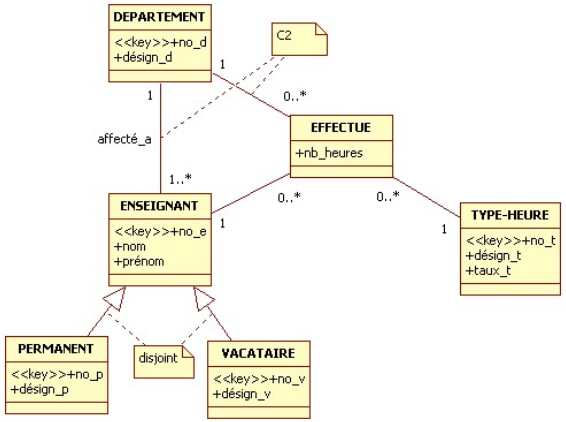
Chapitre 3 : Diagrammes de classes
Éléments de correction

Il s'agit d'UN corrigé. D'autres réponses sont possibles, notamment là où une modélisation est demandée.

Partie I : Liaisons schémas E-A-P ↔ UML

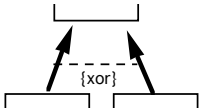
Exercice 3.1

Le diagramme de classes correspondant à ce schéma E-A-P peut être le suivant :



Remarque StarUML Nous n'avons pas trouvé comment visualiser les contraintes avec StarUML. Nous avons donc, dans les deux cas, « usé de la note ».

L'association ternaire a été exprimée sous forme d'une classe. Nous avons conservé les attributs clés du schéma E-A-P et les avons mis en évidence grâce à des stéréotypes. La traduction des contraintes entraîne une différenciation de notre réponse. La contrainte C2 ne peut pas être traduite autrement que par une remarque. C1, par contre, peut l'être, UML offrant cette possibilité :



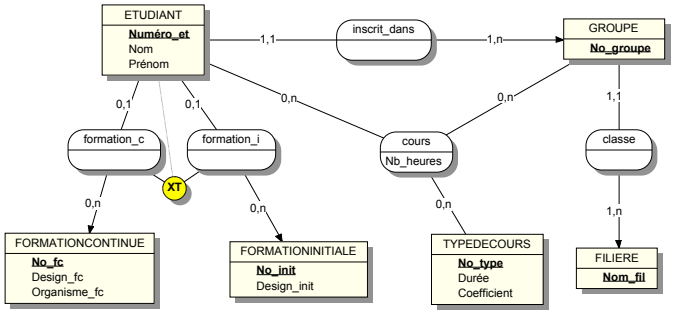
Le logiciel de dessin que nous utilisons, StarUML, ne permet pas cette expression.

Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)

Exercice 3.2

Le schéma E-A-P équivalent au diagramme de classes est le suivant :



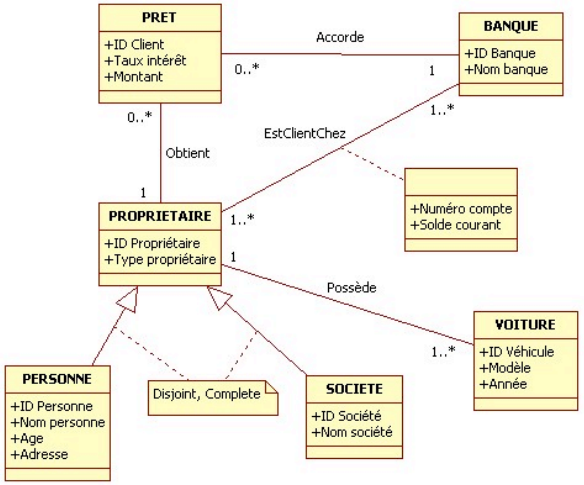
Remarque WinDesign On aura noté trois particularités de ce logiciel. La première consiste à sur-matérialiser les associations fonctionnelles, celles qui ont une cardinalité maximale à 1 (c'est le cas de *inscrit_dans* ou de *formation_c*, par exemple), en orientant l'arc en direction de la cible de la fonction. La deuxième concerne la contrainte de partition, que WinDesign note X (exclusion) + T (totalité) et qui associe ETUDIANT. La troisième et dernière porte sur les attributs clés, qui ne sont pas soulignés mais mis en gras.

