Annexe A

La modélisation avec UML : la facturation de commandes

1 Introduction

Par spécification, nous entendons donner ici un modèle du cas, indépendant de toute décision d'implantation. Nous choisissons comme cas d'étude le cas "facturation" énoncé dans la section 2 du chapitre 4. Ce chapitre est très court, un de ses objectifs et d'illustrer la notation sur un cas simple, l'autre est de pouvoir comparer la spécification Z avec la modélisation UML. Nous avons choisi une présentation par diagramme puis par cas d'étude et non l'inverse, pour mettre en évidence la structure de la modélisation.

2 Cas d'utilisation et scénarios

L'objectif est de préciser les besoins énoncés dans la spécification informelle par des cas d'utilisation.

2.1 Spécification du cas 1

Dans cette phase sont identifiés les acteurs et les cas d'utilisation principaux. La première question à se poser est : "Qui déclenche le cas d'utilisation ?". La seconde question à se poser est : "Que fait le cas d'utilisation ?". Pour chaque cas d'utilisation, une description textuelle précise le cas d'utilisation. Cette description ne doit pas présumer de la modélisation en termes d'objets.

Dans le cas 1, un seul cas d'utilisation semble logique : `facturer des commandes`. Aucun acteur n'apparaît explicitement dans le texte.

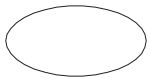

Figure 85 : *Cas d'utilisation - Invoice/Cas 1*

La description du cas d'utilisation nous amène à nous poser les questions suivantes :
- Faut-il facturer une commande ou l'ensemble des commandes ?
- Quel est l'ordre de prise en compte des commandes de l'ensemble ?
- L'ensemble des commandes contient-il des commandes déjà facturées ?
- Le stock évolue lorsqu'une commande est effectivement facturée ?
- Une commande non facturables est-elle une exception ?

Une réponse est apportée progressivement à ces questions. Il faut essayer de ne pas surspécifier à ce niveau. La règle est subjective. Par exemple, nous aurions pu prendre comme

précondition 'la commande n'est pas facturée et le stock est suffisant' et traiter les autres cas en exceptions. Mais à notre avis, ceci reste un cas cohérent et non une exception.

Cas d'utilisation : `Facturer des commandes`	
acteurs primaires :	?
description	
Pour toutes les commandes à facturer, le traitement suivant est effectué :	
cas :	`Facturer une commande`
Si la quantité en stock est suffisante pour satisfaire la commande alors l'état de la commande passe de 'en_attente' à facturé et le stock diminue de la quantité commandée sinon la commande et le stock restent inchangés.	
précondition :	Une commande est sélectionnée dans l'ensemble des commandes en attente.
Cette commande a pour état 'en_attente'.	
invariant :	`L'ensemble des commandes ne change pas.`
exceptions	
cas :	`Pas de commande en attente`
précondition :	Il n'y a pas de commande en attente.
Le traitement est annulé.	

Nous illustrons ce cas d'utilisation par un scénario représentatif, '`facturer une commande`'. Le scénario reprend directement les éléments de la description du cas d'utilisation.

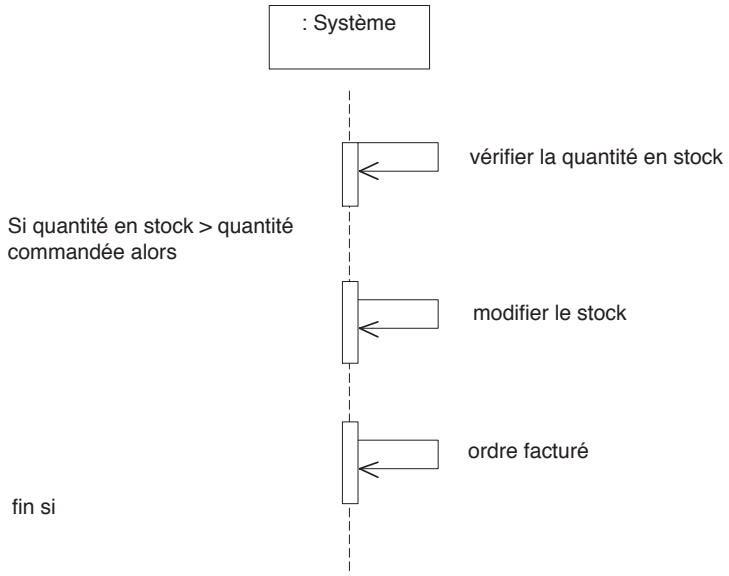

Figure 86 : *Scénario, facturer une commande - Invoice/Cas 1*

2.2 Spécification du cas 2

Les entrées en stock et les nouvelles commandes sont maintenant prises en compte de même que l'annulation de commandes. Ces trois éléments consituent assez naturellement des cas d'utilisation. Nous en rajoutons un troisième : ajout de références dans le stock pour avoir un système assez réaliste et suffisamment complet. Aucun acteur n'apparaît explicitement dans le texte. A ce niveau aussi, nous prenons une hypothèse subjective avec deux acteurs : le *client* et le *magasinier*.

