CAS MAGASIN VIDEO

PASCAL ANDRE
INPHB - DEPARTEMENT MATHEMATIQUES & INFORMATIQUE
BP 1083 YAMOUSSOUKRO - COTE D'IVOIRE
TEL +225 05923485 - FAX +225 30640306
EMAIL andrep@centre.inphb.edu.ci

VERSION DU 16/11/00
MODIFIE LE 04/12/00

I. INTRODUCTION

Ce document est une modélisation UML du cas vidéo. Un club vidéo a une activité de prêt de cassettes magnétoscope (un film par cassette). Le club dispose de plusieurs boutiques où ses adhérents peuvent emprunter des cassettes, qu'ils rapportent par la suite.

Un adhérent peut emprunter un nombre quelconque de cassettes en une ou plusieurs fois. On devra lui interdire de faire plusieurs emprunts le même jour dans la même boutique mais il pourra le même jour faire des emprunts dans des boutiques différentes. On devra lui interdire d'emprunter une cassette s'il a dépassé la date de retour prévue d'une des cassettes qu'il a en emprunt. Lorsqu'il peut emprunter une cassette, l'adhérent donne le nombre de jours pendant lesquels il compte la garder (le boutiquier s'arrangera toujours avec l'adhérent pour que la date de retour prévue ne soit pas un jour de fermeture avant de saisir ce nombre de jours).

L'adhérent peut rendre des cassettes (toutes ou en partie) empruntées par lui ou par un autre adhérent et il n'est pas obligé d'effectuer ce retour dans la ou les boutiques où l'emprunt a eu lieu. Le prix de location d'une cassette est proportionnel au nombre de jours d'emprunt.

Le club veut pouvoir :

- Gérer son stock de cassettes, les emprunts et les retours des cassettes des adhérents.
- Enregistrer de nouveaux adhérents (on ne prévoit pas la suppression des adhérents).
- Effectuer des statistiques.

Le club a déjà prévu l'ensemble des informations suivantes :

- nom, adresse, numéro de téléphone d'une boutique,
- nom, adresse, numéro de téléphone d'un adhérent,
- titre, classification (aventure, historique, policier,...), durée de projection, prix journalier de location d'un film.

Il y a bien sûr plusieurs cassettes pour un même film, on les distinguera par un numéro d'exemplaire. De plus elles peuvent être situées dans des boutiques différentes. Eventuellement on ajoutera un certain nombre d'informations afin de pouvoir effectuer les opérations suivantes :

- prêt et retour de cassettes,
- enregistrement d'un nouvel adhérent,
- édition des films disponibles, des films empruntés,
- édition de statistiques afin de déterminer les films les plus demandés, les adhérents qui sont les plus gros consommateurs, etc.

Le gérant du club souhaite automatiser les requêtes suivantes :

- pour un prêt :
  - vérifier le droit d'un adhérent d'emprunter une cassette,
  - vérifier qu'au titre du film qu'il veut emprunter correspond une cassette disponible dans la boutique où le prêt est demandé.
  - renseigner dans le cas contraire cet adhérent, c'est-à-dire lui fournir les noms des boutiques éventuelles où il y a des cassettes disponibles correspondant à ce film.
  - mettre à jour toute la base quand le prêt peut être réalisé dans la boutique où se trouve l'adhérent.
- pour un retour :

  fournir le prix à payer pour chaque cassette rendue et mettre à jour toute la base, aue le retour d'une cassette soit celui de l'adhérent présent dans la boutique ou celui qu'il effectue pour un autre.

Ce texte de base pourra être enrichi pour illustrer certains aspects de la notation UML. Notez que la précision du texte est augmentée par les cas d'utilisation, dont le rôle est de spécifier les besoins.

La démarche adoptée dans ce cas d'étude nous est propre. Il ne s'agit nullement d'une démarche «unifiée », même si les grandes lignes sont induites par la notation. Ces grandes lignes sont classiques : analyse des besoins (cas d'utilisation +

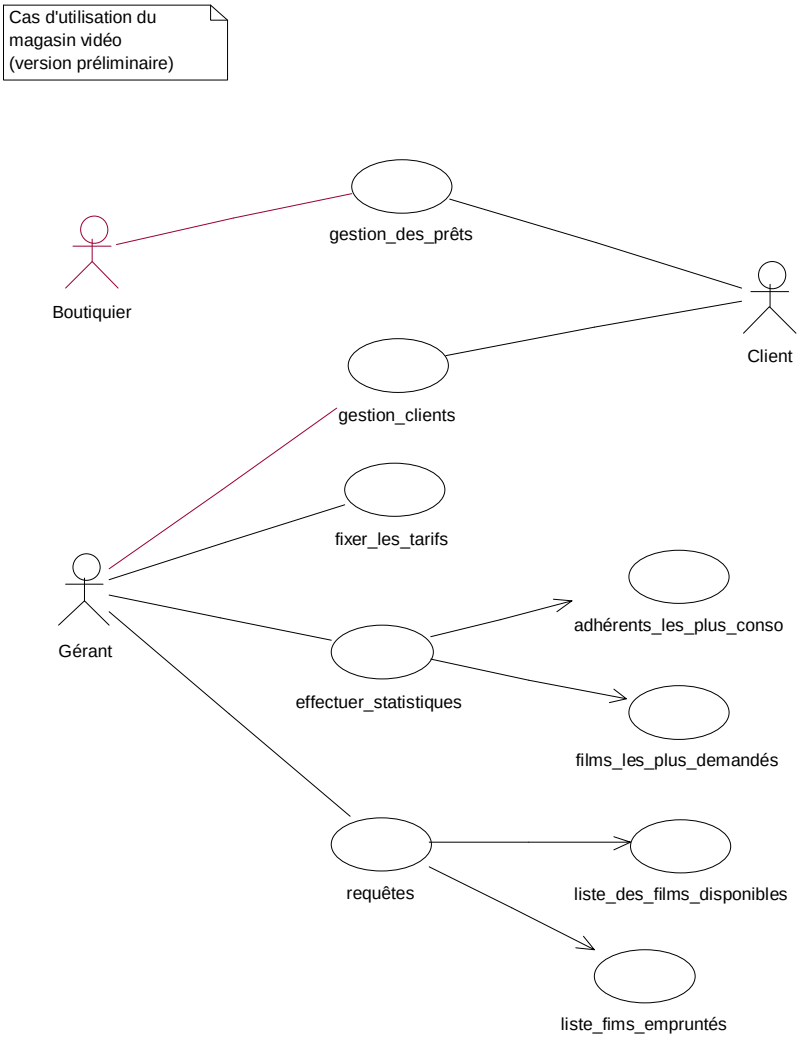
séquences d'illustration), analyse (diagrammes de classes, états transitions), conception (diagrammes de classes, diagrammes de composants, diagramme de déploiement). Les diagrammes de collaboration et de séquence s'utilisent aussi en analyse et pourquoi pas en conception. L'articulation des différents diagrammes et leur niveau de description sont encore l'apanage des spécialistes et font l'expérience des analystes. Nous nous sommes inspirés de la littérature actuelle sur Merise pour proposer une démarche adaptée à la conception de systèmes d'informations. Comme toutes les démarches, celle-ci est modulable selon des problèmes à traiter. Nous avons utilisé l'outil Rose 98 de Rational Software Corporation (`www.rational.com/rose`) pour dessiner les diagrammes.

II. ANALYSE DU BESOIN

Nous commençons par structurer les besoins. L'analyse des besoins peut se faire directement en utilisant les notations d'UML. En particulier, les diagrammes de cas d'utilisation, accompagnés de descriptions textuelles permettent de préciser le besoin, sans entrer dans des détails d'organisation ou d'implantation. Des scénarios enrichissent la description en donnant des exemples de situations. Nous avons choisi de présenter ensemble les cas d'utilisation et les scénarios correspondants.

II.1 Cas d'utilisation et scénarios

Nous distinguons principalement trois acteurs : l'adhérent, le gérant et le boutiquier. En fait, d'autres acteurs peuvent apparaître au cours de la description des besoins. Dans le texte informel, nous relevons des emprunts, des retours de cassettes, des inscriptions d'adhérents, des éditions de films, de la gestion de stock de cassettes et des statistiques. En première approche, nous pouvons établir les cas d'utilisation suivants de \lafigure{viUCprel}.



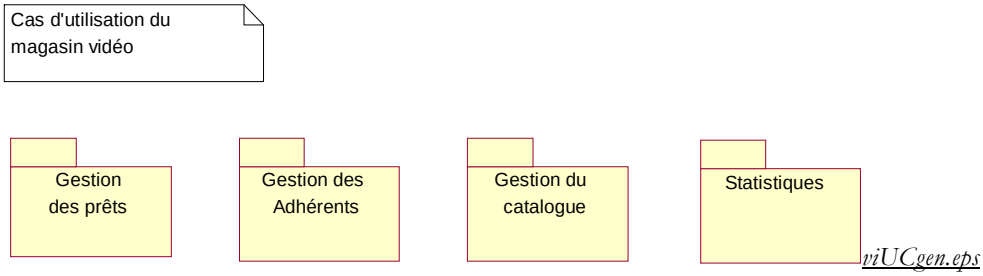
viUCprel.eps

La difficulté majeure, à notre avis, de la modélisation par cas d'utilisation est l'organisation même du diagramme. En particulier, il faut fixer le grain de définition :

- Un UC est plus qu'une fonction. Par exemple, nous avons regroupé emprunts et retours de cassettes dans un seul cas d'utilisation. Nous décrirons des scénarios pour les emprunts et d'autres pour les retours.

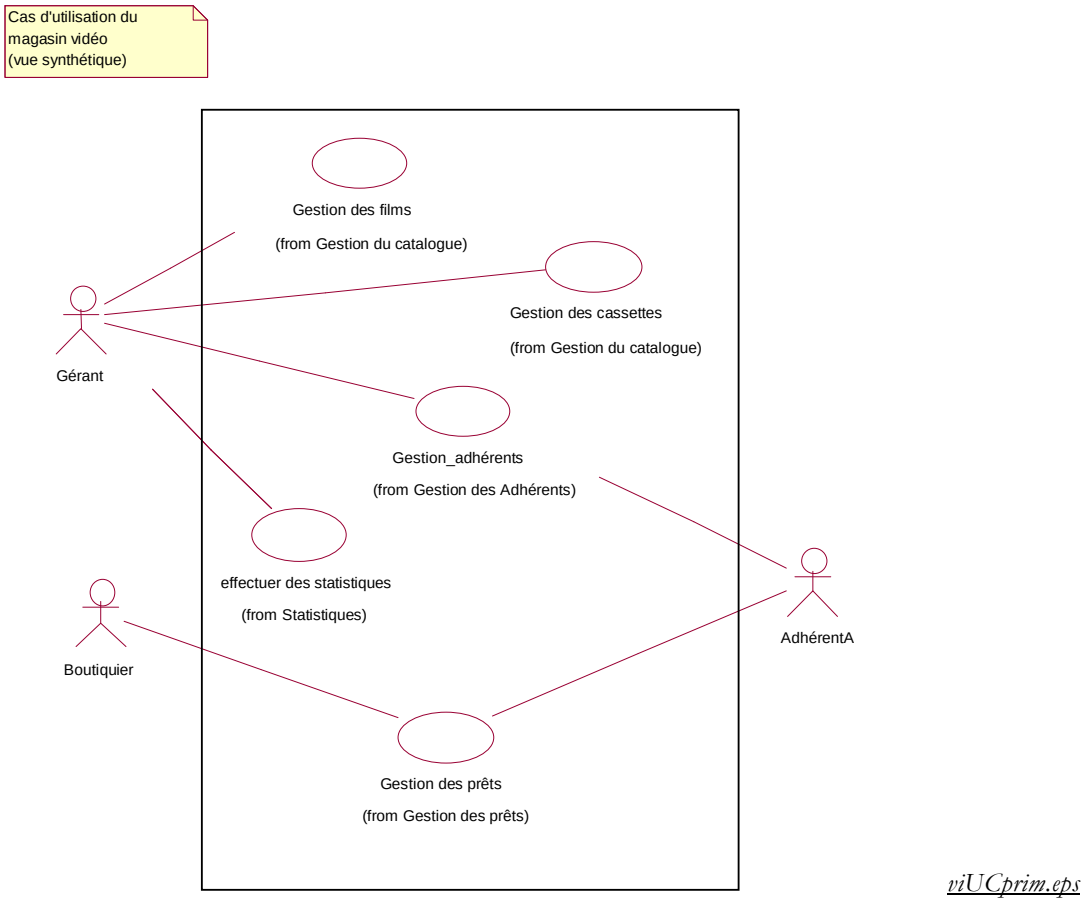
- Un UC n'est pas un découpage fonctionnel. Ainsi, les cas d'utilisation effectuer des statistiques et requêtes font une utilisation biaisée de la relation entre cas d'utilisation.

Par ailleurs, le diagramme précédent est trop partiel. Nous proposons donc une nouvelle organisation du diagramme qui s'appuie sur les catégories (paquetages). Nous distinguons de grands ensembles d'utilisation : gestion des adhérents, des prêts, du catalogue (stock, tarifs, films) et des statistiques. La \lafigure{viUCgen} synthétise cette organisation. Une vue synthétique en a été extraite dans la \lafigure{viUCprim}.



viUCgen.eps

> Notez que cette organisation ne reflète pas le modèle des informations (diagramme des classes). Ainsi les adhérents sont mis à jour dans la gestion des prêts et dans celle des adhérents (la base est partagée par tous les UC).



viUCprim.eps

La \lafigure{viUCprim} est une vue synthétique, extraite des différents diagrammes, que nous présentons dans la suite. Pour chaque cas d'utilisation, nous proposons une description textuelle et des scénarios représentatifs. Les scénarios sont une représentation simplifiée et abstraite des diagrammes de séquence qui font apparaître les interactions entre les acteurs et le système. Adhérent est suffixé par A pour ne pas confondre avec l'adhérent utilisé ensuite en analyse.

> Le type des communications, les paramètres et les annotations temporelles ne sont pas utilisées à ce niveau d'abstraction.
>
> Nous utilisons des structures de contrôle sous forme de commentaires : ils ont pour but de rassembler plusieurs scénarios en un seul diagramme. Ainsi, pour représenter une alternative, nous devrions définir deux scénarios distincts ; nous les rassemblons en ajoutant des commentaires.

II.1.1 Gestion des prêts

Examinons maintenant la catégorie Gestion des prêts. Plusieurs modélisations sont possibles.

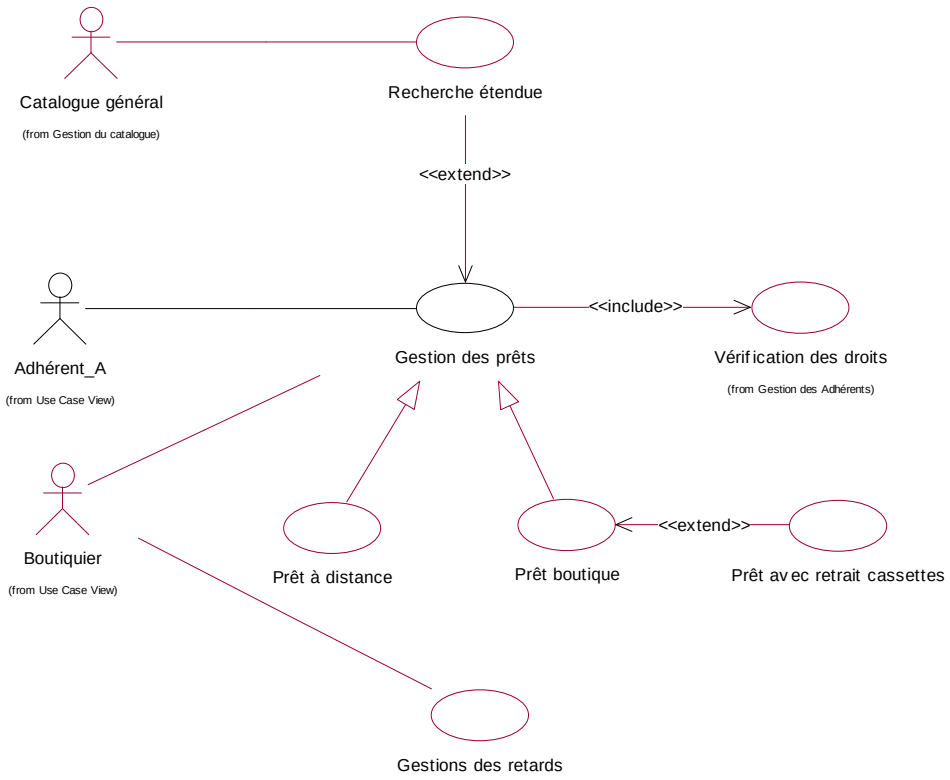
- Dans le premier cas, on distingue deux cas d'utilisation principaux : l'emprunt et le retour de cassettes.

- Dans le second cas, emprunts et retours sont regroupés sous le même cas d'utilisation Gestion des prêts.

> Les deux modélisations sont acceptables. La première fournit des cas plus simples, la seconde est plus adaptée à des modélisations volumineuses. Nous choisissons la seconde parce qu'elle est plus synthétique et justifie un regroupement modulaire : le prêt ou le retour correspondent au même type d'interaction (adhérent/boutiquier) et mobilise sensiblement les mêmes ressources.

La description détaillée des UC fait apparaître une vérification des droits de l'adhérent et une consultation hors boutique. Ces éléments étant relativement indépendants des prêts, et pouvant être invoqués par d'autres cas d'utilisation (voir Gestion des adhérents), nous les modélisons par des cas d'utilisation. La vérification des droits étant systématique, nous utilisons une relation d'inclusion. La consultation hors boutique étant facultative, elle est reliée à la gestion des prêts par une relation d'extension. Les prêts à distance et les prêts boutique sont ajoutés à l'énoncé pour illustrer la relation d'héritage. La \lafigure{viUCpret} illustre ce diagramme.

> Notez que l'UC *Gestion des prêts* est une version générale et qu'elle n'existe pas en pratique (les opérations se font soit en boutique, soit à distance). En ce sens, elle est abstraite (au sens des classes abstraites). On peut lui associer le stéréotype <<abstract>>.



viUCpret.eps

Précisons les différents cas d'utilisation.

a) Gestion des prêts.

La gestion des prêts regroupe les activités d'emprunt et de retour de cassette et leurs variantes.

Cas d'utilisation : *Gestion des prêts*
inclut : Vérification des droits
Acteurs : Adhérent (primaire), Boutiquier (secondaire)
Invariant : Un adhérent ne peut emprunter deux fois dans la même boutique le même jour.
Description générale
La gestion des prêts comprend les opérations courantes de transactions de cassettes : emprunt et retour d'exemplaires de cassettes.
Cas : `Emprunt de cassette`
L'adhérent demande à emprunter des cassettes. Vérifier le droit de l'adhérent d'emprunter une cassette (appel à l'UC *Vérification des droits*). Pour chaque cassette demandée :
1. Vérifier qu'au titre du film qu'il veut emprunter correspond une cassette disponible dans la boutique où le prêt est demandé. Dans le cas contraire, il faudra renseigner cet adhérent, c'est-à-dire lui fournir les noms des boutiques

éventuelles où il y a des cassettes disponibles correspondant à ce film via l'UC *Recherche étendue*. Cette UC fait appel à un programme externe, symbolisé par l'acteur *Catalogue général*, lorsque le film n'est pas enregistré dans le club. Si le film est enregistré dans le club, une réservation dans une autre boutique peut être faite à ce niveau.

2. Quand le prêt peut être réalisé dans la boutique en relation avec l'adhérent, celui-ci demande à retirer la cassette (appel éventuel à l'UC *Prêt avec retrait cassettes*). Le retrait de cassettes met à jour toute la base d'informations.

Cas : `Retour de cassette`

L'adhérent rend des cassettes empruntées. Que le retour d'une cassette soit celui de l'adhérent présent dans la boutique où celui qu'il effectue pour un autre, il faudra fournir le prix à payer pour chaque cassette rendue et mettre à jour toute la base. A la fin, on encaisse le montant dû par l'adhérent. La gestion financière sort du contexte du système, on se contentera de valider le paiement.

Exceptions

Cas : `Prêts, droits non accordés`

précondition : Le refus porte sur des restitutions de cassette ou des emprunts réalisés dans la même boutique.

Le traitement est arrêté.

Cas : `Prêts, non inscrit`

précondition : L'adhérent n'est pas inscrit.

Nous avons deux possibilités : soit on enregistre l'adhérent (appel à l'UC Gestion des adhérents) et on reprend le traitement en cours, soit on arrête le traitement. Par soucis de simplification, et pour éviter trop d'appels d'UC, nous choisissons la seconde option.

> Notez que les exceptions peuvent être groupées par cas. En s'inspirant de Merise \cite{uml_kettani}, KETTANI modélise des processus par des UC et leurs différents flux. Les exceptions sont des sorties de flux principaux. Ainsi, on peut avoir un processus Prêt qui assure le suivi des prêts (emprunts, délais, relances, retour). Nous n'avons pas choisi cette voie de modélisation.

Le retrait de la cassette est modélisé à part car nous supposons qu'un adhérent qui vient chercher une cassette réservée depuis une autre boutique, même si quelque part il achève un ou plusieurs emprunts, mobilise d'autres ressources que l'emprunt initial. Nous aurions pu spécialiser le retrait en boutique et le retrait à distance, mais nous supposons que le retrait de cassettes se fait uniquement en boutique.

> En séparant les activités du prêt, nous nous éloignons de la modélisation par processus de Merise (Tome 1 du livre) : un processus prêt qui enchaîne réservation retrait et retour.

Le retour de cassettes peut se faire en boutique (paiement en espèce, en carte ou en chèque) ou à distance (paiement en chèque uniquement). Nous faisons l'hypothèse que le paiement fait partie du domaine financier et sort donc du cadre d'étude. La distinction entre types de paiement était une utilisation classique de la relation d'extension dans la version 1.1 d'UML. Avec la relation de spécialisation, les deux représentations sont possibles. Nous n'avons pas vu de règles précises pour utiliser l'une ou l'autre des relations.

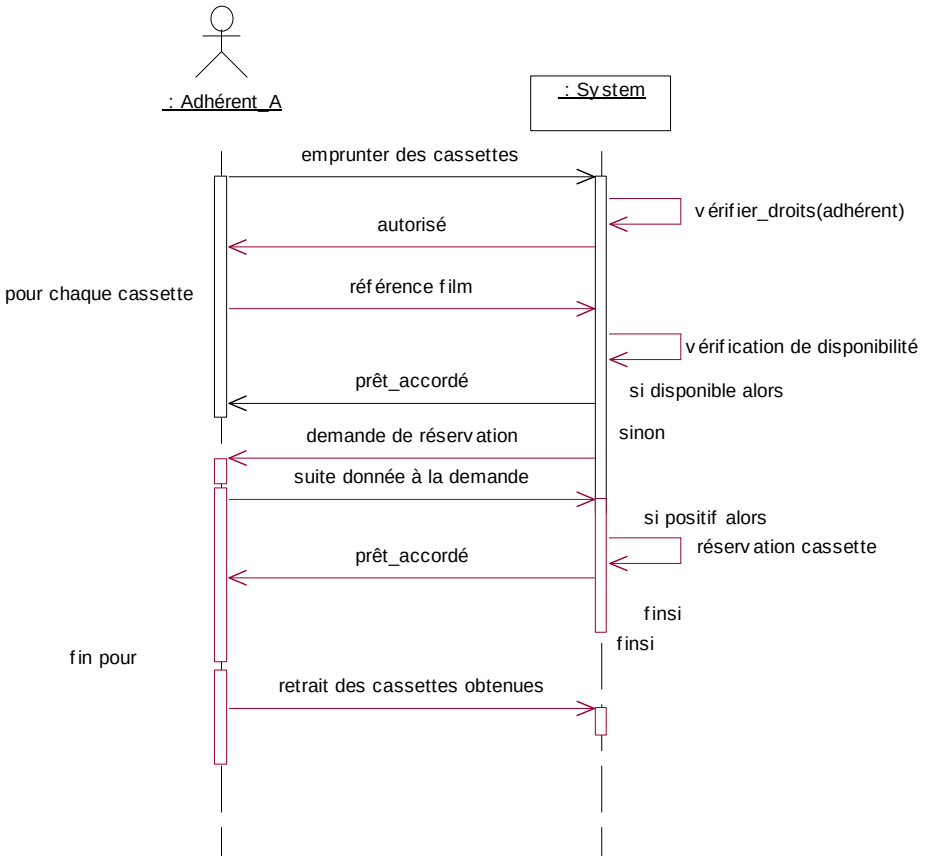
> Notez qu'il n'y a pas symétrie complète (disjonction) entre opérations en boutique et opérations à distance. Même en supposant un retrait à distance (par la poste), rien n'empêche de réserver une cassette d'une boutique dans une autre boutique et de demander le retrait à distance.

Scénario

Un scénario met en évidence des interactions entre les acteurs et le système (noté System ici) à étudier. Une description textuelle peut accompagner le scénario, nous en ferons abstraction ici. La présentation des échanges ne présume en rien de la réalisation de ces échanges, ainsi, l'autorisation d'emprunt est symbolisée par une interaction, au niveau concret, ça peut être un message sur un écran.

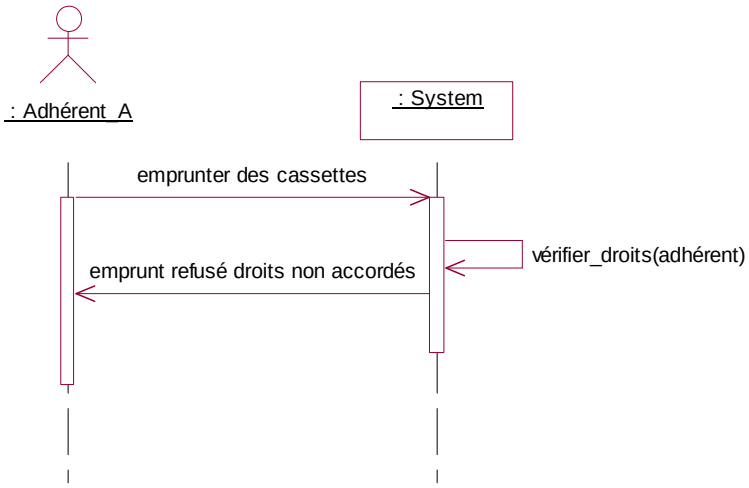
Nous proposons un scénario pour l'emprunt, un scénario pour le retour et un scénario pour chaque exception. Nous faisons abstraction des interventions du boutiquier lorsqu'il s'agit uniquement de transmissions.

- Emprunt normal : le retrait n'est pas spécifié à ce niveau.



