

REPUBLIQUE DE GUINEE

Travail- Justice- Solidarité

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET
DE LA RECHERCHE SCIENTIFIQUE**

CENTRE UNIVESITAIRE DE LABE



FACULTE DE SCIENCES ET TECHNIQUES

DEPARTEMENT INFORMATIQUE

ANNEE UNIVERSITAIRE 2021 – 2021

OPTION : INFOMATIQUE

« Manuel de cours d'Analyse des bases de données sous-objets ; cas d'UML destiné aux Etudiants L2 Informatique. »

Présenté par **Abdoulaye sow**

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

Plan de cours UML

I- Introduction à la modélisation Objet

1. Introduction au Génie logiciel (1h)

- a. L'informatisation
- b. Les logiciels
- c. Le génie logiciel

2. Pourquoi et comment modéliser (2h)

- a. Notion de modèle et modélisation
- b. Objectif de la modélisation
- c. Terminologie de la modélisation
- d. Le cycle de vie d'un logiciel
- e. Le découpage 4 « tiers » d'une application

3. Approche Objet (3h)

- a. Introduction
- b. Caractéristiques
 - i. Identité
 - ii. Classification
 - iii. Polymorphisme
 - iv. Héritage
- c. Concepts orientés objet
 - i. Abstraction
 - ii. Encapsulation
 - iii. Partage

4. UML (1h)

- a. Introduction
- b. Points forts d'UML
- c. Création et contributeur
- d. UML en Œuvre

II- Les Diagrammes UML

a. Diagramme des cas d'utilisation (5h)

- i. Introduction
- ii. Eléments des diagrammes de cas d'utilisation
 - 1. Acteur
 - 2. Cas d'utilisation
 - 3. Représentation d'un diagramme de cas d'utilisation
- iii. Relation dans les diagrammes de cas d'utilisation
 - 1. Relation entre acteur et cas d'utilisation
 - 2. Relation entre cas d'utilisation
 - 3. Relation entre acteur
- iv. **Notions générale du langage UML**
 - 1. Note
 - 2. Stéréotype
 - 3. Classeur
 - 4. Paquetage
 - 5. Espace de noms

v. Modélisation des besoins avec UML

1. Comment identifier les acteurs
2. Comment recenser les cas d'utilisations

Travaux Dirigés 1 (1h)

Exposé Groupe 1.2 (4h)

b. Diagramme de classe (4h)

- i. Introduction
- ii. Les classes
 1. Notions de classe et d'instance de classe
 2. Notions de propriétés
 3. Représentation graphique
 4. Encapsulation, visibilité, interface
 5. Nom d'une classe
 6. Les attributs
 7. Les méthodes
- iii. Relations entre classes
 1. Généralisation et Héritage
 2. Association
 3. Multiplicité ou cardinalité
 4. Navigabilité
 5. Quantification
 6. Classe – association
 7. Agrégation
 8. Composition
 9. Dépendance
- iv. Interface
- v. Elaboration d'un diagramme de classe

c. Diagramme d'objet (1h)

- i. Présentation
 - ii. Représentation
- Travaux dirigés 2 (1h)
- Exposé groupe 3 (4h)

d. Diagramme d'interaction (1h)

- i. Présentation du formalisme
- ii. Introduction
- iii. Classeur structuré
- iv. Collaboration
- v. Interaction et ligne de vie
- vi. Représentation générale

e. Diagrammes de communication (2h)

- i. Représentation de ligne de vie
- ii. Représentation des connecteurs
- iii. Représentation des messages

f. Diagramme de séquence (2h)

- i. Représentation des lignes de vie
- ii. Représentation des messages
- iii. Fragments d'interaction combinés

Travaux dirigés 3 (1h)
Exposé groupe 5,6 (4h)

g. Diagramme d'Etats transition (8h)

- i. Introduction au formalisme
 - 1. Présentation
 - 2. Notion d'automate à états finis
 - 3. Diagrammes d'Etats-transitions
- ii. Etat
 - 1. Les deux acceptions du terme d'Etat
 - 2. Etat final
- iii. Evénement
 - 1. Notion d'événement
 - 2. Evénement de type signal (signal)
 - 3. Evénement d'appel (call)
 - 4. Evénement de changement (change)
 - 5. Evénement temporel (after ou when)
- iv. Transition
 - 1. Définition et syntaxe
 - 2. Condition de grade
 - 3. Effet d'une transition
 - 4. Transition d'achèvement
 - 5. Transition interne
- v. Point d choix
 - 1. Point de jonction
 - 2. Point de décision
- vi. Etats composites
 - 1. Présentation
 - 2. Transition
 - 3. Etat historique
 - 4. Interface : les points de connexion
 - 5. Concurrence

Exposé groupe 7,8 (5h)

III- : Conclusion sur la méthode UML

Titulaire du cours : M. Abdoulaye sow

I- Introduction à la Modélisation Objet

La notion d'objet :

La **modélisation objet** consiste à créer une représentation informatique des éléments du monde réel auxquels on s'intéresse, sans se préoccuper de l'implémentation, ce qui signifie *indépendamment d'un langage de programmation*.

La **programmation orientée objet** consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel (que l'on appelle *domaine*) en un ensemble d'entités informatiques. Ces entités informatiques sont appelées *objets*. Il s'agit de données informatiques regroupant les principales caractéristiques des éléments du monde réel (taille, couleur,...)

La difficulté de cette modélisation consiste à créer une représentation abstraite, sous forme d'objets, d'entités ayant une existence matérielle (chien, voiture, ampoule, ...) ou bien virtuelle (sécurité sociale, temps, ...).

1. Introduction au Génie logiciel

a. L'Informatisation

C'est un processus qui permet de passer de la gestion manuelle (humains) à la question automatique (machine).

Les investissements du système d'information d'une entreprise se composent de matériels et de logiciel : Ils se répartissent généralement de la manière suivante :

- 20% dans le Matériel
- 80% dans les logiciels

b. Les logiciels

Un logiciel ou une application est un ensemble de programmes, qui permet à un ordinateur ou à un système informatique d'assurer une tâche ou une fonction en particulier (ensemble d'instruction).

Exemple : Windows, logiciel Gestion de Prêt, logiciel de comptabilité

En 1995 une Etude du Standish Group dressait un tableau statistique mené sur un échantillon représentatif de 365 entreprises, totalisant 8380 applications ou projet. Ces études établissent que :

- 16,2% seulement des projets étaient conformes aux prévisions initiales
- 52,7% avaient suivis de dépassements en coût et délai.
- 31,1% ont été purement abandonnés durant leurs développements.

D'autres études les grandes entreprises indiquent que le taux de réussite est de 9% seulement 37% des projets sont arrêtés en cours de réalisation, 50% aboutissent hors délai et hors budget et 4% non entamé.

L'examen des causes de succès et d'échec est instructif : la plupart des échecs proviennent non seulement de l'Informatique, mais du maître d'Ouvrage (les dirigeants et les concepteurs des métiers).

c. Le Génie logiciel

Le génie logiciel est un domaine de recherche qui a vu le jour suite à la crise des logiciels dans les années 70. Il a pour objectif de répondre à un problème qui s'énonçait à deux niveaux : d'une part le logiciel n'était pas fiable, d'autre part, il était difficile de réaliser dans des délais prévus des logiciels satisfaisant leur cahier des charges

C'est un ensemble de connaissance permettant d'encadrer le processus de développement des applications (Analyse,..., Maintenance).

Il permet d'optimiser le coût de développement de logiciel.

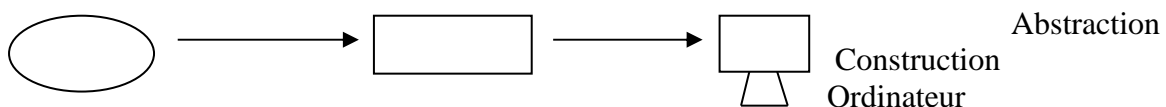
2. Motif et comment modéliser

a. Notion de modèle et de modélisation

Un modèle est une représentation **abstraite** et simplifiée, d'une entité du monde réel en vue de le décrire, de l'expliquer ou de le prévoir.

Exemple : Modèle météorologique, modèle économique, Modèle démographique.

Exemple de la Modélisation



La Modélisation est le processus qui permet d'élaborer les modèles.

b. Objectifs de la Modélisation (OM):

Ils permettent de :

- Comprendre les systèmes complexes.
- Eliminer les ambiguïtés
- Définir les besoins des utilisateurs
- D'encadrer le processus allant de la définition des besoins à l'implantation, l'exploitation et suivis.

c. Terminologie de la Modélisation (TM) :

Maitre d'ouvrage (MOA): est une personne morale (entreprise, direction,...), une entité de l'organisation. Mais aussi un client qui commande un produit nécessaire à son activité.

Maitre d'œuvre (MOE) : est une personne morale (entreprise, direction,...), garante de la bonne réalisation technique des solutions. Mais aussi un fournisseur.

Celui qui réalise le projet est le maitre d'œuvre

Celui qui commande le logiciel est le maitre d'ouvrage.

- Le maitre d'ouvrage exprime les besoins
- Il valide les solutions proposées
- Et il valide les produits livrés.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

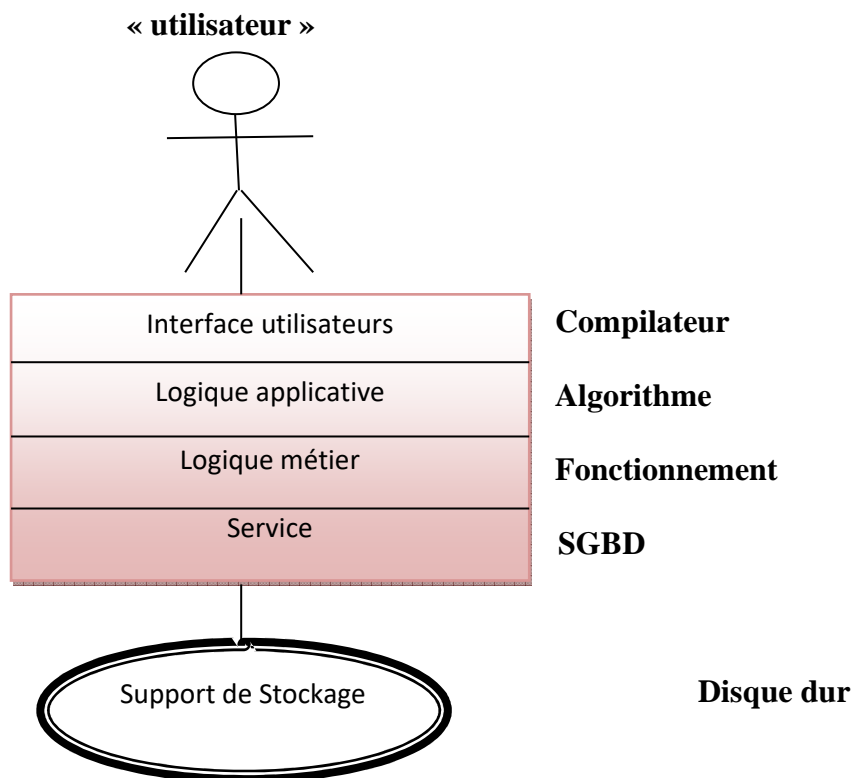
Tel : 622 29 43 67 / 666 97 25 67.

d. Le cycle de vie d'un logiciel (CVL) :

Le cycle de vie d'un logiciel (en anglais software lifecycle), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition.

Il comprend généralement au minimum les étapes suivantes :

- **Une phase d'analyse** : qui se décompose en deux (2)
 - o L'analyse du domaine : comprendre comment fonctionne l'entité ou Entreprise
 - o L'analyse de l'application.
 - **Spécification ou conception** : cette étape consiste à définir précisément chaque sous-ensemble du logiciel.
 - **Codage (implémentation ou programmation)** : c'est la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.
 - **Déploiement (intégration ou test)** : il permet de tester que chaque ensemble du logiciel est implémenté conformément aux spécifications.
 - **Maintenance** : elle comprend toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.
- e. Le découpage à 4 « tiers » d'une application**



3. Approche Objet (AO) :

a. Introduction :

L'approche Objet est une philosophie de programmation impulsée par les premiers langages qui sont :

Simula (1960), Smalltalk (1972 – 1980) par alan kay et Xerox PARC, C++, Java (SUN), etc...

L'une des particularités de cette approche objet est qu'elle rapproche les données et leurs traitements associés au sein d'un seul objet, applicable aux trois paradigmes (modèle) de langages de programmation :

Programmation impérative (tout est « instruction ») c++

Programmation fonctionnelle (tout est « fonction ») ex : camel

Programmation logique (tout est « prédicat ») expression logique dont la valeur peut être vraie ou fausse, selon la valeur des arguments. **Exemple : prologue**

Remarque : chaque objet aura une structure et un fonctionnement (procédure et fonction).

b. Caractéristiques de l'approche objet :

Un objet est caractérisé par plusieurs notions :

- i. **Identité :** signifie que les données d'un système sont quantifiées en entités discrètes que l'on peut distinguer et qu'on appelle Objet.

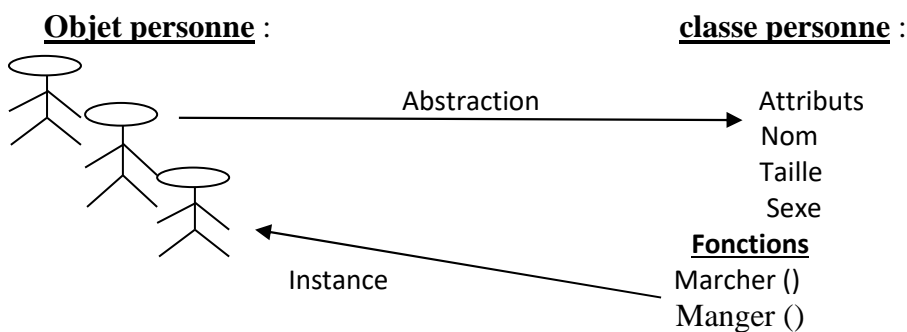
❖ Chaque objet possède sa propre identité, même si ces objets ont des attributs à valeur identiques qui sont distincts. **Exemple :** objet personne.

❖ Chaque objet possède une valeur unique (clé) permettant de l'identifier sans ambiguïté :

Exemple : indice d'un tableau ou valeur d'un attribut.

- ii. **Classification :** ce sont les objets ayant les mêmes structures (attributs) et même comportements (opérations) qu'on regroupe dans une classe.

Exemple :



L'objet est une instance de la classe qui, à son tour, est une abstraction de l'objet.

- L'abstraction décrit les propriétés pertinentes de l'application en ignorant d'autre.

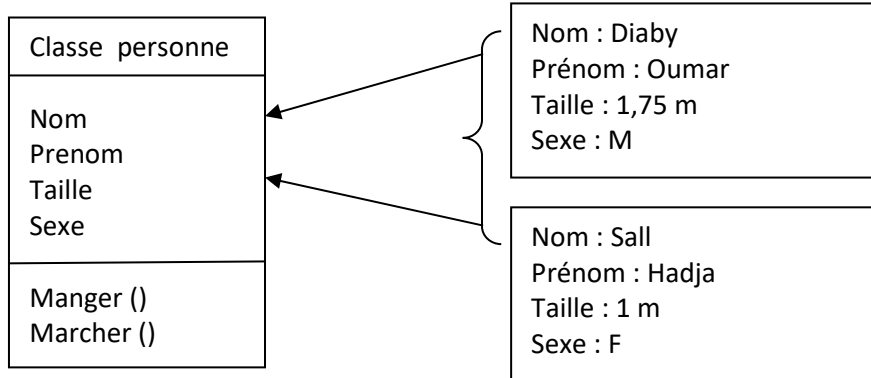
Différence entre Classe et Objet (DECO) :

Un objet est considéré comme instance de sa classe.

- ❖ Chaque instance partage le nom de ses attributs et opérations avec les autres instances de sa classe et possède ses propres valeurs.
- ❖ Chaque objet possède une référence vers sa classe « chaque objet sait ce qu'il est ».