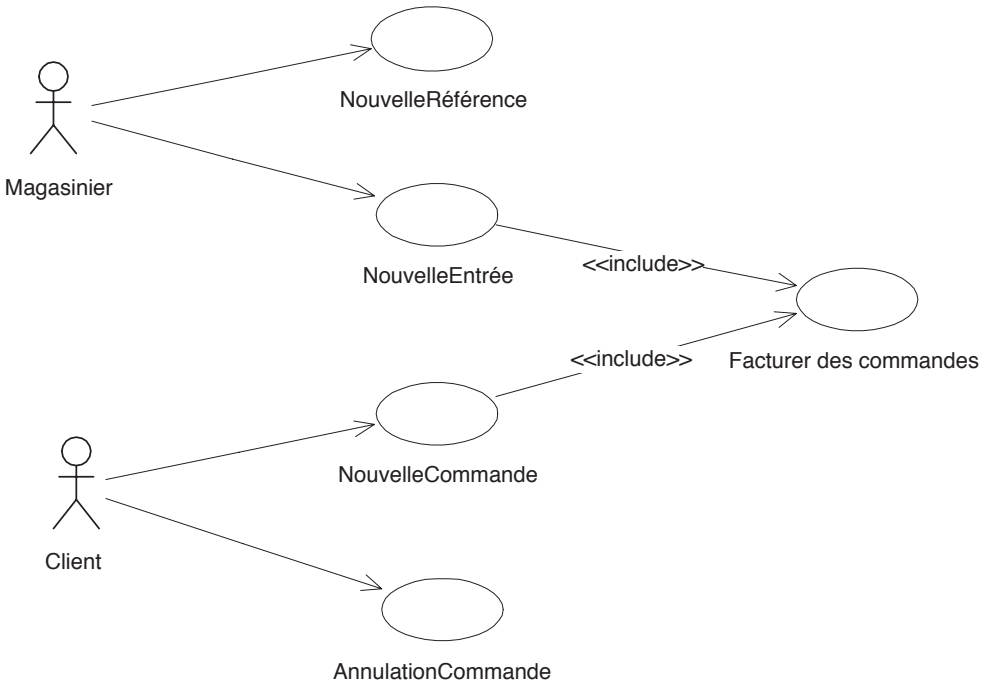

Figure 87 : *Diagramme des cas d'utilisation - Invoice/Cas 2*

Nous avons volontairement lié le cas d'utilisation `Facturer des commandes` aux autres cas d'utilisation pour mettre en valeur les stéréotypes de relations entre cas d'utilisation.

Le stéréotype `«include»` indique une "inclusion de la séquence de comportement du cas utilisé dans l'autre cas". Lorsqu'il y a plusieurs cas d'utilisation inclus, l'ordre d'utilisation est précisé dans le cas d'utilisation englobant. Le stéréotype `«extend»` indique une "extension à une partie du comportement défini dans un autre cas". La relation d'extension inclue une condition d'extension et l'endroit où les extensions sont ajoutées : lors d'un appel du cas d'utilisation, à la rencontre du point d'extension la condition est évaluée et la séquence étendue est ajoutée si l'évaluation est vraie [Con99]. Certains auteurs voient la relation `«include»` comme une utilisation systématique et la relation `«extend»` comme une utilisation facultative. D'autres font le lien entre l'extension la relation de généralisation/spécialisation de telle sorte que les liens du cas d'utilisation étendu sont hérités. Nous utilisons la règle suivante : un cas d'utilisation est une extension si il a le même sens que le cas étendu.

Hypothèse 2.1 *Le stéréotype `«include»` nous permet de déclencher la facturation à chaque nouvelle commande et chaque nouvelle entrée i.e. à chaque perturbation directe ou indirecte du stock. C'est un choix arbitraire. Ce cas d'utilisation aurait tout aussi bien pu être lancé périodiquement automatiquement par le système ou manuellement par le magasinier ou un nouvel acteur `Gestionnaire`.*

Cas d'utilisation : `NouvelleCommande`	
inclut	`Facturer des commandes`
acteurs primaires :	`Client`
invariant :	`Les références ne changent pas mais le stock peut évoluer.`
description	
cas :	`NouvelleCommande`
Le client effectue une commande en fournissant la référence du produit et la quantité commandée.	
précondition :	La référence du produit correspond à un produit référencé dans le stock.
Une nouvelle commande est ajoutée dans l'ensemble des commandes. Puis, la facturation de commande est activée.	
...	

...	
exceptions	
cas :	`Produit inexistant`
précondition :	La référence du produit ne correspond pas à un produit référencé dans le stock.
La commande est refusée.	

Un scénario standard est proposé dans la figure 88.

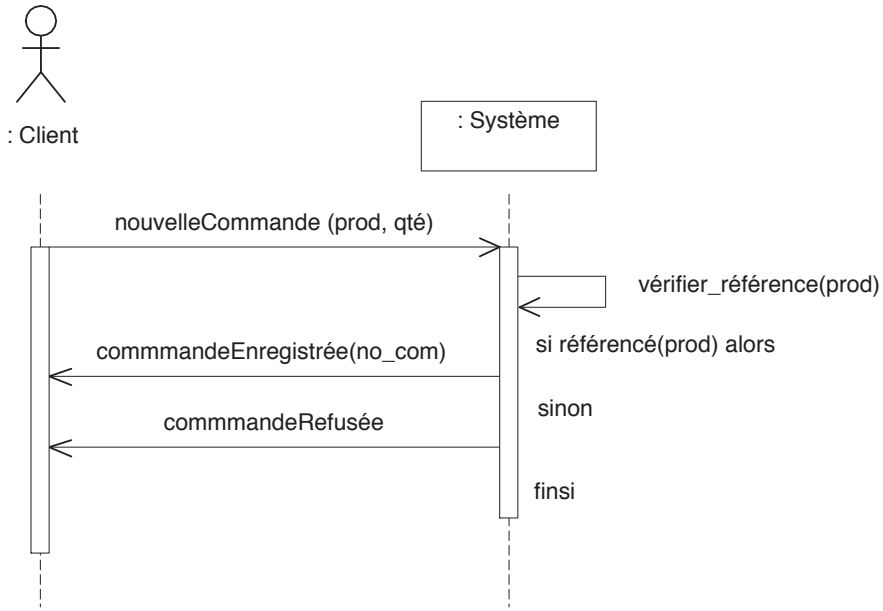

Figure 88 : *Scénario, nouvelle commande- Invoice/Cas 2*

Cas d'utilisation : `NouvelleEntrée`	
inclut	`Facturer des commandes`
acteurs primaires :	`Magasinier`
invariant :	`L'ensemble des commandes ne change pas` mais les commandes peuvent changer.
description	
cas :	`Nouvelle entrée`
La magasinier enregistre une livraison de produit. Il fournit la référence du produit livré et la quantité reçue.	
précondition :	La référence du produit correspond à un produit référencé dans le stock.
La quantité de produit est ajoutée dans le stock. Puis la facturation de commande est activée.	
exceptions	
cas :	`Produit inexistant`
précondition :	La référence du produit ne correspond pas à un produit référencé dans le stock.
L'entrée de produit est refusée.	

Un scénario standard est proposé dans la figure 89.

L'annulation d'une commande nous amène à nous poser diverses questions : qui annule, de quel droit, comment est repérée la commande à annuler dans l'ensemble ds commandes ? Peut-on annuler une commande déjà facturée ?