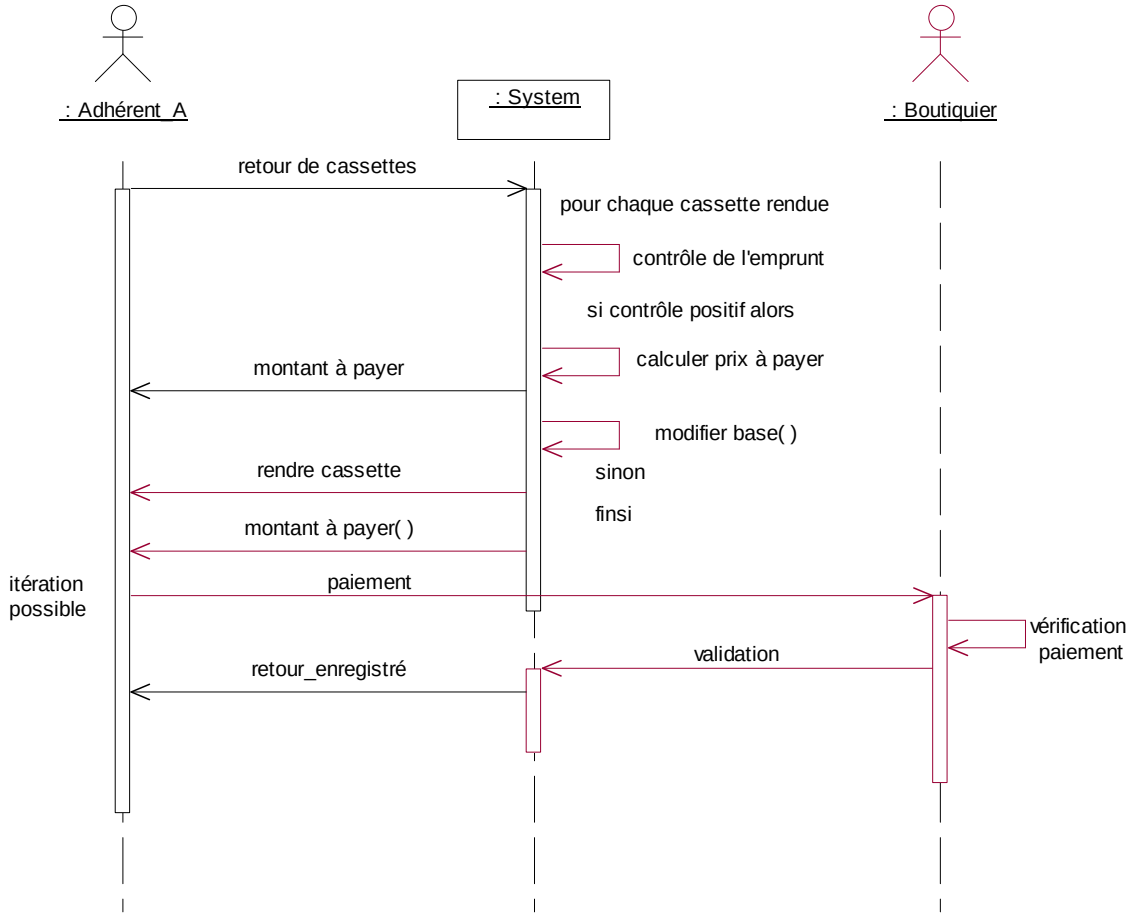
viSCemp.eps

- Emprunt non autorisé.



viSCemp1.eps

- Retour normal.



viSCret.eps

b) Prêt boutique

Les prêts boutiques se font en conversationnel avec un boutiquier. Les prêts à distance ont pour particularité d'utiliser d'autres médias de communication tant pour les ordres donnés (téléphone, fax, mail) que pour la transmission de cassettes (poste, coursier).

Il n'y a pas de règles proposées pour faire le lien entre les relations du diagramme et les descriptions textuelles. Nous supposerons que les extensions et inclusions se font par des appels explicites aux cas d'utilisation invoqués et que par défaut, seules les parties modifiées ou ajoutées sont précisées. Ainsi, les cas invoqués, les contraintes et les acteurs sont directement hérités. Les emprunts et retours sont modifiés pour le prêt en boutique et un nouveau cas vient enrichir le prêt en boutique, celui de la récupération d'une cassette réservée dans une boutique (UC *Prêt boutique* et UC *Prêt à distance*).

Cas d'utilisation : *Prêt boutique*

spécialise : Gestion des prêts

Description générale

La gestion des prêts boutique précise certaines opérations de l'UC *Gestion des prêts* pour les opérations se passant dans les boutiques. Une opération de retrait simple est ajoutée.

Cas : `Emprunt de cassette`

L'adhérent se déplace dans une des boutiques du club et demande à emprunter des cassettes de cette boutique.

1. Le boutiquier vérifie par ordinateur les droit de l'adhérent d'emprunter une cassette (appel à l'UC Vérification des droits). Il l'informe verbalement du résultat de la vérification. L'échec provoque un traitement exceptionnel.

2. Pour chaque cassette demandée :

2.1 Le boutiquier vérifie qu'au titre du film qu'il veut emprunter correspond une cassette disponible dans sa boutique.

2.2. S'il n'en trouve pas, il cherche à fournir les noms des boutiques éventuelles où il y a des cassettes disponibles correspondant à ce film via l'UC *Recherche étendue*. Il informe l'adhérent de l'absence de cassette dans la boutique et du résultat de la recherche étendue. Il lui demande s'il souhaite réserver la cassette dans une autre boutique (s'il y en a). Dans l'affirmative, il réserve pour cet adhérent la cassette en question.

3. Quand le prêt peut être réalisé dans la boutique où se trouve l'adhérent, il retire la cassette (appel éventuel à l'UC *Prêt avec retrait cassettes* qui met à jour toute la base).

Cas : `Retrait de cassettes réservées`

L'adhérent vient retirer des cassettes qu'il a réservées.

1. Vérifier le droit de l'adhérent de retirer des cassettes (appel à l'UC *Vérification des droits*). L'échec provoque un traitement exceptionnel (UC général).

2. Pour chaque cassette, on fait appel à l'UC *Prêt avec retrait cassettes* qui met à jour toute la base.

Cas : `Retour de cassette`

1. L'adhérent rend des cassettes empruntées. Le boutiquier vérifie que les cassettes sont celles du club. Il refuse celles qui ne sont pas enregistrées.

2. Pour chaque cassette les traitements suivants sont opérés :

2.1. Calculer le prix à payer pour chaque cassette rendue en fonction de la durée d'emprunt (cf. l'UC *Tarifs* dans la *Gestion du catalogue*).

2.2. Mettre à jour la base : emprunts en cours de l'adhérent et disponibilité de la cassette.

3. Règlement et reçu (manuel).

Notez que l'invariant, les relations et les exceptions sont héritées.

Cas d'utilisation : *Prêt avec retrait cassettes*

étend : Prêt boutique

Description générale

Le prêt avec retrait cassettes modifie la base de données pour un emprunt validé d'exemplaires de cassettes. Pour chaque cassette à retirer :

1. L'adhérent précise la durée d'emprunt. On vérifie manuellement que cette date est un jour d'ouverture. Dans la négative on propose une autre date au client.

2. Le tarif journalier de location de cette cassette est calculé et mémorisé, rappelons qu'il dépend du genre et de la durée du film.

3. On met à jour la base d'informations : emprunts de l'adhérent (cassette, boutique, date, durée initiale, tarif), non-disponibilité de l'exemplaire dans la boutique (stock, catalogue).

c) Prêt à distance

Etudions maintenant l'UC *Prêt à distance*.

Cas d'utilisation : *Prêt à distance*

spécialise : Gestion des prêts

Description générale

La gestion des prêts à distance précise certaines opérations de l'UC *Gestion des prêts* pour les opérations se passant à distance.

Cas : `Emprunt de cassette`

L'adhérent contacte une des boutiques du club (téléphone, fax ou e-mail) et demande à emprunter des cassettes de cette boutique.

1. Le boutiquier vérifie par ordinateur les droit de l'adhérent d'emprunter une cassette (appel à l'UC Vérification des droits). Il l'informe verbalement du résultat de la vérification. L'échec provoque un traitement exceptionnel.

2. Pour chaque cassette demandée :

2.1 Le boutiquier vérifie qu'au titre du film qu'il veut emprunter correspond une cassette disponible dans sa boutique.

2.2. S'il n'en trouve pas, il cherche à fournir les noms des boutiques éventuelles où il y a des cassettes disponibles correspondant à ce film via l'UC *Recherche étendue*. Il informe l'adhérent de l'absence de cassette dans la boutique et du résultat de la recherche étendue. Il lui demande s'il souhaite réserver la cassette dans une autre boutique (s'il y en a). Dans l'affirmative, il réserve pour cet adhérent la cassette en question.

3. Quand le prêt peut être réalisé dans la boutique contactée par l'adhérent, on réserve la cassette pour l'adhérent dans la boutique.

Cas : `Retour de cassette`

1. Le boutiquier reçoit des cassettes par courrier et un chèque.

2. Pour chaque cassette les traitements suivants sont opérés :

2.1 Le boutiquier vérifie que la cassette a été empruntée en boutique. Il la met de côté si elle n'est pas enregistrée.

2.2. Calculer le prix à payer pour chaque cassette rendue en fonction de la durée d'emprunt (cf. l'UC *Tarifs* dans la *Gestion du catalogue*)

2.3. Mettre à jour la base : emprunts en cours de l'adhérent et disponibilité de la cassette.

3. Vérifier le montant du chèque et établir un reçu (manuel).

d) Gestion des retards.

Cas d'utilisation : *Gestion des retards*

Acteurs : Boutiquier, Adhérent

Description générale

La gestion des retards est un traitement hebdomadaire dont le but est de vérifier les dépassements de délais dans les emprunts et de relancer les adhérents.

Cas : `Hors délais`

Tous les emprunts en cours sont passés en revue. Les emprunts dont la date de retour prévue est inférieure à la date courante donnent lieu au traitement suivant :

1. Classement des emprunts par emprunteur.

2. Rédaction d'une lettre de rappel mentionnant, pour chaque emprunteur, les cassettes non renvoyées.

3. Envoi de la lettre à l'adhérent dont on a mémorisé l'adresse.

Notez que le retour se fait dans la gestion des prêts. Les rappels sont mentionnés dans un historique des adhérents.

Cas : `Perte`

L'adhérent signale la perte d'une cassette.

1. Il paie un forfait (manuel).

2. L'emprunt et l'exemplaire sont supprimés de la base.

Exceptions

Cas : `Lettre déjà envoyée`

précondition : Un retard est notifié mais une lettre a déjà été envoyée la semaine précédente.

Le traitement est arrêté.

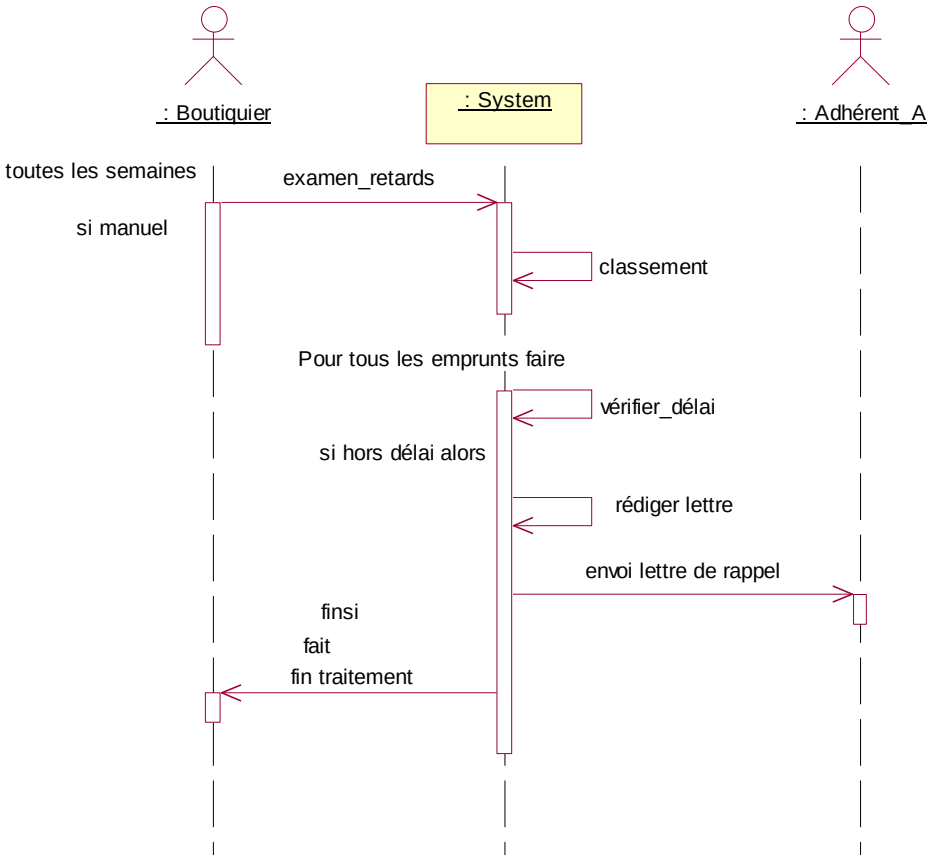
Cas : `Emprunt en retard d'un an`

précondition : Un retard d'une année est notifié.

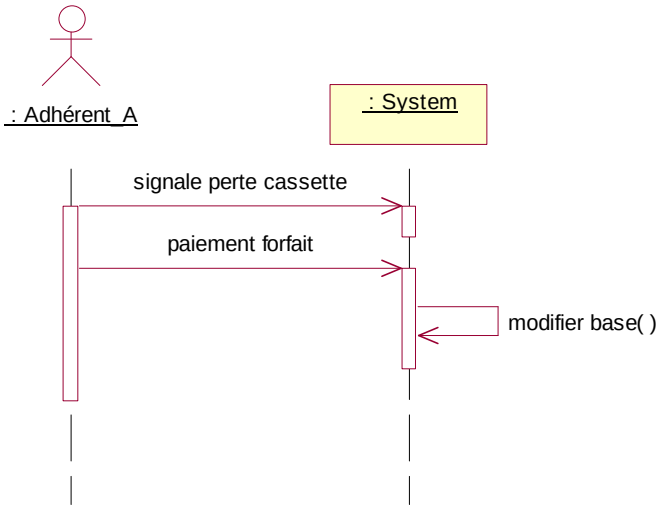
L'adhérent est exclu, il ne peut effectuer de prêts. Ses emprunts sont enregistrés à part (pour ne pas perturber les traitements ordinaires).

Les litiges déclarés ici sont réglés en partie dans la gestion des adhérents.

Nous proposons un scénario pour le dépassement de délai et un scénario pour la perte.



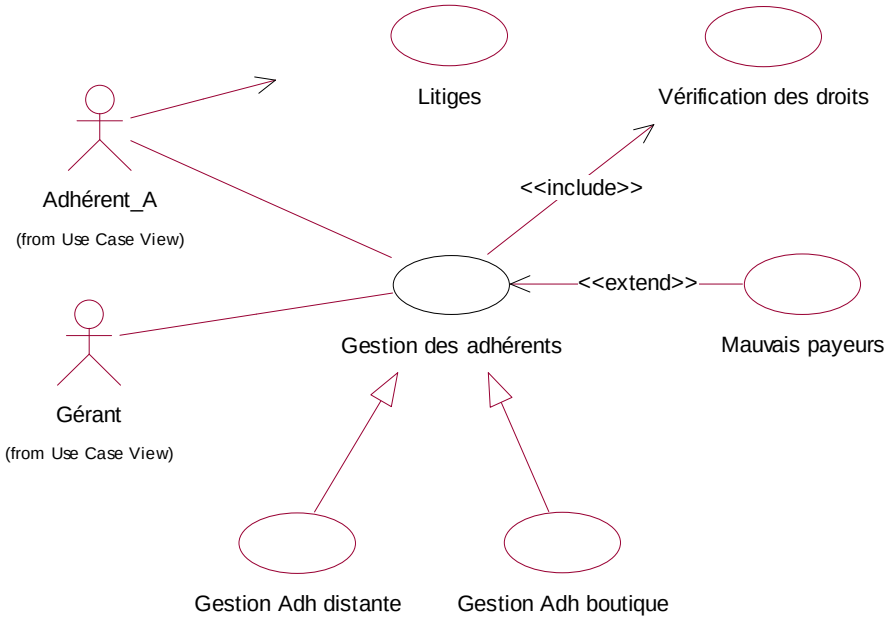
viSCphl.eps



viSCppe.eps

II.1.2 Gestion des adhérents

Dans cette section, nous présentons rapidement la gestion des adhérents. Nous présentons sommairement l'enregistrement des nouveaux adhérents dans l'UC *Gestion des adhérents* et nous décrivons plus en détail l'UC *Vérification des droits*, qui est en rapport avec l'UC *Gestion des prêts*.

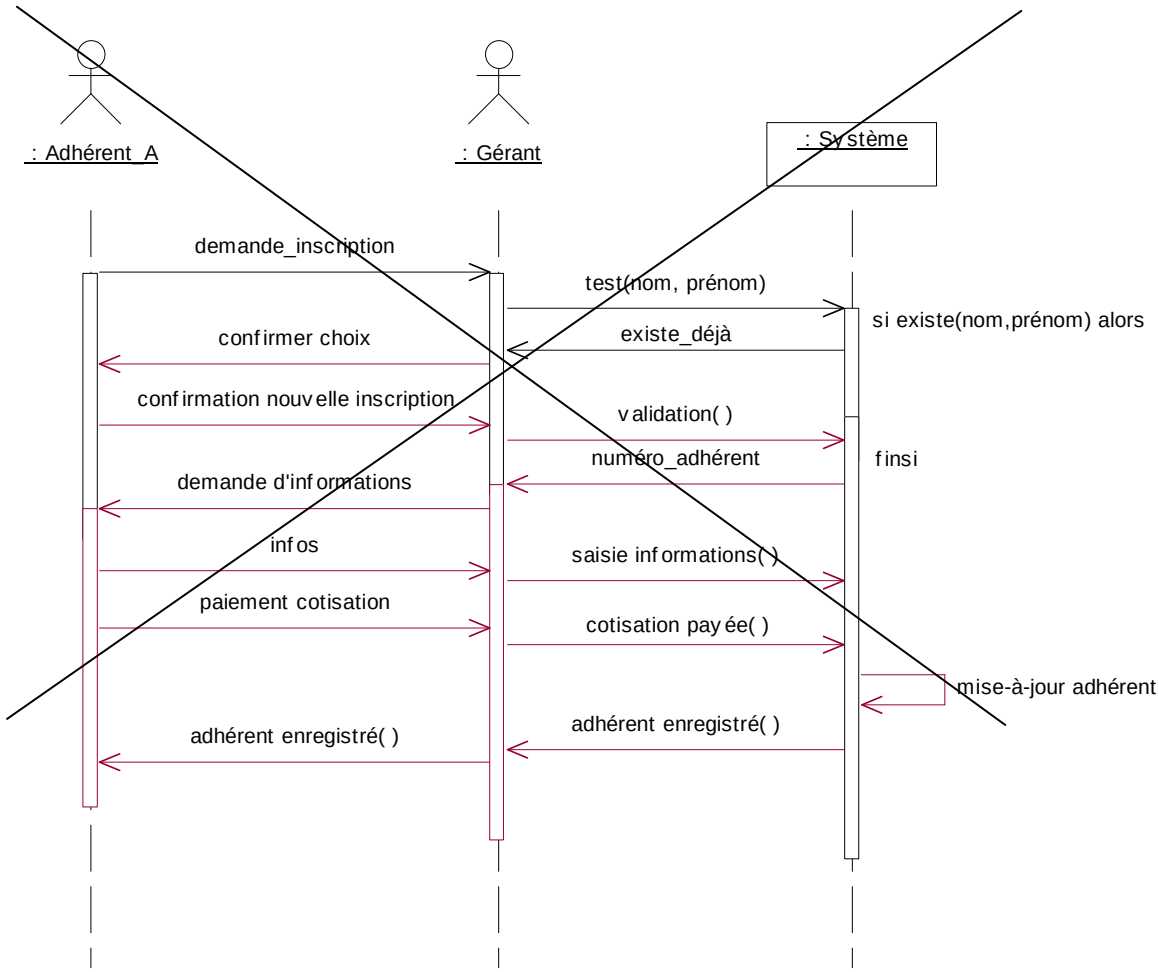


viUCadh.eps

La gestion des adhérents comprend l'enregistrement des nouveaux adhérents (pas la suppression pour l'instant), un traitement spécifique pour les adhérents n'ayant pas réglé les emprunts et leurs cotisations (UC *Mauvais payeurs*) – ajout au texte de base pour illustrer la relation *extend*.

Cas d'utilisation : *Gestion des adhérents*
Acteurs : Gérant, Adhérent **Description générale** La gestion des adhérents comprend principalement l'inscription des adhérents. **Cas :** `Inscription` Un adhérent souhaite s'inscrire. Saisir les noms et prénoms. Faire un test d'existence sur ces deux éléments. En cas d'égalité, valider l'ajout. L'ajout effectif se fait en donnant un numéro d'adhérant, qui servira dans les transactions futures. L'adresse et les coordonnées téléphoniques sont enregistrées. L'adhérent paie sa cotisation. **Cas :** `Paiement cotisation` L'adhérent paie sa cotisation annuelle. La base est mise à jour. **Exceptions** **Cas :** `Adhérent déjà inscrit` précondition : Un adhérent de même nom et prénom est inscrit, l'ajout n'est pas validé. Le traitement est arrêté. **Cas :** `Cotisation ultérieure` précondition : L'adhérent souhaite payer ultérieurement sa cotisation. Le traitement est poursuivi sans ce point.

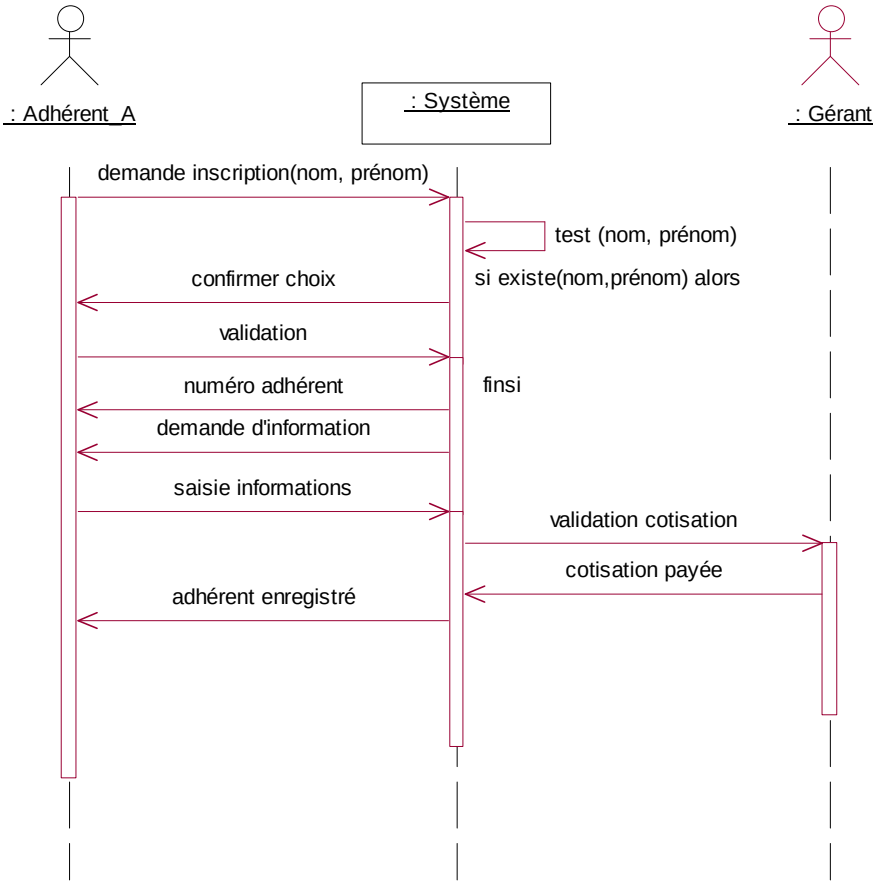
Le scénario suivant illustre le cas général de l'inscription.



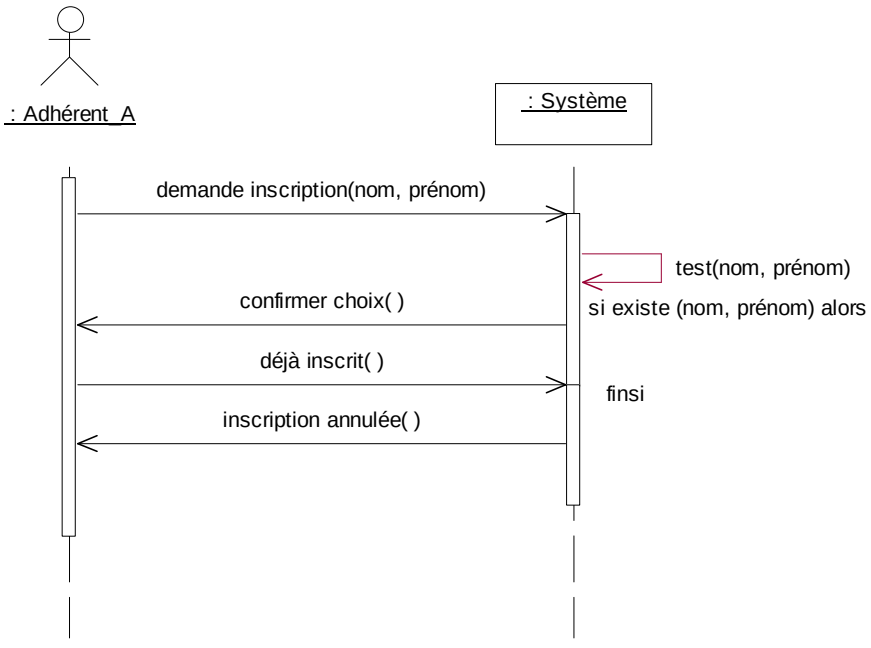
v

iSCinsR.eps

En fait, cette organisation n'est pas souhaitable. Nous avons créé un intermédiaire inutile dans une abstraction de l'inscription réelle. On ne se préoccupe pas de savoir qui dialogue avec le système, mais on se focalise plutôt sur les décisions qui interviennent. On préférera le diagramme suivant de la \lafigure{viSCins}.



viSCins.eps



viSCins1.eps

La gestion des litiges répond entre autres aux problèmes soulevés dans l'UC *Gestion des retards*. Elle comprend aussi la vérification des cotisations annuelles.

La vérification des droits est un bilan sur l'adhérent. Nous aurions pu l'inclure dans l'UC *Gestion des adhérents* mais nous avons considéré que son fonctionnement partagé le rend indépendant.