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

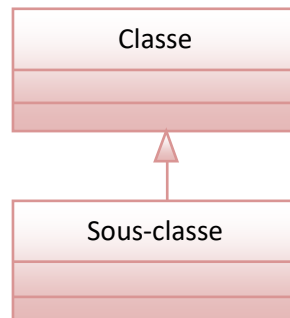
Exemple :**Instance****iii. Le polymorphisme :**

- ❖ Une opération (méthode, fonction) est une action qu'effectue un objet, ou une transformation que subit cet objet.
- ❖ Le polymorphisme signifie que la même opération se comporte différemment sur différents classes.

Exemple : déplacer () pour un personnage et déplacer () pour une fenêtre informatique.

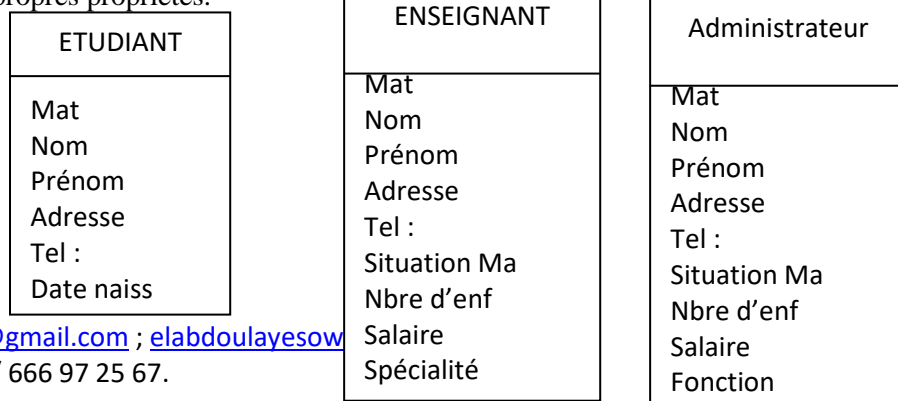
Chaque objet « sait comment » effectuer ses propres opérations.

Le langage sélectionne automatiquement l'opération à exécuter en se basant sur le nom de l'opération et la classe de l'objet.

iv. Héritage : « on ne réinvente pas la roue »

- l'héritage signifie les partages des attributs et opérations en se basant sur une relation hiérarchique.
- Chaque sous-classe incorpore ou hérite toutes les propriétés de sa (ses) superclasse (s) et y ajoute ses propres propriétés.

Exemple :

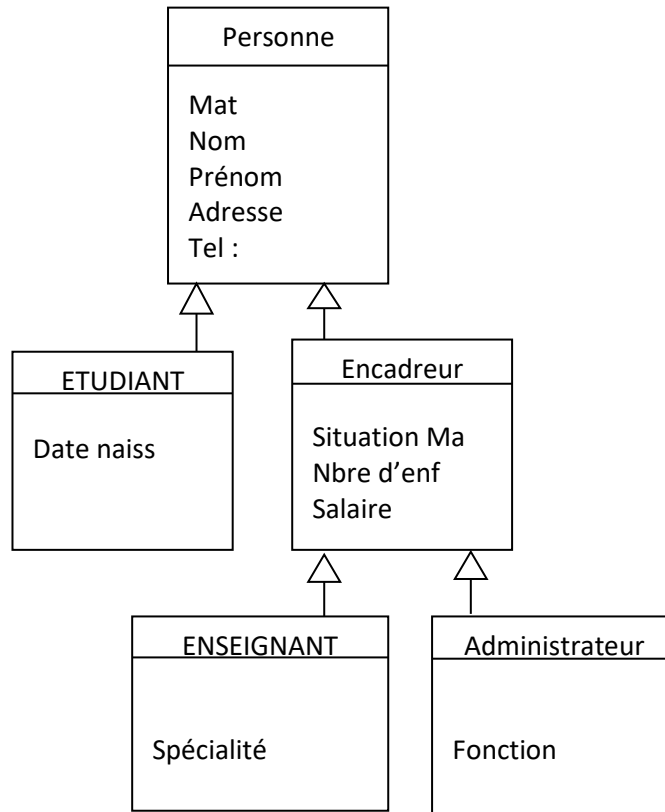


E-mail : tibousow@gmail.com ; elabdoulayesow
 Tel : 622 29 43 67 / 666 97 25 67.

Héritage :

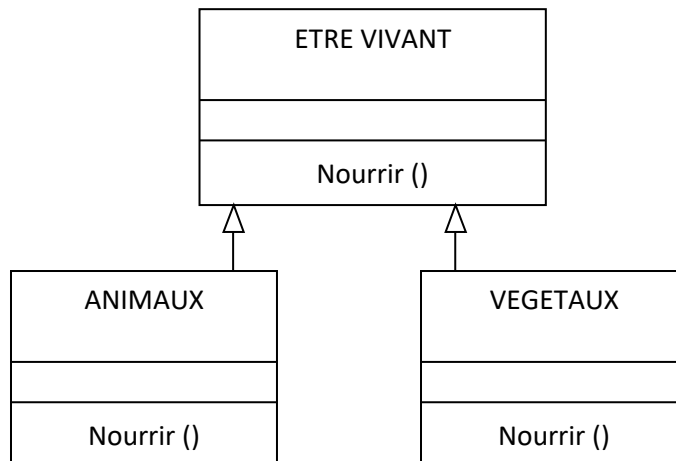
Sur classe

Sous classe



- **Surcharge de méthode :** surcharger une méthode c'est la redéfinir pour les sous classes ou la réunion de classe d'un même domaine en paquetages.

Exemple :

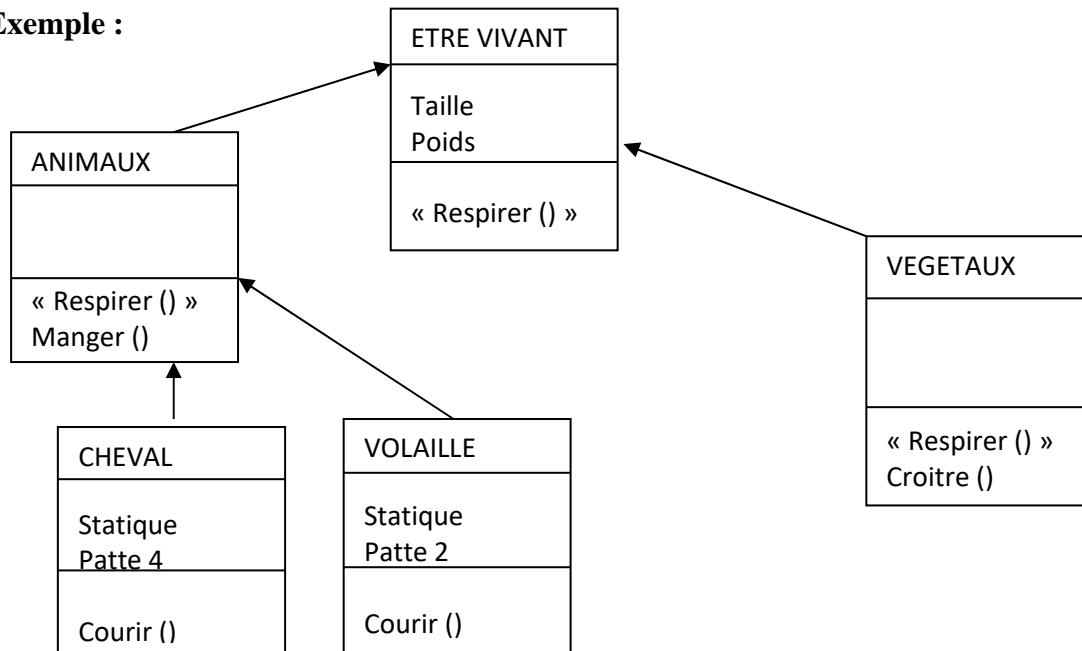


Un paquetage : c'est un ensemble de classe opérant sur un même domaine.

Avantage :

Héritage nous permet de réunir les classes d'un même domaine en paquetages

Inconvénient : c'est la mauvaise gestion de l'analyse

Exemple :

- **Classe abstraite et polymorphisme :**
- **Classe abstraite :**

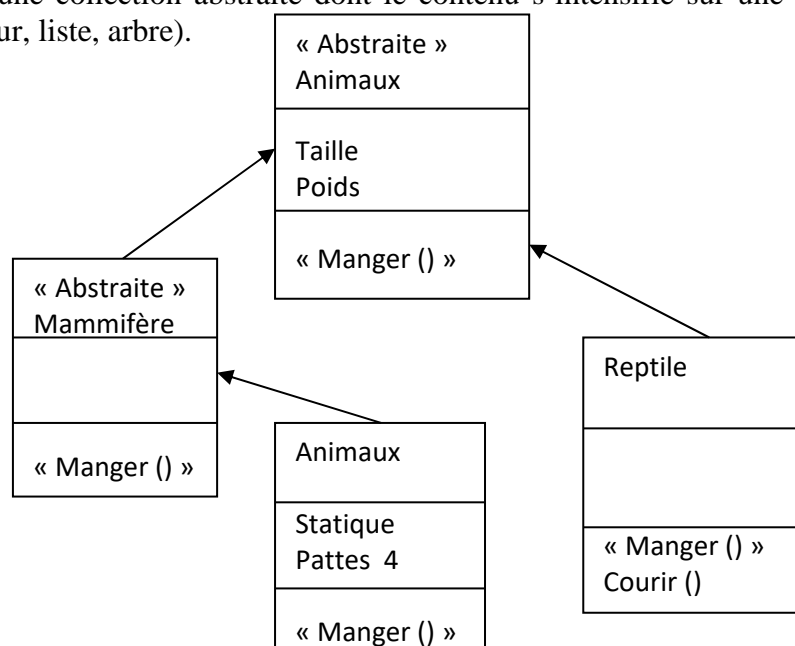
Une classe abstraite est celle qui à une fonction abstraite, c'est une classe sans instance.

Les comportements sont déclarés dans les superclasses et sont définits dans les sous classes.

Dans une classe, lorsqu'un comportement (fonction) est abstrait, alors, la classe est abstraite.

Un comportement est abstrait lorsqu'il est polymorphe (redéfinit).

- **Polymorphe :** signifie des méthodes surchargées du point de vue des fonctions. C'est une collection abstraite dont le contenu s'intensifie sur une classe quelconque (vecteur, liste, arbre).

Exemple :

Héritage Multiple (HM):

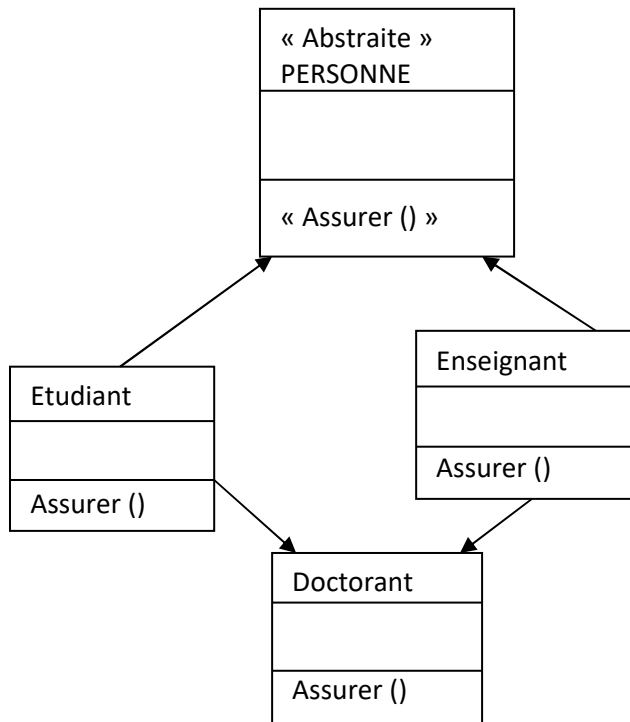
C'est une sous classe de 2 ou plusieurs superclasses

Inconvénient : risque de confusion sur les méthodes

Deux solutions :

- 1- Définir un ordre dans les classes
- 2- Redéfinir la méthode

Exemple :



c. Concepts forts de la philosophie objet (CFPO):

i. Abstraction

- Regrouper les données et les comportements (réunion)
- Partage (héritage).
- Structurer avant de construire (classification)

ii. Encapsulation : c'est donner un accès choisi aux informations

Il existe 4 modes d'accès :

1. Publics (+) : accessible par les objets de toutes les classes y compris les objets de même classe.
2. Privé (-) : accessible seulement par les objets (fonction) de la classe.
3. Protégé (#) : accessible par les objets de la classe et des sous classes qui l'héritent.
4. Paquetage (~) ou rien: accessible par tous les objets appartenant au même paquetage ou la classe.

iii. Masquer les informations :

Lorsque les données ne sont manipulables par l'extérieure signifie que l'on a fait une abstraction vis-à-vis de l'extérieure.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

4. UML (Unified Modeling Language)

a. Introduction :

La description de la programmation par objets a fait ressortir l'étendue du travail conceptuel nécessaire : définition des classes, de leurs relations, des attributs et méthodes, des interfaces etc.

Pour programmer une application, il ne convient pas de se lancer tête baissée dans l'écriture de code : il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et les étapes de la réalisation. Cette démarche est appelée modélisation ; son produit est un model.

UML c'est aussi :

- un langage de modélisation unifié
- Un langage de modélisation graphique orienté vers les systèmes et processus.

Il a été promu pour la première fois par l'OMG (Object Management Group) en novembre 1997.

b. Points forts d'UML :

UML vous permet de représenter les liens entre vos classes.

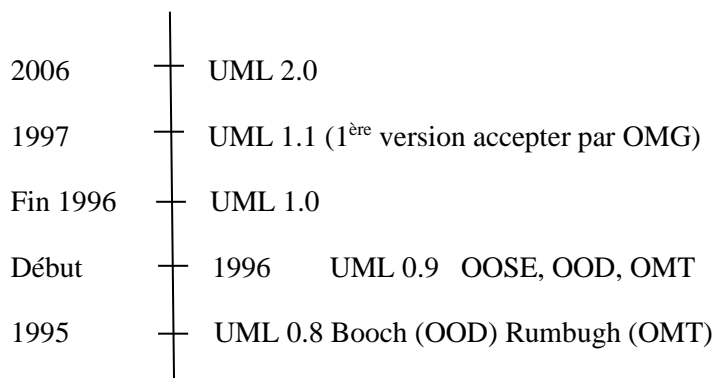
Vous pouvez y modéliser leurs attributs et leurs méthodes.

Il est orienté objet et c'est un standard (compréhensif à la fois par l'homme et la machine)

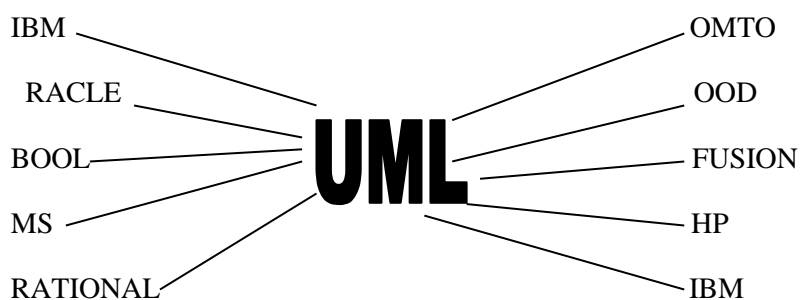
C. Création et Contributeur d'UML :

- création

Les méthodes utilisées dans les années 1980 pour organiser la programmation impérative (notamment Merise) étaient fondées sur la modélisation séparée des données et de traitement. Avec la programmation orienté objet la nécessité d'une méthode qui lui soit adaptée devient évidente. Plus de 50 méthodes apparaissent entre 1990 et 2006.



Contributeur:



E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

UML en Œuvre:

UML n'est pas une méthode (i.e. une description normative des étapes de la modélisation). C'est un ensemble de diagramme permettant de représenter les trois (3) vues d'un système qui sont :

- Fonctionnalités
- Vue Statique
- Vue Dynamique

UML définit aussi un langage graphique qui permet de représenter, de communiquer les divers aspects d'un système d'information. Il est donc un métalangage car il fournit les éléments permettant de construire le model qui, lui, sera le langage de projet.

