Initiation à la démarche d'analyse des traitements d'un Système d'Information avec les cas d'utilisation

La caisse informatisée

Version juin 2009

Jean-Pierre PHILIPPE

Sommaire

Objectif global d'une démarche d'analyse

Préambule

La démarche présentée ici est largement inspirée de l'ouvrage de Craig LARMAN :

UML et les Design Patterns chez CampusPress.

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 en effet un des soucis principaux pour pouvoir réduire le coût des logiciels et augmenter la puissance de développement des programmeurs.

Dans l'analyse et la conception objet nous cherchons à construire une architecture en couche de la forme suivante :

couche I.H.M.	présentation	écrans
couche application (workflow)	application	algorithme
couche objets métiers	métier	domaine fonctionnel
couche accès aux données	accès aux données	Data Access Layer (DAL)
couche base de données	base de données	SQL

Nous verrons que chaque couche définira un ou des contrôleurs pour la rendre indépendante des autres.

Cette architecture ne s'appliquera bien sûr pas à toutes les applications, c'est à prendre comme un exemple complet.

Exemple traité

Nous allons traiter l'ensemble de la démarche d'analyse objet sur un même exemple. Nous ne traiterons pas l'ensemble de l'exemple, mais l'ensemble de la démarche.

Réalisation d'une caisse informatisée : cahier des charges

Un commerçant de produits touristiques (souvenirs, livres régionaux, ...) désire informatiser sa caisse. Chaque type de produit possède un code unique (étiquette à code 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, éditer un ticket qui sera donné uniquement pour une vente effective. Ce ticket mentionne le nom du magasin, un message de bienvenue, la date et l'heure; puis pour chaque vente il donne le code du produit, la désignation du produit, le prix unitaire, la quantité et le sous total; enfin nous y trouvons le total TTC.
Le paiement se fera en monnaie seulement.

La caisse permet de gérer le retour d'une marchandise avec le ticket de caisse.

La caisse permet d'éditer des rapports :
- Le rapport quotidien synthétique 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

La définition des besoins démarre avec le début du projet. Elle a pour but de définir les besoins fonctionnels et opérationnels du client. Dans cette phase nous collectons des informations.

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).

A l'issue de cette phase, 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 ou fonctions

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.

Les acteurs

Regardons maintenant les acteurs qui gravitent autour de notre application.

Un acteur représente l'abstraction d'un rôle joué par des entités externes qui interagissent directement avec le système étudié.

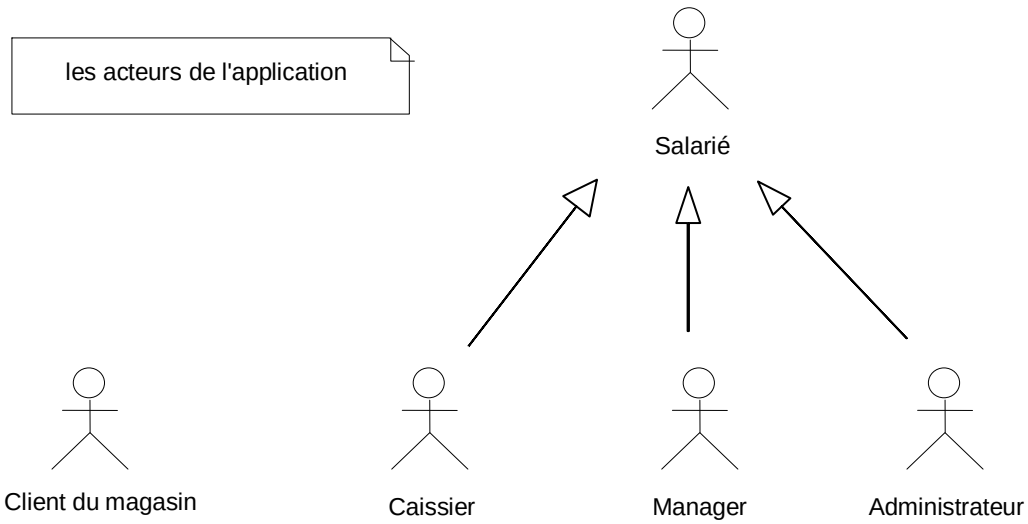
Ce sera le plus souvent un utilisateur, mais cela peut être un dispositif matériel ou un autre système.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages éventuellement porteurs de données.

Les acteurs candidats sont systématiquement ;

- les utilisateurs humains directs (ne pas oublier l'administrateur, l'opérateur de maintenance, etc.)
- les autres systèmes connexes qui interagissent directement avec le système.

Dans notre application, nous pouvons identifier les acteurs suivants :



Ce diagramme, qui utilise le formalisme d'un diagramme de classe, 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.

N'oublions pas qu'un acteur représente un rôle, et non pas une personne. Ici, le Manager peut être aussi l'Administrateur (notion de rôle).

Nous avons représenté un acteur "général" Salarié qui est soit le Caissier, soit le Manager, soit l'administrateur. 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.

Deuxième étape : les messages et le contexte

Un message représente la spécification d'une communication entre objets qui transporte de l'information avec l'intention de déclencher une activité chez le récepteur.
La réception d'un message est normalement considérée comme un événement.

Cette notion de message est tout à fait applicable pour décrire les interactions de plus haut niveau entre les acteurs et le système.

- Pour chaque acteur, demandez-vous quels sont les *messages* qui déclenchent un comportement du système attendu par l'acteur dans le cadre de son activité.
- Pour le système, demandez-vous quels sont les *messages* émis à l'intention d'un acteur particulier, et qui porte une information utilisée par ce destinataire.