Hypothèse 2.2 *Les commandes sont identifiées par un numéro. Seules les commandes non facturées peuvent être annulées, sinon il fadrait remettre la quantité facturée dans le stock.*

Hypothèse 2.3 *Les commandes sont anonymes i.e. sans référence à un client donné. Dans le cas contraire, il faut mémoriser qui a passé la commande. Ainsi, l'annulation de commande peut être conditionnée par le fait que le client a passé la commande auparavant.*

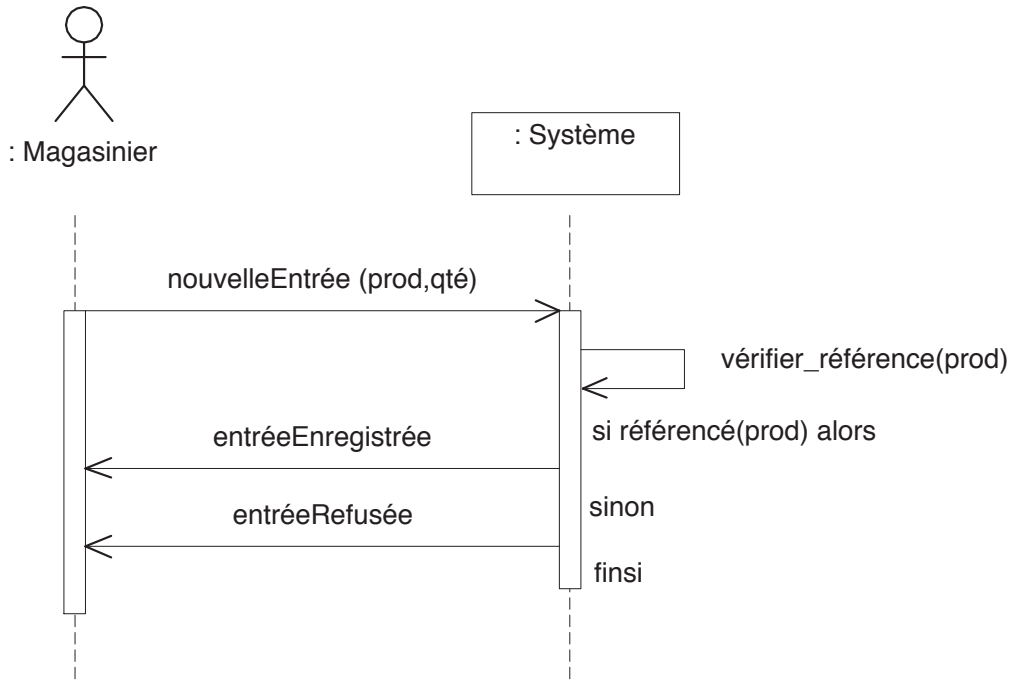

Figure 89 : *Scénario, nouvelle entrée - Invoice/Cas 2*

Cas d'utilisation : `AnnulerCommande`	
acteurs primaires :	`Client`
invariant :	`Le stock ne change pas.`
description	
cas :	`Annuler une commande`
Le client annule une commande en fournissant le numéro de la commande.	
précondition :	Le numéro fourni correspond à celui d'une commande de l'ensemble des commandes ET la commande est en attente.
La commande est ôtée de l'ensemble des commandes.	
exceptions	
cas :	`Commande inexistante`
précondition :	Le numéro fourni ne correspond pas à celui d'une commande de l'ensemble des commandes.
L'annulation de la commande est impossible.	
cas :	`Commande déjà facturée`
précondition :	Le numéro fourni correspond à celui d'une commande déjà facturée de l'ensemble des commandes.
L'annulation de la commande est impossible.	

Un scénario standard est proposé dans la figure 90.

Cas d'utilisation : `NouvelleRéférence`	
acteurs primaires :	`Magasinier`
invariant :	`L'ensemble des commandes ne change pas.`
description	
cas :	`Nouvelle référence`
Le magasinier rentre une nouvelle référence de produit en stock.	
Il fournit une référence de produit.	
précondition :	La référence du produit ne correspond pas à un produit référencé dans le stock.
Le référencement est accepté et le produit est référencé dans le stock avec une quantité nulle.	
exceptions	
cas :	`Référence existante`
précondition :	La référence du produit correspond à un produit déjà référencé dans le stock.
Le référencement est refusé.	

Un scénario standard est proposé dans la figure 91.

