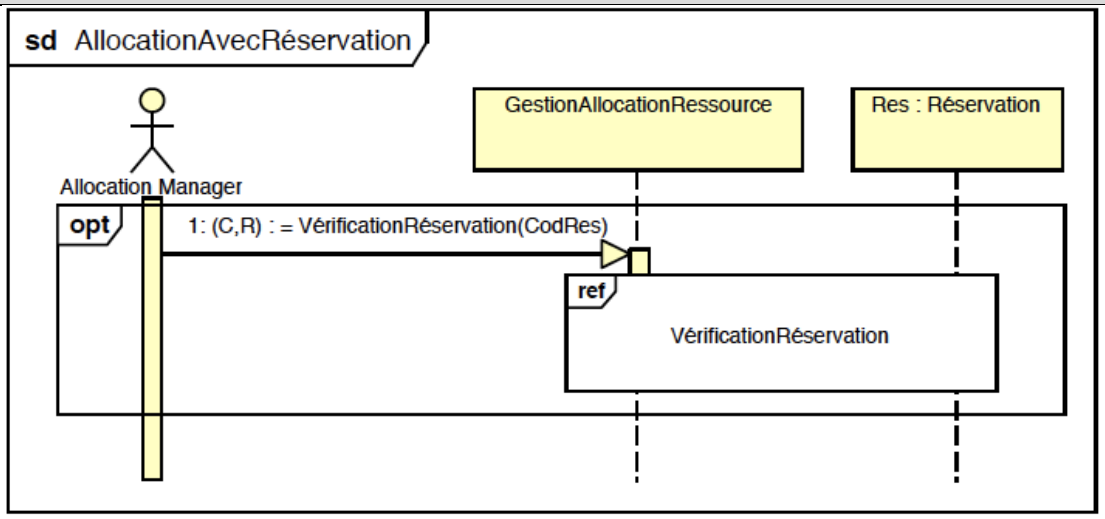
Figure 4-17 : Cas d'utilisation « Allocation »

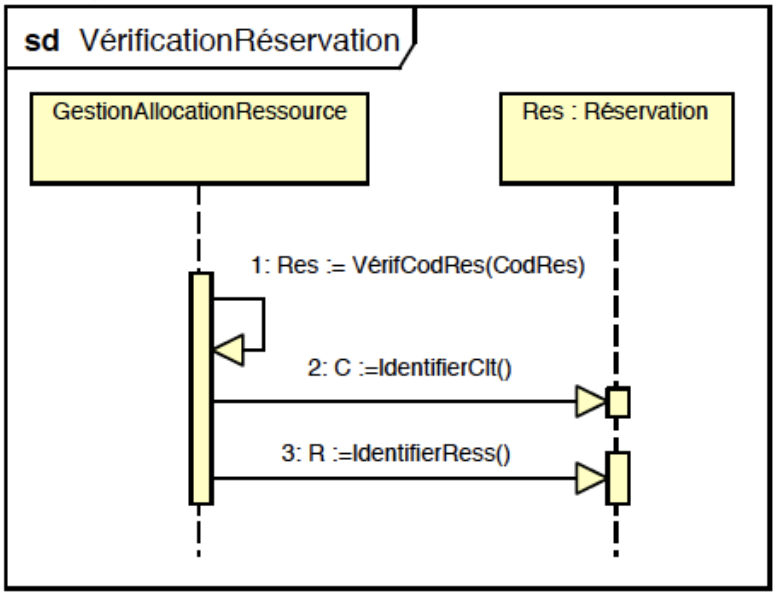
La Figure 4-18.a illustre la spécification dynamique de ce cas d'utilisation représentée par un diagramme de séquence faisant partie de la vue dynamique du CMP « Gestion Allocation Ressource». Elle comporte une première référence au fragment variation « AllocationAvecRéservation », une deuxième référence au fragment fixe « Création Allocation » et une autre référence au fragment variation « ModificationEtatRessource ». Les Figure 4-18.b et Figure 4-18.d exposent comment les parties dynamiques des variantes sont incluses dans chaque point de variation. Elles représentent des choix de comportement entre les variantes incluses dans la variation. La Figure 4-18.e donne un exemple de réduction de cette vue en supprimant la variante « VérificationRéservation » pour la variation « AllocationAvecRéservation » et en ne conservant que la variante «MiseàJourRessource » de la variation « ModificationEtatRessource ». L'ensemble de tous les diagrammes de séquence de cette vue est donné dans l'annexe A.

Références vers les fragments dynamiques (Figure 4-18.a**)**

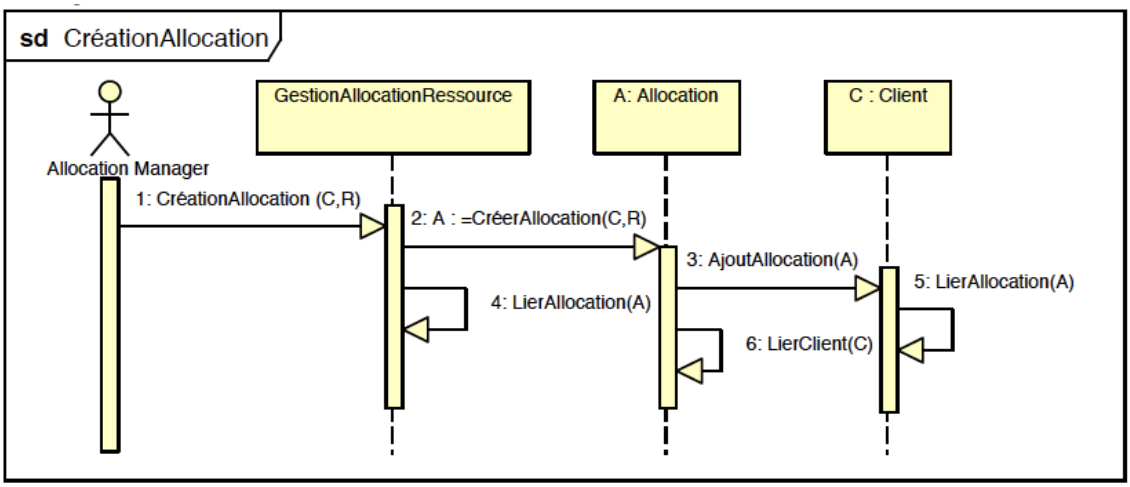


Variation <<AllocationAvecRéservation » (Figure 4-18.b**)**

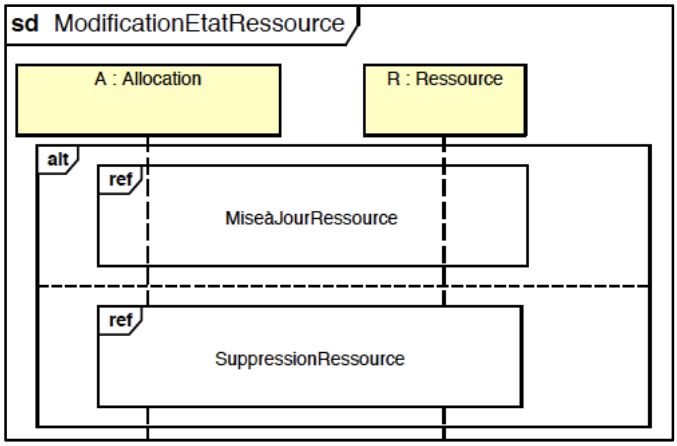


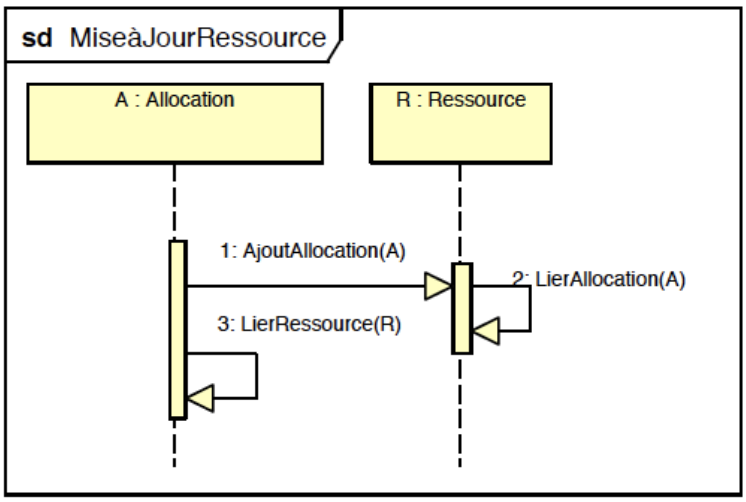


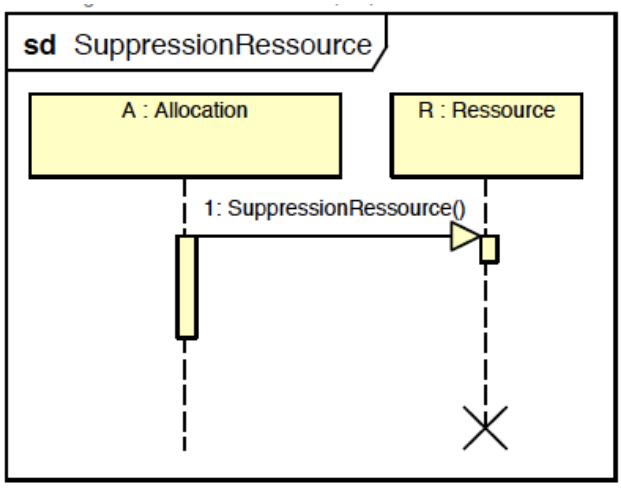
Fragment dynamique fixe <<CréationAllocation» (Figure 4-18.c**)**



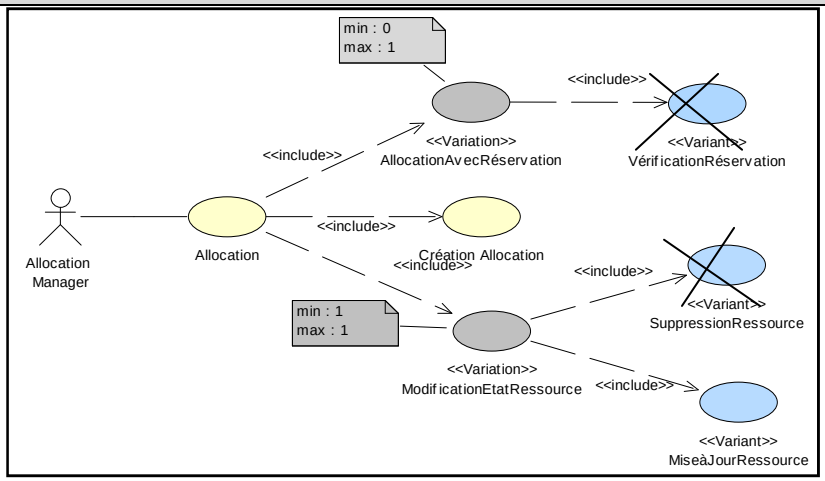
Variation « ModificationEtatRessource » (Figure 4-18.d**)**







Exemple de réduction (Figure 4-18.e**)**



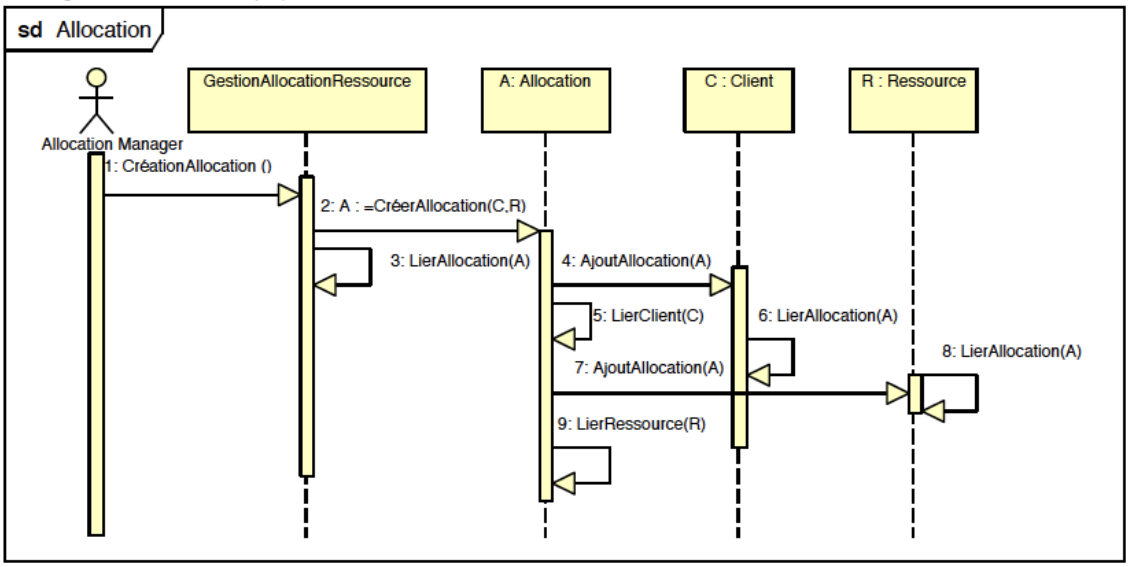


Figure 4-18 : Exemple de diagrammes de séquence inclus dans la vue dynamique du CMP « Gestion Allocation Ressource »

3.5 Phase 5 : Spécification de la vue structurelle

3.5.1 Description

La vue structurelle d'un CMP permet d'identifier les concepts encapsulés par le CMP sous forme de classes, leurs associations avec multiplicité et leurs attributs.

Pour modéliser cette vue, nous adoptons le modèle conceptuel Symphony présenté dans le deuxième chapitre (§1.3.3). La construction de cette vue est fortement couplée avec les vues fonctionnelle et dynamique car elle représente les structures de données manipulées par le comportement du CMP et correspondant à des fonctionnalités du composant. Cette phase inclut cinq étapes (Cf. Figure 4-19) :

- *Spécification de la cartographie du fragment fixe* : il s'agit de l'élaboration de la cartographie de la vue structurelle fixe, incluant les composants (CMP / CME) participants.
- *Spécification de la cartographie des fragments variables* : il s'agit de l'élaboration de la cartographie de chaque variante, incluant les composants participants.

- *Spécification structurelle du fragment fixe :* il s'agit de la modélisation de la classe interface, la classe maître, les classes parties et les classes rôles de chaque composant (CMP / CME) du fragment fixe.
- *Spécification structurelle des fragments variants :* il s'agit de la modélisation de la classe interface, la classe maître, les classes parties et les classes rôles de chaque composant (CMP / CME) d'un fragment variant. Cette étape sera détaillée dans le paragraphe 3.5.2.
- *Établissement des relations entre composants de chaque fragment* : il s'agit d'organiser les composants en assurant leur communication.

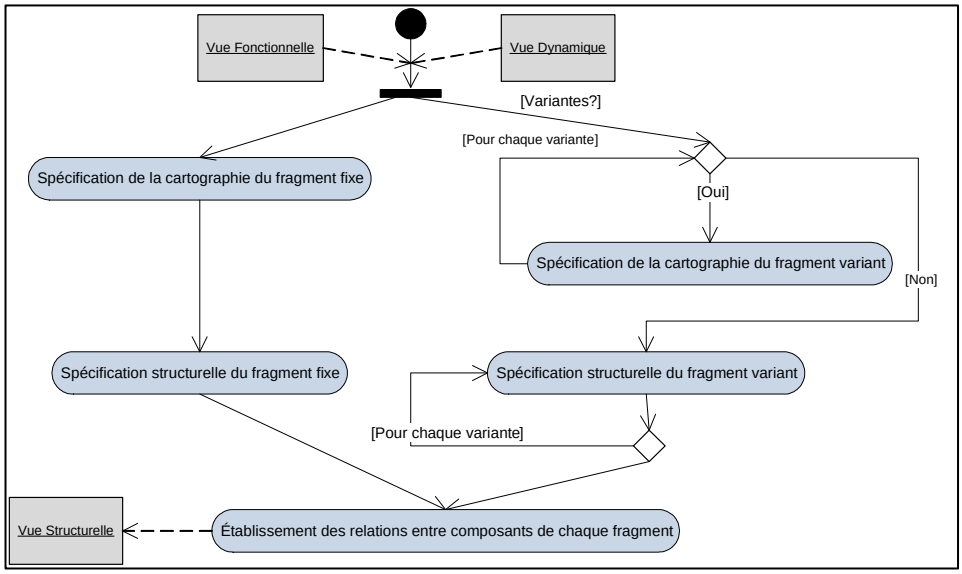


Figure 4-19 : Enchaînement des étapes de la spécification de la vue structurelle

3.5.2 Spécification structurelle des fragments variants

a) De la variabilité fonctionnelle et dynamique vers la variabilité structurelle

Le modèle Symphony est composé de 3 parties : interface, maître (avec des éventuelles classes parties) et rôle. La variabilité sur cette vue peut donc être matérialisée par la variation sur chacune de ces parties. Le Tableau 4-10 illustre la variabilité sur la vue structurelle d'un CMP.

Élément	**Définition**	**Variabilité**
Classe Interface	La classe *interface* joue le rôle de porte d'entrée du composant métier ; elle représente la partie contractuelle avec l'extérieur du composant.	La variation des fonctionnalités offertes par un CMP ainsi que de celle de son comportement influence sur les services fournis par le CMP. En effet, un CMP doit permettre au concepteur le choix entre plusieurs services offerts par le composant. De ce fait, la capacité d'un CMP d'offrir une interface variable est nécessaire.
Classe Maître	La classe *Maître* encapsule la structure interne du composant métier ; elle représente la partie structurelle du composant.	La structure interne d'un CMP doit être variable pour répondre au mieux aux besoins d'un SI donné. La variabilité concerne les attributs de la classe *Maître* ainsi que ses opérations.

Classe Partie	Une classe *Partie* correspond à une structuration partielle des attributs de la classe *Maître*, accentuant de ce fait l'aspect métier du composant.	La variabilité de la classe *Maître* a un impact sur la variabilité de ses classes partie.
Classe Rôle	La classe *Rôle* formalise les différentes collaborations avec les autres composants métier du système ; elle représente la partie collaboratrice du composant.	Un CMP peut avoir un rôle dans un PM d'un SI donné, et un rôle différent dans le même PM d'un autre SI. La capacité d'exprimer des rôles différents d'un CMP est donc essentielle.

Tableau 4-10 : Variabilité de la vue structurelle

À partir du Tableau 4-10, nous extrayons les différentes formes de variabilité qui peuvent être représentées dans la vue structurelle d'un CMP. Le Tableau 4-11 résume l'ensemble de ces formes dont les mécanismes de représentation sont donnés dans le paragraphe suivant.

Élément	**Forme de variabilité**
Classe Interface	Services variables
Classe Maître	Attributs et méthodes variables
Classe Partie	Optionnelle/ Attributs et méthodes variables
Classe Rôle	Optionnelle

Tableau 4-11 : Formes de variabilité dans la vue structurelle

b) Règles de traduction

La vue structurelle variable est composée de plusieurs fragments qui s'assemblent lors de la réutilisation pour former une vue structurelle qui respecte le modèle Symphony. Ces fragments sont des paquetages UML 2 stéréotypés « Fragment Structurel ». Un fragment structurel comporte un package (CMP) stéréotypé « Composant Métier Processus » qui utilise des packages (CME) stéréotypés « Composant Métier Entité ». La communication entre ces CM est assurée par le principe de la classe rôle. En effet, un CM (CMP ou CME) sollicitant un autre CM (CMP ou CME) fournisseur doit connaître son identité et l'appeler via son interface. Cette communication est modélisée dans Symphony par une relation de dépendance spécialisée de type utilisation stéréotypée « utilise » (Cf. Figure 4-20.c).

La spécification des fragments variables de la vue structurelle passe tout d'abord par la traduction de la variabilité fonctionnelle (Cf. Figure 4-20.a) en variabilité structurelle (Cf. Figure 4-20.c). En effet, un cas d'utilisation de type « Variant » donne lieu à un fragment structurel de type « Variant ». Les cas d'utilisation « Variation » ne donnent pas lieu à des fragments structurels car ils seront remplacés par des cas d'utilisation de type « Variant ». En outre, les propriétés structurelles de chaque fragment sont déduites de la vue dynamique (Figure 4-20.b) en adéquation avec la variabilité dynamique. En effet, une classe « Maître » du CMP et de chaque CME d'un fragment, décrit les éléments d'identification du CMP / CME. Ses classes « Partie » y sont complémentaires en mettant en valeur l'aspect métier du CMP / CME. Les services de chaque classe « Interface » sont créés à partir des messages reçus par le CMP /

CME dans la vue dynamique. Enfin, les classes « Rôle » sont obtenues à partir des messages émis du CMP / CME à d'autres CME (Cf. Figure 4-20.b).

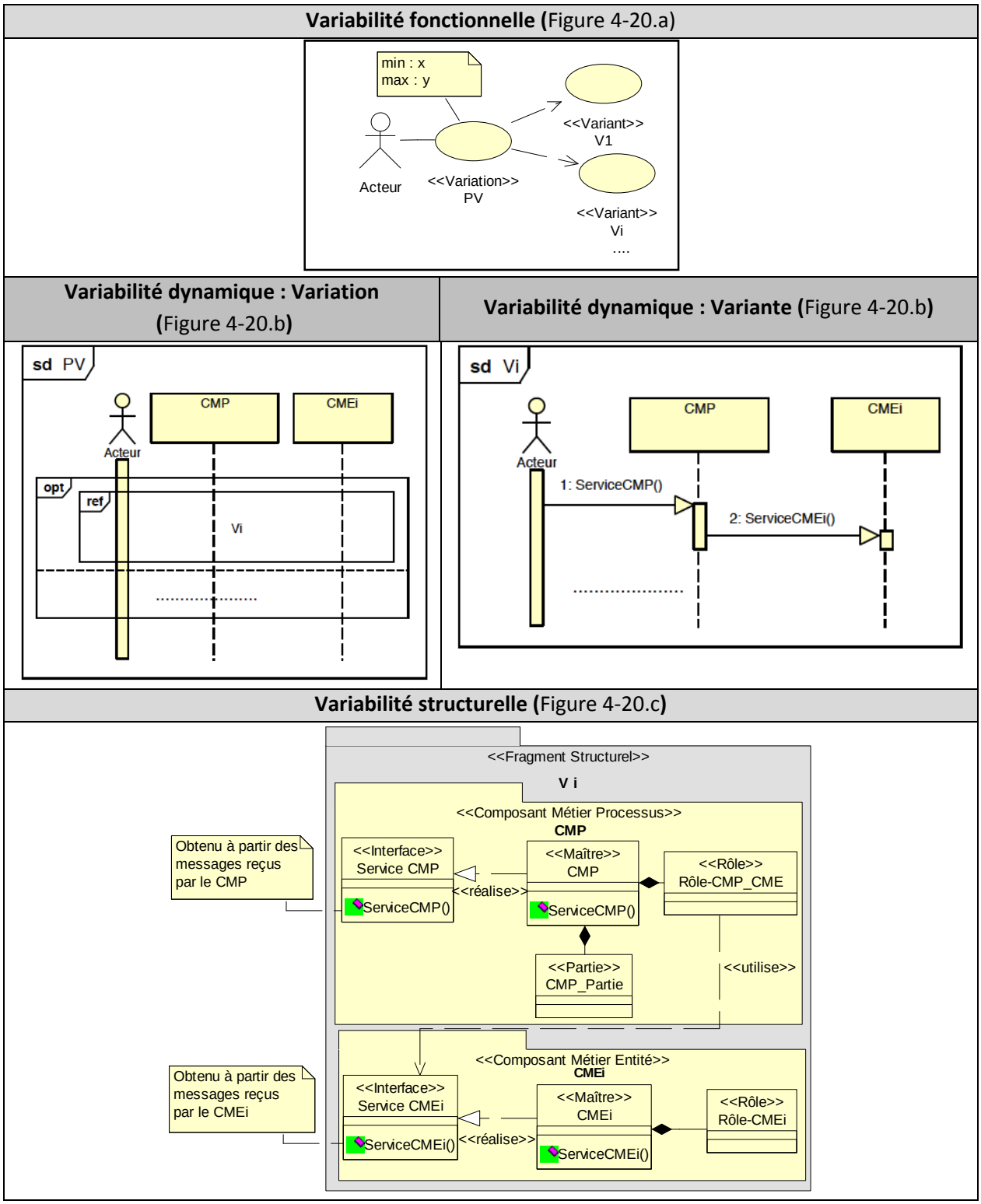


Figure 4-20 : Représentation d'un fragment structurel

D'un autre côté, la spécification du fragment structurel fixe est réalisée de la même manière, à la seule différence qu'un fragment fixe représente la structure de tous les cas d'utilisation fixes en se basant sur les messages échangés dans les interactions fixes.

c) Règles de réduction

La réduction de la vue structurelle consiste en la suppression des fragments non sélectionnés qui eux-mêmes correspondent aux cas d'utilisation, donc aux activités, non sélectionnés. Les fragments choisis sont fusionnés avec le fragment structurel fixe pour former une vue structurelle qui respecte le modèle Symphony. Le fragment qui résulte de cette opération comporte l'union des propriétés de l'ensemble des fragments fusionnés. Ainsi, il comporte les attributs, les méthodes, les classes « Maître », les classes « Partie », les classes « Interface » et les classes « Rôle » de ces fragments.

3.5.3 Exemple d'illustration

La Figure 4-21 illustre la cartographie du fragment structurel fixe du CMP « Gestion Allocation Ressource ». Dans cet exemple, nous identifions le CMP « Gestion Allocation Ressource » qui utilise les CME : « Demande Allocation » et « Allocation » utilisant les CME « Ressource » et « Client ». La Figure 4-22 illustre la spécification structurelle fixe de ces CM. Le CMP a trois services (*DemandeAllocationAbstraite (), TraitementDemandeAllocation() et CréationAllocation ()*) qui jouent le rôle de porte d'entrée du CMP. Il a aussi quatre rôles (*Alloc*, *DemandeAlloc*, *AutorisClt* et *DispoRess*) qui formalisent les différentes collaborations avec les autres CME.

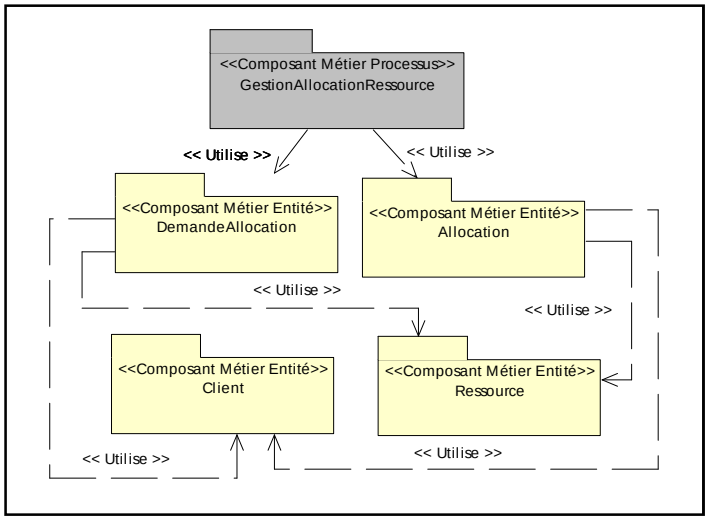


Figure 4-21 : Cartographie du fragment fixe

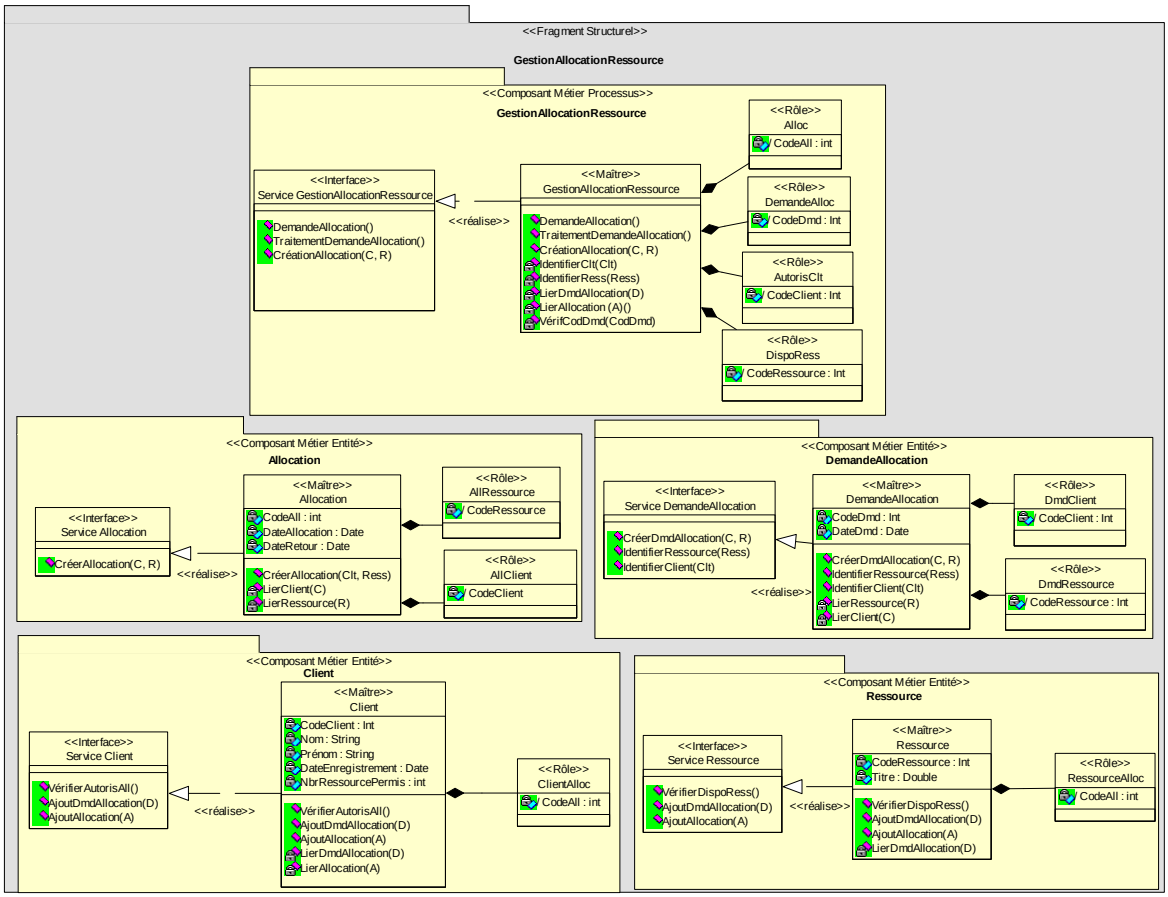


Figure 4-22 : Fragment structurel fixe

Pour illustrer la représentation de la variabilité, nous utilisons le fragment structurel « VérificationRéservation » qui correspond à la variante optionnelle « Vérification Réservation ». Dans cette variante, la variabilité est modélisée par la création de : a) un ***CME*** « Réservation », b) un ***service*** VérificationRéservation (CodRes), c) une opération privée vérifCodRes(CodRes) et d) une **classe *rôle*** vérifRéservation dans le CMP « Gestion Allocation Ressource». Ce rôle permet l'utilisation des services offerts par le CME « Réservation » (IdentifierClt() et IdentifierRess()) (Cf. Figure 4-23).

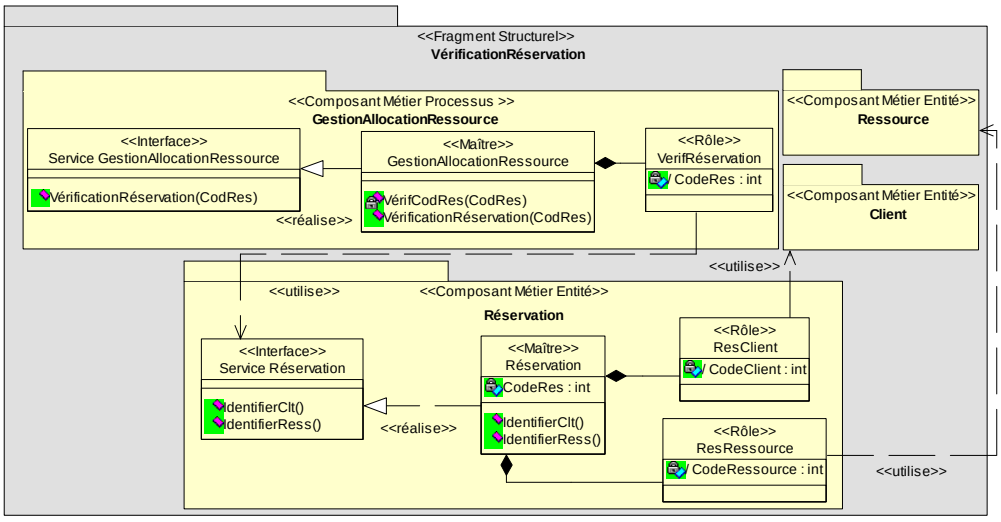


Figure 4-23 : Exemple d'un fragment structurel de type « Variante »

En supposant que le concepteur choisisse la variante « VérificationRéservation », la vue structurelle sera créée par la fusion du fragment fixe (Cf. Figure 4-22) avec le fragment de cette variante (Cf. Figure 4-23). Cette fusion donne lieu à la cartographie illustrée dans la Figure 4-24 et au fragment structurel illustré dans la Figure 4-25 où les classes impactées par la fusion sont colorées différemment. La liste complète des fragments de la vue structurelle est donnée dans l'annexe A.

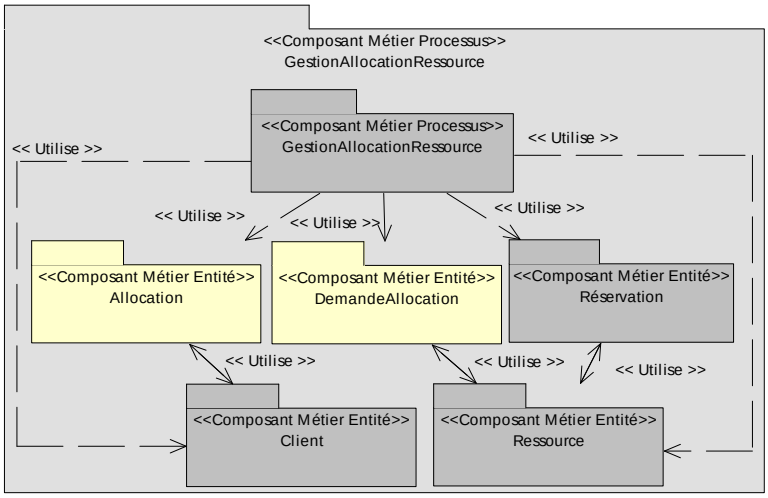


Figure 4-24 : Cartographie après fusion

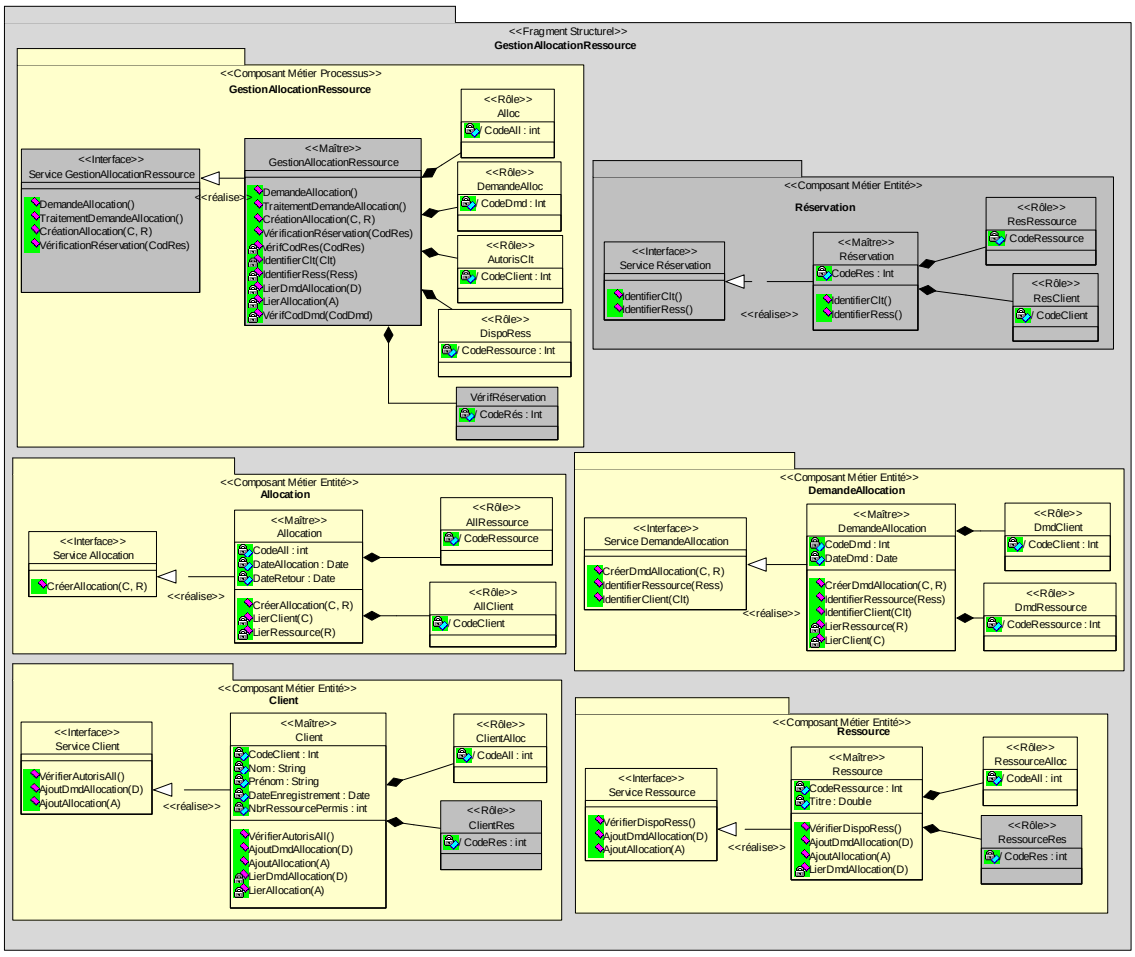


Figure 4-25 : Fragment structurel après fusion

Après avoir décrit le processus d'ingénierie d'un CMP, nous présentons dans la section suivante le méta-modèle de la solution d'un CMP, exprimant ainsi la cohérence inter-vues du développement d'un CMP.

4 Méta-modèle de la solution d'un CMP

4.1 Extensions d'UML

4.1.1 Vue Métier

Les extensions introduites sur le diagramme d'activités sont détaillées dans le chapitre précédent. Nous les rappelons à travers la Figure 4-26.