Tous les messages identifiés peuvent être représentés de façon synthétique sur un **diagramme de contexte dynamique.**

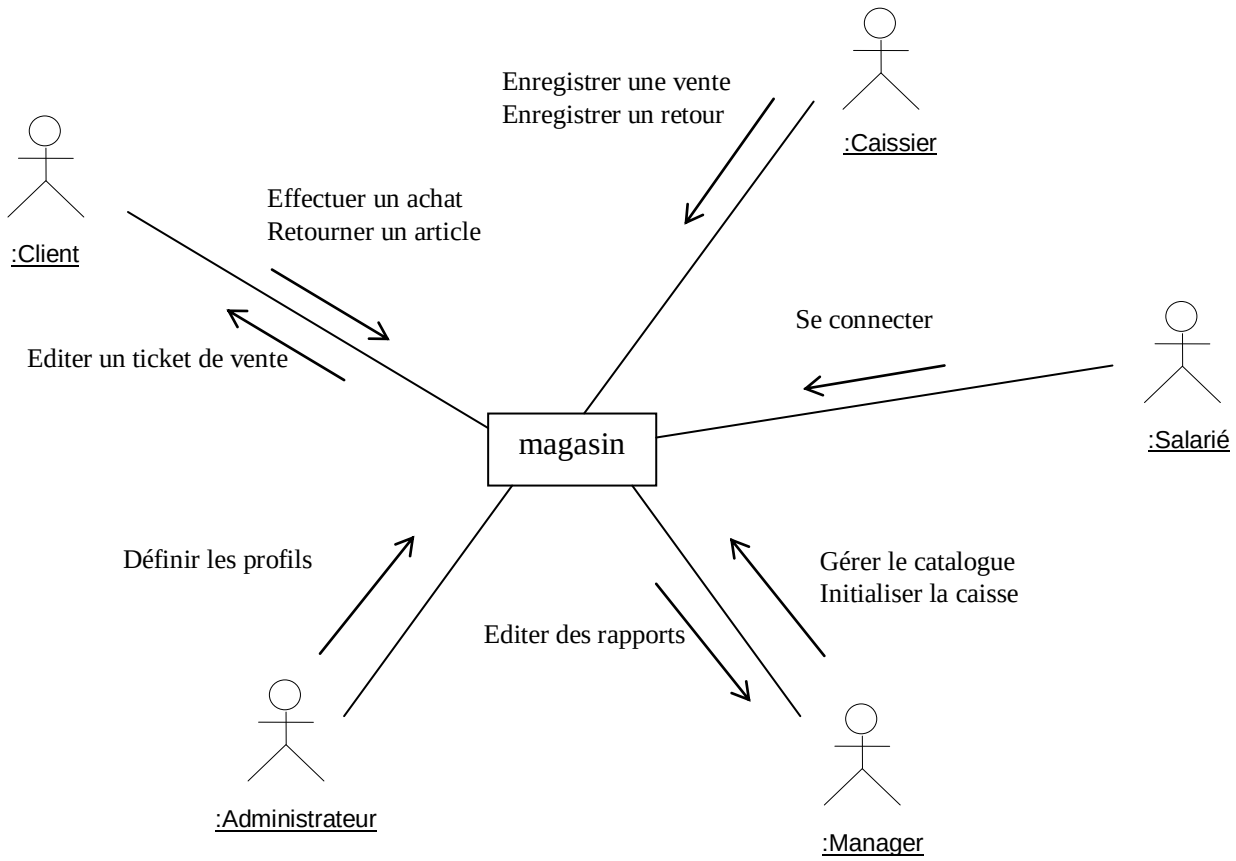


Diagramme de contexte dynamique

On utilise le formalisme d'un diagramme de collaboration :

- le système étudié est représenté par un objet central
- cet objet central est entouré par d'autres objets symbolisant les différents acteurs
- des liens relient le système à chacun des acteurs
- sur chaque lien sont montrés les messages en entrée et en sortie (les messages ne sont pas numérotés et ne précisent donc pas l'ordre des opérations).

On peut donner une description plus détaillée de chaque message sous forme textuelle et/ou en listant l'ensemble des fonctions qui devront être réalisées par le logiciel.

Acteur	**Messages (use case)**	Fonctions
Manager	Gérer le catalogue	Modifier le prix d'un article Ajouter un article au catalogue Supprimer un article du catalogue
Client	Effectuer un achat	Poser ses articles à la caisse Payer
Caissier	Enregistrer une vente	Enregistrer un produit à la caisse Enregistrer une quantité par article Calculer le sous total d'un produit Calculer le total d'une vente Editer un ticket de vente
Client	Retourner un article	Donner le ticket de caisse
Caissier	Enregistrer un retour	Enregistrer un produit Enregistrer une quantité par article Calculer le montant de l'avoir
Manager	Editer des rapports	Editer un rapport synthétique Editer un rapport détaillé
Manager	Initialiser la caisse	Initialiser la caisse
Salarié	Se connecter	Se connecter en caissier Se connecter en manager
Administrateur	Définir les profils	Définir les droits des usagers

Tableau des intentions des acteurs

Remarques :

- La modification du prix sur une étagère n'est pas traitée par le système.
- Se connecter en tant que Manager 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.
- L'initialisation de la caisse est une fonctionnalité masquée.
- Ajouter et supprimer un article du catalogue sont des fonctions tellement évidentes que l'utilisateur n'en parle pas. Elles doivent cependant être intégrées au logiciel.

Le diagramme de contexte statique

On peut compléter le modèle de contexte dynamique par l'étude du contexte statique.

Ce dernier spécifie le nombre d'instances d'acteurs reliées au système à un moment donné.

Le diagramme de contexte statique peut être dessiné au moyen d'un diagramme de classes ne faisant intervenir que les acteurs et le système.

