Démarche d’analyse et de conception avec le langage UML

Je tiens à remercier mon chat et mon poisson rouge (jusqu'à ce que mon chat ne le mange) pour l'aide et le soutien moral qu'ils m'ont apportés.

Index

Préambule

Cette démarche de conception est la démarche proposée par Craig LARMAN. Je l'ai connue à travers le cours "Analyse et conception avec UML et les Patterns " de la société Valtech.
J'ai ici repris cette démarche, je l'ai légèrement simplifiée, et j'ai apporté les informations nécessaires pour que des stagiaires de l'AFPA puissent s'imprégner de cette démarche.
Dans un premier temps j'ai repris l'exercice du cours Valtech. Souhaitons qu'il y ait un deuxième temps …

Les formateurs qui désirent former leurs stagiaires à cette démarche devraient suivre le cours pré cité, afin de prendre du recul par rapport à la démarche.

Je recommande à tous la lecture des ouvrages de Craig LARMAN sur UML.

Objectif global d'une démarche d'analyse et de conception objet

Le but de cette démarche est d'analyser et de concevoir une application objet. L'objectif est d'avoir une application qui puisse vivre (évolution de l'application dans le temps), et qui enrichisse la base de savoir-faire de l'entreprise (développement de nouveaux applicatifs en s'appuyant sur des objets métiers déjà développés) . La réutilisabilité est un des soucis principaux pour pouvoir réduire le coût des logiciels et augmenter la puissance de développement des programmeurs.
Nous nous appuierons sur le workflow, c'est à dire les règles des processus métier pour débusquer les uses cases, et les acteurs. Dans l'analyse et la conception objet nous cherchons à construire une architecture en couche de la forme suivante :

couche I.H.M.	présentation
couche application (workflow)	application
couche objets métiers	métier
couche accès aux données	accès aux données
couche base de données	base de données

Nous verrons que chaque couche définira un ou des contrôleurs pour la rendre indépendante des autres.
Une étude a montré que dans certaines entreprises les règles métier changeaient de 10% par mois ! ! ! Il est facile de comprendre alors la raison de ce découplage des objets métiers. Les objets ont tendance à être stables dans le temps, par contre les IHM et base de données peuvent également être changées sans que l'on touche au métier ni aux règles de l'entreprise.
Cette architecture ne s'appliquera bien sûr pas à toutes les applications, c'est à prendre comme un exemple complet.

Les étapes de la démarche

Ce chapitre est un résumé du processus de développement d'une application. Il est difficile de comprendre ce résumé, sans avoir lu en détail et expérimenté chacun des chapitres auquel il fait référence. Par contre je vous conseille de revenir souvent à ce résumé (textuel et graphique) pour bien vous situer dans la démarche, avant l'étude de tout nouveau chapitre.

- Lister l'ensemble des **exigences** du client issues du cahier des charges, ou issu d'une démarche préalable de collecte d'information (documents électroniques, notes de réunions, notes d'interviews, …). Chaque exigence sera numérotée et ainsi pourra être tracée.
- Deux solutions possibles :
  - Nous allons regrouper les exigences par **intentions d'acteur** complètes puis nous allons faire un diagramme de contexte (nous nous appuierons sur un diagramme de collaboration pour cela)
  - Si toutes les règles de processus métier sont définies, nous réaliserons un diagramme d'activité en colonnes ("swim lane") où les colonnes sont les acteurs. Cela permet de dégager les responsabilités de chaque acteur, et ainsi de connaître les intentions des acteurs.
- Définir les uses cases et construire le diagramme de **uses cases.**
- Faire la **description de haut niveau** de chaque use case **:** chercher des scénarios de base.
- Faire la **description détaillée** de chaque use case : donner les séquences nominales puis les séquences alternatives (Erreurs et exceptions, en limitant au maximum les exceptions). Cette description peut être complétée par un diagramme d'activité qui montre le séquencement des différents scénarios.
  Cette description détaillée comprend les scénarios, les pré conditions, à partir de quoi se déroule le use case, comment se termine le use case, et enfin les post conditions (règles métier assurées quand le use case est terminé).
- Faire un diagramme de séquence par scénario (ici c'est un diagramme de séquence boîte noire, où la boîte noire correspond au système informatique à développer).
- Faire un diagramme de classe par use case. Le diagramme de classe final sera la superposition de ces diagrammes de classe.
  Les classes obtenues sont des classes du monde réel.
  Nous ne mettons pas les opérations car nous ne connaissons pas l'intérieur du système informatique.
- Un contrat est réalisé pour chaque opération du système. Ce contrat contient un nom, des responsabilités, les exigences auxquelles répond notre itération de cycle de vie, les pré conditions, et les post conditions (décrites sous la forme " has been ") et enfin les exceptions.
  Ce contrat peut être réalisé sous forme textuelle, ou plus souvent sous forme d'un diagramme de séquence boîte blanche où les objets échangent des messages pour rendre le service demandé.
- A partir des diagrammes de classe et des contrats nous réaliserons les diagrammes de collaboration qui montrent comment les objets collaborent pour rendre le service demandé. Nous appliquerons les patterns de conception pour cela (GRASP patterns)
- En parallèle nous réaliserons les diagrammes d'état des objets les plus complexes, ainsi nous détecterons des méthodes internes à ces objets.

- Nous réaliserons enfin le diagramme de classe de conception, en tenant compte à nouveau des GRASP patterns. Ceci peut remettre en cause le diagramme de classe précédemment établi.

Processus simplifié

DEFINIR BESOINS

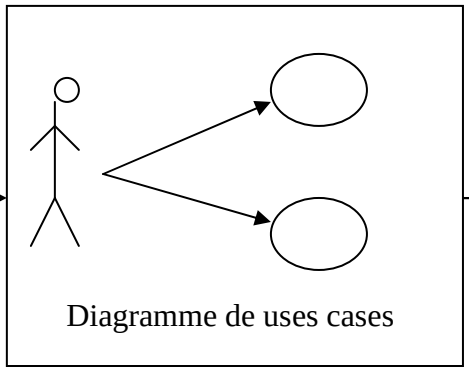


Diagramme de uses cases

nom : effectuer un achat
acteur principal : client
acteur secondaire : caissier
événement déclencheur : …
rôle du use case : …
terminaison du use case : …

Use case haut niveau

ANALYSE

nom : effectuer un achat
acteur principal : client
acteur secondaire : caissier
événement déclencheur : …
rôle du use case : …
terminaison du use case : …
références croisées : …
pré conditions : …
scénarios et alternatives : …

Use case bas niveau

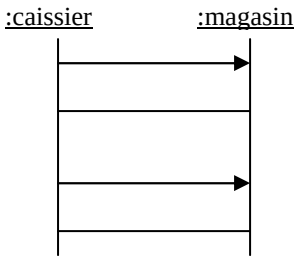


Diagramme de séquence

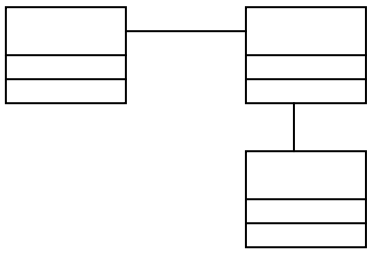


Diagramme de classe d'analyse

nom : …
responsabilité : …
exigences : …
notes : …
pré conditions : …
post conditions : …

Contrat d'opération

CONCEPTION

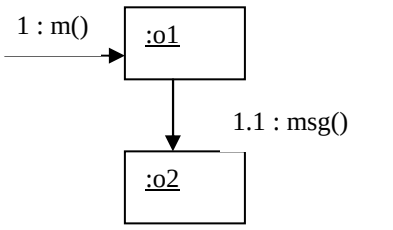


Diagramme collaboration

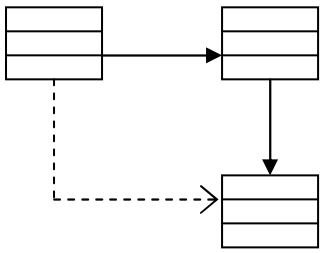
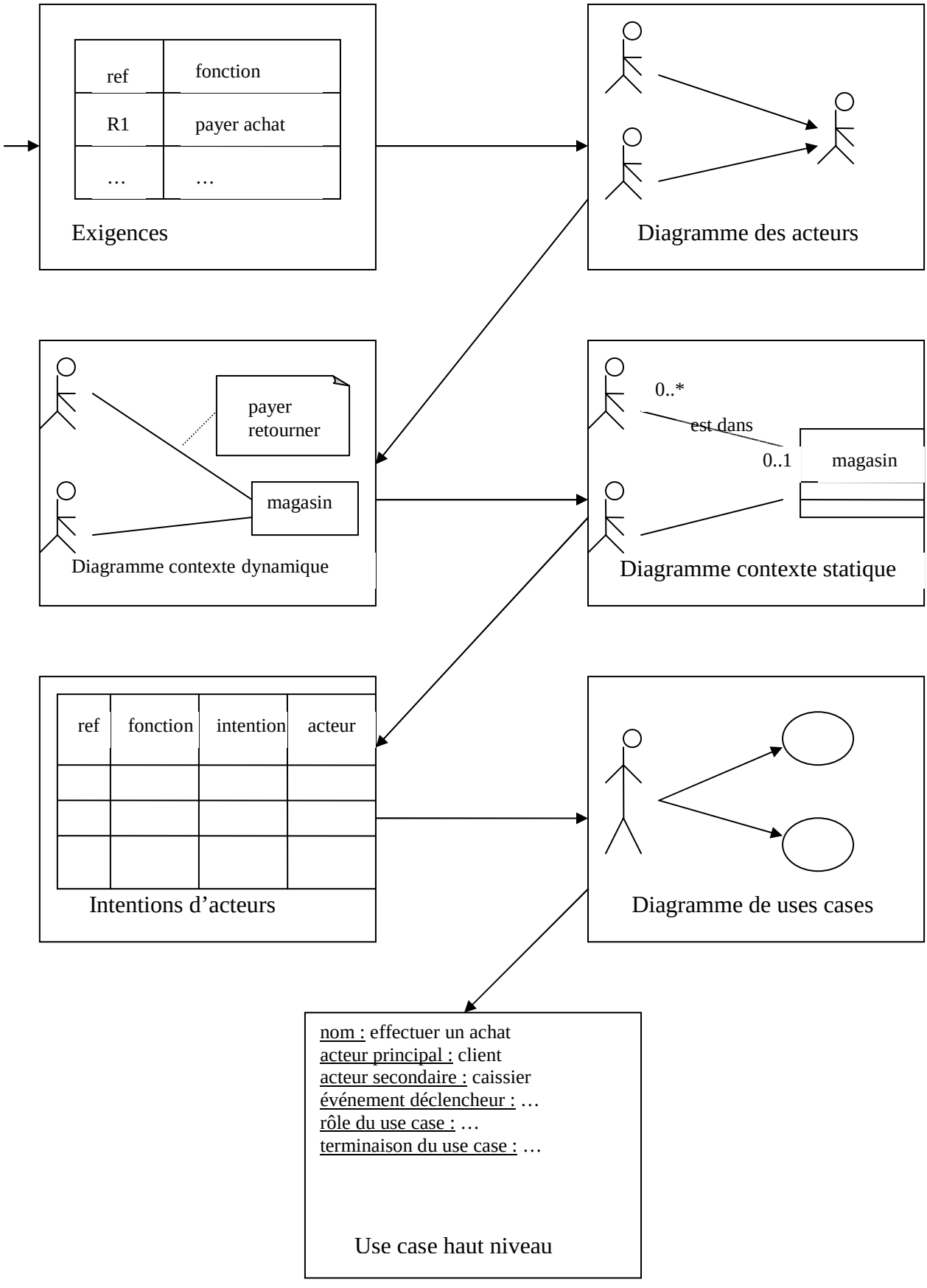