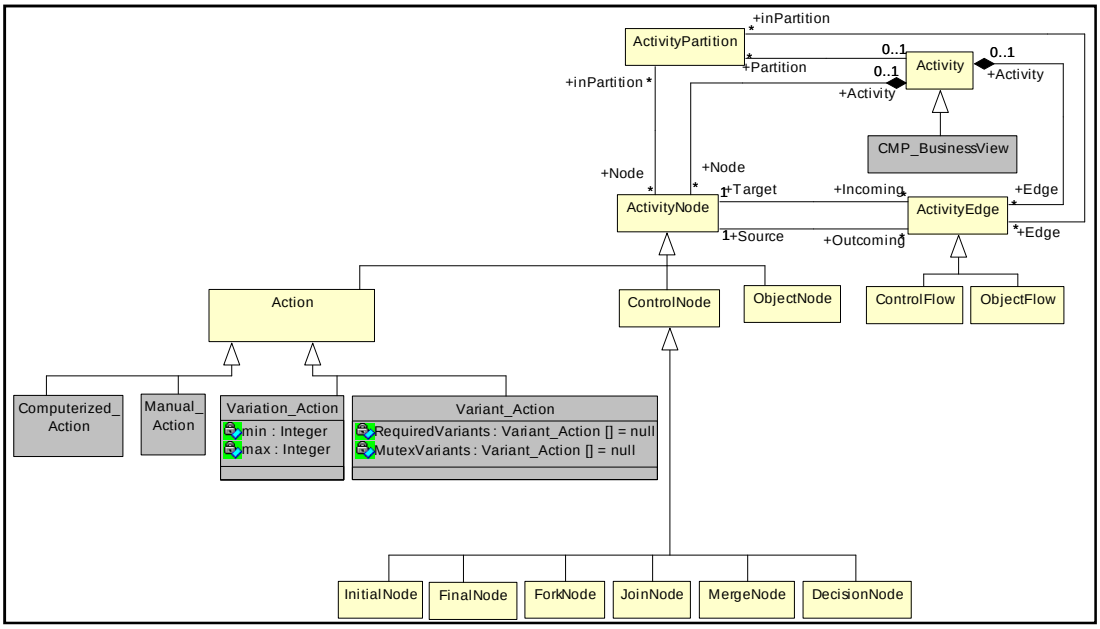


Figure 4-26 : Extension des concepts du diagramme d'activités

4.1.2 Vue fonctionnelle

Les stéréotypes introduits pour la modélisation de la variabilité dans les diagrammes de cas d'utilisation sont définis comme des extensions sur la partie du méta-modèle d'UML 2.0 spécifiant les cas d'utilisation [OMG, 2007]. La Figure 4-27 illustre les méta-classes de cette partie concernées par nos extensions.

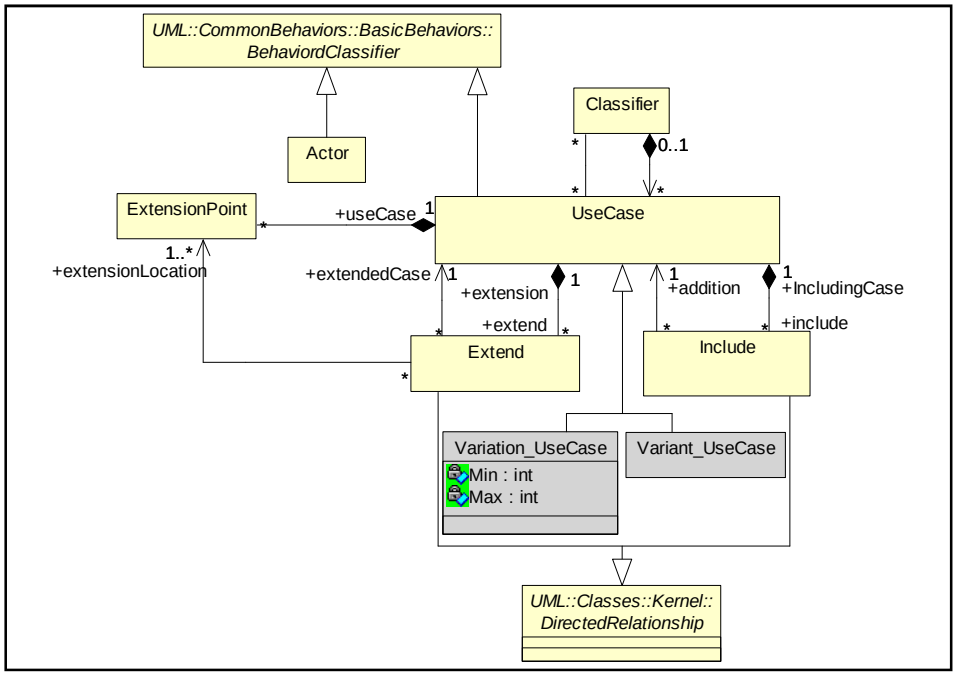


Figure 4-27 : Extension des concepts du diagramme de cas d'utilisation

a. Description des extensions

Classe	**Description**
Variation_UseCase	**Superclasse** : UseCase. **Description** : cette classe modélise un cas d'utilisation de type point de variation. **Attributs** : min et max sont des entiers qui modélisent les cardinalités minimale et maximale pour le choix des variantes. **Notation :** un cas d'utilisation stéréotypé « Variation ».
Variant_UseCase	**Superclasse** : UseCase. **Description** : cette classe modélise un cas d'utilisation de type variante. **Notation :** un cas d'utilisation stéréotypé « Variant ».

Tableau 4-12 : Description des extensions dans un diagramme de cas d'utilisation

b. Expression des contraintes

Une seule contrainte est nécessaire pour la vue fonctionnelle. Elle est donnée de la manière suivante :

Un cas d'utilisation stéréotypé « Variant » doit être inclus dans un et un seul cas d'utilisation stéréotypé « Variation ».

✓ **Formalisation en OCL**

Context Include **inv**: self.addition.isStereotyped ('Variant') **implies** self.includingCase.isStereotyped ('Variation')

4.1.3 Vue dynamique

Dans la vue dynamique, aucune extension n'est introduite, car nous n'utilisons que des concepts des diagrammes de séquence qui existent déjà dans UML. La Figure 4-28 illustre un extrait du méta-modèle UML spécifiant les diagrammes de séquence. Cet extrait met en avant les concepts utilisés, à savoir : les interactions (*Interaction*), les frames *(interactionUse) et* les fragments combinés *(CombinedFragment)*.

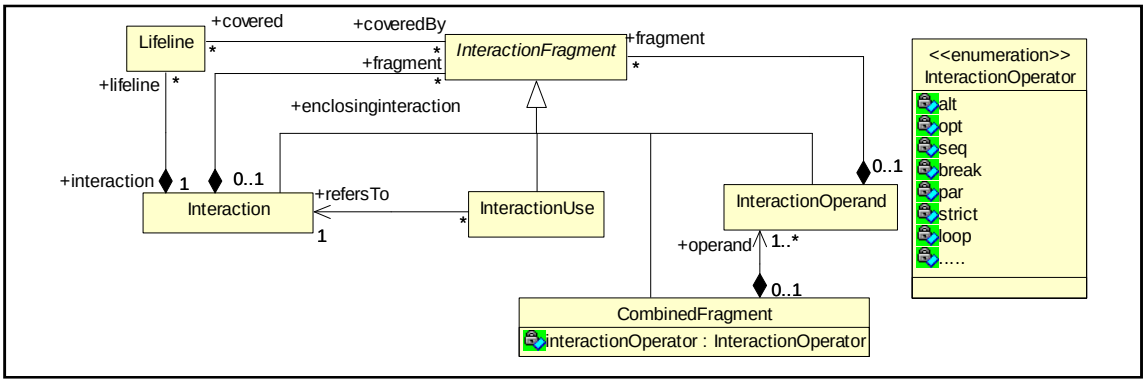


Figure 4-28 : Extrait du méta-modèle UML du diagramme de séquence

4.1.4 Vue structurelle

La Figure 4-29 propose un méta-modèle du modèle Symphony étendu par nos concepts. Dans ce modèle, chaque fragment structurel est composé d'au moins un modèle Symphony décrit par un diagramme de classes. Un modèle Symphony peut être de type « Composant Métier Processus » ou « Composant Métier Entité ». Chaque composant se compose d'une classe *Maître* (éventuellement spécialisée), elle-même composée par des classes *Partie* (éventuellement spécialisées) et d'une ou plusieurs classes *Rôle*. La classe Maître, qui est une classe comportementale, réalise une ou plusieurs interfaces qui représentent les services publiés par le composant. Toutes les classes structurelles maître, partie et rôle admettent des classes rôle et partie. Les classes maître et partie sont spécialisables.

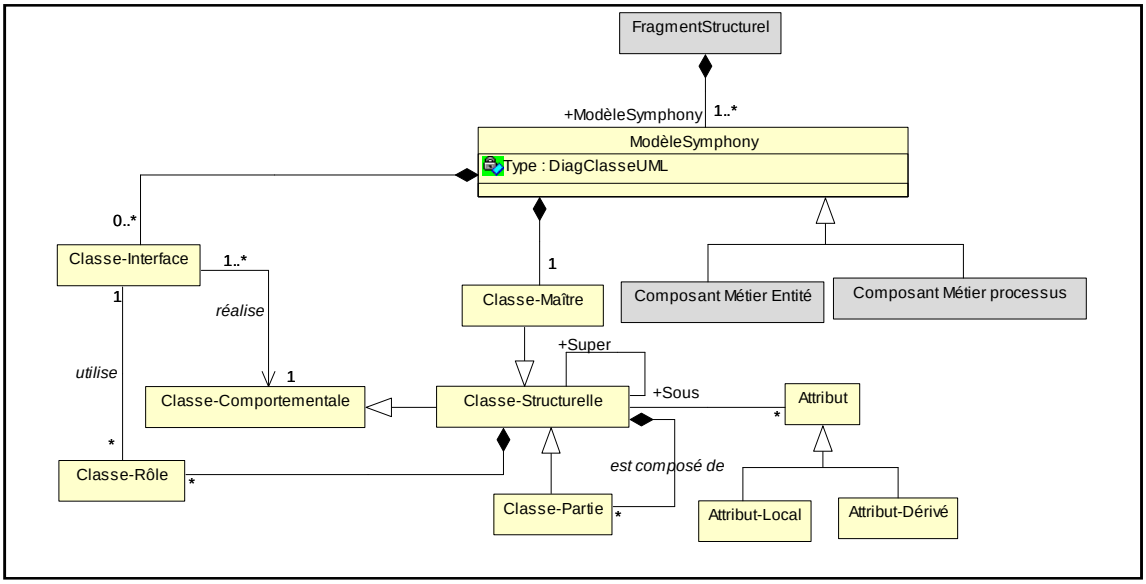


Figure 4-29 : Extension du modèle Symphony

a. Description des extensions

Classe	**Description**
Fragment Structurel	**Description** : cette classe modélise un fragment structurel de type diagramme de classes UML. **Associations :** • *ModèleSymphony* : ModèleSymphony [1,*]: référence l'ensemble des packages Symphony inclus dans le fragment. **Notation :** un package stéréotypé « Fragment Structurel ».
Composant Métier Entité	**Superclasse** : Modèle Symphony. **Description** : cette classe modélise un modèle Symphony de type Composant Métier Entité. **Notation :** un package stéréotypé « Composant Métier Entité ».
Composant Métier Processus	**Généralisation** : Modèle Symphony. **Description** : cette classe modélise un modèle Symphony de type Composant Métier Processus. **Notation :** un package stéréotypé «Composant Métier Processus».

Tableau 4-13 : Description des extensions du modèle Symphony

b. Expression des contraintes

Les contraintes de la vue structurelle sont exprimées sous forme de cardinalités des associations. La seule contrainte supplémentaire concerne les classes structurelles. En effet, toute classe structurelle admet des attributs, mais seule la classe rôle peut admettre des attributs dérivés.

✓ **Formalisation en OCL**

Contexte Classe-Structurelle
inv: **not** self.oclIsTypeOf (Classe-Rôle) **implies** self.attributs -> **not exists** (a|a.ocsIsTypeOf (Attribut-Dérivé))

4.2 Méta-modèle de la solution d'un CMP

La Figure 4-30 illustre le méta-modèle de la solution d'un CMP. La vue métier constitue la vue organisationnelle du CMP constituée par des actions fixes et des actions supportant la variabilité (Variant_Action et Variation_Action). La vue fonctionnelle est réalisée par génération et mise en relation des cas d'utilisation à partir des actions informatisées de la vue métier sous la forme d'un diagramme de cas d'utilisation. Concernant la variabilité, chaque action informatisée de type variation (ou variante) donne lieu à un cas d'utilisation de type variation (ou variante). D'un autre côté, la vue dynamique a pour objectif de préciser la réalisation des éléments de la vue fonctionnelle. Dans un contexte variable, la vue dynamique se doit de préciser comment les comportements associés aux variantes sont organisés et comment ils s'enchaînent. Dans ce sens, et pour maintenir la traçabilité de la variabilité, chaque cas d'utilisation donne lieu à une interaction.

En outre, la spécification structurelle est en grande partie déductible de la spécification fonctionnelle. Puisque la vue fonctionnelle est décomposée selon les cas d'utilisation, il en est

de même pour la vue structurelle : chaque cas d'utilisation donne lieu à un fragment structurel cohérent avec l'interaction définie auparavant.

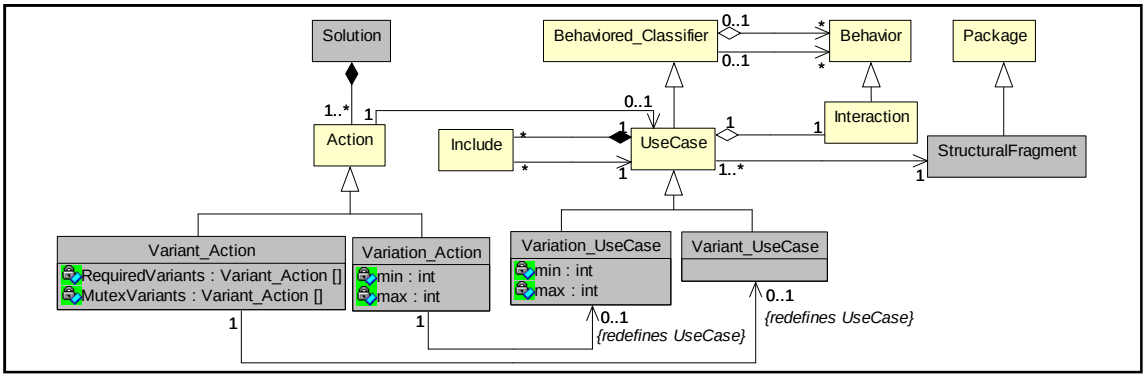


Figure 4-30 : Méta-modèle de la solution d'un CMP

4.3 Vers un profil UML pour les CM

La notion de profil UML est apparue dans le standard UML 1.3 [OMG, 1999], comme un moyen permettant de structurer les extensions UML (valeurs marquées (tagged values), stéréotypes et contraintes). UML est un langage de modélisation à destination d'un grand nombre de domaines d'application. Cependant, chaque domaine a des notions et des besoins particuliers, qu'UML peut supporter par le biais de ses extensions, regroupées en profils UML.

Dans le contexte de représentation de la variabilité pour la modélisation de composants métier, nous avons proposé différents mécanismes pour les quatre vues de développement d'un CM. La Figure 4-31 illustre l'ensemble de ces mécanismes sous forme d'un profil UML pour les CM. Tout stéréotype introduit est représenté par une classe stéréotypée <<stereotype>> spécialisant une méta-classe représentée par une classe stéréotypée <<metaclasse>>.

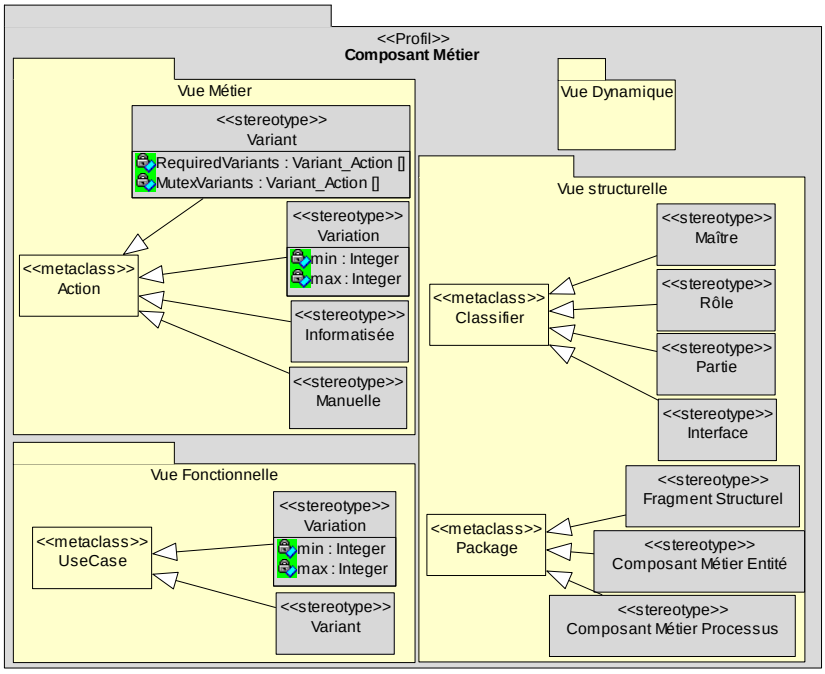


Figure 4-31 : Profil UML pour les composants métier

5 Conclusion

Dans ce chapitre, nous avons présenté nos propositions en termes de spécification d'un processus d'ingénierie de CMP dont la solution est composée de quatre vues de développement organisées autour de l'expression de la variabilité. L'ensemble des extensions apportées aux concepts d'UML est donné sous forme d'un profil UML pour les composants métier. Il nous faut désormais proposer des mécanismes d'organisation des CM dans une bibliothèque de composants ainsi qu'un processus appelé « processus par la réutilisation » capable d'utiliser des composants exprimés selon ce modèle. Ceci fait l'objet du chapitre suivant.

Chapitre 5 : Mise en œuvre et validation

Le chapitre précédent a décrit nos propositions en terme de spécification de CMP composée de quatre vues de développement organisées autour de l'expression de la variabilité. De telles spécifications ne sont utiles que si elles sont organisées dans un environnement pour des futures réutilisations. C'est dans ce but que nous présentons, dans ce chapitre, des mécanismes d'organisation des CMP (§ 1) en proposant un formalisme de documentation ainsi qu'un processus d'imitation (§2) qui assiste l'ingénieur de SI lors de l'imitation d'un CMP. Ce processus est intégré par la suite dans le cycle de vie d'une méthode de développement basée composants en utilisant la méthode Symphony comme référence. Ces aspects sont mis en œuvre par des outils supports (§3). Finalement, nous présentons les résultats des expérimentations élaborées dans le but de valider le modèle de CMP ainsi que les processus proposés dans cette thèse (§ 4).

1 Documentation et organisation des CMP

1.1 Vers un formalisme de documentation de CMP supportant la variabilité

La spécification des composants réutilisables ne doit pas se restreindre seulement à la description des informations effectivement réutilisables, mais aussi à celles requises pour une bonne réutilisation de ces composants. Ces dernières doivent permettre de décider, par exemple, de la pertinence d'utiliser un composant ou un autre au moment de la recherche de ceux-ci.

En outre, les composants doivent être stockés de telle manière qu'ils puissent être intégrés dans un système pour accomplir leur objectif. Cet environnement de stockage et de gestion de composants doit satisfaire les propriétés suivantes :

- Facilité d'utilisation : un client cherchant un composant doit l'obtenir rapidement. Les espaces de stockage des composants doivent être structurés de telle manière que l'utilisateur ne perde pas de temps à les chercher.
- Documentation : la documentation doit être précise et bien organisée, de sorte qu'un utilisateur puisse rapidement trouver les informations appropriées sur un élément spécifique.

Il est donc nécessaire de proposer un formalisme permettant de documenter, d'organiser et de décrire la solution d'un CMP.

L'étude de différents formalismes de description des CMP permet de souligner deux principales faiblesses :

- Absence d'une hiérarchisation de composants par domaines métier et domaines fonctionnels en mettant en évidence une arborescence de réutilisation de composants qui peut faciliter la réutilisation par analogie.
- Absence d'éléments de gestion de la variabilité. Un composant doit offrir plusieurs réalisations possibles : sa solution variable doit être documentée afin d'assister le concepteur d'applications dans le choix d'une ou plusieurs variantes.

Cette section est consacrée à la proposition d'un formalisme de CMP que nous appelons CMP-SIGMA. Nous soulignons les objectifs de ce formalisme (§ 1.2), sa structure (§ 1.3) et sa description (§ 1.4).

1.2 Objectifs du formalisme CMP-SIGMA

1.2.1 Classification des CMP par domaines métier et fonctionnels

Les CMP sont classifiés selon leur spécificité. La spécificité relie l'applicabilité d'un composant à différents domaines. Ainsi, les composants peuvent être marqués fonctionnels (si le problème traité est fréquent dans de nombreux domaines d'applications) ou métier (si le problème traité est fréquent dans un domaine métier ou dans une entreprise particulière). Cela permet une réutilisation par analogie.

1.2.2 Hiérarchisation des CMP

La hiérarchisation des CMP est réalisée par la création de liens d'abstraction et de concrétisation entre CMP de différents niveaux d'abstraction et qui répondent au même problème.

1.2.3 Documentation de la variabilité

La spécification d'un CMP supportant la variabilité exige une bonne documentation de cette dernière, en donnant toutes les informations nécessaires pour contrôler les décisions prises lors de la réduction de la variabilité au moment de la réutilisation d'un composant.
La documentation de la variabilité vise à : a) discriminer les différentes variantes spécifiées dans la solution par l'introduction de leurs contextes de réutilisation et b) expliciter l'intention qui a été à l'origine de la définition des différents points de variation. Ces éléments doivent fournir un guide à l'ingénieur d'une application basée composants, et doivent donc être clairs facilitant ainsi la réduction d'un composant.

1.3 Structure du formalisme CMP-SIGMA

Le formalisme CMP-SIGMA (Cf. Figure 5-1) est une adaptation du formalisme P-SIGMA [Conte *et al*., 2001] proposé dans l'équipe SIGMA pour documenter les patrons de conception. CMP-SIGMA est structuré en trois parties composées d'une ou plusieurs rubriques :

1. Une partie ***Interface*** composée de quatre rubriques (*Identifiant*, *Problème*, *Domaine métier*, *Domaine fonctionnel*) contenant les informations nécessaires sur l'abstraction du CMP. Ces rubriques constituent le point d'entrée du composant et en facilitent l'accès et la sélection.
2. Une partie ***Solution*** composée de quatre rubriques (*vue métier*, *vue fonctionnelle*, *vue dynamique*, *vue structurelle*) représentant la partie effectivement réutilisable du CMP. Chaque rubrique est composée d'un champ de type diagramme UML étendu par les concepts de variabilité. La rubrique vue métier est composée d'un champ supplémentaire qui sert à documenter la variabilité représentée dans la solution du CMP.
3. Une partie ***Relation*** composée de deux rubriques (Concrétise, Abstrait) qui sert à organiser un CMP au sein d'un système de CMP.

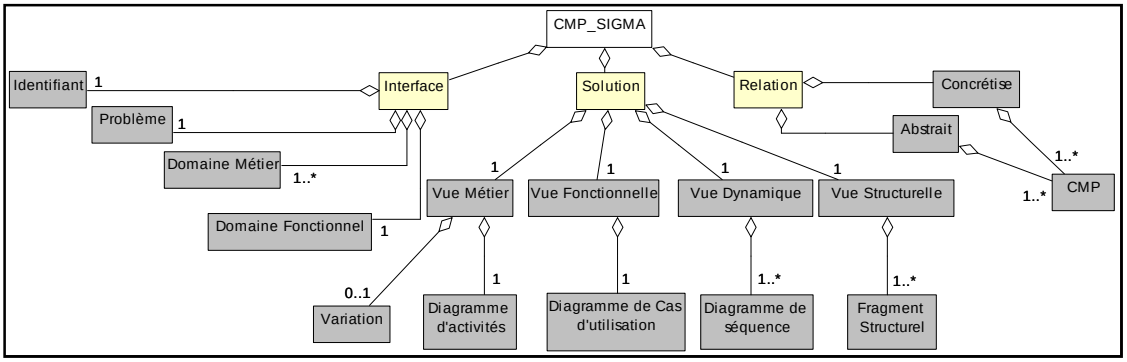


Figure 5-1 : Formalisme CMP-SIGMA

1.4 Description du formalisme

1.4.1 Partie « Interface »

Les rubriques de la partie interface constituent la partie visible du composant et facilitent sa sélection. Le Tableau 5-1 représente les 4 rubriques de cette partie. Des exemples de ces rubriques sont illustrés dans le Tableau 5-2.

Rubrique	Description	Champs
Identifiant	Nom du composant.	Textuel
Problème	Définit l'intention et le problème résolu par le CMP.	Textuel
Domaines Métier	Secteurs d'application du CMP identifiés au moment de sa conception.	{Mots clés}
Domaine Fonctionnel	Description abstraite des fonctionnalités primaires du CMP.	Mot clé

Tableau 5-1 : Rubriques de la partie Interface de CMP-SIGMA

Rubrique	Exemple
Identifiant	« Gestion Allocation Ressource ».
Problème	Permet de gérer l'allocation de ressource avec gestion de réservation, sans gestion de réservation, pour une ressource consommable ou non consommable.
Domaines Métier	Gestion de Bibliothèque. Gestion d'Hôtel. Gestion de location de voiture.
Domaine Fonctionnel	Allocation de Ressource.

Tableau 5-2 : Exemples des rubriques de la partie Interface de CMP-SIGMA

1.4.2 Partie « Solution »

Les rubriques de la partie solution constituent les quatre vues de développement d'un CMP, à savoir : sa vue métier, sa vue fonctionnelle, sa vue dynamique et sa vue structurelle. Ces vues de développement ont été déjà détaillées dans les chapitres précédents.

La vue métier étant la vue qui contrôle la variabilité représentée dans le CMP, elle comporte aussi une documentation de cette variabilité. En effet, le champ variation met en évidence la motivation ou l'intention à l'origine de la définition de chaque point de variation ainsi que des informations qui aident à la prise de décision, entre variantes, d'un même point de variation.

Point de variation	**Motivation**	**Variantes**	**Éléments de décision**
Modification État Ressource	Différencier la gestion de ressources consommables et non consommables	Suppression ressource	Ressource consommable qui n'exige pas un retour
		Mise à jour état ressource	Ressource non consommable qui exige un retour après l'allocation

Tableau 5-3 : Exemple de documentation de la variabilité

Le Tableau 5-3 représente un exemple de documentation du point de variation *Modification État Ressource* extrait du CMP « Gestion Allocation Ressource ».

1.4.3 Partie « Relation »

L'objectif de la partie relation du formalisme est de relier entre eux les CMP associés au même problème de conception. Dans ce but, nous utilisons deux relations principales : « Abstrait » et « Concrétise ». Ces deux relations étant inverses, c'est pour cette raison que nous ne décrivons que la relation « Abstrait » dans le Tableau 5-4 et le Tableau 5-5.

Relation	***Objectif***	***Notation***
Abstrait	Permet de sélectionner les CMP qui peuvent être obtenus par réduction ou raffinement d'un CMP donné. Cette relation permet de construire une hiérarchisation de CMP et d'assurer que des CMP de même domaine fonctionnel sont regroupés.	CMP_1 *abstrait* CMP_2 ⇔ $(CMP_1, CMP_2)_{\text{Abstrait}}$ ⇔ $(CMP_2, CMP_1)_{\text{Concrétise}}$
Exemple	Un CMP « Gestion Allocation Ressource » *abstrait* les CMP « Gestion Emprunt » et « Gestion Location Voiture »	

Tableau 5-4 : Description de la relation « Abstrait »

Relation	**Définition**	**Propriétés**
Abstrait	Si CMP_1 **abstrait** CMP_2, alors : - Le problème de CMP_2 doit être plus spécifique du problème de CMP_1. - Le Domaine Fonctionnel (DF) de CMP_2 est identique au Domaine Fonctionnel de CMP_1. - Les Domaines Métier (DM) de CMP_2 sont inclus dans la liste des Domaines Métier de CMP_1. - La solution de CMP_2 peut être une réduction de la solution de CMP_1.	Non réflexive Antisymétrique Transitive

Tableau 5-5 : Définitions et propriétés de la relation « Abstrait »

D'un autre côté, il existe des relations qui servent à organiser les CME liés à un CMP. Nous distinguons plus particulièrement la relation « Utilise » déjà utilisée dans le modèle Symphony (Cf. Tableau 5-6).

Relation	***Objectif***	***Notation***
Utilise	Permet de localiser les CME qui correspondent à une partie de la solution d'un CMP.	CMP_1 *utilise* CME_2 ⇔ $(CMP_1, CME_2)_{\text{Utilise}}$
Exemple	Un CMP « Gestion Allocation Ressource » *utilise* les CME « Allocation » et « Réservation ».	

Tableau 5-6 : Définitions et propriétés des relations intra-CMP

2 Processus d'imitation d'un CMP supportant la variabilité

2.1 Processus d'imitation : concepts de base

L'**imitation** est le mécanisme de « duplication » qui permet d'extraire la solution d'un CMP et de l'appliquer dans un SI en construction. Nous décrivons ici des concepts liés à la manipulation des CMP lors de la conception des SI par réutilisation. Le processus d'imitation est composé des phases suivantes (Cf. Figure 5-2).

- *Recherche* : la recherche d'un composant est directement liée aux informations employées pour sa description [Khayati, 2003]. La description que nous proposons pour un composant inclut fondamentalement les éléments utilisés pour faciliter sa recherche. Ces éléments dépendent principalement de la partie interface. De plus, l'organisation choisie pour soutenir la communication entre les CMP (partie relation) aide également à accélérer l'activité de recherche.
- *Réduction :* cette activité consiste à choisir les variantes utiles spécifiées dans un CMP.
- *Adaptation* : un CMP est un fragment conceptuel qui dispose de propriétés génériques spécifiées dans le but d'être adaptées à un contexte particulier. Bien souvent, on définit l'adaptation comme suit [Rieu *et al.*, 1999] : *Adapter = (renommer | ajouter | supprimer propriété)**. Dans notre cas, étant donné que la phase de réduction permet de supprimer les variantes inutiles dans le CMP, nous restreignons la phase d'adaptation à un simple renommage des propriétés du CMP selon le contexte de réutilisation.
- *Intégration* **:** l'intégration vise à assembler les solutions obtenues après la réduction des CMP choisis, afin de concevoir un système complet, ou au moins un fragment cohérent du système final. Les composants proposés étant de nature processus, leur granularité est importante, et engendre le fait que l'étape d'intégration reste en retrait.

Notre approche s'articule autour du principe de variabilité, c'est pour cette raison que nous détaillons plus particulièrement la phase de réduction de la variabilité. Cela fait l'objet de la section suivante.

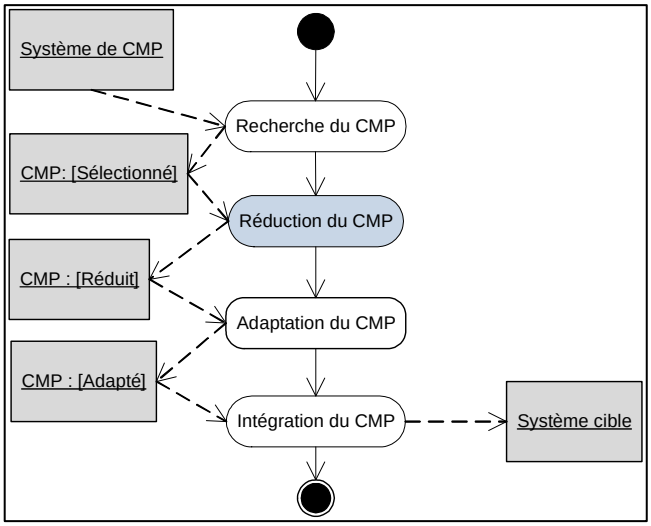


Figure 5-2 : Processus d'imitation d'un CMP

2.2 Réduction du CMP : description des étapes

La réduction d'un CMP (Cf. Figure 5-3) est réalisée selon les étapes suivantes :

- *Sélection des variantes* : il s'agit de sélectionner les variantes de chaque point de variation à l'aide de la documentation fournie pour chaque variante. Cette étape produit un modèle des variantes choisies qui sert à enregistrer le choix du concepteur.
- *Vérification des contraintes de dépendance* : dans cette phase une vérification des contraintes de dépendance est établie dans le but d'assurer une réduction cohérente. En cas de conflit, une nouvelle sélection des variantes est exigée.
- *Réduction du CMP* : lors de cette étape, le CMP est réduit en conservant les variantes enregistrées dans le modèle des variantes sélectionnées. La réduction est réalisée en appliquant les règles que nous avons introduites dans les deux chapitres précédents. Le concepteur effectue sa réduction à partir de la vue métier, les trois autres vues sont automatiquement réduites en utilisant les mécanismes de traçabilité de la variabilité.

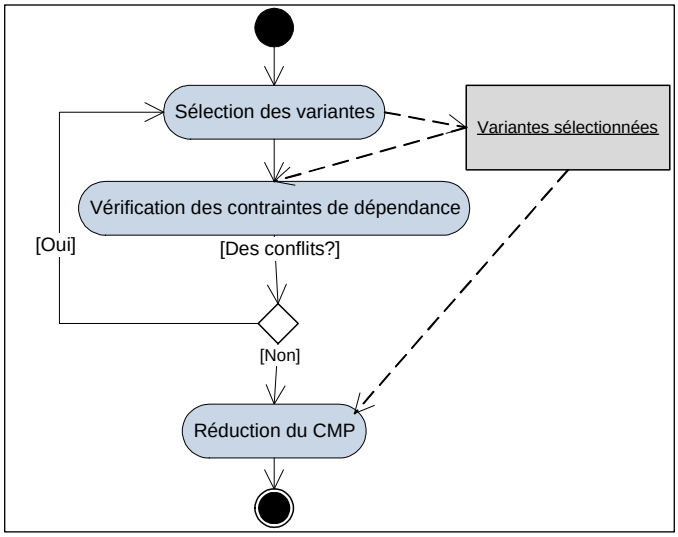


Figure 5-3 : Enchaînement des étapes de la phase Réduction du CMP

Un processus d'imitation d'un CMP n'est utile que s'il est intégré dans un processus de développement de SI à base de composants. Ce constat est traité dans la section suivante.

2.3 Méthode Symphony : extension par les concepts de réutilisation

2.3.1 La méthode symphony

Symphony est une démarche de développement de SI. Sa particularité est la séparation des besoins fonctionnels de ceux techniques et ceci dès le début du cycle de vie des applications. Les deux aspects fonctionnels et techniques sont mis en évidence en adoptant un modèle de cycle de vie en Y. Nous rappelons par la Figure 5-4 le cycle de vie de la méthode Symphony déjà détaillée dans le chapitre état de l'art.

Dans la version actuelle de la démarche Symphony [Hassine, 2005], un apport n'est pas achevé et concerne la prise en compte de la réutilisation dans le cycle de vie de la méthode. Il s'agit de doter Symphony d'un « **processus par la réutilisation »** adapté à son processus de

développement de système d'information. D'un autre côté, la pratique a montré que le modèle d'objet métier utilisé dans Symphony était très orienté « Entité », alors qu'un OM orienté « **Processus** » nous parait plus adapté aux concepts pilotes de la méthode, particulièrement la prise en charge de l'aspect métier. L'objectif de ce paragraphe est dans ce sens, d'étendre Symphony grâce à notre processus d'imitation de CMP.

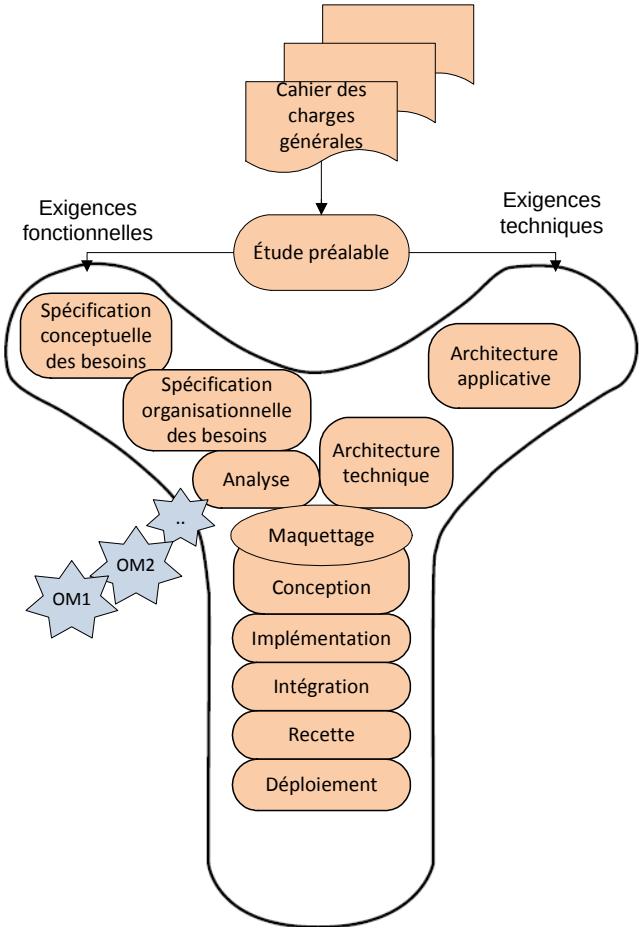


Figure 5-4 : Cycle de vie de Symphony

2.4 Extension de Symphony par la réutilisation

2.4.1 Symphony pour l'ingénierie d'un SI basé composant : concepts de base

Notre extension de la méthode Symphony (Cf. Figure 5-5) concerne principalement l'extension de la branche fonctionnelle. L'extension est basée sur l'utilisation du processus d'imitation pour réutiliser un CMP requis. Ainsi, dans la branche fonctionnelle, des phases seront inutiles si un CMP a pu être identifié comme candidat à la réutilisation, il sera en effet imité est transformé directement en des objets métier qui peuvent être intégrés dans la phase de maquettage. En outre, nous utilisons le processus d'imitation après la phase d'étude préalable dont le but de comprendre les finalités attendues du processus, finalités exprimées sous forme de besoins métier qui peuvent être équivalents au processus lui même ou aux différents processus métier qui le composent. À l'issue de la phase d'étude préalable, il est donc possible de rechercher un CMP candidat à la réutilisation. Si la sélection réussie, le processus d'imitation se substitue aux phases de spécification et d'analyse.