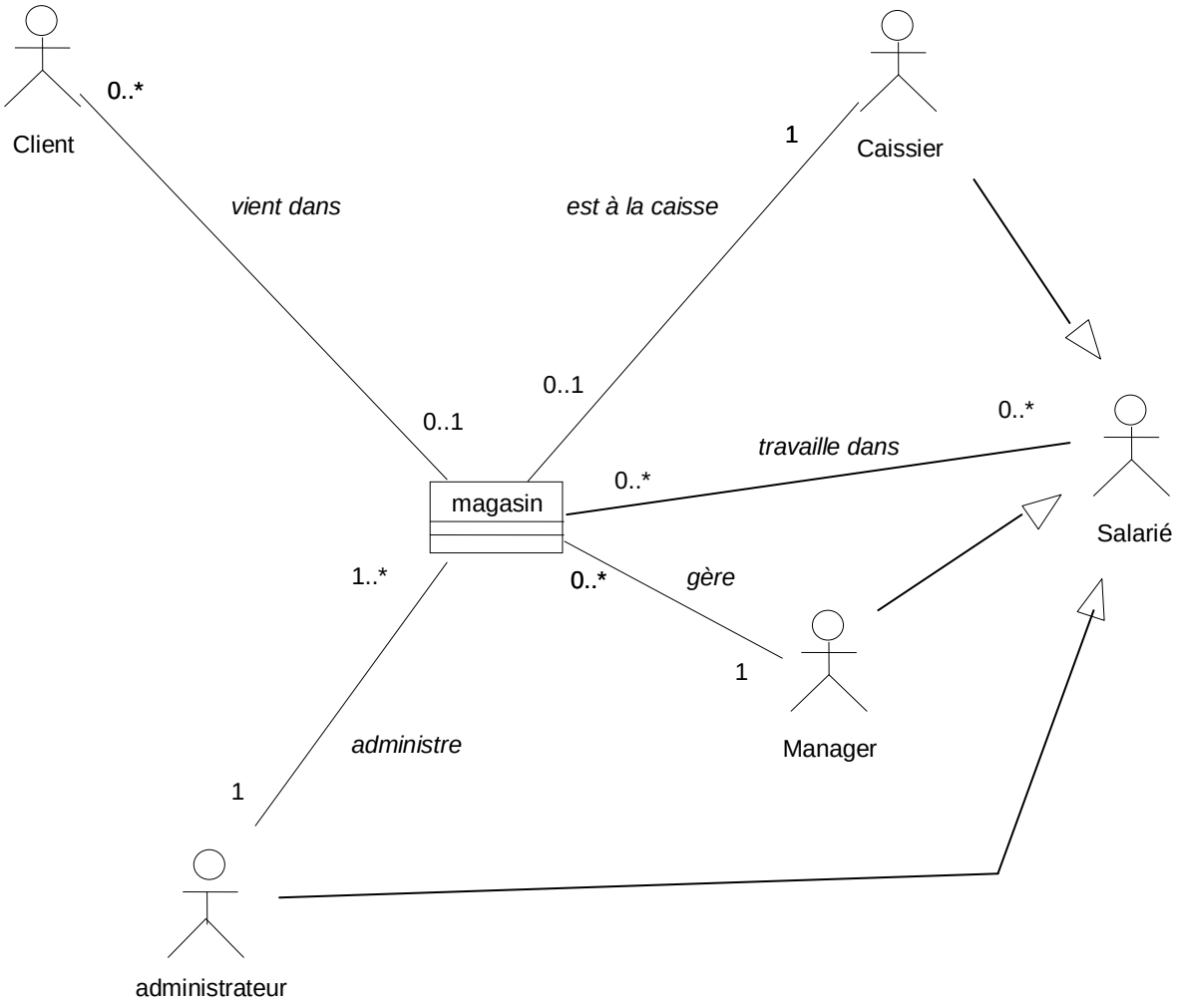


Diagramme de contexte statique (capture initiale des besoins)

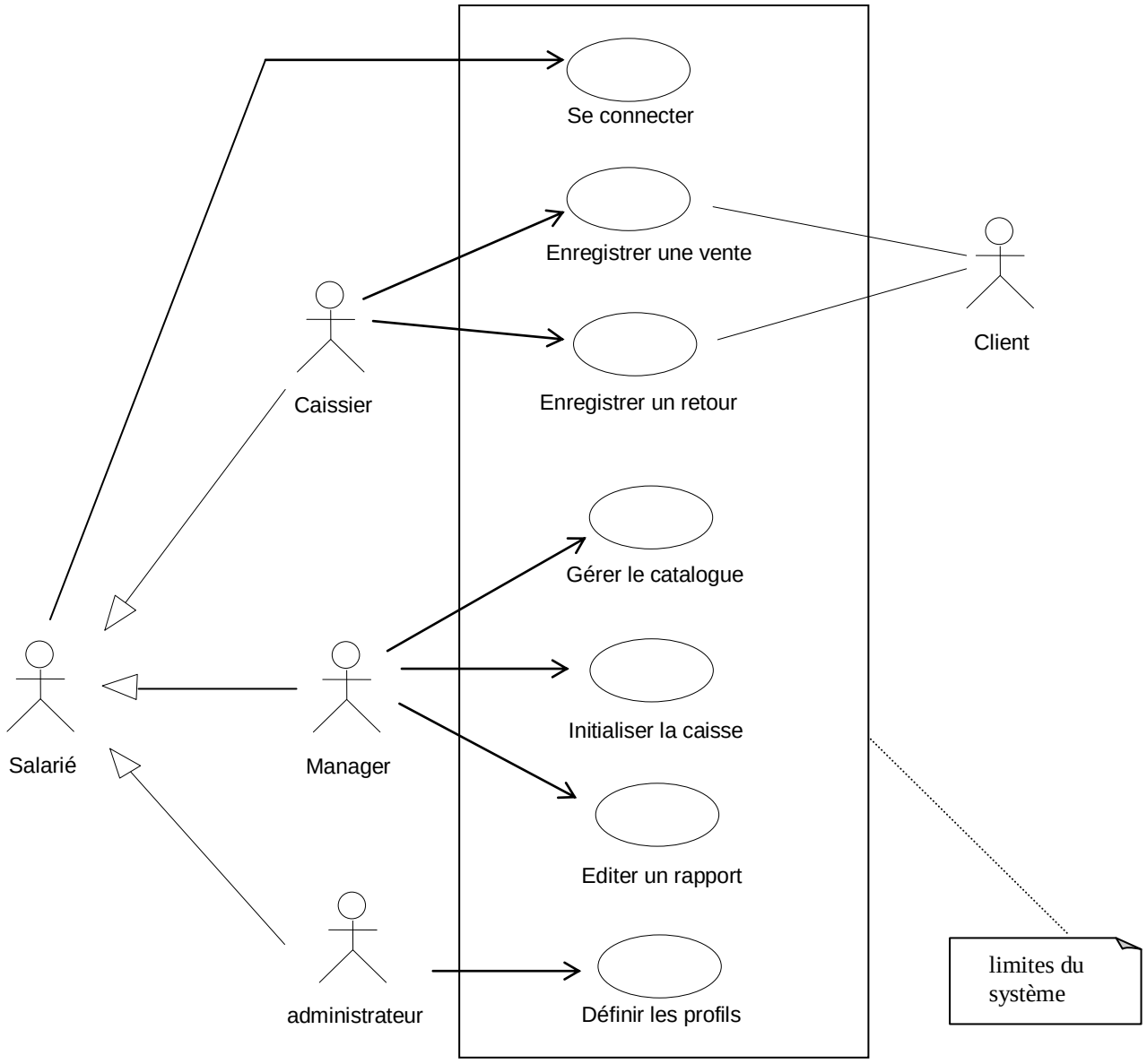
Troisième étape : les cas d'utilisation (ou use case)

