Découpage en catégories

Objectifs du chapitre

Ce chapitre traite du démarrage de l'analyse objet du système à réaliser.

Nous verrons en particulier comment utiliser la notion de package pour définir des catégories de classes d'analyse et découper le modèle UML en blocs logiques les plus indépendants possibles.

Cette structuration logique sera ensuite affinée durant toute l'étape d'analyse, mais néanmoins restera pour le chef de projet un outil essentiel qui lui permettra d'organiser son processus de développement.

Quand intervient le découpage en catégories ?

Le découpage en catégories constitue la première activité de l'étape d'analyse (il s'affine bien sûr de manière itérative au cours du projet). Il se situe sur la branche gauche du cycle en Y et succède à la capture des besoins fonctionnels.

En fin d'analyse des besoins, nous obtenons un découpage fonctionnel exprimé à travers les cas d'utilisation organisés dans le modèle de spécification fonctionnelle.

Pour passer à l'analyse, nous allons changer radicalement l'organisation du modèle et nous fonder sur les principes de l'approche orientée objet, notamment sur celui d'encapsulation. À cet effet, nous allons passer d'une structura-

tion fonctionnelle *via* les cas d'utilisation et les packages de cas d'utilisation, à une structuration objet *via* les classes et les catégories.

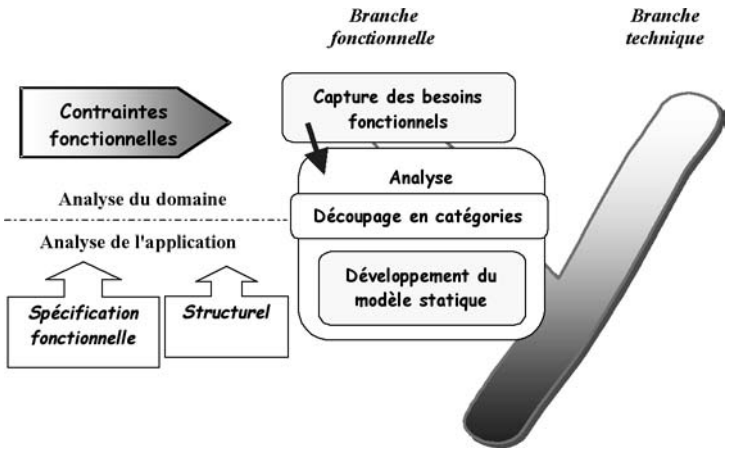

Figure 6-1 : Situation du découpage en catégories dans 2TUP

Éléments mis en jeu

- Catégorie, package,
- Dépendance, importation, visibilité,
- Diagramme de packages,
- Généralisation, association, navigabilité,
- Diagramme de classes par catégorie,
- Structuration logique.

Notion de catégorie

La classe représente une entité de structuration trop petite dès lors qu'on s'attaque à un projet réel. Au-delà d'une douzaine de classes, il est utile de regrouper les classes fortement couplées en unités plus grandes. Le couplage s'exprime à la fois structurellement par des associations, des agrégations, des compositions ou des généralisations, mais aussi dynamiquement par les interactions qui se produisent entre les instances des classes. Bien sûr, plus le nombre de classes devient important, et plus cette structuration s'avère indispensable. G. Booch [Booch 96] a introduit le concept de *catégorie* pour

nommer ce regroupement de classes qui constitue la brique de construction du modèle structurel d'analyse.

QU'EST-CE QU'UNE CATÉGORIE ?

Une *catégorie* consiste en un regroupement logique de classes à forte cohérence interne et faible couplage externe.

Le terme *catégorie* n'appartient pas en standard au langage UML qui comporte en revanche le concept plus général de package. Pour notre part, nous avons conservé ce terme, afin de différencier les catégories qui structurent un modèle construit sur des classes, du concept plus générique de package. Nous représentons graphiquement les *catégories* comme des stéréotypes de packages.

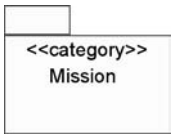

Figure 6-2 : Représentation graphique d'une catégorie

Découpage en catégories

Le découpage fonctionnel induit par les cas d'utilisation permet de trouver les classes fondamentales du projet par le biais des *diagrammes des classes participantes*. Il faut cependant considérer que l'on a seulement identifié des classes *candidates* pour l'analyse, et non les concepts métier stables de l'entreprise. En effet, les diagrammes des classes participantes capturent le vocabulaire employé dans les cas d'utilisation, mais chaque terme n'a pas encore fait l'objet d'une définition élaborée au vu du problème de l'entreprise.

Par conséquent, ces premiers diagrammes de classes doivent être réaménagés pour qu'il soit possible de poursuivre l'analyse, car on rencontre systématiquement les cas suivants :

- la même classe candidate participe à plusieurs cas d'utilisation différents, car certains concepts manipulés par les utilisateurs sont représentatifs du métier de l'entreprise ;
- des classes candidates de noms différents ont les mêmes responsabilités et participent aux mêmes collaborations, surtout si l'étude des cas d'utilisation a été menée par différents analystes. Ainsi, il existe parfois plusieurs termes pour un même concept.