UML 2.0 comporte ainsi treize types de diagrammes représentant autant de vues distinctes pour représenter des concepts particuliers du système d'information. Ils se représentent en deux grands groupes :

Diagrammes structurels ou diagrammes statiques (UML Structure)

- Diagramme de classes (Class diagramm)
- Diagramme d'Objets (Objects diagramm)
- Diagramme de composants (Compoent diagramm)
- Diagramme de déploiement (Deployment diagramm)
- Diagramme de paquetages (Package diagramm)
- Diagramme de Structures composites (Composite structure diagramm)

Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)

- Diagramme de cas d'utilisation (Use case diagramm)
- Diagramme d'activités (Activity diagramm)
- Diagramme d'états-transitions (State machine diagramm)
- Diagramme d'interaction (Interaction diagramm)
- Diagramme de séquence (Sequence diagramm)
- Diagramme de communication (Communication diagramm)
- Diagramme global d'interaction (interaction overview diagramm)
- Diagramme de temps (Timing diagram)

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à l'occasion d'une modélisation.

LES DIAGRAMMES UML

I-Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation permet de capturer (recenser) les fonctionnalités (besoins, attentes) d'un système. C'est un outil de consensus sur le cahier de charge.

Il permet de capturer les comportements du système suivant une vue externe.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

C'est la première étape pour développer un logiciel conforme aux attentes des utilisateurs.

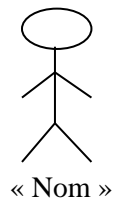
Pour élaborer le diagramme de cas d'utilisation, il faut se fonder sur entretiens avec les utilisateurs du système.

Les éléments d'un diagramme de cas d'utilisation sont :

Acteur : C'est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec le système (plus qu'une personne physique).

Symbole : Il se représente par un petit bonhomme avec son nom (c'est-à-dire rôle) inscrit dessous.

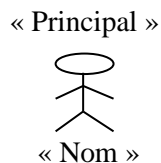
Représentation :



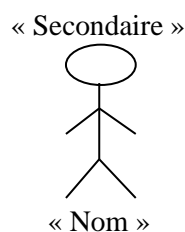
Les catégories d'acteurs :

Acteur principal : un acteur est dit principal lorsqu'il sollicite le système ou bien lorsque la fonctionnalité sollicitée lui est destinée. Si l'acteur est principal on met <<Principal>> au dessus de sa représentation.

Exemple :



Acteur secondaire : un acteur est dit secondaire lorsqu'il est sollicité par le système pour la réalisation d'une fonctionnalité (nécessaire pour la fonctionnalité). Il est représenté comme suit :



L'acteur secondaire est vide ou inexistant (peut être vide ou peut ne pas exister).

Cas d'utilisation

Un cas d'utilisation est une expression des fonctionnalités d'un système existant. Ce sont des services rendus par le système.

Un cas d'utilisation réalise un service de bout en bout avec un déclenchement, un déroulement et une fin.

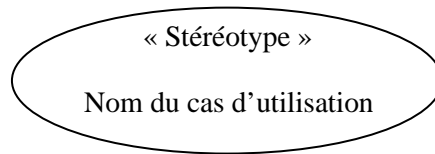
Remarque : le diagramme de cas d'utilisation ne montre pas comment la fonctionnalité est réalisée.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

Représentation : un cas d'utilisation est représenté par une ellipse contenant le nom du cas d'utilisation (verbe à l'infinitif) et optionnellement au dessus d'un stéréotype.

Exemple :



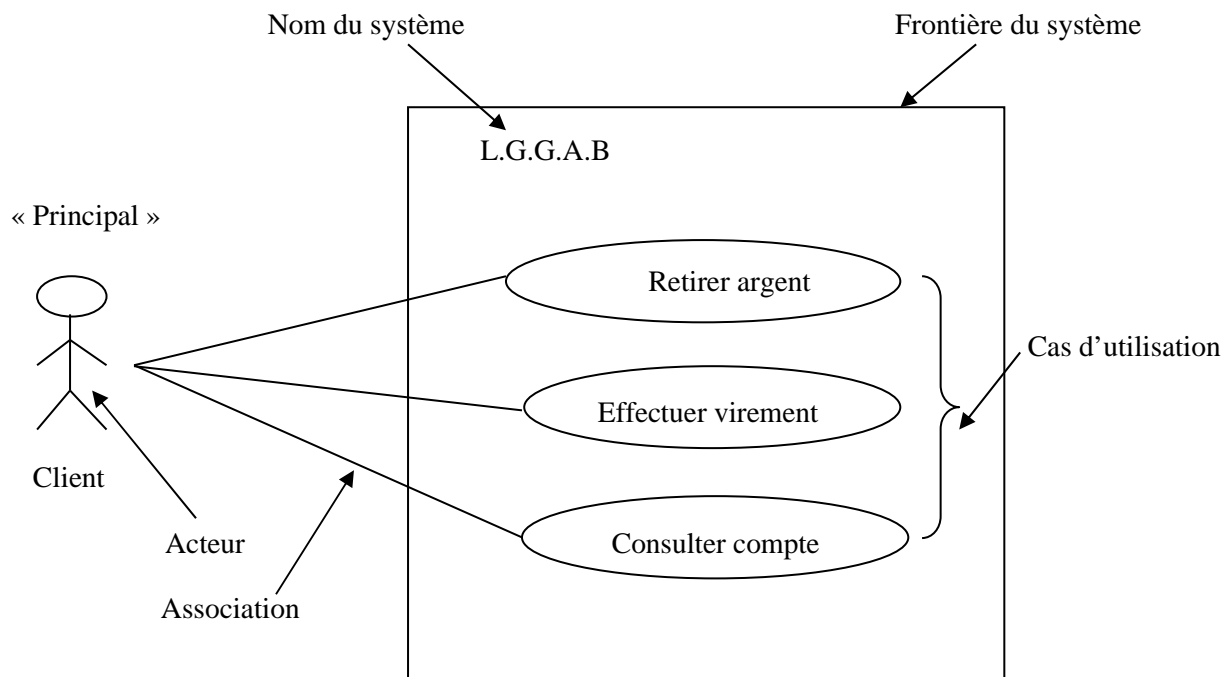
Représentation d'un diagramme de cas d'utilisation

Un diagramme de cas d'utilisation est un ensemble d'acteurs et de cas d'utilisation. Les frontières du système sont représentées par un cadre.

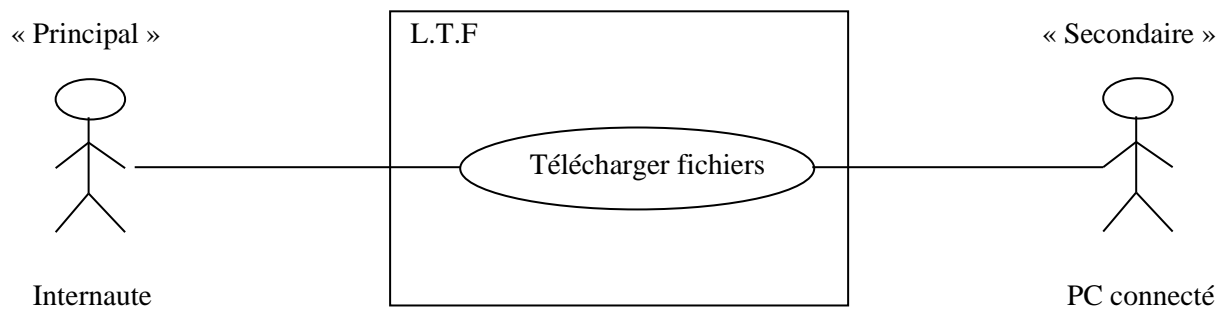
A l'intérieur du cadre nous aurons les cas d'utilisation et à l'extérieur les acteurs en haut du cadre (à l'extrême gauche) le nom du système.

L'acteur sera lié au cas d'utilisation par un trait continu qui montre que cet acteur sollicite le système à travers ce cas d'utilisation.

Exemple 1 : Soit un logiciel de gestion d'un guichet automatique de banque. Les fonctionnalités sollicitées par les clients sont : retirer de l'argent, effectuer un virement et consulter leur compte. Le diagramme de cas d'utilisation sera :



Exemple 2 : Représentons un logiciel de partage de fichiers à travers lequel un internaute télécharge des fichiers via un ordinateur connecté.



Relations entre les cas d'utilisation

On peut classer les relations entre le cas d'utilisation en deux types :

-Les relations stéréotypées : explicitées par un stéréotype. Les plus connues sont : l'inclusion et l'exclusion.

Représentation : Ce sont des flèches avec des traits discontinus.

-Les relations de généralisation ou de spécialisation :

Représentation : Ce sont des flèches avec un trait plein dont la pointe est un triangle fermé désignant le cas le plus général.

Relation d'inclusion

Un cas A inclus un cas B si le comportement décrit dans le cas A inclut le comportement décrit dans le cas B : Si A est sollicité B aussi est sollicité obligatoirement comme une partie de A.

Le stéréotype correspondant est « include ».

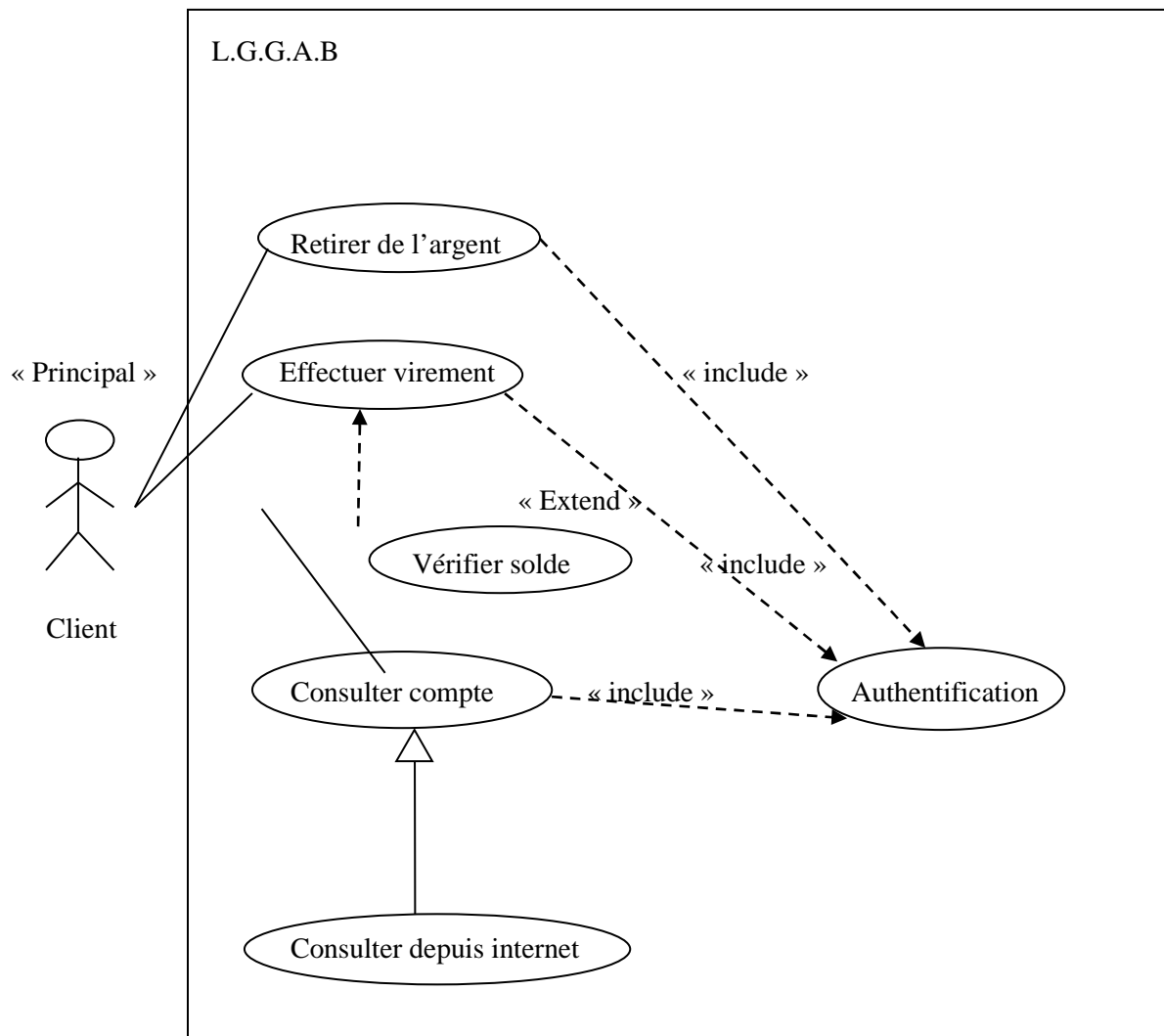
Les inclusions nous permettent de factoriser certains cas d'utilisation.

Relation d'exclusion

Un cas A exclut un cas B si l'appel de A peut s'étendre sur à un appel de B. Contrairement à l'inclusion, l'exclusion est optionnelle (facultative).

En d'autres termes l'appel de B est éventuel. Il est symbolisé par le stéréotype « extend ».

Exemple : Reprenons l'exemple 1 et considérons que les fonctionnalités retirer de l'argent, effectuer un virement et consulter compte passent obligatoirement par une authentification et le cas effectuer un virement donne la possibilité de vérifier son solde. Le client peut outre consulter son solde depuis internet.

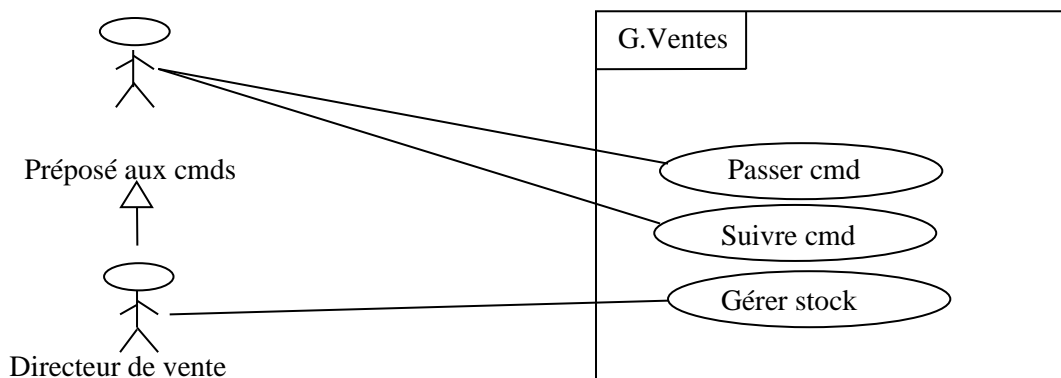


Relation entre les acteurs

-Généralisation : Un acteur A est une généralisation de l'acteur B, si on peut substituer (remplacer) l'acteur A par l'acteur B.

Tous les cas accessibles par A le sont par B mais l'inverse est faux.

Exemple : Le directeur des ventes est un proposé aux commandes avec un pouvoir supplémentaire : en plus de pouvoir passer et suivre une commande, il peut gérer le stock. Par contre, le proposé aux commandes ne peut pas gérer le stock.



II-Diagramme de classe (DC) :

Introduction : c'est une disposition statique des systèmes. Il permet de représenter l'architecture de nos logiciels.

Le diagramme de classe met en évidence des associations entre les éléments du diagramme.

Les associations sont de deux types :

- Relations hiérarchiques
- Relations de dépendance (liens de composition)

Le diagramme de classe sera enrichi à travers le typage des informations et le profil des informations.

Le classeur sera utilisé pour la représentation des classes.