Un cas d'utilisation ou *use case* représente un ensemble de séquences d'actions réalisées par le système, en réponse à un événement, et produisant un résultat observable qui apporte une valeur à un acteur.

Cette approche en termes de service rendu par le système pour un acteur dans le cadre de son métier est fondamentale. Elle permet de construire un système orienté vers le besoin de l'utilisateur.

L'objectif est le suivant: l'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond à une fonction métier, selon le point de vue d'un acteur.

On se sert donc avec profit des messages identifiés dans le modèle de contexte dynamique et des fonctions associées, en les regroupant en unités cohérentes qui représentent les cas d'utilisation.



Un autre point doit retenir notre attention, celui de domaine. Ici nous étudions plutôt la caisse que l'ensemble du magasin, aussi nous avons intérêt à indiquer les limites de notre système pour identifier sans ambiguïté les cas d'utilisation intéressants.

Description d'un cas d'utilisation

La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML.

Toutefois, nous trouvons dans la littérature plusieurs exemples dont le modèle suivant représente une bonne synthèse :

USE CASE	Nom du cas d'utilisation
Acteurs	C'est la liste des acteurs qui interviennent pour la réalisation du Use Case. L'acteur principal est souligné. *Les acteurs secondaires sont indiqués en italique.*
But	Il présente en quelques phrases l'objectif du Use Case.
Pré-conditions	Contraintes fonctionnelles liées à l'exercice du métier dans le domaine d'application, qui doivent être vérifiées pour exécuter le Use Case.
Post-conditions	État du système après l'exécution du Use Case.
Déclenchement	C'est l'événement qui initialise le Use Case.
Scénario nominal	Cette section décrit les interactions entre les acteurs et le système dans le cas ou tout se passe bien.
Scénario alternatif	Cette section reprend en détail les alternatives possibles identifiées dans le scénario nominal. Le traitement associé à chaque alternative doit être décrit.
Exceptions	Cette section reprend en détail les exceptions identifiées dans le scénario. Le traitement associé à chaque exception doit être décrit. Une exception interrompt le traitement et le système reste à l'état initial.
Contexte d'utilisation	Décrit les conditions dans lesquelles ce Use Case est utilisé par les acteurs. Cette section identifie, de manière non exhaustive, toutes les contraintes connues à cette phase du projet : - niveau de priorité (basse, moyenne, haute) - fréquence d'utilisation - contrainte sur le temps d'exécution - contraintes sur l'interface utilisateur - type de matériel utilisé
Questions ouvertes	Cette section évoque des questions ou des alternatives non encore tranchées.

Nom du cas d'utilisation

Il est conseillé de nommer les cas d'utilisation en commençant par un verbe.

Acteurs

L'acteur principal est celui pour qui le cas d'utilisation produit la plus-value métier. En conséquence, l'acteur principal est la plupart du temps (mais pas forcément) celui qui déclenche le cas d'utilisation.

Les autres acteurs éventuels sont des acteurs secondaires. Ils peuvent par exemple consulter ou surveiller le système au cours de l'exécution du cas d'utilisation. Ils peuvent aussi fournir des services au système comme un service d'autorisation de paiement automatisé.

Post-conditions

Un cas d'utilisation doit se terminer en laissant le système et les données dans un état stable et cohérent.

Scénario nominal ou principal

Un scénario est une suite spécifique d'actions et d'interactions entre les acteurs et le système.
On le nomme également instance de cas d'utilisation.

Il est rédigé en style "essentiel", c'est-à-dire qu'on ne fait pas référence à la technologie, notamment on laisse de coté l'interface utilisateur, pour se concentrer sur les *intentions* de l'acteur et les *responsabilités* du système.

Le formalisme est textuel et prend la forme suivante:

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

Le scénario décrit la chronologie des échanges entre le système et les acteurs, dans le cas nominal (celui qui se déroule avec succès).

Scénario alternatif

C'est ici que l'on décrit les variantes au scénario nominal ainsi que les cas d'erreur avec les traitements qui en découlent. La numérotation étend celle du scénario nominal.

2a.	Le code de l'article est inconnu … Le système renvoie un message et l'acteur principal refait …
4a.	Si … alors le système affiche … et le use case s'arrête

Exceptions

5b.	Impossible d'accéder à la base de données
.	Le système affiche un message d'exception et le use case s'arrête

Nous restons bien fonctionnels car nous ne décrivons pas comment est réalisé le système, mais comment sont réalisés les services qu'on lui demande.