Ces désavantages tiennent au fait que les concepts sont dilués dans les fonctions et ne représentent pas la problématique que doit résoudre le logiciel. À l'inverse, une approche objet offre à l'analyste la possibilité de consolider la définition des concepts, en termes de représentation à la fois structurelle et comportementale, puis de réutiliser ces définitions. Le découpage initial fonctionnel doit donc être remis en cause si l'on veut profiter des bénéfices de l'approche objet ; le découpage en catégories devient alors la base du modèle structurel d'analyse.

Si l'on entre dans le détail, on s'aperçoit que les objectifs du découpage en catégories sont multiples et souvent cruciaux pour la réussite du projet. En phase d'analyse, on peut répertorier les objectifs suivants :

- organiser les équipes d'analystes, puisqu'elles vont pouvoir travailler sur des ensembles cohérents et faiblement couplés. La cohérence implique le regroupement par compétence métier, et la diminution du couplage introduit la possibilité d'un travail en parallèle sur différents modules ;
- maîtriser la complexité par la structuration, puisque l'on va pouvoir isoler les mécanismes de détail dans les catégories et faire ressortir les collaborations d'ensemble au niveau de l'organisation du modèle structurel. La catégorie constitue donc une unité fondamentale de décomposition, qui servira les étapes ultérieures de l'analyse : conception, gestion de configuration, estimation de projets et test ;
- assurer l'évolutivité et la facilité de maintenance, et favoriser la réutilisation, en séparant notamment les parties *applicatives*, qui varient avec chaque projet et qui sont sujettes aux changements, des parties *métier* généralement plus stables et meilleures candidates à la réutilisation.

Le découpage en catégories doit être réalisé le plus tôt possible dans la phase d'analyse, en particulier pour pouvoir organiser les équipes. Ce découpage initial doit se fonder sur l'ensemble des classes candidates identifiées durant la phase précédente (voir chapitre 4), mais également sur deux principes fondamentaux : *cohérence* et *indépendance*.

Le premier principe consiste à regrouper les classes sémantiquement proches. Pour cela, il faut chercher la cohérence avec les critères suivants :

- finalité : les classes doivent rendre des services de même nature aux utilisateurs ;

  Exemple SIVEx : *Mission* et *Etape.*

- évolution : on isole ainsi les classes réellement stables de celles qui vont vraisemblablement évoluer au cours du projet, ou même par la suite. On distingue notamment les classes *métier* des classes *applicatives* ;

  Exemple SIVEx : *Facture* et *TransmissionComptable* (décrivant un composite d'objets transmis vers le module CO de SAP).

- cycle de vie des objets : permet de distinguer, et donc de gérer différemment, les classes dont les objets ont des durées de vie très différentes.

  Exemple SIVEx : *Client* et *Commande.*

Le deuxième principe consiste à renforcer ce découpage initial en s'efforçant de minimiser les dépendances entre catégories. Ce second sujet sera détaillé plus tard dans le chapitre.

ÉTUDE DE CAS : PREMIER DÉCOUPAGE EN CATÉGORIES DE SIVEX

Une première répartition des classes découvertes au chapitre 4 est illustrée à la figure 6-3. Elle offre une vue globale des catégories et des classes qu'elles contiennent[1]. Notez que UML 2.0 a officialisé le terme « diagramme de packages » pour qualifier ce type de diagramme ne contenant que des packages.

Concrètement, chaque classe candidate a été affectée à une seule catégorie, conformément aux critères énoncés précédemment. Cela ne constitue bien sûr qu'une première itération de l'organisation du modèle structurel, qui devra être retravaillé une fois les diagrammes de classes de chaque catégorie affinés. En effet, de nouvelles classes vont probablement être introduites ; certaines pourront être déplacées afin de minimiser les dépendances entre catégories.

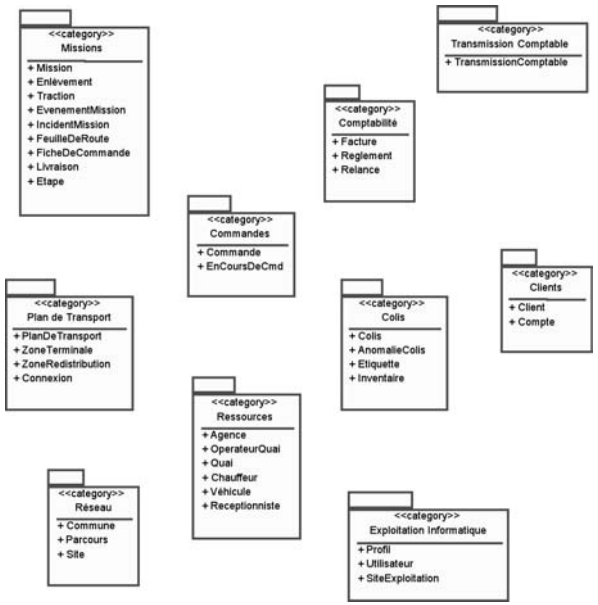

Figure 6-3 : Premier découpage en catégories de SIVEx

1. La notation de la figure 6-3 a été proposée par l'outil Rational Rose. Elle consiste à répertorier les classes appartenant à chaque package à l'intérieur du symbole graphique. Notez que le signe « + » devant les noms des classes signifie simplement « public », au sens : visible de l'extérieur du package.