Exemple :

Nom classe
Attributs
Méthodes

Typage et profil :

A chaque attribut on associe les types suivants :

- Le type scalaire (booléen, entier, réel, caractère, chaîne de caractères) ;
- Le type construit (par exemple une classe) ;
- Possibilité de donner une valeur nulle ;

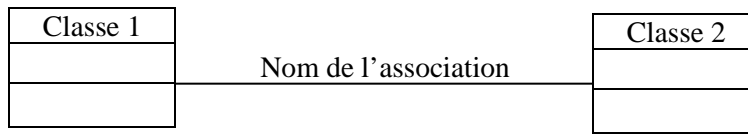
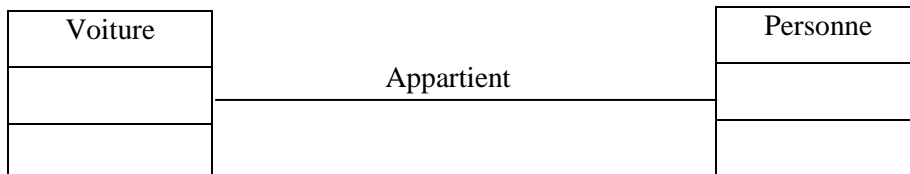
NB : Le typage concerne les attributs et les profils pour les méthodes.

A chaque méthode est associée :

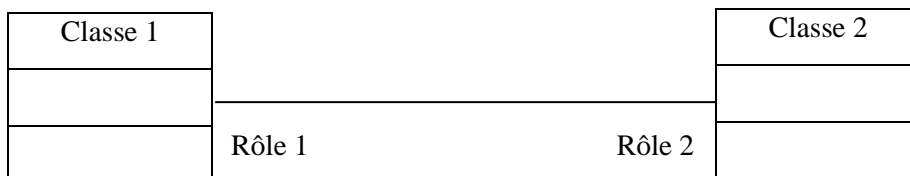
- Un type de retour (s'il en existe) ;
- Le type des paramètres (s'il en existe) ;
- Le statut des paramètres (paramètres de passage) : (in) pour les paramètres en entrée, (out) pour les paramètres en sortie et (in out) pour les paramètres en entrée et en sortie.

Relation entre les classes

-Les associations : Une association permet de relier deux classes. Elle peut porter un nom, elle peut être orientée pour diminuer les dépendances fonctionnelles.

Représentation :**Exemple :**

Le diagramme de classe nous donne la possibilité de spécifier des rôles pour chaque extrémité de l'association.

Représentation :

Sémantique : Une instance de « classe 1 » est un « rôle » pour une instance de « classe 2 ».

Exemple :

-Multiplicité des associations : La multiplicité d'une classe est le nombre de fois qu'une instance de la classe joue son rôle dans l'association.

Représentation :**Les différents types de multiplicité :**

-Exactement n

Représentation :

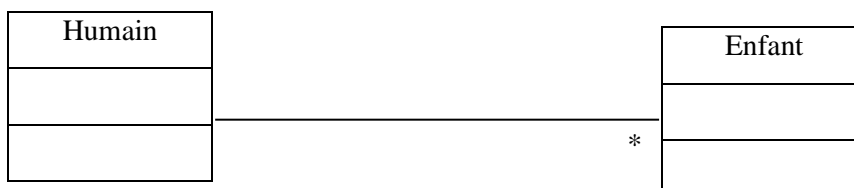
Sémantique : Une instance de « classe 1 » a exactement n instances de « classe 2 ».

Exemple :

Une instance de « triangle » a exactement 3 instances de point et une instance de point a exactement 1 instance de triangle.

-Plusieurs ou zéro :**Représentation :**

Sémantique : Une instance de « classe 1 » a plusieurs ou zéro instances de « classe 2 ».

Exemple :

Une instance de « Humain » a plusieurs ou zéro instances de « Enfant ».

-Entre n et m

Représentation :

Sémantique : Une instance de « classe 1 » a entre n et m instances de « classe 2 ».

Exemple :

Une instance de « voiture » a entre 4 et 7 instances de « siège ».

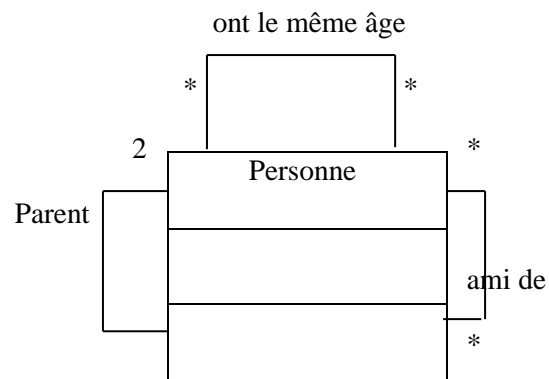
-Au moins n fois**Représentation :**

Sémantique : Une instance de « classe 1 » a au moins n instances de « classe 2 ».

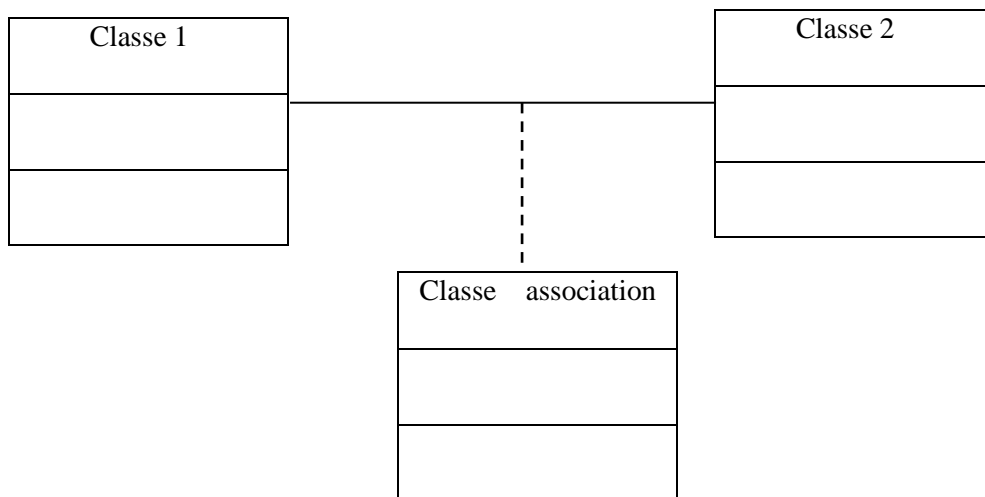
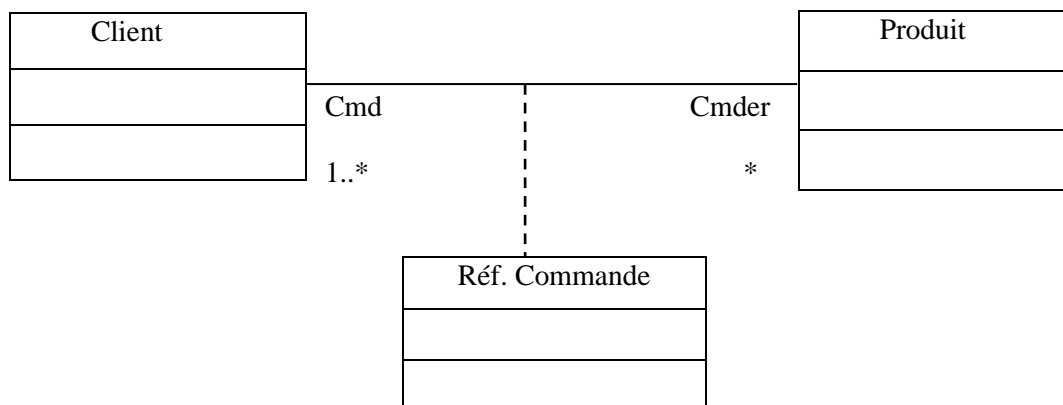
Exemple :

Une instance de polygone a au moins 3 instances de point.

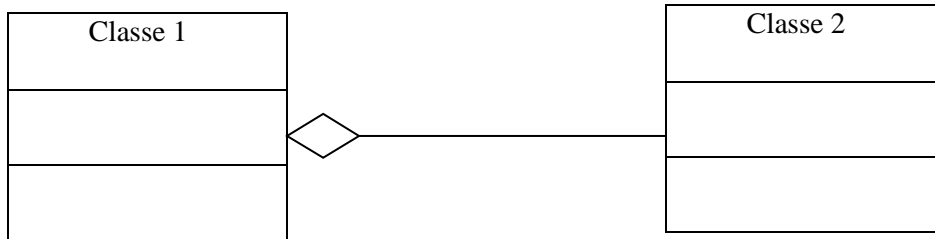
-Association réflexive : une association est réflexive lorsqu'elle relie une classe à elle-même.

Exemple :

-Classe association : une classe association est une association surchargée par une classe.

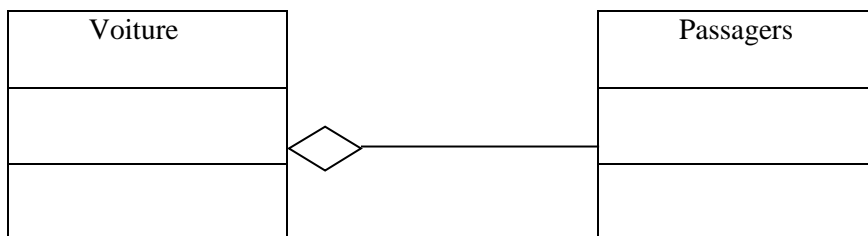
Représentation :**Exemple :**

-Composition faible ou agrégation : elle sert à représenter le fait qu'un objet A est une partie (inclus) dans un objet B.

Représentation :

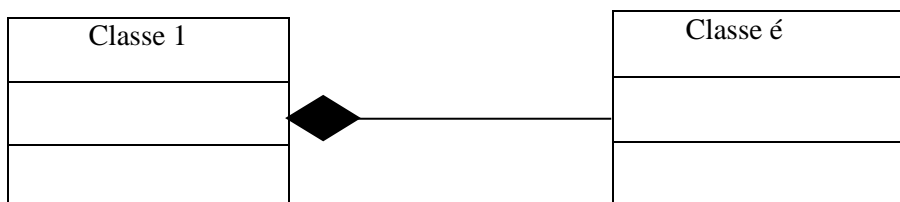
La destruction de l'objet n'entraîne pas la destruction du composant.

Sémantique : si une instance de « classe 1 » est détruite, l'instance de « classe 2 » ne l'est pas nécessairement.

Exemple :

La destruction de l'instance voiture n'est pas celle de l'instance passager associée.

-Composition forte : la destruction de l'objet entraîne la destruction du composant.

Représentation :

Sémantique : si une instance «classe 1 » est détruite alors l'instance « classe 2 » est détruite.

Exemple :

Si une instance de « Animal » est détruite, l'instance de « Organe » est détruite.

Enrichissement du diagramme de classe

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

- **(Objet Contrainte Language) L'OCL** : est un langage formel qui permet d'exprimer des contraintes sur certains éléments du modèle.

- **le diagramme d'objet (DO)** : il permet de donner des détails sur le diagramme de classe.

- **Les paquetages** : c'est ensemble des classes orientées dans le même domaine.

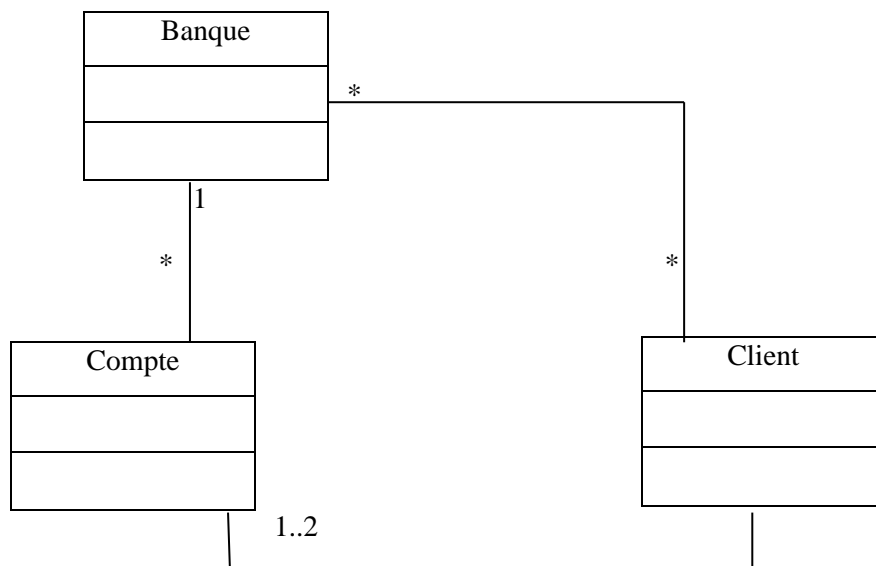
III- Diagramme d'objet

C'est la représentation des objets (instance et classe) et leur lien (instance des associations).

-Il permet d'illustrer le diagramme de classe (en montrant un exemple qui explique le modèle).

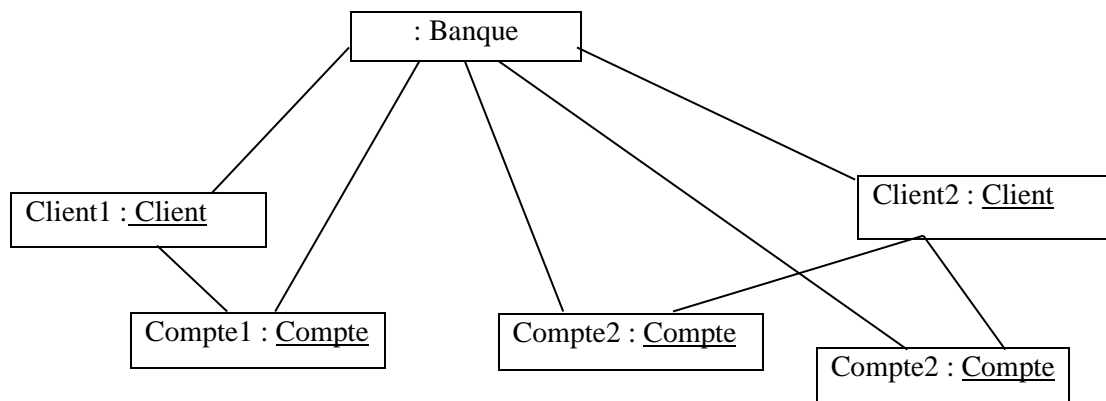
-Il donne une vue sur l'état du système à un instant donné (snapshot) mais aussi fait apparaître les exceptions.

Exemple : soit la gestion des clients d'une banque.

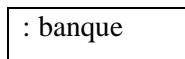


Diagrammes d'objet correspondant :

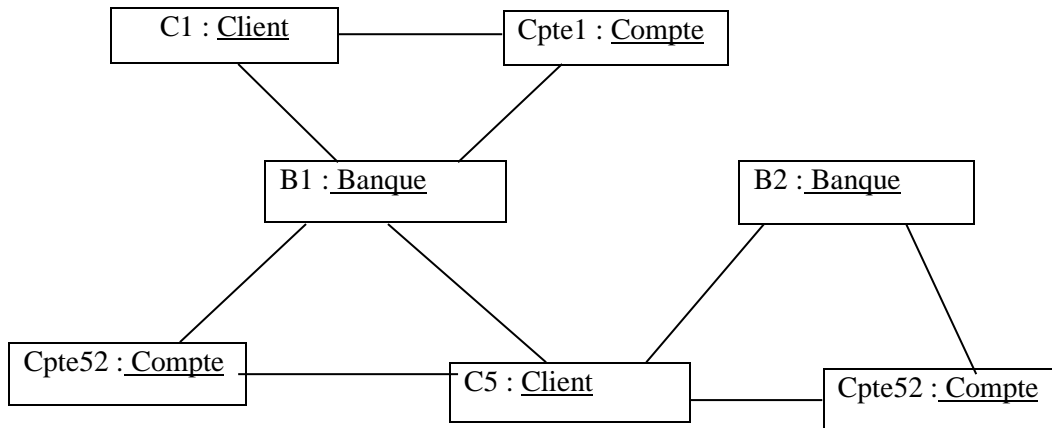
1-Compte



2-Banque



3-Client

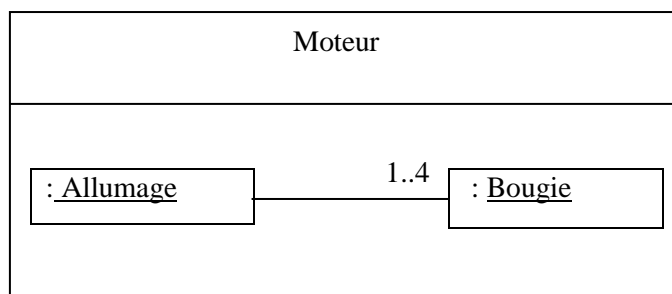
**IV-Diagramme d'interaction (DI)**

Introduction : les diagrammes d'interaction nous permettent de nous focaliser sur la dynamique de nos systèmes.