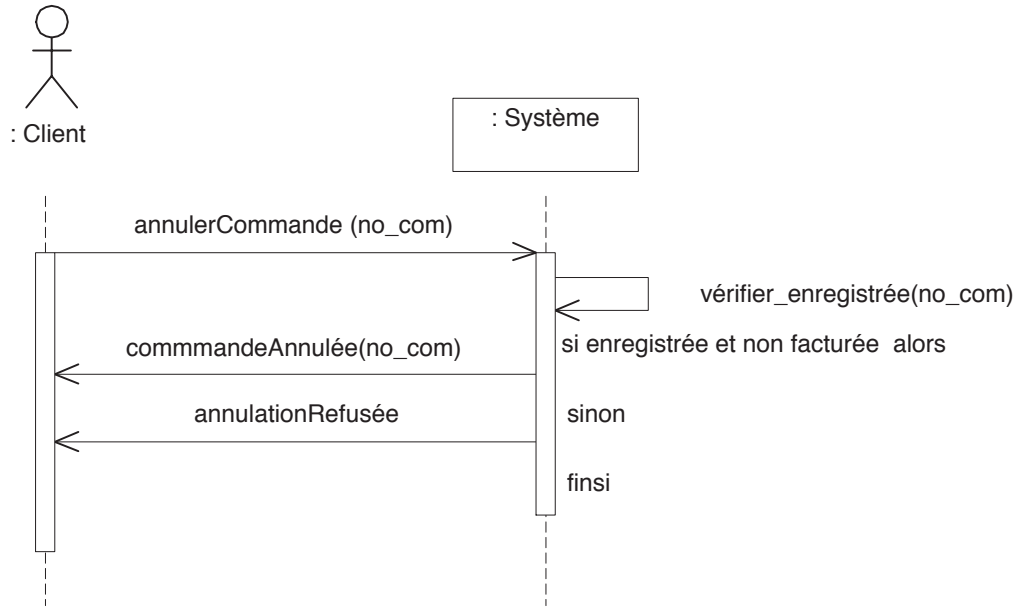

Figure 90 : *Scénario, annuler une commande - Invoice/Cas 2*

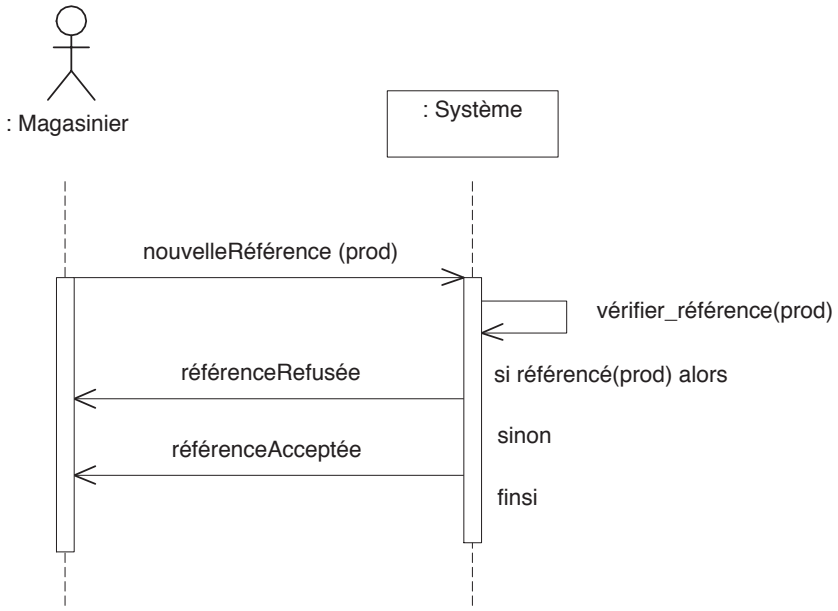

Figure 91 : *Scénario, nouvelle référence - Invoice/Cas 2*

2.3 Bilan

Nous avons donné une idée générale du besoin. Dans les scénarios du cas 2, nous avons inclus le cas normal et le cas exceptionnel. Ceci met en évidence la frontière *fragile* entre alternative et exception. L'étape suivante va consister à préciser les interactions à l'intérieur du système.

3 Diagrammes d'objets

L'objectif est de mettre en évidence une parie des objets du système, qui vont prendre en charge les besoins énoncés dans la spécification des cas d'utilisation. Le type des envois de messages (synchrone, asynchrone...) n'est pas spécifié. Le résultat est noté sous forme d'un message spécial `^(resul)`.

3.1 Spécification du cas 1

La mise en évidence des objets du système peut passer par des diagrammes d'objets, soit des diagrammes de séquence (scénarios enrichis) soit des diagrammes de collaboration. L'objectif est de faire apparaître de nouveaux objets. Nous avons constaté deux habitudes :

- soit introduire des objets interface (inspiration d'Objectory), par exemple des fenêtres ou des menus.
- soit introduire des objets quelconques qui prennent en charge l'activité à réaliser.

Nous avons choisi la première approche, car elle induit une idée de communication entre l'utilisateur et le système d'information. Nous utilisons un objet interface : la `FenêtreFacturation`.

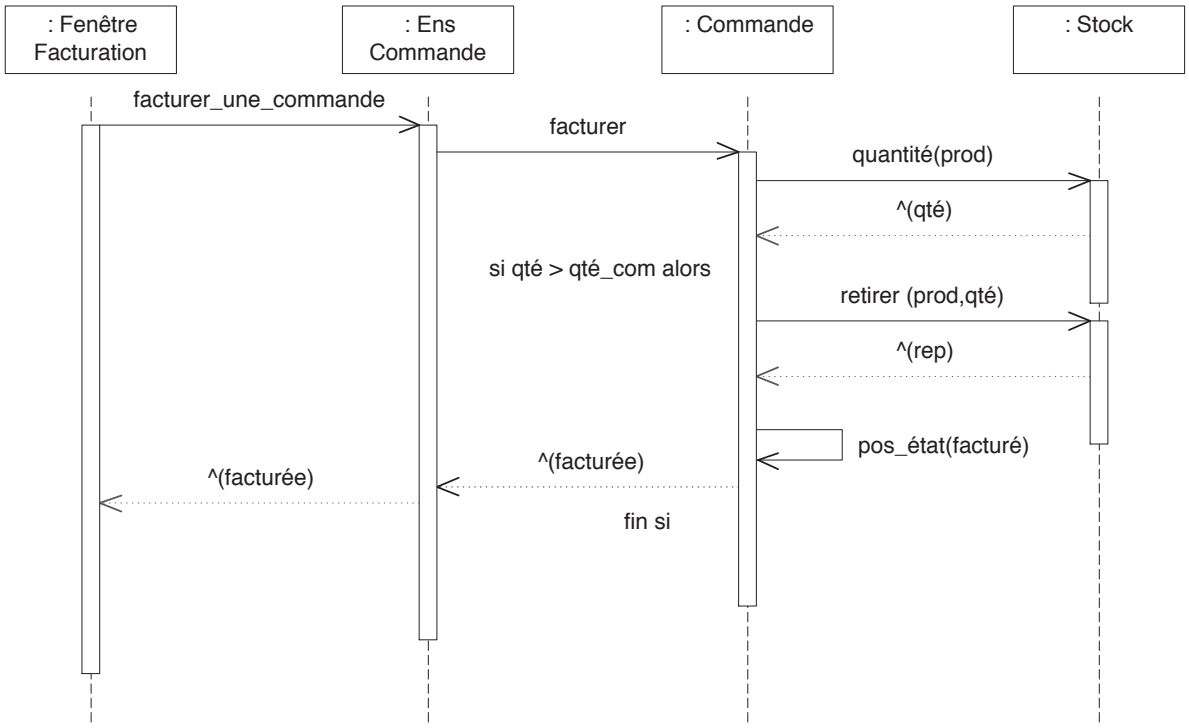

Figure 92 : *Diagramme de séquence, facturer une commande - Invoice/Cas 1*

On suppose un ensemble de commandes, dans lequel on prend une commande à facturer. Celle-ci établit un rapprochement avec le stock pour vérifier la disponibilité du produit. Nous aurions pu faire communiquer la commande avec un produit, mais utiliser le stock nous semble plus général. Le diagramme de collaboration correspondant met en évidence une première structuration des objets, qui peut inspirer un diagramme des classes.