Exemple de description du cas d'utilisation* Enregistrer une vente *:

UC1	Enregistrer une vente
Acteurs	Caissier, *client*
But	Enregistrer une vente correspondant à un achat client, délivrer un ticket de caisse et encaisser le paiement
Pré-conditions	La caisse est allumée et initialisée Le caissier s'est connecté
Post-conditions	Le client a payé et ramassé tous ses articles. La vente est enregistrée.
Déclenchement	Le client arrive à la caisse avec ses achats
Scénario nominal	1. Le caissier enregistre un article avec la douchette.[1.a] [1.b] 2. Le système recherche le prix et la description de l'article, les affiche et ajoute ces informations à la transaction en cours.[2.b] *Le caissier répète les étapes 1 et 2 jusqu'à ce que tous les articles soient passés.* 3. Le caissier indique la fin de vente quand il n'y a plus d'article. 4. Le système calcule et affiche le montant total de l'achat. 5. Le caissier annonce au client le montant total. 6. Le client donne au caissier une somme d'argent supérieure ou égale à la somme demandée.[6.a] 7. Le caissier enregistre la somme donnée par le client. 8. Le système calcule la somme à rendre au client, enregistre la vente effectuée et édite le ticket de caisse.[8.a] 9. Le caissier encaisse la somme remise par le client, lui rend la monnaie et lui donne le ticket de caisse.
Scénario alternatif	1a Le code produit peut être invalide. Le système affiche un message d'erreur. Le caissier saisit le code au clavier. 1b Il peut y avoir plusieurs articles d'un même produit. Le caissier enregistre le produit ainsi que le nombre d'articles. 2b S'il y a plusieurs articles d'un même produit le système calcule le sous total. 6a Le client n'a pas la somme d'argent suffisante. Le caissier annule la vente sur le système. 8a Le système ne peut pas éditer le ticket de caisse. Un message est affiché au caissier, et celui-ci change le rouleau de papier. .
Exceptions	Impossible d'accéder à la base de données
Contexte d'utilisation	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.
Questions ouvertes	

Remarque 1 :

Pour la présentation du scénario, nous pouvons préférer le format sur deux colonnes ou conversationnel, qui met l'accent sur le fait qu'une interaction a lieu entre les acteurs et le système.

Ici cela donnerait :

Actions des acteurs	**Réponses du système**
1) Le caissier enregistre chaque article.	2) Le système recherche le prix et la description de l'article, les affiche et ajoute ces informations à la transaction en cours.
3) Le caissier indique la fin de vente quand il n'y a plus d'article.	4) Le système calcule et affiche le montant total de l'achat.
5) Le caissier annonce au client le montant total.	
6) Le client donne au caissier une somme d'argent supérieure ou égale à la somme demandée.	
7) Le caissier enregistre la somme donnée par le client.	8) Le système calcule la somme à rendre au client, enregistre la vente effectuée et édite le ticket de caisse.
9) Le caissier encaisse la somme remise par le client, lui rend la monnaie et lui donne le ticket de caisse.	

Cette représentation n'est pas fondamentale dans la mesure où si elle est nécessaire, le diagramme de séquence boîte noire le montrera plus clairement.

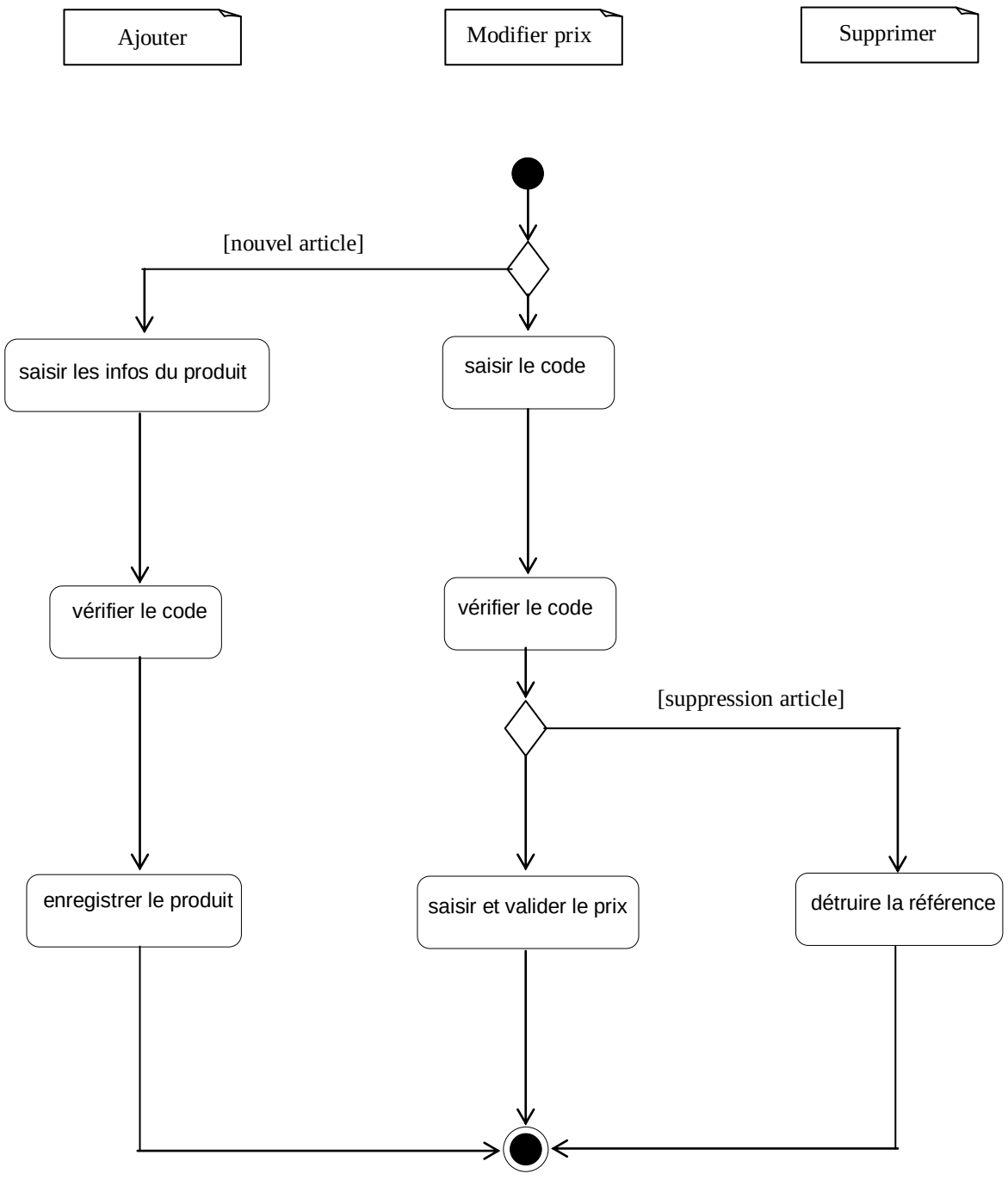
Remarque 2 :

Un cas d'utilisation peut avoir plusieurs scénarios. Pour chacun il va falloir faire une description détaillée (avec traitement d'erreur et d'exception). Mais il est bon de regrouper l'ensemble de ces scénarios sur un diagramme pour avoir une vue complète du cas d'utilisation.