Voici les raisons de ce premier découpage :

- les catégories *Réseau* et *Plan de Transport* ont été séparées selon le critère d'évolution (*Réseau* est potentiellement réutilisable, voire achetable dans le commerce...) ;
- la catégorie Exploitation Informatique a été isolée car elle correspond à un service technique classique dans toute application informatique (et donc potentiellement réutilisable), mais pas à un service métier ;
- les catégories *Comptabilité* et *Transmission Comptable* sont distinctes en fonction des critères de finalité et de cycle de vie. On retrouve également la distinction entre classe métier : Facture, et classe applicative : TransmissionComptable ;
- les catégories *Commandes* et *Clients* ont été séparées selon le critère de cycle de vie, et aussi d'évolution. On espère en effet une grande réutilisabilité pour la catégorie *Clients*.

Conseil

UNE CATÉGORIE D'ANALYSE CONTIENT MOINS DE 10 CLASSES !

Une catégorie ne doit être ni trop grosse ni trop maigre !

- Trop grosse : elle aura beaucoup de responsabilités différentes et ne pourra pas être maîtrisée par une équipe de taille raisonnable.
- Trop maigre : elle risque de ne pas avoir assez de responsabilités et de dépendre alors de nombreuses autres petites catégories. La dilution des responsabilités entraîne généralement une multiplication des couplages.

G. Booch recommandait déjà dans [Booch 96] de décomposer un système en catégories contenant une moyenne d'une douzaine de classes. Or, si nous faisons la moyenne des catégories de SIVEx, nous obtenons seulement 4 classes pour ce découpage préliminaire. D'où vient ce décalage ? N'oublions pas qu'il s'agit de classes candidates, issues d'une première itération d'analyse. Or, les catégories dont parle Booch sont des catégories de conception : elles ont subi de nombreuses itérations et incluent, en supplément, beaucoup de concepts techniques comme l'IHM ou l'accès aux données, qui apportent nettement plus de classes (voir chapitres 9 à 11).

Dépendances entre catégories

Au chapitre 4, nous avons indiqué qu'un package (et donc une *catégorie*) constitue un espace de nommage, et qu'il peut contenir des éléments UML, des diagrammes, voire d'autres packages. Cette relation de contenance est une relation de composition au sens UML. Cela signifie d'une part que tout élément UML est déclaré et possédé par un seul package. D'autre part, si le package est supprimé du modèle, tout élément inclus est également détruit.

Étude

CATÉGORIES ET IMPORTATION

Outre posséder des éléments, un package peut également importer des éléments visibles d'un autre package. Cela signifie que le deuxième package a explicitement déclaré que certains de ses éléments peuvent être utilisés par d'autres packages. UML définit ainsi deux niveaux de visibilité :

- `public` (+) : l'élément est utilisable par tout package relié par une dépendance ;
- `private` (-) : l'élément n'est utilisable que par son package parent.

En analyse, nous préfixerons simplement les classes visibles par le symbole « + ».

La relation d'importation est représentée en UML par une dépendance du package client vers le package fournisseur.

Nous allons illustrer ces différentes notions sur le schéma suivant, qui montre quelques associations concernant la classe *Mission* et des classes d'autres catégories.

ÉTUDE DE CAS : ASSOCIATIONS DE LA CLASSE *MISSION*

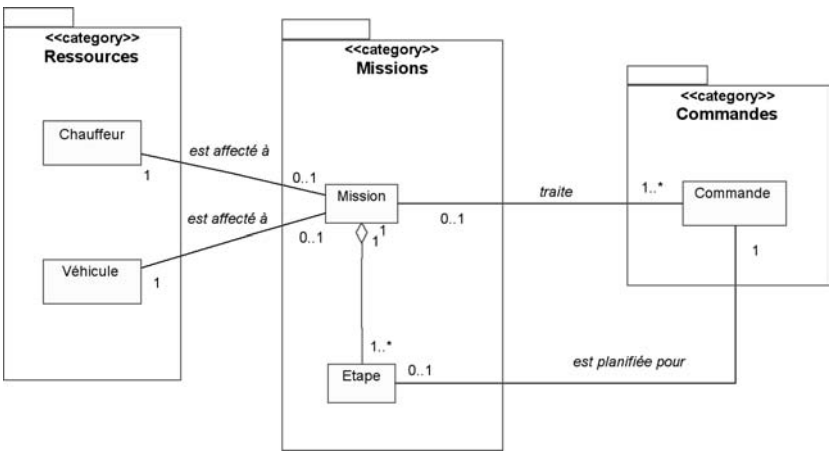

Figure 6-4 : Quelques associations concernant la classe Mission

Les classes *Mission* et *Etape*, qui appartiennent à la même catégorie, ont accès l'une à l'autre au moyen de l'agrégation. En revanche, les classes *Mission* et *Commande* ne peuvent accéder l'une à l'autre, car leurs catégories respectives forment une cloison étanche entre elles. Pour que *Mission*