Les diagrammes d'interaction permettent d'établir un pont entre le diagramme de classe et le diagramme de cas d'utilisation. Il nous donne la possibilité de nous focaliser sur un sous ensemble d'élément du système et étudier leur façon d'interagir pour décrire un comportement particulier du système.

Classeur structuré : les classeurs structurés nous permettent de raffiner les classes découvertes au moment de l'analyse. Graphiquement le classeur structuré est représenté par un rectangle en trait plein comprenant deux compartiments. Le compartiment supérieur qui contient le nom du classeur et le compartiment inférieur les objets et leur connecteur.

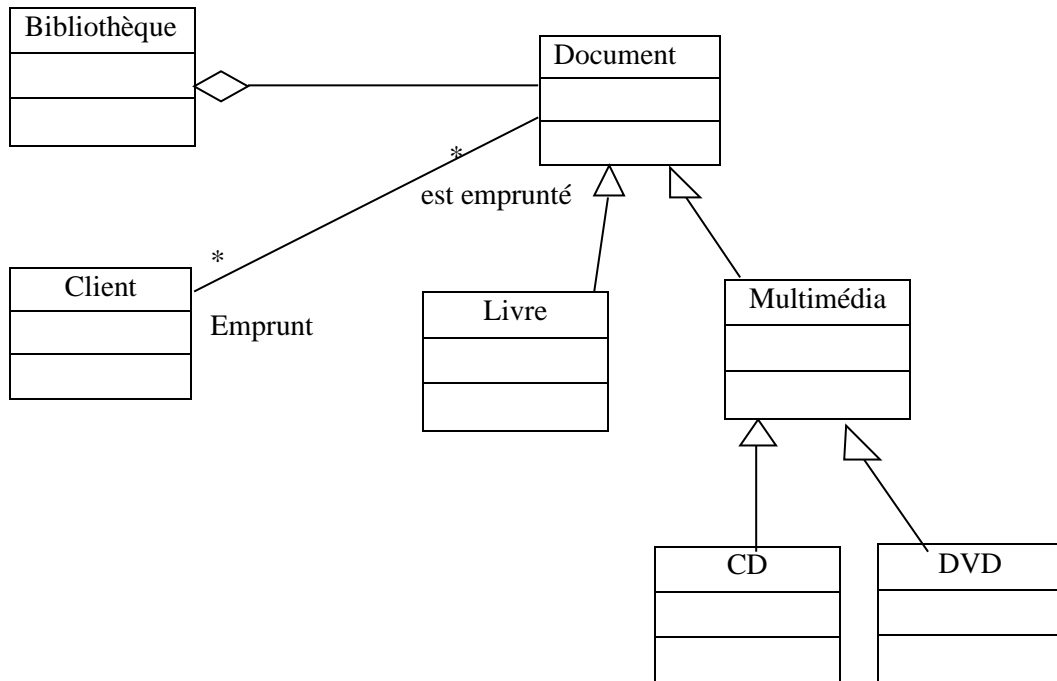
Exemple : pour le cas d'un moteur nous avons le classeur suivant :



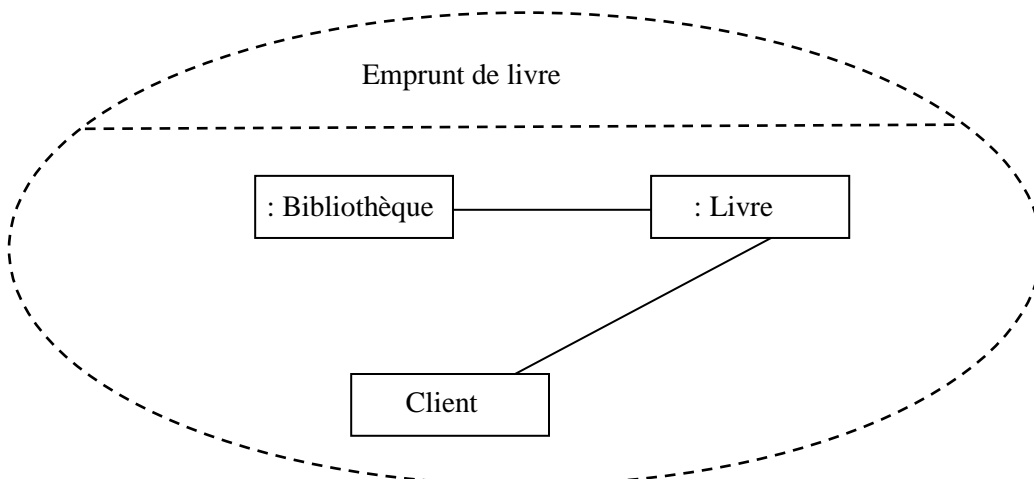
Collaboration : une collaboration montre les instances qui communiquent dans un contexte donné pour mettre en œuvre une fonctionnalité du système.

Il est représenté par une ellipse en trait pointillé comprenant deux compartiments. Le compartiment supérieur indique le nom de la collaboration et le compartiment inférieur montre les intervenants à la collaboration.

Exemple : soit un système de gestion d'une bibliothèque universitaire représenté par le diagramme de classe ci-dessous.



Intéressons-nous à la fonctionnalité « Emprunt de livre ». Les objets interagissant sont client, bibliothèque et livre. Le diagramme de collaboration correspondant est :



Interaction et ligne de vie

Une interaction est la représentation du comportement d'un classeur structuré ou d'une collaboration en faisant apparaître les échanges d'informations. Une interaction fait revenir un jeu de ligne de vie.

Une ligne de vie correspond à une partie interne d'une collaboration ou d'un classeur structuré en faisant apparaître les instances de nos objets à une période donnée.

Pour représenter les interactions UML propose deux diagrammes : le diagramme de séquence et de communication.

Représentation générale

Un diagramme d'interaction est représenté par un rectangle contenant dans le coin supérieur gauche un pentagone accompagné du mot clé «Sd » (Sequence Diagram) lorsqu'il s'agit d'un diagramme de séquence et « Com » (Communication) lorsqu'il s'agit d'un diagramme de communication. Le mot clé est aussi suivi du nom de l'interaction.

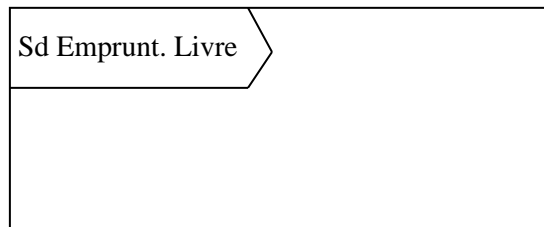


Diagramme de séquence

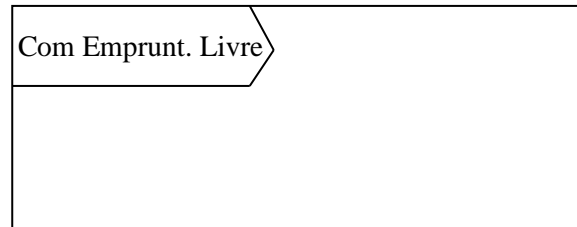


Diagramme de communication

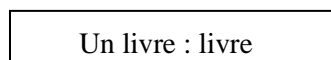
V-Diagramme de communication (DC) :

Il permet de représenter l'intervenant à la collaboration en prenant en compte le critère spatial.

Ligne de vie : les lignes de vie sont représentées par des rectangles contenant une étiquette dont la syntaxe est la suivante : [<Nom rôle >] : [<Nom. type>]

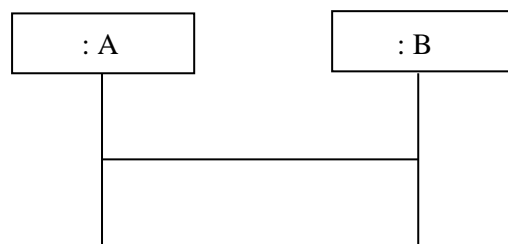
Au moins l'un des deux doit être spécifié dans l'étiquette et les deux points sont obligatoires.

Exemple :



Représentation des connecteurs

Les relations entre les lignes de vie sont appelées des connecteurs et sont représentés par un trait plein reliant deux lignes de vie.



Représentation des messages

Dans un diagramme de communication, les messages sont ordonnés suivant un numéro de séquence croissant.

Un message est spécifié généralement par la forme suivante :

[['cond'] [seq] ['*['||'] ['iter']] :][r :=] msg ([par])

Cond: condition d'émission du message.

Seq : le numéro de séquence du message. On numérote les messages par envoie et sous envoie désigné des chiffres séparés par un point.

Ainsi l'envoi du message 1.4.3 est antérieur à celui du message 1.4.4 mais postérieur à celui de 1.3.5
r : valeur de retour du message.

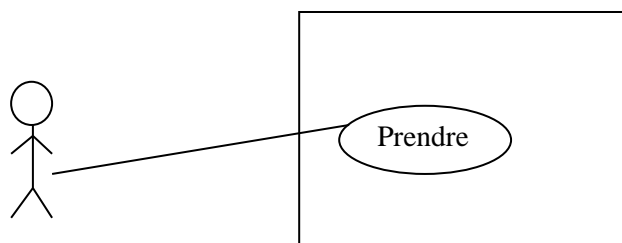
Msg : est le nom du message.

Par : désigne le paramètre passé au message (optionnel).

La direction d'un message est spécifié par une flèche pointant vers l'un ou l'autre des objets de l'interaction reliée par ailleurs avec un trait continu (connecteur).

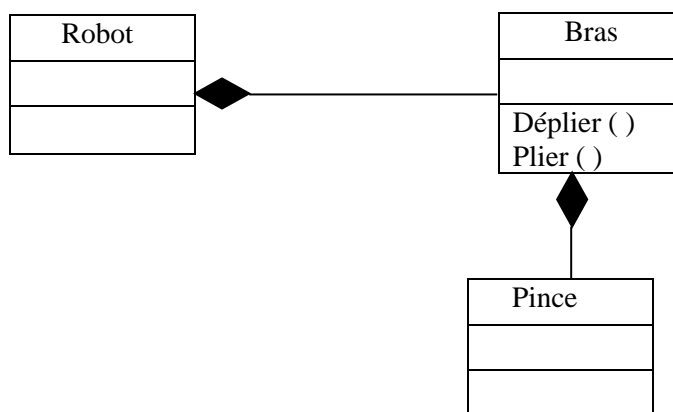
Exemple 1 : Revenons sur l'exemple introductif du cours traitant d'un robot avec un manipulateur automatique.

Représentons le cas d'utilisation :

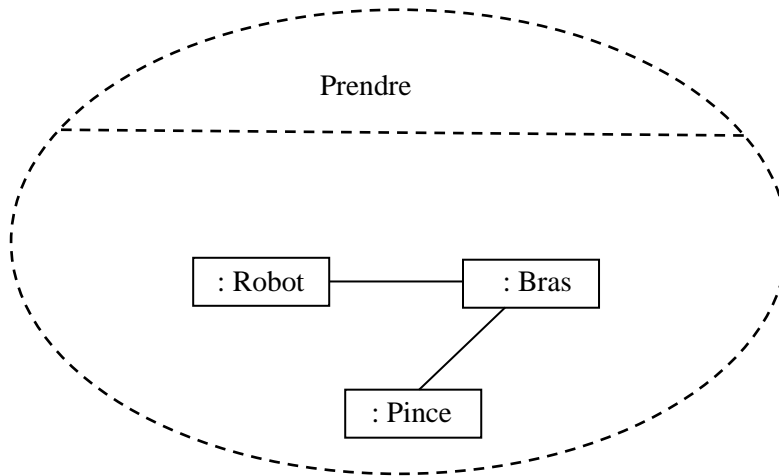


Manipulateur

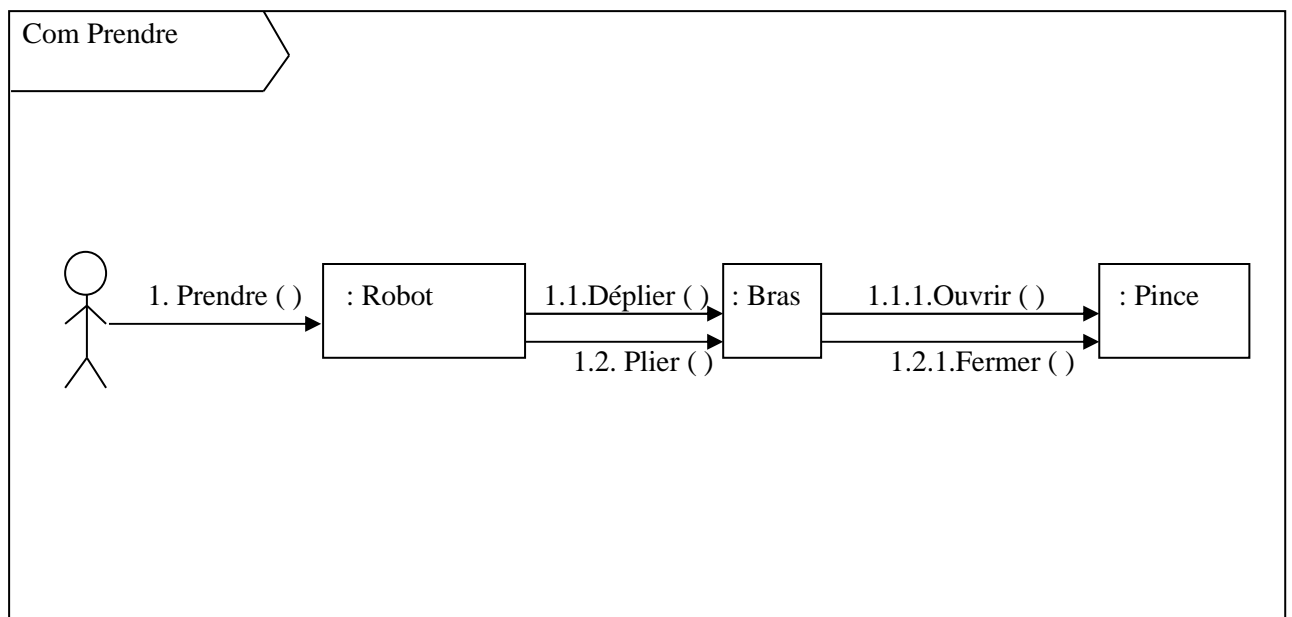
Le diagramme de classe correspondant est :