Par exemple pour le cas d'utilisation *gérer le catalogue* qui comprend les scénarios :

- ajouter un article au catalogue
- modifier le prix d'un article
- supprimer un article du catalogue

Nous pouvons utiliser un diagramme d'activité pour montrer ces 3 scénarios.

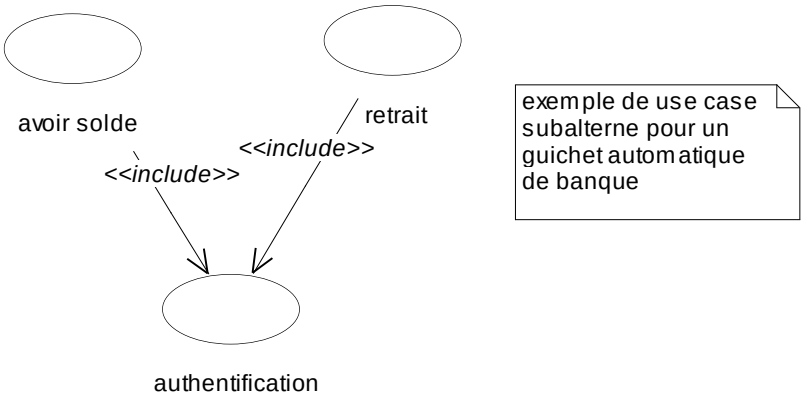


Organisation des cas d'utilisation ***à utiliser avec précaution***

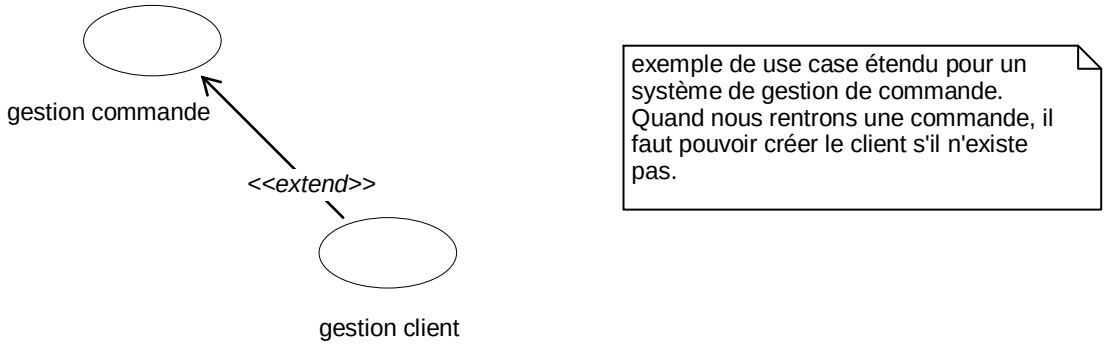
Le schéma de uses cases peut faire apparaître:

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

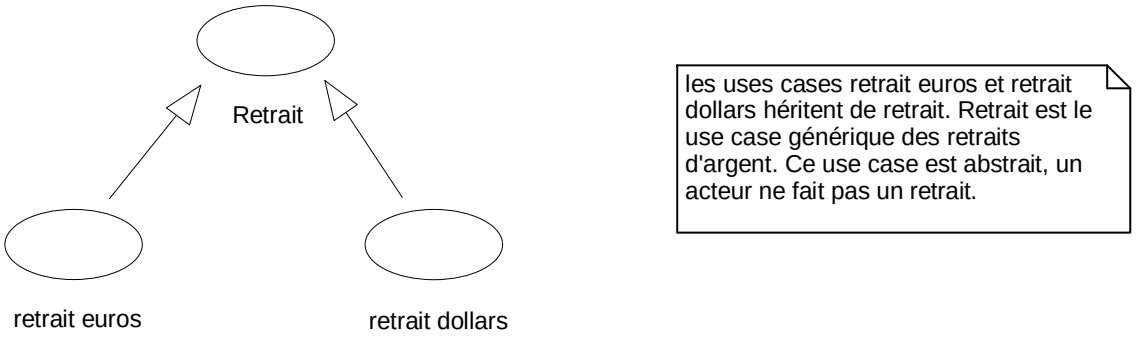
  Un tel use case ne peut être lié qu'à un use case, pas à un acteur. (nous mettrons alors le stéréotype **<<include>>** sur le lien use case vers use case subalterne).



- 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 **généralisation/spécialisation** de cas d'utilisation. Plusieurs uses cases peuvent avoir une trame commune et donc hériter d'un use case parent et abstrait.



Ces représentations de "généralisation/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.

Quatrième étape : les concepts du domaine et les informations attachées

La définition des cas d'utilisation ne doit pas être une fin en soi !

La technique mise au point par Ivar Jacobson comporte deux objectifs principaux :

- dialoguer avec le client sur son expression des besoins grâce à une description fonctionnelle qu'il comprend facilement,
- identifier les principaux concepts et les informations qui leur sont attachées.

Les trois premières étapes traitent du premier objectif. Nous allons maintenant aborder le second.

Les premiers concepts identifiés dans cette étape doivent être des concepts connus des utilisateurs du système. (C'est ce qu'on appelle couramment des classes métier en UML).