Cas d'utilisation : *Vérification des droits*

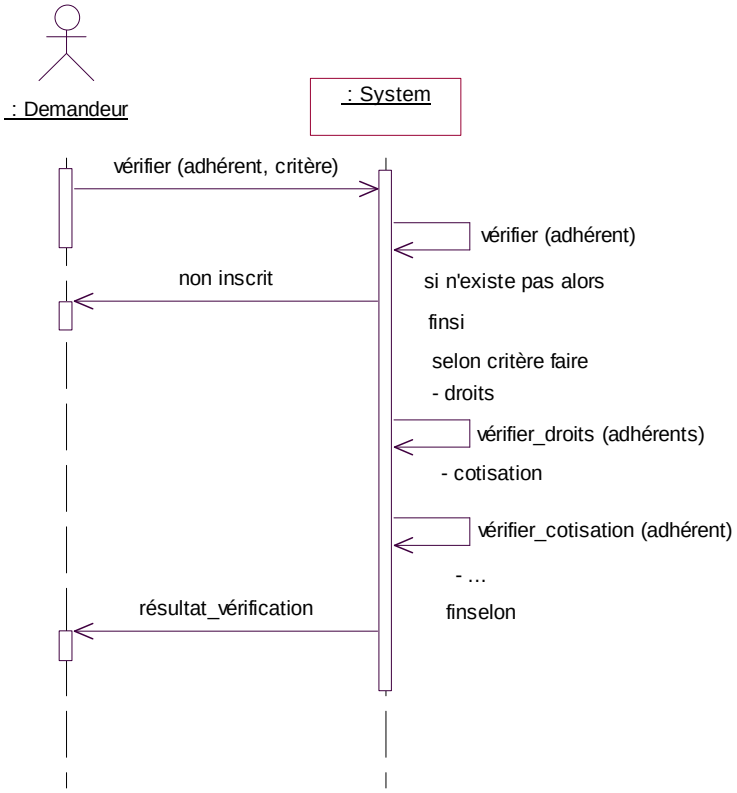
Description générale

La vérification des droits est quasiment un traitement fonctionnel invoqué dans plusieurs interactions avec les adhérents.

Les réponses suivantes sont fournies :

- Adhérent non inscrit.
- Adhérent exclus. L'adhérent a des emprunts en retard d'un an.
- Cotisation non réglée.
- Emprunt impossible : déjà réalisé dans la boutique pour ce jour.
- Emprunt impossible : retard dans le retour d'un prêt.
- Adhérent en règle

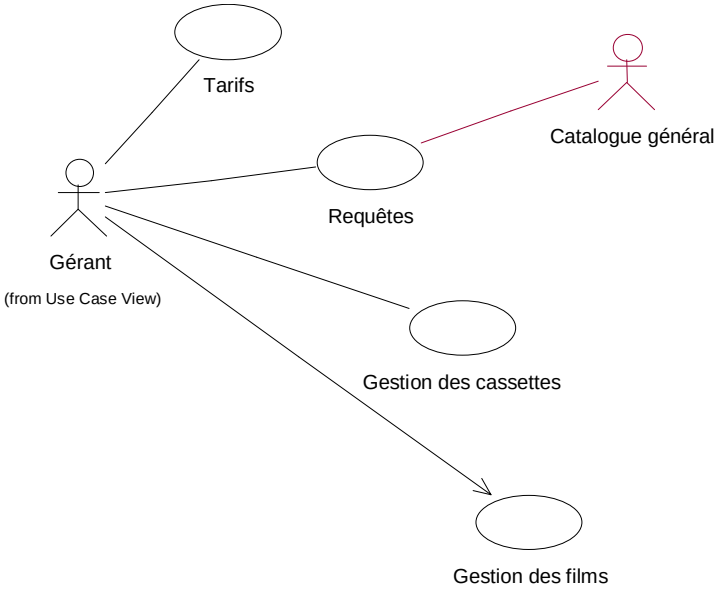
Deux modélisations sont possibles, l'une générant des exceptions et l'autre rendant un résultat. Nous privilégions la seconde au niveau de l'analyse des besoins. Notez que la modélisation de la \lafigure{viSCdroi} est relativement abstraite. Nous avons créé un nouveau rôle (un acteur) pour symboliser l'origine de la demande. Nous aurions pu faire partir le message du système lui-même. Rappelons que l'objectif est de donner une idée de ce qui se passe et non pas l'exacte réalité en termes opérationnels. C'est pourquoi certaines libertés sont prises quand à la modélisation des scénarios et des diagrammes de séquence de l'analyse (voir section suivante).



viSCdroi.eps

II.1.3 Gestion du catalogue

Dans la suite, nous donnons une description sommaire des cas d'utilisation, qui nous serviront à la construction du diagramme des classes.

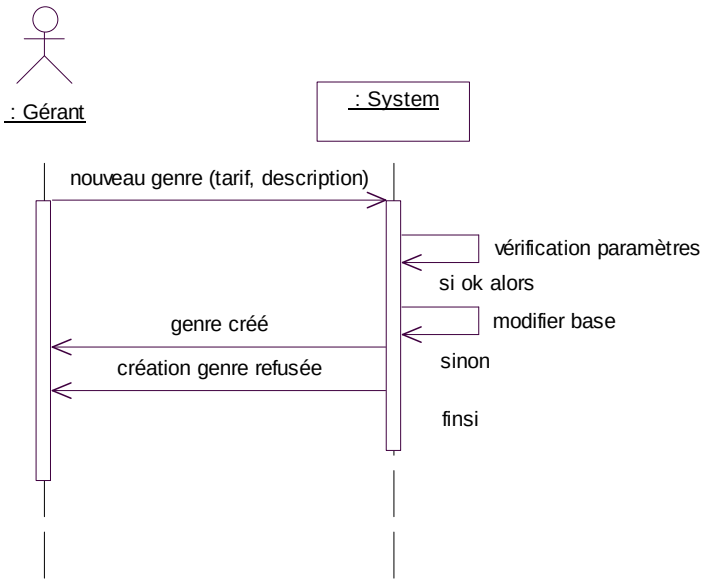


viUCcat.eps

La gestion des tarifs ne prend pas en compte d'éventuelles opérations de fidélisation de la clientèle.

Cas d'utilisation : *Tarifs*
Acteurs : Gérant Invariant : La tarification ne baisse pas. **Description générale** Le tarif est un prix de location à la journée. Il dépend de la catégorie de film (genre) et de la durée du film. La gestion des tarifs comprend principalement l'ajout et la modification de tarifs. **Cas :** `Nouveau genre` Le gérant entre un nouveau genre de film et le tarif associé. **Cas :** `Modification tarif` Le gérant modifie le tarif horaire pour un genre donné.

Illustrons ce cas par le scénario de l'ajout d'un nouveau genre.



viSCtari.eps

Le second UC correspond aux requêtes souhaitées sur la base d'information du magasin.

Cas d'utilisation : *Requêtes*
Acteurs : Gérant **Description générale**

Le gérant peut souhaiter obtenir différentes statistiques sur les boutiques ou sur le club.

Cas : `Liste des cassettes d'une boutique`

Le gérant demande la liste des cassettes d'une boutique dont il précise la référence (numéro).

Cas : `Liste des films disponibles`

Le gérant demande la liste des films dont au moins un exemplaire est disponible.

Cas : `Liste des films du catalogue`

Le gérant demande la liste des films du catalogue (programme externe invoqué : *Catalogue général*).

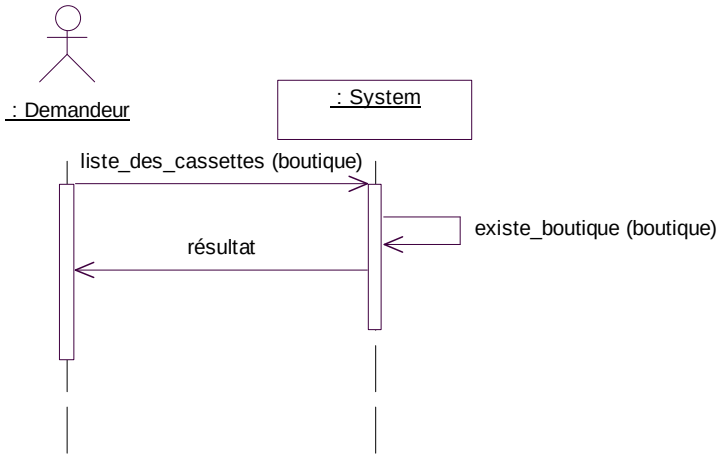
Cas : `Liste des emprunts d'un adhérent`

Le gérant demande la liste des emprunts d'un adhérent dont il précise la référence (numéro).

Cas : `autres`

Nous n'énumérons pas toutes les requêtes possibles.

Nous illustrons ce cas par le scénario d'une requête représentative, les autres sont sur le même modèle. Dans cet exemple, nous avons aussi utilisé le rôle générique (acteur) *Demandeur* pour désigner l'origine de la demande (ici gérant ou boutiquier, ou adhérent si on suppose des requêtes à distance).



viSCreq.eps

Cas d'utilisation : *Gestion des cassettes*

Acteurs : Gérant

Description générale

La gestion des cassettes comprend les ajouts, suppression d'exemplaires de cassettes.

Cas : `Nouvelle cassette`

Le gérant entre un nouvel exemplaire d'une cassette. Le gérant entre les nouvelles informations associées (boîtier, type de bande, dimension, état, type (VHS, PAL)...). Une référence est donnée en résultat.

Cas : `Modification cassette`

Le gérant entre les nouvelles informations associées (boîtier, type de bande, dimension, état, type (VHS, PAL)...).

Cas : `Suppression cassette`

précondition : La cassette doit être en boutique.

Le gérant supprime un exemplaire d'une cassette. Le film n'est pas modifié.

Exceptions

Cas : `Film non enregistré`

précondition : Le film de la cassette à ajouter n'est pas enregistré.

Le film est ajouté, appel à l'UC Gestion des films.

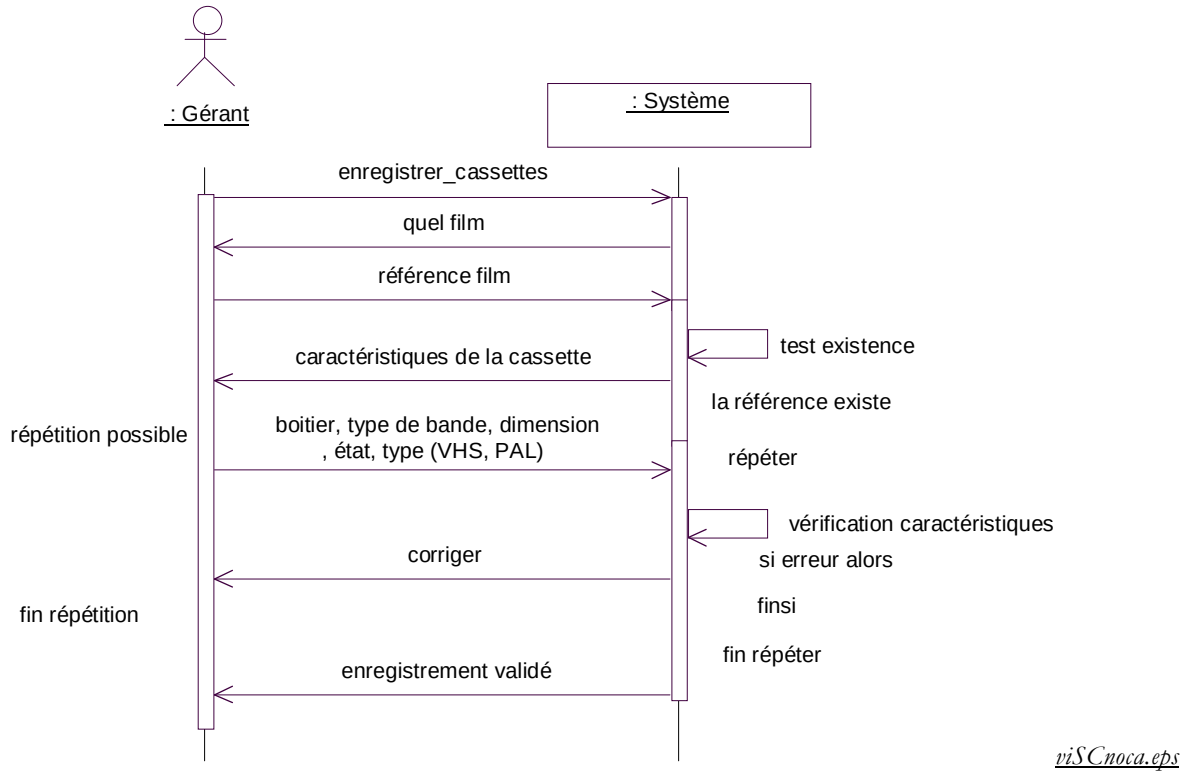
Cas : `Suppression cassette empruntée`

précondition : La cassette à supprimer est empruntée.

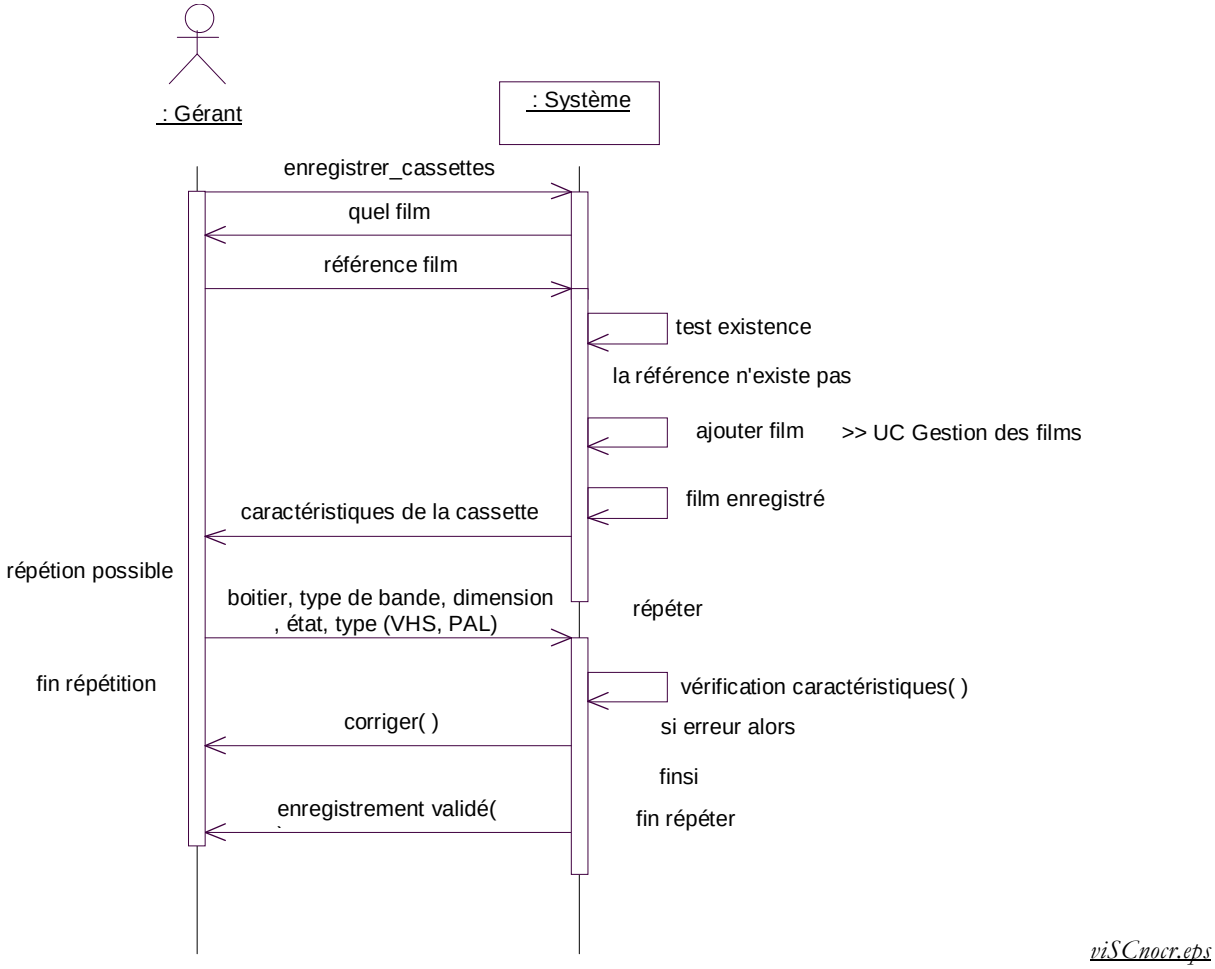
Le traitement est arrêté.

Notez que certains cas d'utilisation peuvent s'invoquer mutuellement. Par exemple, l'ajout d'une cassette d'un film non enregistré implique d'enregistrer d'abord le film. Etablir une relation de dépendance entre les deux est gênant dans la mesure où seule une fonctionnalité est invoquée. C'est encore plus vrai lorsque deux UC s'appellent mutuellement (par exemple, si on rentre un film alors on peut rentrer aussi un exemplaire). Plusieurs solutions sont possibles : extraire un sous-cas d'utilisation commun (exemple : vérification des droits), créer des relations de dépendance avec un nouveau stéréotype ou enfin faire abstraction de ceci au niveau du diagramme des classes. Nous retiendrons la dernière solution.

Nous illustrons ce cas d'utilisation par deux scénarios pour l'enregistrement de nouvelles cassettes (le film existe/le film n'existe pas). Nous avons volontairement développé les interactions mais concrètement les envois de messages paramétrés condensent plusieurs interactions. Cela sort du contexte puisqu'il s'agit d'interfaces homme/machine.



Lorsque le film n'existe pas, on invoque une situation de l'UC *Gestion des films*. Nous masquons la complexité en faisant un appel de l'UC. En fait, la prise en compte des exceptions de l'UC appelée amène un traitement particulier au retour de l'invocation. Par exemple, si le film n'a pu être enregistré, alors il faut abandonner l'enregistrement des cassettes.



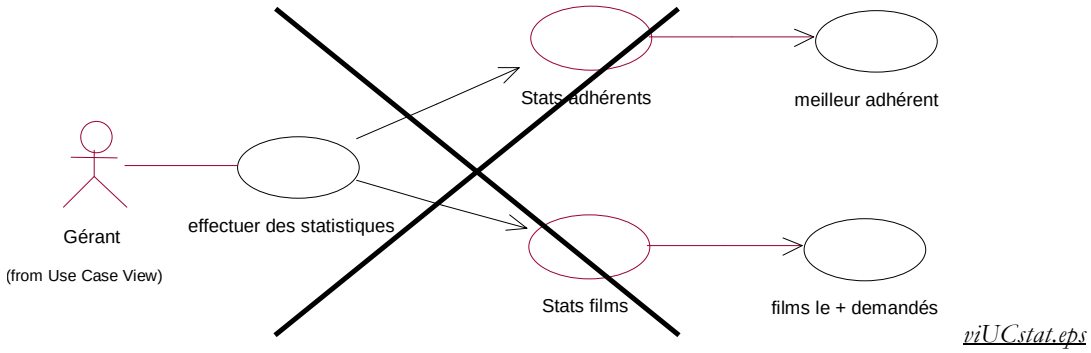
viSCnocr.eps

Cas d'utilisation : *Gestion des films*
Acteurs : Gérant **Description générale** La gestion des films comprend principalement l'ajout et la modification et la suppression de films, elle est fortement liée à la gestion des tarifs et celle des cassettes. **Cas :** `Nouveau film` Le gérant entre un nouveau fil, il rentre les informations associées (auteur, réalisateur, acteurs, durée, date, version...). Une référence est produite en résultat. **Cas :** `Modification film` Le gérant entre les nouvelles informations associées (auteur, réalisateur, acteurs, durée, date, version...). **Cas :** `Suppression film` précondition : Aucunes cassettes du film à supprimer n'existe. Le gérant supprime un film en donnant sa référence. **Exceptions** **Cas :** `Genre non enregistré` précondition : Le genre du film à ajouter n'est pas enregistré. Le genre est ajouté, appel à l'UC Gestion des tarifs. **Cas :** `Suppression film impossible` précondition : Des cassettes du film à supprimer existent. Le gérant modifie le tarif horaire pour un genre donné.

Nous ne proposons pas de scénarios ici, ils sont similaires à ceux de la gestion des cassettes.

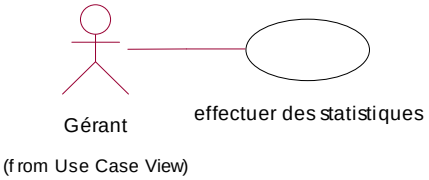
II.1.4 Statistiques

La présentation suivante n'est pas acceptable car elle dénote une organisation fonctionnelle, qui n'est pas souhaitable au niveau de l'analyse des besoins (ce n'est pas un enchaînement de menus).



viUCstat.eps

Nous nous contenterons du cas suivant.



viUCstaR.eps

Cas d'utilisation : *Statistiques*
Acteurs : Gérant **Description générale** Le gérant peut souhaiter obtenir différentes statistiques sur les boutiques ou sur le club. **Cas :** `Meilleur client` Le gérant édite la liste des clients, classés par chiffre d'affaire. **Cas :** `Box office` Le gérant édite la liste des films par nombre de clients l'ayant emprunté. **Cas :** `autres` Nous n'énumérons pas toutes les listes possibles.

Nous ne proposons pas de scénarios ici, ils sont similaires à ceux de l'UC *Requêtes*.

II.1.5 Complément

Dans la plupart des cas présentés, il est possible d'annuler l'opération, ceci représente une exception que nous notifions ici pour ne pas surcharger les diagrammes.

II.2 Diagrammes d'activités

Certains utilisent les diagrammes d'activités (avec des couloirs) pour modéliser le fonctionnement des grandes opérations du système. On se rapproche dans cette optique des MCT de Merise.

A notre avis, cette utilisation ne correspond pas à l'essence même des diagrammes d'activités, qui décrivent des activités à l'intérieur des états des diagrammes états transitions. De plus, on retombe dans le travers de l'analyse fonctionnelle classique. Toutefois, notons que l'emploi de ces diagrammes avec les cas d'utilisation (ils doivent alors remplacer les scénarios, sinon on multiplie les représentations au risque d'introduire des incohérences) permet d'utiliser la démarche de la méthode Fusion : découpage des grandes fonctions sur les objets. Ceci nous renvoie aux préoccupations de l'étape suivante, l'analyse.

Nous préconisons le schéma innovant « `cas d'utilisation + scénarios` » en analyse des besoins.

II.3 Conclusion

Nous avons donné une idée générale du besoin. Les descriptions peuvent parfois apparaître trop ou pas assez détaillées. Le choix du niveau d'abstraction est une difficulté fondamentale de ce type de modélisation. Il est important de ne pas surcharger les diagrammes de détails pour que l'utilisateur puisse valider aisément et éviter orientation précise de l'implantation (le prototypage sert à ça). Inversement, un manque de détails conduit à une sous-spécification et laisse place à des interprétations différentes (ambiguïtés). Il nous semble important de Notez sous forme de commentaires les répétitions ou les analogies (par exemple entre les requêtes et les statistiques) pour condenser la description. La remarque sur les annulations de la section précédente va aussi dans ce sens.

L'étape suivant va consister à préciser les interactions à l'intérieur du système.

III. ANALYSE

L'objectif de l'analyse est de mettre en évidence les entités du système. Pour cela, nous précisons les scénarios en ajoutant des détails sur la prise en compte des interactions à l'intérieur du système. En particulier, on fait apparaître des objets qui prennent en charge une partie du travail. De ces objets et de leurs liens, on extrait le diagramme des classes, qu'on enrichit avec la connaissance du domaine du problème. Ce diagramme des classes est ensuite affiné via l'héritage et les navigations.

III.1 Diagrammes de séquences et de collaborations

L'objectif est d'affiner les modèles de l'analyse des besoins en introduisant quelques rouages essentiels du système à implanter et notamment la prise en compte de l'interface et du contrôle. Pour mettre en évidence les interactions à l'intérieur du système, on a besoin d'exprimer comment les différents objets interagissent. A ce niveau, on décide (choix conceptuels) d'une interface avec les autres acteurs et du contrôle des interactions. Nous reprenons ici la classification de JACOBSON, reprise dans les documents sur UML : objet interface, objet de contrôle et objets métiers (ou entité).

Comme dans l'étape précédente, nous nous efforcerons de rester abstraits. Par exemple, on peut avoir un objet fenêtre de saisie des emprunts sans préciser comment se présente ou s'implante cette fenêtre. Nous ne nous préoccupons pas à ce niveau du contrôle effectif (système réparti, client/serveur), du gestionnaire d'interface (système hôte Windows, X-Window…), de la persistance (base de données). En nous référant à Merise (mais attention ce n'est qu'une image), nous pouvons dire qu'on se situe à un niveau organisationnel.

Les diagrammes de collaborations sont sensiblement équivalents aux diagrammes de séquences mais ils mettent en évidence la structure d'objets, support des interactions.

> Nous utilisons l'annotation '_id' pour désigner un identifiant d'objet. Dans les modèles entités-association, il s'agit d'une clé, faisant partie de l'entité. Ici, aucune hypothèse n'est faite à ce sujet.

a) Interface

L'interface donne une idée de la communication entre le système et les acteurs extérieurs. Nous ajoutons des objets interface pour faire référence aux interactions externes au système. Nous utilisons le stéréotype <<IHM>> pour ne pas confondre avec le stéréotype <<interface>>, qui correspond à la partie visible (exportée) d'une classe.

Dans ce cas, les résultats des opérations sont affichés sur la fenêtre et non transmis directement à l'acteur extérieur. Il y a lecture implicite des informations affichées.

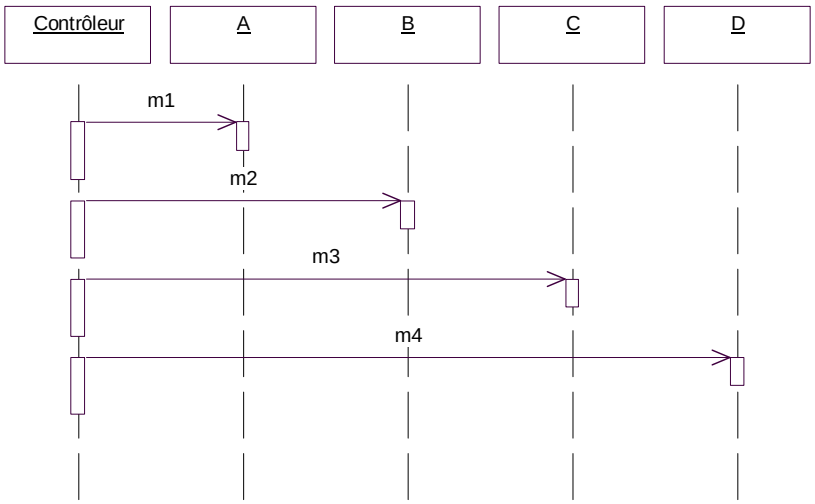
b) Précision

A ce niveau de conception, nous utilisons la notation complète des diagrammes de séquence : spécialisation des interactions (synchronisme ou asynchronisme, résultat), contraintes temporelles, gestion d'objets (création, suppression), flots parallèles d'un objet, structures de contrôle (gardes, itérations), délais temporels de transmission, envois simultanés (multiplexage).

Même si l'algorithmique utilisée doit rester abstraite, il faut pouvoir donner suffisamment de précision. Nous avons introduit ici des traitements d'exception en plus de l'algorithmique classique (alternatives, répétitions…).

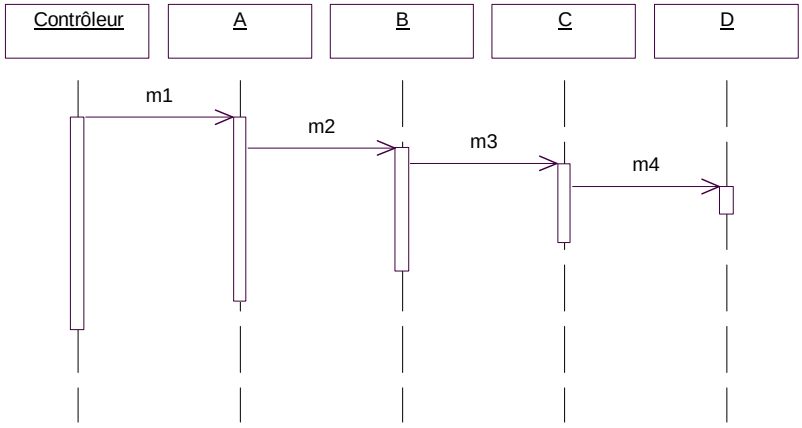
c) Organisation du contrôle

Dans l'ouvrage De Merise à UML \cite{uml_kettani}, deux organisations sont préconisées : centralisée ou avec délégation.