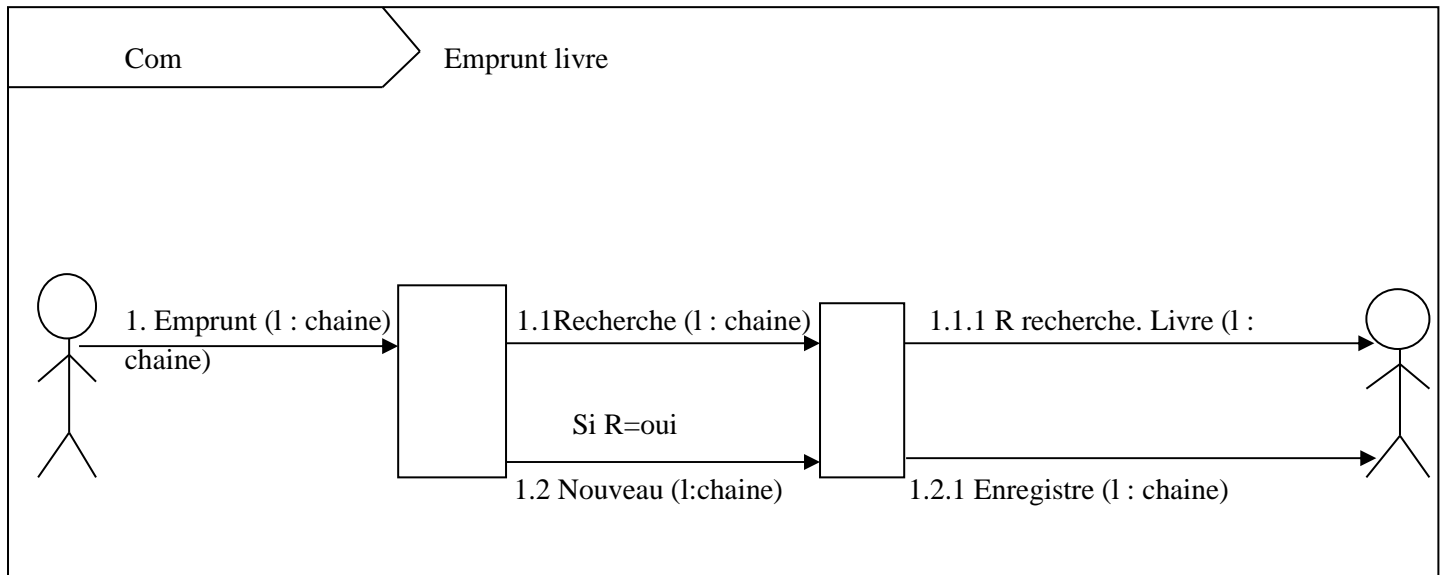
Collaboration



Le diagramme de communication est :



Exemple 2 : Revenons à la gestion de la bibliothèque universitaire et représentons le diagramme de communication correspondant au diagramme de collaboration « Emprunt livre »



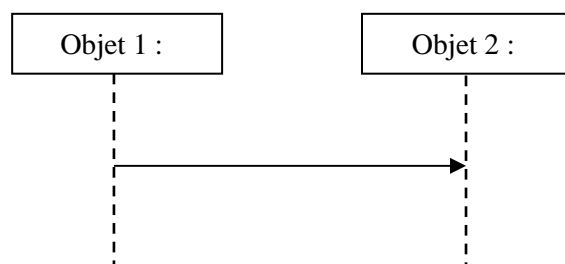
VI-Diagramme de séquence (DS) :

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie présentées dans un ordre chronologique. Ainsi, contrairement au diagramme de communication le temps y est représenté explicitement par une dimension (la dimension verticale) et s'écoule de haut en bas.

Représentation des lignes de vie : une ligne de vie est représenté par un rectangle, auquel est accrochée une ligne verticale en pointillé contenant une étiquette dont la syntaxe est : [< Nom. Rôle >] : [<Nom objet>]

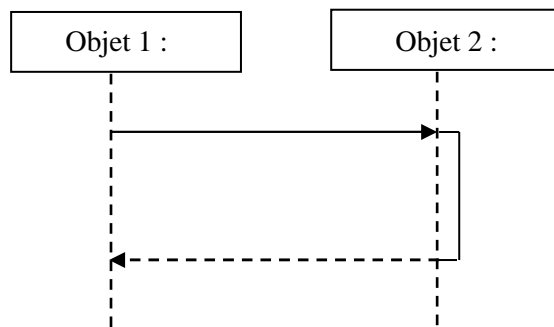
Représentation des messages : nous pouvons avoir deux types de message : les messages synchrones et les messages asynchrones.

Messages asynchrones : les messages asynchrones sont des types de message qui, sont émis et, n'attendent pas (émetteur ou objet) de messages de retour. Les messages les plus courants sont : la création et la destruction des objets et les interruptions. De manière générale ce sont des messages de type signal. Graphiquement, un message asynchrone est représenté par une flèche en trait plein et à l'extrémité ouverte partant de ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible.

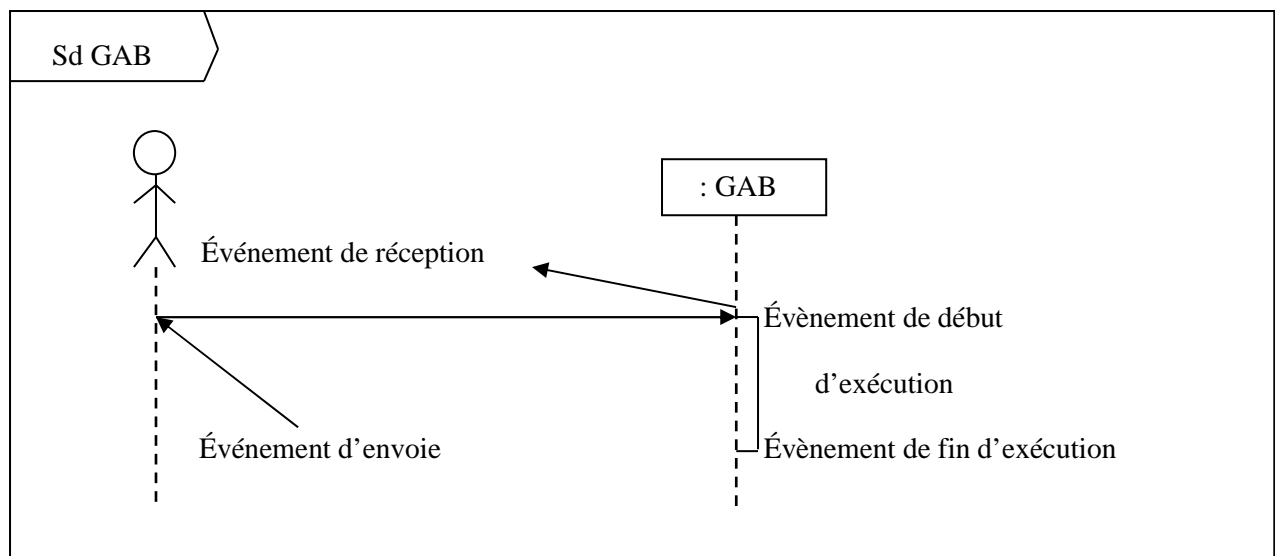


Messages synchrones : un message est dit synchrone lorsque l'émetteur du message est bloqué durant le temps de l'exécution du message. A un message synchrone correspond toujours un message de retour. Ce message retour est implicite en UML (la représentation n'est pas obligatoire mais fortement conseillée). Les types de message les plus courants pour cette catégorie sont les méthodes (opérations). Graphiquement un message synchrone se représente par une flèche en trait plein et à l'extrémité pleine (fumée) partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible.

Ce message peut être suivi d'une réponse qui se représente par une flèche en pointillé.



Événements et messages : UML permet de séparer clairement l'envoi des messages, sa réception, ainsi que le début d'exécution et sa fin. Considérons l'exemple d'un distributeur de billet (GAB)



Syntaxe des messages et des réponses : dans la plupart des cas, la réception d'un message est suivie de l'exécution d'une méthode. Cette méthode peut recevoir des arguments et la syntaxe des messages permet de transmettre ces arguments. La syntaxe de ces messages est la même pour un diagramme de communication excepté :

Les numéros de séquence sont généralement omis puisque l'ordre des messages est déjà matérialisé par l'axe vertical qui représente l'écoulement du temps. La direction du message spécifié par la direction de la flèche qui matérialise le message, et non par une flèche supplémentaire au dessus du connecteur. La syntaxe de réponse à un message est la suivante :

[< Attribut > =] message [: < valeur >]

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

Exemple : Considérons l'exemple précédent traitant de la bibliothèque universitaire et supposons que l'utilisateur veut rechercher le nombre de livres intitulés « Tintin »

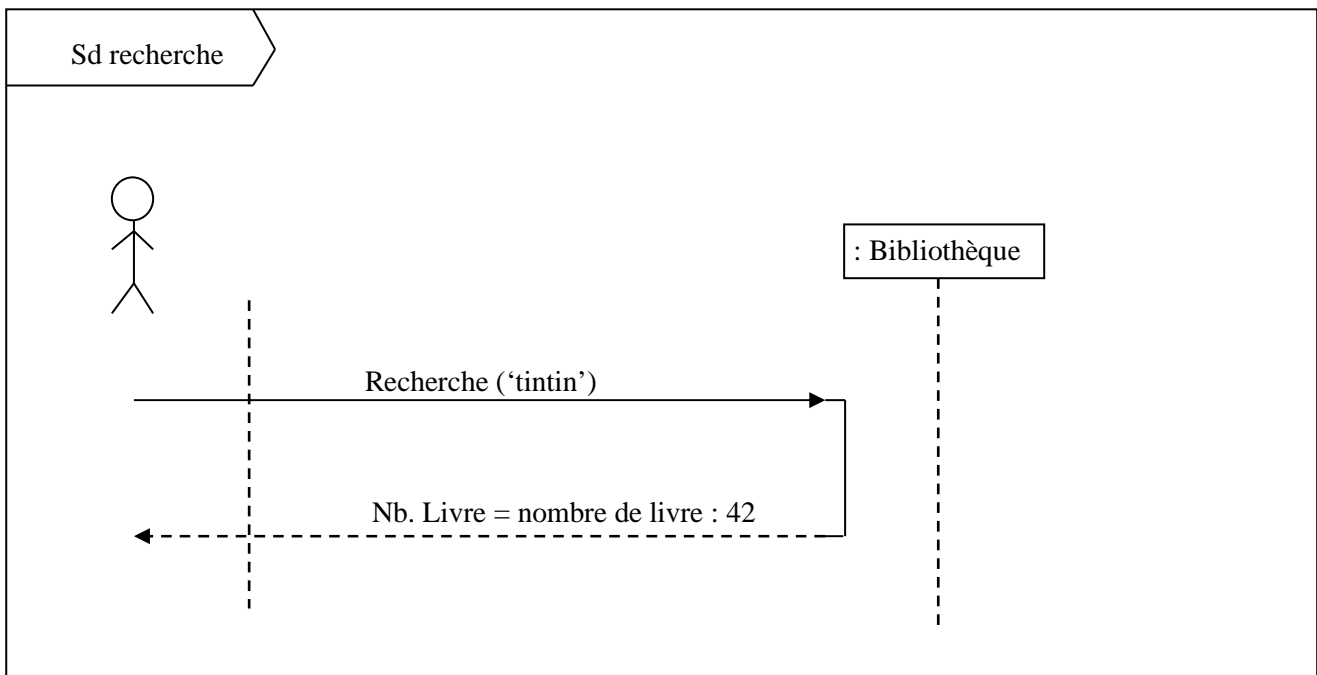
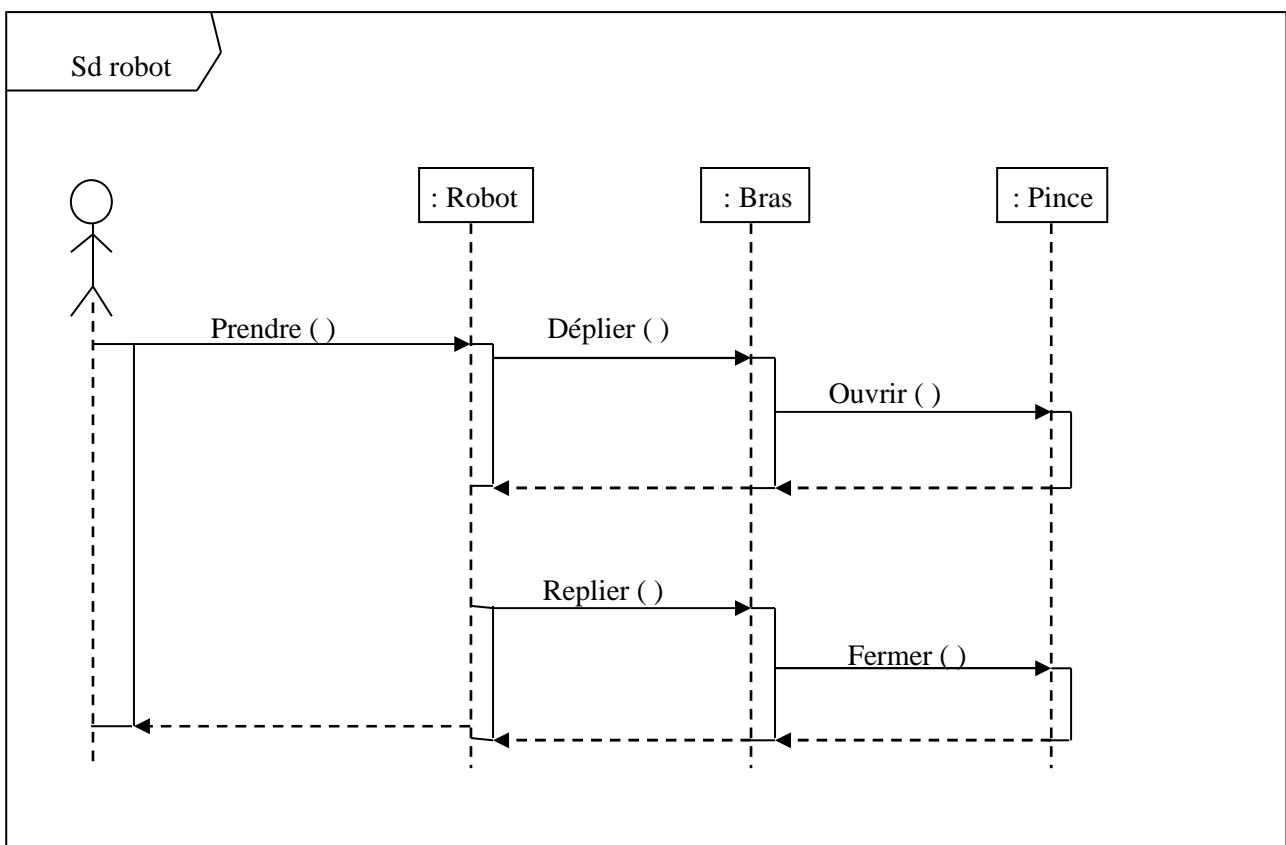