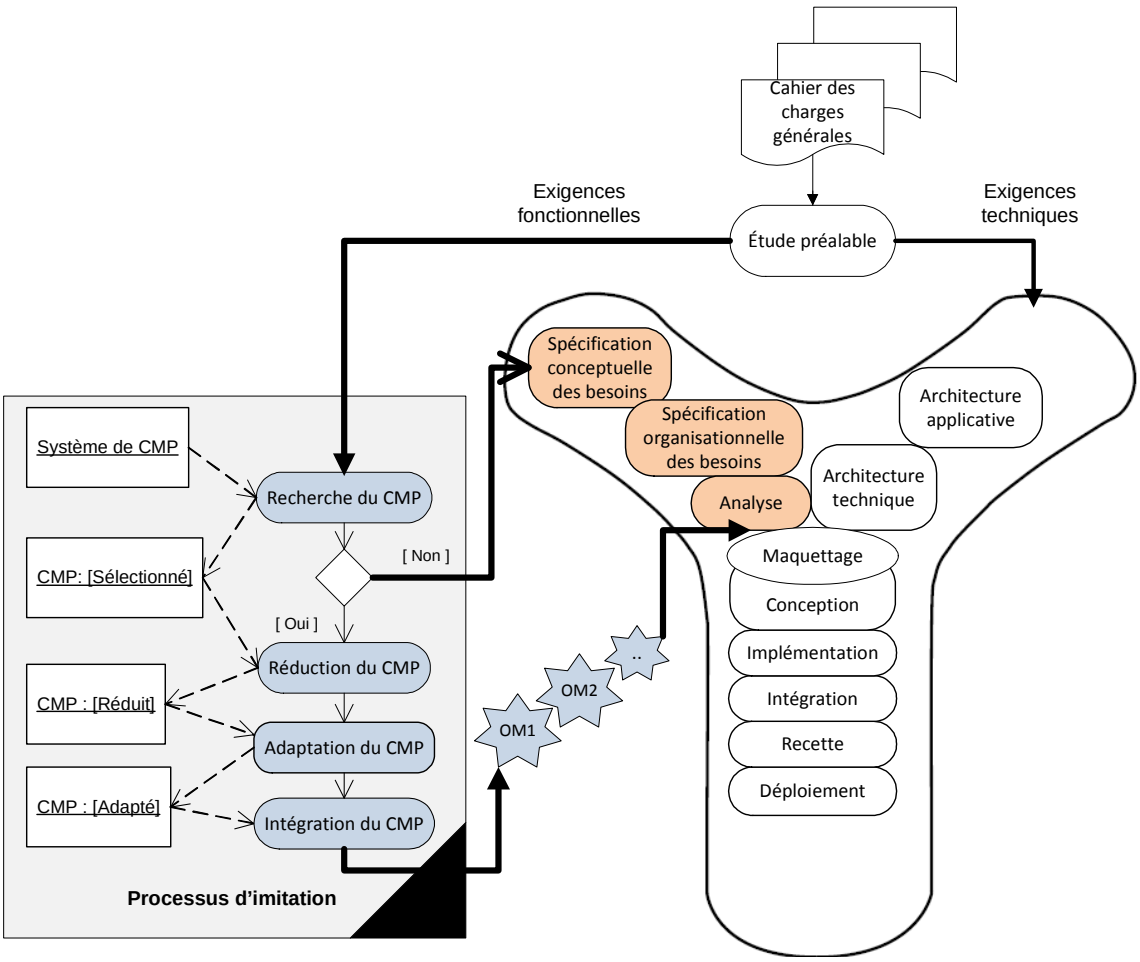


Figure 5-5 : Extension de Symphony

Pour illustrer l'extension de la méthode Symphony, nous présentons dans la section suivante une étude de cas de réutilisation d'un CMP.

2.4.2 Étude de cas : Gestion d'emprunt dans une bibliothèque

a) Cahier des charges

Le cahier des charges est composé d'un ensemble d'exigences métier établies à partir des besoins exprimés par les responsables de la bibliothèque (Cf. Tableau 5-7).

Notre bibliothèque contient un certain nombre d'œuvres (livres, films, dvd, cd...) disponibles à la consultation ou à l'emprunt. L'objectif est d'améliorer la gestion de ces ressources. Cette amélioration concerne en premier lieu la prise en charge informatisée de la plupart des activités du processus d'emprunt qui permet un allégement considérable de la charge de travail et une plus grande souplesse par rapport au traitement manuel de l'information. L'emprunt d'une œuvre obéit à certaines règles : - La demande d'emprunt d'un livre doit être obligatoirement effectuée par envoi de formulaire électronique. - Le bibliothécaire crée l'emprunt lorsque l'adhérent vient récupérer son livre. - Un adhérent peut demander le renouvellement d'un emprunt…

Tableau 5-7 : Extrait de Cahier des charges de l'étude de cas « Gestion emprunt »

b) Phase d'étude préalable : modélisation métier

La phase d'étude préalable dans Symphony précède le processus de développement. Elle consiste à faire une première reformulation et expression des besoins fonctionnels sous forme de processus métier. Cette phase de modélisation métier est nécessaire avant tout effort de développement du système. Les artefacts en sortie de cette phase sont le modèle métier ainsi qu'une description en langage naturel du PM (Cf. Tableau 5-8).

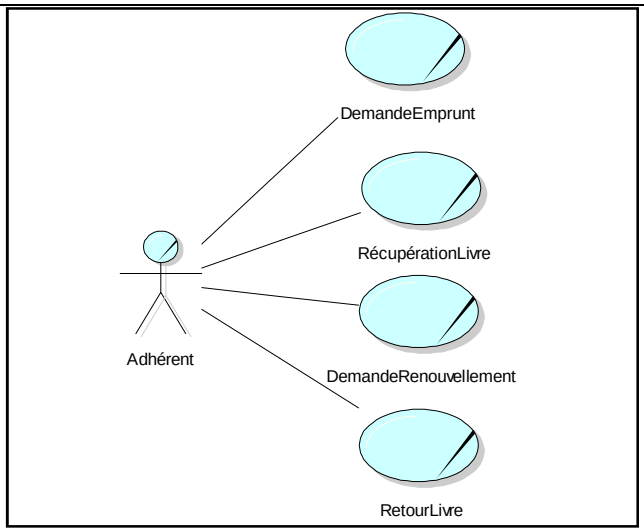
Modèle Métier	
Scénario	**1. Demande d'emprunt** - Un adhérent doit faire une demande d'emprunt par le biais d'un formulaire électronique - Le bibliothécaire traite la demande en vérifiant la disponibilité du livre - Si le livre est disponible, il informe l'adhérent de la possibilité de l'emprunt et lui demande de passer le récupérer **2. Récupération livre** L'adhérent passe pour récupérer le livre Le bibliothécaire doit : - Saisir et vérifier le code de l'adhérent - Créer l'emprunt et mettre à jour l'état du livre en le rendant emprunté **3. Demande de renouvellement** - Un adhérent peut créer une demande de renouvellement de son emprunt par envoi d'un formulaire électronique - Le bibliothécaire vérifie la disponibilité du livre à la date demandée et met à jour l'emprunt **4. Retour livre** L'adhérent passe pour rendre le livre Le bibliothécaire doit : - Saisir le code d'emprunt - Mettre à jour l'état du livre en le rendant disponible et confirmer la restitution

Tableau 5-8 : Modèle métier et description du PM

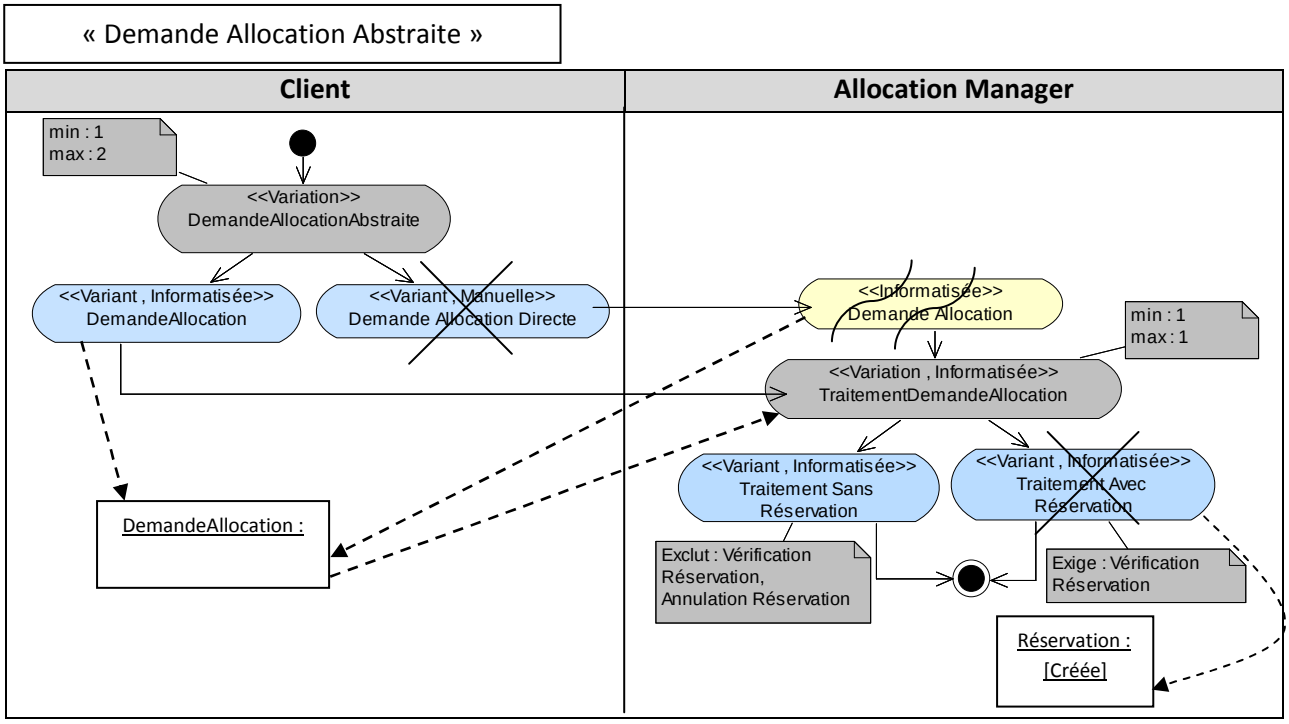
c) Imitation d'un CMP

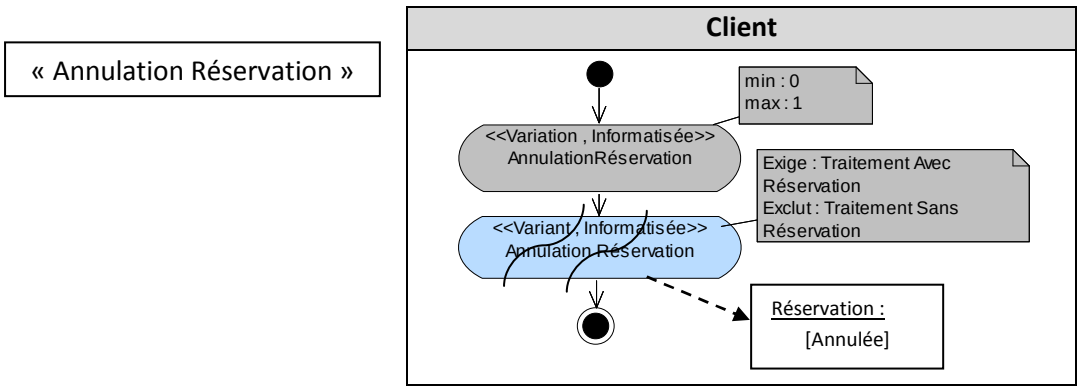
✓ *Recherche et sélection du CMP candidat à la réutilisation*

La recherche et la sélection d'un CMP reposent sur les propriétés utilisées pour décrire sa partie interface. Par exemple, la rubrique domaine métier du CMP « Gestion Allocation Ressource » (i.e. Gestion de Bibliothèque, Gestion d'Hôtel, Gestion de location de voiture) permet de considérer ce CMP comme candidat à la réutilisation pour le développement du PM « Gestion Emprunt ».

✓ *Réduction du CMP*

La réduction de la variabilité introduite dans le CMP est réalisée en se basant sur les règles métier identifiées lors de la phase d'étude préalable ainsi que sur la documentation de la variabilité intégrée dans la spécification du CMP. Les variantes non sélectionnées par l'ingénieur de SI sont illustrées dans la Figure 5-6 par une croix, les activités (variantes ou non) supprimées automatiquement par application des contraintes de dépendances, sont illustrées par des traits en arc. La réduction est achevée au niveau de la vue métier du CMP, les autres vues sont automatiquement réduites.





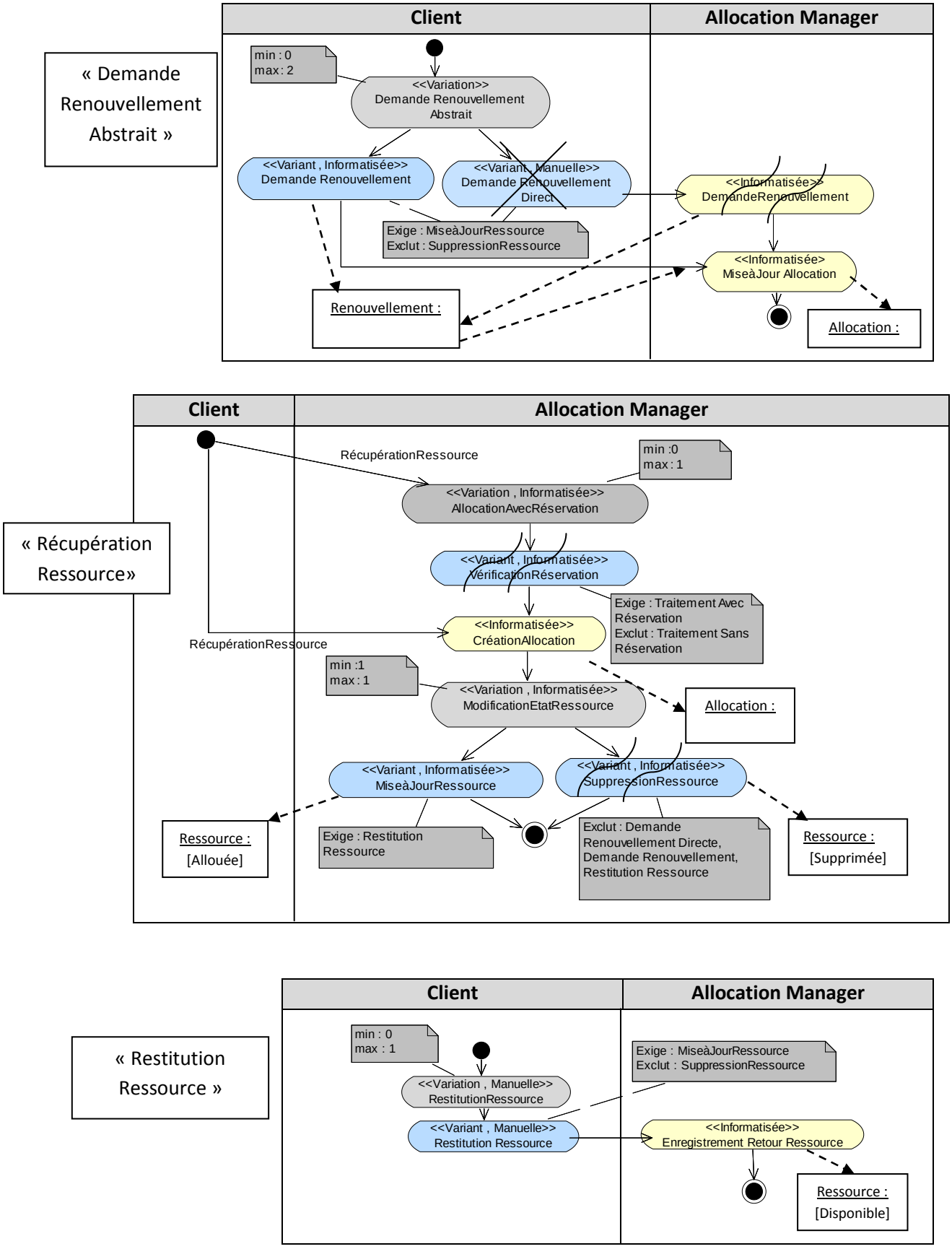


Figure 5-6 : Réduction de la variabilité

✓ *Adaptation du CMP*

Il s'agit essentiellement de renommer les propriétés du CMP réutilisé en se basant sur le vocabulaire lié au domaine du PM en cours de développement. La vue métier réduite et adaptée du CMP « Gestion Allocation Ressource » est illustrée dans la Figure 5-7.

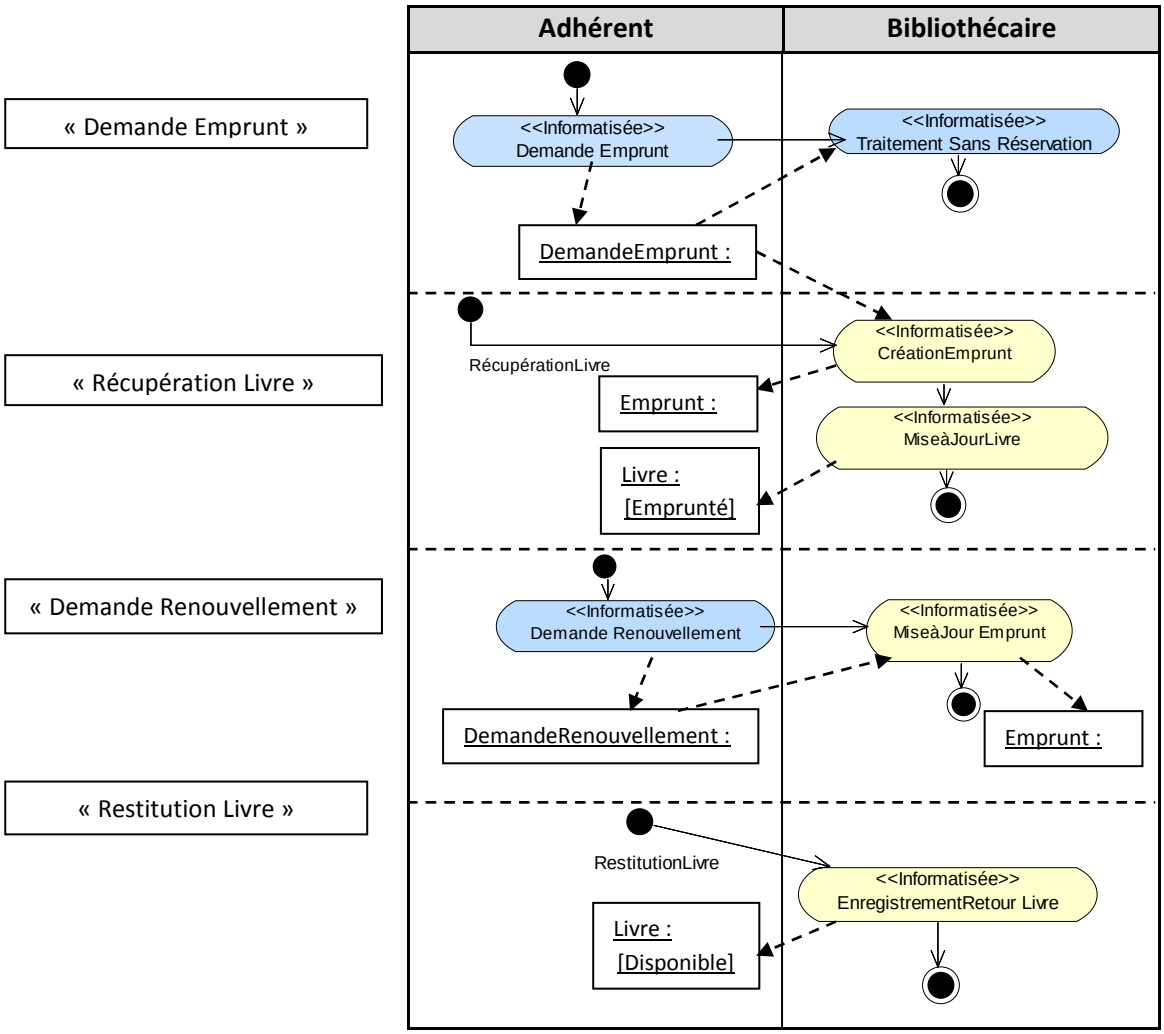


Figure 5-7 : Vue métier réduite et adaptée

✓ *Intégration du CMP*

L'intégration vise à assembler les solutions fournies par les CMP sélectionnés afin de pouvoir concevoir un système complet. Dans l'exemple utilisé, le besoin en matière de réutilisation se restreint à l'utilisation d'un seul composant. Étant donné que le CMP réutilisé est de granularité importante, l'étape d'intégration reste en retrait.

2.5 Impacts de la réutilisation sur le cycle de vie de Symphony

L'extension de Symphony par les concepts de la réutilisation a un impact sur la qualité, le coût et la productivité du système conçu. L'apport du CMP réutilisé en terme de fiabilité est simple : il s'agit d'un composant éprouvé, car déjà utilisé par d'autres. Il dispose d'une maintenance de meilleure qualité et plus efficace, et c'est ce qui justifie, entre autres, une raison de son choix.

La réutilisation est aussi essentielle pour diminuer le coût de développement, parce que réutiliser prend moins de temps que de réinventer. Ainsi, la réutilisation peut réduire de manière drastique les coûts de développement des produits et donc augmenter la productivité de l'entreprise.

En revanche, le développement d'un composant réutilisable souffre d'un coût important car il est nécessaire de :

- le concevoir et le réaliser de manière rigoureuse, notamment en ce qui concerne la documentation.
- le maintenir, car personne ne veut utiliser un composant qui n'est plus maintenu.
- le stocker de manière optimale pour que sa réutilisation soit facilitée.

Afin d'assister, d'une part, l'ingénieur de CMP à documenter et organiser ses CMP, et d'autre part, l'ingénieur de SI à réduire la variabilité, nous avons proposé des outils supports. Cela fait l'objet de la section suivante.

3 Mise en œuvre

3.1 AGAP : Outil support de l'organisation des CMP

L'outil AGAP (Atelier de Gestion et d'Application de Patrons) est un environnement permettant la réutilisation dans l'ingénierie des systèmes d'information. Cet outil a été développé par l'équipe SIGMA afin de démontrer le potentiel des patrons dans l'ingénierie des systèmes d'information et d'apporter des réponses aux limites des produits existants.

Deux modules coopèrent au cœur d'AGAP et sont utilisés par les deux principaux acteurs de l'atelier : l'ingénieur de patrons et l'ingénieur d'applications. D'autre part, AGAP est en mesure de gérer simultanément plusieurs formalismes et systèmes de patrons. L'un des grands avantages d'AGAP est de considérer les formalismes distinctement des systèmes de patrons. Cela permet ainsi à plusieurs systèmes de patrons de pouvoir utiliser le même formalisme ou bien de construire de nouveaux formalismes à partir de ceux existants. De ce fait, l'utilisation d'AGAP pour l'organisation des CMP est possible. En effet, la possibilité de créer de nouveaux formalismes dans AGAP nous permet d'intégrer le formalisme CMP-SIGMA et d'y appliquer différentes actions de gestion de CMP (créer, modifier, valider, …).

3.1.1 AGAP : Architecture fonctionnelle

La Figure 5-8 illustre l'utilisation d'AGAP dans le contexte de l'ingénierie de CMP et celle des SI à base de composants.

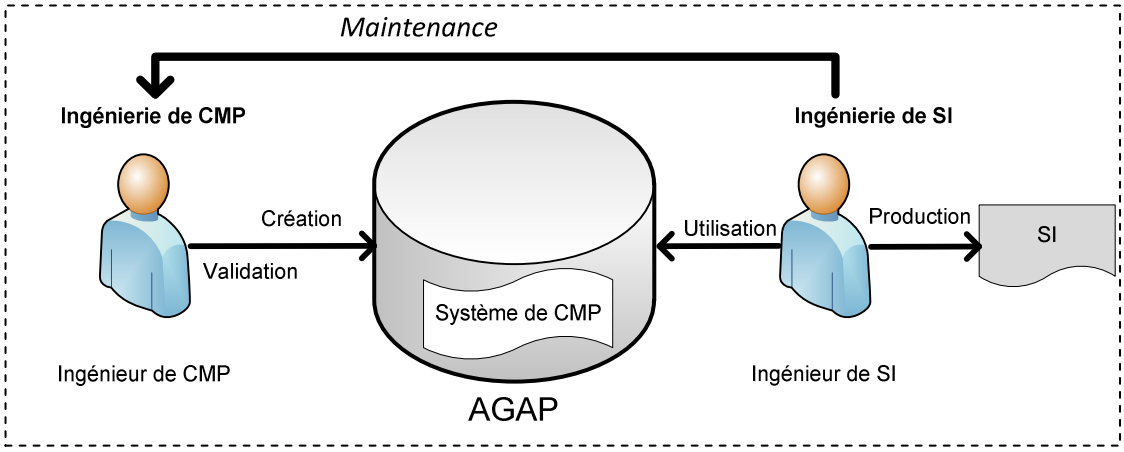


Figure 5-8 : AGAP, cycle de vie d'un système de CMP

L'objectif d'un *ingénieur de CMP* est de constituer un ou plusieurs systèmes de CMP (ensemble cohérent et structuré de CMP) validés et de les mettre à disposition des ingénieurs de SI. Trois phases composent le cycle de vie d'un système de CMP :

- *Création* : l'ingénieur de CMP analyse le problème à résoudre et conçoit de nouveaux CMP (organisés en systèmes), avec création d'un nouveau formalisme le cas échéant ;
- *Validation* : pendant cette phase, l'ingénieur de CMP teste la validité des solutions offertes par les CMP et vérifie l'intégrité et la cohérence des interactions entre CMP (relations) au sein du système de CMP ;
- *Utilisation et maintenance* : le système de CMP devient prêt pour d'éventuelles réutilisations dans des SI. Au cours de cette phase, l'ingénieur de SI peut détecter des anomalies ou bien être confronté à des problèmes nouveaux. Dans ce cas, il peut formuler des demandes vis-à-vis de l'ingénieur de CMP pour lui faire part d'anomalies ou de nouveaux problèmes à résoudre.

3.1.2 Organisation des CMP dans AGAP

Grâce à son méta-modèle, AGAP permet de gérer les formalismes de manière générique avec une structure hiérarchique à deux niveaux : rubrique et champ. La Figure 5-9 illustre une adaptation du méta-modèle d'AGAP [Tastet, 2004] dans le contexte des CMP.

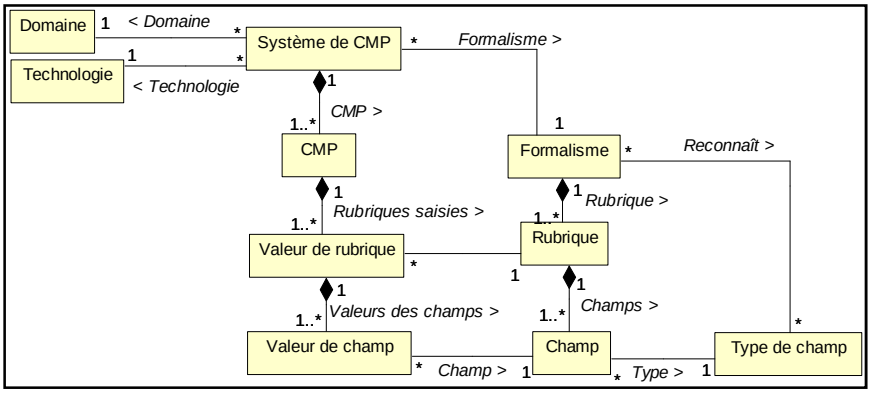


Figure 5-9 : Adaptation du méta-modèle d'AGAP

a) Création du formalisme CMP-SIGMA

Avant d'organiser les CMP dans AGAP, il faut tout d'abord créer le formalisme adapté à ces composants. La Figure 5-10 illustre la création du formalisme CMP-SIGMA.

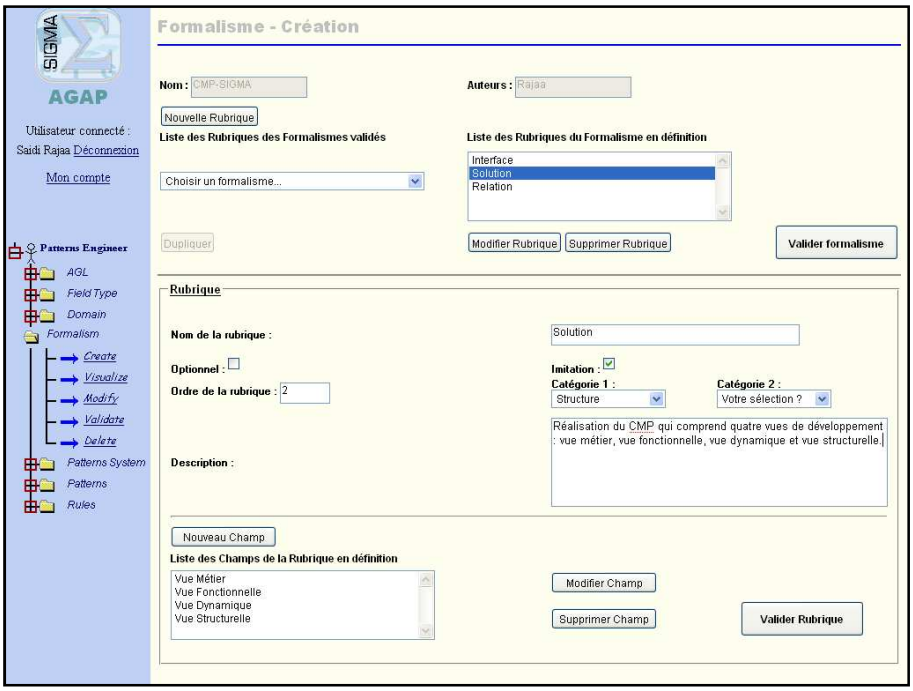


Figure 5-10 : Formalisme CMP-SIGMA sur AGAP

b) Visualisation des CMP

La Figure 5-11 présente un extrait de la visualisation du CMP « Gestion Allocation Ressource » illustré dans le chapitre précédent.

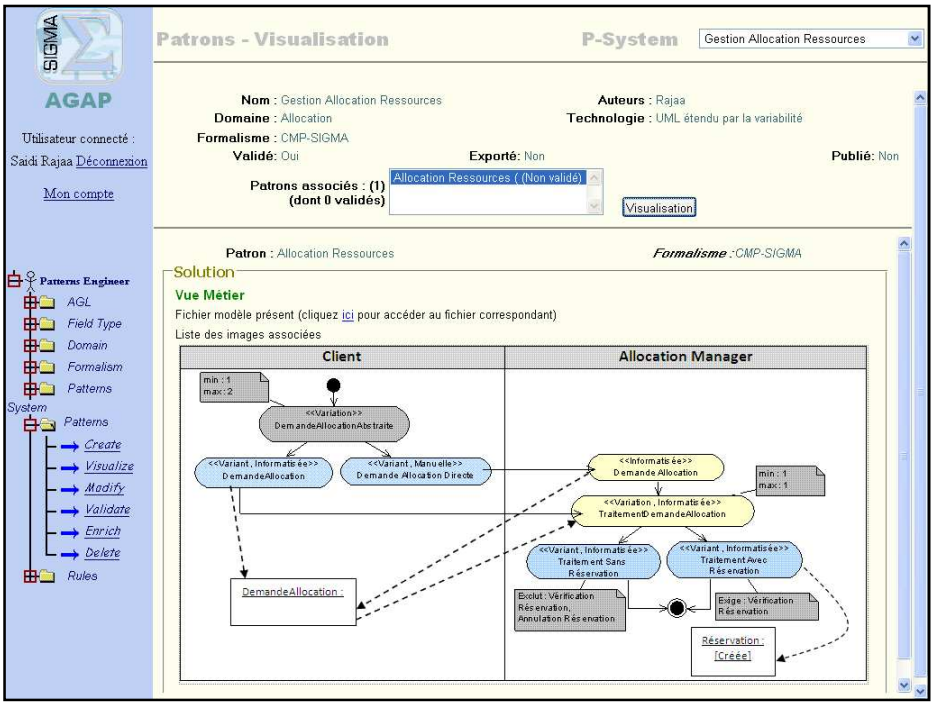


Figure 5-11 : Extrait de la partie solution d'un CMP sous AGAP

L'objectif d'un *ingénieur de SI* dans AGAP est de rechercher et de sélectionner des CMP pour modéliser et concevoir un SI. Ces activités s'appuient sur un ou plusieurs systèmes de CMP disponibles et validés dans l'atelier AGAP. En revanche, lors de l'imitation des CMP, une phase de réduction de la variabilité est nécessaire. Cette fonctionnalité n'étant pas prise en compte dans la version actuelle d'AGAP, nous avons développé un nouveau module qui prend en compte cette activité. Ce module est décrit dans la section suivante.

3.2 RCMP : Un prototype de réduction de CMP

Après avoir été une tendance, l'Ingénierie Dirigée par les Modèles (IDM) [Favre *et al.*, 2006] est devenue un paradigme reconnu avec des applications variées. Cette approche met le modèle au centre des préoccupations des analystes /concepteurs en permettant de conceptualiser le développement sous la forme de transformations de modèles. Ces modèles et ces transformations sont décrits conformément à des méta-modèles.

Face à la problématique de réduction de la variabilité dans un CMP, l'IDM apparaît comme une solution idéale puisque l'on peut considérer la réduction comme une suite de transformations de modèles. C'est pour cette raison que nous utilisons les transformations de modèles pour automatiser la réduction d'un CMP dans le but d'assurer la cohérence de la solution réduite.

3.2.1 Transformation de modèles : Langages et outils

L'IDM met à disposition des concepts, des langages et des outils pour créer et transformer des modèles en se basant sur leur méta-modèle. La recherche dans ce sens dégage particulièrement une famille de solutions autour de la plate-forme *Eclipse* [Eclipse, 2009]. Ces solutions font référence de manière récurrente aux *plugins* EMF (*Eclipse Modeling Framework*), GEF (*Graphical Editing Framework*) et GMF (*Graphical Modeling Framework*) ou encore à la plateforme OpenEmbeDD [INRIA, 2009] dérivée d'Eclipse et dédiée aux systèmes embarqués et temps réel.

Plus particulièrement, EMF est un « framework » qui traite des modèles en permettant leur stockage sous forme de fichiers pour en assurer la persistance. Les types de fichiers traités sont multiples et restent conformes à des standards reconnus (*XML, XMI*), mais aussi à des formes spécifiques (code Java). EMF permet de définir un méta-modèle (conforme à Ecore) à partir duquel est généré le code de manipulation et d'édition des modèles conformes à ce méta-modèle.

Pour réaliser les transformations de modèles, différents langages sont utilisés. Nous citons en particulier le standard QVT (*Query View Transformation*) de l'OMG [Jouault *et al.,* 2006], intégré dans l'outil *SmartQVT* [SmartQVT, 2007] et le langage ATL *(ATLAS Transformation Language*) développé par l'équipe ATLAS [ATLAS, 2009]. Ce dernier est inspiré du standard QVT et disponible en tant que plugin ADT *(ATL Development Tools)* d'Eclipse.

3.2.2 RCMP : Architecture fonctionnelle

RCMP (Réduction de Composant Métier Processus) est un prototype développé pour assister l'ingénieur de SI lors de la réduction d'un CMP. La Figure 5-12 illustre les principales fonctionnalités fournies par RCMP. Un ingénieur de SI sélectionne ses variantes en visualisant

l'enchaînement des activités de la vue métier. Une fois les variantes sélectionnées, il déclenche la réduction du CMP et peut par la suite extraire la solution réduite.

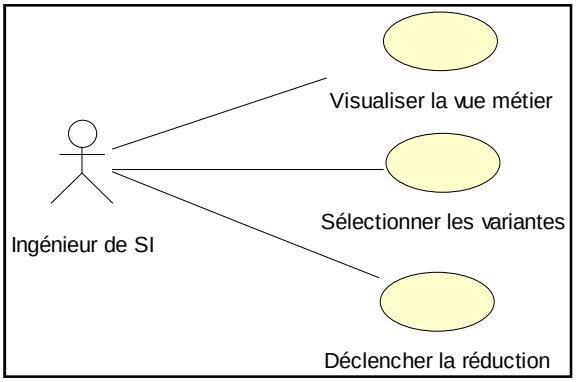


Figure 5-12 : Architecture fonctionnelle de RCMP

Notons que la version actuelle de cet outil se restreint sur la réduction de la vue métier.

3.2.3 RCMP : Architecture technique

a) Choix du langage ATL

Le choix du langage d'implémentation des règles de réduction (données en langage naturel dans les chapitres 3 et 4) s'est porté sur ATL . ATL utilise la syntaxe du langage OCL pour la navigation à travers le modèle d'entrée. Les transformations réalisées sont exogènes puisque les méta-modèles en entrée (MétamodèleSource.ecore) et en sortie (MétamodèleCible.ecore) peuvent être différents. La transformation (Transformation.atl) applique les règles nécessaires sur le modèle source (ModèleSource.xmi) pour aboutir à un modèle Cible (ModèleCible.xmi) (Cf. Figure 5-13).

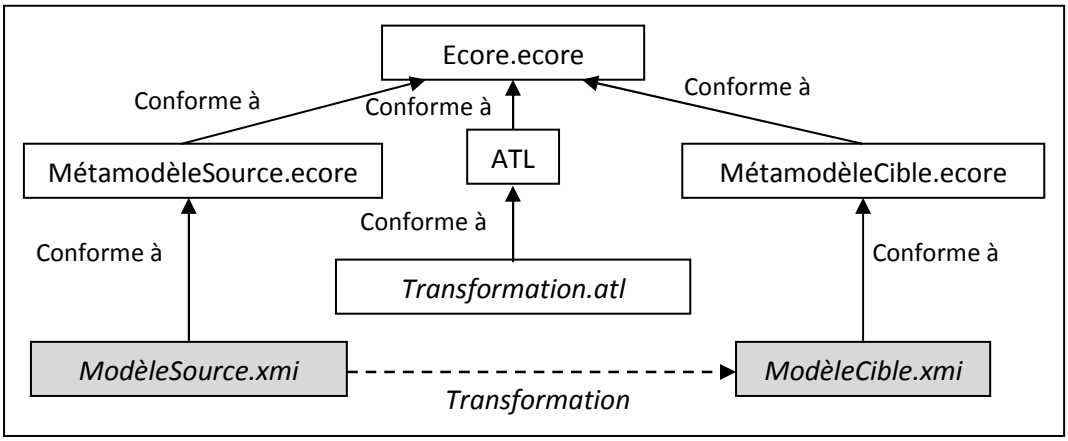


Figure 5-13 : Mécanisme de la transformation de modèles avec ATL

b) Transformation de modèles dans RCMP

La première étape de la réalisation consiste à traduire les méta-modèles mis en jeu tout au long du processus de réduction d'un CMP, i.e. le méta-modèle d'entrée (CMP.ecore) (donné sous forme d'un diagramme de classes dans le chapitre 4 (§ 4.2)), et le méta-modèle de sortie (UML.ecore conforme à UML) de façon à faciliter l'intégration du modèle obtenu dans les environnements de développement des ingénieurs de SI. Pour effectuer la réduction (CMP_Réduction.atl) d'un CMP (ex. AllocRess.xmi) et obtenir un CMP réduit (ex.

AllocRess_Réduit.xmi) il faut que le modèle source soit conforme au méta-modèle CMP.ecore. Le modèle obtenu sera quant à lui conforme à UML.ecore (Cf. Figure 5-14).