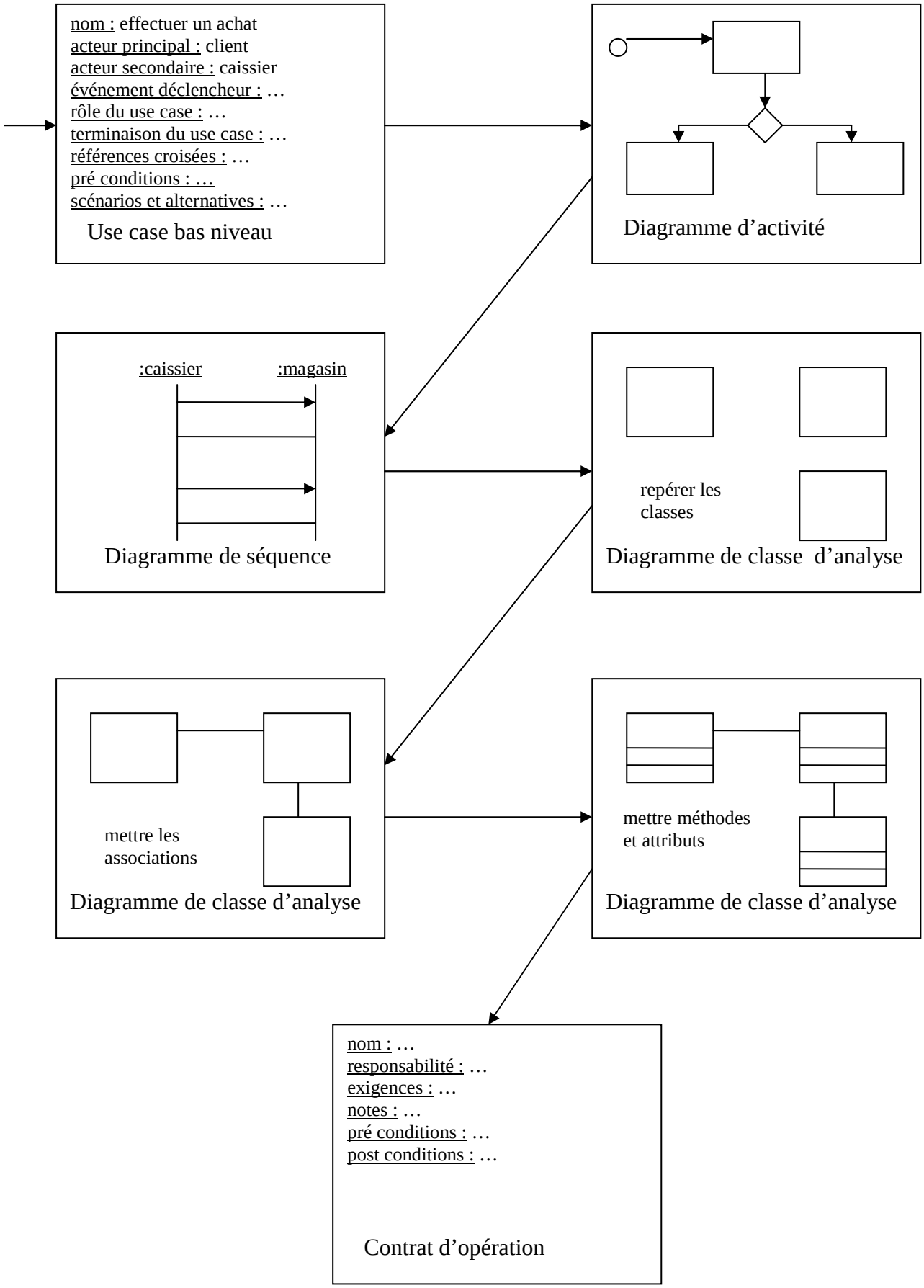
Diagramme de classe de conception

Processus détaillé

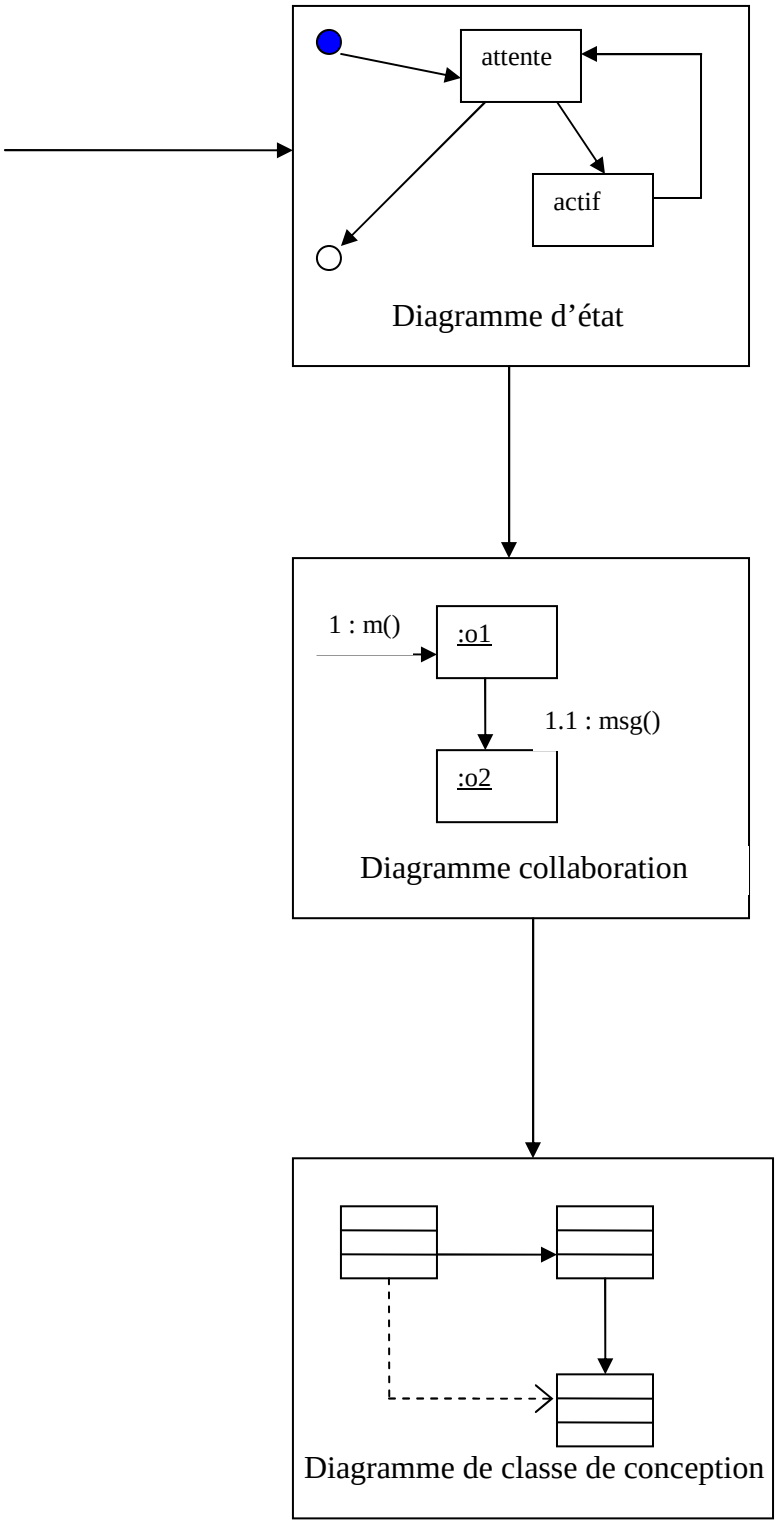
Processus de définition des besoins



Processus d'analyse



Processus de conception



Exemple traité

Nous allons traiter l'ensemble de la démarche d'analyse objet sur un même exemple. Cet exemple a été pris dans la littérature (voir la bibliographie). Nous ne traiterons pas l'ensemble de l'exemple, mais l'ensemble de la démarche.

Réalisation d'une caisse informatisée.

Un commerçant de produits touristiques (souvenirs, livres régionaux, ...) désire informatiser sa caisse. Chaque type de produit possède un code unique (étiquette à code à barres), et un même prix pour tous les produits de ce type. L'objectif est de faciliter la maintenance des prix des articles.

Chaque type de produit est référencé dans un catalogue, avec son prix associé. Quand le prix d'un produit doit être modifié, le manager modifie son prix dans le catalogue, puis sur l'étagère où il est rangé.

Le caissier s'identifie pour démarrer la caisse (avec mot de passe).

La caisse fera les fonctions habituelles d'une caisse : calcul du sous total, calcul du total, possibilité de rentrer plusieurs articles ayant un même code, retour d'une marchandise avec le ticket de caisse. Le paiement se fera en monnaie seulement.

La caisse permet d'éditer des rapports :
- Le reçu qui sera donné uniquement pour une vente effective. Il contient le nom du magasin, un message de bienvenue, la date et l'heure. Puis pour chaque vente il donne le code du produit, la description du produit, le prix unitaire, la quantité et le sous total. Enfin nous y trouvons le total TTC.
- Le rapport quotidien de l'ensemble des ventes (date, heure, total).
- Le rapport quotidien détaillé: liste de l'ensemble détaillé des ventes de la journée.

La caisse s'exécute sur un PC. Une douchette permettra de lire les codes à barres. Les informations peuvent être rentrées au clavier, ou à la souris.