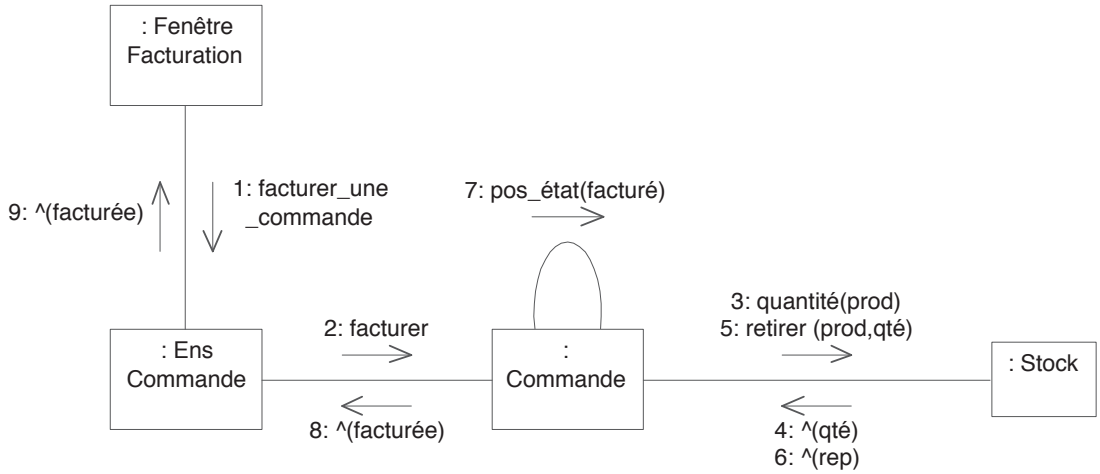

Figure 93 : *Diagramme de collaboration, facturer une commande - Invoice/Cas 1*

3.2 Spécification du cas 2

Le travail réalisé ici reprend les principes du cas 1. Nous illustrons indifféremment les cas par des diagrammes de séquence et des diagrammes de collaboration. En effet, peu de contraintes temporelles sont mise en évidence.