puisse accéder à *Commande*, il suffit cependant que la catégorie *Missions* importe la catégorie *Commandes*. Du même coup, la classe *Etape* accédera aussi à *Commande*. Pareillement, pour que *Mission* puisse accéder à *Chauffeur* et *Vehicule*, il faut que la catégorie *Missions* importe la catégorie *Ressources*. Les relations entre les catégories correspondantes sont décrites à la figure ci-après.

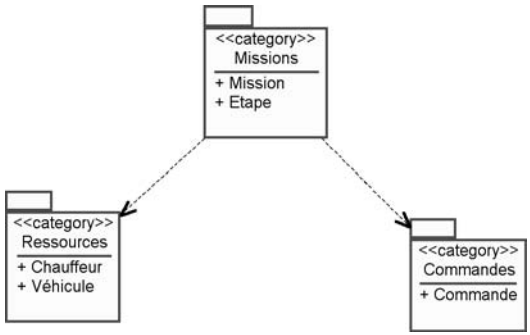

Figure 6-5 : Exemple d'importations entre catégories

Notons que l'importation est une relation unidirectionnelle, qui offre un accès aux éléments du package fournisseur pour les éléments du package client. Donc, même si la catégorie *Missions* importe la catégorie *Ressources*, les classes *Chauffeur et Véhicule* n'ont toujours pas accès à la classe *Mission*. Cette différence vient du fait que l'association est par défaut une relation bi-directionnelle entre classes. Nous allons maintenant détailler ce point.

INFLUENCE DES RELATIONS ENTRE CLASSES SUR LES DÉPENDANCES ENTRE CATÉGORIES

D'une manière générale, rappelons que si le langage UML définit quatre types de relations entre classes (dépendance, association, généralisation et réalisation), nous n'utilisons en phase d'analyse que l'association et la généralisation. Si l'on y regarde de plus près, trois de ces quatre relations sont orientées ; seule l'association est par défaut bidirectionnelle.

Lors du découpage en catégories, nous allons essayer de limiter au minimum le nombre de relations qui traversent les catégories. En effet, dès qu'une généralisation entre classes sort d'une catégorie, une dépendance de même sens entre les catégories parentes s'impose. Mais c'est encore pire pour les associations, puisque par défaut, elles vont dans les deux sens, et imposent donc une importation mutuelle des deux catégories parentes.

Dans l'exemple ci-dessous, C1 dépend de C2, mais C2 ne dépend pas de C1. En effet, si A spécialise B, et donc en dépend, la réciproque n'est pas vraie. En revanche, C3 dépend de C4, et C4 dépend également de C3, car l'association entre les classes C et D est bidirectionnelle.

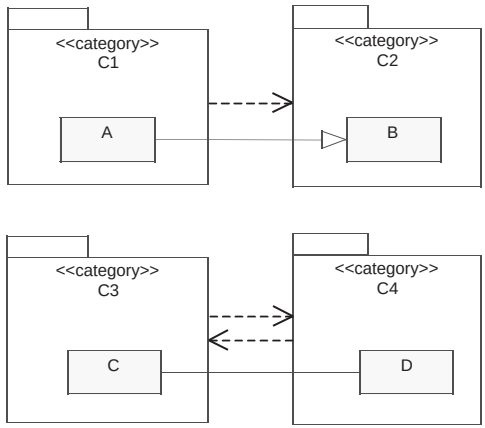

Figure 6-6 : Généralisation et association entre classes et dépendance entre packages

Le style d'architecture visé dans un projet est l'orientation *composant*, car nous en connaissons les bénéfices en termes de facilité de modularité, de maintenance, d'évolution et de réutilisation. C'est ce style qui influence ici le modèle structurel d'analyse, car il demande de minimiser les dépendances entre catégories. Le but est de pouvoir définir, dès ce stade, des composants d'analyse qui capitalisent les concepts métier d'une entreprise et sont réutilisables pour l'analyse d'autres projets. Dans un second temps, l'organisation du modèle structurel pourra persister dans le modèle logique de conception, et faciliter l'identification des composants distribués (voir chapitre 10).