Définition des besoins

Pour bien comprendre le fonctionnement du logiciel, et son interaction avec son environnement, nous avons intérêt à travailler non pas sur le logiciel demandé, mais sur le fonctionnement intégral du processus d'entreprise (workflow). Ainsi nous aurons une meilleure approche du logiciel. Ainsi nous considérerons l'ensemble des acteurs qui interagissent sur le processus d'entreprise, même s'ils n'agissent pas sur le logiciel lui-même. Cela permettra de déterminer les intentions d'acteurs qui nous donneront les uses cases.

La définition des besoins démarre avec le début du projet. Elle a pour but d'obtenir un premier jet des besoins fonctionnels et opérationnels du client. Dans cette phase nous collectons des informations pour définir le besoin du client.
A l'issue de cette étape, nous aurons mis textuellement ou à l'aide de schémas très simples, notre compréhension du problème à traiter sur le papier.

Le client doit être capable de valider l'étude ainsi faite. Elle lui est donc accessible.

première étape : les exigences et les acteurs

Les exigences que nous allons découvrir sont les exigences fonctionnelles. A partir du problème posé, c'est à dire de tout ce qui est écrit, plus tout ce qui ne l'est pas, nous allons lister l'ensemble des fonctions qui pourront être réalisées par le logiciel. Ces exigences seront numérotées pour pouvoir les tracer dans les intentions d'acteur puis dans les uses cases.

Référence	Fonction
R1	Modifier le prix d'un produit
R2	Calculer le total d'une vente
R3	Rentrer une quantité par article
R4	Calculer le sous total d'un produit
R5	Retourner une marchandise
R6	Payer l'achat
R7	Editer un ticket de vente
R8	Editer un rapport succinct
R9	Editer un rapport détaillé
R10	Se connecter à la caisse
R11	Se connecter en gérant
R12	Définir les droits des usagers
R13	Entrer un produit au catalogue
R14	Supprimer un produit du catalogue
R15	Enregistrer un produit à la caisse
R16	Initialiser la caisse

Dans la référence R veut dire Requirement (c’est à dire exigence en Anglais).

La modification du prix sur une étagère n'est pas traitée par le système. Donc ce n'est pas une exigence fonctionnelle pour notre logiciel.

Se connecter en gérant permet de modifier les prix des articles. Ce n'est pas permis pour un caissier.

Définir les droits des usagers n'est pas indiqué dans le texte, mais est nécessaire au bon fonctionnement du logiciel.

Le PC ainsi que la douchette sont des exigences d'architecture, elles ne seront pas prises en compte ici.

L'initialisation de la caisse est une fonctionnalité masquée.

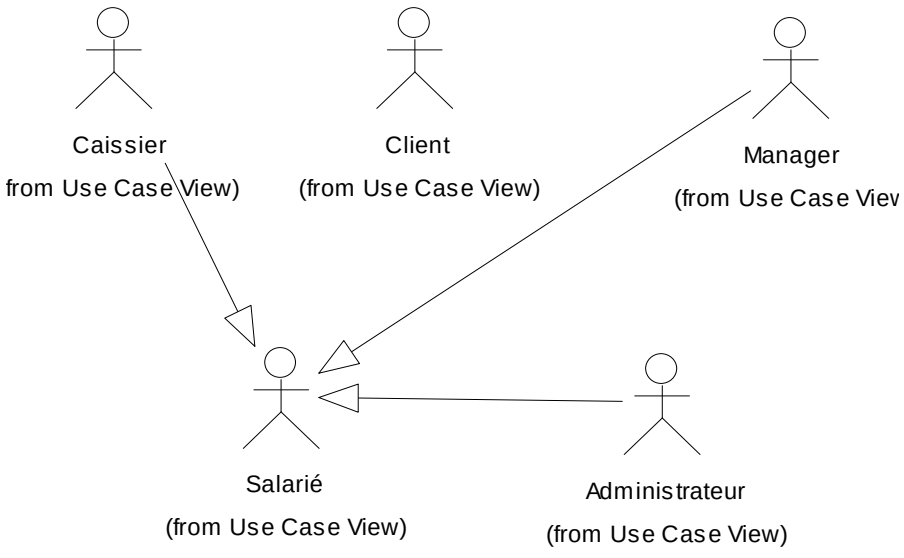
Entrer et supprimer un produit du catalogue sont des fonctions tellement évidentes que le client n'en parle pas. Elles doivent cependant être intégrées au logiciel.

Les informations rentrées au clavier ou à la souris sont des exigences d'interface qui seront prises en compte ultérieurement.

Notons que les exigences ne sont pas classées ici.

Regardons maintenant les acteurs qui gravitent autour de notre application.

Les acteurs seront vus dans le sens processus transversal de l'entreprise (workflow). Préférer un acteur humain plutôt que le dispositif électronique devant lequel il est. L'acteur humain a bien une intention quand il fait quelque chose. Un dispositif électronique beaucoup moins en général.



les acteurs de l'application

Ce diagramme est un diagramme de classe qui permet de lister les différents acteurs. Il se peut que notre liste ne soit pas complète, une itération suivante du processus nous permettra de compléter ce schéma.

Nous avons défini 5 acteurs. N'oublions pas qu'un acteur représente un rôle, et non pas une personne.

Le Manager peut être aussi l'Administrateur (notion de rôle).
Un Salarié est soit un Caissier, soit le Manager. Il se peut que dans la première itération l'analyste ne remarque pas ceci. C'est souvent dans une itération ultérieure que nous verrons les généralisations d'acteurs.

Nous connaissons les acteurs, et les exigences de l'application. Nous pouvons donc faire un diagramme de contexte dynamique. Ce diagramme de contexte qui définit les interactions entre les acteurs et le système (pas encore système informatique car nous en sommes au processus métier donc à découvrir les uses cases) sera réalisé en UML par un diagramme de collaboration. Il représente le fonctionnement de l'entreprise. Ici nous ne préciserons pas forcément l'ordre et le séquencement des opérations.

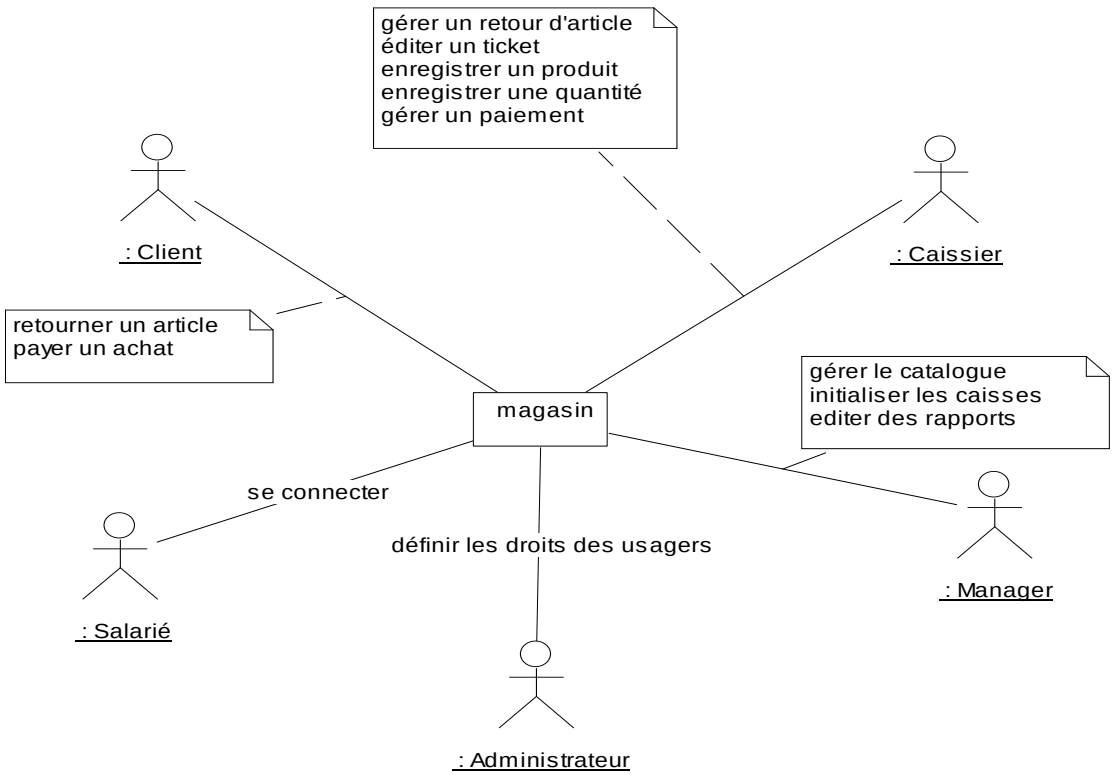


Diagramme de contexte du magasin

Nous pouvons aussi en complément tracer un diagramme de classe des acteurs du système pour préciser les cardinalités des différents acteurs du système. Nous utiliserons alors un diagramme de classes de UML. Ce diagramme s'appelle un diagramme de contexte statique.