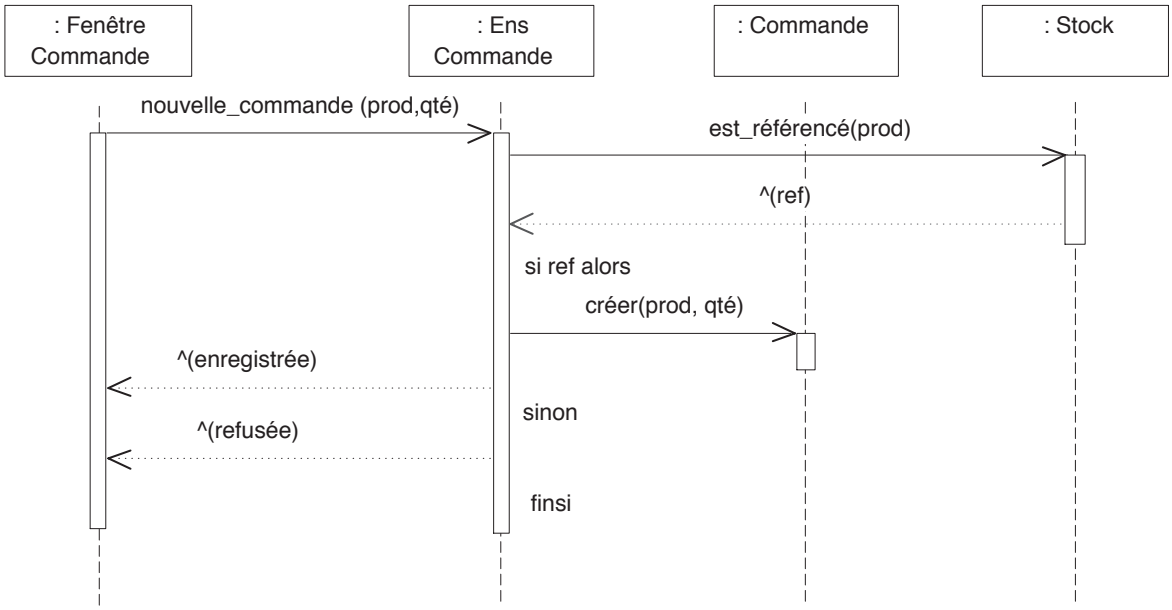

Figure 94 : *Diagramme de séquence, nouvelle commande - Invoice/Cas 2*

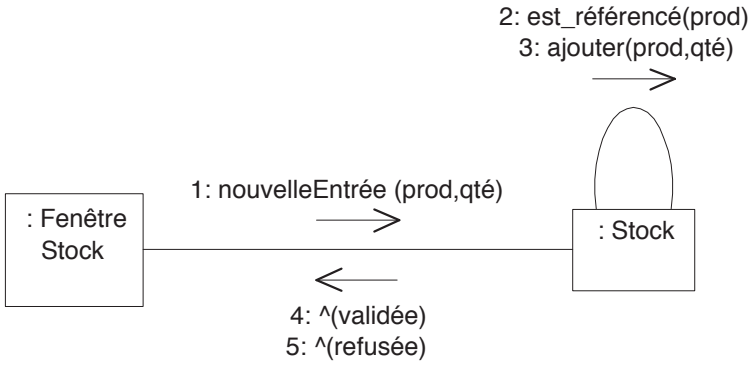

Figure 95 : *Diagramme de collaboration, nouvelle entrée - Invoice/Cas 2*

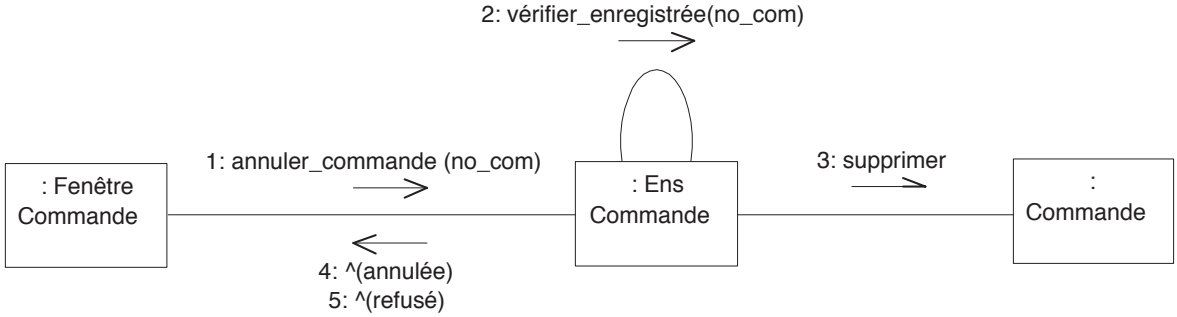

Figure 96 : *Diagramme de collaboration, annuler une commande - Invoice/Cas 2*

Noter que l'envoi de message de suppression est asynchrone. On n'attend pas le résultat pour poursuivre. Dans la figure 97, l'opération *créer* est une opération de classe de la classe *Produit*. Nous l'avons préfixée par '$' pour la mettre en évidence. Dans le diagramme des classes, les propriétés de classes sont soulignées.

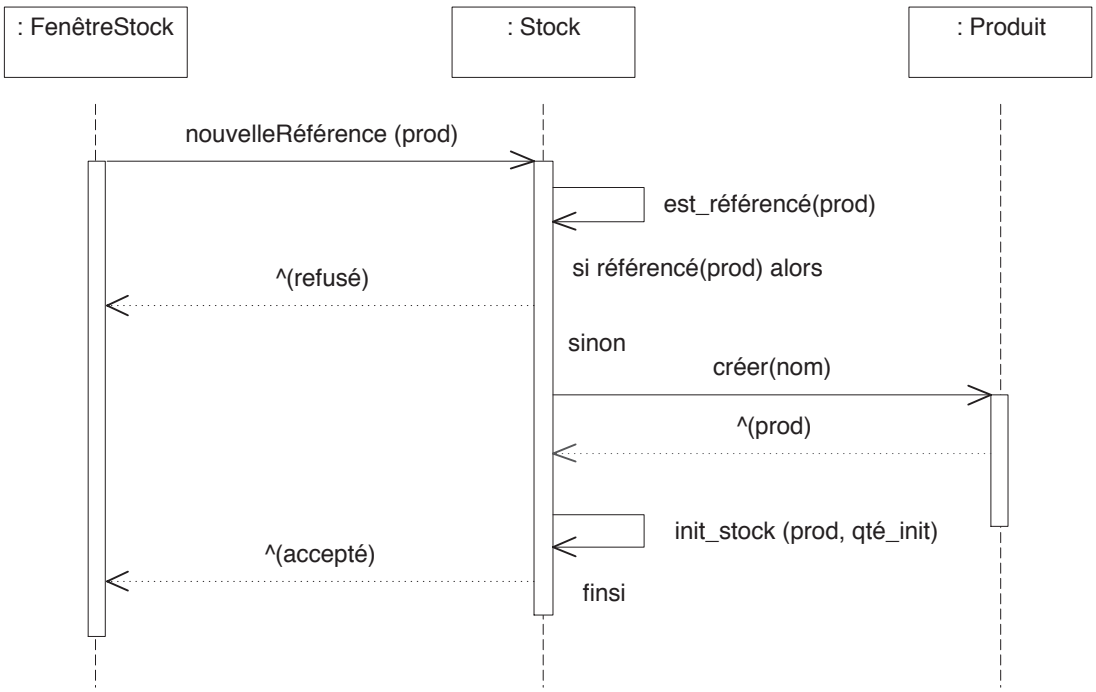

Figure 97 : *Diagramme de collaboration, nouvelle référence - Invoice/Cas 2*

3.3 Bilan

Nous avons donné une idée générale des objets du système. L'étape suivante va consister à abstraire ces représentations et mettre en évidence des structures générale du système.

4 Diagrammes des classes

L'objectif est de mettre en évidence les classes du système.

4.1 Spécification du cas 1

Des diagrammes d'objets précédents, nous déduisons le diagramme de classe suivant. Toujours selon les habitudes d'Objectory, les objets sont regroupés par nature. Ces natures peuvent conduire à des stéréotypes.

- objet `interface` pour exprimer les interactions entre le système et les son environnement logique (utilisateurs),
- objet `dispositifs` pour exprimer les interactions entre le système et son environnement physique,
- objet `du domaine` ou objets métier qui sont les objets essentiels du discours de l'utilisateur,
- objet `de contrôle` qui font le lien entre les objets interface et les objets du domaine,
- objet `utilitaires` ou techniques qui sont des objets utiles à l'implantation.

Ces stéréotypes d'objets sont en général regroupés dans des paquetages distincts. Le modèle statique est, par convention, principalement le modèle des objets métiers. Nous l'enrichissons de propriétés (attributs et méthodes).

La connaissance du domaine permet de préciser ou d'ajouter certaines classes. Ainsi, nous avons mis en évidence une classe produit, qui établit une relation (indirecte) entre la commande et le stock.

Le modèle obtenu est incomplet mais se doit de respecter autant que possible et le plus précisément le texte de départ.