L'association ternaire est la seule traduction possible de la notion de cours, du fait des clés. Nous avons volontairement transformé le nom d'une propriété de l'entité *FormationInitiale*, *Design_fc*, qui faisait doublon avec celle de *FormationContinue* et que nous avons appelé *Design_init*.

Partie II : « petits » exercices, de « mise en bouche »

Exercice 3.3

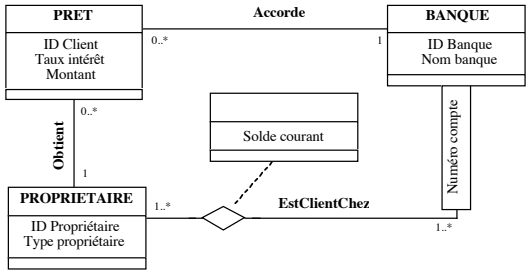
Il y a plusieurs réponses possibles à ce « problème », surtout du fait que l'interprétation de l'énoncé et du cas modélisé est autorisée. En voici une :



Remarque StarUML Là encore, nous avons dû utiliser une note pour représenter la contrainte de partition entre *PERSONNE* et *SOCIETE*.

Nous avons supposé, pour atteindre cette solution, que les propriétaires de voitures pouvaient avoir plusieurs comptes bancaires, mais que ceux-ci étaient ouverts dans des banques différentes. Nous avons également pris l'énumération des numéros d'employé (*ID Employé 1*, *ID Employé 2*…) pour des erreurs. Nous avons enfin ajouté la propriété *Montant*, correspondant au montant du prêt et qui nous paraît nécessaire.

Il est possible d'utiliser la notion de qualification pour représenter les numéros de compte, chaque client étant identifié, dans chaque banque, par son numéro de compte :



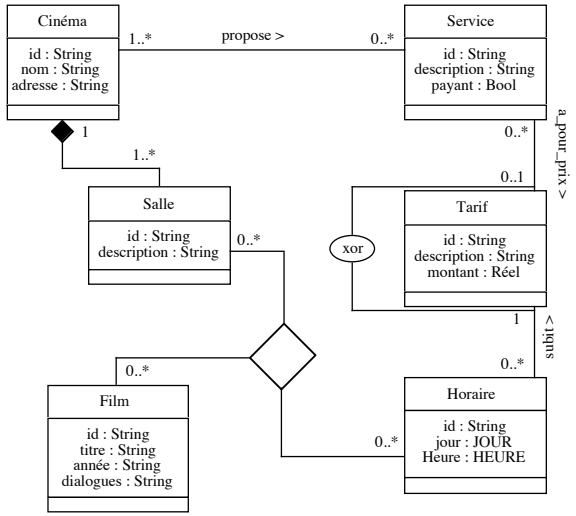
Remarque StarUML Là encore, le logiciel StarUML nous a posé un problème (d'où le recours à MacDraw). La représentation de la qualification ne semble, en effet, pas possible.

Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)

Exercice 3.4

Ci-après une solution possible :



Remarque StarUML Ce schéma ne peut pas être réalisé, comme cela, avec StarUML. Il contient une association ternaire et ce type de relation, non encouragé par la notation UML, n'est tout simplement pas proposé par StarUML. Il contient aussi une contrainte xor (un ou exclusif) que nous savons mal exprimé par le logiciel, sauf au travers d'une note. Nous avons donc préféré renoncer et avons conçu ce schéma avc Macdraw.

Voici les quelques explications que l'on peut donner :

- une association ternaire est utilisée car les associations binaires suivantes ne "marchent" pas :

• association ((0..*) - (0..*)) entre *Film* et *Salle* : pas de problème ;
• association ((0..*) - (0..*)) entre *Film* et *Horaire* : on ne sait pas à quelle salle correspond quel horaire ;
• association ((0..*) - (0..*)) entre *Salle* et *Horaire* : on ne sait pas à quel film correspond quel horaire.

- la contrainte *xor* sert à exprimer le fait qu'un tarif est soit celui d'un horaire, soit celui d'un service, pas des deux, mais l'un ou l'autre.

- la cardinalité minimum "du côté" de *Salle* "vers" *Cinéma* est 1 car il y a au moins une salle dans le cinéma.