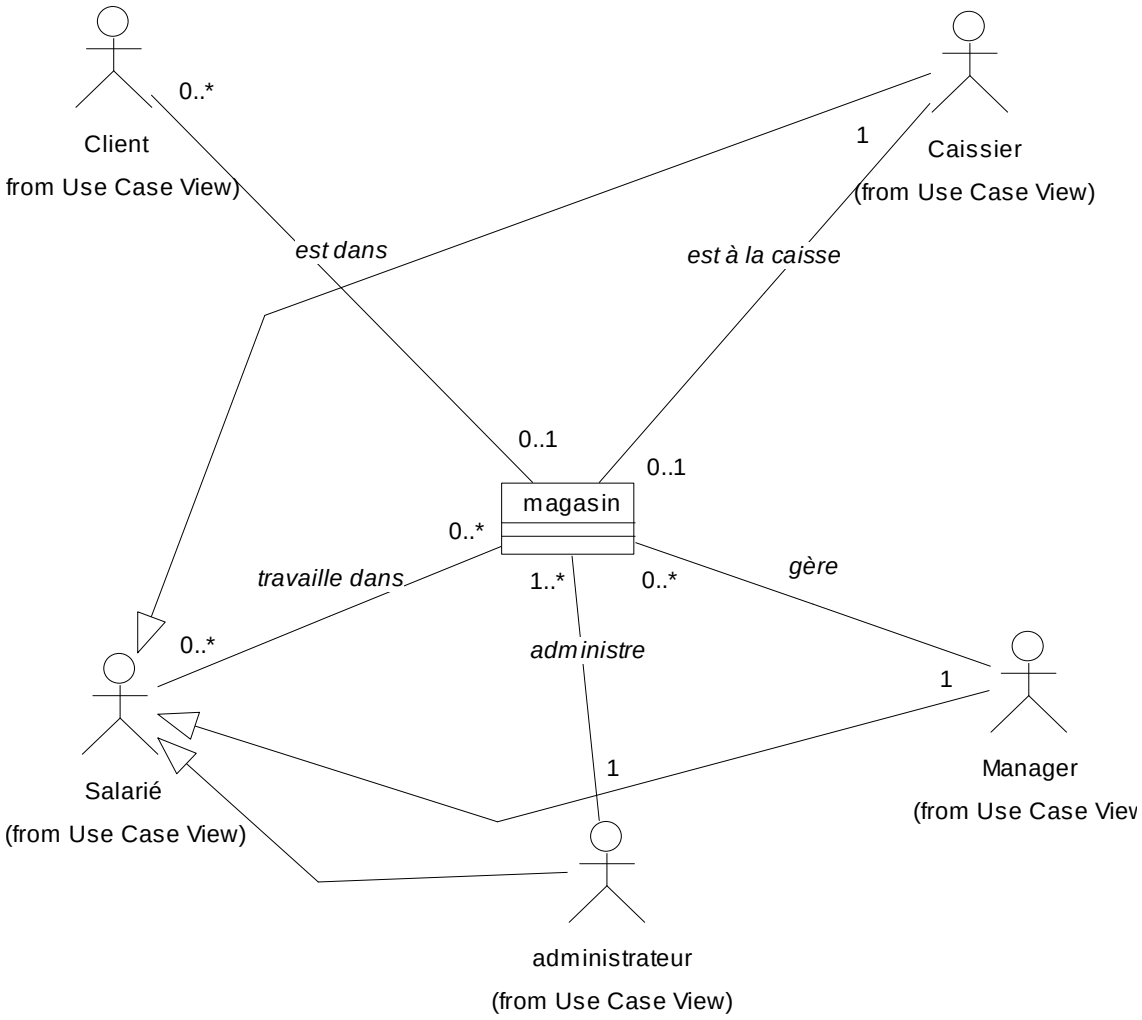


Diagramme de contexte statique (capture initiale des besoins)

deuxième étape : les intentions d'acteurs

Nous allons maintenant regrouper les exigences en intentions d'acteurs. Une intention d'acteur est un enchaînement de fonctions qui rend un service complet à un acteur (et pas une fraction de service).

Référence	Fonction	Intention d'acteur	Acteurs
R1	Modifier le prix d'un produit	Gérer le catalogue	Manager
R13	Entrer un produit au catalogue	Gérer le catalogue	
R14	Supprimer un produit du catalogue	Gérer le catalogue	
R16	Initialiser la caisse	Initialiser la caisse	Manager
R5	Retourner une marchandise	Retourner un article	Client, Caissier
R10	Se connecter à la caisse	Se connecter	Salarié
R11	Se connecter en gérant	Se connecter	
R8	Editer un rapport succinct	Editer un rapport	Manager
R9	Editer un rapport détaillé	Editer un rapport	
R12	Définir les droits des usagers	Définir les profils	Administrateur
R7	Editer un ticket de vente	Effectuer un achat	Client, Caissier
R6	Payer l'achat	Effectuer un achat	
R2	Calculer le total d'une vente	Effectuer un achat	
R3	Rentrer une quantité par article	Effectuer un achat	
R15	Enregistrer un produit à la caisse	Effectuer un achat	
R4	Calculer le sous total d'un produit	Effectuer un achat	

Ici dans le tableau la notion d'acteur repose sur l'intention d'acteur, pas sur la fonction. Ainsi toutes les lignes d'une même intention d'acteur ne sont-elles pas renseignées pour les acteurs.

Ici nous distinguerons l'acteur principal des acteurs secondaires en soulignant l'acteur qui est à l'origine de l'intention.

Nous allons bien sûr vérifier que les exigences définies précédemment sont toutes incluses dans les intentions d'acteur. La traçabilité est un facteur essentiel des phases de définition des besoins et de celle d'analyse.

Nous avons les intentions d'acteurs, nous pouvons donc faire un diagramme de Use Case.

Toute cette démarche peut être faite d'une autre manière:

- Nous déterminons les acteurs et les exigences
- A partir des exigences nous faisons un diagramme d'activité (UML) en colonnes (swim lane). Chaque colonne correspond à un acteur. Cela nous permet de définir les responsabilités des acteurs, donc de définir les uses cases correspondant.

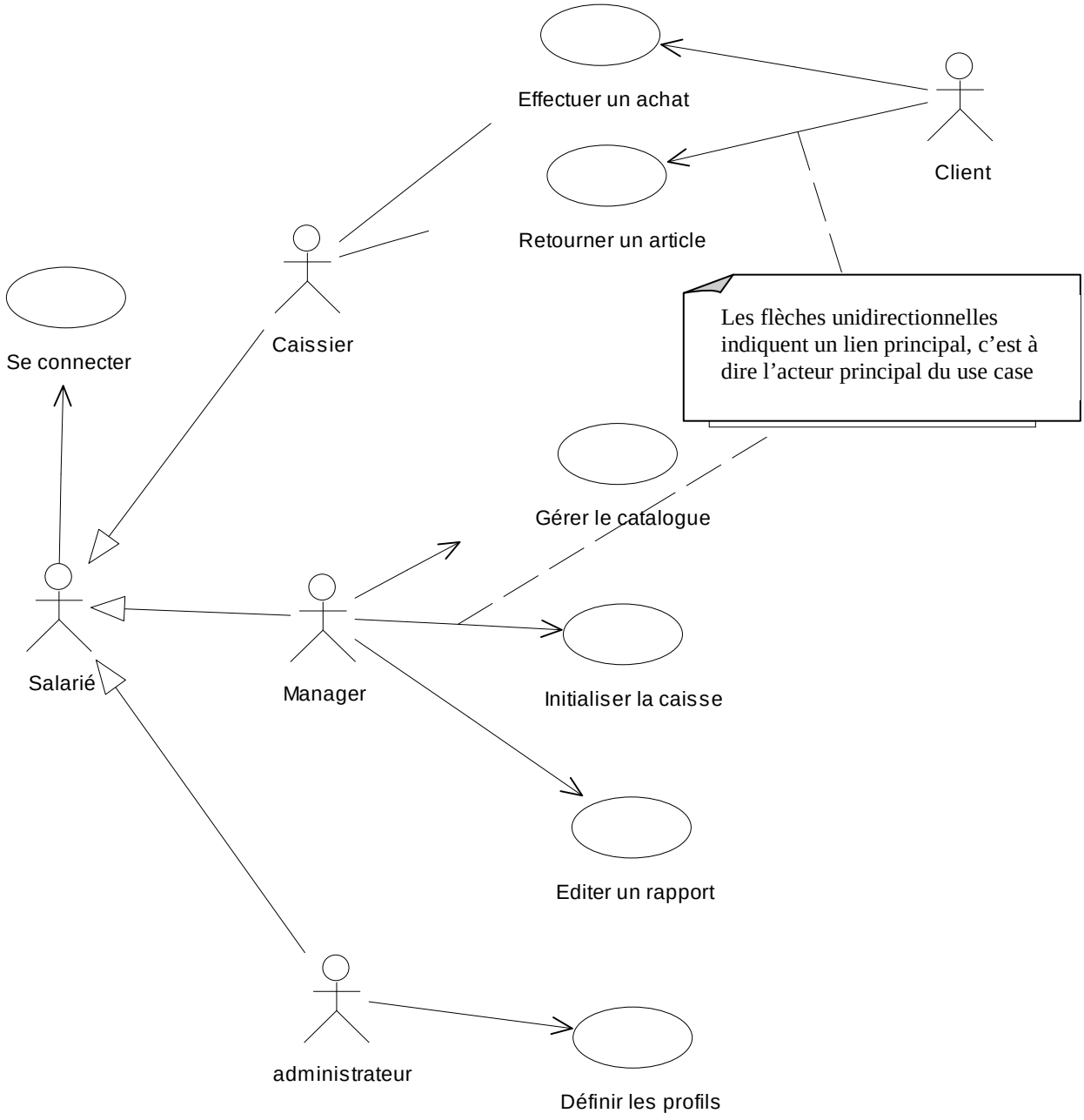
troisième étape : le diagramme de use case

Un use case est une intention d'acteur. Quand un acteur démarre un use case, il a une intention précise. Quelle est cette intention? Effectuer un achat est une intention qui recouvre un certain nombre d'exigences. C'est une unité d'intention complète d'un acteur: c'est donc un use case. Payer ses achats n'est pas une intention complète d'acteur. Nous n'allons pas dans un magasin pour payer, mais bien pour faire des achats. C'est donc une partie (et pas la plus agréable) de effectuer un achat. Ce n'est donc pas un use case.

L'acteur déclencheur de l'achat est le client. C'est lui qui est venu effectuer un achat.

Le diagramme de uses cases est la représentation graphique du tableau d'intention d'acteurs précédent.

Diagramme de use case

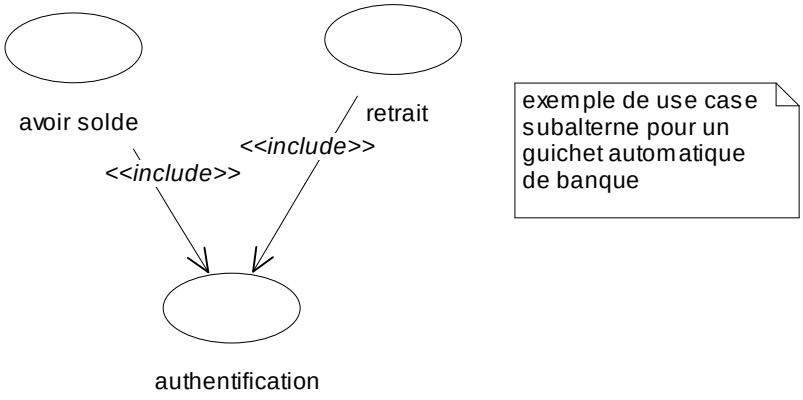


Quand nous définissons les uses cases, il faut faire extrêmement attention à la notion de point de vue. Pour le magasin le client est un acteur essentiel. Par rapport au système informatique ce n'est pas un acteur. L'acteur qui est en interface avec le système informatique est souvent le caissier, mais jamais le client.
Nous voulons définir le système informatique du magasin, pour cela il faut partir de l'ensemble des acteurs qui gravitent autour du magasin, sinon nous risquons d'oublier un certain nombre de fonctionnalités. Le fait de retourner un article est un souci du client, pas du caissier. Si nous ne prenons le point de vue que du système informatique nous risquons de ne pas penser à ce problème. Dans la phase d'analyse nous délimiterons le système informatique dans le processus global du magasin.