Pour comprendre en quoi le couplage influence la qualité du développement, et ce dès l'étape d'analyse, nous comparons les deux situations du schéma suivant. Supposons que l'on vous propose de devenir chef de projet sur P1 ou P2. Si vous choisissez P2, c'est que vous aimez vraiment la difficulté ! En effet, eu égard aux dépendances entre catégories, le chef du projet P1 pourra raisonnablement s'organiser de la façon suivante :

- concentrer d'abord les forces sur l'analyse de C1 ;
- dès que l'analyse de C1 sera terminée, une partie des troupes basculera sur l'analyse de C2 et une autre partie sur celle de C3, qui pourront alors se dérouler en parallèle.

Son collègue malchanceux du projet P2 sait simplement qu'il aura un très gros travail de consolidation et de communication à effectuer entre ses équipes...

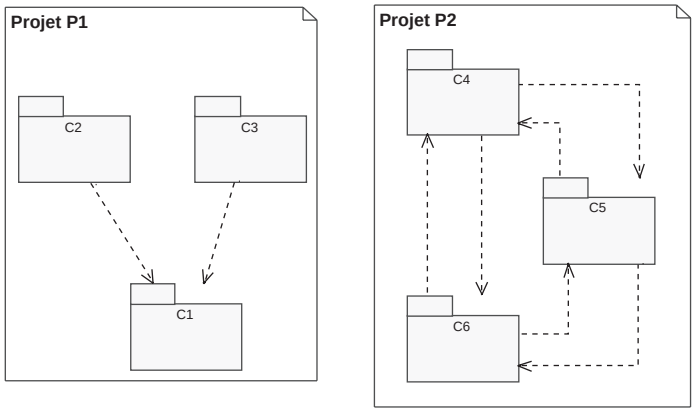

Figure 6-7 : Deux exemples de dépendances entre catégories (diagrammes de packages)

L'analyste doit être conscient qu'il existe plusieurs découpages possibles. Il se fonde sur les grands principes que nous avons énoncés précédemment. Une fois son choix effectué, il est prescriptif au niveau des dépendances entre catégories. En effet, il doit fixer des objectifs d'organisation et, éventuellement, de réutilisation.

Le concepteur, en revanche, considère le modèle d'analyse comme un ensemble d'exigences fonctionnelles à réaliser. Il ne pourra conserver les dépendances que si ses choix de conception sont compatibles avec l'organisation préconisée par l'analyste. Dans tous les cas, il devra finalement décrire les dépendances réelles du modèle de conception.

LES DÉPENDANCES ENTRE CATÉGORIES PEUVENT GUIDER LE CHOIX DE NAVIGABILITÉ DES ASSOCIATIONS !

Une association entre deux classes A et B permet par défaut de naviguer dans les deux sens entre des objets de la classe A et des objets de la classe B. Cependant, il peut être utile de limiter cette navigation à une seule des deux directions. C'est le cas pour les associations qui sortent des catégories, sans quoi, nous récupérons systématiquement une paire de dépendances. UML nous permet de représenter explicitement cette navigabilité en ajoutant sur l'association une flèche indiquant le seul sens possible.

Même si la navigabilité est le plus souvent liée à un choix de conception, et à des considérations d'efficacité, on peut l'utiliser avec discernement dès

l'analyse, pour isoler les concepts appartenant à des catégories différentes. L'exemple suivant va illustrer notre propos : la catégorie *Réseau* a été isolée précisément parce qu'elle contient des classes hautement réutilisables. Si l'on veut qu'elle ne dépende pas des autres catégories, les classes *Parcours* et *Commune* ne doivent en aucun cas avoir une quelconque connaissance des classes *Connexion* et *ZoneTerminale*. En UML, cela se traduit par la flèche de navigabilité sur les deux associations qui vont de la catégorie *Plan de Transport* à la catégorie *Réseau*.

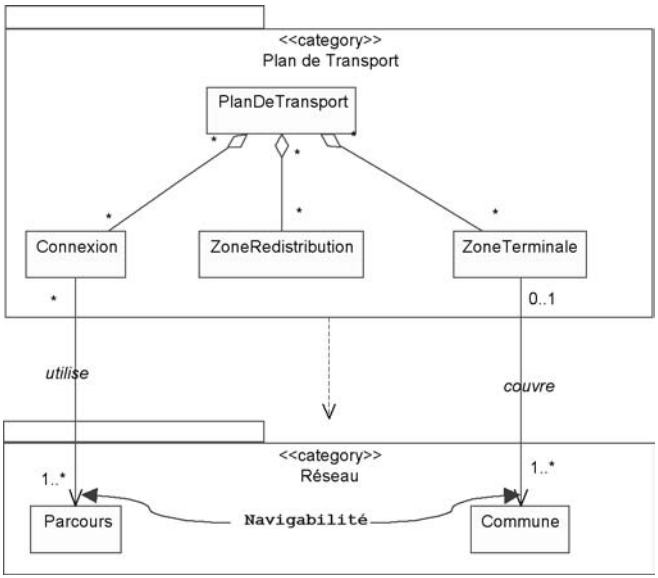

Figure 6-8 : Exemples de navigabilité d'association

Plus généralement, les dépendances souhaitées entre catégories déterminent les décisions relatives au sens des relations entre classes : associations, mais aussi généralisations, et par la suite en conception : dépendances et réalisations.