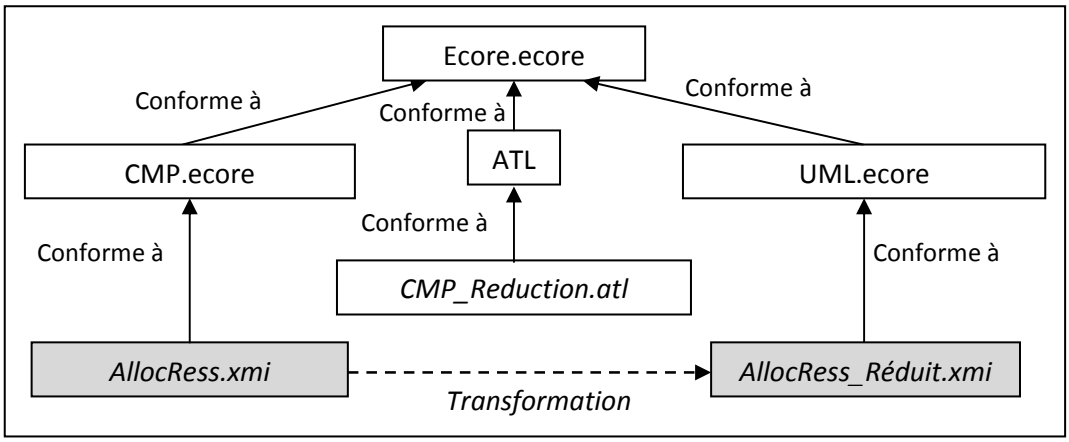


Figure 5-14 : Transformation utilisée dans RCMP

c) Méta-modèle source utilisé

Dans Eclipse, un fichier Ecore (.ecore) peut être représenté sous forme d'un diagramme de classes (.diagecore) grâce au projet Ecore tools. Le diagramme de classes obtenu permet de visualiser graphiquement le méta-modèle source utilisé (CMP.ecore) (Cf. Figure 5-15). Notons que pour répondre à des besoins techniques, ce méta-modèle est légèrement différent de celui présenté dans le chapitre 4.

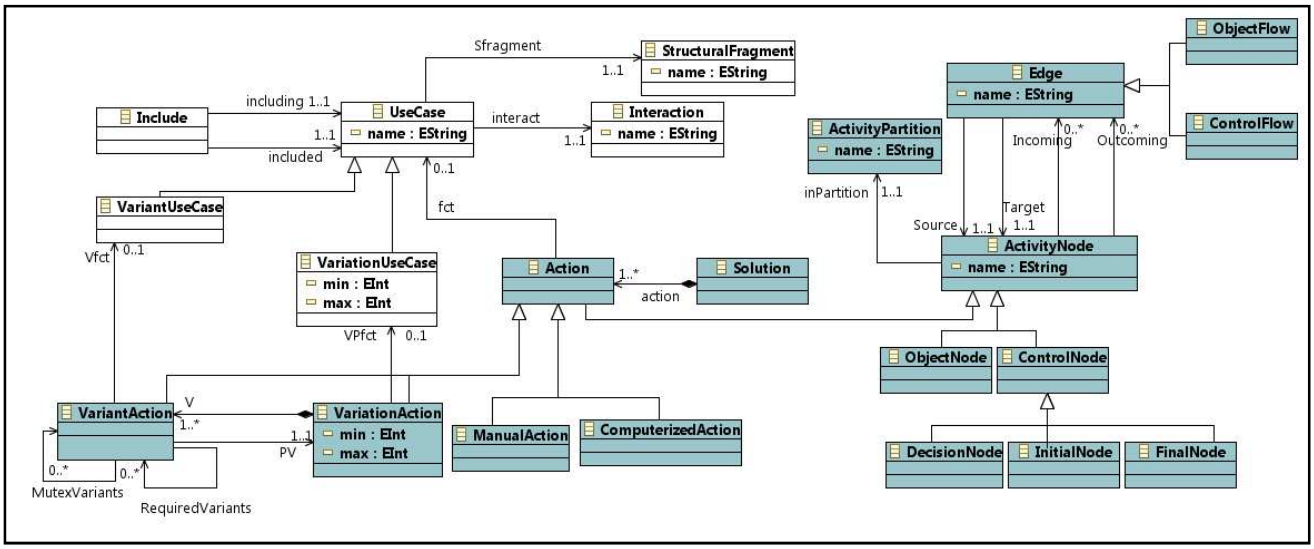


Figure 5-15 : Méta-modèle source utilisé sous Eclipse

d) Règles de transformation

Pour implémenter les règles de réduction d'un CMP, nous utilisons la notion de 'helpers', qui sont l'équivalent de fonctions globales dans les langages de programmation. Nous donnons ci-dessous un extrait des helpers et des règles utilisées dans RCMP. L'ensemble des règles de transformations est donné dans l'annexe B.

Helper de vérification si une variante est sélectionnée ou non.

```
helper context CMPE!ActivityNode
def:isSelected():  Boolean = thisModule.selection.includes (self.name);
```

Helper de vérification si une activité est de type alternative.

```
helper context CMPE!Action def:estVPalt(): Boolean=
        if(not self.oclIsTypeOf (CMPE!VariationAction)) then false
        else
                if(self.min=1 and self.max=1) then true
                else false
                endif
        endif;
```

Helper qui retourne le nombre de variantes sélectionnées d'un point de variation.

```
helper def: NbrVariant(a: CMPE!VariationAction):
Integer = a.V->select (e|e.isSelected())->size();
```

Règle qui copie les variantes sélectionnées d'un point de variation. Cette règle fait appel à deux autres règles « lazy rule » selon le nombre de variantes sélectionnées.

```
rule Copy_PV_SelectedVariant
{
        from u:CMPE!VariationAction
        do {    if (u.singleVariant())
                thisModule. OneSelected(u);
                if (not u.nullVariant() and not u.singleVariant())
                thisModule. UptoOne(u);
           }
}
--Règle de copie si une seule variante est sélectionnée
lazy rule OneSelected
{       from u:CMPE!VariationAction
        to a:CMPS!Action
        ( name<-u.getSelectedfirst(), inPartition<-u.inPartition )
}
--Règle de création d'un nœud de décision dans le cas de sélection de plusieurs variantes
lazy rule UptoOne
{       from u:CMPE!VariationAction
        to uc:CMPS!Action
        (       name<-u.getSelectedfirst(), inPartition<-u.inPartition ),
        c:CMPS!Action ( name<-u.getSelectedlast(), inPartition<-u.inPartition ),
        n:CMPS!DecisionNode ( name<-u.name, inPartition<-u.inPartition )
}
```

3.2.4 Utilisation du prototype

Nous présentons dans cette section comment un ingénieur de SI peut utiliser le prototype RCMP développé pour réduire un CMP sélectionné. Tout d'abord, l'ingénieur de SI doit importer le modèle (.xmi) du CMP sélectionné. Ce modèle constitue le modèle source de la transformation, il est réduit en appliquant les règles de réduction selon les variantes choisies.

a) Modèle source

La Figure 5-16 illustre le modèle source « AllocRess.xmi » utilisé comme entrée de l'activité de réduction. Il est conforme au méta-modèle illustré dans la Figure 5-15. Ce modèle correspond à la vue métier du CMP « Gestion Allocation Ressource ».

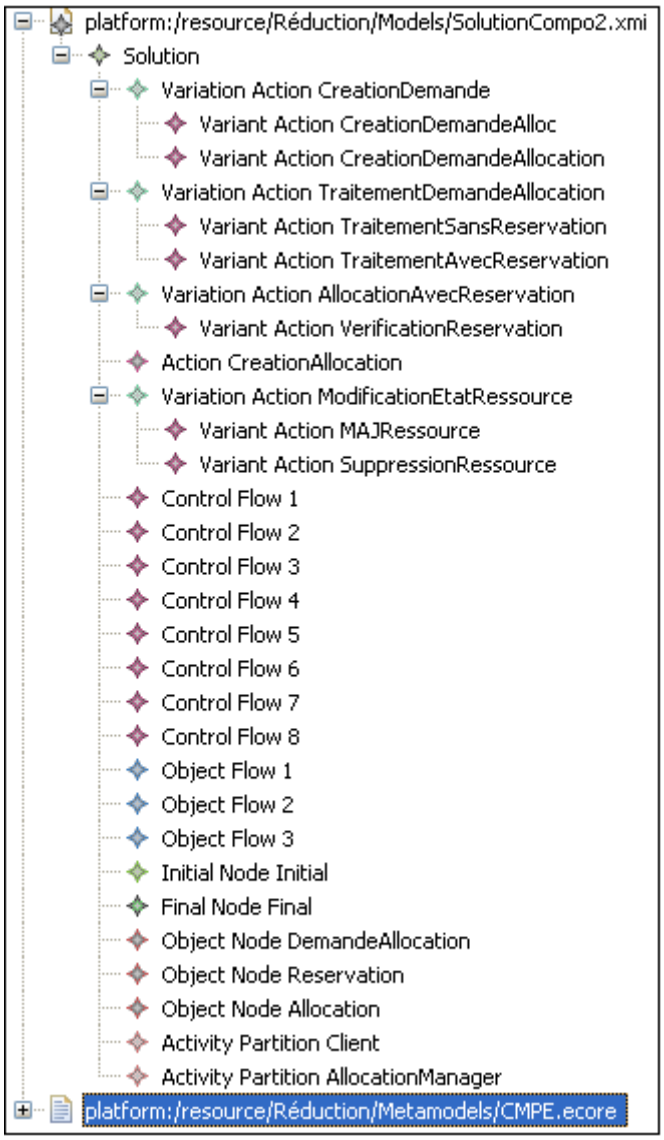


Figure 5-16 : Modèle source « AllocRess.xmi »

b) Modèle réduit

La Figure 5-17 illustre le modèle cible « AllocRess_Réduit » obtenu après exécution des règles de transformation (*Transformation.atl*). La réduction est réalisée, dans ce cas, en sélectionnant les variantes choisies (*CréationDemandeAllocation*, *TraitementSansRéservation* et *MAJRessource)*, et en supprimant les variantes non choisies (*CréationDemandeAlloc, TraitementAvecRéservation, VérificationRéservation et Suppression Ressource)*.

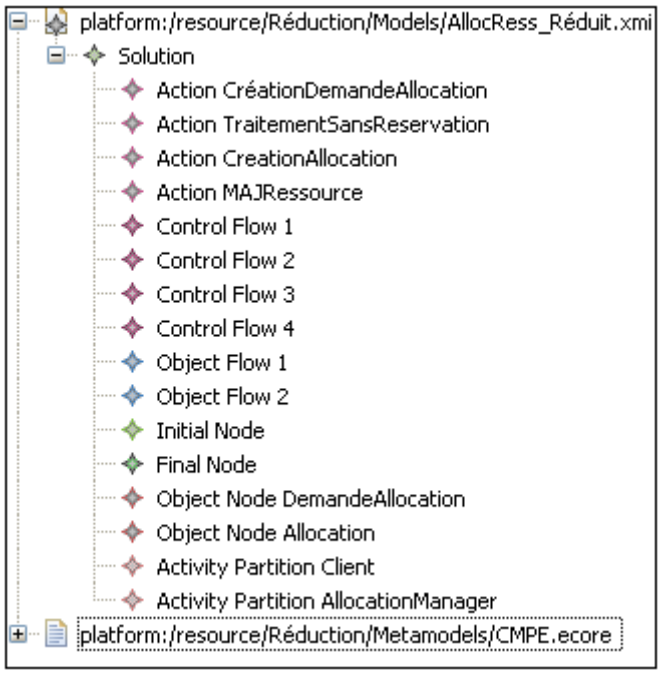


Figure 5-17 : Modèle cible « AllocRess_Réduit.xmi »

4 Expérimentations utilisateurs

4.1 Cadre des expérimentations

Marvelig [Marvelig, 2009], pépinière d'expérimentation scientifique du Laboratoire d'Informatique de Grenoble (LIG) a pour objectif de développer une recherche collaborative et transversale au sein du LIG. Son rôle se décline selon trois axes. Le premier axe concerne la **capitalisation des prototypes produits par les équipes du LIG**, pour les pérenniser en aidant à élaborer des tests expérimentaux pour valider leur usage, des tests pour assurer leur robustesse ainsi que des appuis et des conseils pour les valider. Le deuxième axe s'inscrit dans le cadre de la **mutualisation des potentiels techniques et méthodologiques présents au LIG** en recensant les instruments, leurs fonctionnalités et les personnes ressources et en donnant accès à une base de logiciels. Enfin, le troisième axe concerne la **communication sur les produits de la recherche du LIG** en informant au sein du LIG sur ces outils ainsi qu'à l'extérieur pour les industriels et le grand public.

Dans son axe « **mutualisation des ressources** », Marvelig nous a aidé et appuyé dans la mise en œuvre d'expérimentations orientées utilisateur, dans le but d'expérimenter nos propositions en matière de réutilisation. Ces expérimentations se sont déroulées au sein du LIG en deux séances : le 30 juin et le 03 juillet 2009.

4.2 Objectifs des expérimentations

Les expérimentations que nous avons menées afin d'évaluer, d'un point de vue **qualitatif**, notre modèle de CMP, visaient deux objectifs. Le premier était de connaître les avis des personnes sur le modèle proposé en vérifiant que notre modèle répondait aux objectifs de réutilisation auxquels nous prétendions répondre et de mesurer l'utilisabilité de ce modèle. Le

second était de vérifier que la solution offerte par notre modèle de CMP correspondait bien au critère de complétude que nous visions.

Pour ce faire, nous avons employé des mesures étalées sur les deux principaux axes de cette thèse :

1) Axe processus de spécification des CMP

- Au niveau du processus (usage) : mesurer le guidage du processus, les difficultés rencontrées et le fait que la représentation de la variabilité guide l'abstraction de PM similaires issus de domaines différents.
- Au niveau de la notation de la variabilité (utilité, suffisance, complétude) : mesurer la représentation de plusieurs types de variabilité, la représentation des dépendances entre les variantes et le niveau de clarté de la représentation.
- Au niveau du CMP spécifié (utilisabilité) : mesurer si la spécification obtenue est lisible, suffisamment générique et complète.

2) Axe processus de réutilisation des CMP

- Au niveau du processus : mesurer le guidage du processus, les difficultés rencontrées, et le fait que la vue métier soit suffisante pour pouvoir choisir les variantes.
- Au niveau des règles de réduction : mesurer la pertinence, la suffisance et la complétude.
- Au niveau du PM spécifié (utilisabilité) : mesurer le niveau de satisfaction et la complétude de la spécification réduite obtenue.

4.3 Protocole des expérimentations

4.3.1 Profils des sujets

Les sujets participants à ces expérimentations sont divisés en deux groupes (Cf. Tableau 5-9). Le premier groupe est constitué de personnes qui connaissent la modélisation des processus métier et utilisent régulièrement des spécifications conceptuelles, et jouent le rôle de l'ingénieur de CMP. Le deuxième groupe est constitué de personnes qui ont l'habitude de réutiliser des modèles conceptuels, et jouent le rôle de l'ingénieur de SI.

	Durée	Rôle	Sexe	Moyenne d'âge	Profil
Phase 1	3h	7 concepteurs de processus métier / Ingénieurs de CMP	5h / 2f	30 ans	1 Licence professionnelle 4 Doctorants 2 Enseignants chercheurs
Phase 2	2h30	8 utilisateurs de processus métier / Ingénieurs de SI	5h / 3f	34 ans	1 DUT informatique 3 Master professionnel 3 Doctorants 1 Enseignant chercheur

Tableau 5-9 : Répartition des sujets

4.3.2 Protocole

Dans le but de réaliser nos expérimentations, un protocole a été mis en place. Il est donné de la manière suivante (Cf. Tableau 5-10) :

Étape		Description
Processus de spécification des CMP	*Introduction*	– Situer le contexte général des expérimentations.
	Prise de conscience de la proximité thématique	– Entrée : distribuer trois fiches descriptives de PM assez simples que les utilisateurs doivent généraliser. – Sortie : leur faire élaborer le diagramme d'activités générique.
	Introduction d'éléments variables	– Entrée : leur donner des fiches descriptives de PM similaires mais avec des parties variables. – Sortie : regarder les adaptations (leur faire utiliser deux feuilles).
	Discussion	– Animer une discussion sur leurs résultats. Mesurer ici ce qui peut leur manquer pour généraliser les différents PM. Leur faire imaginer les éléments qui pourraient leur être utiles.
	Introduction du concept de variabilité	– Formation sur la variabilité (notion de points de variation et variantes, types de variabilité, notations proposées).
	Utilisation du formalisme proposé	– Leur faire élaborer le diagramme d'activités avec le formalisme proposé.
	Observation	– Observer les différences entre la première et la seconde version.
	Présentation du résultat final	– Présentation du CMP supportant la variabilité résultant. – Comparer avec le résultat des utilisateurs.
	Évaluation	– Questionnaire d'évaluation (Cf. Annexe C) sur les éléments servant de guides méthodologiques pendant le processus de spécification du CMP, points forts et points faibles.
Processus de réutilisation des CMP	*Introduction*	– Situer le contexte général des expérimentations. – Formation sur la variabilité (notion de points de variation et variantes, types de variabilité, notations, règles de réduction).
	Réduction du CMP	– Entrée : distribuer un CMP, suffisamment complexe pour que la création du modèle réduit ne soit pas tautologique, leur donner deux scénarios d'utilisation. – Sortie : leur faire effectuer la réduction avec les règles de réduction proposées (les scénarios proposés devraient couvrir l'ensemble des règles proposées).

	Discussion	– Animer une discussion sur leurs résultats. – Faire émerger les opinions sur les règles de réduction ? les manques ? les redondances ? les superflues ? Comment peut-on améliorer ces règles de réduction ? Quel est le degré de complétude de la spécification obtenue ?
	Présentation du résultat final	– Présentation des scénarios avec les réductions réalisées. – Comparer avec le résultat des utilisateurs.
	Évaluation	– Questionnaire d'évaluation (Cf. Annexe C) sur les points forts et points faibles de ces règles de réduction, ainsi que sur le niveau de satisfaction des utilisateurs de la spécification obtenue.

Tableau 5-10 : Protocole des expérimentations

4.3.3 Données produites

Les données produites à l'issue des expérimentations réalisées concernent :

- Des enregistrements audio des focus ;
- Des questionnaires d'évaluation ;
- Des représentations papier des exercices.

4.4 Analyse des résultats

L'analyse des résultats des expérimentations repose essentiellement sur l'analyse des discussions, menées tout au long des séances, ainsi que sur des questionnaires d'évaluation remplis par les sujets. Dans chaque séance, deux questionnaires sont fournis :

- Le premier est distribué avant de commencer les expériences. Son objectif est de connaître les habitudes des sujets en matière de spécification et de réutilisation de composants métier.
- Le second est distribué à la fin de la séance. Son objectif est d'évaluer et de mesurer les opinions des sujets quant à l'usage et l'utilisabilité de nos propositions.

4.4.1 Axe processus de spécification des CMP

a) Points à tester et hypothèses

Le Tableau 5-11 résume les différents points à tester et les hypothèses sous-jacentes de la première expérience.

Rubrique	Points à tester	Hypothèses
Processus de spécification	Guidage du processus	Le processus est un bon guide pour l'ingénieur de CMP lors de la spécification d'un composant
Notation de la variabilité	Utilité	La notation proposée est utile
	Suffisance	La notation proposée est complète
	Clarté	La notation proposée est claire et facile à utiliser
CMP spécifié	Lisibilité	La spécification obtenue est lisible hormis l'introduction

		de la variabilité
	Généricité	La spécification obtenue est suffisamment générique
	Complétude	La spécification obtenue est complète

Tableau 5-11 : Points à tester et hypothèses

b) Analyse des résultats

- ✓ Analyse des pratiques

L'analyse des pratiques des sujets en matière de réutilisation, a révélé l'utilité de l'utilisation de processus métier pour décrire certaines fonctionnalités dans un domaine particulier. Trois sujets parmi sept ont l'habitude de réutiliser des processus métier. Ces trois sujets réutilisent les CM dans le cadre de l'adaptation de modèles déjà conçus généralement en UML, pour illustrer des cours ou faire des examens.

- ✓ Analyse des données recueillies

L'analyse des données issues de l'expérimentation a donné lieu à des résultats illustrant la répartition des opinions selon les différentes rubriques à tester. Nous synthétisons les opinions selon les différentes hypothèses énoncées ci-dessous, et citons, entre « », certaines phrases données par les utilisateurs.

1) Guidage du processus de spécification

L'opinion dominante (6/7 sujets) sur le guidage du processus de spécification concerne l'utilité du processus, car il permet de vérifier si tous les besoins sont réalisés « *utile pour vérifier si toutes les spécifications ont été réalisées* ». Les sujets ont tous déclaré que le processus est bien guidé en ayant des étapes précises et primordiales. En revanche, le reproche adressé (1/7 sujets), concerne le fait que le processus est parfois trop détaillé « *on trouve beaucoup d'information dans le processus* », ce qui exige une phase d'apprentissage avant son utilisation. Dans ce sens, les fiches descriptives d'un processus métier ont été jugées utiles en présentant un intérêt particulier pour le modèle organisationnel.

2) Notation de la variabilité

Concernant l'indicateur notation de la variabilité, les sujets ont déclaré que la notation proposée permet de représenter l'ensemble des types « *représentation de tous les types d'une manière simple et claire* ». Les sujets ont également mentionné que l'introduction de la variabilité permet de mieux cerner les propriétés réutilisables ce qui favorise la réutilisation.

3) CMP spécifié

La lisibilité de la spécification obtenue a fait émerger un débat entre les sujets. La plupart d'entre eux (5/7) a trouvé la spécification obtenue plutôt lisible hormis l'introduction de la variabilité. Or, certains parmi eux (2/7) ont exprimé le contraire. Ces derniers ont initialement proposé de représenter la variabilité séparément, mais après avoir vu le résultat obtenu en appliquant correctement le processus de spécification, ils ont modifié leur point de vue en signalant que si la granularité du modèle est importante, la spécification pourrait dans ce cas être difficilement lisible.

D'un autre côté, la plupart des sujets (6/7) a déclaré la généricité et la complétude de la spécification obtenue. Cependant, un sujet sur les sept participants, a indiqué qu'il lui

manquait une phase de réutilisation du composant pour pourvoir bien juger la généricité et la complétude du modèle.

c) Synthèse

D'une manière générale, les sujets pensent que le processus proposé permet de mieux guider la représentation de la généricité et de la variabilité, identifiées au niveau de l'ensemble des PM, d'une manière complète « *elle permet de représenter facilement et de manière complète la généricité et la variabilité dans des diagrammes d'activités, ce qui permet de mieux cerner les propriétés réutilisables* ». En revanche, des exemples illustrant nos propositions étaient recommandés par les sujets à chaque utilisation d'un nouveau concept « *lors de l'utilisation d'un nouveau concept, il est préférable d'utiliser des exemples pour mieux appréhender la méthode*». Résultat qui se retrouve dans le besoin d'une phase d'apprentissage à la méthode perçue par l'ensemble des sujets. La phase d'apprentissage déclarée est de quelques heures, afin de comprendre les étapes du processus et la notation de la variabilité séparément de leur usage. Le Tableau 5-12 résume les résultats de la première expérimentation.

Rubrique	Points à tester	Points forts	Points faibles
Processus	Guidage du processus	- Utile / guidé / ayant des étapes précises et primordiales - Fiche descriptive utile (7/7)	Parfois trop détaillé : phase d'apprentissage exigée avant toute utilisation (7/7)
Notation de la variabilité	Utilité	Favorise la réutilisation (3/7)	----
	Suffisance	Représentation de tous les types de variabilité (7/7)	---
	Clarté	Représentation d'une manière simple et claire (5/7)	---
CMP spécifié	Lisibilité	Reste lisible si la granularité du composant n'est pas importante (6/7)	Peut être difficilement lisible si le composant est complexe (1/7)
	Généricité	Spécification générique (7/7)	---
	Complétude	Spécification complète (7/7)	---

Tableau 5-12 : Grille de synthèse de la première expérimentation

4.4.2 Axe processus de réutilisation des CMP

a) Points à tester et hypothèses

Le Tableau 5-13 récapitule les différents points à tester et les hypothèses sous-jacentes de la deuxième expérience.

Rubrique	Points à tester	Hypothèses
Processus de réutilisation	Guidage de la vue métier	La vue métier est le point d'entrée pour la réutilisation
Règles de réduction	Suffisance	Les règles proposées sont complètes
	Clarté	Les règles proposées sont claires

PM spécifié	Satisfaction	La spécification obtenue est satisfaisante
	Complétude	La spécification obtenue est complète

Tableau 5-13 : Points à tester et hypothèses

b) Analyse des résultats

- ✓ Analyse des pratiques

Les sujets participants à la deuxième expérimentation n'ont pas tous l'habitude de réutiliser des processus métier. 6 sujets seulement parmi 8 avaient parfois utilisé des processus métier dans le cadre de développement d'un SI. La réutilisation, telle qu'elle a été décrite par les sujets, concerne surtout « *l'analyse des activités constituant un PM réutilisable et la décision entre laquelle on doit réutiliser* ».

Cela, selon eux, dépend de plusieurs contraintes, comme par exemple, le contexte de réutilisation et le profil de l'utilisateur « *la réutilisation est liée au contexte qui peut dépendre du profil de l'utilisateur* ». En outre, la réutilisation d'un composant est liée à sa généricité et au nombre d'activités qui y sont incluses. Au niveau de sa description, « *un composant doit avoir un nom, une description de son objectif et des différentes situations où il a été réutilisé…* ». Dans ce sens, « *il faut un exemple d'application de ce composant et des synonymes des termes utilisés afin d'éviter l'incompréhension* ». Enfin, un sujet a mentionné le fait qu'il faut penser à « *plutôt s'inspirer que de réutiliser* ».

- ✓ Analyse des données recueillies

L'analyse des données issues de la deuxième expérimentation a donné lieu à des résultats illustrant la répartition des opinions selon les différentes rubriques à tester. Nous synthétisons ci-dessous ces opinions.

1) Guidage de la vue métier

En ce qui concerne la vue métier, 5 sujets sur 8 l'ont jugée satisfaisante pour le choix des variantes ainsi que pour la réutilisation du composant. La mise en évidence de la variabilité augmente également sa réutilisabilité (7/8). D'un autre côté, un CM de nature processus est flexible (6/8) et aide à mieux décrire le PM en cours de développement.

2) Règle de réduction

Concernant les règles de réduction de la variabilité, elles ont été jugées claires et complètes. Cependant, un sujet a suggéré de rajouter « *des conditions structurelles, c'est-à-dire utiliser des critères de réutilisation au lieu d'utiliser seulement des cardinalités ».* Un autre sujet a préféré *« ajouter des exemples et commentaires sur chaque variante pour éviter les ambigüités terminologiques lors des choix des variantes ».*

3) PM spécifié

En ce qui concerne la spécification réduite, 7/8 l'ont trouvée satisfaisante, et 6/8 l'ont jugée complète. L'insatisfaction de certains sujets est due à leur « *besoin de plus de documentation du composant utilisé* ».

c) Synthèse

Selon les sujets, un CM est un bon support de réutilisation s'il est bien documenté (1/8) en fournissant un exemple de réutilisation de ce CM. Cela peut aider à réduire l'incompréhension du modèle abstrait « *problème de terminologie, diagramme trop abstrait pour pouvoir l'utiliser* ». Le CM supportant la variabilité « *favorise la communication (référence pour la modélisation)* », ce qui favorise la réutilisation et l'adaptation (1/8).

D'un autre côté, la méthode nécessite une phase d'apprentissage de quelques heures dans le but de comprendre les étapes du processus et la notation de la variabilité séparément de leur usage. Le Tableau 5-14 résume les résultats de la deuxième expérimentation.

Rubrique	Points à tester	Points forts	Points faibles
Processus de réutilisation	Guidage de la vue métier	Réduit la complexité lors de la réutilisation des PM (1/8). Donne une idée sur la solution (1/8).	Difficulté d'adapter un CM générique (1/8). Besoin de documentation.
Règles de réduction	Suffisance	Suffisantes (8/8)	---
	Clarté	Claires (8/8)	---
PM spécifié	Satisfaction	Gain de temps (1/8)	Si le cahier des charges est complexe, il est difficile de réutiliser un CMP (2/8).
	Complétude	Solution complète (6/8)	---

Tableau 5-14 : Grille de synthèse de la deuxième expérimentation

4.5 Synthèse

À l'issue des deux expérimentations, nous avons pu valider, d'un point de vue **qualitatif,** certaines hypothèses que nous prétendions vérifier. Concernant le processus de spécification de CMP, nos propositions ont été validées au niveau du guidage du processus ainsi que de la notation proposée : « *c'est un concept très intéressant du point de vue coût d'implémentation* », « *c'est un sujet important pour gagner du temps, l'argent, surtout si les composants de base sont prétestés* ». Quant au processus de réutilisation du CMP, les expérimentations ont montré une tendance vers la réutilisation de CMP assez génériques et supportant la variabilité, en exigeant une bonne documentation de ceux-ci, ce qui est déjà vérifié dans nos propositions. Toutefois, nous avons pu aussi constater certaines limites de la réutilisation, un exemple de ces limites concerne la complexité des besoins et du cahier des charges d'un SI : « s*i le cahier des charges est compliqué et précis, il faut s'inspirer plutôt que de réutiliser* ».

5 Conclusion

L'objectif de ce chapitre était de mettre en œuvre et de valider l'ensemble des propositions présentées dans cette thèse. Pour ce faire, nous avons présenté des mécanismes d'organisation des CMP dans un environnement en proposant un formalisme de documentation de CMP. Nous avons également présenté un processus d'imitation qui assiste

l'ingénieur des SI pour imiter un CMP. Ce processus a été par la suite intégré dans le cycle de vie d'une méthode de développement basée composants, en l'occurrence la méthode Symphony. L'ensemble de ces mécanismes est mis en œuvre par l'utilisation d'outils supports. Enfin, nous avons présenté les résultats d'expérimentations utilisateurs réalisées dans le but d'évaluer nos propositions.

Chapitre 6 : Conclusion et Perspectives

L'ingénierie de systèmes d'information par réutilisation est devenue largement adoptée et utilisée, et a fait émerger des méthodes et outils innovants. C'est dans cette perspective que l'approche *Composant Métier* a été proposée.

La mise en œuvre de cette approche pose plusieurs problématiques de recherche, en particulier l'identification et la modélisation des propriétés candidates à la réutilisation.

Ce chapitre clôture ce mémoire en rappelant les contributions réalisées qui répondent aux problématiques exposées (§1) et en explicitant des perspectives possibles des travaux présentés (§2).

1 Bilan des contributions

Les propositions présentées dans le cadre de cette thèse apportent des réponses à plusieurs problématiques de recherche dans le domaine de la réutilisation de composants métier. Ces résultats sont les suivants.

1. *Une réutilisation de CM de type processus*

Dans les différentes approches antérieures, l'information réutilisable encapsulée dans un CM était souvent limitée à une information de type entité. Nous avons montré que la réutilisation de CM de type processus s'avère plus pertinente qu'une simple réutilisation de CM entité.

2. *Un modèle de CM de portée conceptuelle, encapsulant des besoins fonctionnels*

Des CMP décrits selon ce modèle, constituent de véritables productions capitalisables et réutilisables dans le cycle de conception d'un SI à base de composants ; ils sont technologiquement neutres et n'évoluent qu'en fonction de l'évolution des besoins auxquels ils répondent.

3. *Une spécification complète et variable d'un CMP*

D'une part, la modélisation multi-vues de la solution d'un CMP apporte une bonne qualité de conception, par une spécification complète matérialisée par quatre vues de développement : vue métier, vue fonctionnelle, vue dynamique et vue structurelle. D'autre part, la variabilité introduite permet aux CMP d'avoir la capacité de répondre aux besoins variables des SI.

4. *Un processus de développement de CMP, dédié aux ingénieurs de composants*

Nos propositions procurent aux concepteurs de CMP un processus dédié à l'ingénierie des CMP multi-vues supportant la variabilité. Ce processus fournit un guide méthodologique pour la spécification de CMP supportant la variabilité, tout en permettant de maintenir la traçabilité de la variabilité dans les quatre vues du développement du CMP. De telles spécifications ne sont utiles que si elles sont organisées dans un environnement dédié à la réutilisation. C'est dans ce but que nous avons proposé des mécanismes d'organisation des CMP en proposant un formalisme de documentation de ceux-ci.

5. *Un processus de réutilisation d'un CMP, dédié aux ingénieurs de SI*

Ce processus assiste l'ingénieur des SI lors de l'imitation d'un CMP. Pour illustrer la pertinence de ce processus, nous l'avons intégré dans le cycle de vie d'une méthode de développement en utilisant la méthode Symphony comme référence.

Pour évaluer nos propositions, nous avons développé un environnement support à nos propositions. En outre, nous avons effectué des expérimentations utilisateurs qui nous ont aidés à mesurer l'usage du modèle de CMP ainsi que le guidage des processus proposés.

2 Perspectives

Les propositions présentées dans cette thèse peuvent être améliorées à plusieurs niveaux, tant sur l'axe de l'approfondissement des travaux réalisés que sur celui de l'élargissement du domaine de la recherche.

- ✓ Approfondissement des propositions

1. *Identification de la variabilité*

L'identification de la variabilité est liée à la définition des caractéristiques qui différencient des processus métier de même nature. La mise en œuvre de cette activité reste complexe du moment où l'étude de similarité et d'analogie entre les informations, utilisées dans les PM étudiés, est freinée par la divergence de ces informations. L'introduction d'une ontologie de domaine [Hernandez, 2005] à ce niveau pourrait réduire, voire éliminer, la confusion conceptuelle et terminologique et établir une compréhension partagée. Une telle approche doit fournir les concepts-clés ainsi que les attributs relatifs aux domaines métier étudiés.

2. *Composition des solutions*

La spécification de CMP sur plusieurs vues de développement est insuffisante et nécessite des mécanismes de composition. Les CMP que nous proposons sont d'une granularité importante puisque ils sont de nature processus. Cependant, l'imitation de CMP pour spécifier des SI concernant, par exemple, plusieurs domaines fonctionnels, est nécessaire et devra faire l'objet d'une étude approfondie.

- ✓ Élargissement du domaine de la recherche

1. *Variabilité pour la personnalisation*

Si les travaux réalisés dans cette thèse concernent particulièrement la variabilité pour la réutilisation, il est nécessaire de penser également à appliquer les concepts de variabilité que nous avons dégagés et qui restent des concepts transversaux à plusieurs champs de recherche, pour d'autres finalités.

Une première proposition concerne l'étude de la variabilité pour la personnalisation. Les systèmes d'information ont évolué en terme d'architecture (ils sont hétérogènes et distribués) et d'usage (ils sont ouverts et donc accessibles à une large variété d'usagers). Pour toutes ces raisons, l'expression de la variabilité pour la personnalisation s'avère utile. Un premier travail [Hachani, 2009] réalisé au sein de l'équipe SIGMA a d'ailleurs montré des besoins forts en termes de prise en compte de la variabilité pour la personnalisation des SI.

2. *Intégration de la variabilité dans d'autres langages de modélisation de processus métier*

Nos travaux étendent le méta-modèle d'UML par les concepts de variabilité. Cependant, l'introduction de la variabilité dans un méta-modèle générique, qui peut être adapté dans le contexte de plusieurs langages de modélisation des processus métier, mériterait d'être approfondie. Nous avons d'ailleurs démarré cette étude en collaboration avec d'autres doctorants de l'équipe SIGMA, cette étude vise l'extension, par les concepts de variabilité, du méta-modèle de définition des processus métier : Business Process Definition Metamodel (BPDM).

3. *Évolution du CMP*

La variabilité dans le temps concerne le fait qu'il y ait différentes versions d'un CMP à différents moments. Il s'agit donc de l'*évolution* du CMP en fonction de l'évolution de son domaine métier.

Pour définir un domaine métier, il est nécessaire de définir sa frontière. Or, cette frontière n'est pas souvent facile à délimiter. Ceci peut être dû à l'inexpérience de l'analyste du domaine, au manque de la maturité du domaine ou encore à la nature de certains domaines qui utilisent ou fournissent des services à d'autres domaines.

La gestion de l'évolution du CMP en assurant l'intégrité de sa solution reste un challenge et un sujet de recherche ouvert.

Annexes

Annexe A. Étude de cas « Gestion Allocation de Ressource »

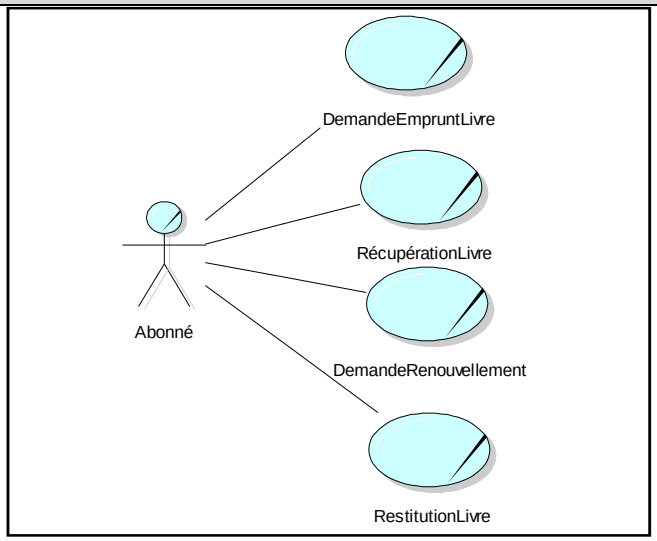
Cette annexe présente l'ensemble de l'étude de cas utilisée dans le chapitre 4. Il concerne les quatre vues de développement du CMP « Gestion Allocation Ressource ».