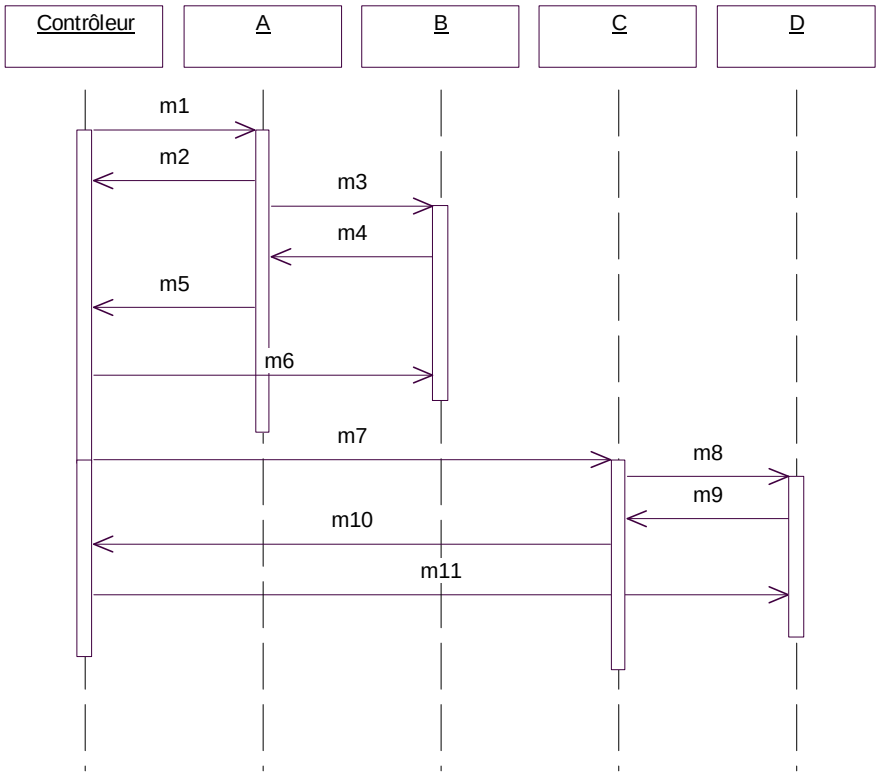
diDScent.eps

Le contrôle est centralisé par un objet qui maîtrise l'interaction.



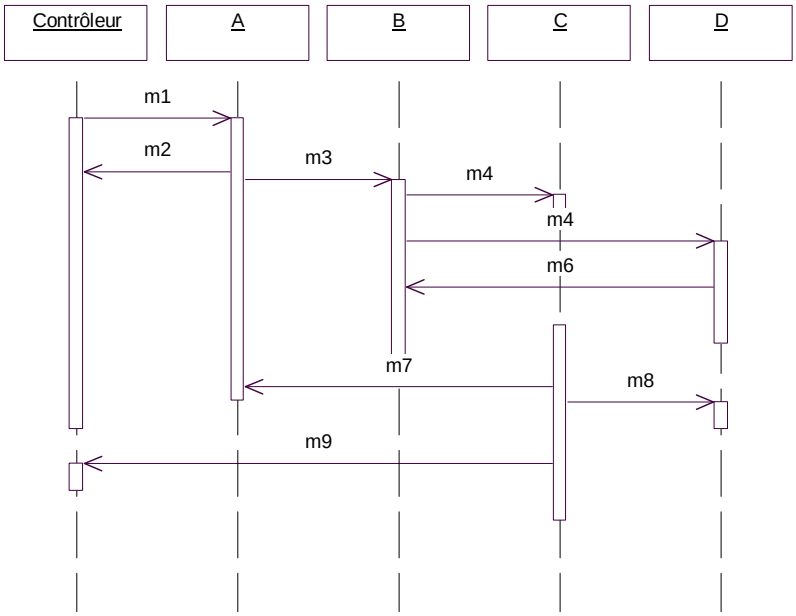
diDSdist.eps

Le contrôle est décentralisé par délégation. Plus généralement les diagrammes faisant apparaître une organisation du contrôle sont souhaitables. Dans le schéma suivant, les relations clients/serveurs dirigent l'interaction.



diDSdiv1.eps

Par contre, les diagrammes au contrôle décentralisé et anarchique ne sont pas souhaitables car difficilement vérifiables.

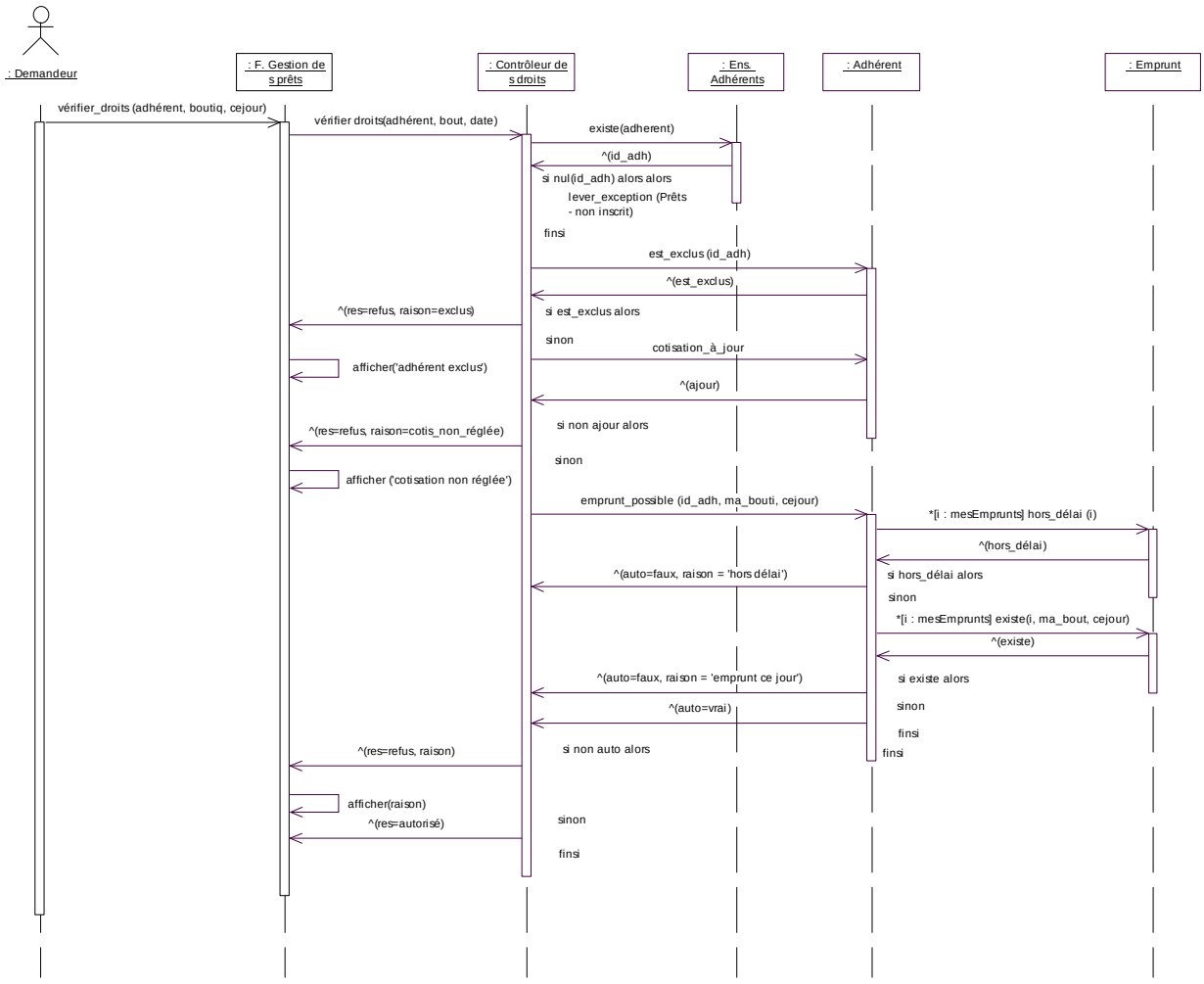


diDSdiv2.eps

> Attention, les conseils précédents n'empêchent en rien le parallélisme en scénarios. La gestion de la concurrence est traitée dans la conception détaillée.

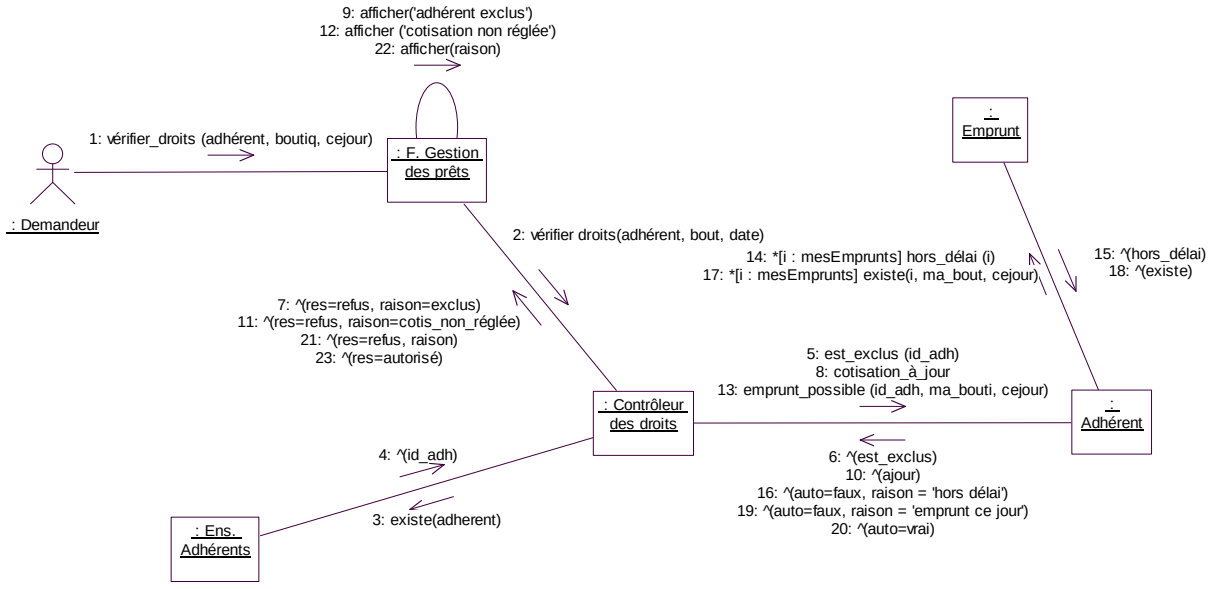
III.1.2 Gestion des adhérents

Nous développons le scénario de la vérification des droits de \lafigure{viSCdroi.eps} par le diagramme de séquence de \lafigure{viDSdroi.eps}. Notez la représentation des envois de messages synchrones (envoi + réponse) sous forme ^*réponse*. La notation UML habituelle du retour de valeur est le trait discontinu. Nous avons représenté une exception et plusieurs structures de contrôle. On distingue quatre couches d'objets : les acteurs externes (*Demandeur*), les objets de l'interface (*F. Gestion des prêts*), les objets de contrôle et les objets métiers. En particulier, la fenêtre symbolise la communication avec l'utilisateur. Ainsi, certaines interactions du scénarios se représentent sous forme de messages (afficher).



viDSdroi.eps

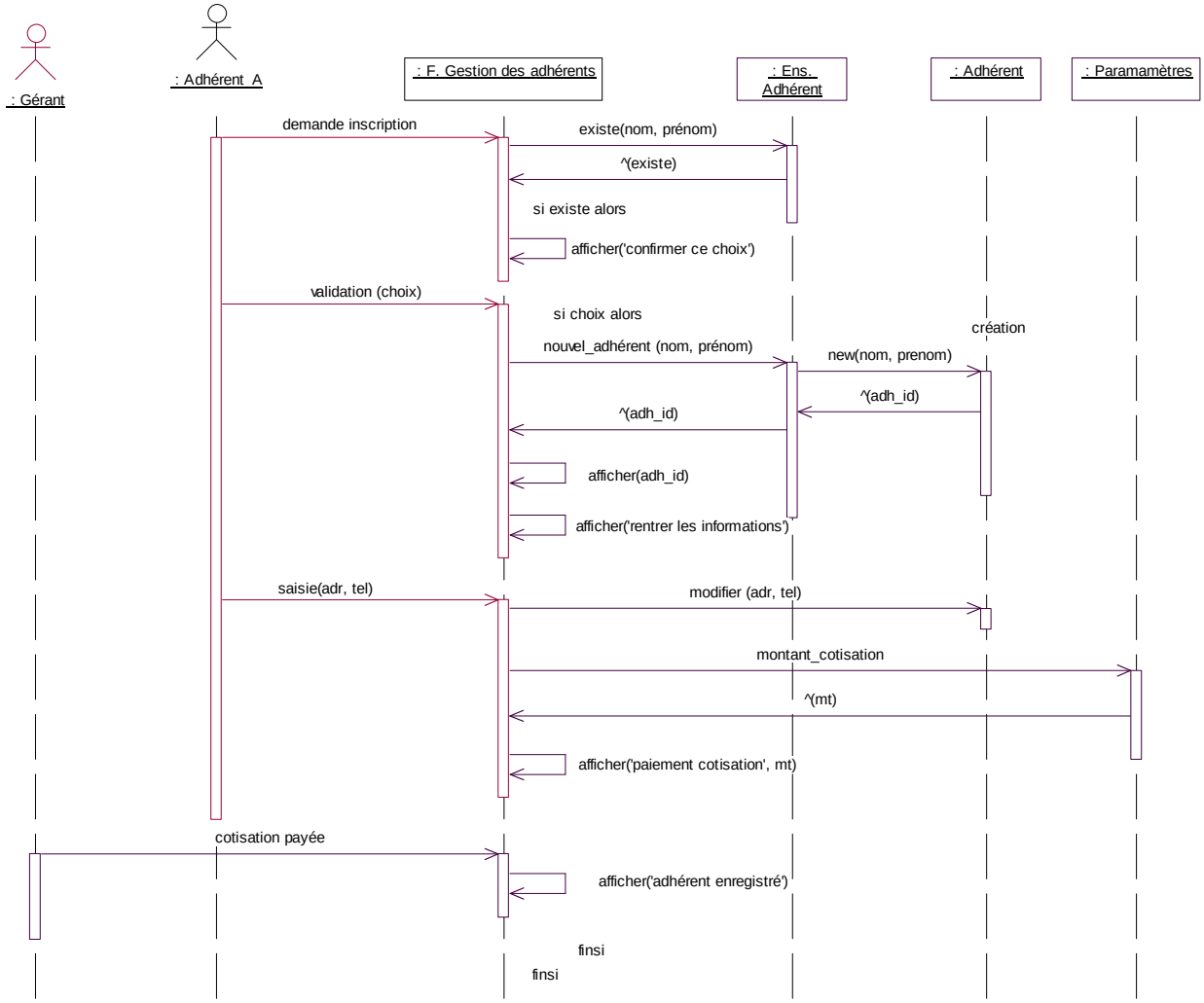
On en déduit le diagramme des collaborations suivantes (\lafigure{viDCdroi.eps}).



viDCdroi.eps

Le diagramme suivant traite l'inscription d'un nouvel adhérent (enrichissement du scénario de la \lafigure{viSCins.eps }). Contrairement au diagramme de la \lafigure{viDSdroi.eps}, nous avons regroupé ensemble l'objet de contrôle et l'objet

interface. L'objet interface prend en charge la coordination de l'activité. En fait, il faudrait raisonnablement faire le même choix pour toute l'analyse.

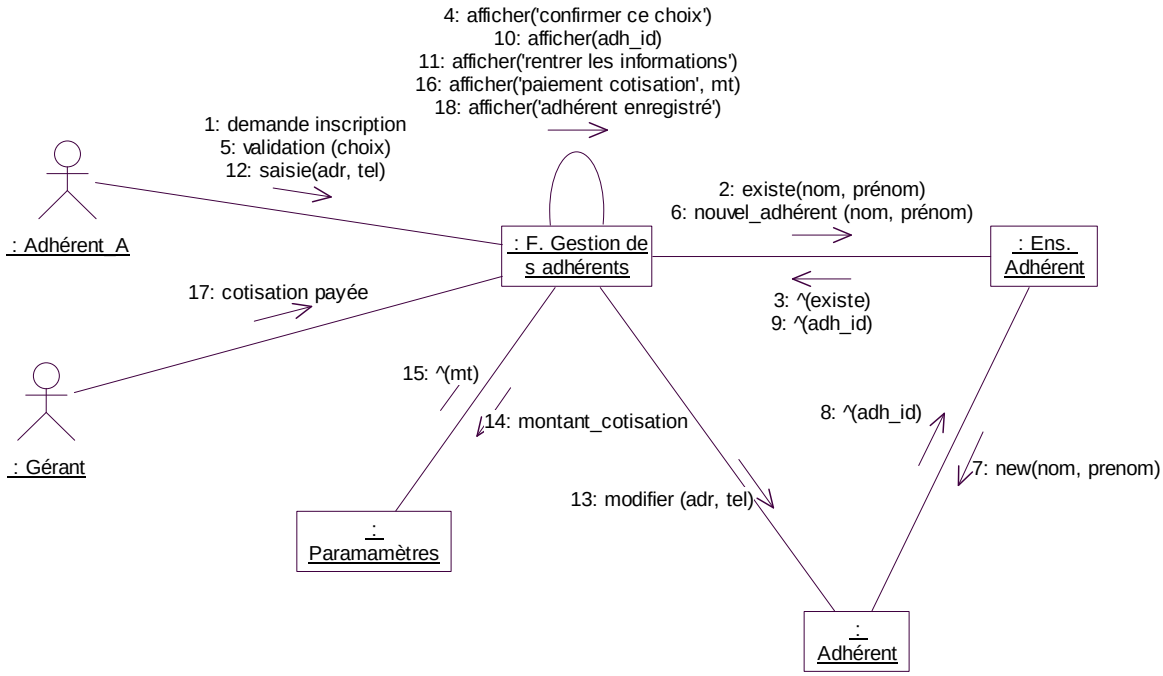


viDSins.eps

> Notez que cette description est proche d'un algorithme. En fait, plus les interactions sont simples ou atomiques, plus la description algorithmique permet d'exprimer le besoin précis.

Nous avons défini la création d'instance par une création d'objets (*Adhérent*) et le stockage dans un ensemble d'instances de la classe (*Ens. Adhérents*). Ainsi, nous pourrons retrouver cet objet en interrogeant l'ensemble. En quelque sorte, nous avons ici émulé un comportement de métaclasse. L'idée est de mettre en valeur le fait que l'objet créé est rattaché à quelque chose dans le système. Nous n'avons pas ce problème pour la création d'emprunt (bien que nous ayons pu aussi modéliser l'ensemble des emprunts) car l'emprunt est relié à l'adhérent. Cependant, dans un soucis de cohérence, il faudrait aussi créer un ensemble d'emprunts et ainsi pouvoir faire des requêtes sur les emprunts, indépendamment des cassettes, des adhérents ou des boutiques.

> Notez qu'on retrouve là la notion de ligne dans une table de base de données relationnelle. L'ensemble représente la table, l'objet représente la ligne dans la table. Ce point est essentiel lorsqu'on traite la persistance.



viDCins.eps

III.1.3 Gestion des prêts

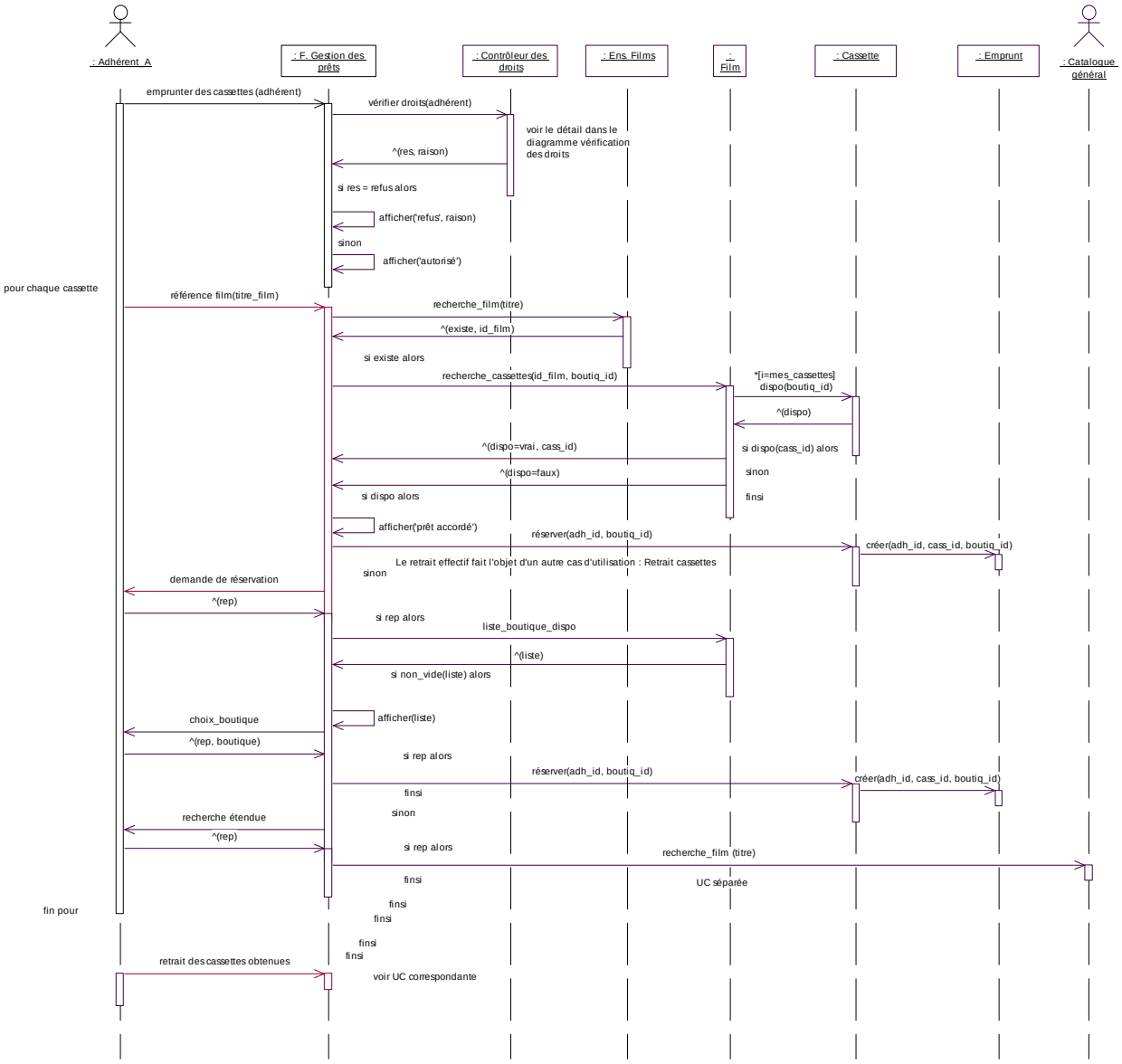
On suppose un formulaire de saisie (ou une fenêtre, si on considère une interface graphique).

Dans le diagramme de séquences de l'emprunt, nous avons regroupé le contrôle et l'interface dans l'objet instance de la classe *F. Gestion des prêts* et nous faisons appel à un autre diagramme pour la vérification des droits. Plusieurs alternatives s'offrent à l'organisations des interactions. Prenons l'exemple du choix du film.

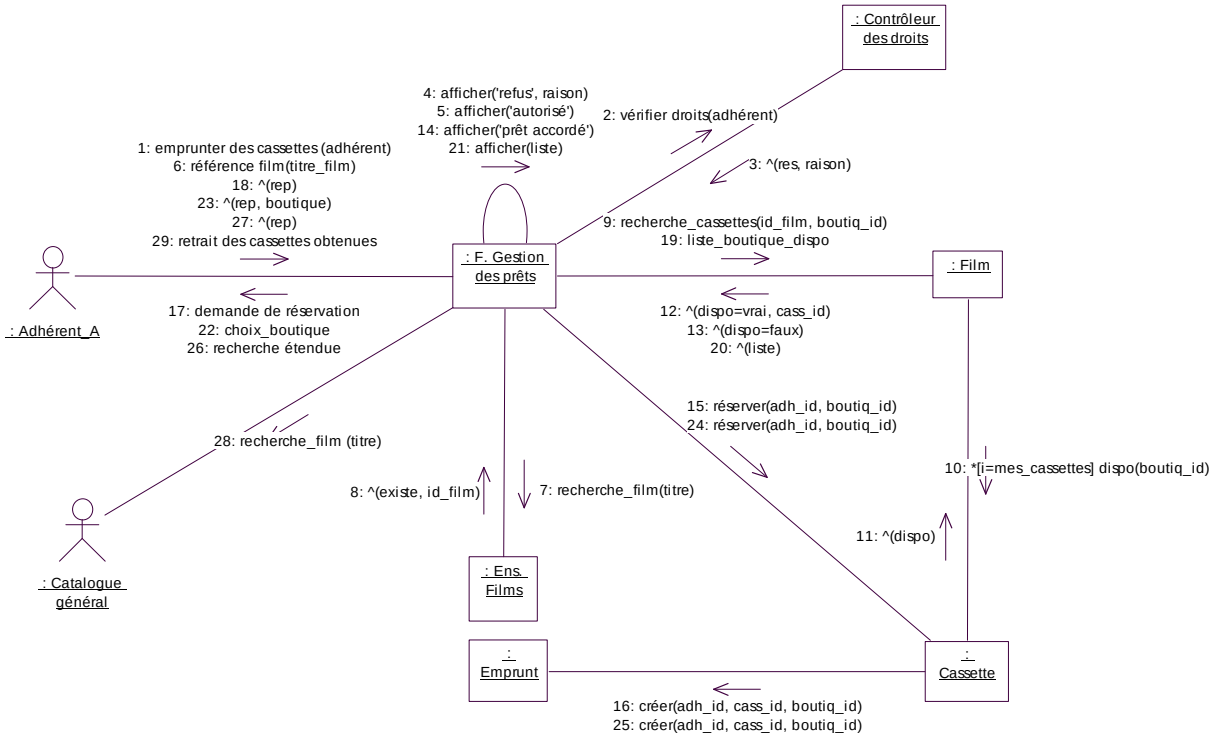
- Sélection
  - Sélection dans une liste
  - Recherche en fonction du titre ou d'une référence
- Présentation
  - Diagramme de séquence inclus
  - Diagramme de séquence à part car partagé par plusieurs diagrammes

Dans l'exemple de la \lafigure{viDSemp1.eps}, nous choisissons d'intégrer la recherche du film par son titre. Notez que l'envoi de message 'réserver cassette' unifie les emprunts boutique et à distance.