ÉTUDE DE CAS : DIAGRAMME DE CLASSES PRÉLIMINAIRE DE LA CATÉGORIE *MISSIONS*

Si nous prenons comme exemple la catégorie *Missions*, le diagramme de classes préliminaire va comprendre :

- les classes appartenant en propre à la catégorie, c'est-à-dire celles qui apparaissaient déjà à la figure 6-3 ;

Figure 6-9 : Classes propres à la catégorie Missions

- les classes des autres catégories reliées aux précédentes. Dans le diagramme de classes, une indication particulière stipule qu'elles n'appartiennent pas à la catégorie courante. Elles figurent dans ce cas sous leur nom complet, par exemple : « Ressources::Agence » ; certains outils CASE (comme Rational/Rose) ajoutent une mention de type « from PackageXXX », éventuellement renforcée par l'utilisation d'une couleur différente.

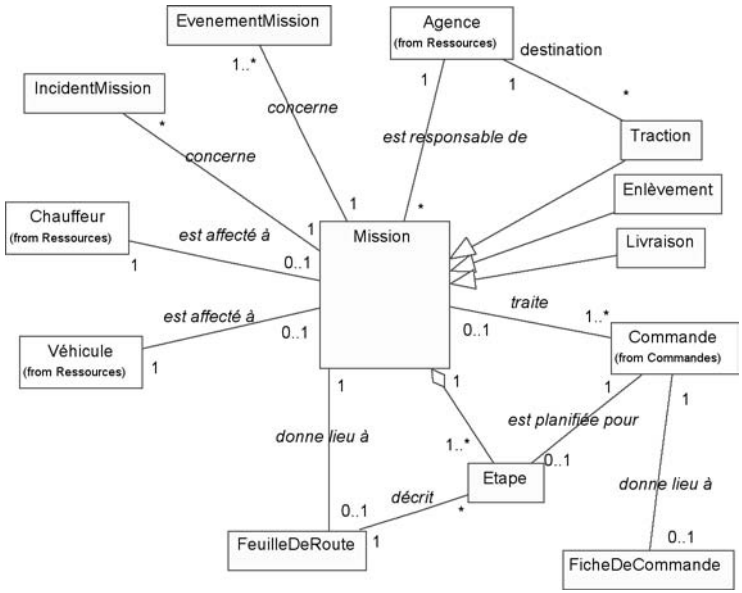

Figure 6-10 : Diagramme de classes préliminaire de la catégorie « Missions »

Le diagramme précédent montre qu'il existe des associations qui sortent de la catégorie *Missions*, et qui concernent également les catégories *Ressources* et *Commandes*. Or, l'analyste a fait le choix suivant de dépendances entre ces catégories :

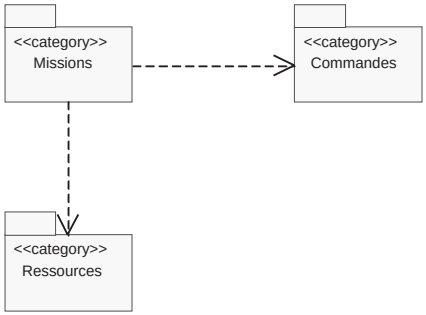

Figure 6-11 : Objectifs de dépendance de la catégorie Missions

Nous devons donc limiter la navigabilité de ces associations pour nous conformer au choix de dépendances entre catégories. Par exemple :

- chaque objet *Mission* doit connaître les ressources qui lui ont été affectées, ainsi que son *Agence* responsable, mais non l'inverse ;
- chaque objet *Mission* doit connaître les *Commandes* qu'il traite, mais non l'inverse.

Le diagramme de classes de la catégorie *Missions* devient alors :

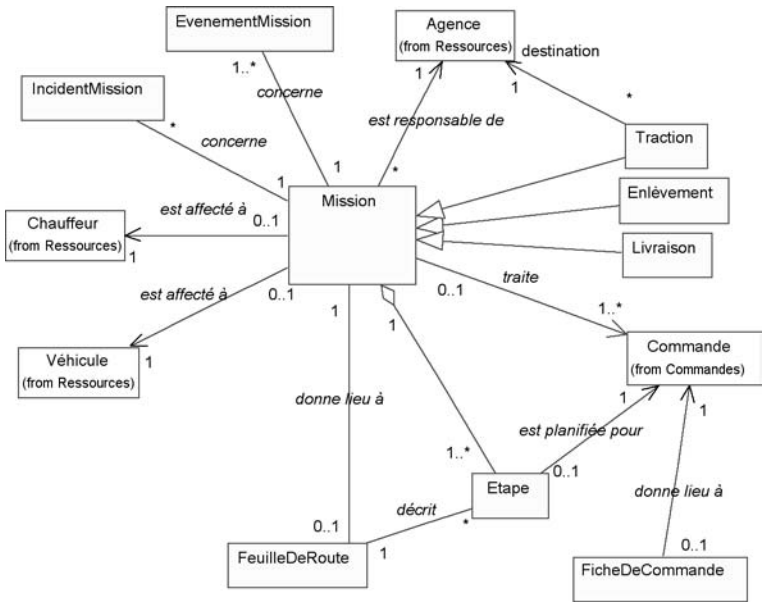

Figure 6-12 : Diagramme de classes de la catégorie Missions avec les navigabilités

LES DÉPENDANCES ENTRE CATÉGORIES NE SONT PAS TRANSITIVES !