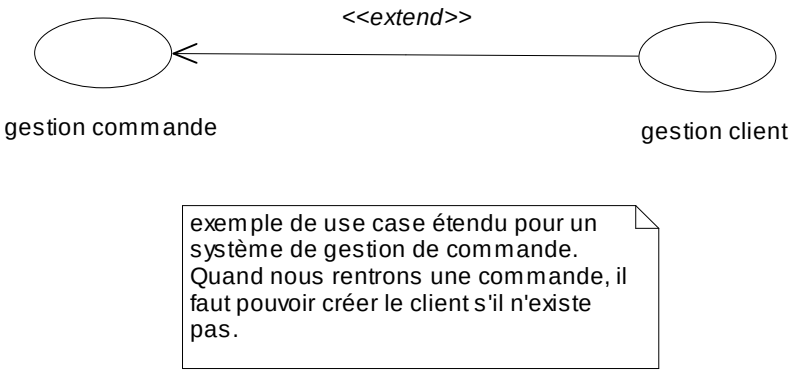
Le schéma de uses cases peut faire apparaître:

- Des fonctionnalités bien précises qui se retrouvent parmi plusieurs uses cases. Nous parlerons alors de use case included ou subalterne.

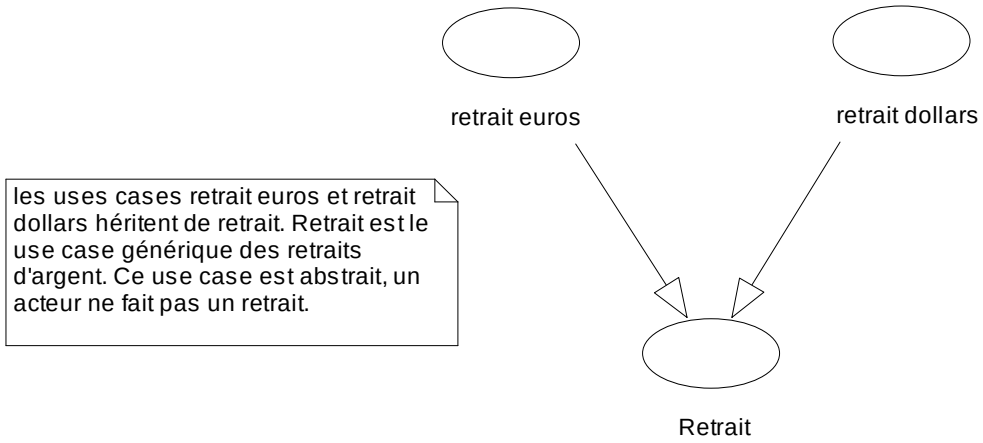
  Un tel use case ne peut être lié qu'à un use case, pas à un acteur. (nous mettrons alors le stéréotype include sur le lien use case vers use case subalterne).



- Un use case peut nécessiter le service d'un autre use case. Le use case dont nous avons besoin du service étend le use case demandeur. Nous utiliserons le stéréotype extend. Ici les uses cases peuvent être sollicités par des acteurs.



Nous pouvons avoir des cas de spécialisation de cas d'utilisation. Plusieurs uses cases peuvent avoir une trame commune et donc hériter d'un use case parent et abstrait.



Ces spécialisation, include et extend ne se mettent bien souvent pas dans le premier cycle de développement, car c'est l'étude de l'application qui mettra en évidence ces caractéristiques.

Nous verrons plus tard ces notions de cycle dans le déroulement d’une démarche comme celle-ci.

quatrième étape : use case description de haut niveau

Il nous faut maintenant définir les uses cases. Cette description est une description de haut niveau qui se fait avant l'analyse. Nous sommes toujours dans la définition du besoin.

Une description de haut niveau d'un use case est une description textuelle qui comprend les chapitre suivants:

Nom du Use Case:
Acteur principal:
Acteurs secondaires:
Evénement déclencheur du Use Case:
Rôle du Use Case:
Terminaison du Use Case:

Cette description est assez succincte 3 ou 4 lignes maximum pour le rôle du use case. C'est une description narrative d'un processus métier. Cette description ne repose pas sur la technologie objet.

Les cas d'utilisation (autre nom pour un use case) vus dans l'analyse sont dits essentiels. Ici nous ferons donc la description de cas d'utilisation de haut niveau essentiels.

Essentiel signifie que l'on ne fait pas référence à la technologie, que le use case décrit le cœur du métier, ils sont appelés use case pour 100 ans. De haut niveau veut dire que l'on en fait une description brève, sans entrer dans le détail du séquencement du use case.

Nous verrons par la suite des uses cases détaillés (en phase d'analyse), et des uses cases réels (en phase de conception).

Description d'un use case essentiel de haut niveau:

Nom du Use Case:	**Effectuer un achat**
Acteur principal:	Client
Acteurs secondaires:	Caissier
Evénement déclencheur du Use Case:	Un client arrive à la caisse avec ses achats
Rôle du Use Case:	Le caissier passe tous les articles, fait la note, et encaisse le paiement.
Terminaison du Use Case:	Le client a payé et a ramassé tous ses articles.

L'analyse

Après une première ébauche des besoins des clients, il va falloir approfondir ses besoins, examiner les différents scénarios des uses cases (éventuellement avec un diagramme d'activité pour exprimer ces différents scénarios), tenir compte des traitements exceptionnels et cas d'erreur. Il va être nécessaire de regarder le séquencement des opérations d'un point de vue fonctionnel, pour voir comment les différents acteurs interagissent avec le logiciel, à l'aide des diagrammes de séquence boîte noire (la boîte noire c'est le système informatique à développer). Il est alors nécessaire de distinguer les différents objets qui collaborent dans notre système informatique (avec un diagramme de classe). Enfin nous allons détailler le rôle des opérations en définissant des contrats d'opération (soit sous forme textuelle, soit sous forme de diagramme de séquence). Nous aurons alors tous les éléments pour passer à la conception.

Nous n'avons ici comme souci que de détailler les besoins du client pour s'en imprégner, afin de pouvoir le formaliser et le préciser.

cinquième étape : use case description de bas niveau

La description détaillée d'un use case permet de bien décrire les enchaînements des services rendus par le logiciel pour un usage précis. N'oublions pas que nous restons au niveau essentiel, c'est à dire que nous ne donnerons pas de solution au problème, nous ne faisons que préciser sa description. Nous sommes toujours dans l'espace du problème, pas dans celui de la solution.

Une description détaillée de use case comprend:

- Nom du Use Case:
- Acteur principal:
- Acteurs secondaires:
- Evénement déclencheur du Use Case:
- Rôle du Use Case:
- Terminaison du Use Case:
- Les références croisées des exigences:
- Les pré conditions nécessaires au fonctionnement du use case:
- Description complète du use case, avec les différents scénarios: Cette description intègre les cas d'erreur (traités par le use case) et les exceptions (forçant la sortie du use case).
- Contraintes d'IHM (optionnel): Attention de ne pas décrire l'interface ici, mais bien de préciser des éléments à prendre en compte lors de la réalisation des interfaces.
- Contraintes non fonctionnelles (optionnel): Ici nous prendrons en compte les dimensionnements, les performances attendues, … Ceci permettra de mieux évaluer les contraintes techniques.

La description complète du use case donne la priorité aux choses que les acteurs perçoivent. Nous décrirons les actions des acteurs en commençant par l'acteur principal qui initie le use case, puis nous donnerons les réponses du système (vu comme une boîte noire). Le premier travail se fait sur un cas nominal (c'est à dire idéal). Les cas d'erreur (avec correction et reprise) et les exceptions (avec abandon du use case) sont traités ensuite.
Le formalisme est textuel et prend la forme suivante:

Actions des acteurs	**Réponses du système**
1. L'acteur principal fait …	2. Le système lui envoie …
3. L'acteur principal calcule …	
4. L'acteur secondaire traite …	5. Le système envoie le compte rendu

Traitement des cas d'erreur et d'exception:
A l'étape 2 si le code de … alors le système renvoie un message et l'acteur principal refait …
A l'étape 4 si … alors le système affiche … et le use case s'arrête

En premier lieu nous voyons les échanges entre le système et les acteurs, ainsi que la chronologie et l'enchaînement des actions, dans le cas nominal.
Dans un deuxième temps nous listons les cas d'erreur (avec traitement de récupération) et les traitements d'exception avec arrêt du use case.
Nous restons bien fonctionnel car nous ne décrivons pas comment est réalisé le système, mais comment sont réalisés les services qu'il rend.

Notion de scénario: un use case est un regroupement d'actions plus élémentaires qui permet de traiter une intention d'acteur assez générale. Un scénario est une réalisation du use case, donc une "intention élémentaire". Par exemple pour une gestion de compte client dans une banque, nous avons un use case gérer les comptes. Nous avons plusieurs scénarios: créer un compte, consulter un compte, modifier un compte, …
Pour chacun des scénarios il va falloir faire une description détaillée de use case (avec traitement d'erreur et d'exception pour chacun). Un diagramme d'activité va permettre de montrer l'ensemble d'un use case, en regroupant l'ensemble des scénarios de ce use case.

Exemple de description d'un use case (effectuer un achat):

Nom du Use Case:	**Effectuer un achat**
Acteur principal:	Client
Acteurs secondaires:	Caissier
Evénement déclencheur du Use Case:	Un client arrive à la caisse avec ses achats
Rôle du Use Case:	Le caissier passe tous les articles, fait la note, et encaisse le paiement.
Terminaison du Use Case:	Le client a payé et a ramassé tous ses articles.
Les références croisées des exigences:	R2, R3, R4, R6, R7, R15

Les pré conditions nécessaires au fonctionnement du use case: La caisse est allumée et initialisée. Un caissier est connecté à la caisse.

Description complète du use case, avec les différents scénarios:

Ici il n'y a qu'un scénario possible. Nous verrons dans l'exemple suivant un use case avec plusieurs scénarios pour mettre en évidence le diagramme d'activité.

Actions des acteurs	**Réponses du système**
1) Le client arrive à la caisse avec les articles qu'il désire acheter.	
2) Le caissier enregistre chaque article.	3) Le système détermine le prix de l'article, les informations sur l'article et ajoute ces informations à la transaction en cours. Il affiche ces informations sur l'écran du client et du caissier.
4) Le caissier indique la fin de vente quand il n'y a plus d'article.	5) Le système calcule et affiche le montant total de l'achat.
6) Le caissier annonce au client le montant total.	
7) Le client donne au caissier une somme d'argent supérieure ou égale à la somme demandée.	
8) Le caissier enregistre la somme donnée par le client.	9) Le système calcule la somme à rendre au client.
10) Le caissier encaisse la somme remise par le client et rend la monnaie au client.	11) Le système enregistre la vente effectuée
	12) Le système édite le ticket de caisse
13) Le caissier donne le ticket de caisse au client	
14) le client s'en va avec les articles qu'il a achetés.	