- le couple de cardinalités "du côté" de *Tarif* "vers" *Service* est (0..1) car un tarif est payant ou non.

- la cardinalité minimum "du côté " de *Tarif* "vers" *Horaire* est 1 car le tarif est fonction de l'horaire.

Exercice 3.5

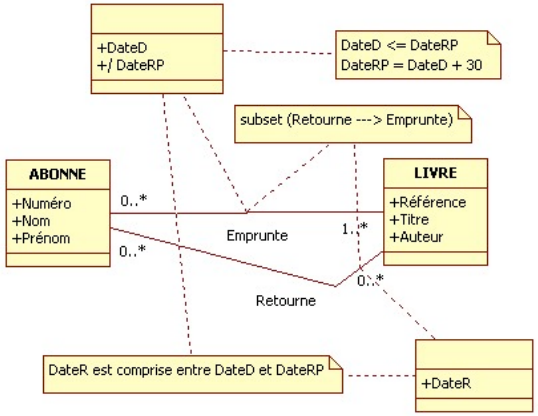
a) Compte-tenu du texte et de notre expérience, nous proposons les cardinalités suivantes :

que nous justifierons, si besoin est, de la façon suivante :

- un livre peut ne pas être emprunté (au début de son acquisition par la bibliothèque, pendant un laps de temps…) ; il peut l'être par plusieurs personnes… l'une après l'autre.
- un abonné ne sera inscrit que s'il emprunte. Il va, bien évidemment, emprunter plusieurs ouvrages pendant les nombreuses années où il sera inscrit à la bibliothèque.

Comment traduire alors la contrainte « pas plus d'un livre à la fois » ? La façon la plus élégante est de placer un commentaire indiquant « pas plus d'un livre à la fois ». Mettre une cardinalité 1..1 reviendrait à empêcher la mémorisation, par le système, des emprunts successifs.

b) Il nous faut ajouter une date d'emprunt, une date de retour au plus tard (elle peut être calculée) et une variable qui permettra de bloquer tout prêt pour l'abonné indélicat. Nous allons également (mais ce n'est pas indispensable) placer une seconde relation entre *Abonné* et *Livre*, pour le retour. Date d'emprunt et date de retour au plus tard sont des attributs de la relation *Emprunte*. La date de retour effectif est, quant à elle, un attribut de la relation *Retourne*. Il nous faut, enfin, une contrainte d'inclusion entre les deux relations pour éviter d'enregistrer des retours sans emprunts.

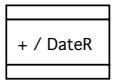


Remarque StarUML Là encore, nous avons dû utiliser une note pour représenter la contrainte d'inclusion existant entre *RETOURNE* et *EMPRUNTE*. Le fait que l'attribut DateR est calculé est matérialisé dans le nom de l'attribut (par le caractère /), ce qui est dommage.

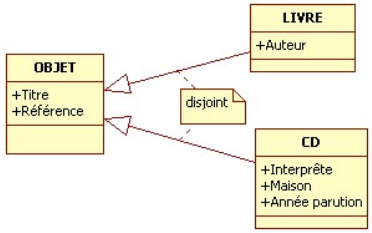
Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)
Région PAYS DE LA LOIRE

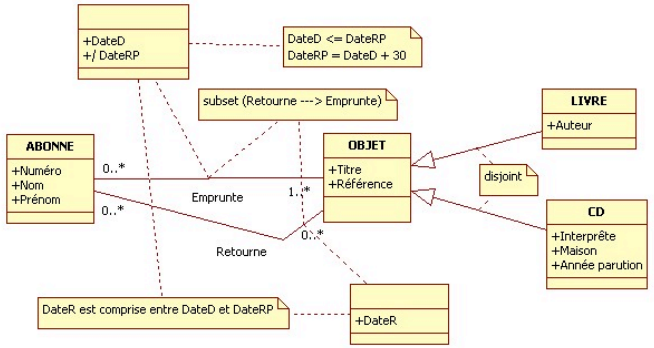
Ce que nous aurions aimer pour dire est ceci :



c) Les objets empruntables sont maintenant de deux natures possibles : livre ou CD. Le plus simple est d'utiliser l'héritage :