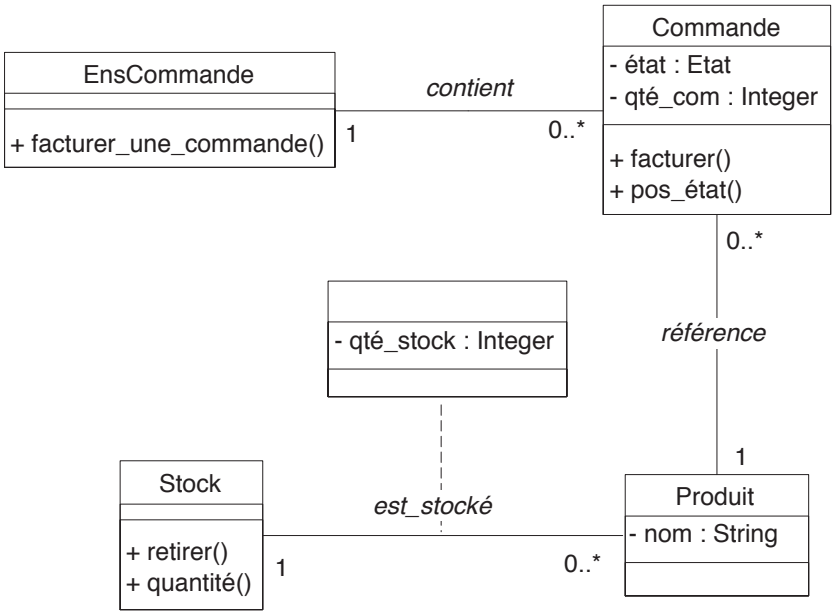

Figure 98 : *Diagramme des classes - Invoice/Cas 1*

4.2 Spécification du cas 2

L'étude des collaborations fait émerger deux groupes de classe : les classes d'interface et les classes du domaine du problème de la facturation. Nous les avons rangées en deux paquetages : *Interface* et *Domaine* .

Le diagramme de la figure 99 met en évidence les classes métier.

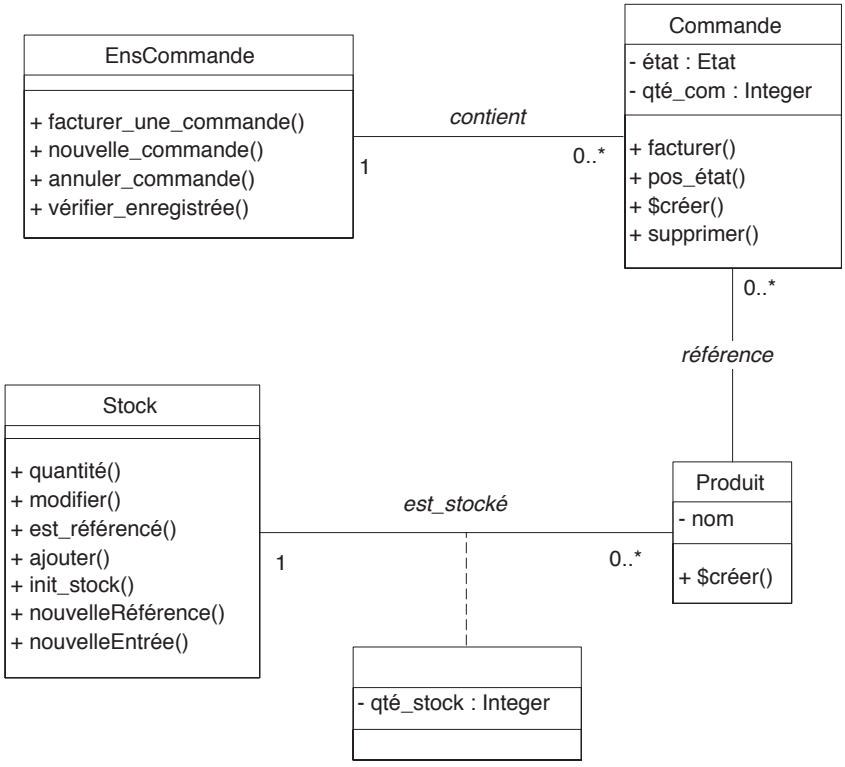

Figure 99 : *Diagramme des classes, métier - Invoice/Cas 2*

Le diagramme de la figure 100 met en évidence les fenêtres de l'interface. Nous avons représenté les relations entre fenêtres et éléments du domaine. Ces relations ne sont pas forcément implantées sous cette forme en conception détaillée.

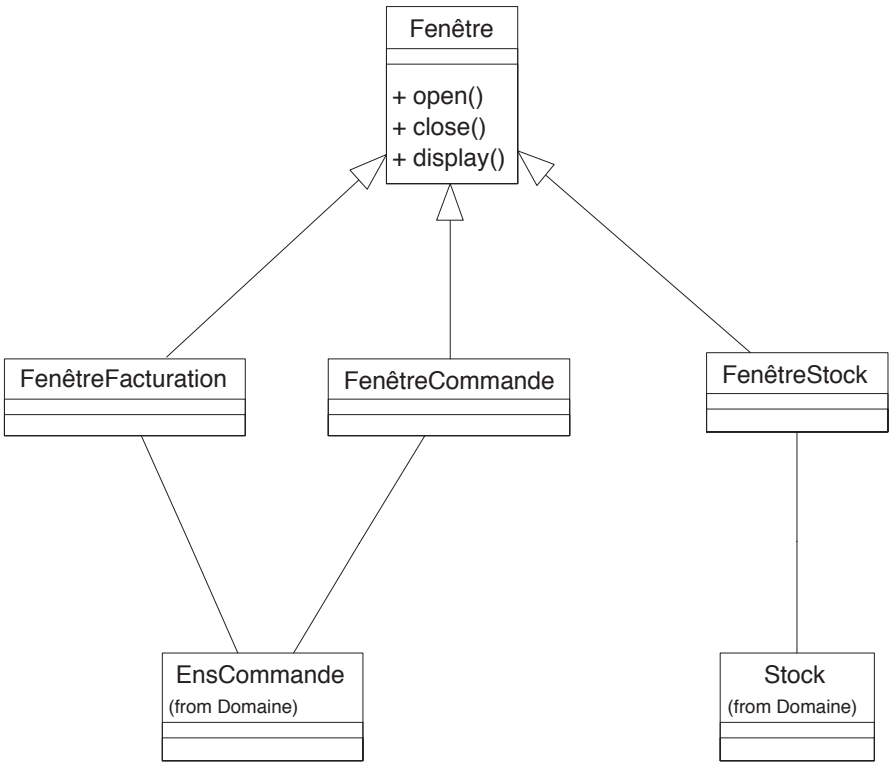

Figure 100 : *Diagramme des classes, interface - Invoice/Cas 2*

5 Diagrammes états/transitions

Un seul objet a un comportement dynamique complexe : la `Commande`. Nous en précisons les états (abstraits) et les transitions. Les événements correspondent aux actions (opérations) invoquées.