1. Étude préalable

✓ **Liste des PM utilisés**

Ressource	PM	SI
Livre	PM_1 : Gestion Emprunt Livre	Bibliothèque
Voiture	PM_2 : Gestion Allocation Voiture	Agence de voiture
Chambre	PM_3 : Gestion Allocation Chambre d'Hôtel	Hôtel
Billet d'avion	PM_4 : Gestion Allocation Billet d'Avion	Compagnie aérienne
Billet de spectacle	PM_5 : Gestion Allocation Billet Spectacle	Agence artistique

a) Identification du PM réutilisable

<table>
<tr><th colspan="2">PM $_1$: Gestion Emprunt Livre</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un abonné (acteur principal) émet une demande d'emprunt d'un livre au bibliothécaire en remplissant un formulaire électronique.
Le bibliothécaire (acteur secondaire) traite la demande :
1. Identifie l'abonné par le biais du code abonné.
2. Vérifie la disponibilité du livre.
3. Informe l'abonné de la disponibilité du livre et enregistre la demande.
- L'abonné se présente à la bibliothèque pour récupérer le livre.
Le bibliothécaire :
1. Identifie l'abonné par le biais du code demande.
2. Crée l'emprunt.
3. Remet le livre à l'abonné.
- L'abonné peut émettre une demande de renouvellement de l'emprunt.
Le bibliothécaire :
1. Identifie l'abonné par le biais du code d'emprunt.
2. Vérifie l'autorisation de l'abonné à renouveler l'emprunt.
3. Met à jour l'emprunt.
- L'abonné se présente à la bibliothèque pour rendre le livre.
Le bibliothécaire enregistre le retour du livre en mettant à jour l'état du livre.</td></tr>
</table>

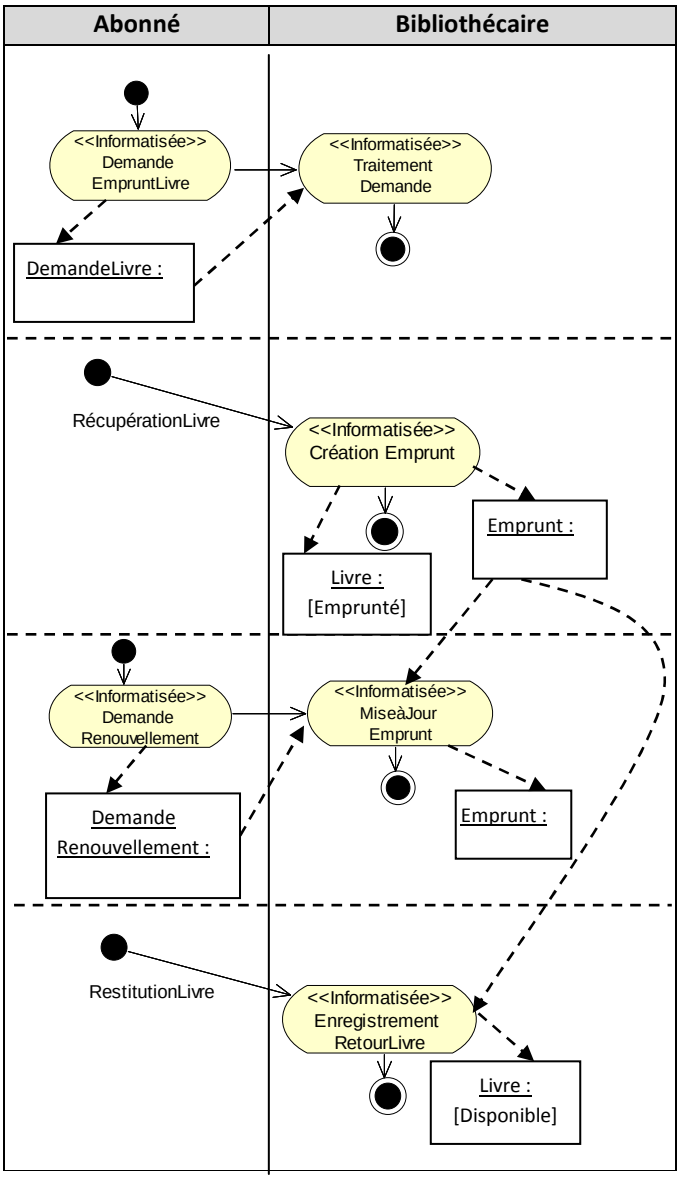
Modèle organisationnel

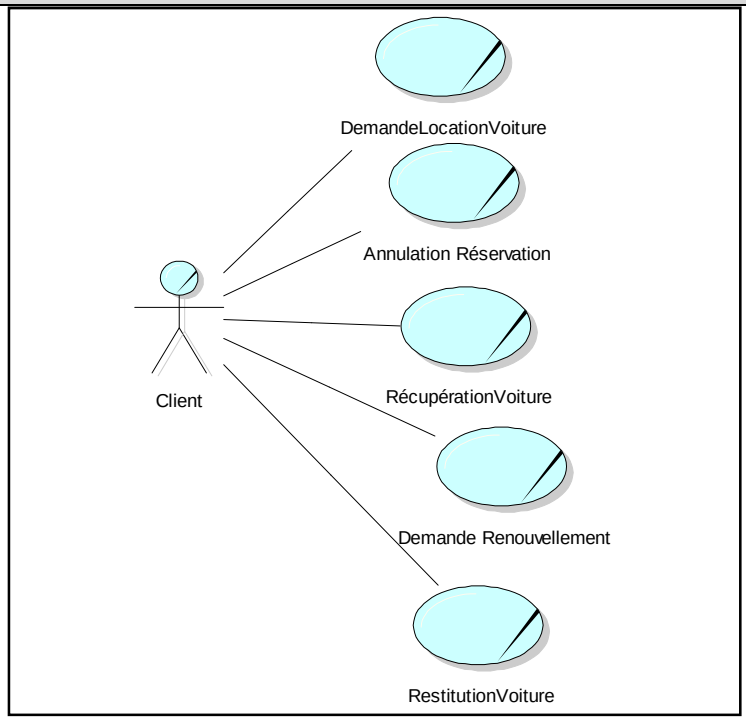
« Demande Emprunt Livre »

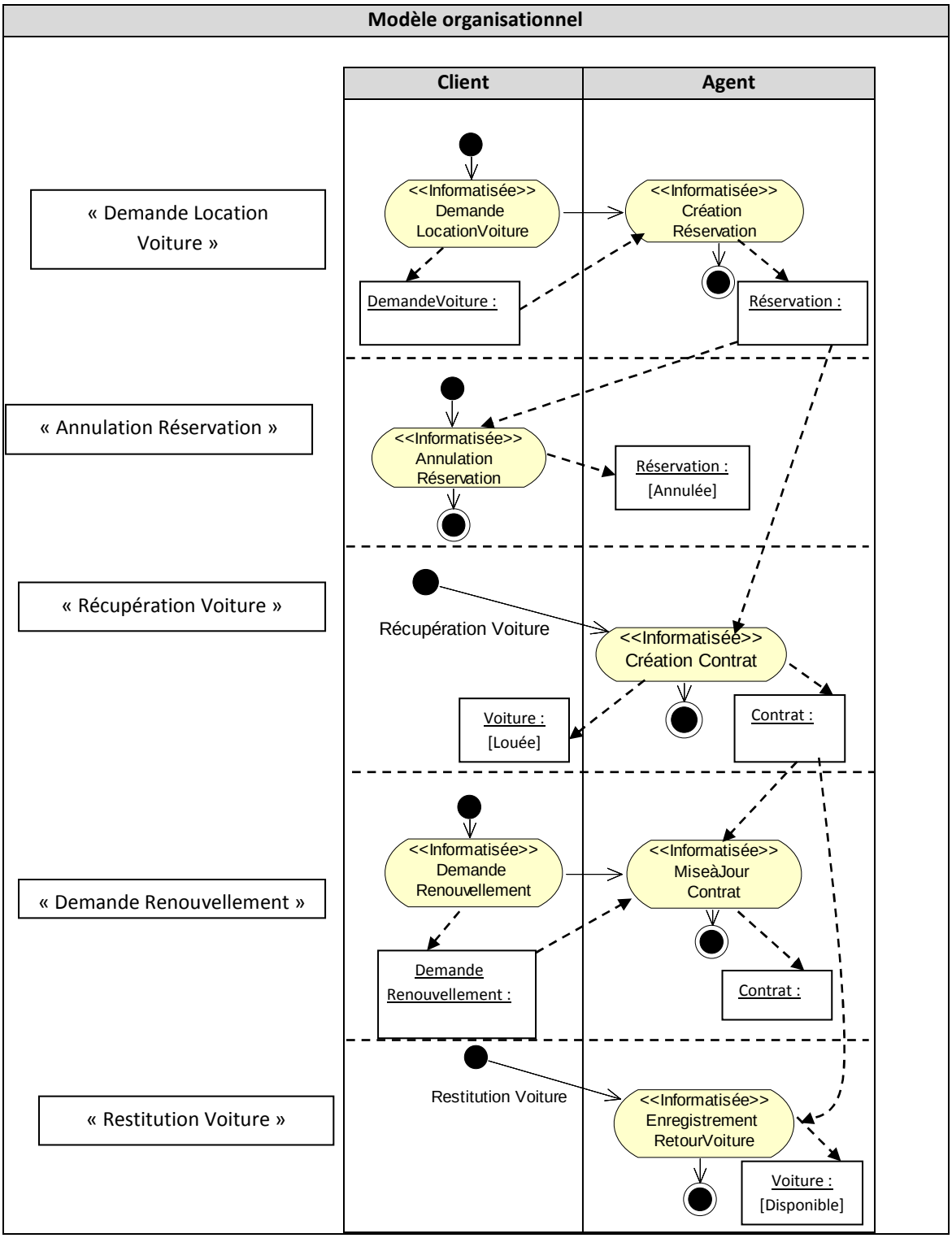
« Récupération Livre »

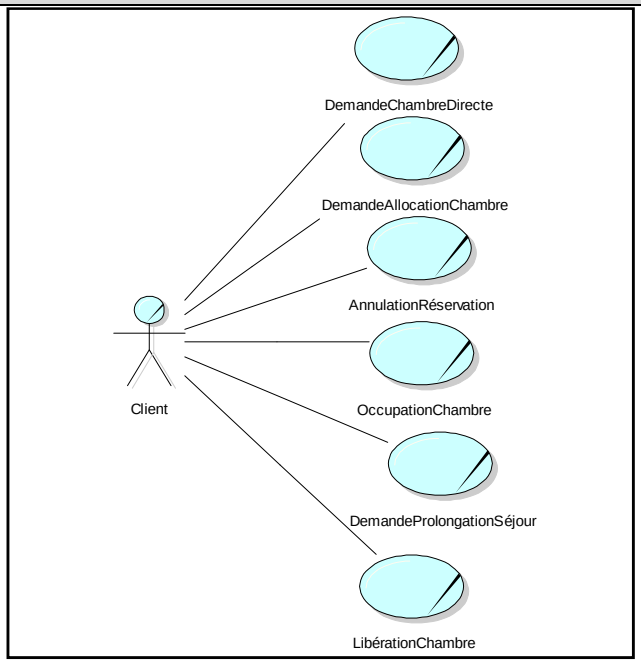
« Demande Renouvellement »

« Restitution Livre »



PM $_2$: Gestion Allocation Voiture	
Modèle métier	
Scénario	- Un *client (acteur principal)* émet une demande de location d'une voiture en remplissant un formulaire électronique. L'*agent (acteur secondaire)* : 1. Vérifie la disponibilité d'une voiture. 2. Crée une réservation. 3. Communique au client le code de réservation. - Le *client* peut annuler sa réservation via un formulaire électronique. - Le *client* se présente à l'agence de voiture pour récupérer la voiture. L'*agent* : 1. Identifie le client par le biais du code réservation. 2. Crée le contrat de location. 3. Remet au client la voiture. - Le *client* peut demander le renouvellement du contrat. L'*agent* : 1. Identifie le client par le biais du code contrat. 2. Vérifie l'autorisation du client à renouveler le contrat. 3. Met à jour le contrat. - Le *client* se présente à l'agence de voiture pour rendre la voiture. L'*agent* enregistre le retour de la voiture en mettant à jour l'état de la voiture.


Modèle organisationnel
Client
Agent
« Demande Location Voiture »
<<Informatisée>> Demande LocationVoiture
<<Informatisée>> Création Réservation
DemandeVoiture :
Réservation :
« Annulation Réservation »
<<Informatisée>> Annulation Réservation
Réservation : [Annulée]
« Récupération Voiture »
Récupération Voiture
<<Informatisée>> Création Contrat
Voiture : [Louée]
Contrat :
« Demande Renouvellement »
<<Informatisée>> Demande Renouvellement
<<Informatisée>> MiseàJour Contrat
Demande Renouvellement :
Contrat :
« Restitution Voiture »
Restitution Voiture
<<Informatisée>> Enregistrement RetourVoiture
Voiture : [Disponible]

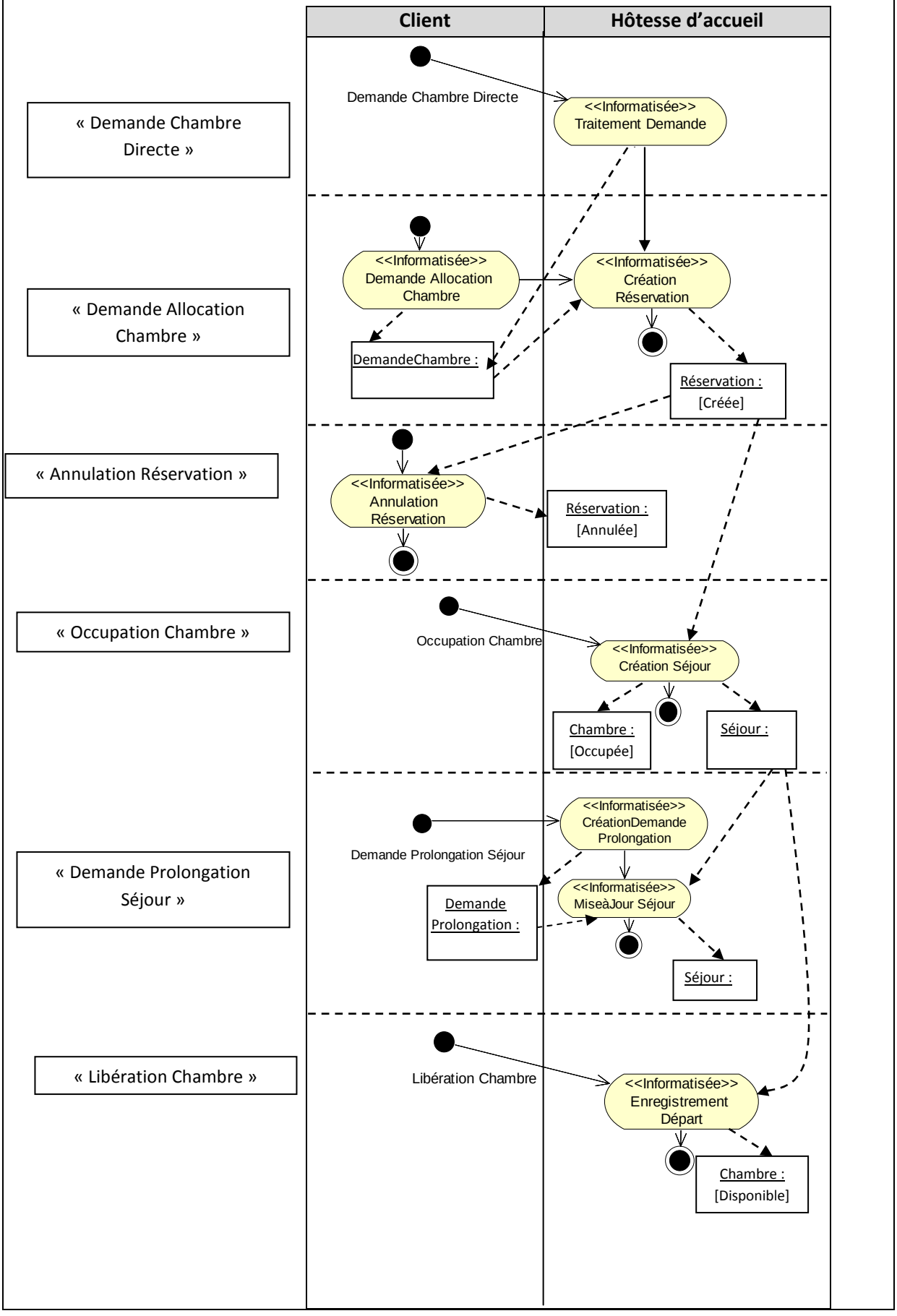
<table>
<tr><th colspan="2">PM ₃ : Gestion Allocation Chambre d'hôtel</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) se présente à un hôtel et demande une chambre.
L'hôtesse d'accueil (acteur secondaire) :
1. Crée la demande.
2. Vérifie la disponibilité d'une chambre et enregistre la réservation.
3. Crée le séjour et remet la clé de la chambre au client.
- Un client émet une demande de réservation de chambre d'hôtel en remplissant un formulaire électronique.
L'hôtesse d'accueil :
1. Vérifie la disponibilité d'une chambre.
2. Crée une réservation.
3. Communique au client le code réservation.

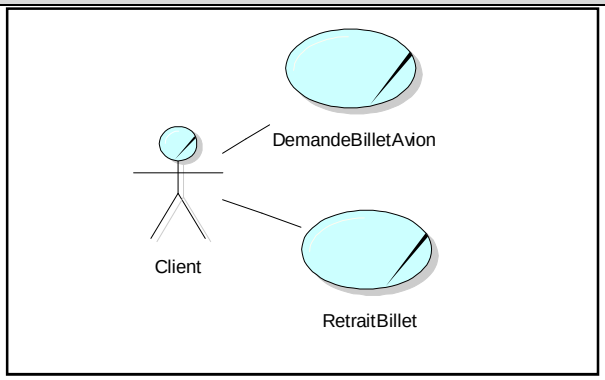
- Le client peut annuler sa réservation via un formulaire électronique.
- Le client se présente à l'hôtel pour récupérer la clé de sa chambre.
L'hôtesse d'accueil :
1. Identifie le client par le biais du code de la réservation.
2. Crée le séjour.
3. Remet la clé de la chambre au client.

- Le client peut demander une prolongation du séjour.
L'hôtesse d'accueil :
1. Identifie le client par le biais du code séjour.
2. Crée la demande de prolongation du séjour.
3. Vérifie la disponibilité de la chambre ou d'une autre à la date demandée.
4. Met à jour l'ancien séjour.

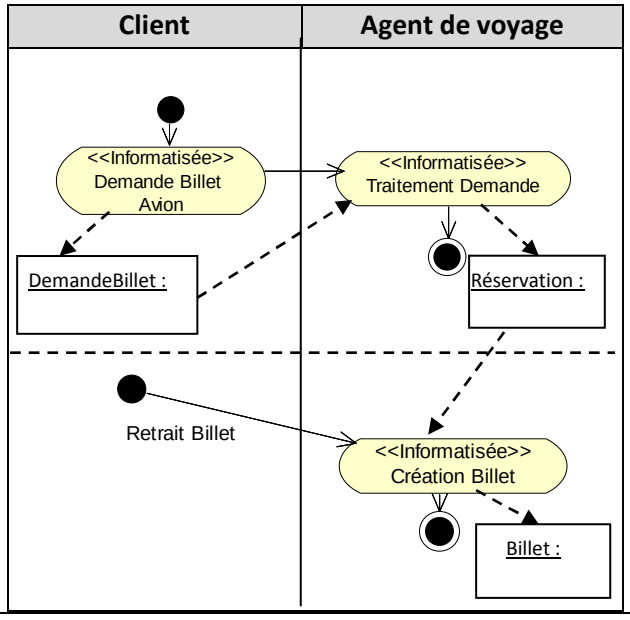
- Le client libère la chambre.
L'hôtesse d'accueil met à jour l'état de la chambre.</td></tr>
</table>

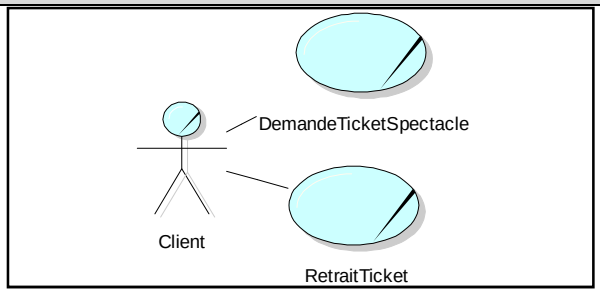
Modèle organisationnel



<table>
<tr><th colspan="2">PM $_4$: Gestion Allocation Billet Avion</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) émet une demande d’achat d’un billet d’avion à une agence de voyage Lowcost en remplissant un formulaire électronique.

L’agent de voyage (acteur secondaire) :
1. Vérifie la disponibilité d’un billet.
2. Crée une réservation.
3. Communique au client le code de sa réservation en mentionnant l’impossibilité d’annulation.

- Le client se présente à l’agence de voyage pour payer et retirer son billet.
L’agent de voyage :
1. Identifie le client par le biais du code réservation.
2. Crée le billet d’avion.
3. Imprime le billet et le remet au client.</td></tr>
<tr><th colspan="2">Modèle organisationnel</th></tr>
<tr><td colspan="2">« Demande Billet Avion »
« Retrait Billet »
 
</td></tr>
</table>

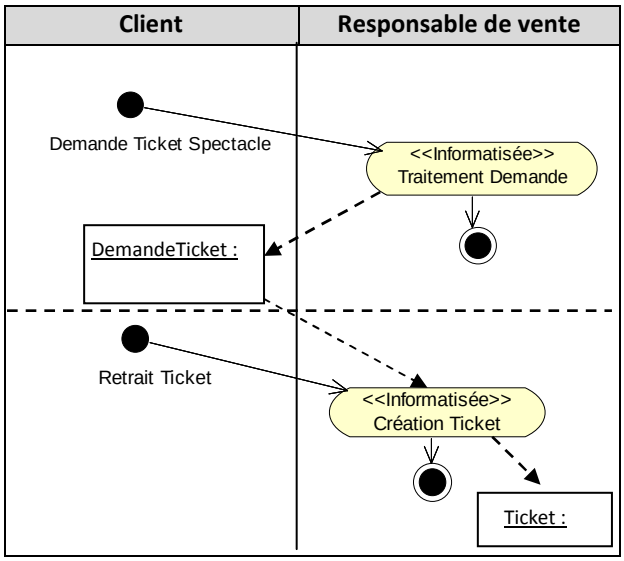
<table>
<tr><th colspan="2">PM $_5$: Gestion Allocation Billet Spectacle</th></tr>
<tr><td>Modèle métier</td><td> 
</td></tr>
<tr><td>Scénario</td><td>- Un client (acteur principal) se présente à une agence artistique et demande l’achat d’un billet de spectacle au responsable de vente.

Le responsable de vente (acteur secondaire) :

1. Vérifie la disponibilité d’un ticket.
2. Informe le client de la disponibilité du ticket.
- Le client confirme sa demande en demandant le retrait du ticket.
Le responsable de vente :

1. Crée le ticket du spectacle.
2. Imprime le ticket et le remet au client.</td></tr>
<tr><th colspan="2">Modèle organisationnel</th></tr>
<tr><td colspan="2">« Demande Ticket Spectacle»

« Retrait Ticket »

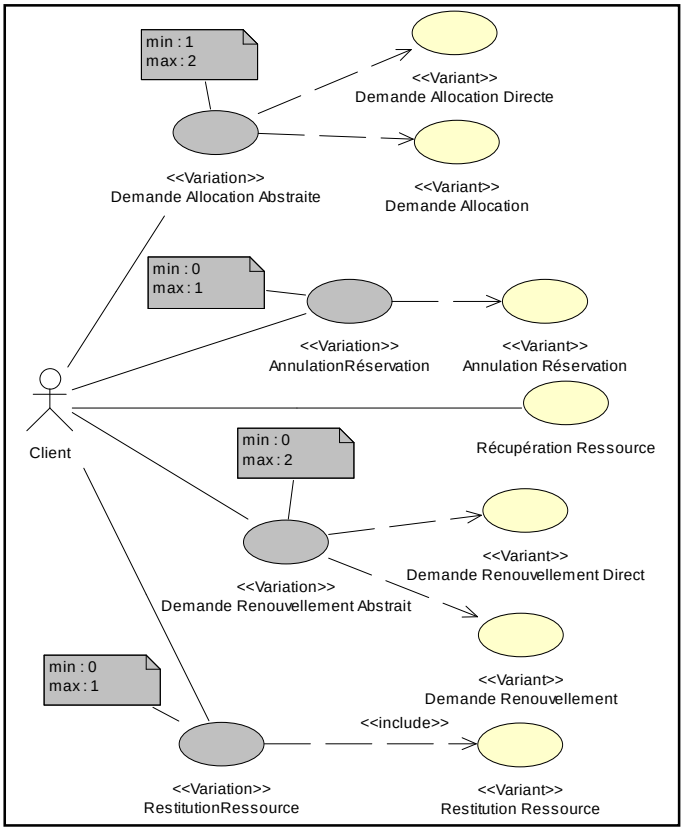
 
</td></tr>
</table>

✓ **Abstraction et analyse de similarités entre les PM**

Acteurs /Activités/Objets des PM	Acteurs/ Activités/Objets - Abstraction -
Acteurs	
Abonné	Client
Client	
Bibliothécaire	Allocation Manager
Agent	
Hôtesse d’accueil	
Agent de voyage	
Responsable de Vente	
Activités /événements	

Demande Chambre Directe	Demande Allocation Directe
Demande Ticket Spectacle	
Demande Emprunt Livre	Demande Allocation
Demande Location Voiture	
Demande Allocation Chambre	
Demande Billet Avion	
Traitement Demande	Traitement Sans Réservation
Création Réservation	Traitement Avec Réservation
Annulation Réservation	Annulation Réservation
Récupération Livre	Récupération Ressource
Récupération Voiture	
Occupation Chambre	
Retrait Billet	
Retrait Ticket	
Création Emprunt	Création Allocation
Création Contrat	
Création Séjour	
Création Billet	
Création Ticket	
Demande Renouvellement	Demande Renouvellement
Création Demande Prolongation	
Demande Prolongation Séjour	Demande Renouvellement Direct
Mise à Jour Emprunt	Mise à Jour Allocation
Mise à Jour Contrat	
Mise à Jour Séjour	
Restitution Livre	Restitution Ressource
Restitution Voiture	
Libération Chambre	
Enregistrement Retour Livre	Enregistrement Retour Ressource
Enregistrement Retour Voiture	
Enregistrement Départ	
Objets	
Demande Livre	Demande Allocation
Demande Voiture	
Demande Chambre	
Demande Billet	
Demande Ticket	
Réservation	Réservation
Emprunt	Allocation
Contrat	
Séjour	
Demande Renouvellement	Renouvellement
Demande Prolongation	
Livre	Ressource
Voiture	
Chambre	
Billet	
Ticket	

✓ **Modèle métier du PM « Gestion Allocation Ressource »**



✓ **Scénario variable**

Scénario	**- Variation** : Demande Allocation abstraite • **Variante :** *Demande Allocation Directe* **-** Le *client (acteur principal)* se présente directement pour demander sa ressource. - L'*Allocation Manager (acteur secondaire)* crée la demande d'allocation. • **Variante :** *Demande Allocation* - Le *client* crée la demande d'allocation via un formulaire électronique. **Variation** : Traitement Demande Allocation *L'Allocation Manager* traite la demande : 1. Identifie le client par le biais du code demande. 2. Vérifie la disponibilité de la ressource. • **Variante :** *Traitement Avec Réservation* - Crée une réservation et communique au client le code de réservation. • **Variante :** *Traitement Sans Réservation* - Confirme au client la disponibilité de la ressource à la date demandée.
	- **Variation** : Annulation Réservation • **Variante** : *Annulation Réservation*

	- Le client peut annuler sa réservation en la supprimant.
	- Le *client* se présente pour récupérer la ressource. L'*Allocation Manager* : 1. **Variation :** Allocation avec Réservation • **Variante** : *Vérification Réservation* - Vérifie le code de réservation. 2. Crée l'allocation. 3. **Variation :** Modification État Ressource • **Variante** : *Suppression Ressource* - Supprime la ressource *(*ressource consommable). • **Variante** : *Mise à jour Ressource* - Met à jour l'état de la ressource (ressource non consommable).
	- Variation : Demande Renouvellement Abstrait • **Variante** : *Demande Renouvellement Direct* - Le *client* se présente directement pour demander le renouvellement de l'allocation. - L'*Allocation Manager* crée la demande de renouvellement. • **Variante** : *Demande Renouvellement* - Le *client* peut créer une demande de renouvellement d'une allocation. - L'*Allocation Manager* : 1. Identifie l'allocation. 2. Vérifie la disponibilité de la ressource. 3. Met à jour l'allocation.
	- Variation : Restitution Ressource • **Variante** : *Restitution Ressource* - Le *client* restitue la ressource. - L'*Allocation Manager* : 1. Récupère la ressource. 2. Enregistre le retour de la ressource.

✓ **Variabilité identifiée**

Points de variation	**Type**	**Cardinalités**	**Variantes**
PV $_1$: Demande Allocation Abstraite	Degré d'informatisation / Ensemble d'alternatives	Min : 1 Max : 2	V $_{11}$: Demande Allocation / Informatisée
			V $_{12}$: Demande Allocation Directe / Manuelle
PV $_2$: Traitement Demande Allocation	Alternative	Min : 1 Max : 1	V $_{21}$: Traitement Avec Réservation
			V $_{22}$: Traitement Sans Réservation

PV 3 : Annulation Réservation	Option	Min : 0 Max : 1	V 31 : Annulation Réservation
PV 4 : Allocation Avec Réservation	Option	Min : 0 Max : 1	V41 : Vérification Réservation
PV5 : Modification État Ressource	Alternative	Min : 1 Max : 1	V 51 : Mise à Jour Ressource
			V 52 : Suppression Ressource
PV 6 : Demande Renouvellement Abstrait	Degré d'informatisation / Option	Min : 0 Max : 2	V 61 : Demande Renouvellement / Informatisée
			V 62 : Demande Renouvellement Direct / Manuelle
PV 7 : Restitution Ressource	Option	Min : 0 Max : 1	V 71 : Restitution Ressource

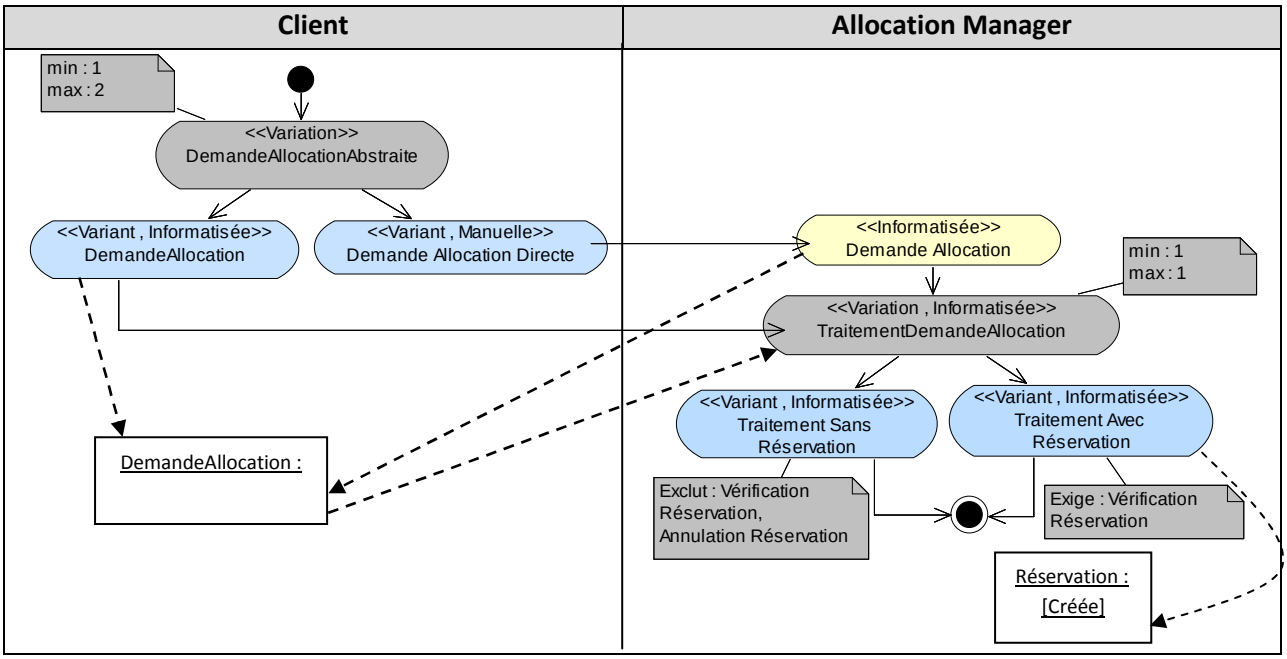
✓ **Contraintes de dépendances**

	V 11	V 12	V 21	V 22	V 31	V 41	V 51	V 52	V 61	V 62	V 71
V 11											
V 12											
V 21						Exige					
V 22					Exclut	Exclut					
V 31			Exige	Exclut							
V 41			Exige	Exclut							
V 51											Exige
V 52									Exclut	Exclut	Exclut
V 61							Exige	Exclut			
V 62							Exige	Exclut			
V 71							Exige	Exclut			

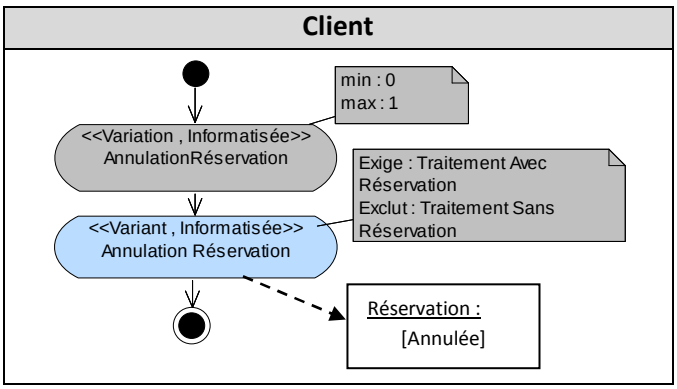
✓ **Description du PM générique**

2. Vue Métier

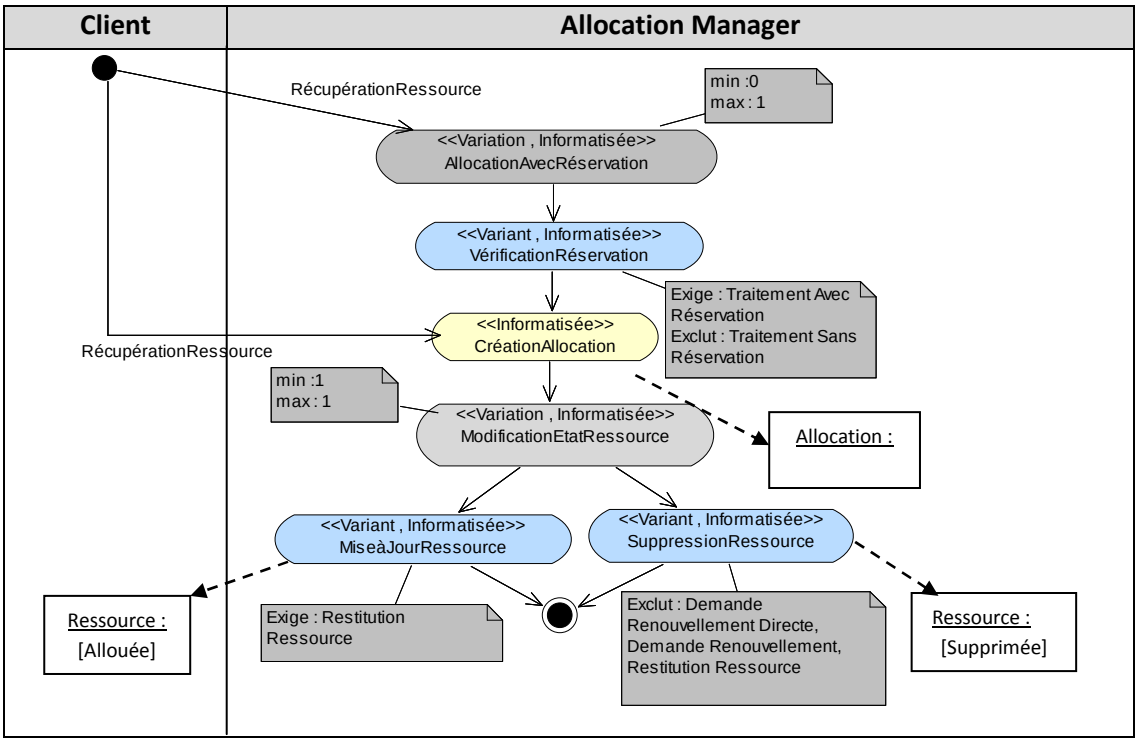
« Demande Allocation Abstraite »



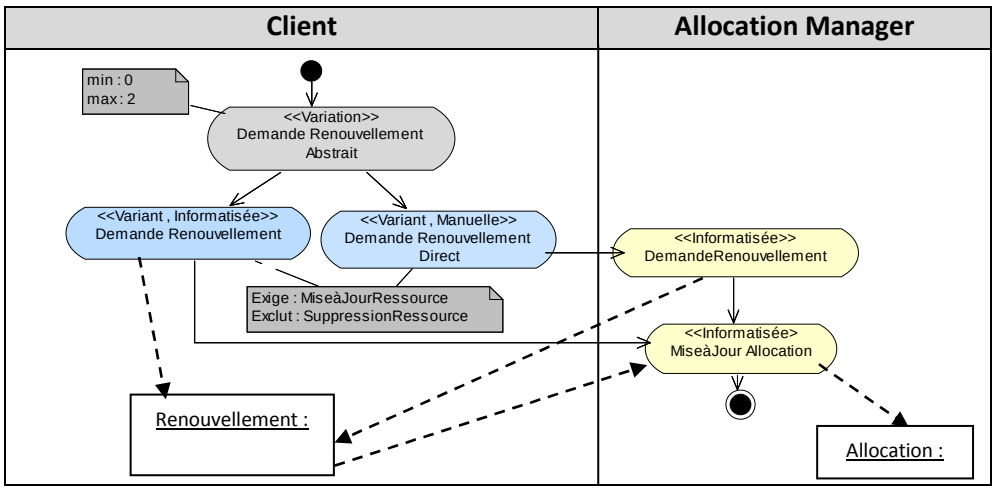
« Annulation Réservation »



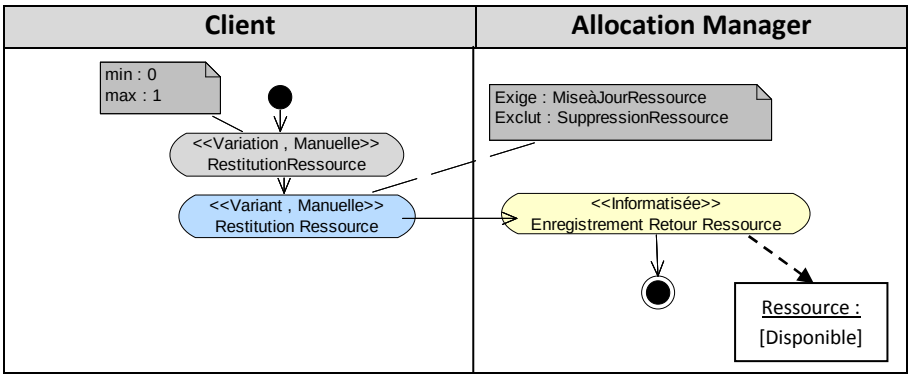
« Récupération Ressource»