> Notez que nous utilisons un nom générique pour les fenêtres en fonction de leur domaine. L'interface précise fera appel à des différentes fenêtres et différents fonctions (menus, boutons…).



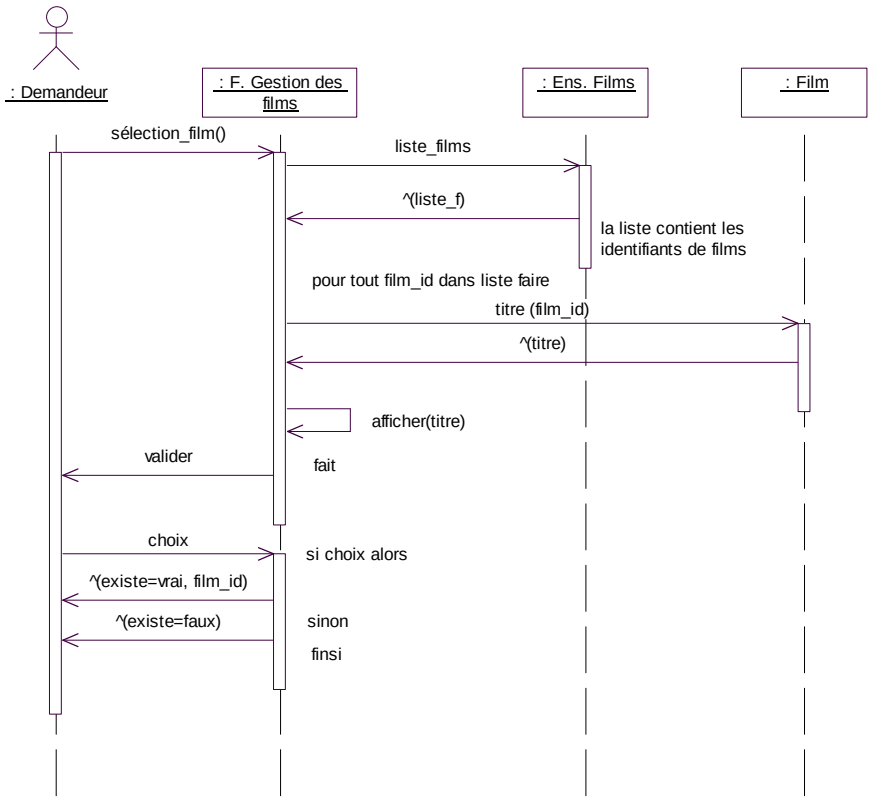
viDSemp1.eps



viDCemp1.eps

Dans l'exemple de la \lafigure{viDSemp2.eps}, nous choisissons d'intégrer la recherche du film et de la cassette par sélection dans une liste. On invoque ainsi d'autres scénarios, nous avons choisi une interface externe par fenêtre et non un objet de contrôle pour mettre en évidence les interactions avec le boutiquier ou les adhérents.

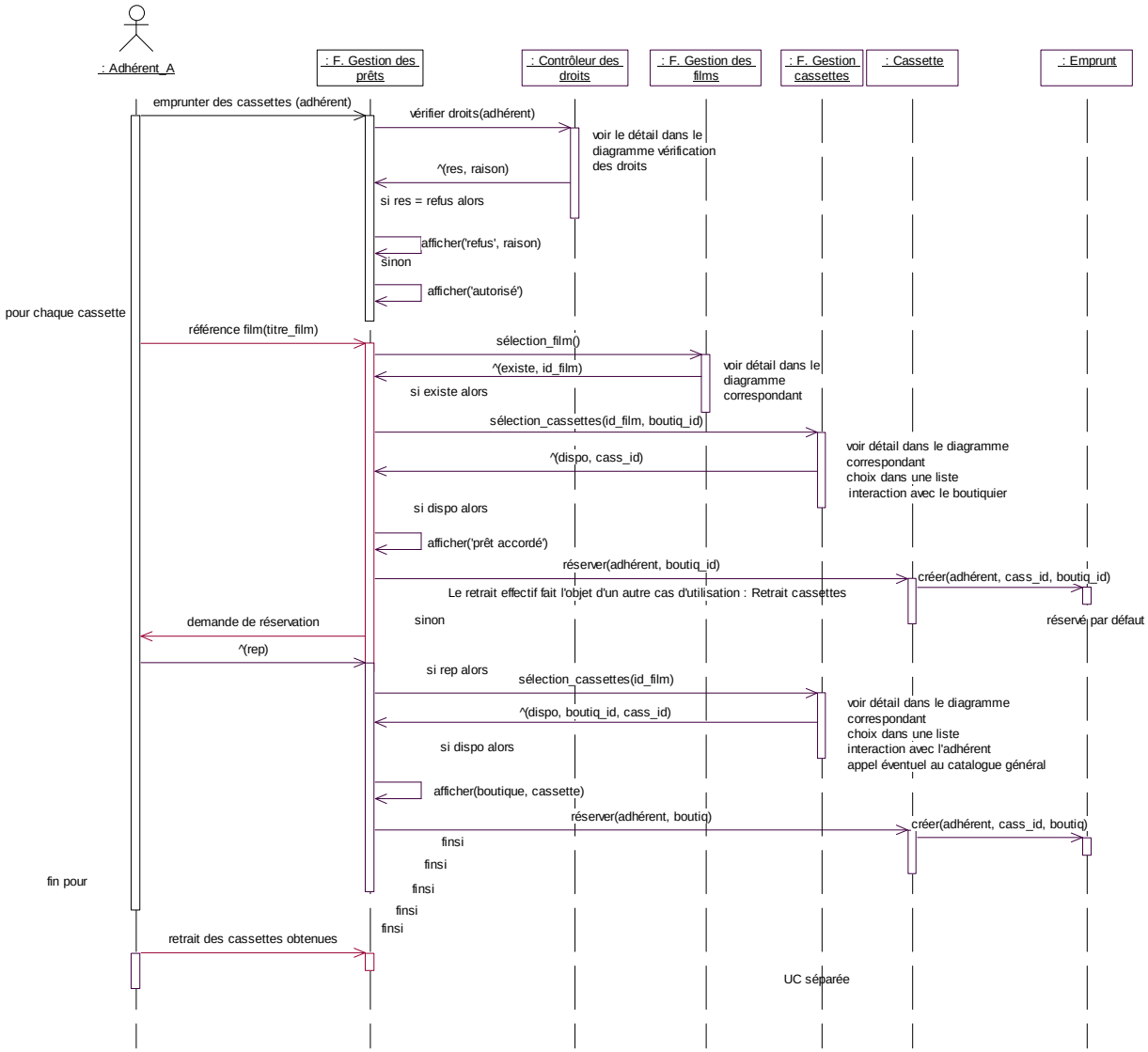
> Notez que dans ce cas, des résultats sont rendus. Ce qui n'était pas le cas directement dans les interactions immédiates avec les acteurs extérieurs.



viDSrefi.eps

Le demandeur est un rôle générique (acteur extérieur ou objet de contrôle d'un autre diagramme de collaboration).

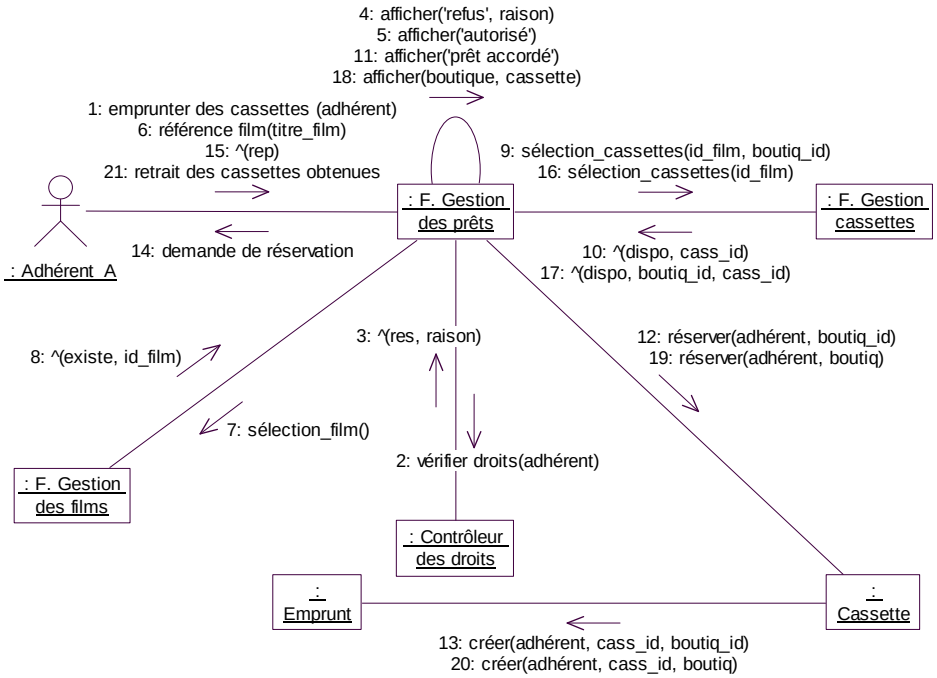
L'affichage des titres met en évidence un traitement séquentiel mais on peut aussi construire une liste et l'afficher ensuite, c'est une décision de conception détaillée, fonction du système d'interface retenu.



viDSemp2.eps

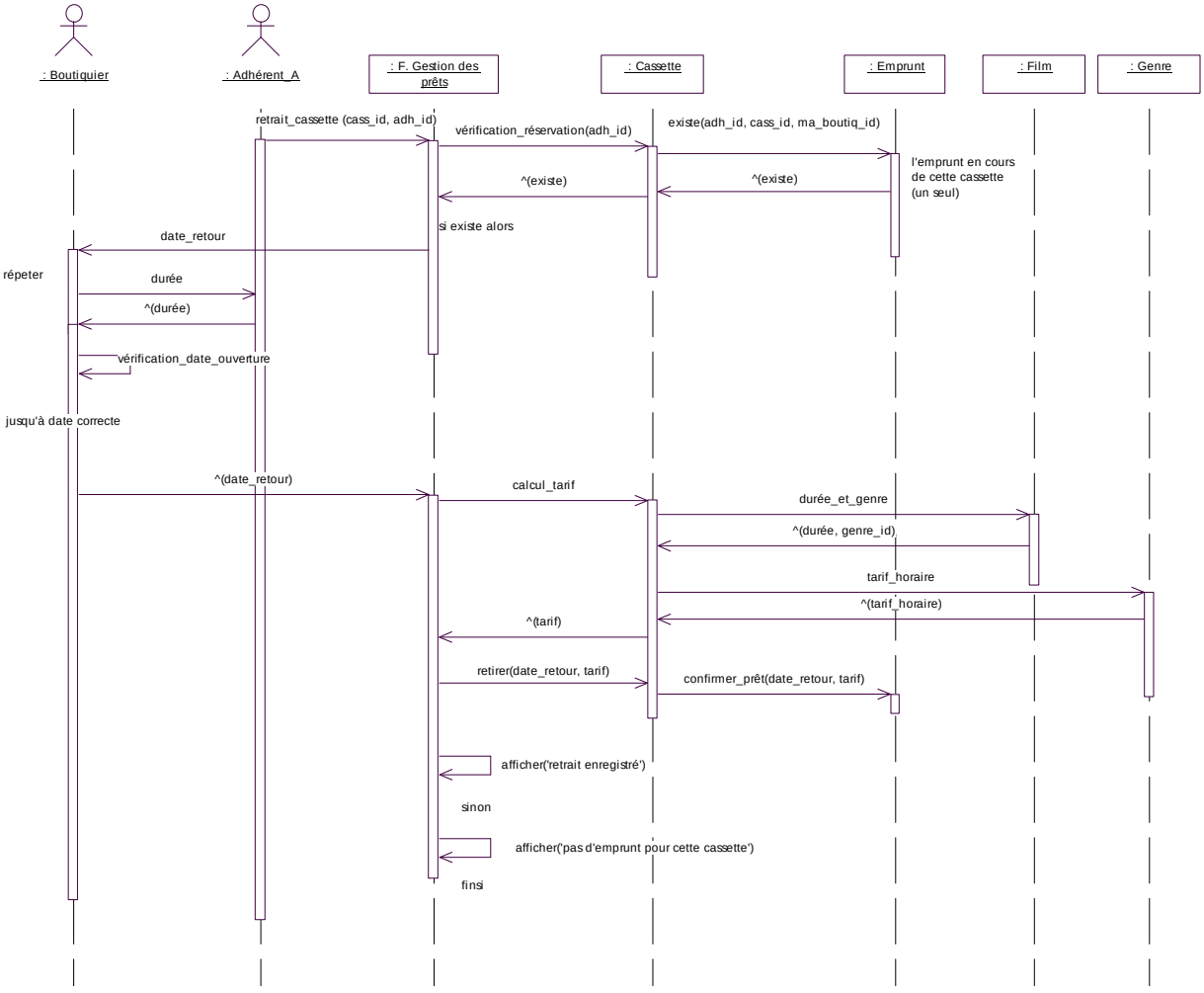
La recherche étendue est maintenant intégrée dans la recherche de cassettes.

Notez que nous utilisons différentes variantes pour représenter les interactions avec les acteurs extérieurs (envoi de message, affichage et événement externe). Là encore, toute liberté est laissée à la conception détaillée des interfaces pour implanter cette communication.

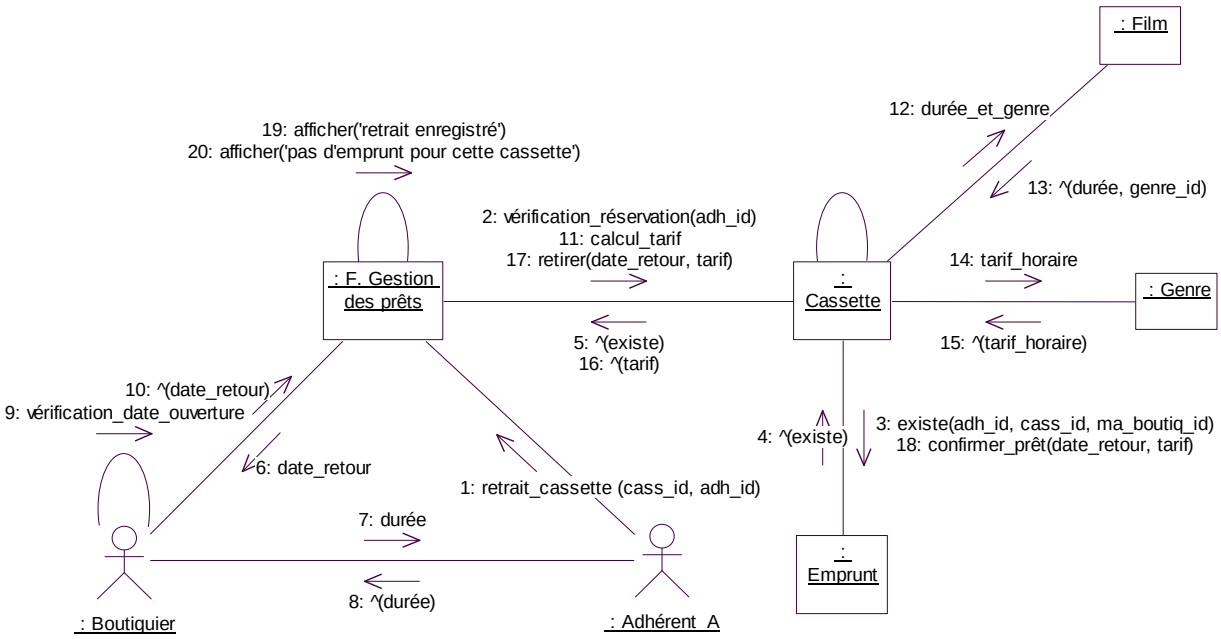


viDCemp2.eps

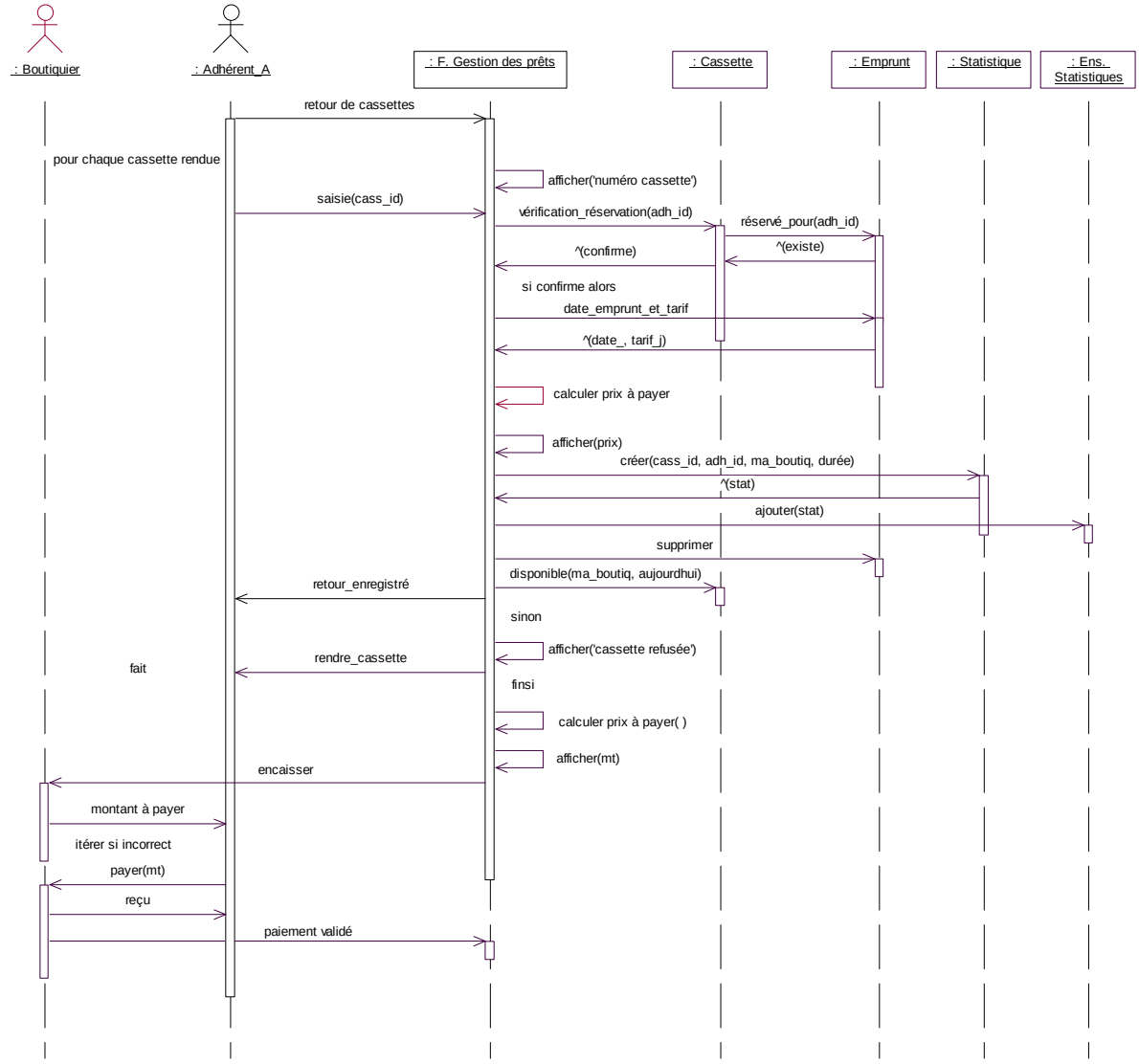
Le diagramme de la \lafigure{viDSretr} met en évidence le retrait effectif de la cassette.



viDSretr.eps

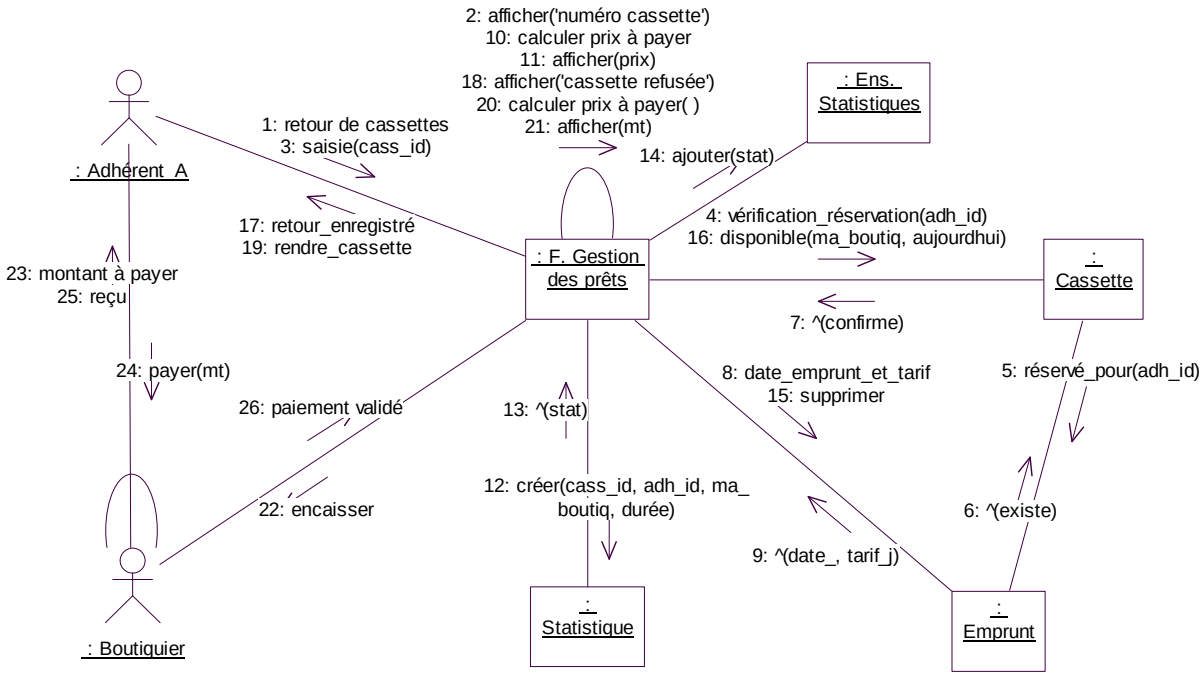


viDCretr.eps



viDSret.eps

Le diagramme de la \lafigure{viDSret} met en évidence le retour des cassettes. Nous avons centralisé le contrôle de la mise à jour de la base. Cela aurait pu être délégué à l'objet cassette. Notez que pour respecter la contrainte de non-emprunt d'une cassette disponible, nous devons supprimer l'emprunt avant de rendre disponible la cassette (cardinalités 0..1 dans le diagramme des classes). Le règlement est manuel mais on notifie sa réalisation correcte.

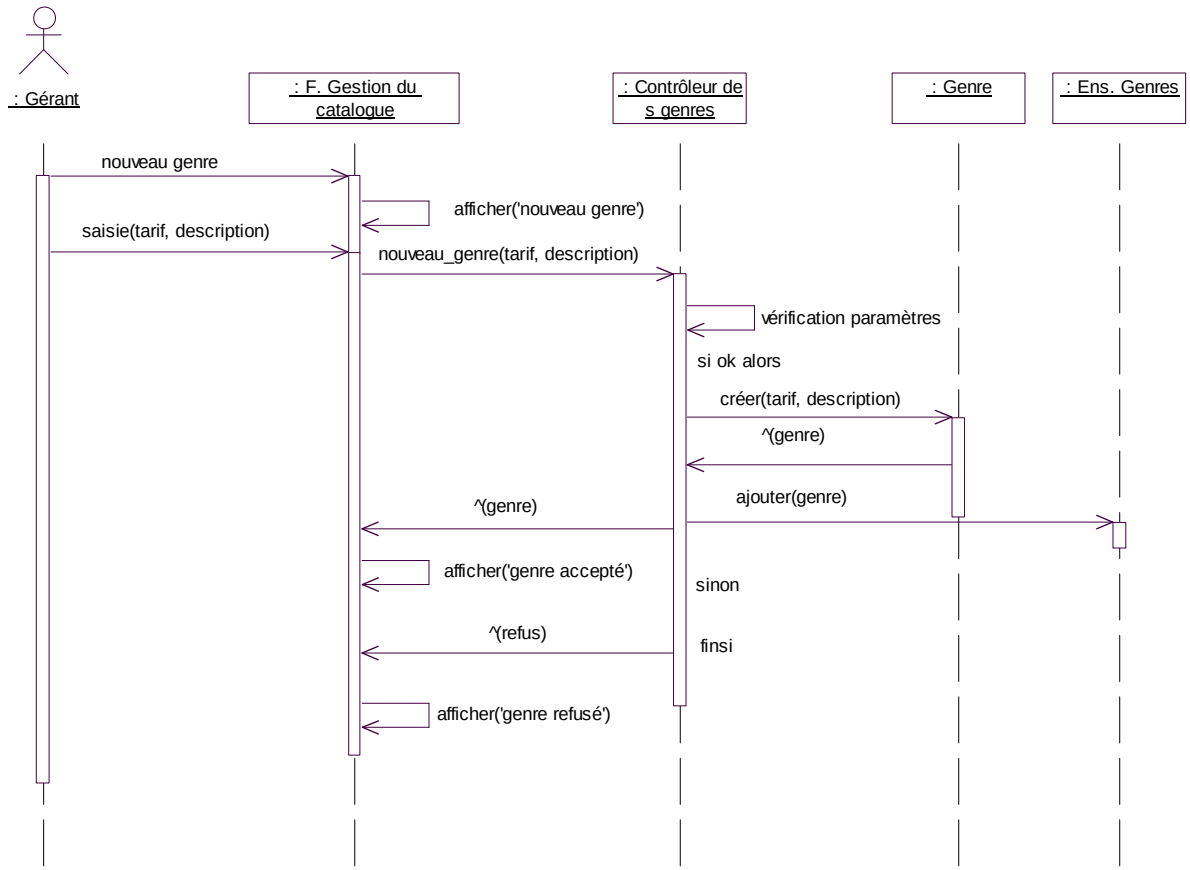


viDCret.eps

Le diagramme de la \lafigure{viDSretr} met en évidence le retrait effectif de la cassette.