Diagramme de séquence correspondant à l'exemple Robot



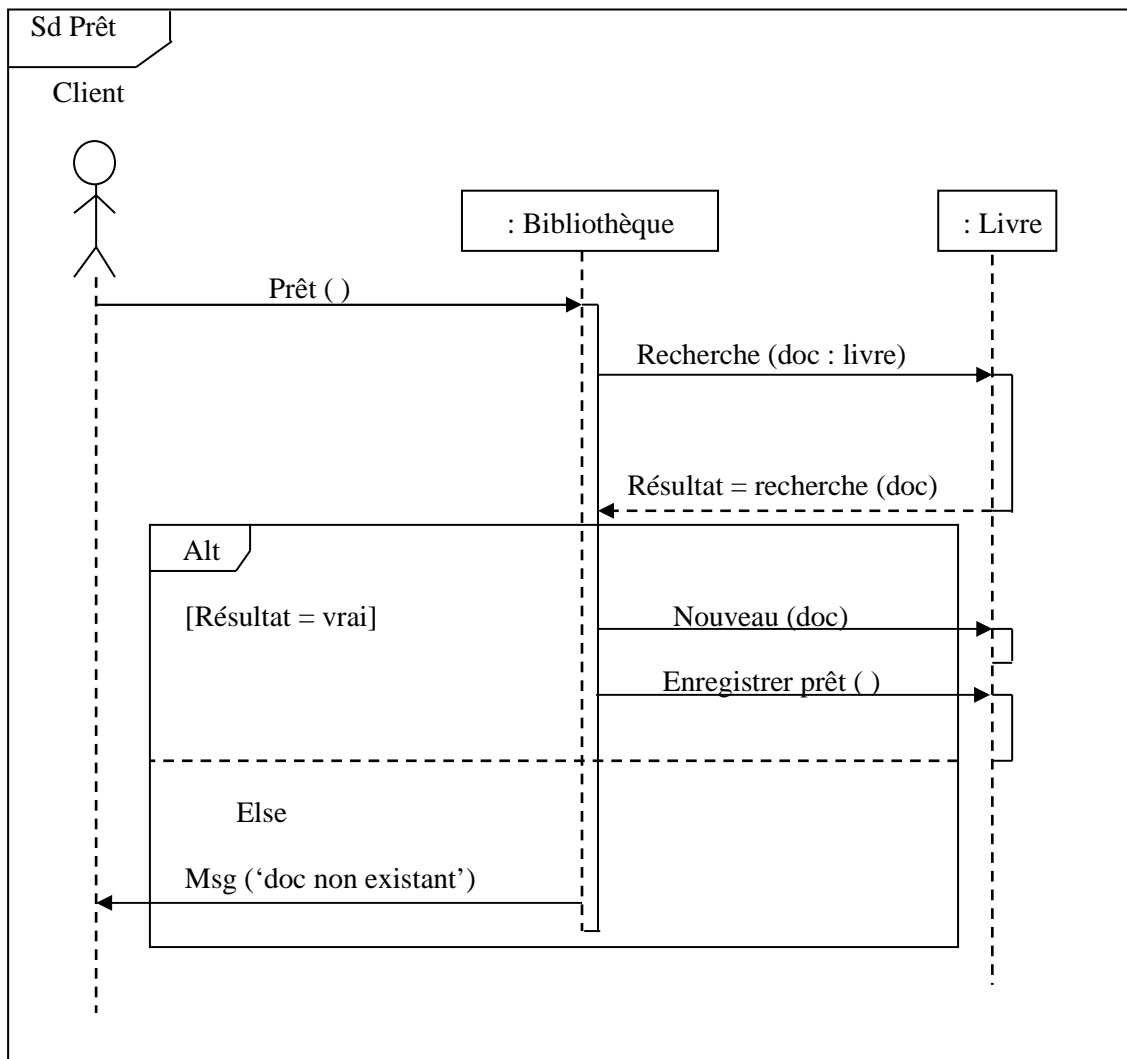
Fragment d'interaction combinée : un fragment combiné représente des articulations d'interaction, il est défini par un opérateur et des opérandes. L'opérateur conditionne la signification du fragment combiné. IL existe 12 types d'opérations définies dans la notation UML 2. Graphiquement un

fragment combiné est représenté par un rectangle. Le coin supérieur gauche contient un pentagone. Dans le rectangle (la partie du pentagone) figure le type de combinaison appelé opérateur d'interaction. Les opérandes d'un opérateur d'interaction sont séparés par un lien en pointillé. Les conditions de choix sont représentées par des expressions booléennes entre crochet.

La liste des opérateurs est la suivante :

- * Alternative (si)
- * Option (cas selon)
- * Break
- * Parallèle (message parallèle)
- * Critical
- * Region
- * Loop (répétition)
- * Ignore
- * Consider
- * Assertion
- * Négative
- * Weak sequencing
- * Strick sequencing

Exemple : représentons le processus de prêt d'un livre dans notre bibliothèque universitaire et supposons que le client recherche son document avant de lancer le prêt.



VII-Diagramme d'état transition (DEI) :

Introduction au formalisme

Les diagrammes d'état transition d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à l'état fini. Ils présentent les séquences possibles d'état et d'action qu'une instance de classe peut prêter au cours de son cycle de vie en réaction à des événements discrets (signaux ou bien l'invocation d'une méthode).

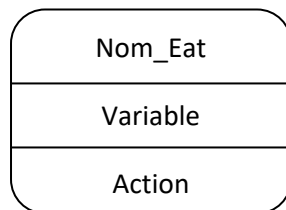
Concrètement un diagramme d'état transition est un graphe qui représente un automate à l'état fini c'est-à-dire une machine dont le comportement de sorti ne dépend pas seulement de l'état de ses entrées mais aussi d'un historique de sollicitation passé.

Etat : c'est un Comportement stable qui prend plus ou moins un temps.

Il représente une période dans la vie d'un objet pendant laquelle ce dernier attend un événement pour accomplir une activité.

Graphiquement un état se représente par un rectangle avec ses quatre coins arrondis. Il peut avoir trois compartiments, le premier représentant le nom de l'état, le deuxième les variables de l'état et le troisième les actions exécutés dans l'état.


Exemple :



L'état initial :

C'est un pseudo (Faux) état qui indique l'état de départ par défaut lorsque le diagramme d'état transition est invoqué. Lorsqu'un objet est créé, il entre dans l'état initial.

Représentation graphique

Un état initial est représenté par .

Etat final : est un pseudo état qui indique que le diagramme d'état transition est terminé.

Représentation graphique

Un état final est représenté par .

L'état final n'est pas obligatoire.

Évènement : un événement est quelque chose qui se produit pendant l'exécution du système et qui mérite d'être modélisée. Un événement se produit à un instant précis et est dépourvu de durée.

Quand un événement est reçu, une transaction peut être déclenchée et faire basculer l'objet dans un nouvel état.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

Les différents types d'évènement sont :

-Evènement de type signal : ce sont des évènements qui sont destinés à véhiculer une communication asynchrone à sens unique entre deux objets. Il n'attend pas que le destinataire traite le signal pour poursuivre son déroulement.

< Nom évènement> ([<paramètre> :<type> [; <paramètre> :<type> - - -]])

-Evènement d'appel (call) : un évènement d'appel représente la réception de l'appel d'une opération par un objet. Les paramètres de l'opération sont ceux de l'évènement d'appel. La syntaxe est la même que pour les évènements de type signal.

-Evènement de type changement (change) : un évènement de type changement est généré par la satisfaction (c'est-à-dire par de faux à vrai) d'une expression booléenne sur des valeurs d'attribut.

Il s'agit d'une manière déclarative d'attendre qu'une condition soit satisfaite.

La syntaxe est la suivante : when (condition booléenne)

-Evènement temporaire (after) : les évènements temporaires sont générés par le passage du temps. Ils spécifient soit de manière absolue (date précise), soit de manière relative aux temps écoulés.

Par défaut le temps commence à s'écouler dès l'entrée dans le temps courant.

La syntaxe est la suivante : after (< durée >)

When (date=<date>)

Transition: une transition définit la réponse d'un objet à un évènement. Elle relie généralement deux états E1 et E2 et indique qu'un objet dans un état E1 peut passer à un état E2 et exécute certaines activités, si un évènement déclencheur se produit et que la condition de garde est vérifiée. La syntaxe est la suivante : [<évènement>][['<garde>']][['<activité>]

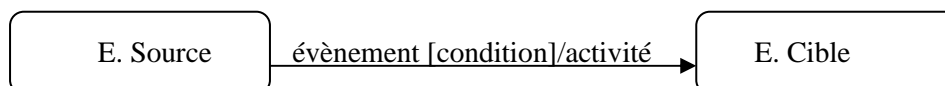
Condition de garde : c'est une expression booléenne généralement un attribut de l'objet.

Activité : c'est l'effet de la transaction, il s'agit généralement d'une activité qui peut être : l'envoi d'un signal, l'appel d'une méthode etc.

Principalement on a deux types de transaction :

-transaction externe : une transaction est externe lorsqu'elle modifie l'état actif. C'est le type de transaction le plus courant.

Représentation



-Transaction interne : contrairement aux transactions externes, les transactions internes ne possèdent pas d'état cible et que l'état actif reste le même à la suite de son déclenchement. La syntaxe est la même. Par contre les transactions internes ne sont pas représentées par des arcs mais plutôt par des compartiments.

Les transactions internes possèdent des noms prédéfinis correspondant à des déclencheurs particuliers :

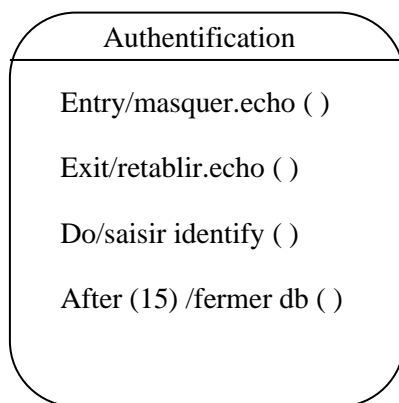
Entry : c'est un évènement qui permet de spécifier une autre activité qui s'exécute quand on entre dans l'état.

Exit : permet de spécifier une activité qui s'accomplit quand on sort de l'état.

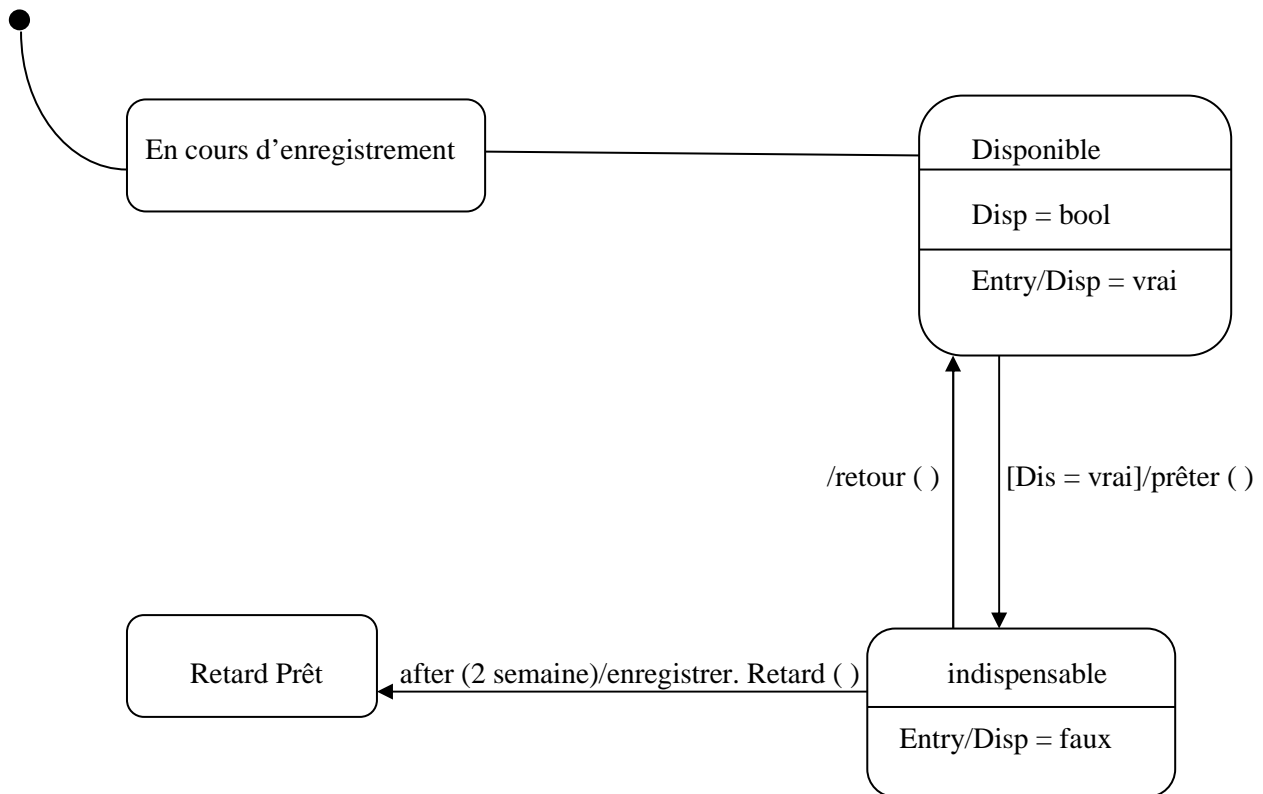
Do : une activité do commence dès que l'activité entry est terminée, lorsque cette activité est terminée, la transition d'achèvement peut être déclenchée après de l'activité exit.

Exemple 1 : Considérons un système d'identification à une base de données. Au lancement de l'identification les caractères seront masqués et à la sortie les caractères seront rétablis.

Après 5 secondes la fenêtre d'authentification disparaît.

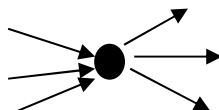


Exemple 2 : Représentons un diagramme d'état transition correspondant à la procédure de prêt de livre dans notre bibliothèque universitaire.

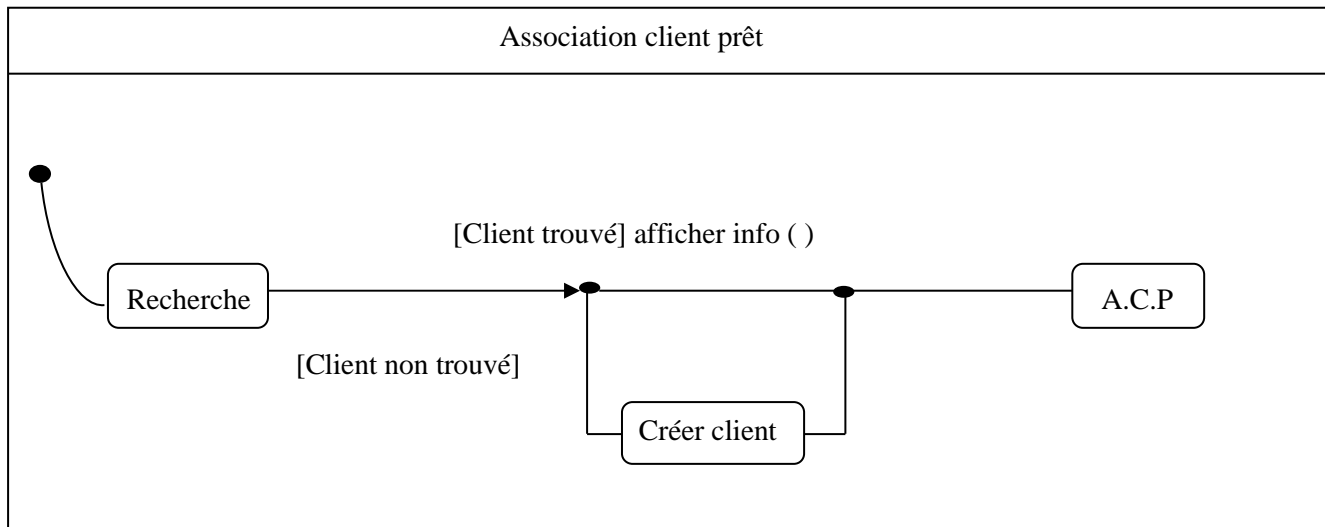


Les points de choix : il est possible de représenter les alternatives pour le franchissement d'une transition. Pour cela on utilise des pseudo-états particuliers qui sont : le point de jonction et le point de décision.

Le point de jonction : Les points de jonction sont des objets graphiques qui permettent de partager des segments de transition. Un point de jonction peut avoir plusieurs segments de transition entrant et plusieurs segments de transition sortant.