Variantes du scénario nominal (celui qui se déroule quand tout se passe bien).

Paragraphe 2: il peut y avoir plusieurs articles d'un même produit. Le caissier enregistre le produit ainsi que le nombre d'articles.
Paragraphe 3: s'il y a plusieurs articles d'un même produit le système calcule le sous total.

Paragraphe 2: Le code produit peut être invalide. Le système affiche un message d'erreur.

Paragraphe 7: Le client n'a pas la somme d'argent suffisante. La vente est annulée. Remarque: Ceci est une exception qui termine anormalement le use case.

Paragraphe 8: Le caissier n'a pas la monnaie pour rendre au client. Il va faire la monnaie à la caisse principale. Remarque: ici nous avons fait apparaître un nouvel acteur: le caissier principal (ce sera probablement la même personne que le manager mais un acteur différent).

Paragraphe 12: le système ne peut pas éditer le ticket de caisse. Un message est affiché au caissier, et celui-ci change le rouleau de papier.

Paragraphe 14: remarquons que si le client repart sans ses articles, il est probable que le caissier les lui mettra de coté. Cet événement ne change rien à notre système futur. Ce genre d'incident ne sera pas pris en considération.

En terme de représentation nous pouvons préférer mettre une colonne par acteur pour bien voir les responsabilités des acteurs vis à vis du système. Ici cela donnerait:

Actions du client	**Actions du caissier**	**Réponses du système**
1) Le client arrive à la caisse avec les articles qu'il désire acheter.	2) Le caissier enregistre chaque article.	3) Le système détermine le prix de l'article, les informations sur l'article et ajoute ces informations à la transaction en cours. Il affiche ces informations sur l'écran du client et du caissier.
	4) Le caissier indique la fin de vente quand il n'y a plus d'article.	5) Le système calcule et affiche le montant total de l'achat.
	6) Le caissier annonce au client le montant total.	
7) Le client donne au caissier une somme d'argent supérieure ou égale à la somme demandée.	8) Le caissier enregistre la somme donnée par le client.	9) Le système calcule la somme à rendre au client.
	10) Le caissier encaisse la somme remise par le client et rend la monnaie au client.	11) Le système enregistre la vente effectuée
		12) Le système édite le ticket de caisse
	13) Le caissier donne le ticket de caisse au client	
14) le client s'en va avec les articles qu'il a achetés.		

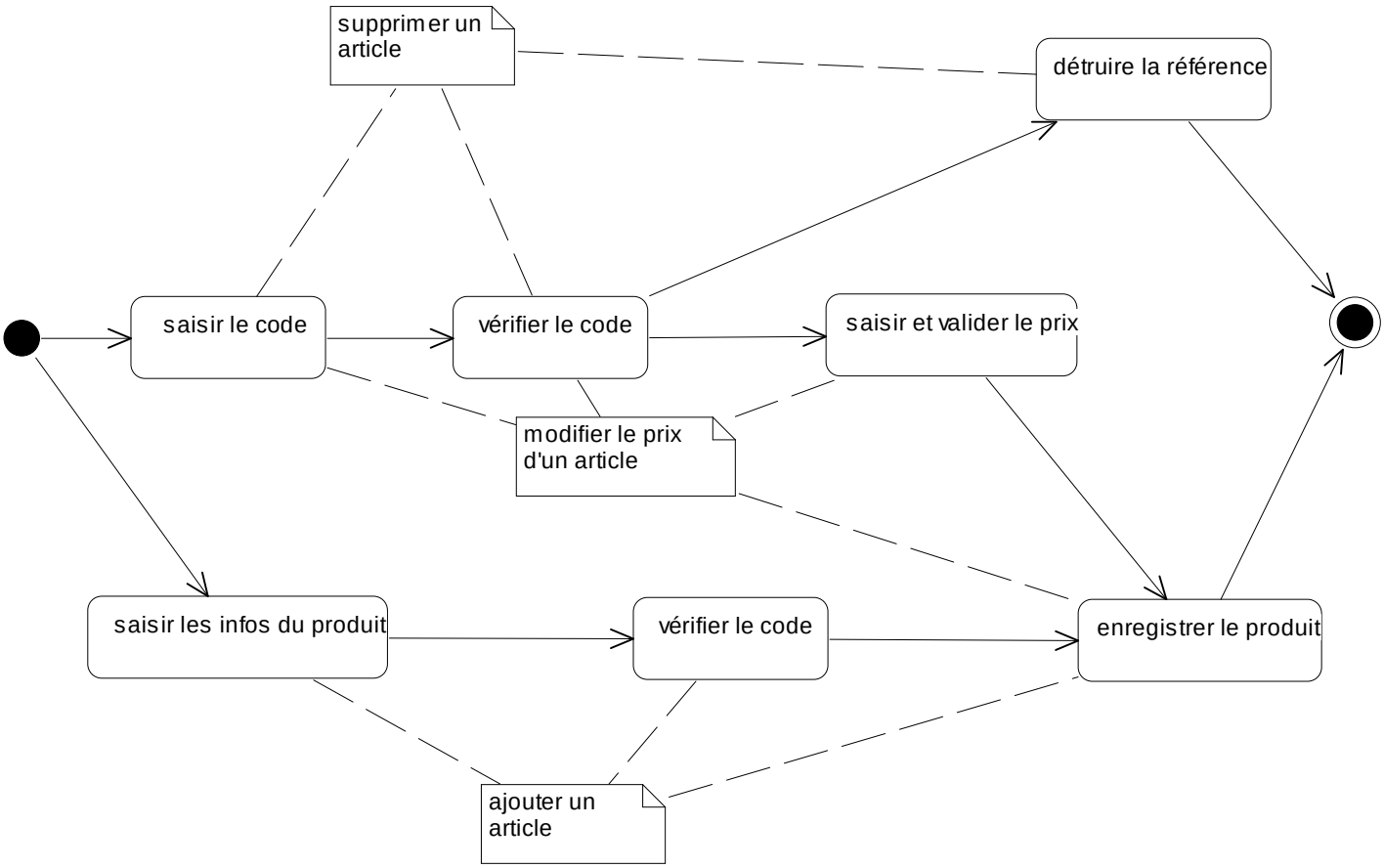
Cette représentation met en évidence que le client n'a pas accès au système informatique.

Contraintes d'IHM (optionnel): La désignation de l'article et son prix (éventuellement le sous total si plusieurs occurrences du même article) sont affichés à chaque article au caissier et au client. Le total est affiché également aux deux.

Contraintes non fonctionnelles (optionnel): Le magasin reçoit en moyenne 200 clients par jour. Chaque client achète en moyenne 10 articles.

Voici un exemple de diagramme d'activité représentant la réunion de scénarios modélisant un use case.

Nous allons gérer le catalogue en utilisant un diagramme d'activité pour représenter l'ensemble des scénarios (un scénario est un déroulement d'un use case de bout en bout, en commençant par le début et se terminant par la fin.)
Un scénario supporte aussi des variantes et des exceptions. Chaque scénario peut être décrit comme le use case ci-dessus. Mais il est bon de regrouper l'ensemble de ces scénarios dans un diagramme d'activité pour avoir une vue complète du use case.



Note : 

Ceci représente un commentaire dans tout schéma UML.

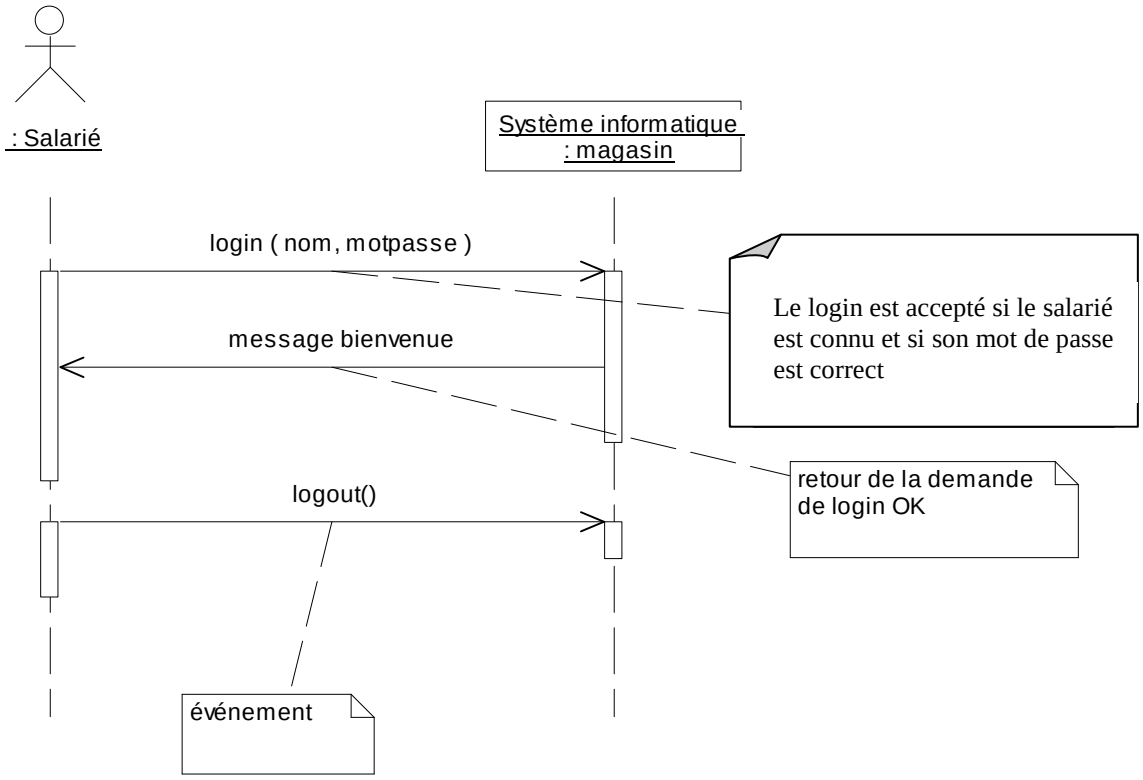
Les notes nous montrent les trois scénarios possibles pour ce use case. Pour chacun de ces scénarios il est nécessaire de faire le tableau précédent pour mieux détailler le use case dans le scénario particulier. En particulier il faut penser aux cas d'erreur et aux exceptions.