On entend par relation transitive, la propriété de propagation d'une relation de type « est supérieur à » : ainsi si A > B et B > C, on peut en déduire que A > C. La relation « est une sous-classe de » est un exemple de relation transitive entre classes. Les dépendances entre catégories ne sont pas transitives. Par exemple, *Missions* importe *Commandes*, et *Commandes* importe *Clients.* Cela ne signifie pourtant pas que *Missions* a un accès direct à *Clients*. Cette non-transitivité des dépendances entre catégories est précieuse puisqu'elle permet d'éviter une propagation anarchique des modifications dans toute l'application. À chaque niveau, les catégories forment une sorte de rempart contre les modifications, permettant ainsi aux systèmes d'être plus évolutifs et maintenables.

ÉTUDE DE CAS : AUTRES DIAGRAMMES DE CLASSES PRÉLIMINAIRES

Pour illustrer plus avant les principes de ce chapitre, nous indiquons ci-après quelques diagrammes de classes préliminaires des catégories de SIVEx. Sur les schémas, les classes importées sont en blanc tandis que la classe considérée comme centrale pour la catégorie apparaît en foncé.

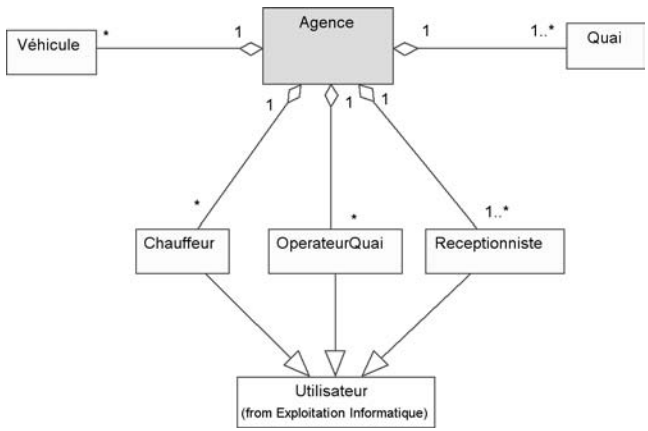

Figure 6-13 : Diagramme de classes préliminaire de la catégorie Ressources

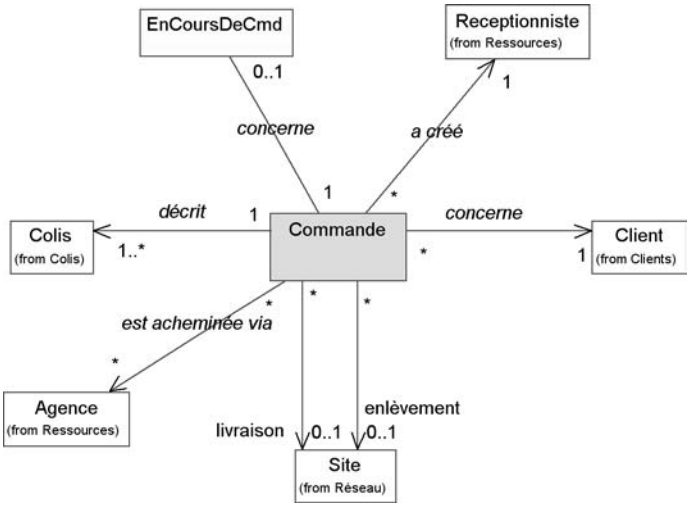

Figure 6-14 : Diagramme de classes préliminaire de la catégorie Commandes

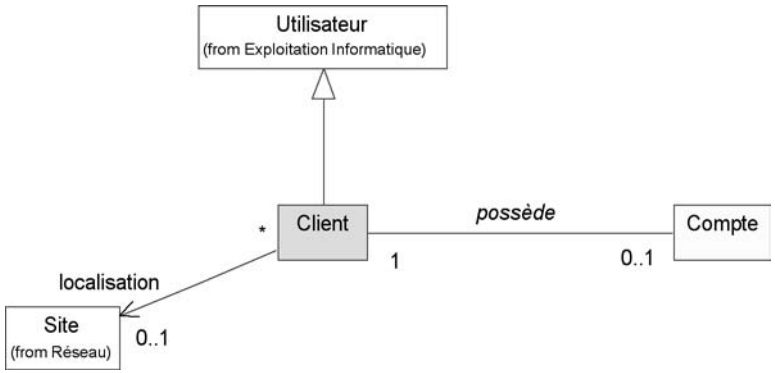

Figure 6-15 : Diagramme de classes préliminaire de la catégorie Clients

Le dernier schéma de la série représente ainsi l'état préliminaire des dépendances souhaitées entre les catégories au début de la phase d'analyse. C'est un diagramme de packages au sens UML 2.0.