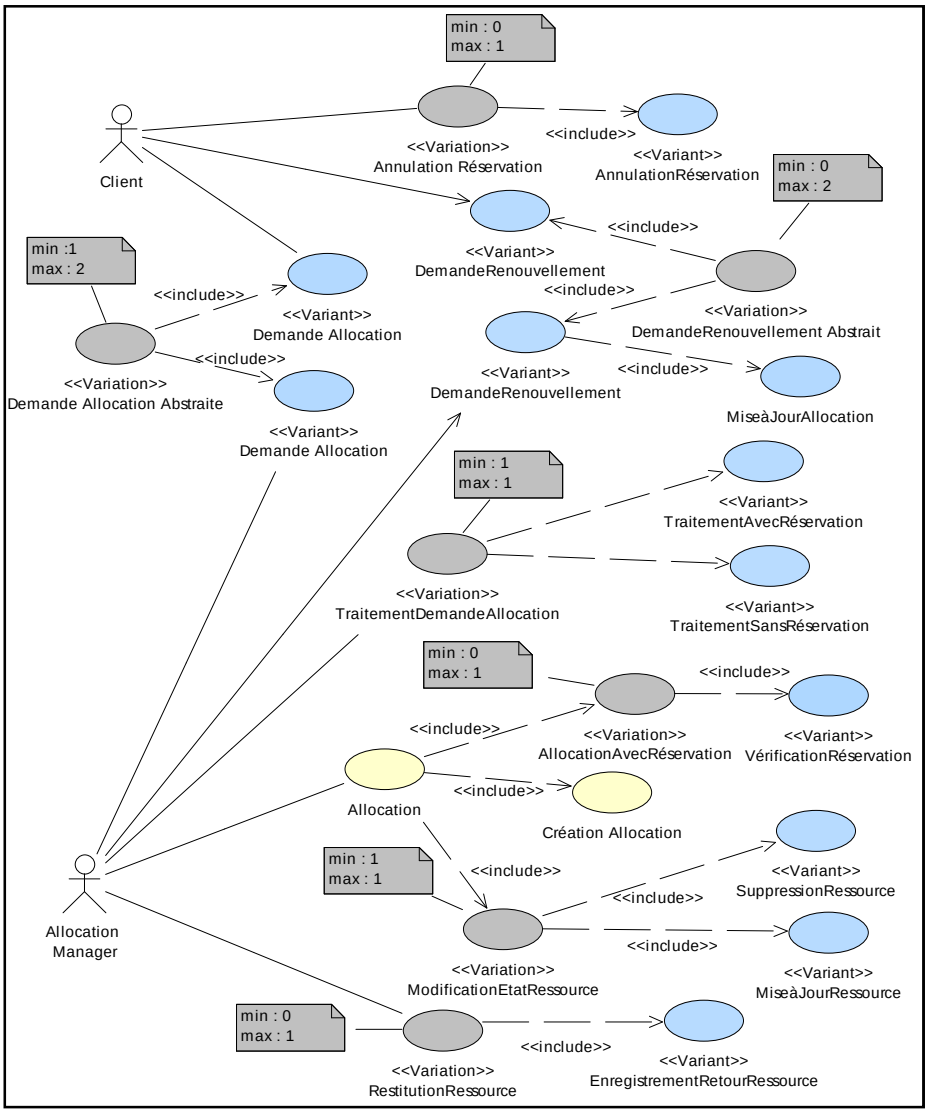
« Demande Renouvellement Abstrait »



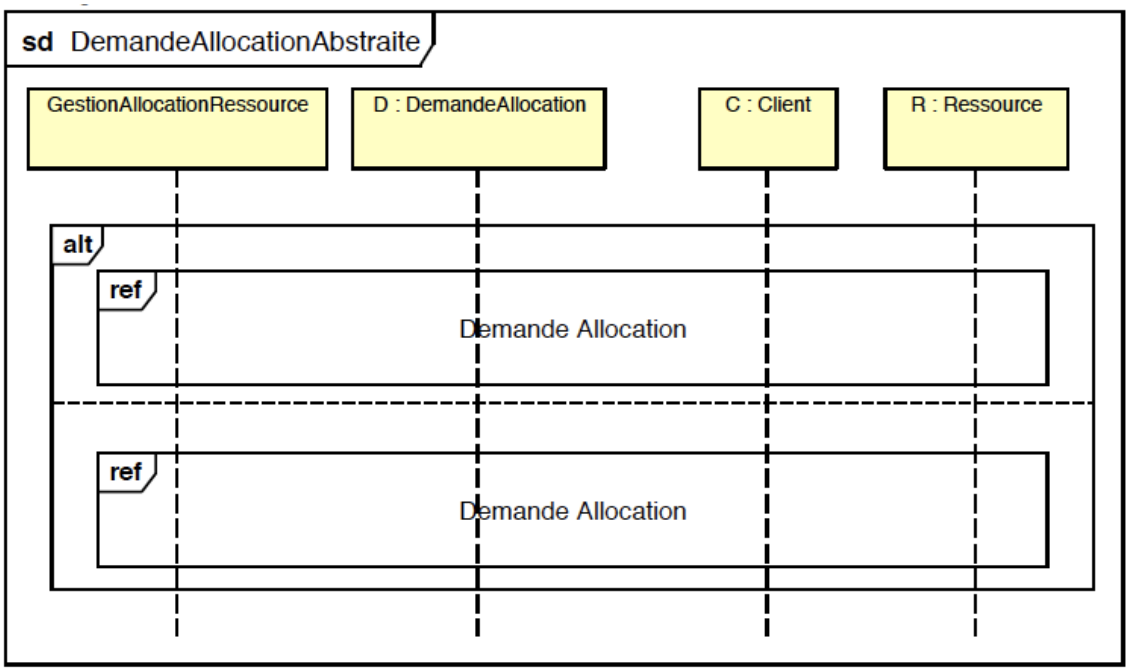
« Restitution Ressource »

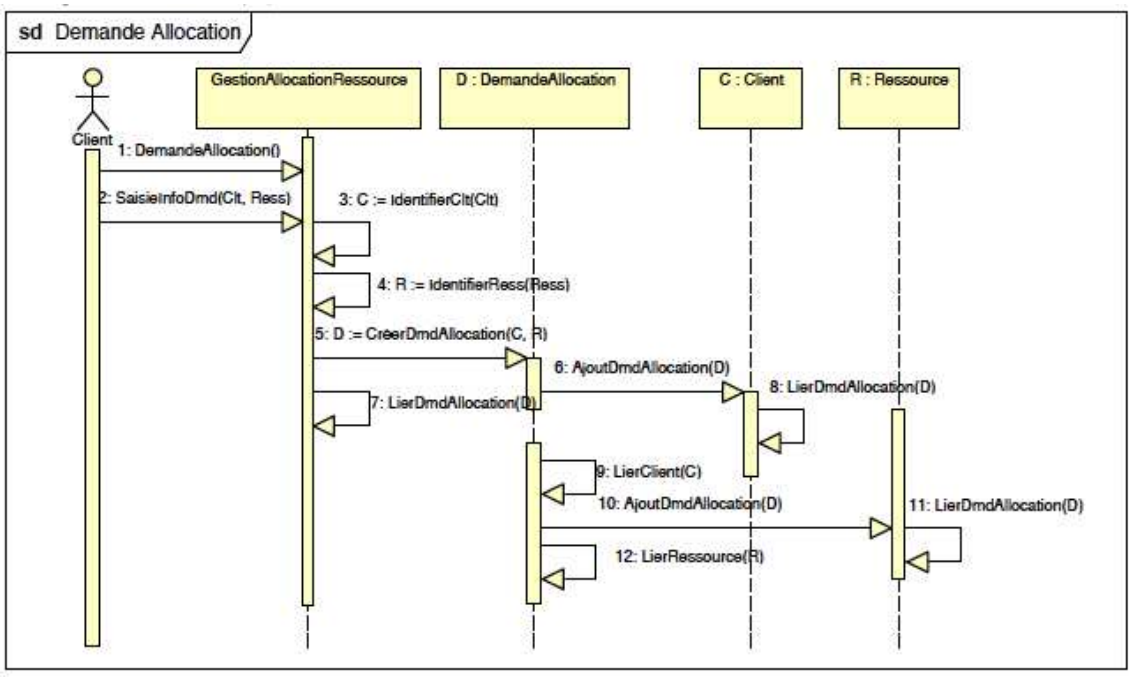


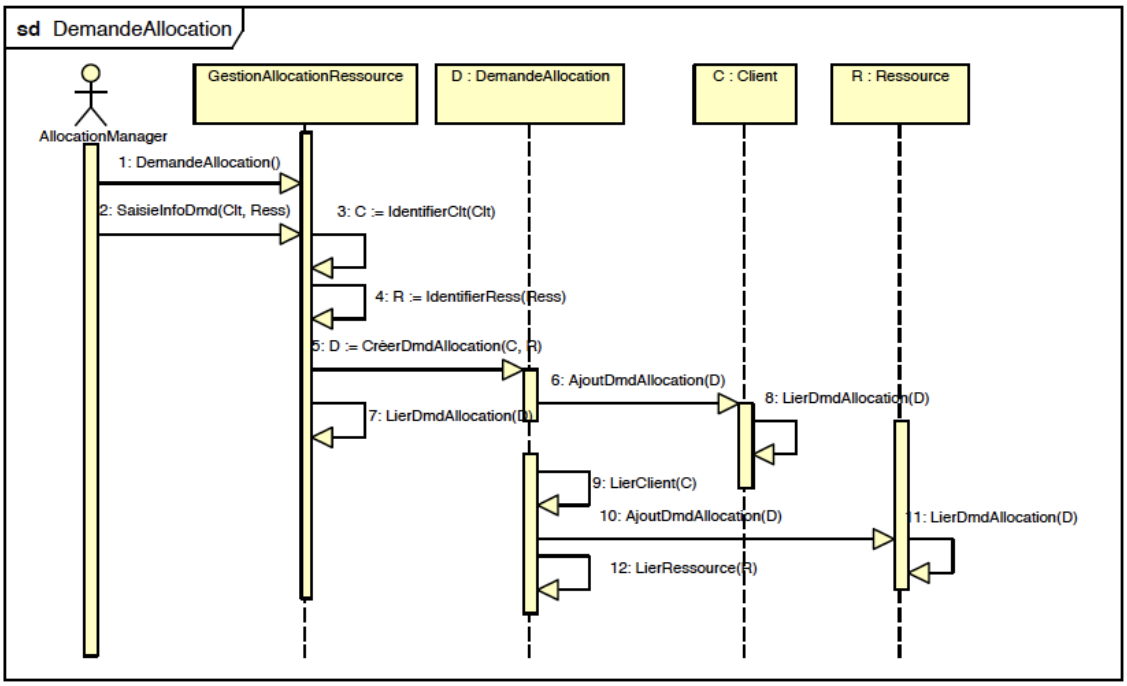
3. Vue fonctionnelle

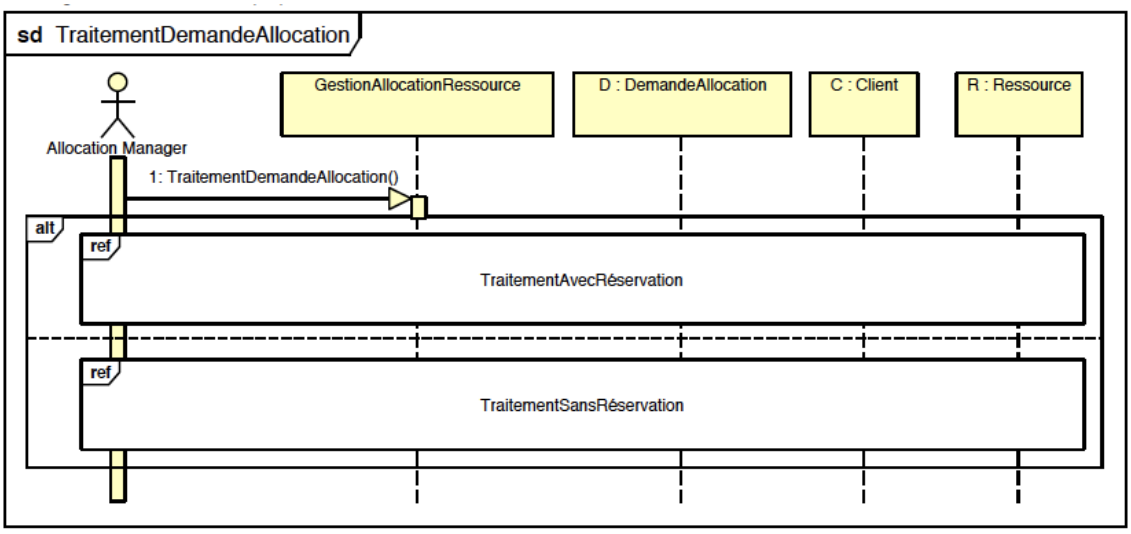


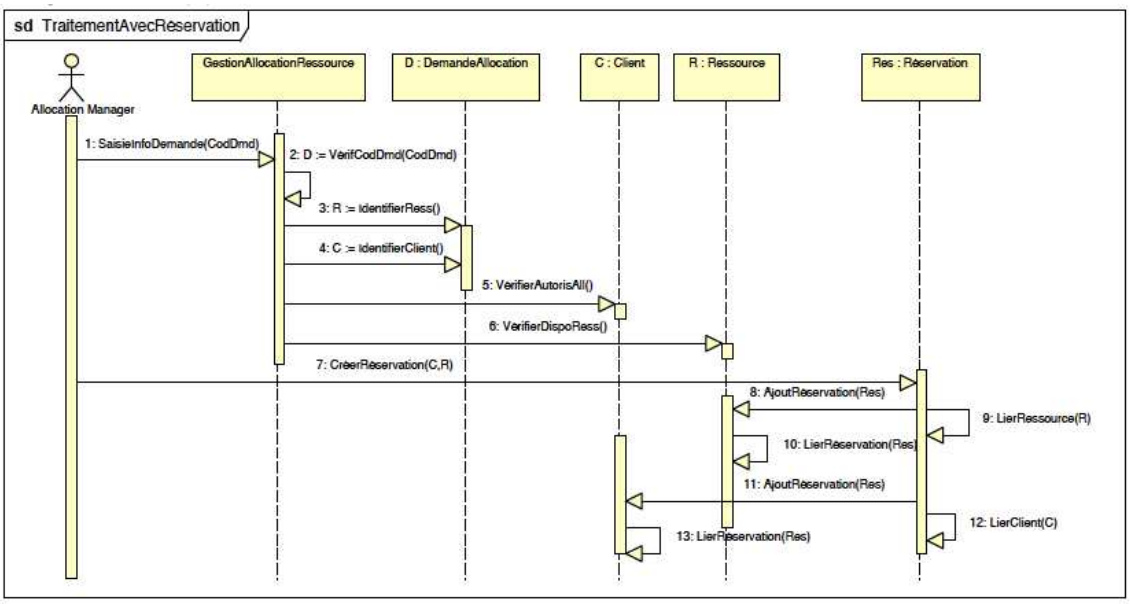
4. Vue Dynamique

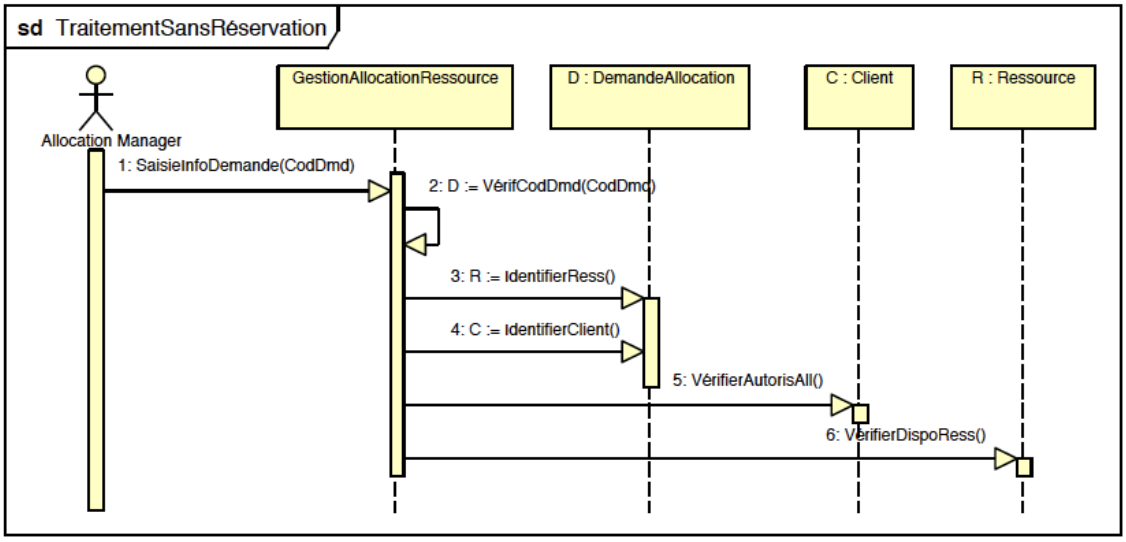


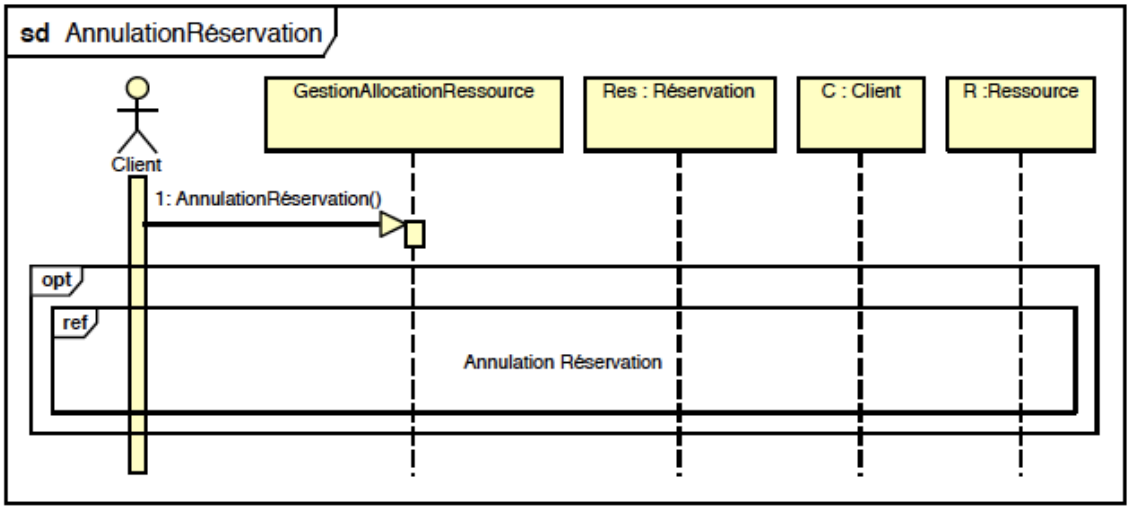


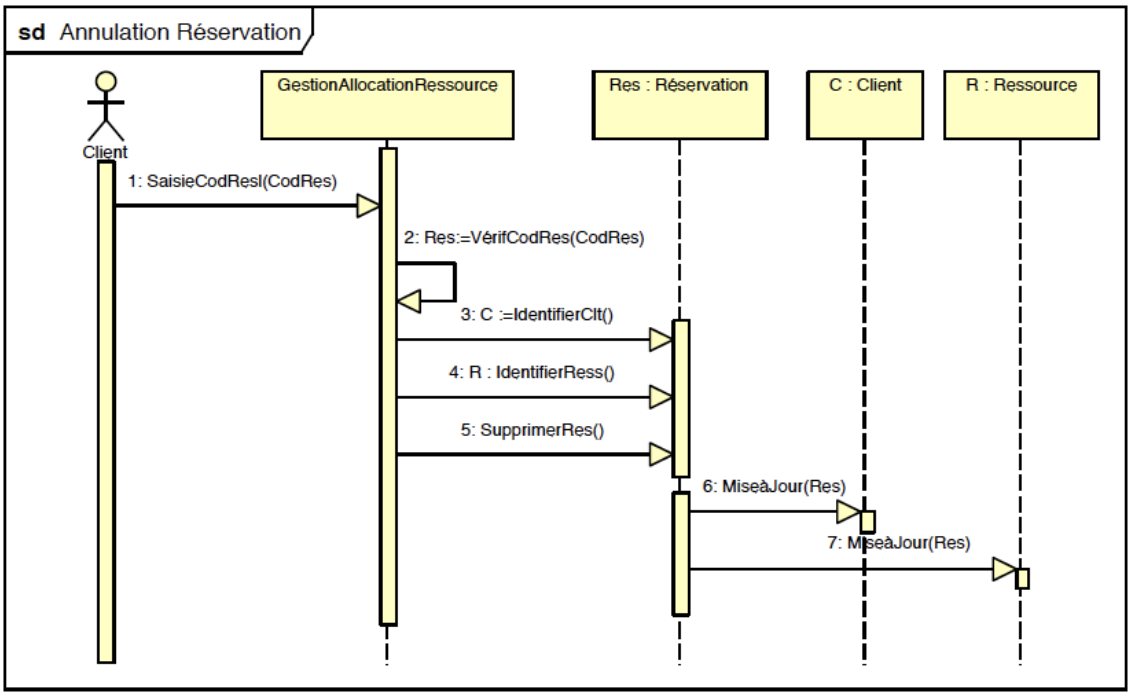

sd DemandeAllocation
AllocationManager
GestionAllocationRessource
D : DemandeAllocation
C : Client
R : Ressource
1: DemandeAllocation()
2: SaisieInfoDmd(Clt, Ress)
3: C := IdentifierClt(Clt)
4: R := IdentifierRess(Ress)
5: D := CréerDmdAllocation(C, R)
6: AjoutDmdAllocation(D)
7: LierDmdAllocation(D)
8: LierDmdAllocation(D)
9: LierClient(C)
10: AjoutDmdAllocation(D)
11: LierDmdAllocation(D)
12: LierRessource(R)

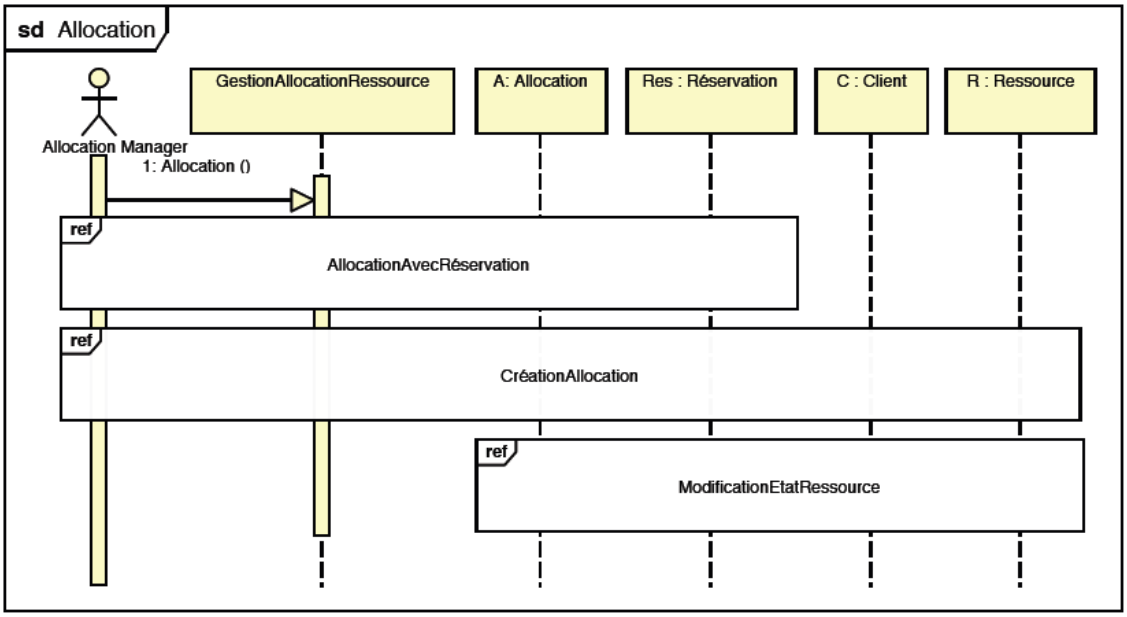

sd TraitementDemandeAllocation
Allocation Manager
GestionAllocationRessource
D : DemandeAllocation
C : Client
R : Ressource
1: TraitementDemandeAllocation()
alt
ref
TraitementAvecRéservation
ref
TraitementSansRéservation

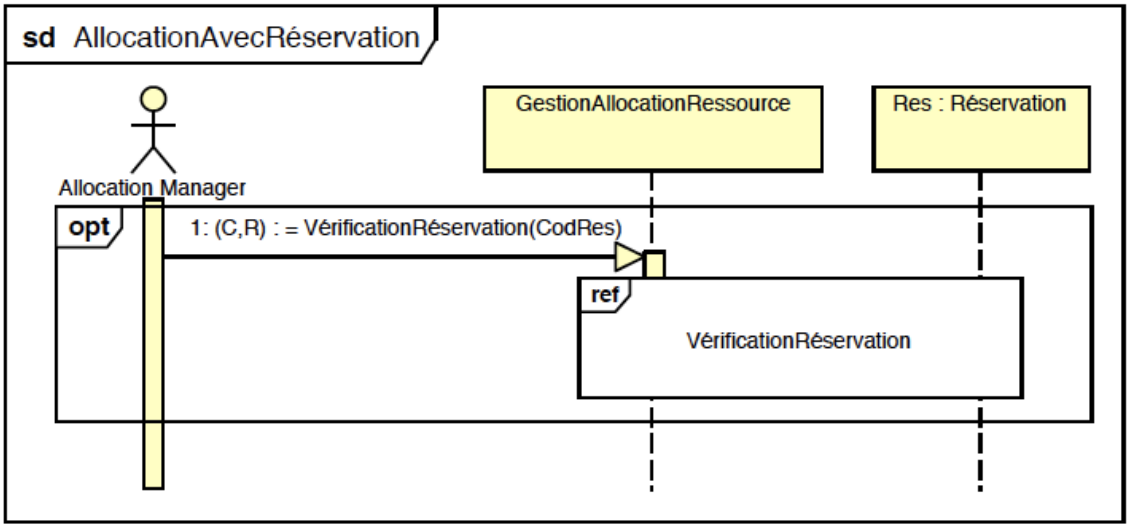

sd TraitementAvecReservation
Allocation Manager
GestionAllocationRessource
D : DemandeAllocation
C : Client
R : Ressource
Res : Réservation
1: SaisieInfoDemande(CodDmd)
2: D := VérifCodDmd(CodDmd)
3: R := IdentifierRess()
4: C := IdentifierClient()
5: VerifierAutorisAll()
6: VerifierDispoRess()
7: CréerReservation(C,R)
8: AjoutReservation(Res)
9: LierRessource(R)
10: LierReservation(Res)
11: AjoutReservation(Res)
12: LierClient(C)
13: LierReservation(Res)

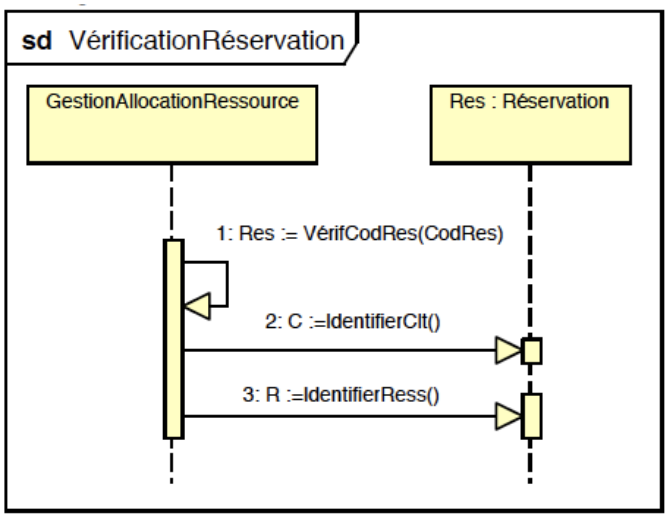

sd TraitementSansRéservation
Allocation Manager
GestionAllocationRessource
D : DemandeAllocation
C : Client
R : Ressource
1: SaisieInfoDemande(CodDmd)
2: D := VérifCodDmd(CodDmd)
3: R := IdentifierRess()
4: C := IdentifierClient()
5: VérifierAutorisAll()
6: VérifierDispoRess()

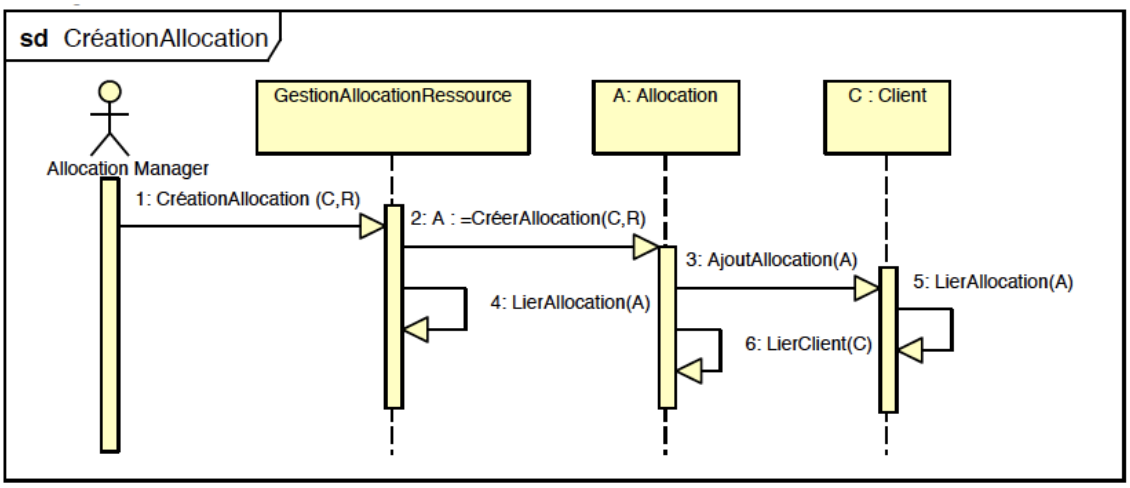

sd AnnulationRéservation
Client
GestionAllocationRessource
Res : Réservation
C : Client
R :Ressource
1: AnnulationRéservation()
opt
ref
Annulation Réservation

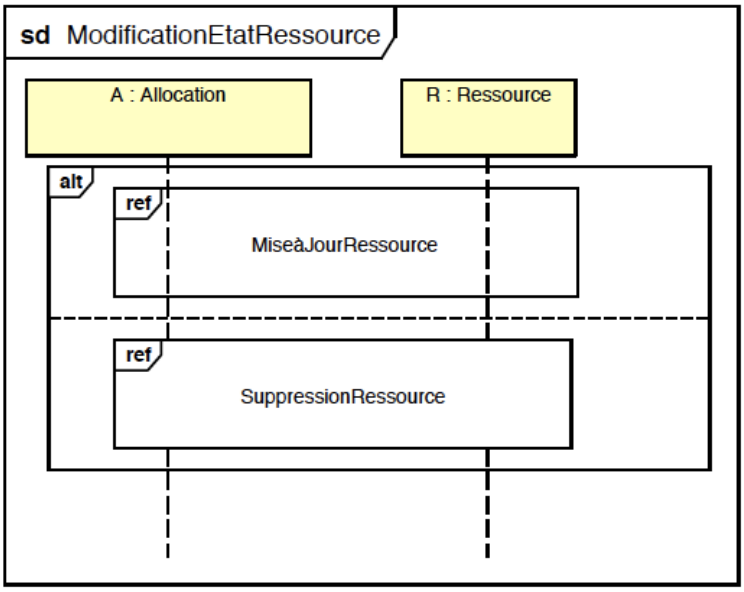

sd Annulation Réservation
Client
GestionAllocationRessource
Res : Réservation
C : Client
R : Ressource
1: SaisieCodResl(CodRes)
2: Res:=VérifCodRes(CodRes)
3: C :=IdentifierClt()
4: R : IdentifierRess()
5: SupprimerRes()
6: MiseàJour(Res)
7: MiseàJour(Res)

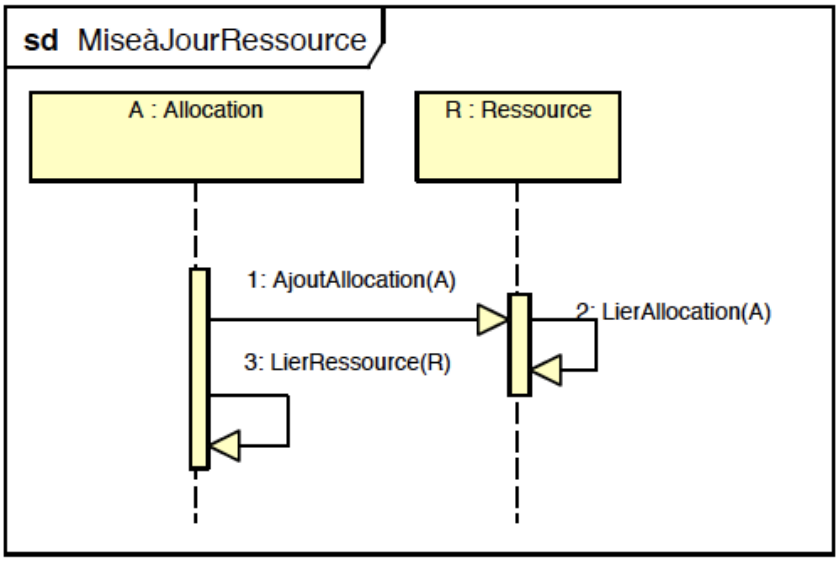

sd Allocation
Allocation Manager
GestionAllocationRessource
A: Allocation
Res : Réservation
C : Client
R : Ressource
1: Allocation ()
ref
AllocationAvecRéservation
ref
CréationAllocation
ref
ModificationEtatRessource

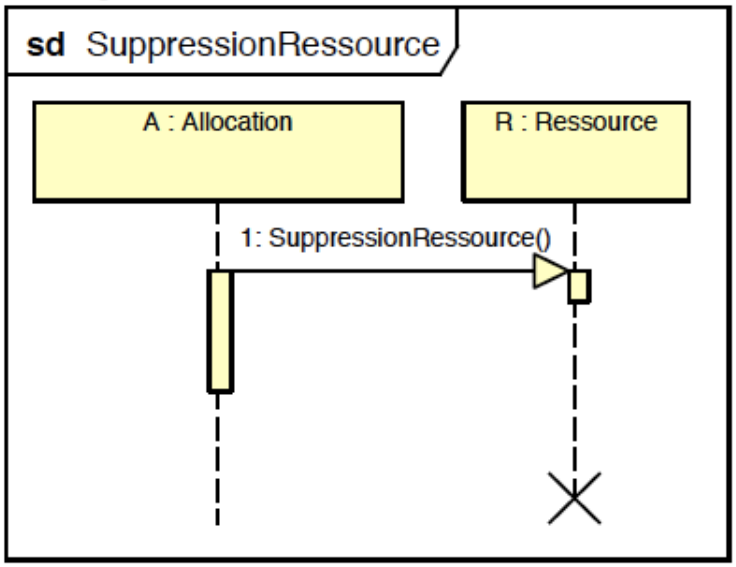

sd AllocationAvecRéservation
Allocation Manager
GestionAllocationRessource
Res : Réservation
opt
1: (C,R) : = VérificationRéservation(CodRes)
ref
VérificationRéservation

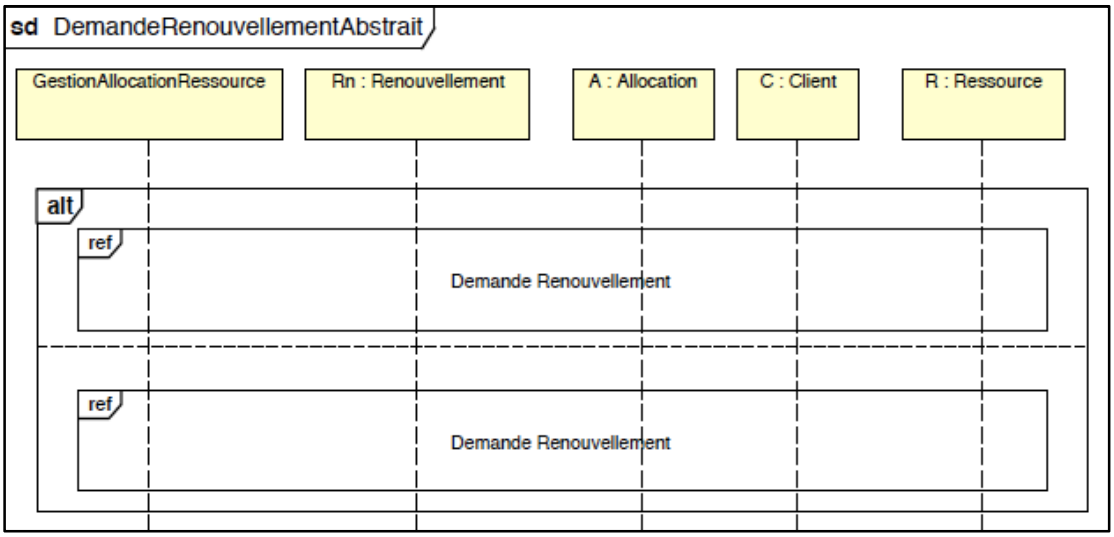

sd VérificationRéservation
GestionAllocationRessource
Res : Réservation
1: Res := VérifCodRes(CodRes)
2: C :=IdentifierClt()
3: R :=IdentifierRess()

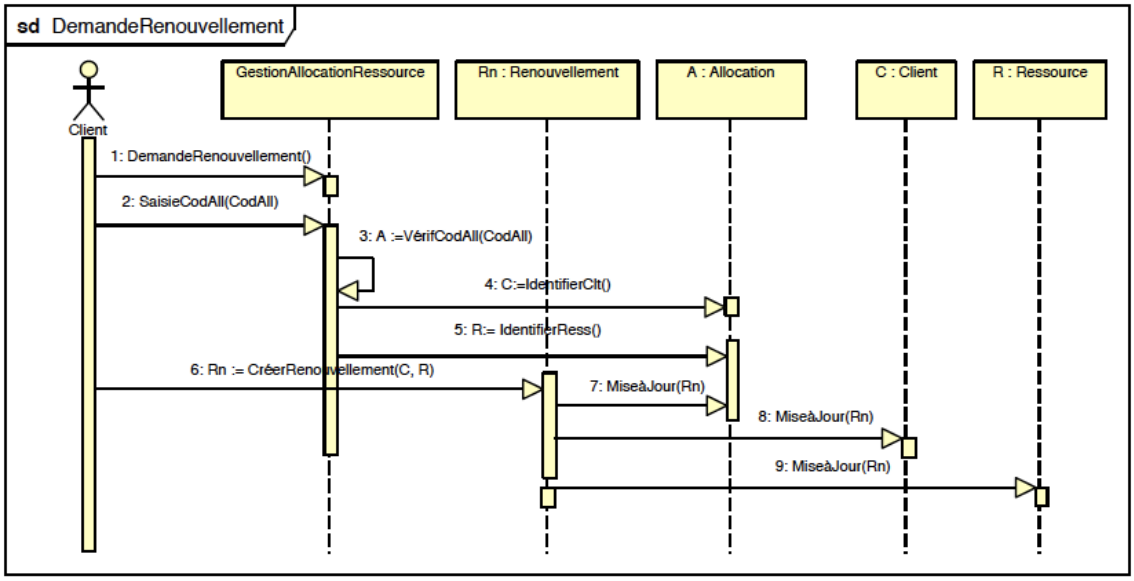

sd CréationAllocation
Allocation Manager
GestionAllocationRessource
A: Allocation
C : Client
1: CréationAllocation (C,R)
2: A : =CréerAllocation(C,R)
3: AjoutAllocation(A)
4: LierAllocation(A)
5: LierAllocation(A)
6: LierClient(C)