Remarquons qu'un certain nombre de symboles du diagramme d'activité ne sont pas disponibles comme l'alternative, et les conditions de cette alternative.

sixième étape : diagramme de séquence boîte noire

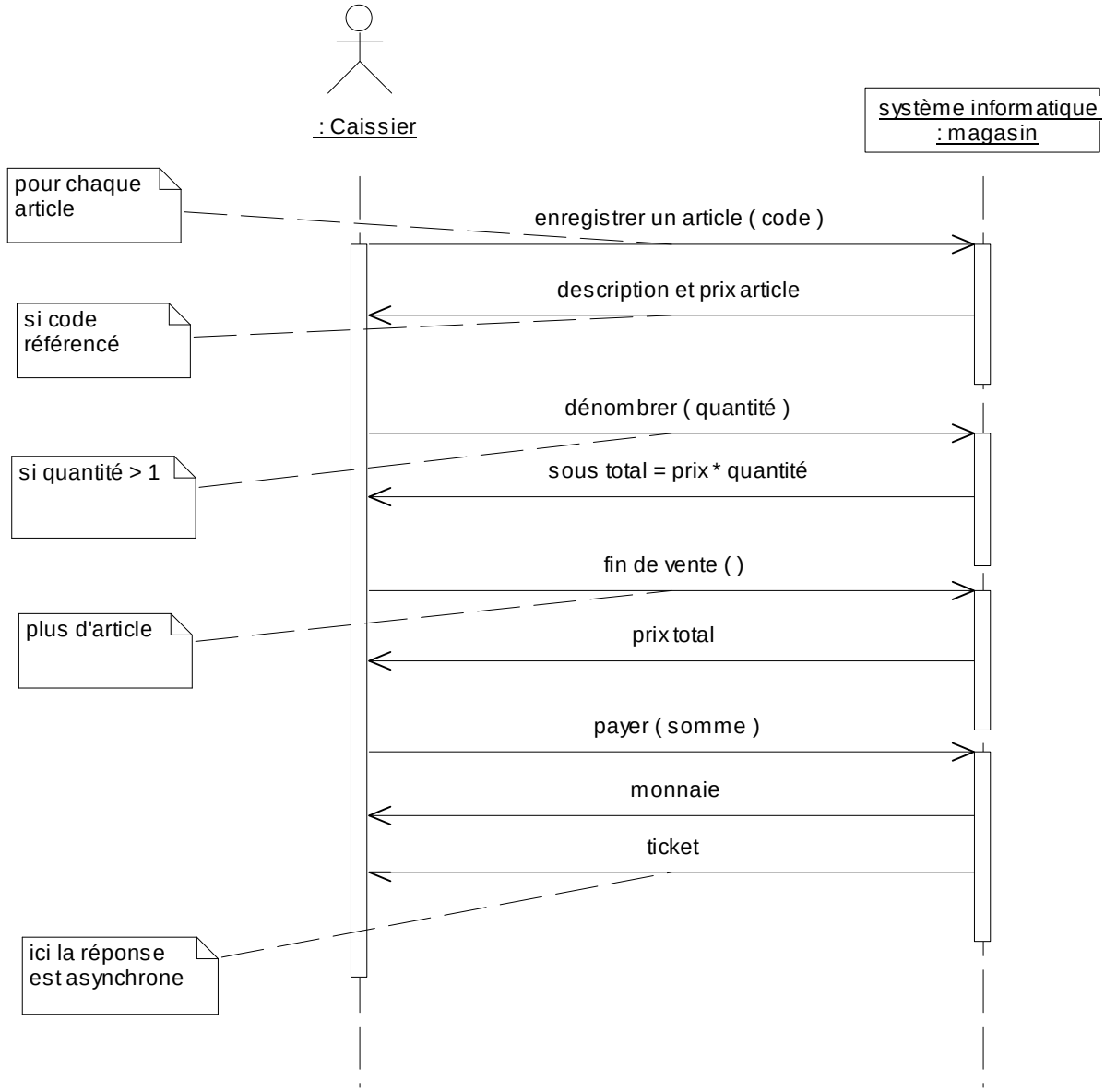
Les uses cases ont été validés par le client. Nous allons donc changer notre point de vue (workflow c'est à dire processus métier) pour adopter un point de vue système informatique. Nous abandonnons donc les événements métier pour nous intéresser aux événements informatiques.

Nous allons développer un diagramme de séquence par scénario de use case détaillé. Ces diagrammes de séquence sont dits boîte noire car nous ne savons toujours pas comment sera fait le système informatique. Ici nous précisons les échanges entre les acteurs utilisateurs du système informatique et le système proprement dit.
Les interactions des acteurs directement sur le système sont appelées des événements. Les actions engendrées par le système suite à ces événements sont appelées des opérations. A un événement correspondra une opération. Dans la terminologie message et événement sont équivalents entre eux, ainsi que méthode et opération.
Les diagrammes de séquence seront agrémentés de notes et commentaires qui permettront d'illustrer le diagramme: explication de contrôles, itérations, logique, choix, …
Rappelons qu'un diagramme de séquence boîte noire est de la forme:



Les valeurs de retour se formalisent de différentes manières suivant la version d'UML choisie. Ici j'ai pris un exemple de notation la plus simple possible. Attention à ne pas confondre avec un événement (sens acteur vers système informatique).
En fin de phase nous connaissons les échanges entre les utilisateurs et le système informatique. Nous pourrions construire une maquette des écrans pour les acteurs. Formellement nous le ferons en phase de conception. Notre maquette d'écran serait une maquette "fonctionnelle" (c'est à dire que le dessin de l'écran n'est pas figé).

Diagramme de séquence boîte noire du use case détaillé "effectuer des achats":



septième étape : diagramme de classe d'analyse

Le diagramme de classes est un des documents les plus importants, voire le plus important de l'analyse d'un logiciel par une méthode objet. C'est le premier document qui est dans le monde des objets (les acteurs n'étant pas forcément représentés dans le système informatique). C'est donc l'entrée dans le monde des objets.
Ce diagramme est un diagramme de classe d'analyse. Il ne représente pas les classes Java ou C++ que nous développerons par la suite.
Ici nous nous intéressons aux classes du domaine métier, c'est à dire des objets dont on parle dans le cahier des charges et dans les uses cases principalement. Nous ne nous intéressons pas aux objets informatiques dans ce diagramme (sauf bien entendu si le métier est l'informatique !!). Cela viendra en son temps, dans le diagramme de classe de conception. Quand nous passerons à la conception des classes disparaîtront, d'autres apparaîtront, dont les classes informatiques (collections, …). C'est normal. En phase d'analyse, nous voulons simplement faire un diagramme de classe d'objets manipulés dans le domaine métier considéré.
Dans ce diagramme de classe les opérations ne sont pas représentées (ce sera lors de la conception).

Ce diagramme montrera les concepts du domaine métier, les liens (associations) entre ces concepts (avec leur cardinalité ou multiplicité), et les attributs de ces concepts (souvent sous forme de données simples).
Le but de ce diagramme est de clarifier le domaine métier du client, et de se familiariser avec ce domaine.

Dans une analyse structurée nous décomposons l'analyse suivant les tâches ou les fonctions. Dans une application à dominante gestion, nous décomposerons notre application suivant les données, et les traitements. Ici nous analyserons la complexité du domaine en regardant les objets métier et leurs interactions entre eux.

Comment identifier les bons objets métier pour construire notre diagramme de classe d'analyse?
Il va falloir recenser dans les documents de définition des besoins et dans les documents d'analyse l'ensemble des objets métier.
Les noms ou groupes nominaux vont être de bons candidats pour être élus au titre de classe, objet ou attribut. Cependant, il faut être prudent car il y aura aussi un certain nombre de faux amis et de doublons.

Voici une démarche de sélection des classes candidates:
Par use case, nous réaliserons un diagramme de classe, le diagramme final étant la superposition de tous ces diagrammes.

Pour un use case nous recenserons les groupes nominaux rencontrés. Nous identifierons alors:
- Les classes (noms communs).
- Les objets (noms propres ou nom référencés).
- Les attributs (données simples qui qualifient une classe ou un objet).
- Les éléments qui sont étrangers à notre problème ou qui n'y ont pas de rôle réel.

- Les " classes fonctionnelles " qui ne sont porteuses d'aucune information, mais seulement de traitement, qui ne sont souvent pas des classes de notre diagramme. Par exemple gestionnaire du planning, décodeur de message, …

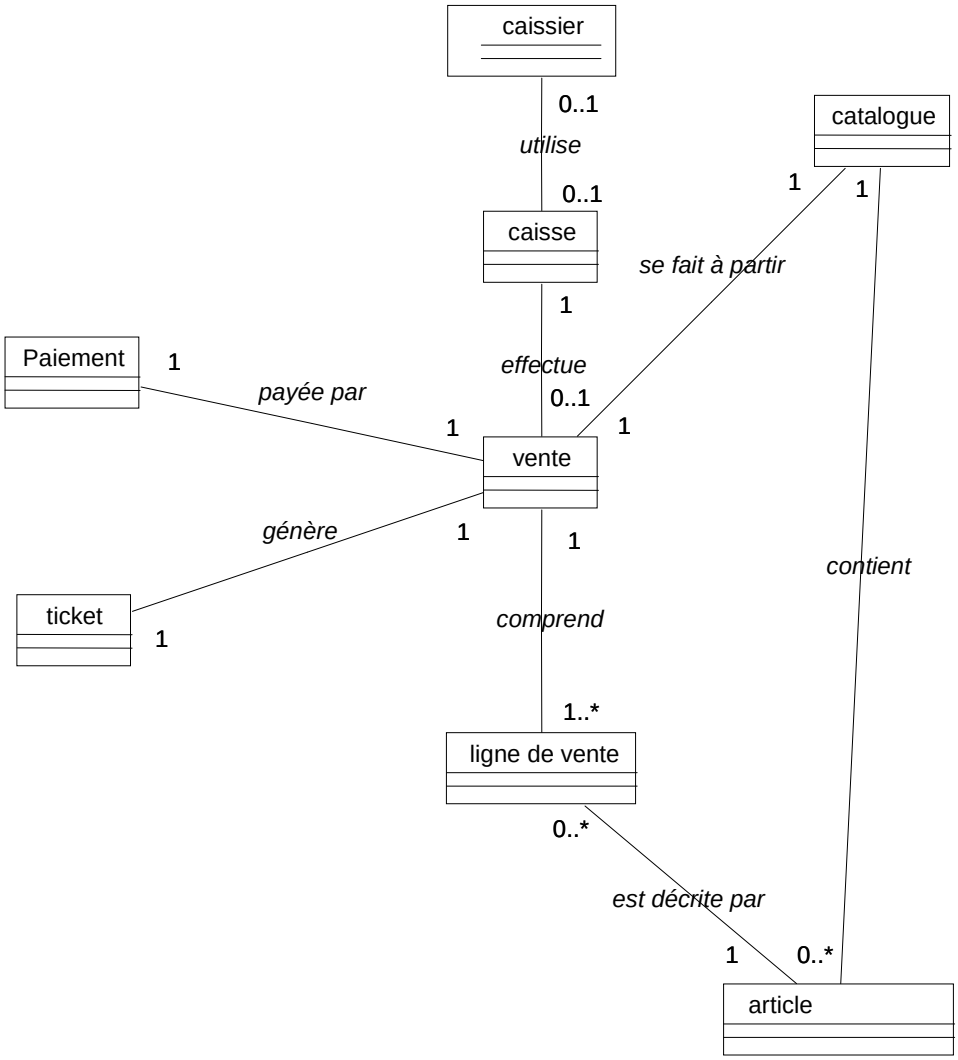
- Il est parfois difficile de savoir si une information est un attribut d'une classe ou une classe (avec une association avec la classe). Dans le doute il vaut mieux faire une classe, quitte à revenir en arrière dans une itération ultérieure. Par exemple pour la classe personne l'âge est un attribut (donnée simple) l'adresse étant à priori une donnée complexe sera plutôt une autre classe associée à la personne.