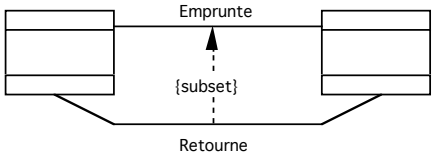


d) Dans la mesure où l'historique des emprunts est déjà prévu (il est possible de le mettre en place à partir de la réponse à la question a), nous devons essentiellement prendre en compte la notion d'avis donné par un lecteur sur un objet emprunté. La solution la plus simple est sans doute d'ajouter une propriété dans la classe-association *Retourne*. La limite de trois ouvrages empruntés simultanément sera exprimée dans un commentaire. Le schéma final répondant à TOUTES les questions pourrait donc être le suivant :



Remarque StarUML La note placée entre les associations Emprunte et Retourne est peu explicite. Elle signifie que les occurrences de Retourne –ce sont toutes des couples (a, o) dans lesquels a est une abonné et O un objet- sont incluses dans l'ensemble des occurrences de Emprunte. Il n'y pas de retour sans emprunt, en quelque sorte.

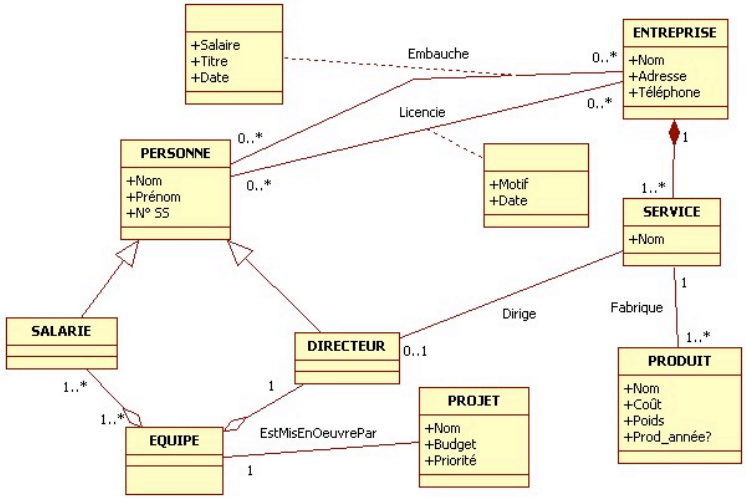
Voici la notation UML que nous avons voulu employer :



Partie III : exercices de modélisation « normaux »

Exercice 3.6

Pour résoudre ce « problème », nous allons procéder en deux temps. Les éléments contenus dans le texte vont d'abord servir pour produire une première ébauche de diagramme de classes :



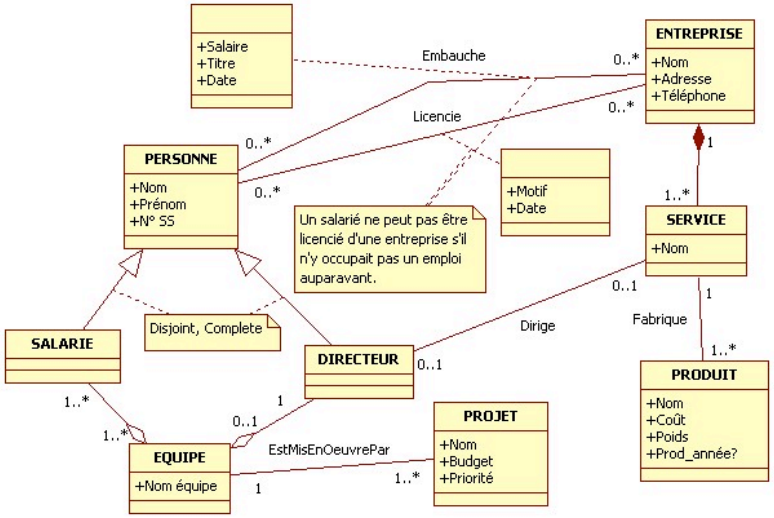
NB : une hypothèse (un employé et un salarié sont deux noms correspondant à la même réalité) a été faite pour obtenir cette ébauche.