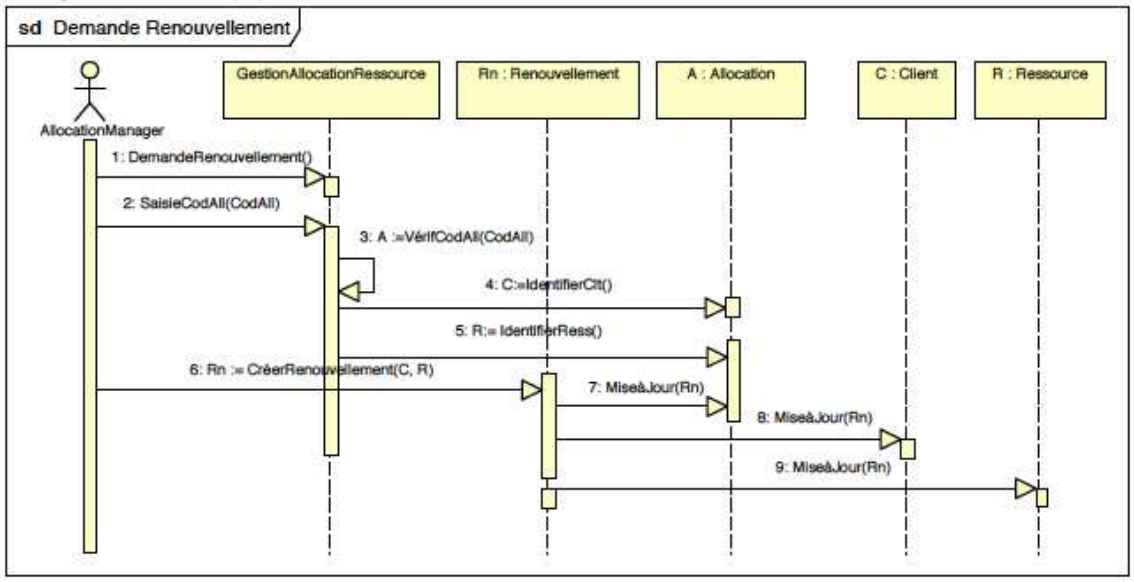

sd ModificationEtatRessource
A : Allocation
R : Ressource
alt
ref
MiseàJourRessource
ref
SuppressionRessource

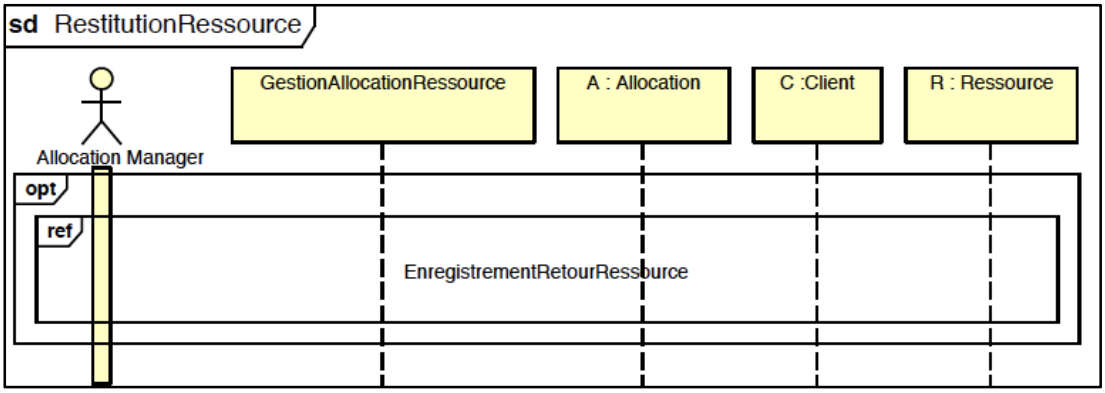

sd MiseàJourRessource
A : Allocation
R : Ressource
1: AjoutAllocation(A)
2: LierAllocation(A)
3: LierRessource(R)

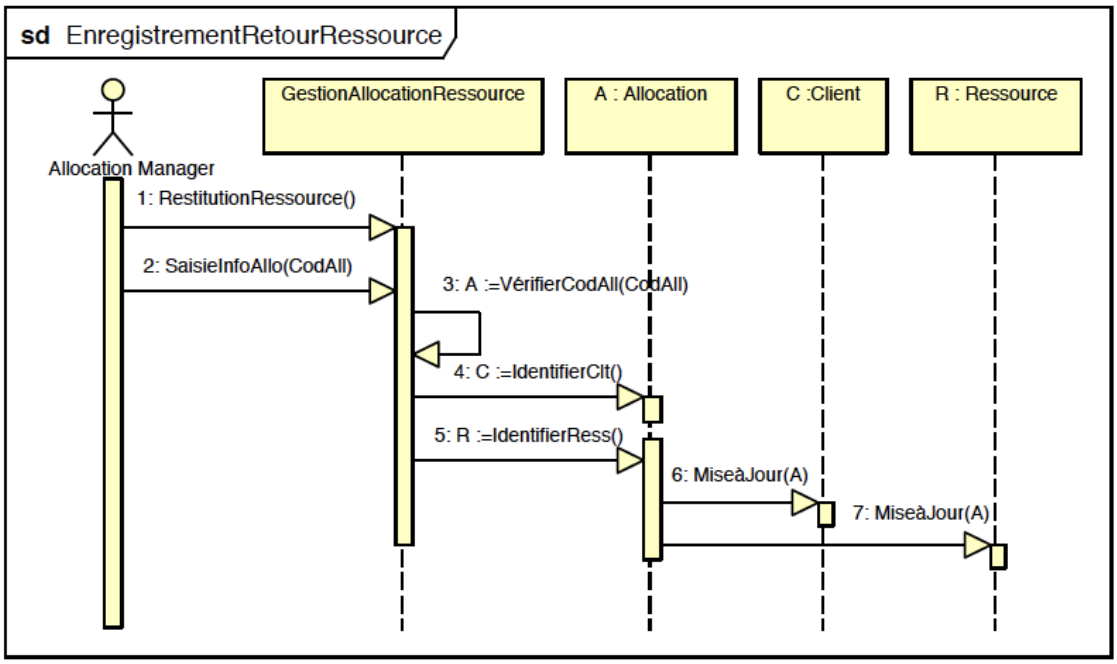

sd SuppressionRessource
A : Allocation
R : Ressource
1: SuppressionRessource()


sd DemandeRenouvellementAbstrait
GestionAllocationRessource
Rn : Renouvellement
A : Allocation
C : Client
R : Ressource
alt
ref
Demande Renouvellement
ref
Demande Renouvellement


sd DemandeRenouvellement
Client
GestionAllocationRessource
Rn : Renouvellement
A : Allocation
C : Client
R : Ressource
1: DemandeRenouvellement()
2: SaisieCodAll(CodAll)
3: A :=VérifCodAll(CodAll)
4: C:=IdentifierClt()
5: R:= IdentifierRess()
6: Rn := CréerRenouvellement(C, R)
7: MiseàJour(Rn)
8: MiseàJour(Rn)
9: MiseàJour(Rn)

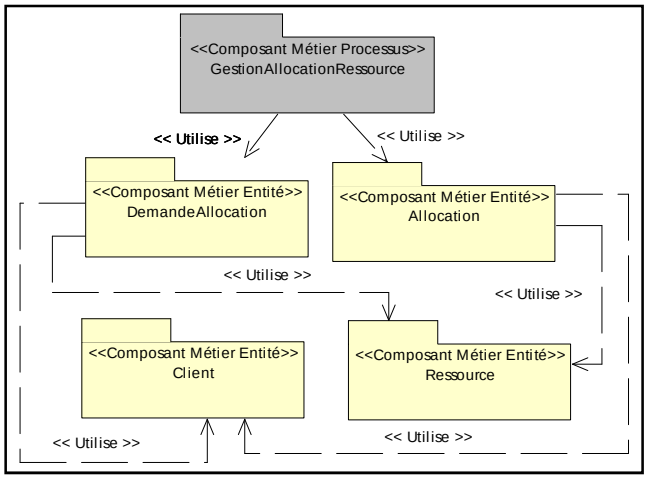

sd Demande Renouvellement
AllocationManager
GestionAllocationRessource
Rn : Renouvellement
A : Allocation
C : Client
R : Ressource
1: DemandeRenouvellement()
2: SaisieCodAll(CodAll)
3: A :=VérifCodAll(CodAll)
4: C:=IdentifierClt()
5: R:= IdentifierRess()
6: Rn := CréerRenouvellement(C, R)
7: MiseàJour(Rn)
8: MiseàJour(Rn)
9: MiseàJour(Rn)

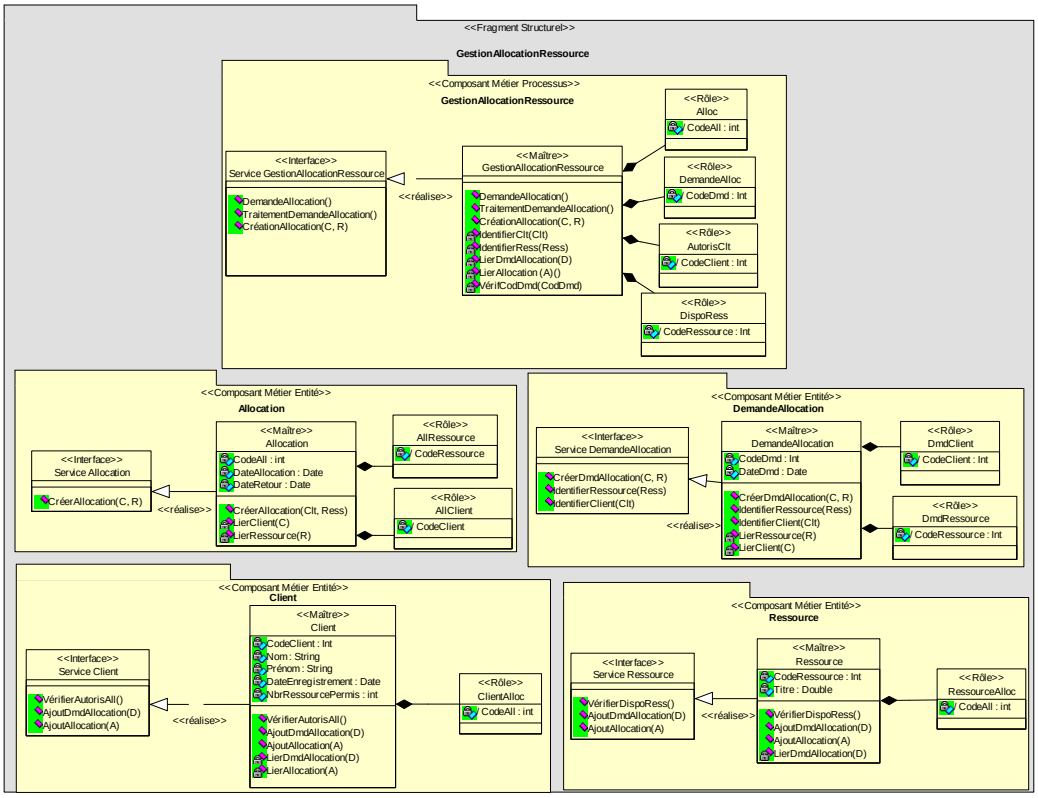

sd RestitutionRessource
Allocation Manager
GestionAllocationRessource
A : Allocation
C :Client
R : Ressource
opt
ref
EnregistrementRetourRessource



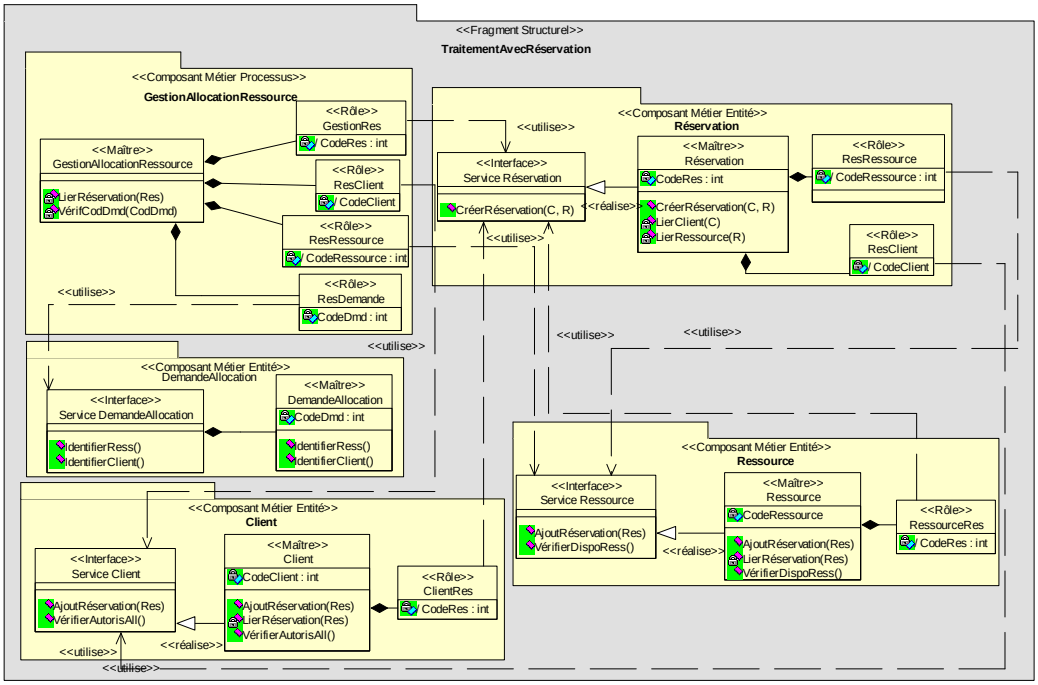
5. Vue structurelle

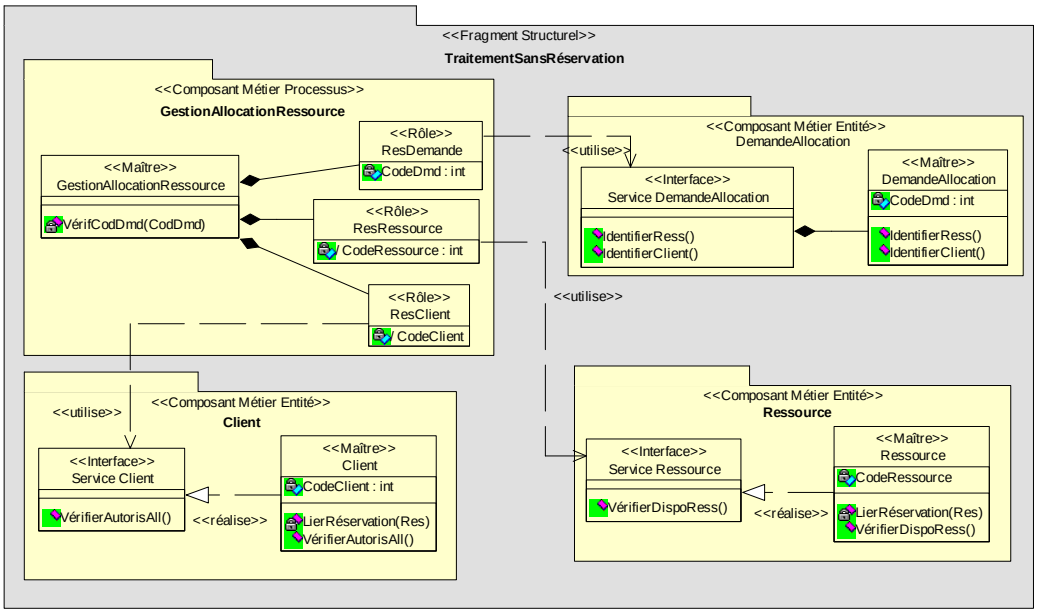
b) Fragment structurel fixe

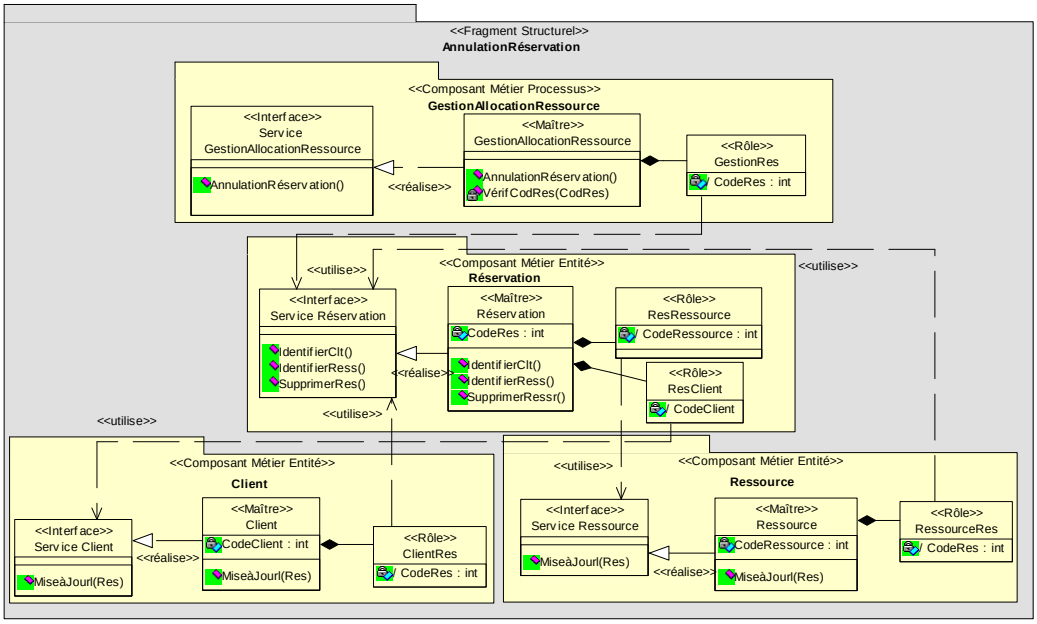


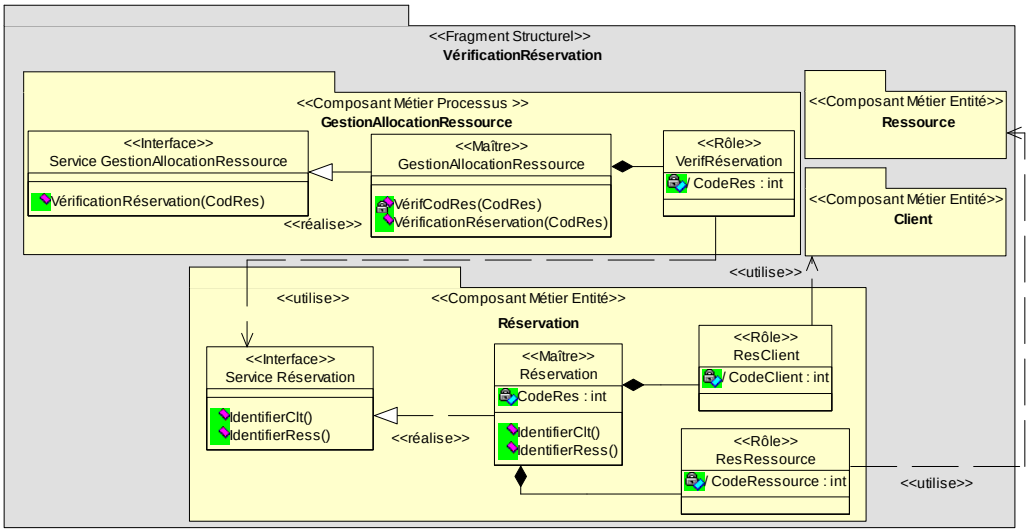


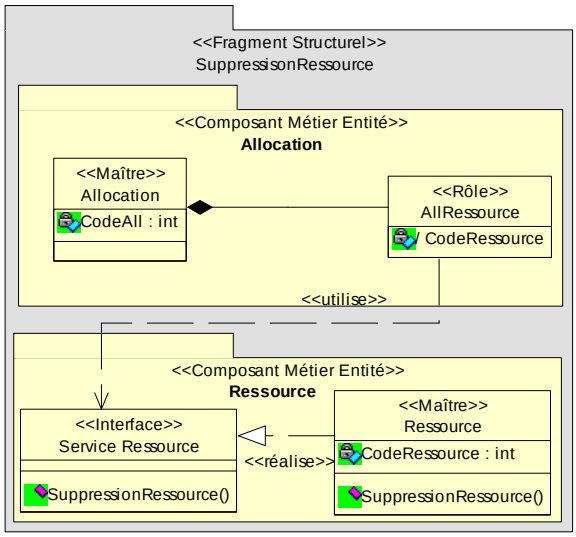
c) Fragments structurels variantes

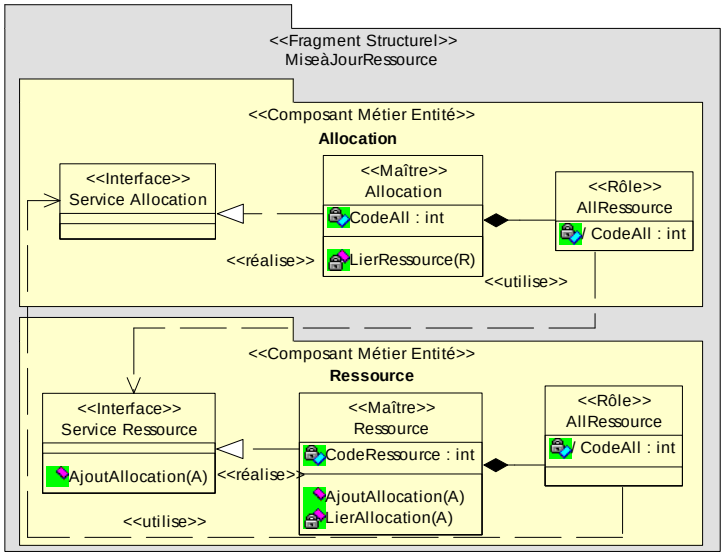


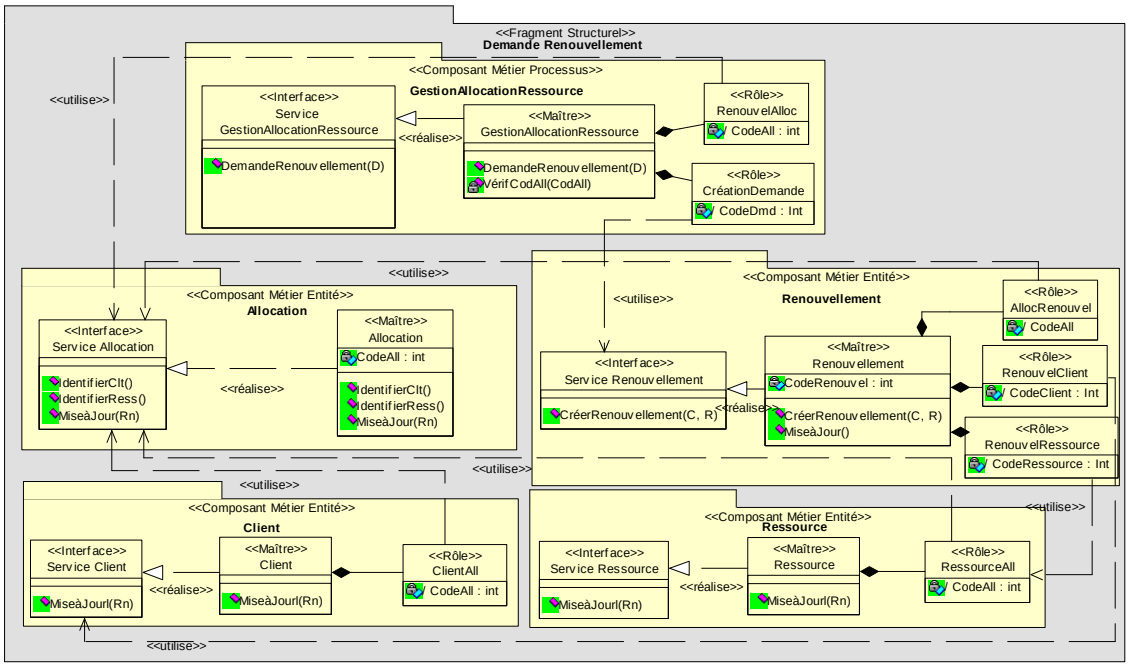

<<Fragment Structurel>>
TraitementSansRéservation
<<Composant Métier Processus>>
GestionAllocationRessource
<<Maître>>
GestionAllocationRessource
VérifCodDmd(CodDmd)
<<Rôle>>
ResDemande
CodeDmd : int
<<Rôle>>
ResRessource
CodeRessource : int
<<Rôle>>
ResClient
CodeClient
<<utilise>>
<<Composant Métier Entité>>
DemandeAllocation
<<Interface>>
Service DemandeAllocation
IdentifierRess()
IdentifierClient()
<<Maître>>
DemandeAllocation
CodeDmd : int
IdentifierRess()
IdentifierClient()
<<utilise>>
<<Composant Métier Entité>>
Client
<<Interface>>
Service Client
VérifierAutorisAll()
<<réalise>>
<<Maître>>
Client
CodeClient : int
LierRéservation(Res)
VérifierAutorisAll()
<<Composant Métier Entité>>
Ressource
<<Interface>>
Service Ressource
VérifierDispoRess()
<<réalise>>
<<Maître>>
Ressource
CodeRessource
LierRéservation(Res)
VérifierDispoRess()

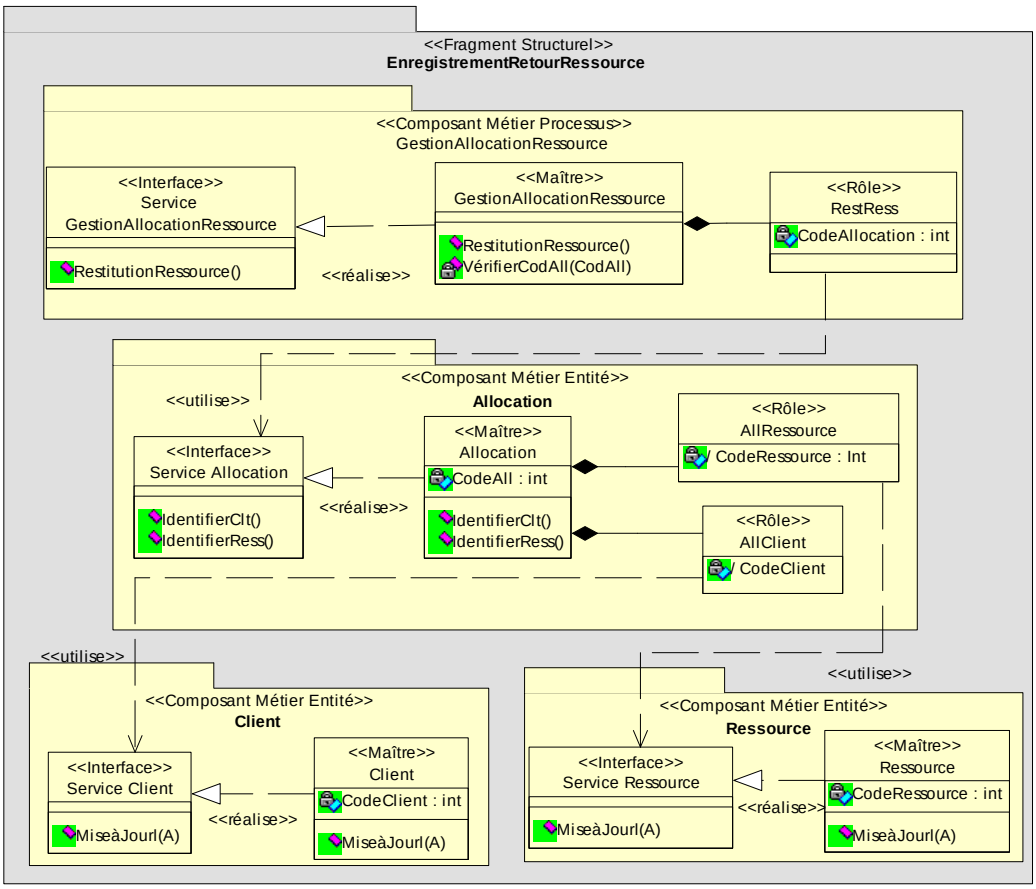

<<Fragment Structurel>>
AnnulationRéservation
<<Composant Métier Processus>>
GestionAllocationRessource
<<Interface>>
Service GestionAllocationRessource
AnnulationRéservation()
<<réalise>>
<<Maître>>
GestionAllocationRessource
AnnulationRéservation()
VérifCodRes(CodRes)
<<Rôle>>
GestionRes
CodeRes : int
<<utilise>>
<<Composant Métier Entité>>
Réservation
<<Interface>>
Service Réservation
IdentifierClt()
IdentifierRess()
SupprimerRes()
<<réalise>>
<<Maître>>
Réservation
CodeRes : int
IdentifierClt()
IdentifierRess()
SupprimerRessr()
<<Rôle>>
ResRessource
CodeRessource : int
<<Rôle>>
ResClient
CodeClient
<<utilise>>
<<Composant Métier Entité>>
Client
<<Interface>>
Service Client
MiseàJourl(Res)
<<réalise>>
<<Maître>>
Client
CodeClient : int
MiseàJourl(Res)
<<Rôle>>
ClientRes
CodeRes : int
<<Composant Métier Entité>>
Ressource
<<Interface>>
Service Ressource
MiseàJourl(Res)
<<réalise>>
<<Maître>>
Ressource
CodeRessource : int
MiseàJourl(Res)
<<Rôle>>
RessourceRes
CodeRes : int


<<Fragment Structurel>>
VérificationRéservation
<<Composant Métier Processus >>
GestionAllocationRessource
<<Interface>>
Service GestionAllocationRessource
VérificationRéservation(CodRes)
<<réalise>>
<<Maître>>
GestionAllocationRessource
VérifCodRes(CodRes)
VérificationRéservation(CodRes)
<<Rôle>>
VerifRéservation
CodeRes : int
<<Composant Métier Entité>>
Ressource
<<Composant Métier Entité>>
Client
<<utilise>>
<<utilise>>
<<Composant Métier Entité>>
Réservation
<<Interface>>
Service Réservation
IdentifierClt()
IdentifierRess()
<<réalise>>
<<Maître>>
Réservation
CodeRes : int
IdentifierClt()
IdentifierRess()
<<Rôle>>
ResClient
CodeClient : int
<<Rôle>>
ResRessource
CodeRessource : int
<<utilise>>


<<Fragment Structurel>>
SuppressisonRessource
<<Composant Métier Entité>>
Allocation
<<Maître>>
Allocation
CodeAll : int
<<Rôle>>
AllRessource
/ CodeRessource
<<utilise>>
<<Composant Métier Entité>>
Ressource
<<Interface>>
Service Ressource
SuppressionRessource()
<<réalise>>
<<Maître>>
Ressource
CodeRessource : int
SuppressionRessource()


<<Fragment Structurel>>
MiseàJourRessource
<<Composant Métier Entité>>
Allocation
<<Interface>>
Service Allocation
<<Maître>>
Allocation
CodeAll : int
LierRessource(R)
<<Rôle>>
AllRessource
/ CodeAll : int
<<réalise>>
<<utilise>>
<<Composant Métier Entité>>
Ressource
<<Interface>>
Service Ressource
AjoutAllocation(A)
<<Maître>>
Ressource
CodeRessource : int
AjoutAllocation(A)
LierAllocation(A)
<<Rôle>>
AllRessource
CodeAll : int
<<réalise>>
<<utilise>>


<<Fragment Structurel>>
Demande Renouvellement
<<Composant Métier Processus>>
GestionAllocationRessource
<<utilise>>
<<Interface>>
Service
GestionAllocationRessource
DemandeRenouvellement(D)
<<réalise>>
<<Maître>>
GestionAllocationRessource
DemandeRenouvellement(D)
VérifCodAll(CodAll)
<<Rôle>>
RenouvelAlloc
CodeAll : int
<<Rôle>>
CréationDemande
CodeDmd : Int
<<utilise>>
<<Composant Métier Entité>>
Allocation
<<Interface>>
Service Allocation
IdentifierClt()
IdentifierRess()
MiseàJour(Rn)
<<réalise>>
<<Maître>>
Allocation
CodeAll : int
IdentifierClt()
IdentifierRess()
MiseàJour(Rn)
<<utilise>>
<<Composant Métier Entité>>
Renouvellement
<<Interface>>
Service Renouvellement
CréerRenouvellement(C, R)
<<réalise>>
<<Maître>>
Renouvellement
CodeRenouvel : int
CréerRenouvellement(C, R)
MiseàJour()
<<Rôle>>
AllocRenouvel
CodeAll
<<Rôle>>
RenouvelClient
/ CodeClient : Int
<<Rôle>>
RenouvelRessource
CodeRessource : Int
<<utilise>>
<<utilise>>
<<Composant Métier Entité>>
Client
<<Interface>>
Service Client
MiseàJourl(Rn)
<<réalise>>
<<Maître>>
Client
MiseàJourl(Rn)
<<Rôle>>
ClientAll
CodeAll : int
<<utilise>>
<<Composant Métier Entité>>
Ressource
<<Interface>>
Service Ressource
MiseàJourl(Rn)
<<réalise>>
<<Maître>>
Ressource
MiseàJourl(Rn)
<<Rôle>>
RessourceAll
CodeAll : int
<<utilise>>


<<Fragment Structurel>>
EnregistrementRetourRessource
<<Composant Métier Processus>>
GestionAllocationRessource
<<Interface>>
Service
GestionAllocationRessource
RestitutionRessource()
<<réalise>>
<<Maître>>
GestionAllocationRessource
RestitutionRessource()
VérifierCodAll(CodAll)
<<Rôle>>
RestRess
CodeAllocation : int
<<utilise>>
<<Composant Métier Entité>>
Allocation
<<Interface>>
Service Allocation
IdentifierClt()
IdentifierRess()
<<réalise>>
<<Maître>>
Allocation
CodeAll : int
IdentifierClt()
IdentifierRess()
<<Rôle>>
AllRessource
CodeRessource : Int
<<Rôle>>
AllClient
CodeClient
<<utilise>>
<<Composant Métier Entité>>
Client
<<Interface>>
Service Client
MiseàJourl(A)
<<réalise>>
<<Maître>>
Client
CodeClient : int
MiseàJourl(A)
<<utilise>>
<<Composant Métier Entité>>
Ressource
<<Interface>>
Service Ressource
MiseàJourl(A)
<<réalise>>
<<Maître>>
Ressource
CodeRessource : int
MiseàJourl(A)

Annexe B. Code ATL des transformations

Cette annexe présente l'intégralité du code ATL des transformations utilisées dans RCMP.

```
-- Module Template
-- nom de la transformation modulaire (M2M)
module lastRule;

-- Les méta-modèles E/S des modèles Source/Target.
create OUT: CMPS from IN:CMPE;

-------------------------------------------------------------------------------------------------------------------------
                                                Helpers
-------------------------------------------------------------------------------------------------------------------------
-- Attributs des variantes de la solution
helper def: selectionVariant:
        Set(CMPE!VariantAction) = CMPE!VariantAction.allInstances()->collect(e|e.name);

-- Attributs de l'expérience 1
helper def: selection1:
Set(String) = Set{'CreationDemandeAlloc','TraitementSansReservation',
'MAJRessource','CreationDemandeAllocation'};

-- Attributs de l'expérience 2
helper def: selection2:
Set(String) = Set{'CreationDemandeAlloc','CreationDemandeAllocation',
'TraitementAvecReservation','VerificationReservation','MAJRessource'};

--helper isSelected d'une variantes
--Retourne vrai si la variation est sélectionnée pour l'expérimentation
helper context CMPE!Action
def:isSelected():
        Boolean = thisModule.selection1.includes(self.name);

--helper getselected de la variante d'une variation
--Retourne la première variante sélectionnée de la variation
helper context CMPE!VariationAction
def: getSelectedfirst() : CMPE!VariantAction = self.V->select(i|i.isSelected())->collect(e|e.name)->first();

--helper getselected de la variante d'une variation
--Retourne la deuxième variante sélectionnée de la variation
helper context CMPE!VariationAction
def: getSelectedlast() : CMPE!VariantAction = self.V->select(i|i.isSelected())->collect(e|e.name)->last();

--helper isnotPVV
-- Retourne vrai si l'action en cours est fixe
helper context CMPE!Action
def : isnotPVV():
        Boolean = if (not self.oclIsTypeOf(CMPE!VariationAction) and not
self.oclIsTypeOf(CMPE!VariantAction))
        then true
          else false
        endif;

--helper singleVariant
-- Retourne vrai si la variation choisie possède une seule variante sélectionnée
helper context CMPE!VariationAction
```

```
def:singleVariant():
        Boolean = self.V->select (e|e.isSelected())->size()=1;

--helper doubleVariant
-- Retourne vrai si la variation choisie possède plus d'une seule variante sélectionnée
helper context CMPE!VariationAction
def:doubleVariant():
        Boolean = self.V->select (e|e.isSelected())->size()>1;

--helper isMatchedN
-- copie les controlFlow si la source est : nœud initial, variante sélectionnée, action fixe et si la
destination est un nœud final
helper context CMPE!ControlFlow
def:isMatchedN():
                -- pour le nœud initial
        Boolean = if (self.Source.oclIsTypeOf(CMPE!InitialNode))
                  then true
                -- pour le nœud final ayant une variante sélectionnée
                        else if (self.Target.oclIsTypeOf(CMPE!FinalNode)
                                                and self.Source.isSelected())
                                then true
                -- pour les variantes sélectionnées
                                else if  (self.Source.oclIsTypeOf(CMPE!VariantAction)
                                                and self.Source.isSelected())
                                        then true
                -- pour les actions fixes
                                                else if (self.Source.oclIsTypeOf(CMPE!Action)
                                                   and self.Source.isnotPVV())
                                                        then true
                                                        else false
                                                        endif
                                                endif
                                        endif
                                endif;

--helper isMatchedPV
-- copie de controlFlow si la source est une variation ayant deux variantes sélectionnées sinon
isMatchedN
helper context CMPE!ControlFlow
def:isMatchedPV():
        Boolean = if (self.Source.oclIsTypeOf(CMPE!VariationAction))
                  then if (self.Source.doubleVariant()
                                                and self.Target.isSelected())
                          then true
                          else false
                          endif
                  else self.isMatchedN()
                  endif;

-- helper isMatcheObject
-- copie les ObjectFlow si la source est : ObjetcNode, variante sélectionnée ou action fixe et si la
destination est un nœud final
helper context CMPE!ObjectFlow
def:isMatchedObject():
                        Boolean =
                 -- pour les objectNode
                if (self.Source.oclIsTypeOf(CMPE!ObjectNode))
```

```
                then false
                  -- pour les actions fixes
                else if (not self.Source.oclIsTypeOf(CMPE!VariationAction)
            and not self.Source.oclIsTypeOf(CMPE!VariantAction))
                    then true
                  --pour les variantes sélectionnées
                    else if (self.Source.oclIsTypeOf(CMPE!VariantAction)
                        and self.Source.isSelected())
                then true
                  -- pour les points de variations
                            else if
              (self.Source.oclIsTypeOf(CMPE!VariationAction))
                                    then false
                                    else false
                                    endif
                            endif
                          endif
                endif;

-- helper isMatche
-- copie les Edges selon la nature
helper context CMPE!Edge
def:isMatched():
        Boolean = if (self.oclIsTypeOf(CMPE!ControlFlow))
                                   then self.isMatchedPV()
                          else if (self.oclIsTypeOf(CMPE!ObjectFlow))
                                        then self.isMatchedObject()
                                    else false
                                    endif
                          endif;

------------------------------------------------------------------------------------------------------------------------------------------
                                                        Règles
------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------Règles d'appel----------------------------------------------------------

lazy rule singleVar
{
        from u:CMPE!VariationAction (u.singleVariant())
        to a:CMPS!Action
        (
                name<-u.getSelectedfirst(),
                inPartition<-u.inPartition
        )
}

-- Recopie des variantes sélectionnées d'une variation (first and last)
-- Création du nœud de décision
lazy rule doubleVar
{
        from u:CMPE!VariationAction (u.doubleVariant())
        -- Action1
        to uc:CMPS!Action
        (
                name<-u.getSelectedfirst(),
                inPartition<-u.inPartition
```

```
		),
		--Action2
		c:CMPS!Action
		(
			name<-u.getSelectedlast(),
			inPartition<-u.inPartition
		),
		--Point de decision
		n:CMPS!DecisionNode
		(
			name<-u.name,
			inPartition<-u.inPartition
		)
}

-- copie des controlFlows.
lazy rule copieMatchedControlFlow {
		from u:CMPE!ControlFlow
		to uc:CMPS!ControlFlow
		(
		name<-u.name,
		Source<-u.Source,
		Target<-u.Target
		)
}

-- copie des ObjetFlows.
lazy rule copieMatchedObjectFlow {
		from u:CMPE!ObjectFlow
		to uc:CMPS!ObjectFlow
		(
		name<-u.name,
		Source<-u.Source,
		Target<-u.Target
		)
}

------------------------------------------------------------ Règles de transformations----------------------------------------------
-- Recopie des partitions de sortie
rule copie_partition
{
		from In_AP:CMPE!ActivityPartition
		to Out_AP:CMPS!ActivityPartition
		(
			name<-In_AP.name
		)
}
-- Recopie des actions fixes
rule copie_Actionsfixes
{
		from u:CMPE!Action (u.isnotPVV())
		to
		debut:CMPS!Action
		(
			name<-u.name,
			inPartition <- u.inPartition
		)
```

```
}
--Recopie du nœud initial de la solution
rule copie_Noeud_Initial
{
	from u:CMPE!InitialNode
	to
	debut:CMPS!InitialNode
	(
		name<-u.name,
		inPartition <- u.inPartition
	)
}

-- Recopie du nœud initial de la solution
rule copie_Noeud_Final
{
	from u:CMPE!FinalNode
	to
	fin:CMPS!FinalNode
	(
		name<-u.name,
		inPartition <- u.inPartition
	)
}

--Recopie des variantes sélectionnées des variations
rule copie_variantes_PV_selected
{
	from u:CMPE!VariationAction

	do {
		if (u.singleVariant())
			thisModule.singleVar(u);
		if (u.doubleVariant())
			thisModule.doubleVar(u);
	}
}

--Recopie des controlFlows de l'expérimentation
rule copied_ControlFlow
{
	from u:CMPE!ControlFlow  (u.isMatchedPV() )
	do
	   {
	     thisModule.copieMatchedControlFlow(u);
	   }
}
--Recopie des ObjectFlow de l'expérimentation
rule copied_ObjectFlow
{
	from u:CMPE!ObjectFlow  (u.isMatchedObject() )
	do
	 {
	  thisModule.copieMatchedObjectFlow(u);
	 }
}
```

Annexe C. Questionnaires des expérimentations

1. Pratiques des utilisateurs

L'objectif de ce questionnaire est de connaître les habitudes des sujets en matière de spécification et de réutilisation de composants métier.