- Les entités suivantes peuvent prétendre à devenir des classes :
  - les objets physiques (produit…),
  - les transactions (réservation, commande,….),
  - les lignes de transaction (ligne de commande…),
  - les conteneurs (catalogues, lots,…),
  - les organisations (services, départements,….),
  - les acteurs ou les rôles (client, fournisseur….),
  - les regroupements en abstraction (modèle, genre, type, catégorie….),
  - les évènements (vol,…)

Pour le use case " effectuer un achat " voici la liste des classes sélectionnées:

Nous ne gèrerons pas les exemplaires des articles. Nous appèlerons article un ensemble d'exemplaires identifiés par le même code barre, et comportant des informations identiques (prix …).

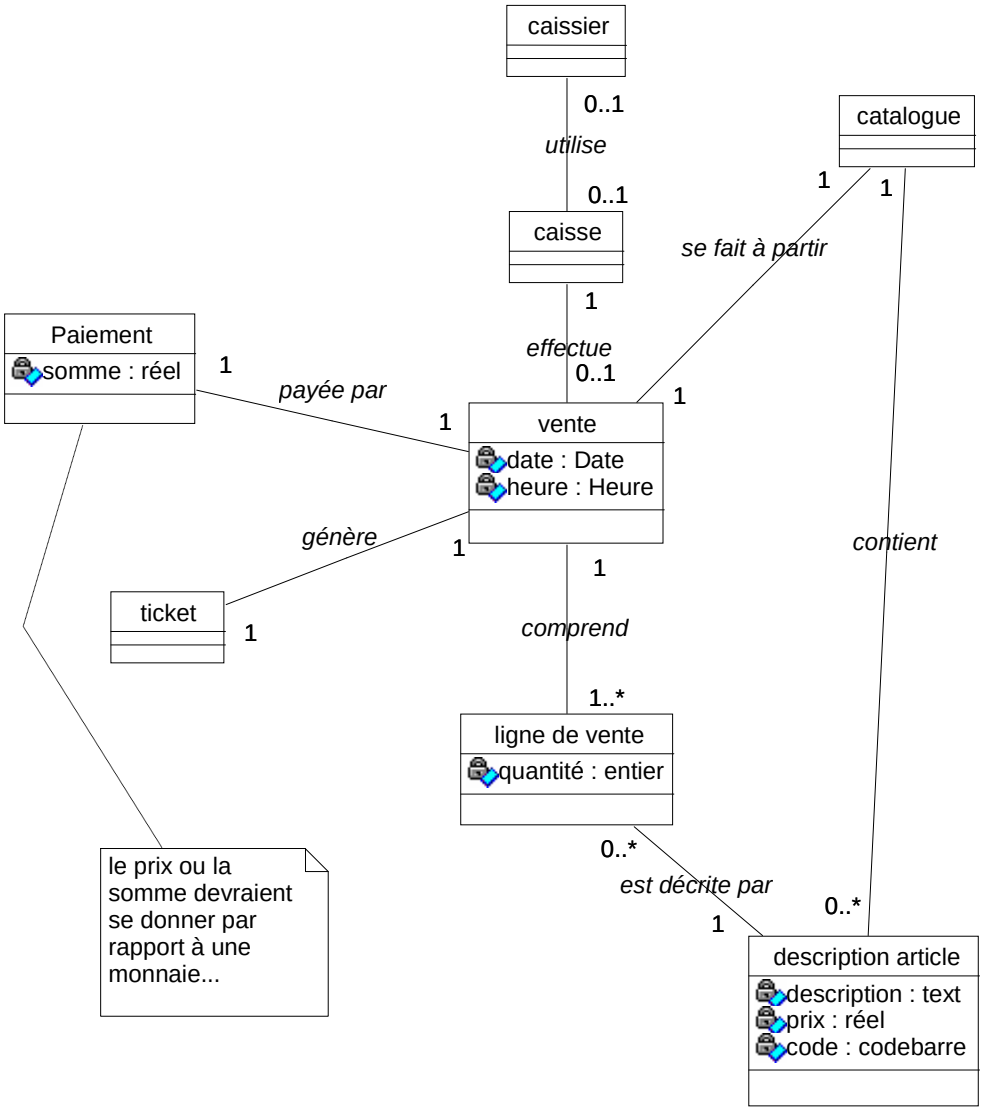
Nous allons maintenant rajouter les associations entre les classes, en donnant un intitulé et des cardinalités à ces associations.
Notre but n'est pas de mettre à jour toutes les associations entre les différentes classes, mais de noter les associations pertinentes pour comprendre le problème. Il faut privilégier les associations qui durent dans le temps, et dont la connaissance est nécessaire à la compréhension du problème.
Il faut éviter les associations redondantes ou dérivables d'autres associations.
L'établissement de ces associations nous amène à poser des questions au client. C'est un des buts recherchés. Voici une possibilité de diagramme de classe avec ses associations.



Sur ce diagramme de classe d'analyse il faut maintenant mettre les attributs des classes qui ont été repérés dans le cahier des charges, le diagramme de use case, ou dans le diagramme de séquence boîte noire du use case.

A priori, les attributs présents dans notre diagramme de classe d'analyse sont des attributs simples. On pourra faire exception des données pures qui sont des regroupements de données simples (ici code, adresse, prix) .
Les attributs qui ont un comportement sont des classes associées à notre classe. Ceci ne présage en rien du diagramme de classe de conception. Nous verrons ultérieurement ce que deviennent les associations dans ce schéma…
Dans les attributs, il ne doit pas y avoir de clef sur un autre objet. Cela se représente par une association.

Voici notre diagramme de classe enrichi des attributs d'analyse.



huitième étape : Le glossaire

Le glossaire répertorie tous les termes et leur définition. Il n'y a pas de graphisme particulier pour ce document. Il peut être fait sous la forme suivante:

terme	catégorie	description
Effectuer un achat	Use case	C'est le processus que suit un client pour effectuer ses achats dans le magasin.
Quantité	Attribut	C'est un attribut de la classe ligne de vente qui donne le nombre d'occurrences d'un même article lors d'un achat.
Vente	Classe	C'est la classe qui représente une vente en cours ou quand elle est finie.
…		

neuvième étape : les contrats d'opération

Nous allons maintenant étudier ce qui est fait dans le système, pour chaque flèche qui attaque le système dans les diagrammes de séquence.
Notre but est de comprendre la logique de fonctionnement du système. Les flèches représentées dans les diagrammes de séquence boîte noire déclenchent des opérations sur le système. Nous allons donc définir précisément le rôle de ces opérations en nous appuyant sur le diagramme de classe (appelé aussi modèle de domaine). C'est ce qui s'appelle des contrats d'opération.

Le système informatique a été vu comme une boîte noire (en particulier à travers les diagrammes de séquence). Il serait, d'un point de vue fonctionnel, intéressant de décrire ce que fait le système, en se basant sur le diagramme de classe, sans pour autant décrire comment il le fait. Les contrats d'opération vont nous permettre de décrire les changements d'états du système (c'est à dire les changements sur les objets et leurs associations) quand les opérations (issues des diagrammes de séquence) sont invoquées par les acteurs.

Un contrat d'opération est un document textuel qui décrit les changements provoqués par une opération sur le système. Le contrat d'opération est écrit sous forme textuelle et comporte des pré conditions et des post conditions. Les pré conditions décrivent l'état du système nécessaire pour que l'opération puisse s'effectuer. Les post conditions décrivent l'état du système lorsque l'opération s'est achevée.

Contrat d'opération type:

- Nom: Nom de l'opération avec ses paramètres.
- Responsabilités: Description du rôle de l'opération.
- Exigences: Liste des exigences dont le use case tient compte dans cette itération.
- Notes: Remarques diverses.
- Pré conditions: Etat nécessaire du système pour que l'opération se fasse.
- Post conditions: Etat du système lorsque l'opération s'est déroulée entièrement.

Les Pré conditions décrivent l'état du système, c'est-à-dire l'état des objets du système comme décrit dans le diagramme des classes d'analyse.
Nous jouons l'opération sur le système, en aveugle, et jusqu'à sa fin.
Notons les changements de l'état du système (des objets et de leurs associations) et nous obtenons les post conditions. Ces post conditions sont décrites sous la forme "has been", c'est-à-dire l'objet X **a été** modifié, son attribut Y **a été** mis à vrai.

Les post conditions portent sur 6 clauses:
- Les objets créés.
- Les objets détruits.
- Les associations créées.
- Les associations détruites.
- Les attributs modifiés.
- Les événements d'affichage (fonctionnels bien sûr !!!).

Prenons un exemple simple pour traiter ces contrats d'opération.

Modifier le prix d'un article au catalogue.

- Nom: modifier (cecode, nouveauprix).
- Responsabilités: modifier le prix d'un article de code donné, présent dans le catalogue.
- Exigences: R1, R13, R14 pour le use case gérer le catalogue.
- Notes: Si le code ne correspond pas a un article du catalogue, un message d'erreur sera envoyé au manager et l'opération s'arrête.
- Pré conditions: Il y a un article correspondant au code donné.
- Post conditions: l'attribut prix de l'objet description article, dont le code est cecode, a été changé par nouveauprix.