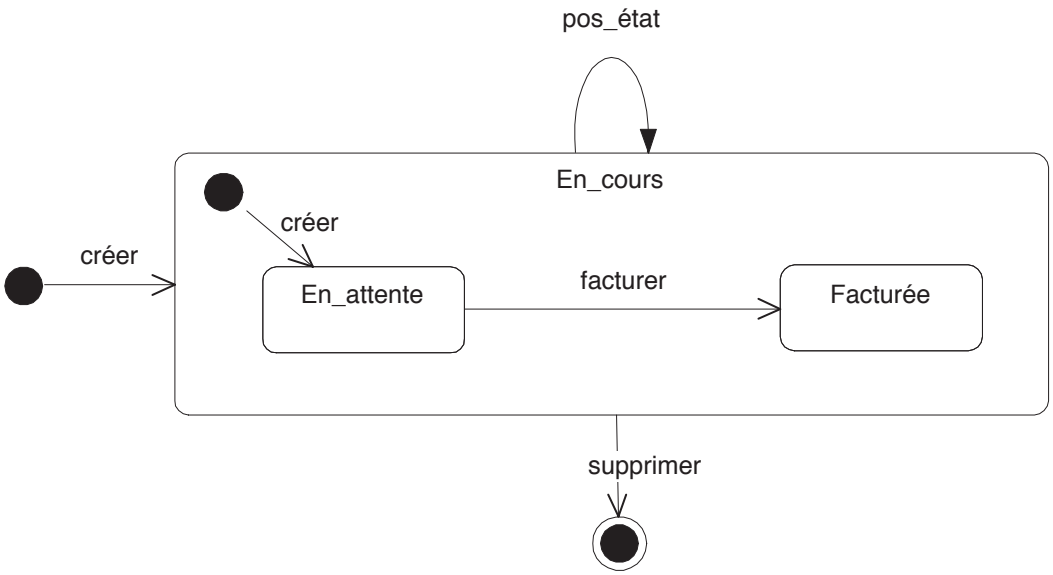

Figure 101 : *Diagramme états-transitions, classe* `Commande` *- Invoice/Cas 2*

6 Conclusion

Nous avons utilisé la notation UML pour modéliser le cas facturation. Nous avons mis en évidence une démarche de modélisation : à savoir préciser l'énoncé par les cas d'utilisa-

tion, extraire les objets nécessaires et structurer l'ensemble par un diagramme des classes. L'organisation générale de la spécification est représentée dans la figure 102.

Le cas étant relativement simpliste, les cas d'utilisation deviennent proches d'une description fonctionnelle, même si elle est plus abstraite, dans la mesure où la marge de manœuvre de l'implantation est relativement large. Le modèle obtenu est relativement complet et homogène. Une programmation des interfaces a été réalisée en Smalltalk-80. Une autre modélisation de ce cas en UML est proposée à titre de comparaison dans l'article de C. SAINT-MARCEL et al. *The Invoicing System : in UML*, dans la conférence "Invoice'98" des 26 et 27 mars 98 à Nantes.

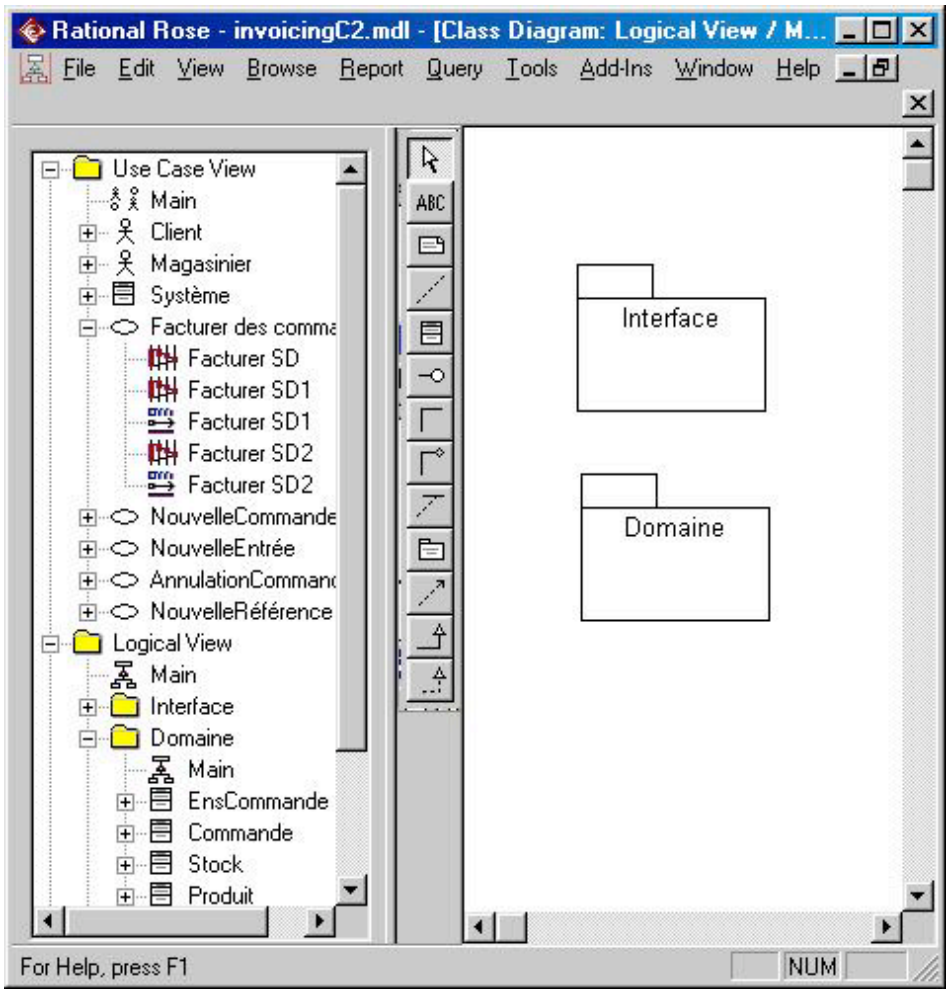

Figure 102 : *Organisation générale - Invoice/Cas 2*