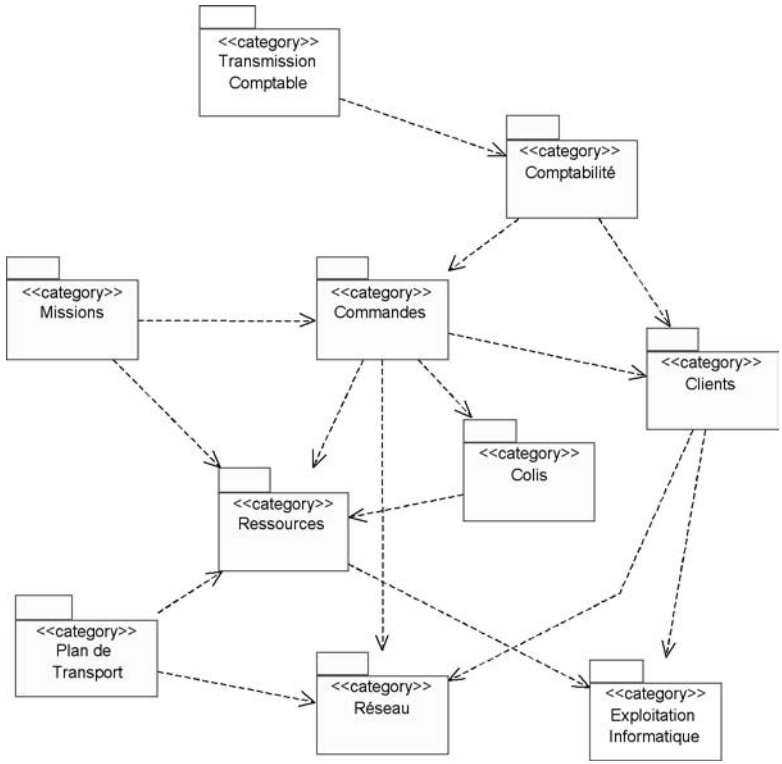

Figure 6-16 : Diagramme de packages d'analyse

Phases de réalisation du modèle structurel d'analyse

En résumé, le découpage en catégories constitue la première activité de l'étape d'analyse. C'est à ce moment-là qu'on reporte la définition des classes candidates provenant du modèle de spécification fonctionnelle au modèle structurel contenant les classes et les catégories de l'analyse. Une *catégorie* consiste en un regroupement logique de classes à forte cohérence interne et faible couplage externe. Nous la représentons par un stéréotype de package.

En analyse, pour identifier les bonnes catégories, il faut se fonder sur deux principes fondamentaux : cohérence et indépendance. Un bon découpage permet d'être plus efficace pour organiser les équipes, maîtriser la complexité, assurer l'évolutivité et favoriser la réutilisation. En d'autres termes, le découpage conditionne l'application du style d'architecture orienté composant. La recherche de limitation des dépendances entre catégories impose également des contraintes aux relations entre classes.

La démarche mise en œuvre dans ce chapitre est synthétisée par la figure suivante :

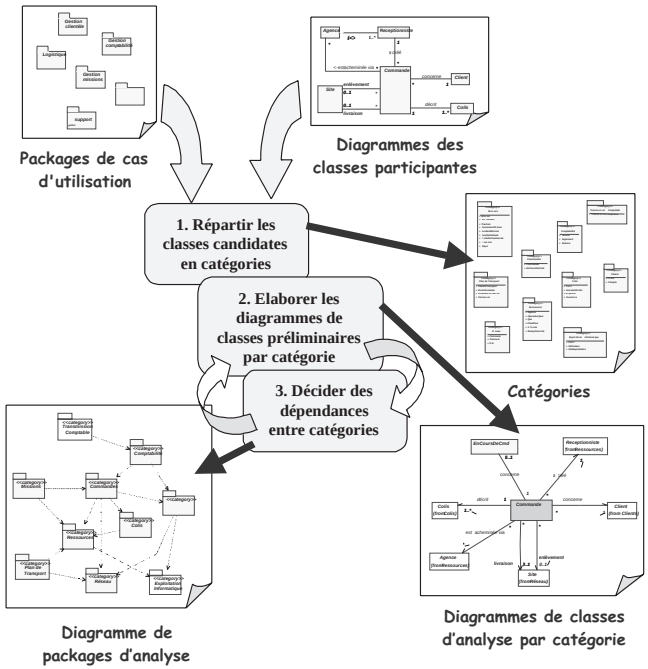

Figure 6-17 : Démarche d'élaboration du modèle structurel