Le deuxième temps va voir ce schéma enrichi, complété. Nous supposons qu'un salarié ne pouvait pas être directeur (ce qui justifie la contrainte d'exclusion entre ces deux entités) et que les personnes qui ne sont ni salariées, ni directeur ne figurent pas dans la base de données (ce qui justifie la contrainte de totalité).

Nous expliciterons, par une note, placée entre les associations *Embauche* et *Licencie* le fait qu'un salarié ne peut pas être licencié d'une entreprise s'il n'y occupait pas un emploi auparavant.

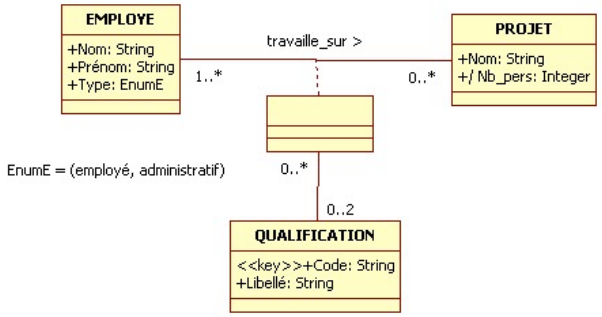
Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)



Exercice 3.7

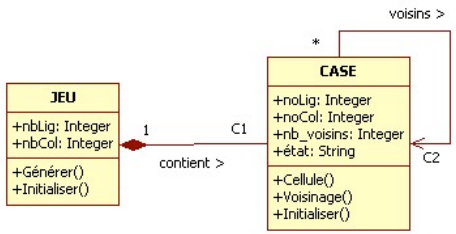
Il y a plusieurs solutions. En voici une :



Les variantes concernent la représentation du type d'employé. Celle-ci peut faire appel à un héritage (il faut alors mettre en place une contrainte de type *{disjoint, complete}*. Elles concernent également la modélisation de la liaison conceptuelle « autour » des classes *EMPLOYE* et *PROJET* que nous pouvons décrire comme une classe-association (c'est la solution que nous avons retenue), une relation ternaire ou une classe. Nous préfèrons la première.

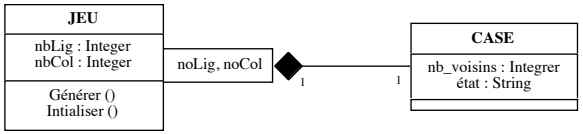
Exercice 3.8

Pour décrire les concepts mis en œuvre dans ce jeu de la vie, leur structuraton, il y a plusieurs solutions. Voici celle que nous avons retenue :



C1 = nbLig * nbCol
C2 : chaque case a entre 3 et 8 voisins.

Il y a d'autres alternatives, notamment la qualification pour désigner chaque case :



La modélisation des contraintes sera plus dificile. Les notions de ligne et de colonne sont, en outre, légèrement différentes. Absolues dans le premier cas, relatives à un jeu dans le cas de la qualification.

Exercice 3.9

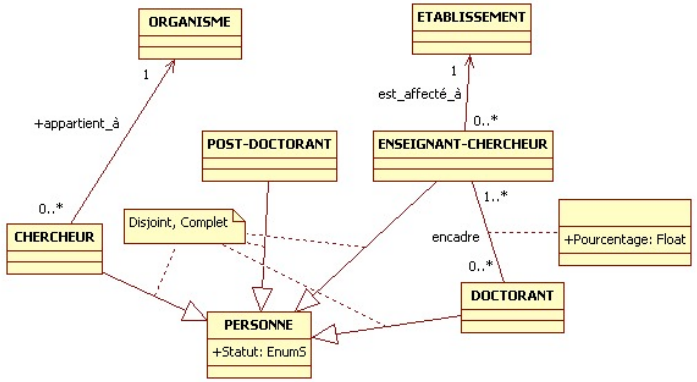
La structure de données minimale (on peut sans doute faire un peu plus) est constituée d'une classe *Personne*, avec un attribut *Statut* de type *EnumS* (énumération de valeurs prises par l'attribut).