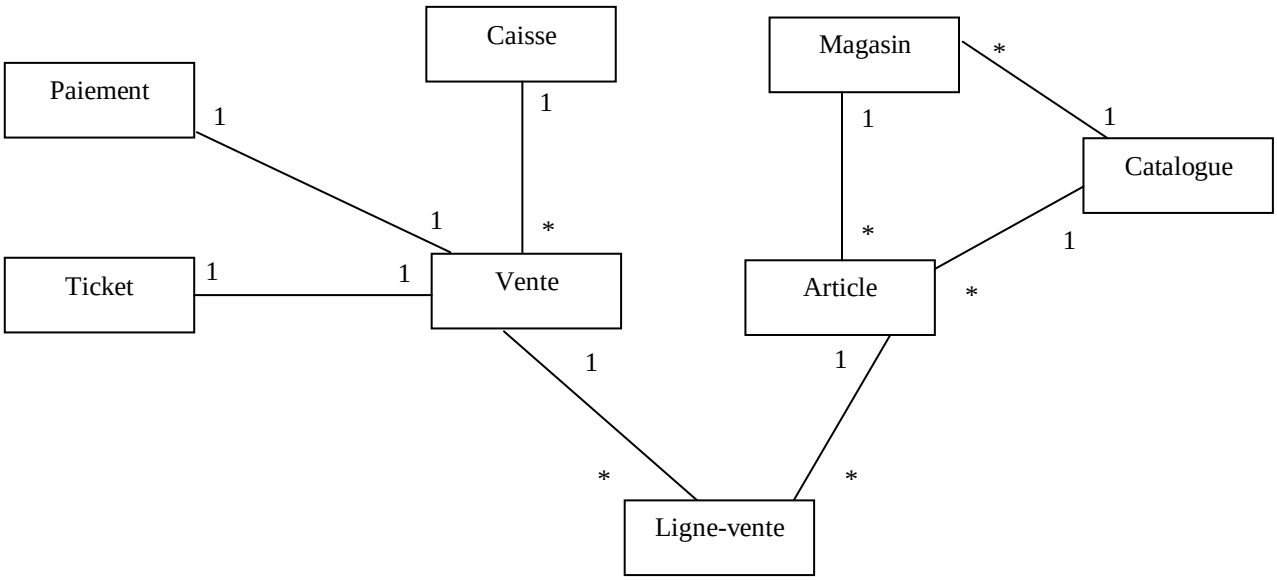
Le concepteur ajoutera dans un second temps des concepts "applicatifs" liés à l'informatisation (classes graphiques, classes utilitaires, classes de gestion de la persistance pour l'accès aux bases de données).

Comment identifier les bonnes classes métier ?

- Cherchez les noms communs importants dans les descriptions textuelles des cas d'utilisation.
-
- Vérifiez les propriétés "qui caractérisent ces concepts (identité, état, comportement).

On formalise ces concepts métier sous forme d'un schéma.

Concepts identifiés pour le cas d'utilisation *Enregistrer une vente*



La réunion de tous ces diagrammes, après élimination des redondances, va représenter l'ensemble des concepts du domaine avec leurs relations.

Les informations

Un système d'information est composé de traitements et de données.

Pour le cas d'utilisation Enregistrer une vente, on peut repérer les informations suivantes :

- Le numéro de la caisse
- Le numéro de la vente
- La date de la vente
- Le code article
- Le numéro du magasin
- Le nom du magasin
- L'adresse du magasin
- Le numéro de catalogue
- La date de mise en application du catalogue
- Le prix unitaire de l'article
- Le libellé de l'article
- Le numéro de la ligne de vente
- Le code article de la ligne de vente
- La quantité d'article acheté pour la ligne de vente
- La date et l'heure de la vente
- Le numéro de ticket
- La date et l'heure d'impression du ticket
- Le mode de paiement : espèces, chèque, cb
- Etc…

Toutes ces informations seront décrites dans un document ou sur un support magnétique sous forme de dictionnaire des données.

Ensuite, ces données seront regroupées par concept que nous appellerons Entités avec la méthode MERISE, ou classes avec le langage de modélisation UML, ou fichiers dans un système plus ancien.

L'analyse des traitements

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 sous forme textuelle.

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.

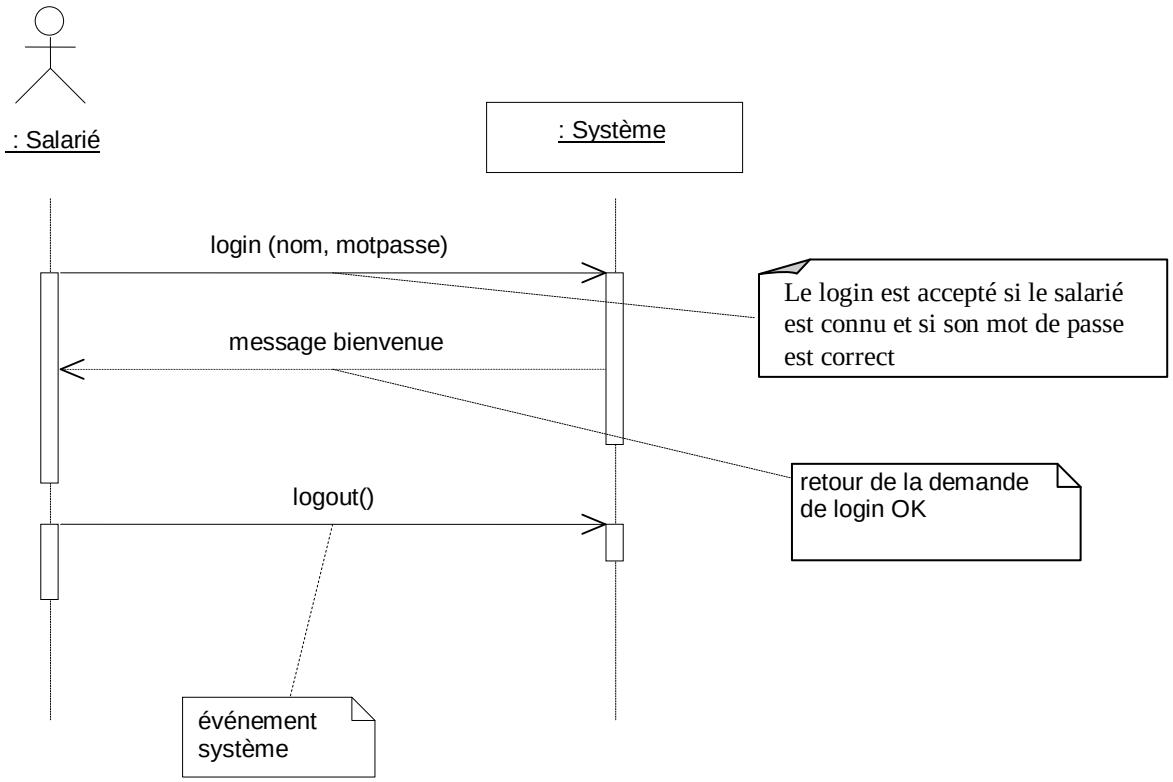
Cinquième étape : diagramme de séquence boîte noire

Avant de passer à la conception logique du fonctionnement de l'application, il est utile d'explorer et de définir son comportement en tant que "boîte noire". Ce comportement est une description de *ce que* le système réalise, mais n'explique pas *comment* il le réalise.