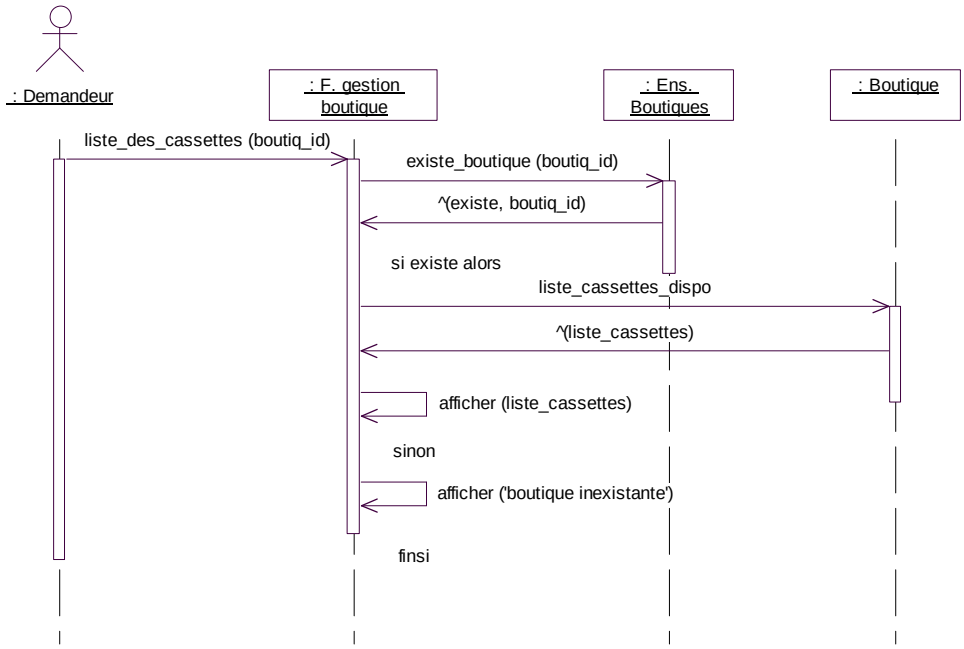
III.1.4 Gestion du catalogue

Nous reprenons ici quelques scénarios de la gestion du catalogue, qui mettent en évidence des objets métiers et interface. La \lafigure{viDStari} décrit l'ajout d'un nouveau tarif (via le genre). Nous avons introduit un niveau intermédiaire, dans la mesure où l'ajout d'un film peut invoquer ce scénario sans ouvrir de fenêtre.



viDStari.eps

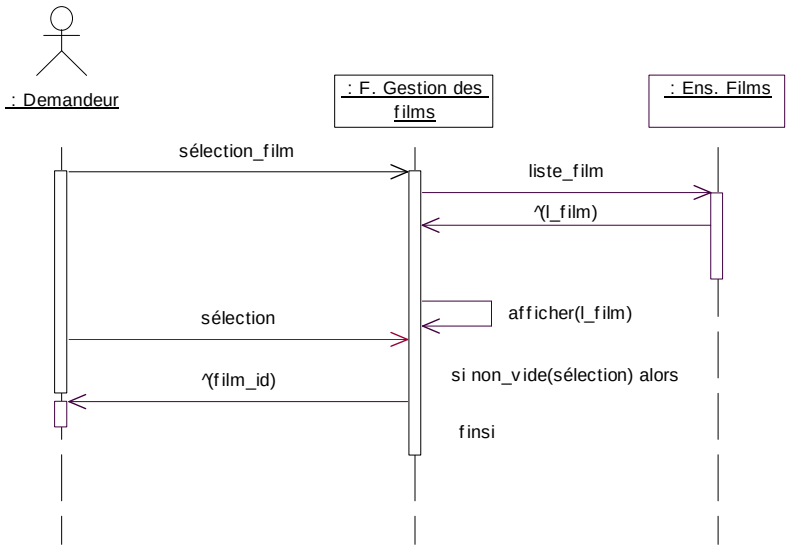
La \lafigure{viDSreq} spécifie la requête *Liste des cassettes d'une boutique*.



viDSreq.eps

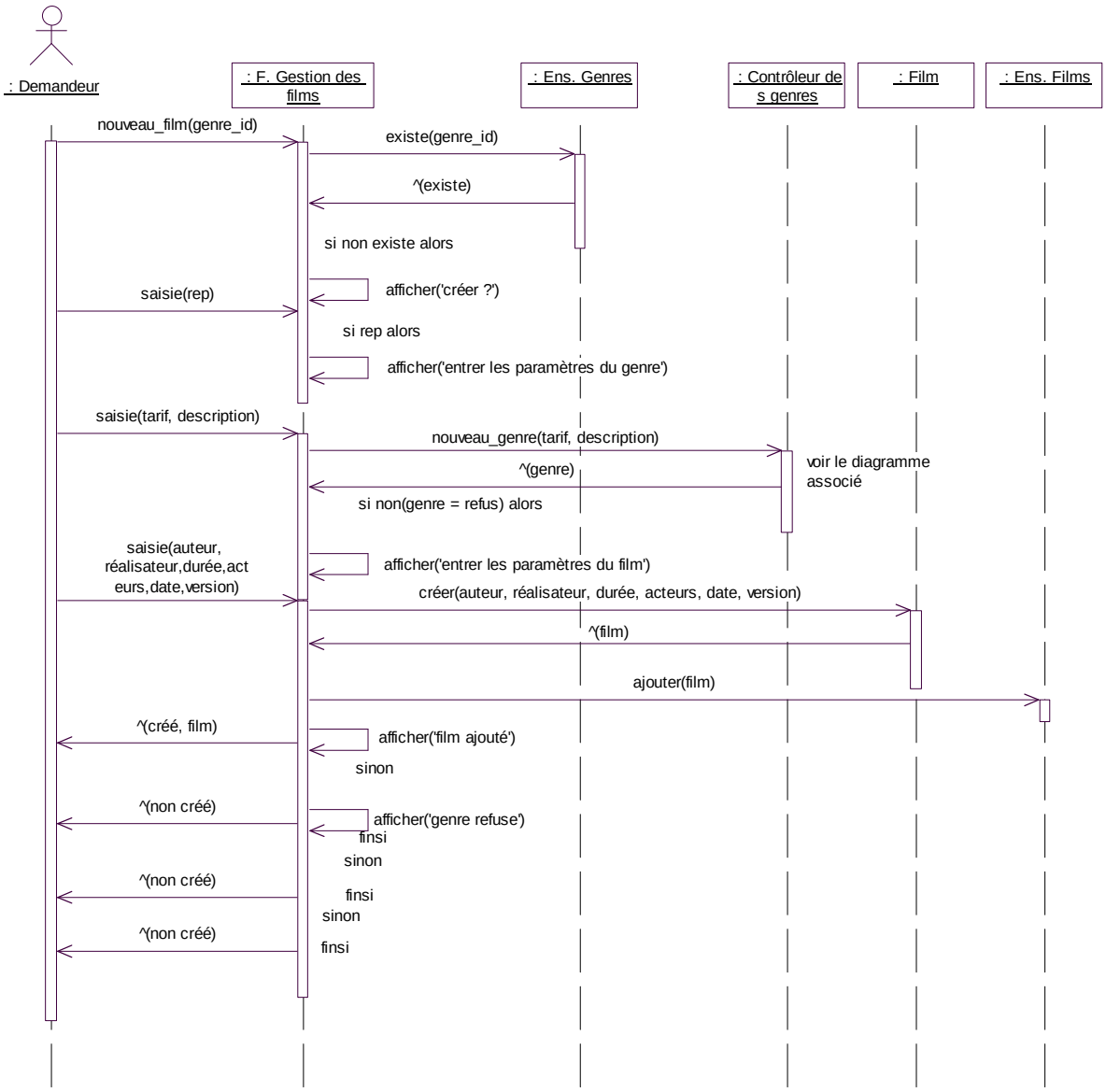
Nous avons fait apparaître les boutiques du club sous forme d'un ensemble, comme pour les genres.

La \lafigure{viDSrefi} précisait la recherche d'un film, la \lafigure{viDSsefi} propose une autre sélection d'un film par une liste.



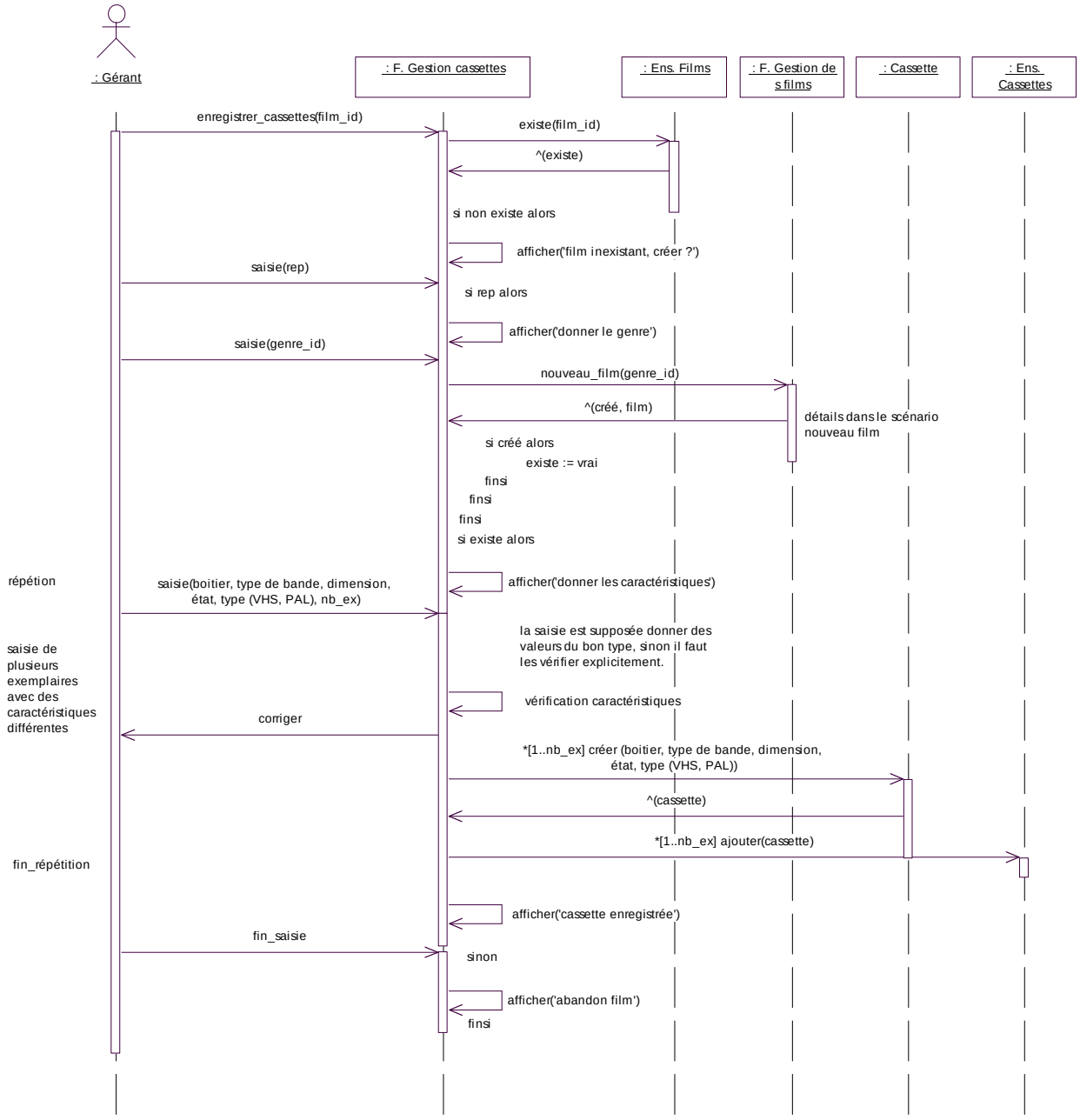
viDSsefi.eps

La \lafigure{viDSnofi} décrit l'ajout d'un nouveau film. Nous faisons appel au scénario de création du genre (nouveau tarif). Nous avons choisi de rendre une réponse en plus de l'affichage car cette séquence est invoquée dans l'ajout de nouvelles cassettes.



viDSnofi.eps

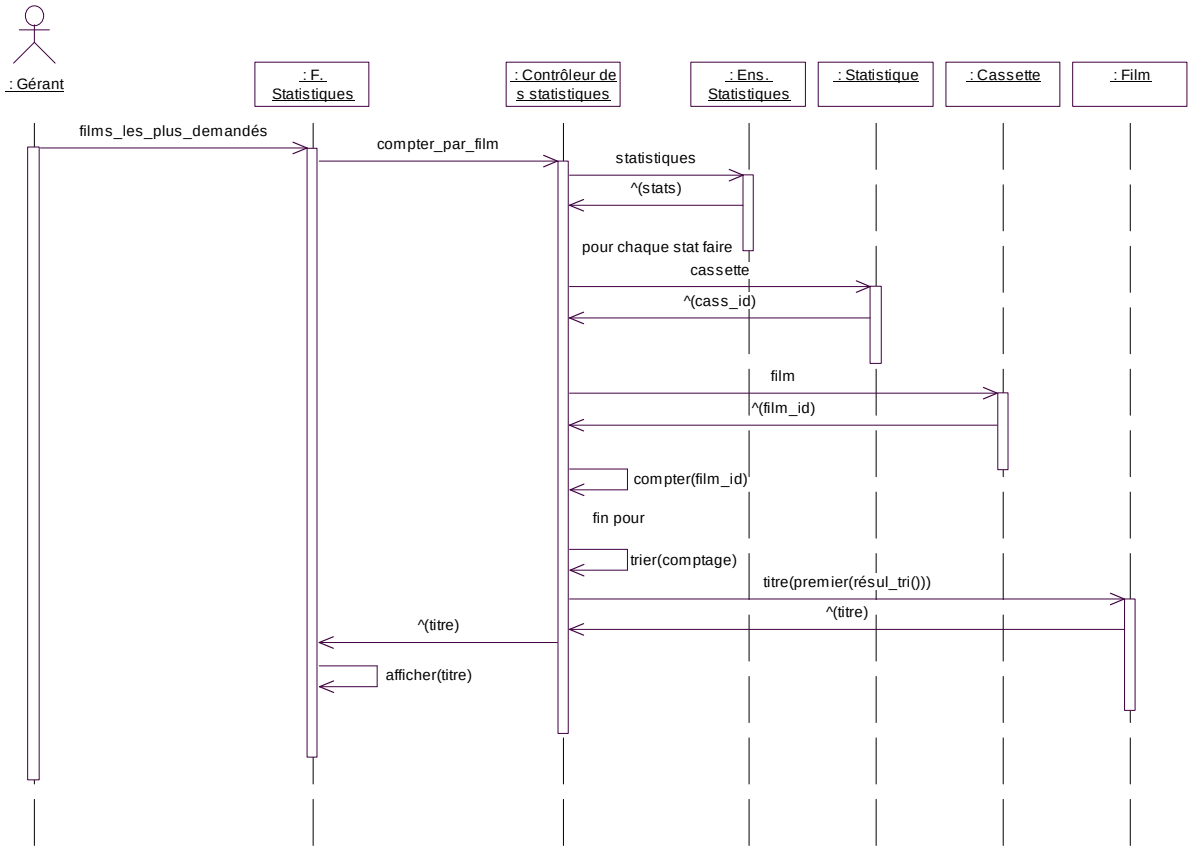
La \lafigure{viDSnoca} décrit l'ajout d'une nouvelle cassette en affinant les scénarios des \lafigure{viSCnoca} et \lafigure{viSCnocr}. Nous avons introduit un niveau intermédiaire, dans la mesure où l'ajout d'un film peut invoquer ce scénario sans ouvrir de fenêtre.



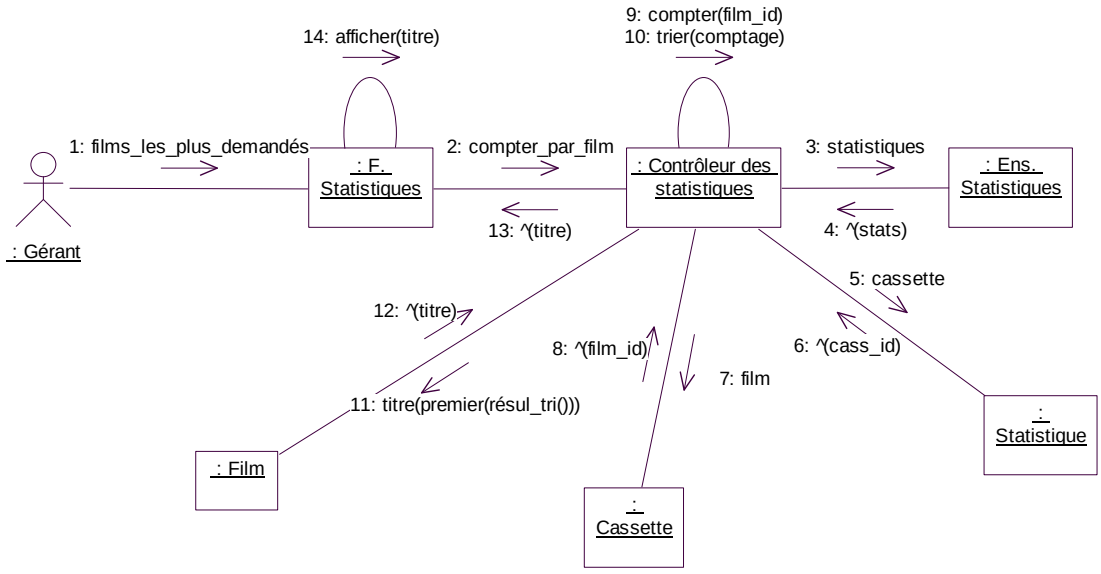
viDSnoca.eps

III.1.5 Statistiques

Les statistiques ne portent pas sur les emprunts en cours. La \lafigure{viDSstat} illustre un exemple de traitement statistique.

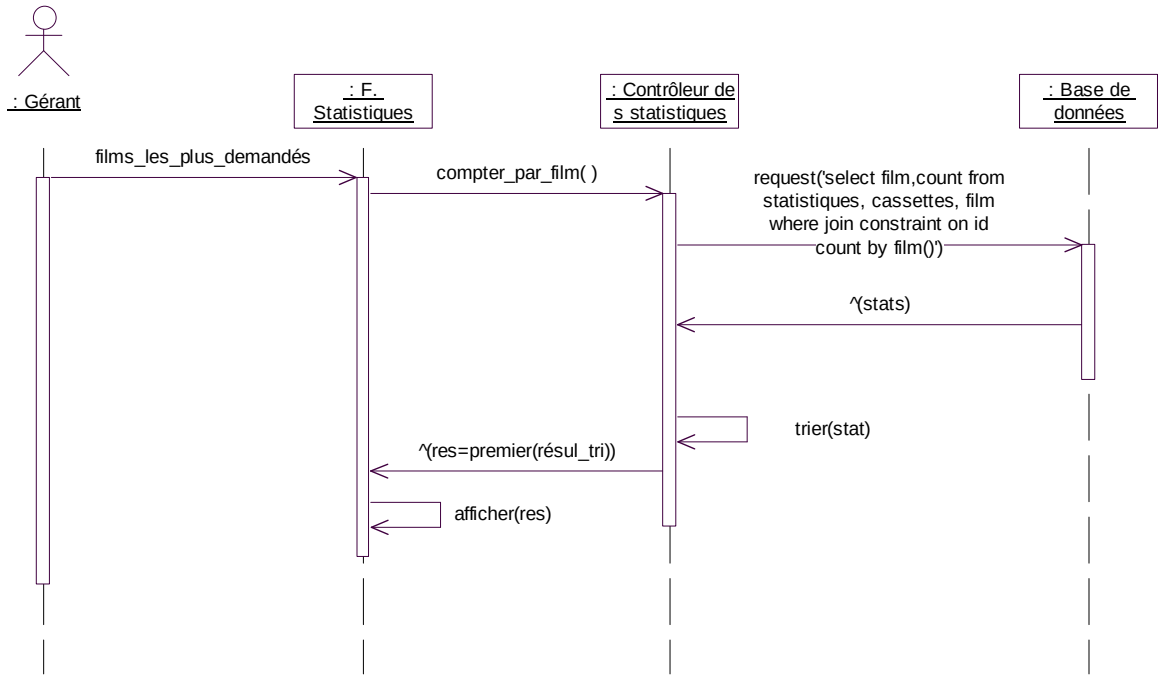


viDSstat.eps



viDCstat.eps

On peut reprocher le style opérationnel. Nous l'avons fait pour mettre en évidence les données utiles au traitement. Mais une fonction unique pouvait servir. Le diagramme de séquences de la \lafigure{viDSstat} fait penser à des requêtes SQL. La \lafigure{viDSstat1} présente une autre version, à la fois « plus abstraite » car on ne précise pas le mode de calcul et plus concrète puisqu'on fait appel à une base de données qu'il faudra concevoir. Bien que plus concise, cette solution n'est pas satisfaisante car elle présume l'existence d'une base de données.



ViDSstat1.eps

III.1.6 Commentaires

Nous avons utilisé des objets anonymes (objets dont on précise la classe mais pas le nom d'objet) car l'énoncé informel ne cite pas nommément d'objets candidats.

> De manière générale, nous préconisons de toujours préciser les classes dans les diagrammes de séquences ou de collaborations au niveau de l'analyse.

Les objets mis en évidence dans les diagrammes de séquence ou de collaboration, sont un support à la découverte des classes du système.

III.2 Diagrammes de classe

L'ensemble des diagrammes de séquence et de collaboration, enrichie de la connaissance du domaine, permet d'établir les classes des objets utilisés dans les interaction.

> D'un point de vue méthodologique, un premier diagramme aurait pu être réalisé à l'issue de l'analyse des besoins, en s'inspirant des méthodes employées pour l'extraction des entités et associations dans Merise. Nous avons établi un tel diagramme, bien que nous ne l'ayons pas présenté ici. Ce brouillon de diagramme a permis de répondre à certaines de nos questions pour la description des séquences.

Nous avons obtenu dans la phase précédente une ébauche des classes du système. Une étude systématique peut s'opérer au niveau des liens et des interactions (événement, envois de messages).

- Les diagrammes de séquence ou de collaboration mettent en évidence des liens entre objets. Une abstraction de ces liens au niveau des classes de leurs objets établit des associations candidates entre classes.
- Les interactions permettent une ébauche du comportement des classes (opération). Dans une version opérationnelle très détaillée, chaque interaction induit une opération de la classe de l'objet receveur. Certains outils comme l'AGL Rose de Rational, vérifient ce type de cohérence. Ainsi, nous avons construit les classes de l'annexe VIII.1 pour vérifier la cohérence dans l'analyse des besoins.

a) Structure

On obtient un diagramme de classe redondant, volumineux et inutilisable en tant que tel. Il faut ensuite raffiner ce diagramme en travaillant sur plusieurs axes :

- Pour maîtrise la complexité, les classes sont regroupées selon des critères choisis. Nous supposons ici qu'une classe peut apparaître dans plusieurs diagrammes. On utilise les paquetages pour regrouper (logiquement) les classes qui obéissent au même critère. Voici quelques exemples de critères :
  - Objets Métiers/Interface/Contrôle/Utilitaires : on groupe les classes par nature.
  - Héritage/Association/Instanciation : on groupe les classes par type de relation.
  - Organisation logique e.g. Prêts/Catalogue/Adhérents/Statistiques
  - Répartition géographique ou d'application (le gérant n'a pas la même application que le boutiquier).

- Séparer les liens dynamiques des liens statiques. Par exemple, la relation entre une fenêtre et un objet métier est dynamique.
- Supprimer les relations transitives i.e. s'obtenant par d'autres relations.
- Construire des hiérarchies d'héritage pour factoriser les comportements communs.
- Affiner les associations en agrégation et composition si nécessaire.
- Valider et vérifier la relation entre deux classes : héritage/association et dérivées/instanciation. Choisir, en le motivant, la relation la plus adaptée. Par exemple, la relation entre *Cassette* et *Film* peut se définir par association (ou agrégation), héritage ou instanciation.
- Ajouter de nouvelles informations issues de l'expertise dans le domaine.

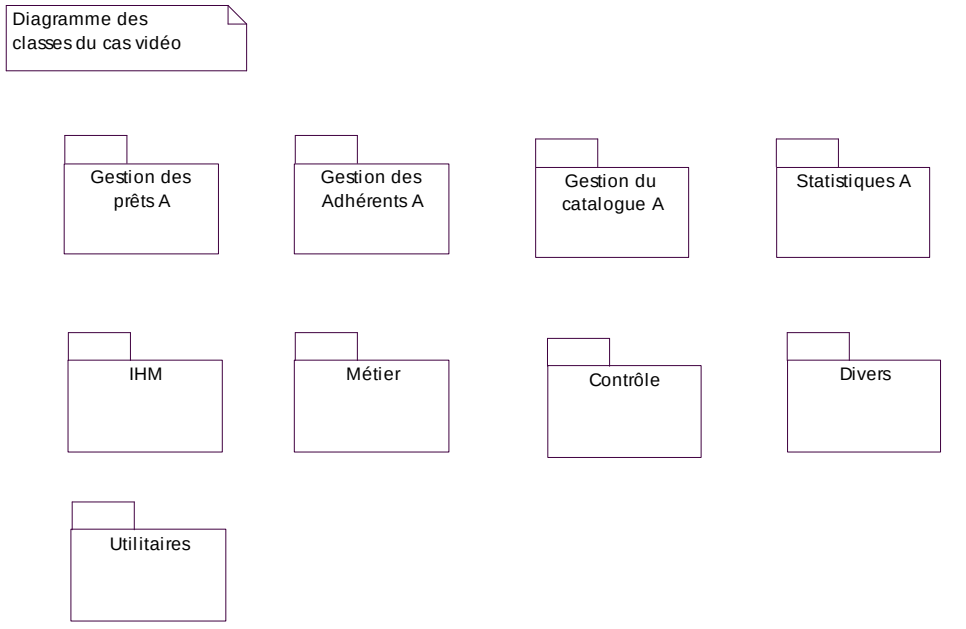
b) Comportement

Lorsqu'on examine le comportement, on a aussi des diagrammes bruts qu'il faut retravailler, dont il faut en particulier éliminer les parasites.

- A priori, aucune structure n'est donnée pour les classes, seul le profil des opérations est fourni (avec une explication). Cette présentation est conforme avec le principe d'encapsulation des objets : seule l'interface des classes est proposée aux clients de la classe. En fait, on peut décrire plus finement les choses et distinguer des opérations atomiques, qu'on rangera, par convenance, dans le compartiment des attributs, et les autres qu'on rangera dans le comportement des opérations.
- L'opération de création (new, create, …) est supposée implicite sauf dans les classes abstraites. En effet, cela relève du comportement de la classe (métaclasse) et non des objets de la classe.
- Le comportement exact des classes interface ou de contrôle ne peut être défini qu'au niveau de la conception détaillée lorsque l'environnement de développement est connu. On pourra ne pas spécifier de comportement pour ces classes, ou lorsqu'on utilise un outils de spécification, y laisser toutes les opérations apparaissant dans les diagrammes de séquence, pour ne pas perturber le contrôle de cohérence entre diagrammes.
- Ajouter de nouvelles informations issues de l'expertise dans le domaine (opérations atomiques sous forme d'attributs, opérations).
- Supprimer les opérations inutiles ou redondantes. Par exemple, dans nos diagrammes de séquences, la symbolisation du retour par `^info` est simplement l'expression du fait que le message déclencheur est synchrone.
- Raffiner les opérations trop abstraites.

> Nous conseillons de ne conserver dans les classes que les opérations pertinentes. Le but n'est pas de définir complètement les classes mais de donner une idée plus précise de la structure et du comportement.