Pratiques

Prénom : Date :

1. Vous utilisez les processus métier ou les diagrammes d'activités UML ...

|__| 1. Très souvent

|__| 2. Souvent

|__| 3. Parfois

|__| 4. Jamais

2. De manière générale, comment spécifiez-vous un processus métier (diagramme d'activités d'un processus) ?

3. Est-ce que vous avez l'habitude de réutiliser des processus métier (diagramme d'activités) que vous avez conçus ?

|__| 1. Très souvent

|__| 2. Souvent

|__| 3. Parfois

|__| 4. Jamais

4. Comment faites-vous pour les réutiliser ?

5. Dans quelles circonstances, réutilisez-vous des processus métier (diagramme d'activités) ?

6. Est-ce que la réutilisation est liée à certaines conditions ? si oui, lesquelles ?

7. Au niveau de la description du modèle réutilisable, quels sont les éléments de base dont vous avez besoin pour pouvoir le réutiliser ?

8. Pour accéder à un modèle réutilisable, quels sont les éléments de base dont vous avez besoin ?

Année de Naissance : Vous êtes : ❒ Une femme ❒ Un homme

2. Concepteurs / Ingénieurs de CMP

L'objectif de ce questionnaire est d'évaluer et de mesurer le comportement des sujets quant au guidage du processus de spécification d'un CMP.

Concepteurs

Prénom : Date :

1. Globalement, qu'avez vous pensé de la réutilisation des processus métier ?

2. Lors de cet exercice, quelles sont les difficultés que vous avez rencontrées ?

3. Pour vous, quels sont les manques de ce processus ?

4. Quelles sont les étapes que vous aimeriez ajouter à ce processus ?

5. Quelles sont les étapes qui pourraient être supprimées ?

6. Que pensez-vous du guidage du processus ?

7. En ce qui concerne, la représentation de la variabilité par rapport à l'abstraction de Processus métier similaires mais de domaines différents, Est-ce que c'est une aide ?

|__| 1. Très importante
|__| 2. Plutôt importante
|__| 3. Plutôt pas importante
|__| 4. Pas du tout importante
|__| 5. Sans opinions

8. La représentation de plusieurs types de variabilité (alternative/option/ensemble d'alternatives…), est-elle ...

|__| 1. Tout à fait satisfaisante
|__| 2. Plutôt satisfaisante
|__| 3. Plutôt pas satisfaisante
|__| 4. Pas du tout satisfaisante
|__| 5. Sans opinion

9. En quoi est-elle satisfaisante ou insatisfaisante ?

10. En ce qui concerne, les formes de la variabilité (des activités, du degré d'informatisation, du rôle), est-ce que c'est une aide ?

|__| 1. Très importante
|__| 2. Plutôt importante
|__| 3. Plutôt pas importante
|__| 4. Pas du tout importante
|__| 5. Sans opinions

11. Quelles sont les formes que vous aimeriez ajouter à cette variabilité ?

12. En ce qui concerne la représentation des dépendances entre les parties variables (exige/exclut), vous diriez qu'elle est ...

|__| 1. Tout à fait complète
|__| 2. Plutôt complète
|__| 3. Plutôt pas complète
|__| 4. Pas du tout complète
|__| 5. Sans opinion

13. Si vous la jugez incomplète que manque-t-il ?

14. Toujours pour la représentation des dépendances entre les parties variables, vous diriez qu'elle est ...

|__| 1. Tout à fait claire
|__| 2. Plutôt claire
|__| 3. Plutôt pas claire
|__| 4. Pas du tout claire
|__| 5. Sans opinion

15. Les fiches descriptives d'un processus métier qui vous ont été distribuées ont été ...

|__| 1. Très utiles
|__| 2. Plutôt utiles

|__| 3. Plutôt pas utiles
|__| 4. Pas du tout utiles
|__| 5. sans opinion

16. Pour vous quel est l'élément le plus important de la fiche ? (Vous pouvez cocher plusieurs réponses)

|__| 1. Modèle métier (diagramme de cas d'utilisation métier)
|__| 2. Scénario
|__| 3. Modèle organisationnel (diagramme d'activités)
|__| 4. Sans opinion

Vous pouvez cocher plusieurs cases (3 au maximum).

17. Les étapes d'abstraction sont-elles ?

|__| 1. Tout à fait complètes
|__| 2. Plutôt complètes
|__| 3. Plutôt pas complètes
|__| 4. Pas du tout complètes
|__| 5. Sans opinion

18. La spécification obtenue est ...

|__| 1. Tout à fait lisible
|__| 2. Plutôt lisible
|__| 3. Plutôt pas lisible
|__| 4. Pas du tout lisible
|__| 5. sans opinion

19. L'introduction de la variabilité est ...

|__| 1. Tout à fait satisfaisante
|__| 2. Plutôt satisfaisante
|__| 3. Plutôt pas satisfaisante
|__| 4. Pas du tout satisfaisante
|__| 5. sans opinion

20. La spécification obtenue est ...

|__| 1. Très générique
|__| 2. Plutôt générique
|__| 3. Plutôt pas générique
|__| 4. Pas du tout générique
|__| 5. Sans opinion

21. La spécification obtenue est...

|__| 1. Tout à fait complète
|__| 2. Plutôt complète
|__| 3. Plutôt pas complète
|__| 4. Pas du tout complète
|__| 5. Sans opinion

22. Est-ce que vous pensez que cette méthode nécessite une phase d'apprentissage ?

|__| 1. Oui, tout à fait
|__| 2. Oui, Plutôt
|__| 3. Non, plutôt pas
|__| 4. Non, pas du tout

23. Vous pensez que la phase d'appropriation de cette méthode demande ...

|__| 1. quelques minutes
|__| 2. quelques heures
|__| 3. quelques jours
|__| 4. quelques semaines

|__| 5. plus longtemps

24. Conseilleriez-vous l'utilisation de cette méthode ?

|__| 1. Oui, tout à fait
|__| 2. Oui, Plutôt
|__| 3. Non, plutôt pas
|__| 4. Non, pas du tout

25. Pourquoi ?

26. Est-ce que vous auriez eu besoin de conseil en cours d'utilisation ?

|__| 1. Oui, tout à fait
|__| 2. Oui, Plutôt
|__| 3. Non, plutôt pas
|__| 4. Non, pas du tout

27. pourquoi ?

3. Utilisateurs / Ingénieurs de SI

L'objectif de ce questionnaire est d'évaluer et de mesurer le comportement des sujets quant à l'usage du modèle de CMP pour la conception d'un SI.

Utilisateurs

Prénom : Date :

1. Globalement, qu'avez vous pensé de la réutilisation des composants métier ?

2. Lors de cet exercice, quelles sont les difficultés que vous avez rencontrées ?

3. Les règles de réduction sont ...

|__| 1. Tout à fait claires
|__| 2. Plutôt claires
|__| 3. Plutôt pas claires
|__| 4. Pas du tout claires
|__| 5. Sans opinion

4. Les règles de réduction sont ...

|__| 1. Tout à fait complètes
|__| 2. Plutôt complètes
|__| 3. Plutôt pas complètes

|__| 4. Pas du tout complètes
|__| 5. Sans opinion

5. Si elles ne sont pas complètes que manque-t-il ?

6. Pour pouvoir choisir les variantes, la vue métier est ...
|__| 1. Tout à fait satisfaisante
|__| 2. Plutôt satisfaisante
|__| 3. Plutôt pas satisfaisante
|__| 4. Pas du tout satisfaisante
|__| 5. sans opinion

7. Pour réutiliser un composant, la vue métier est ...
|__| 1. Tout à fait satisfaisante
|__| 2. Plutôt satisfaisante
|__| 3. Plutôt pas satisfaisante
|__| 4. Pas du tout satisfaisante
|__| 5. sans opinion

8. Pour accéder aux vues informatisées du CMP, la vue métier est ...
|__| 1. Tout à fait satisfaisante
|__| 2. Plutôt satisfaisante
|__| 3. Plutôt pas satisfaisante
|__| 4. Pas du tout satisfaisante
|__| 5. sans opinion

9. La spécification réduite obtenue est ...
|__| 1. Tout à fait satisfaisante
|__| 2. Plutôt satisfaisante
|__| 3. Plutôt pas satisfaisante
|__| 4. Pas du tout satisfaisante
|__| 5. sans opinion

10. La spécification obtenue est ...
|__| 1. Tout à fait complète
|__| 2. Plutôt complète
|__| 3. Plutôt pas complète
|__| 4. Pas du tout complète
|__| 5. Sans opinion

11. À votre avis, quels sont les avantages d'un modèle supportant la variabilité ?

12. Pour réutiliser un composant, vous pensez que ce sera ...
|__| 1. Très facile
|__| 2. Plutôt facile
|__| 3. Plutôt pas facile
|__| 4. Pas facile du tout
|__| 5. Sans opinion

13. La mise en évidence de la variabilité peut éviter des erreurs de conception, vous êtes ...
|__| 1. Tout à fait d'accord
|__| 2. Plutôt d'accord
|__| 3. Plutôt pas d'accord

|__| 4. Pas du tout d'accord
|__| 5. Sans opinion

14. Le fait que cette solution est orientée processus peut favoriser la réutilisation, Vous êtes...

|__| 1. Tout à fait d'accord
|__| 2. Plutôt d'accord
|__| 3. Plutôt pas d'accord
|__| 4. Pas du tout d'accord
|__| 5. Sans opinion

15. Le modèle à réutiliser est ?

|__| 1. Tout à fait flexible
|__| 2. Plutôt flexible
|__| 3. Plutôt pas flexible
|__| 4. Pas du tout flexible
|__| 5. Sans opinion

16. Est-ce que vous pensez que cette méthode nécessite une phase d'apprentissage ?

|__| 1. Oui, tout à fait
|__| 2. Oui, Plutôt
|__| 3. Non, plutôt pas
|__| 4. Non, pas du tout

17. Vous pensez que la phase d'appropriation de cette méthode demande ...

|__| 1. quelques minutes
|__| 2. quelques heures
|__| 3. quelques jours
|__| 4. quelques semaines
|__| 5. plus longtemps

18. Conseilleriez-vous l'utilisation de cette méthode ?

|__| 1. Oui, tout à fait
|__| 2. Oui, Plutôt
|__| 3. Non, plutôt pas
|__| 4. Non, pas du tout

19. Dans quelles circonstances ?

Bibliographie

A	
[Ambler, 1998]	Ambler S.W., “Process Patterns: building Large Scale Systems using Object Technology”, SIGS Books, Cambridge University Press, Décembre, 1998.
[André, 1994]	André J., “Moving From Merise to Schlaer and Mellor”, SIG-Publication vol. 3, 1994.
[Andro *et al.*, 1998]	Andro T., Chauvet J.-M., « Objets métier », Eyrolles, Paris, 232 pp., 1998.
[Arnaud, 2008]	Arnaud N., « Fiabiliser la réutilisation des patrons par une approche orientée complétude, variabilité et généricité des spécifications», Thèse de doctorat, Université Joseph Fourrier - Grenoble 1, Octobre, 2008.
[ATLAS, 2009]	The Atlas Transformation Language (ATL), http://modelware.inria.fr/ rubrique12.html, dernière consultation Juin 2009.
B	
[Bachmann *et al.,* 2001]	Bachmann F., Bass L., “Managing variability in software architecture”, ACM SIGSOFT Software Engineering Notes, Volume 26, n°3, 2001.
[Bachmann *et al.*, 2003]	Bachmann F., Goedicke M., Leite J., Nord R., Pohl K., Ramesh B. and Vilbig A., “A Meta-model for Representing Variability in Product Family Development”, 5th International Workshop, PFE 2003, Siena, Italy, Novembre 4-6, 2003.
[Bashroush *et al.*, 2008]	Bashroush R., Spence I., Kilpatrick P., Brown TJ., Gillan C., “A Multiple Views Model for Variability Management in Software Product Lines”, Second international workshop on variability Modelling of Software Intensive System (VaMoS’08), Essen, Germany, Janvier, 2008.
[Barbier *et al.*, 2002]	Barbier F., Atkinson C., "Business Components", Business Component-Based Software Engineering, Kluwer, vol. 705, Chap. 1, pp. 1-26, 2002.
[Bass *et al.*, 2000]	Bass L., Buhman C., Comella-Dorda S., Long F., Robert J., Seacord R., Wallnau K., “Volume I: Market Assessment of Component-Based Software Engineering”, Carnegie Mellon University,

	Software Engineering Institute, TECHNICAL REPORT CMU/SEI-2000-TR-008, ESC-TR-2000-007, Mai, 2000.
[Blevins, 2001]	Blevins D., "Overview of the Enterprise JavaBeans Component Model", in Component-Based Software Engineering, Addison-Wesley, 2001.
[Bosch, 2007]	Bosch J., "Invited Talk: Expanding Software Product Families: From Integration to Composition", Architecture of Computing Systems - ARCS 2007, 20th International Conference, Zurich, Switzerland, Mars 12-15, 2007.
[Bosch *et al.*, 2001]	Bosch J., Florijn G., Greefhorst D., Kuusela J., Obbink H., and Pohl K., "Variability Issues in Software Product Lines", Fourth workshop Product Familly Engineering (PFE4), pages 11–19, 2001.
[Boehm, 1988]	Boehm B.W., "A Spiral Model of Software Development and Enhancement", IEEE Computer, 21(5), 61-73, Mai 1988.
[Bohrer, 1998]	Bohrer K.A., "Architecture of the SanFransisco Framework", IBM Systems Journal, Vol 37, N° 2, 1998.
[Bûhne *et al.*, 2005]	Bühne S., Lauenroth K., Pohl K., "Modelling Requirements Variability across Product Lines", Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05), Paris, France, 2005.
C	
[Casanave, 1996]	Casanave C., "Business Object Architectures and standards", Data Access Corporation, Miami, USA, 1996.
[Castano *et al.*, 1994]	Castano S., De Antonellis V., "The F^3 Reuse Environment for Requirements Engineering", ACM SIGSOFT Software Engineering Notes, 19 (3), pp. 62-65, 1994.
[Cauvet *et al.*, 2005]	Cauvet C., Ramadour P., « Les composants métier dans l'ingénierie des systèmes d'information », Vuibert (Ed.), « Composants : concepts, techniques et outils », 2005.
[Cauvet *et al.*, 2001]	Cauvet C., Rieu D., Front-Conte A. et Ramadour P., « Réutilisation dans l'ingénierie des SI », Chapitre 5 du livre Ingénierie des SI, Editions Hermès, pp. 115-147, 2001.
[Cauvet *et al.*, 1999]	Cauvet C., Semmak F., « La réutilisation dans l'ingénierie des systèmes d'information », « Génie objet - Analyse et conception de l'évolution », Hermès, pp. 25-55, 1999.
[Cherbakov *et al.*, 2005]	Cherbakov L., Galambos G., Harishankar R., Kalyana S., Rackham G., "Impact of service orientation at the business level", IBM Systems Journal 44(4): 653-668, 2005.

[Clauss, 2001]	Clauss M., “Generic modeling using UML extensions for variability”, Workshop on Domain Specific Visual Languages, pages 11-18, 2001.
[Conte *et al.*, 2001]	Conte A., Fredj M., Giraudin J-P., Rieu D., « P-Sigma : un formalisme pour une représentation unifiée de patrons », Inforsid’01, Martigny, Juin, 2001.
[Cummins, 1999]	Cummins F., “OMG Business Object Concept – BOTF –”, White paper – EDS- BOM/99-12-42, 1999.
D	
[David *et al.*, 2006]	David P.C., Ledoux T., « Une approche par aspects pour le développement de composants Fractal adaptatifs », L’Objet, vol. 12, no 2-3 (210 p.), 2006.
[D'Souza *et al.*, 1998]	D’Souza D., Wills A., “Objects, Components and Frameworks with UML: The Catalysis Approach”, Addison Wesley, Reading, MA, 1998.
[Dumas *et al.*, 2008]	Dumas M., Reichert M., Shan M. (Eds.), “Business Process Management”, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008.
E	
[Eclipse, 2009]	www.eclipse.org, dernière consultation Juin 2009.
[Ewald, 2001]	Ewald T., “Overview of COM+, in Component-Based Software Engineering: Putting the Pieces Together”, Addison-Wesley, 2001.
F	
[Favre *et al.*, 2006]	Favre J. M., Estublier J., Blayfornarino M., « L'ingénierie dirigée par les modèles, au-delà du MDA », Lavoisier, 2006.
[Finkelstein, 1988]	Finkelstein A., “Re-use of formatted requirements specifications”, Software Eng. J., pp. 186-197, Septembre 1988.
[Frasincar *et al.*, 2002]	Frasincar F., Houben G.J., “Hypermedia Presentation Adaptation on the Semantic Web”, Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems Second International Conference, (AH’02), vol. 2347, pp. 133-142, 2002.
G	
[Griss, 2001]	Griss M.L., “Product Line Architectures, in Component-Based Software Engineering”, G.T. Heineman and W.L. Councill, Editors, Addison-Wesley, pp. 405-419, 2001.
[Grosz *et al.*, 96]	Grosz G., Si-Said S. et Rolland C., « MENTOR : un environnement pour l'ingénierie des méthodes et des besoins », INFORSID,

	Bordeaux, Juin, 1996.
[Guzélian *et al.*, 2005]	Guzelian G., Cauvet C., « Modèles Sémantiques de Composants pour l'Ingénierie des Systèmes d'Information », 16es journées francophones d'ingénierie des connaissances, IC 2005, Plate-forme AFIA, pp. 145-156, Nice, 1-3 Juin, 2005.
H	
[Hachani, 2009]	Hachani S., "Prise en compte de la personnalisation dans les méthodes de conception de SI", Master 2 Recherche, Université Joseph Fourier –Grenoble, 2009.
[Harsu, 2002]	Harsu M., "A survey on domain engineering", Institute of Software Systems, Tampere University of Technology, Décembre, 2002.
[Hassine, 2005]	Hassine I., « Spécification et formalisation des démarches de développement a base de composants métier : la démarche Symphony", Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, 2005.
[Heineman *et al.*, 2001]	Heineman G., Councill W., "Component-Based Software Engineering: Putting the Pieces Together", Addison-Wesley, 2001.
[Hernandez, 2005]	Hernandez N., « Ontologie de domaine pour la modélisation du contexte en recherche d'information », Thèse de doctorat, Université Paul Sabatier de Toulouse, Décembre, 2005.
[Herzum *et al.*, 2000]	Herzum P., Sims O., "Business Component Factory: a Comprehensive Overview of Component-Based Development for the Enterprise", Wiley Computer Publishing, 1999.
I	
[IBM, 2009]	Site Internet IBM Rational, logiciel d'évaluation RUP, www-01.ibm.com/software/awdtools/rup/, dernière consultation janvier 2009.
[IBM, 2006]	http://www.ibm.com/developerworks/library/specification/ws-sca/, 2006.
[INRIA, 2005]	Inria model transformation language, http ://modelware.inria.fr, 2005.
[INRIA, 2009]	OpenEmbeDD home: Model Driven Engineering open-source platform for Real-Time & Embedded systems, http://openembedd.inria.fr/ home_html, 2009.
J	

[Jacob *et al.*, 200]	Jacob I., Krivine J.P., Monclar F.R., « De LISA à ELICO, des langages pour le développement de systèmes d'assistance à l'opérateur, « Ingénierie des connaissances, évolution récentes et nouveaux défis », Charlet J., Zacklad M., Kassel G. and Bourigault D. coordinators, Eyrolles, 2000.
[Jacobson *et al.*, 1997]	Jacobson I., Griss M., and Jonsson P., "Software Reuse: Architecture Process and Organization for business Success", ACM Press New York, 1997.
[Jacobson *et al.*, 1999]	Jacobson I., Booch G., Rumbaugh J., "The Unified Software Development Process", Addison-Wesley Object Technology Series, 1999.
[Jouault *et al.*, 2006]	Jouault, F, Bézivin, J. « KM3: a DSL for Metamodel Specification ». In proc. of 8th FMOODS, LNCS 4037, Bologna, Italy, pp 171-185.Kolski C., Interfaces hommemachine, Paris, Hermès, 2006.
K	
[Kang *et al.*, 1990]	Kang K., Cohen S., Hess J., Novak W., Peterson S., "Feature-Oriented Domain Analysis (FODA) feasibility study", Technical report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.
[Kang *et al.*, 1998]	Kang K.C., Kim S., Lee J., Kim K., Shin E., Huh M., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures", Annals of Software Engineering 5, 143-168, 1998.
[Kappel *et al.*, 2000]	Kappel G., Retschitzegger W. and Schwinger W. "Modelling Customizable Web Applications - A Requirement's Perspective". In Proceedings of the International Conference on Digital Libraries: Research and Practice, Koyoto, 2000.
[Kephart, 2002]	Kephart J., "A Vision of Autonomic Computing", ACM, Seattle, WA, USA, p. 13-36, Novembre, 2002.
[Khayati, 2003]	Khayati O., « Modèles formels et outils génériques pour la gestion et la recherche de composants", Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, 2005.
[Klement *et al.*, 2000]	Klement J.F., Turowski K., "Classification Framework for business components", 33rd Hawaii International conference on system sciences, USA, Hawaii, 2000.
L	
[Larvet, 1994]	Larvet P., « Analyse des systèmes : de l'approche fonctionnelle à l'approche objet », InterEditions, 1994.

[Lung *et al.*, 1995] Lung C., Urban J.E., “An Approach to the Classification of Domain Models in Support of Analogical Reuse”, SSR ’95, Seattle, WA, USA, 1995.

M

[Maiden *et al.*, 1992] Maiden N. A. M., Sutcliffe A. G., “Exploiting reusable specifications through analogy”, Commun of the ACM, vol. 35, no. 4, pp. 55-64, Avril, 1992.

[Maccari *et al.*, 2000] Maccari A. and Tuvinen A.P., “System family architectures : Current challenges at nokia”, F. van der Linden, editor, IW-SAPF, volume LNCS, pages 107–115. Springer, 2000.

[Maccari *et al.*, 2003] Maccari A., and Heie A., “Managing infinite variability”, Jilles van Gurp and Jan Bosch, editors, Software Variability Management Workshop, number IWI preprint 2003-7-01, pages 28–34. Research institute of Mathematics and Computing Science of the university of Groningen, 2003.

[Marvelig, 2009] https://marvelig.imag.fr/

[Mineau *et al.*, 1995] Mineau G.W., Godin R., “Automatic structuring of knowledge bases by conceptual clustering”, IEEE Transactions (Data and Knowledge Engineering, 7 (5), pp.824-829, 1995.

[Molina *et al.*, 2000] Molina J.G., José M., Begoña M., Joaquín N., Ambrosio T., “Towards Use Case and Conceptual Models through Business Modeling”, Proceedings of (ER2000) 19th International Conference on Conceptual Modeling Salt Lake City, Utah, USA, October 9–12, 2000.

[Morley *et al.*, 2004] Morley C., Hugues J., Leblanc B., « Processus métiers et systèmes d'information : Evaluation, modélisation, mise en œuvre », Dunod, 2004.

[Morley *et al.*, 2000] Morley C., Hugues J., Leblanc B., « UML pour l’analyse d’un système d’information : le cahier des charges du maître d’ouvrage », Dunod, 2000.

[Muller, 1998] Muller P.A., « Modélisation objet avec UML », Eyrolles, Décembre, 1998.

[Mineau *et al.*, 1995] Mineau G.W., Godin R., “Automatic structuring of knowledge bases by conceptual clustering”, IEEE Transactions (Data and Knowledge Engeering, 7 (5), pp.824-829, 1995.

N

[Neighbors, 1984] Neighbors J. M., “The Draco approach to constructing software from reusable components”, lEEE Trans. on Soflware Engineering,

vol. 10, no. 5, pp. 564-574, Septembre, 1984.

O

[Occello *et al.*, 2006] Occello A., Pinna-Déry A-M, Riveill M., “Capitalizing Adaptation Safety: a Service oriented Approach”, Third International Workshop on Coordination and Adaptation Techniques for Software Entities (in conjunction with ECOOP'06) (WCAT06), pp.71-79, Nantes, France, Juillet, 2006.

[Oliveira *et al.*, 2005] Oliveira E.A., Gimenes I., Huzita E., Maldonado J.C., “A Variability Management Process for Software Product Lines”, In Proc. of CASCON 2005, Toronto, Canada, 2005.

[OMG, 1998] Herzum P. and Sims O., “The Business Component Approach”, OOPSLA'98 Business Object Workshop, Octobre, Canada, 1998.

[OMG, 1997] Object constraint language specification, version 1.1, OMG document ad970808, 1997.

[OMG, 2007] OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, http://www.omg.org/spec/UML/2.1.2/.

[OMG, 1999] OMG, UML 1.3, ftp://ftp.omg.org/pub/docs/ad/99-06-08.pdf.

[Oussalah et *al.*, 1999] Oussalah M.C., Génie Objet- Analyse et conception de l’évolution, Hermès, Septembre, 1999.

P

[Pohl *et al.*, 2005] Pohl K., Bockle G., Linden F.V.D., “Software Product Line Engineering: Foundations, Principles, and Techniques”, Springer-Verlag Berlin Heidelberg, 2005.

[Pujalte *et al.*, 2004] Pujalte V., Ramadour P., Cauvet C., « Recherche de composants réutilisables : une approche centrée sur l’assistance à l’utilisateur », XXIIème Congrès INFORSID, Biarritz, Mai 2004.

R

[Ramadour, 2001] Ramadour P., « Modèles et langage pour la conception et la manipulation de composants réutilisables de domaine », Thèse de Doctorat, Université d’Aix-Marseille III, 2001.

[Razavian *et al.*, 2008] Razavian M., Khosravi R., “Modeling Variability in Business Process Models Using UML”, Fifth International Conference on Information Technology: New Generations, Las Vegas, Nevada April 7-9, 2008.

[Redolfi *et al.*, 2005] Redolfi G., Spagnoli L., Hemesath P., Melo Bastos R., Blois M., Cristal R.M., Espindola A., “A Reference Model for Reusable

Components Description", Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

[Rieu *et al.*, 1999] Rieu D., Giraudin J.P., Saint-Marcel C., Front-Conte A., « Des opérations et des relations pour les patrons de conception », Congrès INFORSID'99, Toulon, Juin 1999.

[Rocques *et al.*, 2004] Roques P., Vallée F., « UML 2 en action : De l'analyse des besoins à la conception J2EE », Eyrolles Editions, 2004.

[Rolland *et al.*, 1988] Rolland C., Foucaut O., Benci G., « Conception des systèmes d'information : la méthode REMORA », Eyrolles, 1988.

[Rouillard *et al.*, 2007] Rouillard J., Vantroys T., Chevrin V., « Architectures orientées services », Vuibert, 2007.

S

[Saidi *et al.*, 2009a] Saidi R., Fredj M., Front A., Mouline S., « Variabilité dans les composants métiers multivues », Revue RTSI-ISI : Revue des Sciences et Technologies de l'Information - Ingénierie des Systèmes d'Information, éditeur Hermès-Lavoisier, numéro 14/2009**,** PP. 61-86.

[Saidi *et al.*, 2009b] Saidi R., Front A., Rieu D., Fredj M., Mouline S., "Component-Based Development : Extension with Business Component Reuse", The third IEEE International Conference on Research Challenges in Information Science (RCIS), 22- 24 Avril, Fès, Maroc, 2009.

[Saidi *et al.*, 2008a] Saidi R., Fredj M., Mouline S., Front A., Rieu D., « Spécification de composants métier : une approche par expression de variabilité multi-vue », XXVIème Congrès INFORSID, 27-30 Mai, Fontainebleau, France, 2008.

[Saidi *et al.*, 2008b] Saidi R., Front A., Rieu D., Fredj M., Mouline S., "From a Business Component to a Functional Component using a Multi-View Variability Modelling", MoDISE-EUS'2008 workshop in conjunction with The 20th International Conference on Advanced Information Systems Engineering (CaiSE'08), 16-17 Juin, Montpellier, France, 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online CEUR-WS.org/Vol-341/.

[Saidi *et al.*, 2008c] Saidi R., Fredj M., Mouline S., Front A., Rieu D., "Variability Modelling for Business Component Customization", The IEEE Symposium on Computers and Communications (ISCC'08), 6 - 9 Juillet, Marrakech, Maroc, 2008.

[Saidi *et al.*, 2007a] Saidi R., Arnaud N., Rieu D., Fredj M., "Multi-View Variability Modelling for Business Component Reuse", The Second IEEE International Conference on Digital Information Management

	(ICDIM), 31 Octobre, Lyon, France, 2007.
[Saidi *et al.*, 2007b]	Saidi R., Fredj M., Mouline S., Front A., Rieu D., « Towards Managing Variability Across Business Component Development », (IRI), The 2007 IEEE International Conference on Information Reuse and Integration, IRI – 2007, 13-15 Août, Las Vegas, USA, 2007.
[Select, 2009]	Site internet Select perspective, www.selectbs.com/solutions/ component-based-development.htm, dernière consultation janvier 2009.
[SEI, 2007]	Software Engineering Institute, Carnegie-Mellon University, "Domain", http://www.sei.cmu.edu/legacy/scm/tech_rep/ TR11_90/ 3.2.1_Domain. html, Janvier, 2007.
[Semmak, 1998]	Semmak F., « Réutilisation de composants de domaine dans la conception des systèmes d'information », Thèse de doctorat, Université Paris I, 1998.
[Schmid, 1999]	Schmid H.A., "Business entity and process components", Springer (Editor) OOPSLA'99, Business Object Design and Implementation, Denver, USA, 2 Novembre, 1999.
[Sipka, 2005]	Sipka M., "Exploring the Commonality in Feature Modeling Notations", informatics and information technologies student Research Conference, IIT.SRC 2005, Slovenie , Avril, 2005.
[SmartQVT, 2007]	SmartQVT - A QVT implementation, http://smartqvt.elibel.tm.fr/, 2007.
[Snoeck *et al.*, 2000]	Snoeck M., Poels G., "Analogical Reuse of Structural and Behavioural Aspects of Event-Based Object-Oriented Domain Models", Domain Engineering Workshop, Proceedings of the 11th International Workshop on Database and Expert Systems Applications, pp. 802-806, London (Greenwich), 4-8 Sept. 2000.
[Sunyé, 1999]	Sunyé G., « Génération de code à l'aide de patrons de conception », Langages et Modèles à Objets - LMO'99, 1999.
[Szyperski, 1998]	Szyperski C., "Component Software: Beyond Object-Oriented Programming", 2. ed., Harlow, 1998.
T	
[Tastet, 2004]	Tastet L., « AGAP, un Atelier de Gestion et d'Application de Patrons », mémoire d'ingénieur CNAM en Informatique, Mars 2004.
V	
[VanDerMaβen *et al.*, 2002]	Van der Maβen T., Lichter H., "Modeling variability by UML use case diagrams", International Workshop on Requirements

	Engineering for Product Lines (REPL'02), pages 19-25. AVAYA labs, Septembre 2002.
[VanGurp *et al.*, 2000]	VanGurp J., Bosch J., and Svahnberg M., "Managing variability in software product lines", Landelijk Architectuur Congres, Amsterdam, 2000.
W	
[Wang *et al.*, 2001]	Wang N., Schmidt D. C., O'Ryan C., "Overview of the CORBA Component Model, in Component-Based Software Engineering – Putting the Pieces Together", Heineman G.T. and Council W. T. coordinators, Addison-Wesley, 2001.
[Wartik *et al.*, 1992]	Wartik S., Prieto-Diaz R., "Criteria for Comparing Reuse-Oriented Domain Analysis Approaches", International Journal of Software Engineering and Knowledge Engineering, 2(3), pp. 403-431, 1992.
[Wirfs-Brock *et al.*, 1990]	Wirfs-Brock R., Wilkerson B., Weiner L., "Designing Object-Oriented Software Prentice-Hall", Englewood Cliffs, New Jersey, 1990.
Z	
[Ziadi, 2004]	Ziadi T., « Manipulation de Lignes de Produits en UML », Thèse de doctorat, Université de Rennes 1, Décembre, 2004.

Résumé

Les Systèmes d'Information (SI) de même domaine d'activité gèrent de nombreux concepts similaires. Ces concepts peuvent être analysés et généralisés dans des abstractions informatiques qui seront réutilisées lors de nouveaux développements. De telles abstractions sont appelées Composants Métier (CM). Cependant, il est souvent difficile d'expliciter des critères clairs de réutilisation, en particulier la manière dont on peut identifier, spécifier, organiser et en grande partie automatiser la réutilisation de ces CM. Les contributions de cette thèse adressent cette problématique et s'articulent autour de trois principaux résultats.

La première contribution concerne un modèle de CM de nature processus appelé « CMP ». Ce modèle est centré sur les propriétés fonctionnelles des composants. L'accent est mis sur la complétude et la variabilité de la solution exprimée sous la forme de quatre vues complémentaires intégrant des points de variation. Le travail réalisé sur les mécanismes de spécification de la variabilité a abouti à un profil UML pouvant être utilisé pour modéliser non seulement des processus réutilisables mais aussi des processus flexibles. Une deuxième contribution s'inscrit dans le cadre de la proposition d'un processus permettant la spécification d'un CMP selon le modèle proposé. Dans ce processus, nous définissons un ensemble de règles de construction, de traduction et de cohérence qui assurent la traçabilité des artefacts produits tout au long du cycle de développement. Une troisième contribution concerne la proposition de directives qui assistent l'ingénieur de SI lors de la réutilisation de CMP. L'accent est particulièrement mis sur la proposition d'un formalisme de documentation et de classification de CMP validé dans le cadre d'un environnement de stockage de composants. Nous proposons également un processus d'imitation intégré à la méthode de développement Symphony et permettant de tirer partie de la spécification d'un CMP lors de la conception d'un SI.

L'ensemble des propositions est accompagné d'outils et d'expérimentations utilisateurs servant de supports de validation et de mise en œuvre des travaux réalisés.

Mots-clefs

Composant Métier Processus, Réutilisation, Variabilité, Complétude, UML, Système d'Information

---

Title

Design and usage of process business components for information systems

Abstract

Information Systems (IS) with the same business domain manage similar concepts. These concepts can be analyzed and generalized in abstractions reused in new developments. Such abstractions are called Business Components (BC). However, it is often difficult to express criteria of reuse, particularly how to identify, specify, organize and largely automate the reuse of the BC. The contributions of this thesis address this issue and focus on three main results.

The first contribution concerns a process BC model called "PBC". This model focuses on the functional properties of components. Emphasis is placed on the completeness and the variability of the solution expressed in the form of four complementary views incorporating variation points. Variability integration has led to an UML profile that can be used to model reusable and flexible processes. The second contribution proposes a specification process of a PBC according to the proposed model. In this process, we propose a set of rules for construction, translation and consistency to ensure the traceability of artifacts produced throughout the cycle of development. The third contribution is related to a set of guidelines to assist the IS engineer at the PBC reuse. Emphasis is placed on the proposal of the documentation and the classification of PBC validated in a component storage environment. We also propose an imitation process incorporated into the Symphony method to take advantage of the specification of a PBC in the design of an IS. All proposals are accompanied by tools and experiments used as supports of validation and implementation of the achieved works.

Keywords

Process Business Component, Reuse, Variability, Completeness, UML, Information System