Les cas d'utilisation ont été validés par le client. Ils décrivent de son point de vue, l'interaction des acteurs externes avec le système logiciel que l'on vent créer. Lors de cette interaction, les acteurs envoient des messages au système qui appellent généralement une réponse de ce dernier.

Il faut maintenant nous placer du point de vue système informatique.
Les messages des acteurs reçus directement par le système sont appelées des **événements système**. Les actions engendrées par le système suite à ces événements sont appelées des opérations. A un événement système correspondra une opération. Les événements système peuvent comprendre des paramètres.

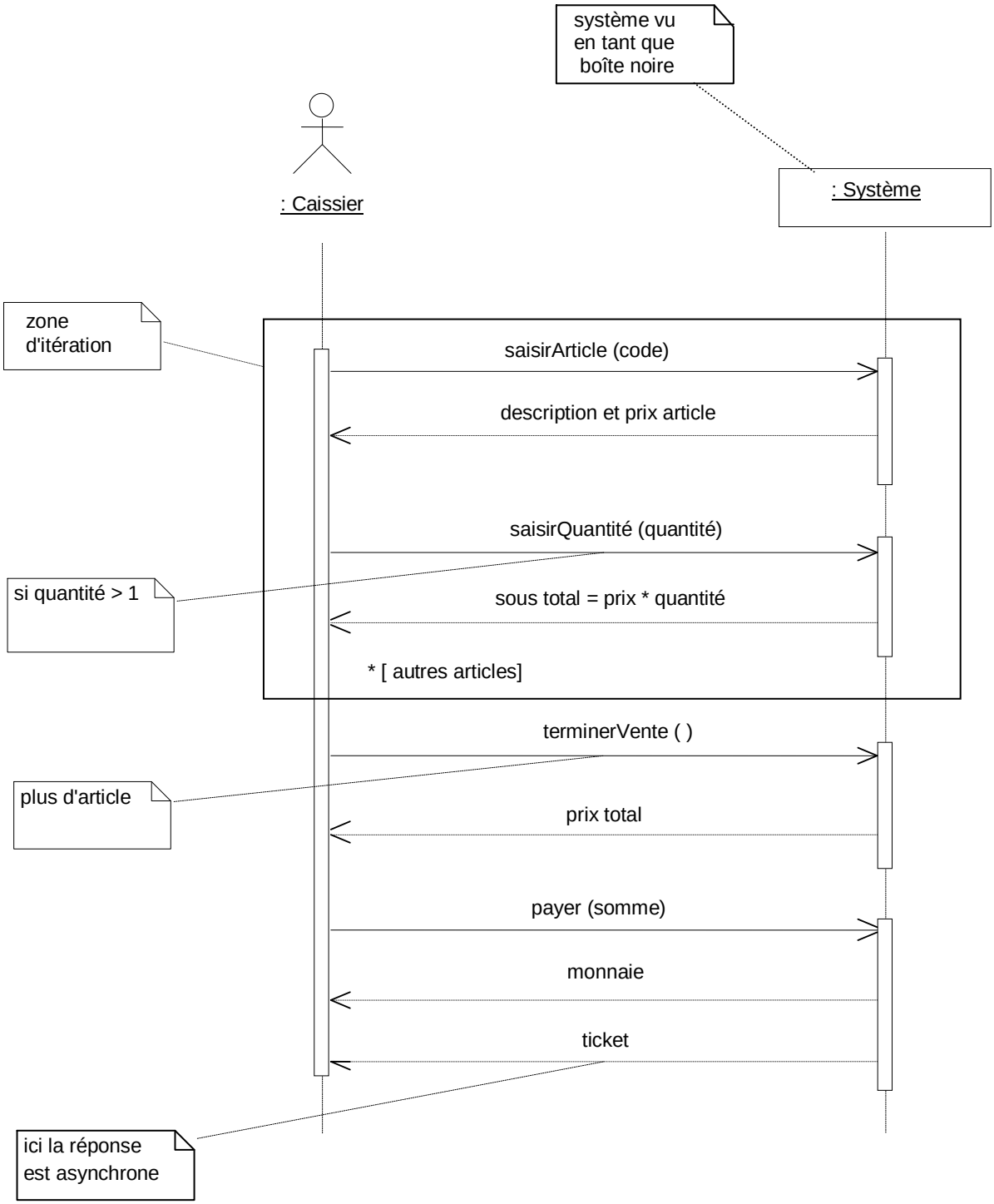
Le diagramme de séquence boîte noire montre, pour chaque scénario des cas d'utilisation, les événements système déclenchés par les acteurs dans l'ordre chronologique, et les réponses éventuelles du système.
Il se présente ainsi :



Les retours se formalisent de différentes manières suivant la version d'UML choisie. Ici nous avons choisi des flèches pointillées pour ne pas les confondre avec un événement (sens acteur vers système informatique).

A cette étape, nous pourrions construire une maquette des écrans pour les acteurs. Notre maquette d'écran serait une maquette "fonctionnelle" (c'est-à-dire que le dessin de l'écran ne serait pas figé). Formellement nous le ferons en phase de conception.

Diagramme de séquence boîte noire du use case détaillé *Enregistrer une vente*



Dans ce use case, le client n'est jamais en interaction directe avec le système. C'est donc bien le caissier qui génère les événements système.

Remarque :
Il est bon de commencer l'appellation des événements système par un verbe car cela souligne bien le caractère de commande de ces événements.