Le diagramme des classes s'exprime maintenant en plusieurs diagrammes. Ces diagrammes peuvent être distincts ou se chevaucher selon les critères de regroupement sélectionnés (héritage, association, composition, géographie, interface, etc.). Voici une organisation possible du diagramme des classes.



viCLgen.eps

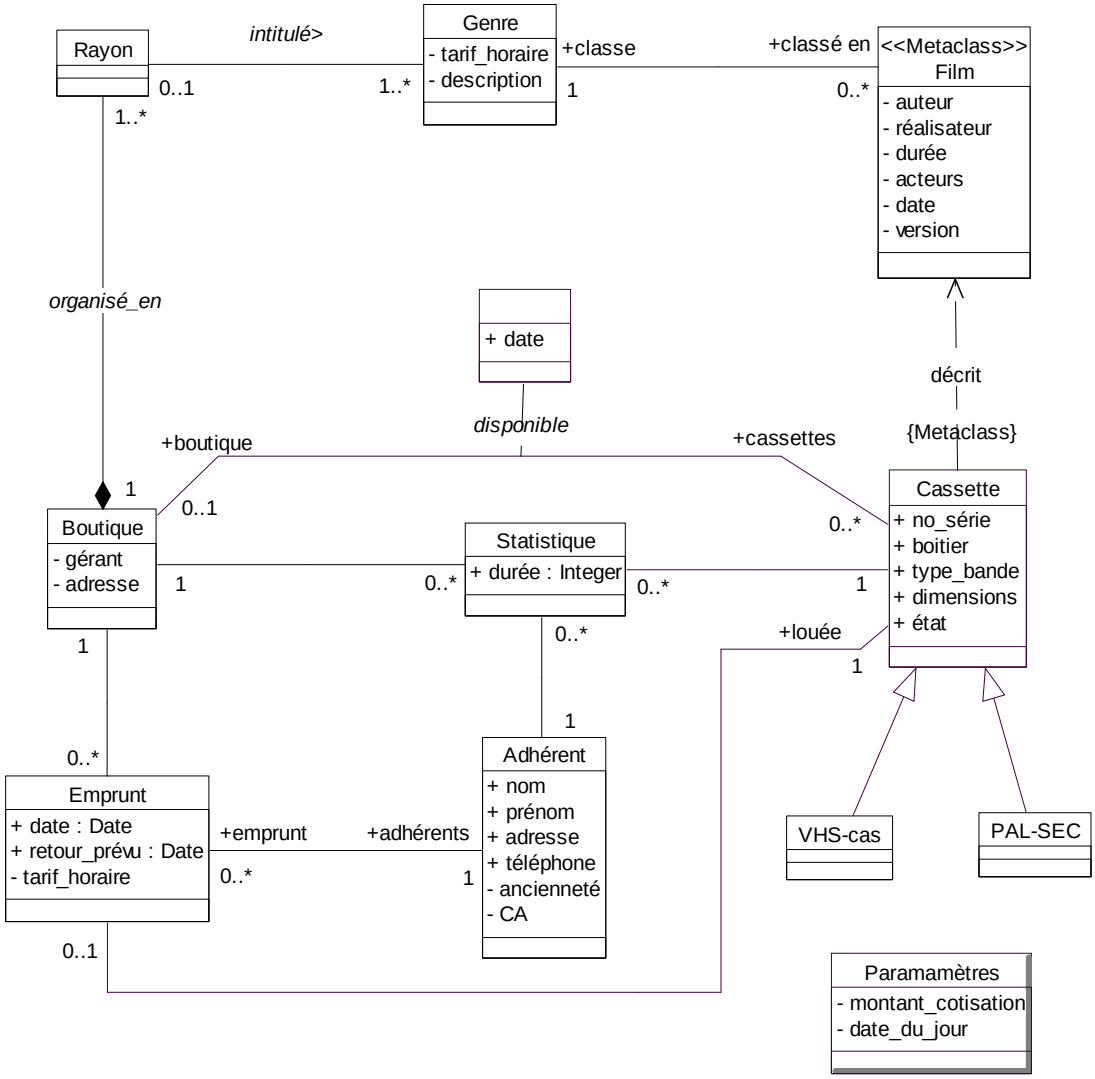
Les sections suivantes décrivent une partie des catégories de la \lafigure{viCLgen.eps}.

III.2.2 Catégories UC

Le suffixe A de certaines catégories de la \lafigure{viCLgen.eps} indique qu'elles correspondent aux paquetages de l'analyse des besoins. Ces catégories contiennent les diagrammes de séquence et de collaboration présentés dans la section précédente.

III.2.3 Objets métiers

Les objets métiers sont relèvent du domaine de l'application et sont indépendants de toute implantation. On parle aussi de vocabulaire du domaine du problème pour désigner les objets métiers. La \lafigure{viCLgen.eps} présente les classes métiers du cas vidéo. Nous en donnons une version avec opérations en annexe.



viCLmet.eps

Le diagramme de la \lafigure{viCLmet.eps} nécessite quelques explications :

- Les annotations d'attributs et opérations définissent la visiblité (public +, protégé #, privé -).
- Chaque association est éventuellement annotée par un nom, des rôles et des cardinalités, des propriétés. Par exemple, la relation *disponible* entre *Boutique* et *Cassette* indique qu'une cassette est disponible dans au plus une boutique (si c'est le cas, alors la boutique est désignée depuis la cassette en utilisant le rôle associé par *cassette.boutique*, c'est la navigation (OCL). Une boutique dispose d'un nombre quelconque de cassettes disponibles. Pour chacune, on a la *date* de disponiblité. La date est rangée dans les propriétés de l'association. Il est possible de nommer la classe contenant la propriété date, dans ce cas, elle est qualifiée de classe-association. Un sens de lecture de l'association peut être donné, par exemple, les rayons sont intitulés par genre.
- La relation entre Boutique et Rayon est de type composition. En effet, un rayon se trouve physiquement dans la boutique.
- La classe *Statistique* correspond typiquement à une association ternaire dans le modèle Entité/Association. Nous l'avons représenté par une classe. Nous n'avons pas précisé les noms et rôles d'associations. Par défaut, le rôle est

celui de la classe atteinte. Par exemple, *cassette.statistiques* est la navigation qui fait passer d'une cassette à ses statistiques.

Les relations n-aires existent dans UML (sous forme de losange). La plupart du temps elles contiennent des propriétés et sont donc représentées par des classes-associations. De plus les cardinalités (lecture à l'anglo-saxonne, inverse de celle de Merise) sont illisibles dans ce cas. En conséquence, nous préconisons, par soucis de simplification, de les représentér par des classes en UML (comme dans OMT, qui n'autorise que les associations binaires).

- Le lien entre Film et Cassette est très intéressant d'un point de vue représentation à objets. Quatre types de relations sont utilisables :
  - Association. C'est le cas classique qui marche à tous les coups mais n'utilise pas forcément la puissance des concepts à objets.
  - Agrégation. L'agrégation peut se lire dans les deux sens :
    - une cassette contient un film : le film est l'agrégé. Cette orientation est même une composition, puisque le film de la cassette est unique.
    - un film est copié en plusieurs cassettes : la cassette est l'agrégée.

    Aucune de ces interprétations n'est intéressante car il existe des traitements spécifiques et indépendants aux films et aux cassettes. Il n'y a donc pas de domination de l'un sur l'autre.
  - Héritage. Les cassettes ont les mêmes propriétés que les films (elles contiennent un titre, des auteurs…). L'héritage semble donc une factorisation intéressante des propriétés. Cette représentation est incorrecte pour deux raisons :
    - Premièrement, il existe des traitements spécifiques et indépendants aux films et aux cassettes. L'héritage implique que toute modification de film est une modification de cassette. Ce qui ne correspond pas à la réalité d'une cassette.
    - Deuxièmement, cette représentation crée des duplications préjudiciables. En effet, pour chaque cassette d'un film, on duplique les informations du film, si on veut modifier le titre du film, il faut alors le modifier dans toutes les cassettes. Tout oubli engendre des incohérences dans le système d'information.

Le critère d'héritage doit inclure le comportement (opérations) en plus de l'intuition et des attributs. C'est ce qui fait la différence avec les modèles Entité/Associations. On factorise le comportement et pas seulement les informations !

  - Protocole de classe. L'objectif est de factoriser des propriétés communes à toutes les instances d'une classe au niveau de la classe. Rappelons que dans la relation d'instanciation, les objets ont une valeur pour chaque variable d'instance (schématiquement et abusivement, pour chaque attribut de la classe) mais que leurs opérations sont « partagées » au niveau de la classe. Le principe est de partager non seulement des opérations mais aussi des attributs. Différentes représentations existent en pratique pour ce concept : variables de classes ou méta protocole.
    - Variables de classe. Les variables de classes contiennent des valeurs partagées par les instances de la classe. Dans UML, les propriétés de classes (attributs et opérations) sont soulignées.
    - Méta-protocole et relation d'instanciation. L'idée est d'utiliser la puissance des métaclasses. Les variables d'instance de la métaclasses (ses attributs) peuvent ainsi mémoriser des attributs communs aux instances. Les métaclasses permettent de supporter « esthétiquement » les variables de classes.

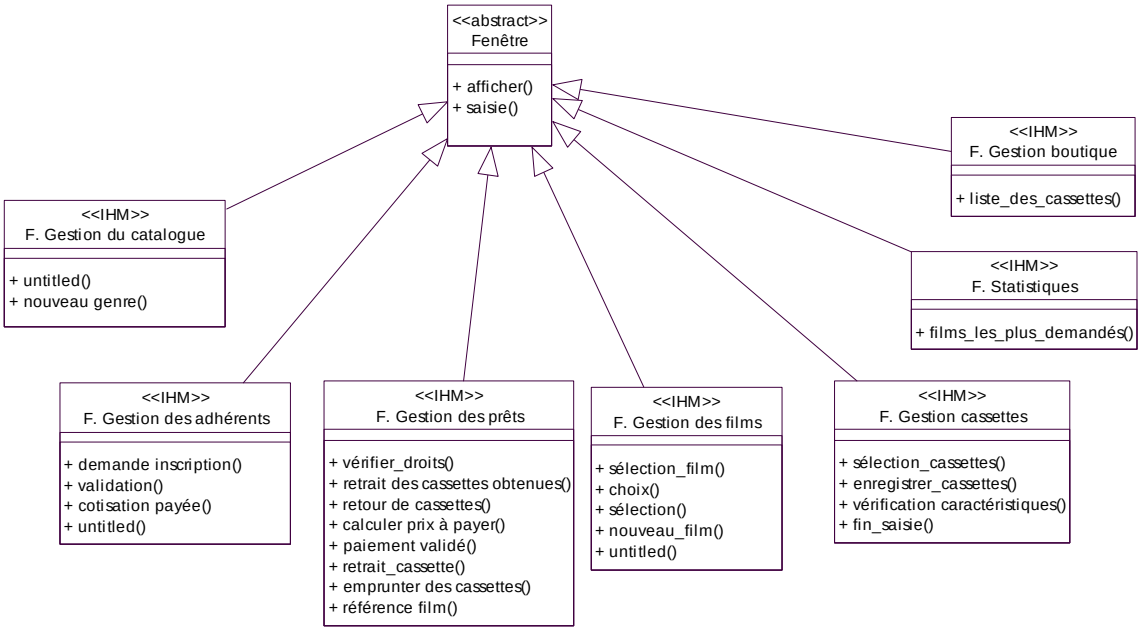
Nous avons utilisé la relation d'instanciation, qui exploite la puissance de l'objet, avec une présentation harmonieuse.

- La classe *Paramètre* contient des informations comme l'identifiant de la boutique ou la date du jour. Elle n'est pas reliée aux autres classes métier.

Nous n'avons pas utilisé ici tous les concepts du diagramme de classe.

III.2.4 Interfaces

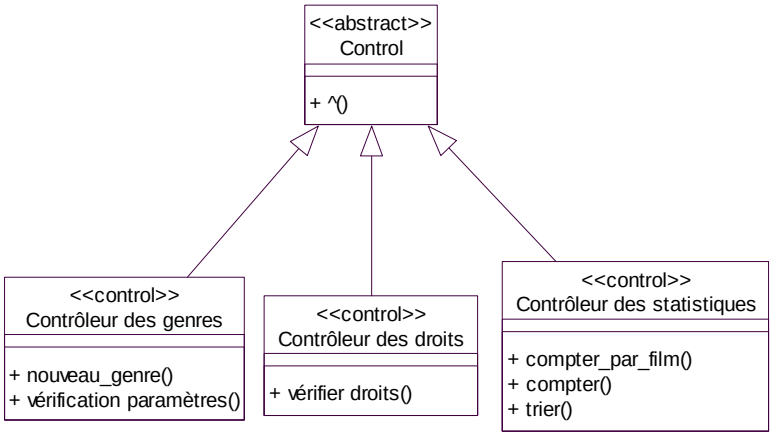
Les classes d'interface sont la représentation des fenêtres utilisées dans les diagrammes de séquence.



viCLint.eps

III.2.5 Contrôle

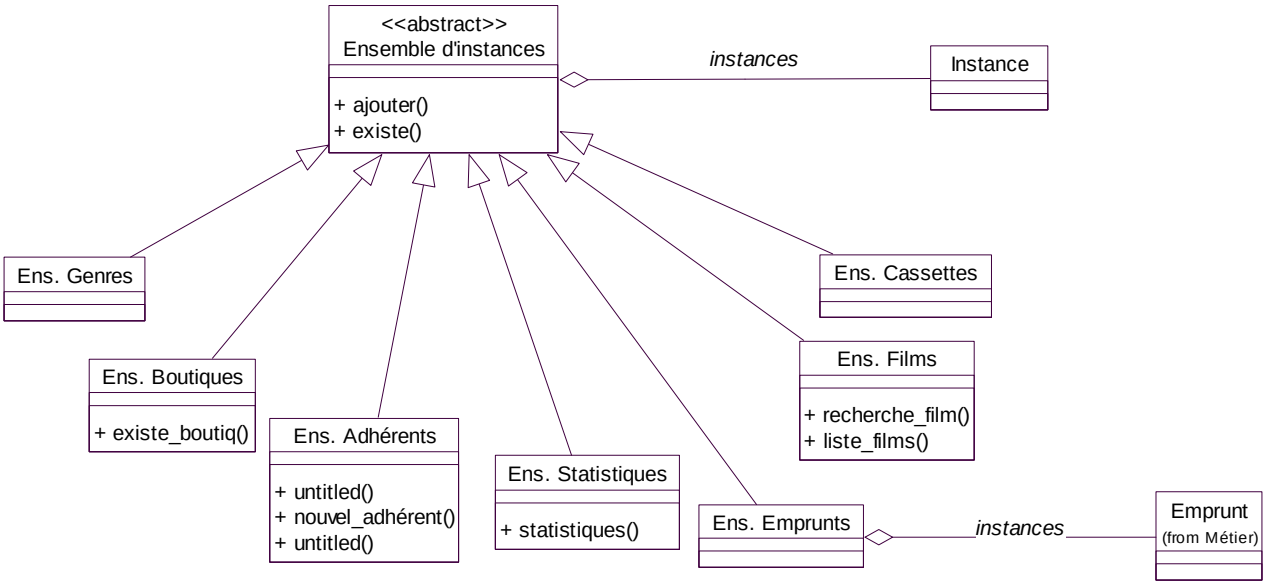
Les classes de la \lafigure{viCLctl} illustrent quelques classes d'objets prenant en charge des collaborations. Seule l'immersion dans l'environnement de développement cible peut affiner ce diagramme.



viCLctl.eps

III.2.6 Divers

Le diagramme de la \lafigure{viCLctl} met en évidence le besoin de générer des instances d'objets. Nous représentons ceci par une relation d'agrégation. Ce n'est pas une composition car les instances sont indépendantes. Un exemple est donné pour les emprunts, les autres sous-classes sont similairement reliées à la classe décrivant leurs instances.

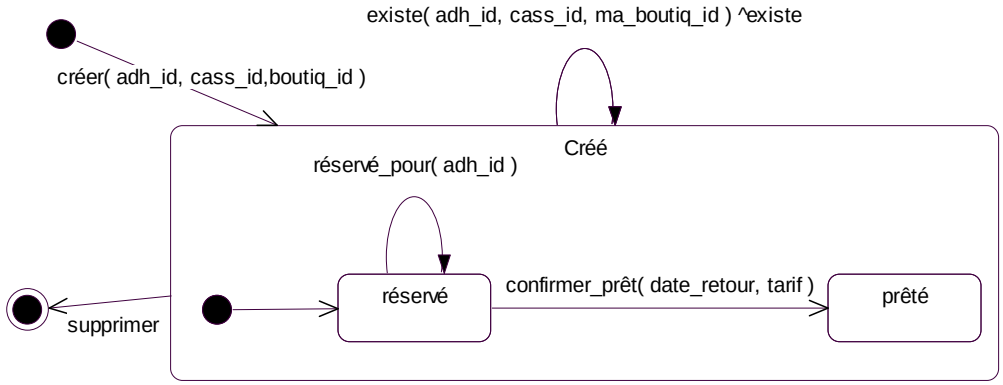


viCLdiv.eps

Si l'environnement cible autorise les métaclasses, alors *Instance* et *Ens. d'instances* sont une même classe.

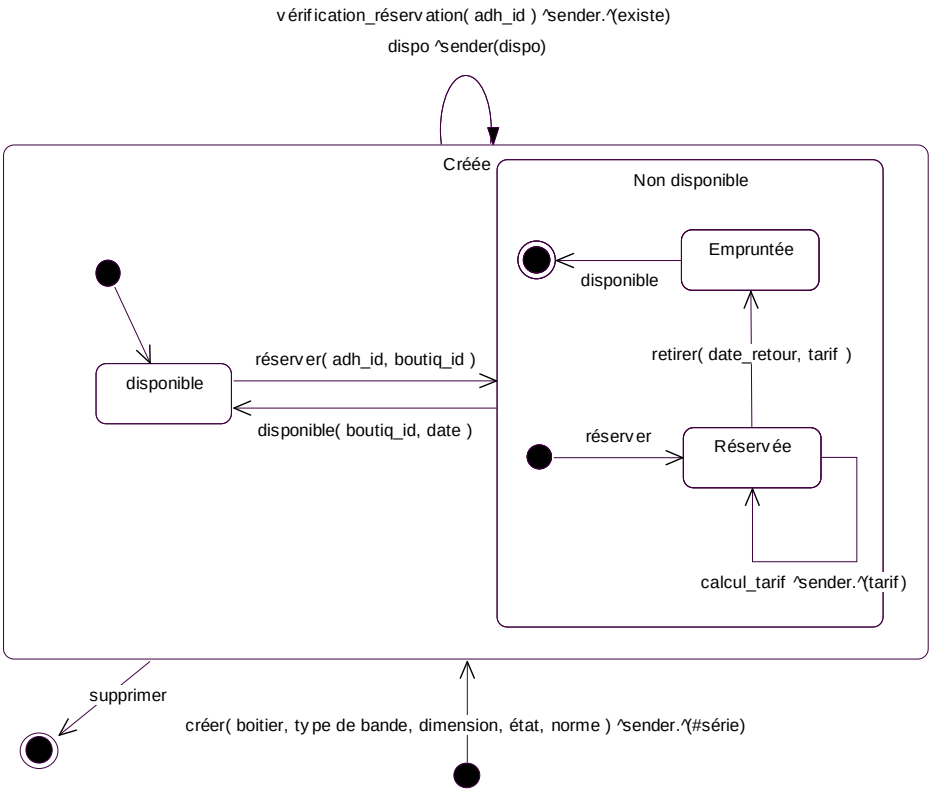
III.2.7 Activités

Dans cette section, nous illustrons le diagramme état-transition. Les objets actifs de ce cas sont les acteurs, dont on ne décrit pas l'activité, l'interface et le contrôle du système. Ces derniers ne sont pas à spécifier ici car leur nature concurrente n'est pas fondamentale dans la compréhension du système. De plus elles dépendent de l'environnement d'exécution cible. La plupart des objets manipulés ici ont un comportement dynamique trivial : stimulus-réponse. Nous nous contentons de tracer le diagramme état transition pour les cassettes et les emprunts.



viETemp.eps

L'emprunt est créé puis automatiquement réservé. Ensuite, on peut interroger et confirmer le prêt. Par exemple, la transition en boucle *'existe(adh_id, cass_id, ma_boutiq_id) ^existe'* signifie que si le message existe est reçu dans l'état *réservé*, la valeur *existe* est rendue. Rappelons que ce test d'existence est invoqué lors du retrait d'une cassette. La variable existe prend la valeur *vrai* si les paramètres *'adh_id, cass_id, ma_boutiq_id'* sont les mêmes que ceux fournis à la création de l'emprunt. A tout moment, l'emprunt peut être supprimé. Bien que l'événement *supprimer* n'existait pas, nous l'ajoutons.



viETcass.eps

L'objet cassette, une fois créé est disponible dans la boutique dans laquelle il a été créé. La cassette peut ensuite être réservée puis empruntée, elle est alors non disponible. Lorsque la cassette existe, on peut à tout moment vérifier quelle a été réservée par un adhérent donné. Dans le diagramme de la \lafigure{viETcass}, nous avons représenté différemment le retour du résultat en ce sens que nous avons donné l'objet cible du retour et le message associé au retour '^()'. On retrouve la notation employée dans les diagrammes de séquences et de collaborations.

Les exemples précédents mettent en évidence la nécessité de vérifier la cohérence entre les diagrammes d'états, les diagrammes de séquence et la spécification de la classe.

III.3 Conclusion

Les diagrammes de séquence décrivent plus précisément l'organisation du système pour la prise en compte des événements extérieurs. Nous avons mis en évidence des objets interface et de contrôle pour indiquer comment se fait cette prise en compte et comment s'organise le contrôle de l'activité.

Il est important de Notez que ceci n'est qu'une représentation abstraite. En effet, l'utilisation d'un système d'interface particulier va changer la réalisation de ces collaborations. Par exemple, sous Windows, la prise en compte des événements est centralisée alors qu'avec Smalltalk elle est distribuée. Une fenêtre interface en analyse peut correspondre à plusieurs fenêtres avec menus et boutons dans la conception, cela reste à préciser.

> Le lecteur aura noté l'usage intensif de structures de données abstraites et d'algorithmes. Ce besoin s'est fait sentir dans UML et les versions récentes ont inclus les propositions d'un langage déclaratif permettant d'exprimer des assertions du type souhaité.

Toutes les modélisation ne sont pas du même type. En effet, lorsque plusieurs alternatives de modélisation se présentent, nous en choisissons une, mais pas forcément la même à chaque fois. Cette pratique présente l'intérêt de varier les exemples de modélisation. Par exemple, une interaction avec une fenêtre de l'interface résulte soit en un affichage, soit en un envoi de message.

IV. CONCEPTION

Dans l'analyse, nous avons mis en évidence une première structure logique du système. L'objectif de la conception est de définir l'architecture du système informatique et l'organisation des composants logiciels. Nous nous sommes inspirés pour cette partie de l'ouvrage de KETTANI cite{uml_kettani}. Dans cette partie nous serons assez succincts.

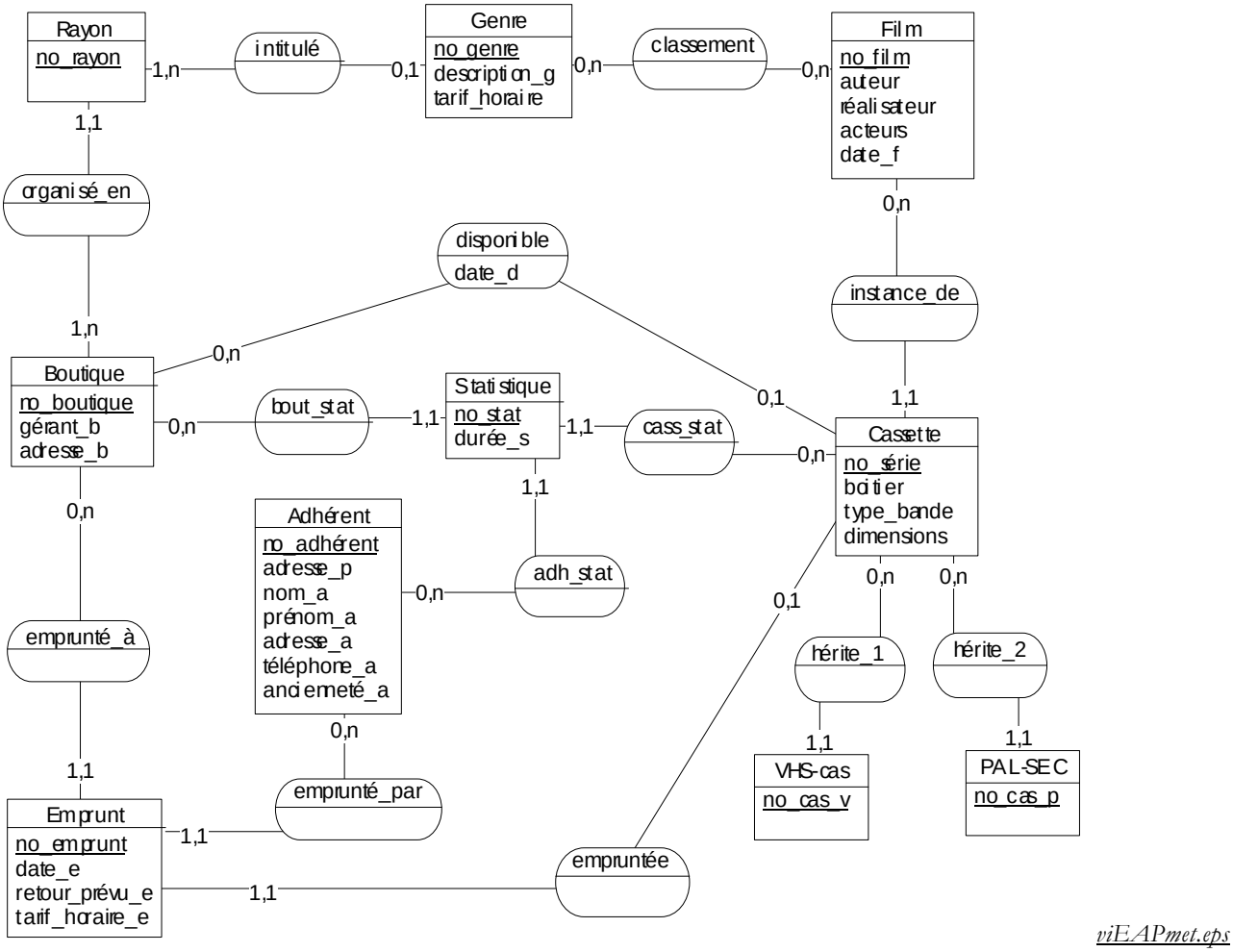
IV.1 Architecture

Nous nous basons sur un modèle d'architecture en couches (Applications/Objets métiers/Objets techniques/Objets d'infrastructure). L'architecture est dictée par un certain nombre de choix techniques (architecture centralisée ou distribuée, environnement de programmation, systèmes existant, etc.) et d'organisation. On distingue deux niveaux organisationnels : le club et la boutique. Un certain nombre de traitements faits dans une boutique impliquent des consultations et mises-à-jour dans d'autres boutiques. Nous préconisons donc une architecture client-serveur avec une base de donnée générale au niveau du club. Pour des réduire les coûts de communication, on envisage une répartition de la base ou des réplications locales.