NB : cet attribut EnumS est un ce que l'on appelle un type énuméré. On peut le définir en OCL ainsi :

EnumS = {Chercheur, Post-doctorant, Doctorant, Enseignant-chercheur}

Projet ENRC-DVD
MIAGE

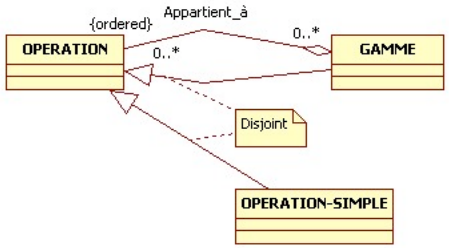
Module UML
(Outils de modélisation)



Exercice 3.10

a) L'énoncé définit ainsi la notion de gamme : « ... une gamme opératoire, c'est-à-dire une séquence d'opérations. Une opération est soit une opération élémentaire, soit une gamme opératoire, mais pas les deux. »

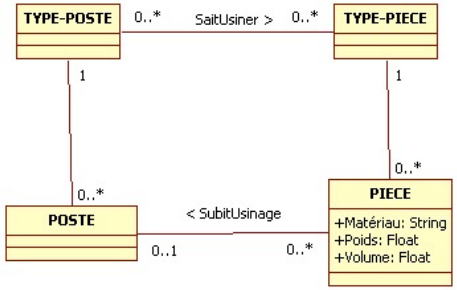
Deux notions se dégagent de ce (court) texte : l'*opération* et la *gamme*. Le texte apporte une précision supplémentaire, celle d'élémentarité. Certaines opérations sont elles-même des gammes. On est donc bien dans une problématique de modélisation de composants, composés, dont nous avons donné plusieurs solutions dans le recueil d'exercices corrigés que nous avons fait paraître (Exercices corrigés en UML ; passeport pour une maîtrise de la notation, Pascal ANDRE et Alain VAILLY, Éditions Ellipses, septembre 2003, ISBN 2-7298-1725-5). En voici une :



La notion de séquence (ie. un ensemble ordonné d'éléments) est traduite par la contrainte {ordered}. La contrainte d'exclusion (une opération est soit une gamme, soit une opération élémentaire, mais pas les deux) est naturellement traduite par la contrainte {disjoint}, du côté de la classe *Opération*.

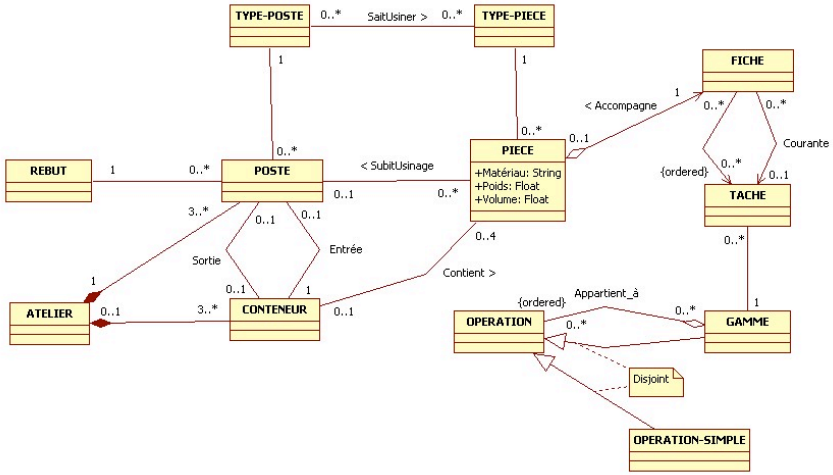
Les cardinalités sont fournies à titre indicatif. Elles sont, pour certaines, discutables.

b) Postes, types de postes, pièces et types de pièces sont les concepts à modéliser. Il est demandé de décrire les relations existant entre ces concepts. Le texte nous fournit deux indications : les pièces sont usinées sur les postes. On sait également qu'à chaque type de poste correspond des types de pièces qu'il sait usiner. Une réponse à la question posée est donc la suivante :



Certaines cardinalités sont, comme dans le schéma précédent, discutables. Les associations entre *Poste* et *Typeposte* et entre *Pièce* et *TypePièce* peuvent aussi être des relations d'instanciation (entre métaclasse et classe).

c) L'atelier est constitué des postes et des conteneurs. Il faut également prendre en compte des notions comme les tâches, les fiches et les rebuts. Le diagramme de classes final doit être constitué de tous les morceaux déjà fournis et de quelques autres. Voici une solution possible :



----------fin du texte----------