Les orientations suivantes sont prises :

- Le système emploie la technologie client/serveur. Le serveur gère la base de données, dont une représentation conceptuelle est obtenue à partir du diagramme des classes métier. La base est gérée par un SGBD relationnel et la programmation se fait dans un langage à objet (Smalltalk-80, C++ ou Java). L'interface est réalisée par un *middleware* du marché.
- Le gérant et le boutiquier utilisent différemment le système. Deux options sont possibles pour mettre en évidence les différences :
  - Définir deux applications différentes. Pour chaque application on indiquera les objets métier, les objets interface et de contrôle utilisés.
  - Définir une seule application avec des niveaux d'utilisation. Ainsi, certaines fonctions ne seront accessibles que sous certaines conditions (menus "grisés").
- Nous choisissons la seconde option. Il faut prévoir une gestion des comptes par type d'utilisateur (identification, mots de passe).

Lorsqu'on a affaire à une application distribuée, on met en évidence les processus concurrents et le moniteur temps-réel utilisé. L'énoncé ne met pas en évidence de contraintes de performance, de qualité ou de coûts.



viEAPmet.eps

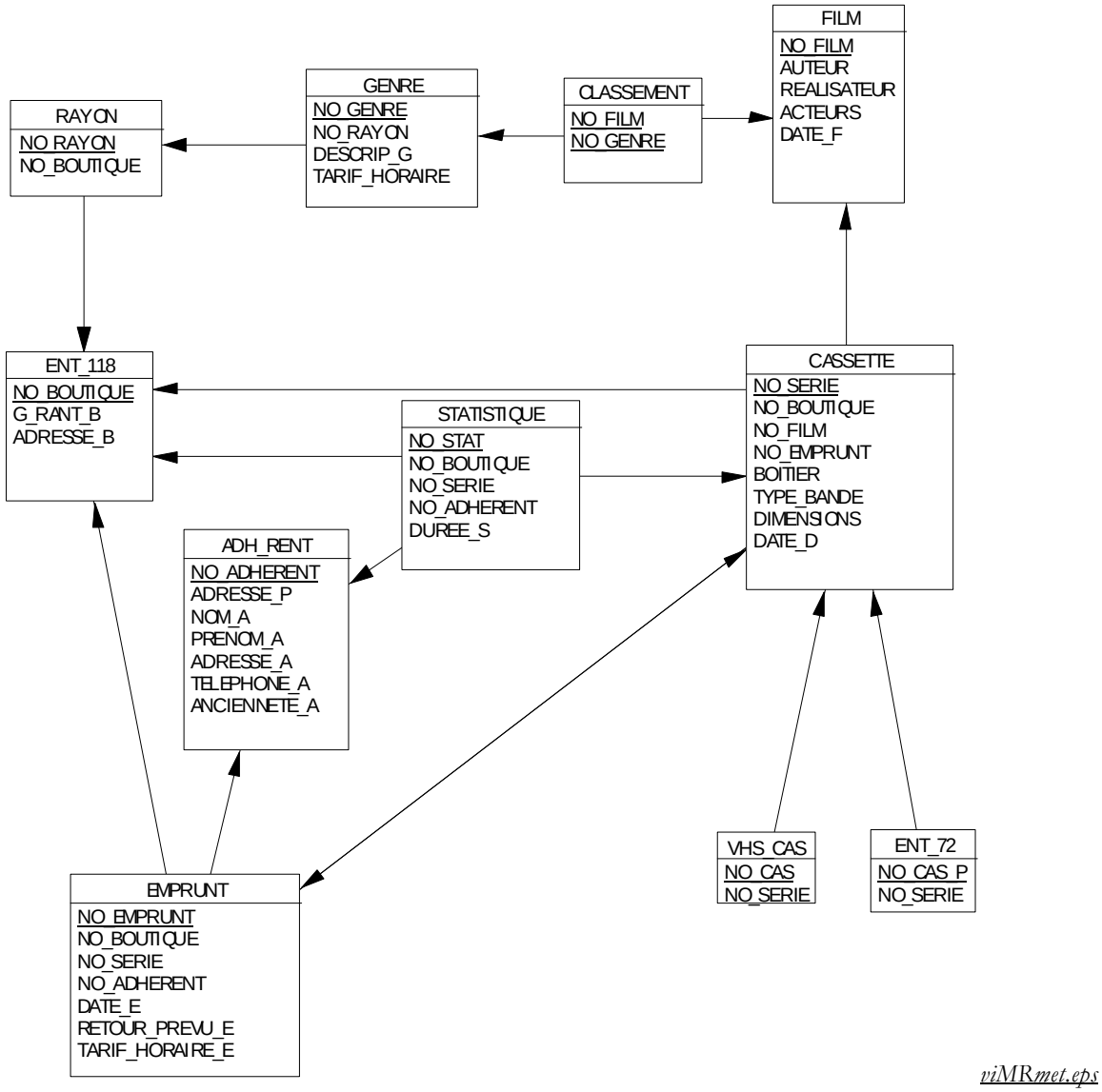
IV.2 Interfaces

Les dessins des interfaces des feêtres peuvent être fournis à ce niveau. L'interface suit les standards du marché et est fonction de l'environnement de développement choisi.

IV.3 Persistance

Les objets métiers constituent le coeur de la base de données. Après le choix d'une base de données, il faut coupler les classes du programme avec celles de la base et implanter les requêtes. Lorsqu'on a plusieurs paquetages d'objets métiers, on peut associer à chacun un paquetage de persistance. Le transformation en table relationnelles est relative à la conception des associations (section V.2) et à transformation d'un modèle E-A-P en modèle logique (Merise). Si le modèle n'autorise pas les relations relations d'héritage ou d'instanciation, on peut les décrire en utilisant uniquement des associations.

Le modèle relationnel de la \lafigure{viMRmet} est une implantation possible des objets métiers de \lafigure{viCLmet}.



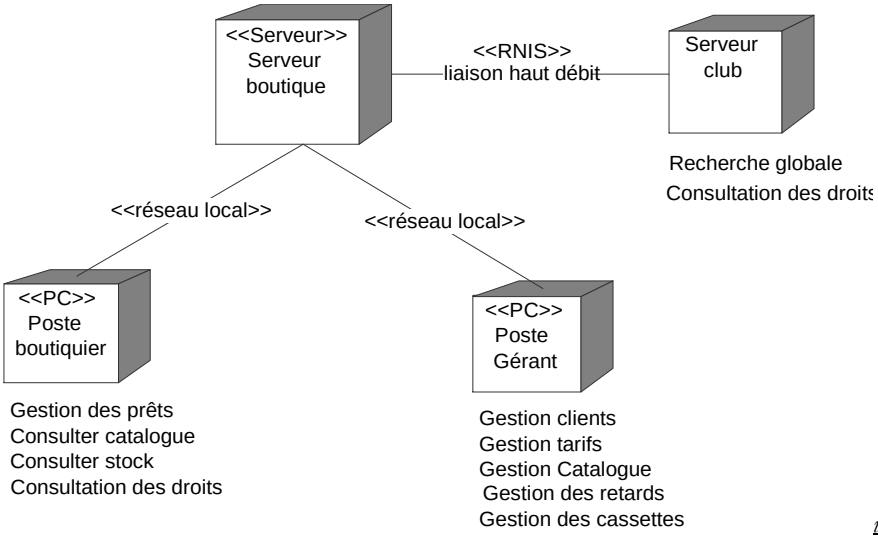
viMRmet.eps

IV.4 Répartition

La répartition consiste à mettre en évidence les composants et leur déploiement. KETTANI présente ainsi la transition des paquetages logiques aux composants : "*Chaque catégorie de l'architecture logique se transforme en composant ; ces derniers sont regroupés en sous-systèmes. Chaque sous-système doit correspondre à un environnement de développement [...] Chaque composant, qui peut contenir une ou plusieurs classes, doit être une unité de réalisation et de maintenance.*" \cite{uml_kettani}.

A ce niveau de conception, la granularité de définition est forte. La conception détaillée affinera les composants. Des choix de réalisation conditionnent cette répartition. Ils sont fonction de la technologie utilisée et des coûts liés au déploiement (cohérence des information, limitation des communications, informations à jour, sécurité des transactions, etc.).

Dans l'exemple du club vidéo, les boutiques travaillent en autonome, hormis pour l'emprunt à distance et les vérifications de droits des adhérents. Il y a deux types de sites : système boutique et système club. Chacun fournissant un nœ ud du graphe de déploiement. Une partie de l'application se trouve sur chaque nœud. Nous avons séparé le poste du gérant de celui du boutiquier pour indiquer que certaines fonctions diffèrent.



viDDgen.eps

En fait, l'étude de la répartition, peut nous faire revenir sur l'étude des besoins. En effet, jusqu'ici nous avons traité des requêtes ou des statistiques sans nous préoccuper du fait qu'elles concernaient une boutique ou le club. La répartition des composants met en évidence le fait que chaque boutique dispose d'une partie seulement des informations. Pour simplifier, on considère que la base de donnée de référence se trouve sur le serveur du club. Chaque boutique dispose d'une mise à jour quotidienne de l'ensemble de la base (adhérents, catalogue, statistiques, etc.). De même le club met à jour sa base chaque jour en interrogeant les bases de données locales. Les modifications suivantes sont apportées, pour des raisons techniques, aux traitements de l'emprunt :

- La vérification des droits à l'emprunt qui concernent la contrainte du nombre d'emprunts journalier par boutique ne pose pas de problèmes de cohérence car elle local à la boutique.
- La vérification des droits à l'emprunt qui concernent la situation de l'adhérent varie au jour près, la situation de la veille est donc suffisante.
- La réservation dans une autre boutique est conditionnée par un nouveau traitement, en temps réel, de vérification et mise-à-jour de la base de la boutique en question.
- Les autres traitements se font à la situation de la veille.

V. CONCEPTION DETAILLEE

V.1 Eléments généraux

La conception détaillée précise le contenu des composants (classes, autres composants, bibliothèques, programmes). La description est enrichie à partir des composants techniques issus de l'environnement de développement cible pour réutiliser ses classes. Les patterns peuvent être utilisés en conception et en conception détaillée pour mettre en évidence les structures réutilisables.

V.2 Conception des associations

En programmation à objets, une association s'implante habituellement avec des pointeurs. On peut s'inspirer des règles de transformation du schéma E-A-P en Z (section 2 du chapitre 5) ou en relations du modèle relationnel (chapitre 3 du tome 1). Examinons les alternatives de modélisation.

1. L'association ne possède pas de propriétés. Les liens sont représentés par des attributs de navigation, un par classes de la relations. Par exemple, *film.classe* donne le genre et *genre.classé_en* donne les films. Ces attributs prennent en général les noms des rôles. Il y a quelques cas particuliers :
   a. Navigation unidirectionnelle : un seul attribut de lien, dans la classe origine de la navigation.
   b. Cardinalité maximale dans un sens : on peut choisir une navigation unidirectionnelle.
2. L'association possède des propriétés et
   c. une cardinalité maximale de 1 (ex : relation *disponible*: la propriété migre dans la classe *Cassette*, on se retrouve dans le cas 1.

d. aucune cardinalité maximale de 1, l'association donne lieu à une nouvelle classe. Elle est reliée aux classes sous-jacentes par des associations binaires sans propriétés (cas 1). Implicitement, c'était le cas de la classe *Statistique*.

Notons pour terminer que l'outils AGL utilisé influence fortement la conception de l'application. Un AGL complet inclue en effet des éditerus de documents, des générateurs d'interface, des éditeurs et générateurs de code, des outils de vérification, de test et de prototypage.

VI. CONCLUSION

Ce cas d'étude met en œuvre la notation UML dans la modélisation d'un système d'information. Nous avons essayé de faire un tour d'horizon des points abordés dans une analyse conception avec UML. Nous sommes conscients des erreurs qui subsistent dans la spécification résultante. Les objectifs qui nous ont guidé sont les suivants :

- Définir un cas d'étude relativement simple pour illustrer une démarche avec UML.
- Fournir un nombre suffisant de diagrammes pour appréhender la complexité de la modélisation, le besoin de cohérence entre diagrammes d'un même type et entre diagrammes de types différents.
- Mettre en évidence la nature d'un processus sans couture (*seamless*) : utiliser les mêmes notations au fil du processus, affiner progressivement les modèles en conservant à chaque niveau un degré d'abstraction suffisant par rapport au niveau suivant.
- La représentation idéale est l'image même du ou des programmes. UML peut servir à décrire de tels programmes mais les schémas risquent d'être volumineux. Il s'agit là d'un travail de rétro-ingénierie fortement utile dans la documentation des projets. Automatiser la production de schémas UML à partir de code dans un langage de programmation serait un progrès énorme dans la compréhension et l'application d'UML. En effet, cela impliquerait une sémantique formelle des concepts.

Nous avons mis en évidence un certain nombre d'enseignements. En particulier, nous notons les points suivants :

- Il n'y a pas de méthode miracle pour concevoir un système d'information. L'approche présentée ici s'applique aux systèmes d'informations classiques, de type gestion.
- Bon nombre de commentaires sont à formaliser dans la notation pour permettre un contrôle plus fin de la spécification. Rappelons qu'il s'agit là d'un des objectifs du langage OCL.

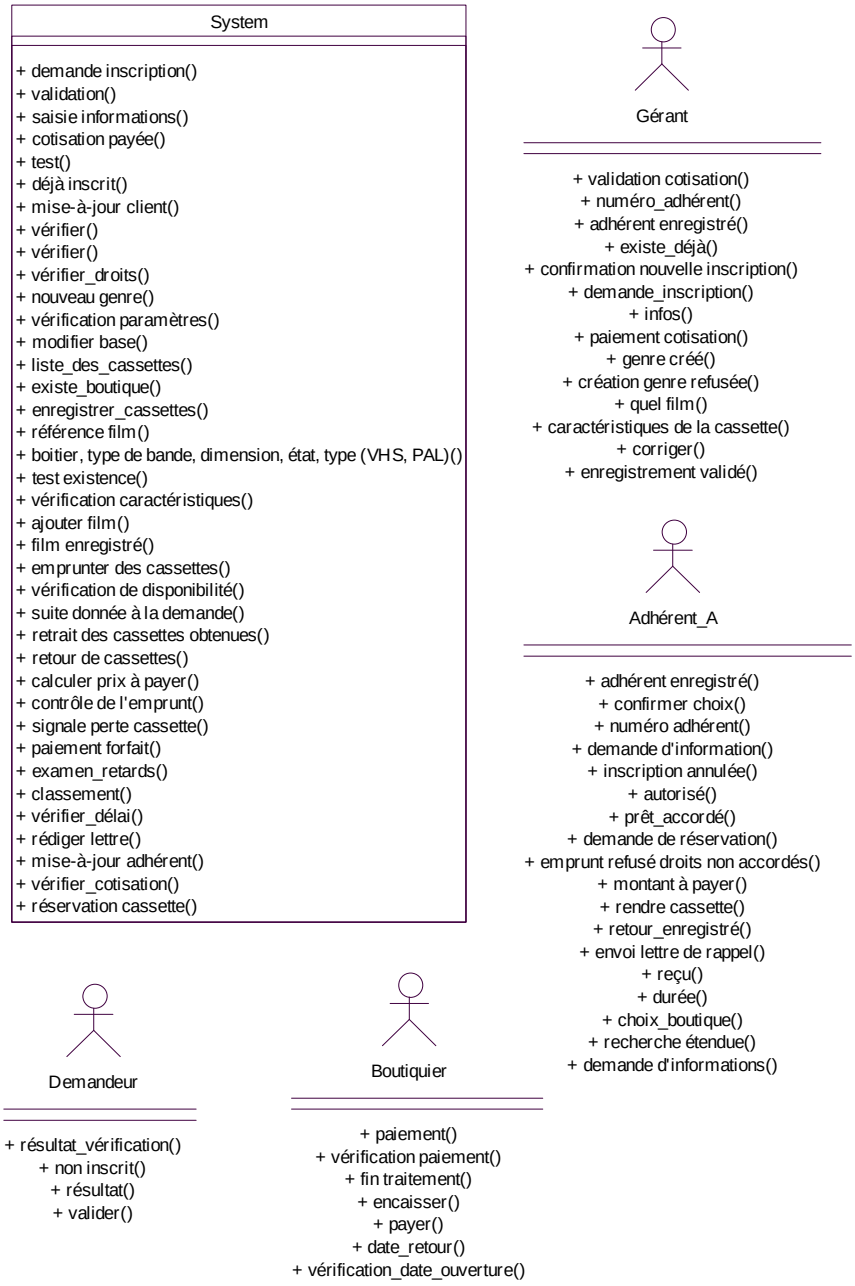
Nous espérons que le lecteur aura lu avec profit ce travail.

VII. BIBLIOGRAPHIE


```
@book{uml_kettani,
 author = {Kettani, Nasser and Mignet, Dominique and Paré, Pascal
  and Rosenthal-Sabroux, Camille},
 isbn = {2-212-08997-X},
 note = {ISBN 2-212-08997-X},
 publisher = {Eyrolles},
 title = {{Modélisation objet avec UML}},
 year = {1998}
}

@techreport{uml,
 author = {UML Consortium},
 address = {available at ftp://ftp.omg.org/pub/docs/ad/99-06-08.pdf},
 institution = {Object Management Group},
 month = jun,
 title = {{The omg unified modeling language specification, version 1.3}},
 year = {1999}
}
```

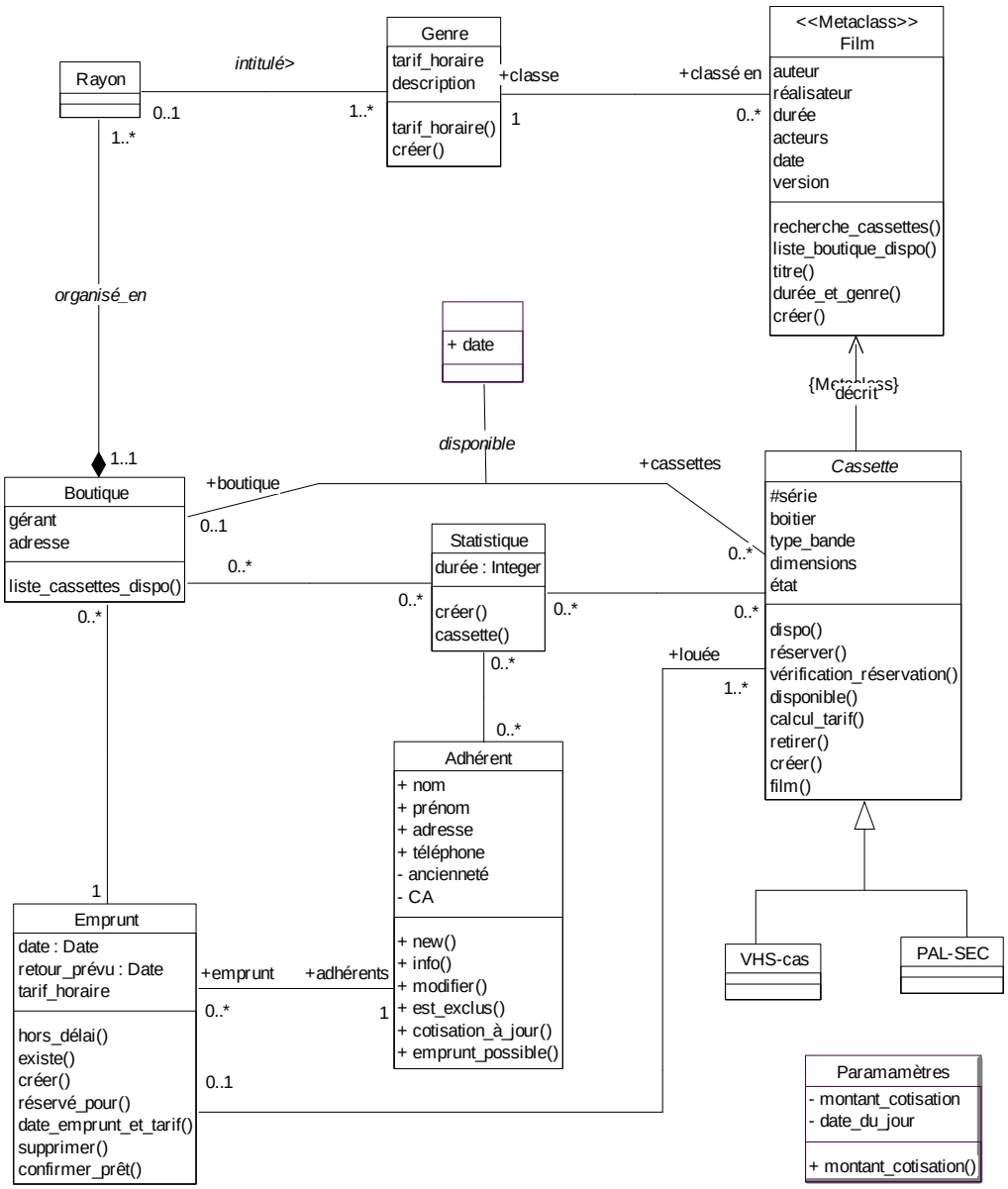
VIII.1Classes de l'analyse des besoins



viCLanbe.eps

VIII.2Classes de l'analyse

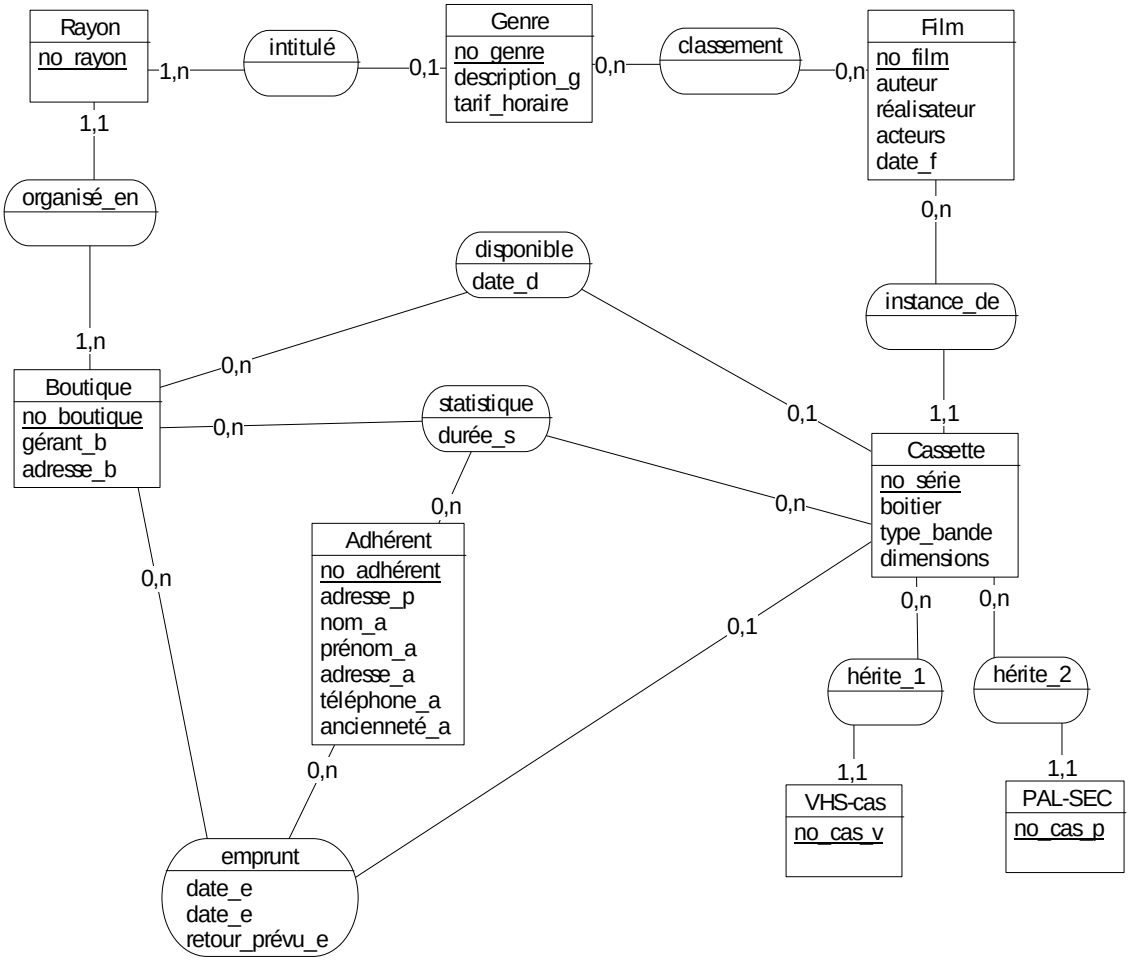
VIII.2.1 Classes métier avec opérations



viCLmeto.eps

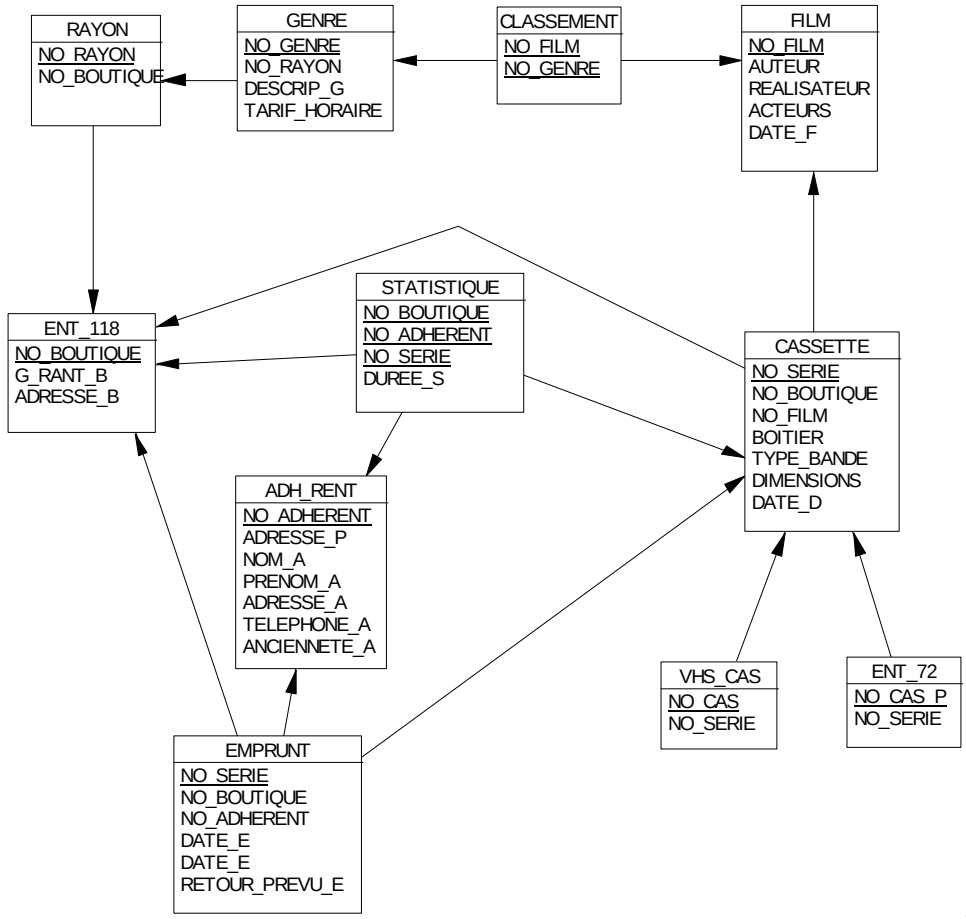
VIII.3Modèles de persistance

VIII.3.1 Modèle E-A-P



viEA1met.eps

VIII.3.2 Modèles relationnel



viMR1met.eps

TABLE DES MATIERES