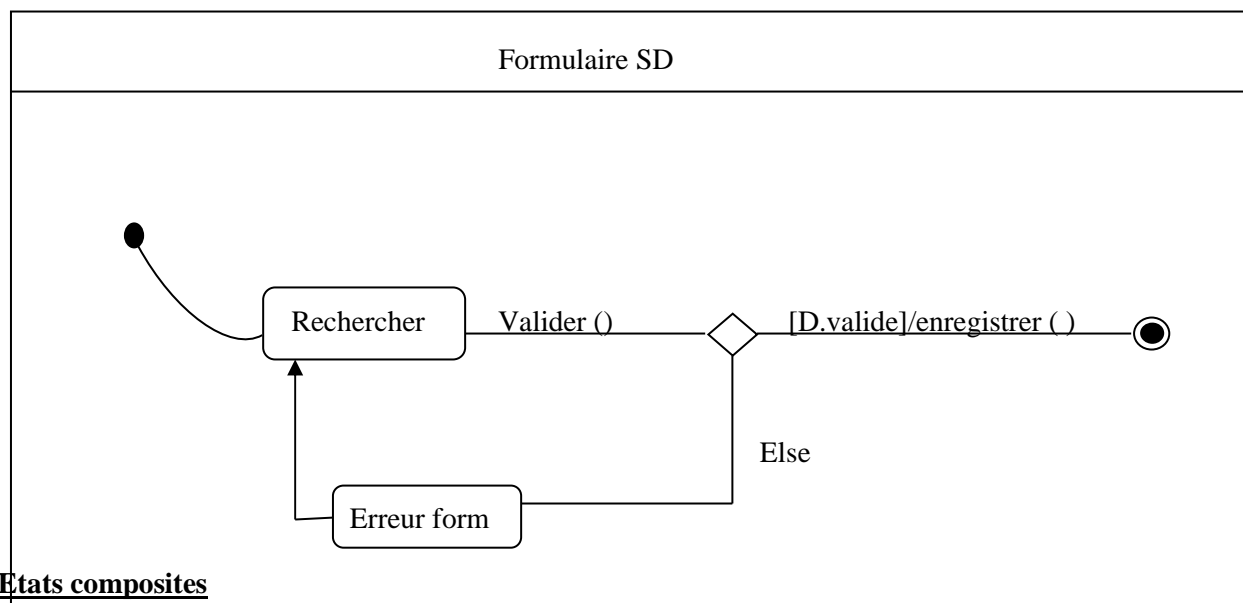


Exemple : dans la gestion de la bibliothèque universitaire le digramme d'état correspondant à l'association d'un client à un prêt se représente comme suit :



Point de décision : un point de décision possède une seule entrée et au moins deux sorties. Contrairement à un point de jonction les gardes situés après le point de décision sont évalués au moment où il est atteint.

Exemple : représentons le fonctionnement d'un formulaire de saisie de données.



Etats composites

Un état composite est un état décomposé dont les compartiments contiennent des sous états.

Etats historiques

Un état historique est un pseudo-état qui nécessite le dernier sous état visité dans un état composite. Une transition ayant pour cible un état historique est équivalente à une transaction qui a pour cible le dernier état visité de l'état englobant.

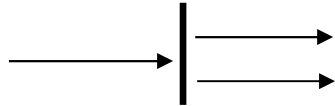
VIII-Diagramme d'activité

Ce sont des diagrammes dynamiques qui servent soit à expliciter (détailler) un cas d'utilisation ou bien à représenter les algorithmes de nos méthodes. Ce sont des outils pour générer du code. En d'autres termes les digrammes d'activités sont une variante des diagrammes d'état transition.

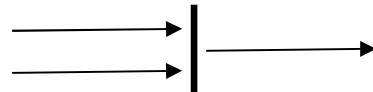
Nous aurons à représenter des activités avec des instructions textuelles et des structures de contrôle (alternative, boucle) avec des instructions graphiques.

A côté des éléments du diagramme d'état transition nous aurons les objets graphiques suivants :

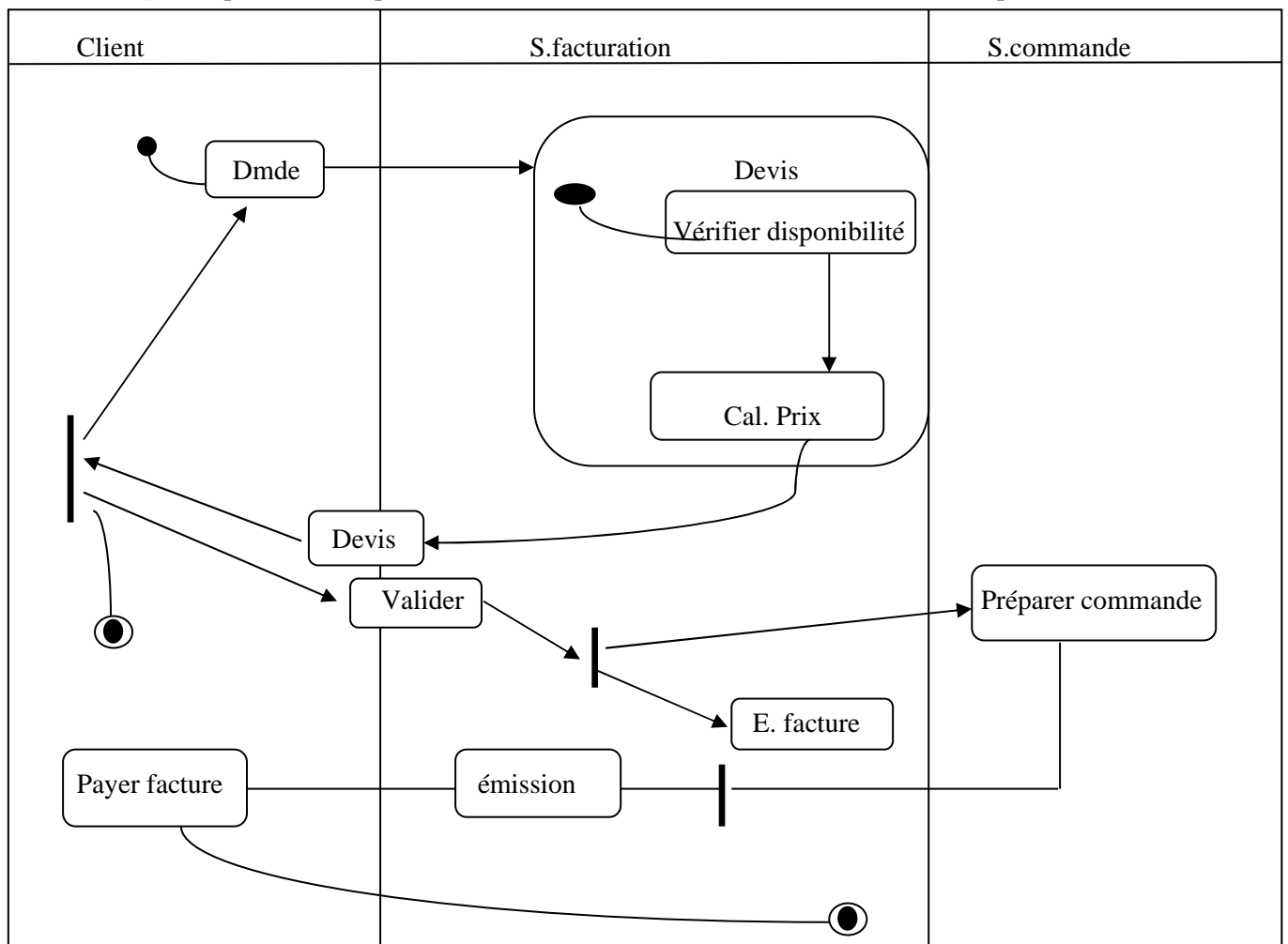
Le fork



Le join



Exemple : représentons le processus de traitement d'une commande dans une entreprise.



TD UML N°1

Diagramme de Cas d'Utilisation

EXO 01 : Projet de Micro Finance.

La Fédération préfectorale des artisans de Labé (FEPAL) vous demande de développer une application pour la gestion des ses institutions de micro finance (IMF). A la réception d'un client on enregistre ses informations personnelles qui sont : nom et Prénom, Domaine d'activité et adresse ; après un compte actionnaire est obligatoirement ouvert pour ce client et éventuellement un compte dépôt a vue, le client peut effectuer des emprunts et rembourser ses emprunts. L'ouverture d'un compte actionnaire passe obligatoirement par l'achat de 5 actions ou plus.

Donner le Diagramme de cas d'utilisation correspondant.

EXO 02 : Agence de Voyage.

Une agence de voyage organise des voyages pour des clients. Lors d'un voyage l'hébergement se fait à l'hôtel. Le client voyage soit en avion, soit en train.

Le client a besoin d'un taxi pour faire le trajet jusqu'à l'hôtel. Il ya des clients qui désirent une facture détaillée.

Donner le diagramme de cas d'utilisation correspondant.

EXO 03 : Consultation d'un train sur Internet

Le client souhaite consulter les horaires d'un train par internet. Le client doit saisir des données qui sont : la gare de départ, la gare d'arriver, la date et l'heure de départ.

Le client peut éventuellement consulter la liste des gares et acheter un billet. Dans ce dernier cas le client doit sélectionner un horaire, un tarif, payer et alors les billets lui seront envoyer par poste.

Donner le diagramme de cas d'utilisation correspondant.

EXO 04 : *Projet de gestion d'une industrie de textile.*

En vue de la mise en place d'un logiciel dédié à l'industrie textile, nous étudions tout d'abord quelques fonctionnalités requises. Ce logiciel s'adresse à différentes catégories d'utilisateurs, et doit servir principalement à recueillir l'information sur les produits développés dans l'entreprise.

Tous les personnels de l'entreprise peuvent consulter le système. Toute consultation doit être précédée d'une authentification légère dans laquelle la personne précise son Nom et son service à des fins de statistiques ultérieures.

Les ingénieurs peuvent effectuer différentes opérations de gestion relatives aux produits dont ils sont responsables : ajout, retrait, modification. Ces opérations doivent être précédées d'une authentification plus appondis lors de laquelle l'ingénieur précise son Nom, son service et donne un mot de passe qui est vérifié en contactant le système de gestion des personnels.

Toutes les opérations (consultation et gestion) donnent lieu à un enregistrement dans le journal des accès (trace de la session).

Donner un diagramme de cas d'utilisation correspondant.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

TD UML N°II

Diagramme de classe

EXO : 01 Les chaînes proposent des émissions.

Une émission peut être l'objet de plusieurs diffusions. Une émission a une certaine durée. Une diffusion a une certaine date/heure de début.

Une plage horaire est relative à un certain jour de la semaine, elle commence et se termine à des heures rondes. Une session est l'utilisation du système par un utilisateur, d'une certaine date/heure de début à une certaine date/heure de fin, au cours de laquelle il peut zapper d'une chaîne à une autre.

Une session n'est autorisée que si l'utilisateur qui la déclenche est un utilisateur autorisé pour le type d'émission, correspondant à la diffusion regardée, Si l'heure de début de session appartient à une plage horaire autorisée pour cet utilisateur, et Si le crédit hebdomadaire de cet utilisateur n'est pas atteint.

Proposer le diagramme de classe.

EXO : 02 Une entreprise de transports en commun.

Une entreprise de transports en commun souhaite informatiser la gestion de ses voyages organisés en car. Les passagers sont caractérisés par un numéro de client, leur nom, leur prénom, leur adresse et leur numéro de téléphone. On considèrera l'adresse comme un attribut simple (indivisible). Chaque passager peut effectuer plusieurs voyages (à des périodes différentes). Un voyage est constitué d'une ou plusieurs destinations. Une destination est caractérisée par un code voyage, une date de départ et une durée. Un voyage est constitué d'une ou plusieurs destinations. Une destination est caractérisée par un code destination et un nom de ville. Pour chaque couple (voyage, destination), on souhaite mémoriser la durée du séjour. A chaque voyage est également associé un car. Un car est caractérisé par un numéro d'immatriculation, sa marque, son modèle et le nombre de places assises. A un voyage sont finalement associés un ou deux chauffeurs (selon la durée du circuit). Un chauffeur est caractérisé par son numéro de sécurité sociale, son nom, son prénom et son ancienneté dans la société.

Donner le Diagramme de classe.

EXO : 03 Un système de gestion du personnel

Nous étudions à présent une partie du système de gestion du personnel. Tout membre du personnel possède un nom, un numéro de sécurité sociale, un login et un mot de passe. Il est affecté à un service dont on connaît le nom et le numéro. Les ingénieurs ont de plus une spécialité (mécanique, informatique, ect..).

Donner un diagramme de classe

EXO : 04 les Fibres

Les fibres constituent la matière première utilisée dans les textiles. Elles sont décrites par un nom générique (Coton, laine, amiante, polyester). Une fibre peut être :

- **Végétale**, et se caractérise alors par la plante de provenance (par exemple coton, lin, sial, chanvre, coco) et la partie de la plante utilisée (par exemple poils séminaux des graines, tiges, feuilles, troncs, enveloppes des fruits) ;
- **Animale**, on connaît dans ce cas l'animal (par exemple le mouton ou le ver à soie) et la source de la fibre (le poil ou la bave) ;
- **Minérale**, comme l'amiante ; elle est alors décrite par la roche d'origine ;
- **Synthétique**, comme l'acrylique, l'élasthanne ou le polyester ; elle est dans ce cas décrite par une formule chimique, le nom de leur inventeur et la date de dépôt du brevet.

Indépendamment de leur nature, les fibres peuvent être obtenues selon un procédé naturel ou artificiel. Ainsi certains procédés chimiques permettent de transformer le bois ou les algues pour obtenir des fibres artificielles.

Donner le diagramme de classe correspondant.

TD N° III**Diagramme d'Etat transition :****EXO 01 :** (Cycle de vie d'une personne)

A sa naissance une personne est considérée comme un bébé, après 15 ans il devient un ado puis à 25 ans il est adulte. Au statu adulte il commence par être célibataire lorsqu'il se mari, il passe au statu marié et quand il divorce il devient célibataire. Lorsqu'il perd sa femme il devient veuf et quand il se remarie il devient à nouveau marié. Son cycle de vie se termine à sa mort.

TAF : Modéliser le cycle de vie de la personne à travers un diagramme d'état transition.

Exo 02 et 03 : Diagramme de Séquence / Communication

On s'intéresse à la gestion des emprunts. Si un client émet une demande de prêt, on vérifie la liste de ses derniers prêts, s'ils sont amortis on l'accorde un nouveau prêt ou dans le cas échéant un message lui est renvoyé pour signifier qu'il n'est pas autorisé à emprunter de l'argent.

TAF : élaborer le diagramme de séquence et communication.

EXO 04 : diagramme d'état transition

Donner le diagramme d'état transition pour un client sachant qu'à sa création il est actionnaire, quant il emprunt de l'argent il devient redevable et quant il rembourse les prêt il devient solvable.

EXO 05 : ascenseur

Diagramme de séquence

Un produit va être installé pour contrôler **N** ascenseurs dans un gratte-ciel de **M** étages. Notre problème concerne la logique nécessaire au déplacement des ascenseurs entre les étages en accord avec les contraintes suivantes:

Chaque ascenseur possède un ensemble de **M** boutons, un pour chaque étage. Un bouton s'allume lorsqu'il est appuyé et provoque le déplacement de l'ascenseur vers l'étage correspondant.

Chaque étage, à l'exception du premier et du dernier, possède deux boutons, un pour demander la montée et un pour demander la descente. Ces boutons s'allument lorsqu'ils sont appuyés. Ils s'éteignent quand l'ascenseur arrive à l'étage, et celui ci se déplace ensuite dans la direction demandée. Quand un ascenseur n'est pas requis, il reste à l'étage où il se trouve et ferme ses portes.

Décrire à l'aide d'un diagramme de séquence chacun des scénarios suivants:

Requête d'ascenseur depuis l'étage

Requête d'étage depuis l'ascenseur

III- : Conclusion sur la méthode UML

Jusque là, les concepts utilisés pour modéliser les objets du domaine étaient

différents de ceux

utilisés pour spécifier les objets du système, les méthodes ne pouvaient pas grand chose pour rationaliser la démarche.

La généralisation de l'approche objet à l'ensemble du processus de développement a rendu possible

l'unification du cadre conceptuel et UML a permis de supprimer la rupture méthodologique entre l'analyse et la conception.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;

Tel : 622 29 43 67 / 666 97 25 67.

Cette approche permet de passer du modèle au système de manière lisible et argumentée. Lisible, car, pour faire évoluer le système, il faut pouvoir faire le lien entre les objets du domaine et les objets du système ; Argumentée car, les choix techniques sont complexes, interdépendants et doivent être périodiquement révisés pour tenir compte de l'évolution des environnements.

Si UML s'est aussi rapidement imposé à tous, c'est parce qu'il ne s'agit que d'un langage, dont l'efficacité n'est liée ni à la nature du problème ni à une démarche méthodologique particulière. Pour autant, la mise en œuvre d'UML suppose une définition des tâches et des procédures adaptées au contexte, et en particulier de préciser :

La nature des problèmes

Les technologies et outils de développement utilisés

La culture méthodologique des participants, l'expérience des problèmes, les technologies des outils

La dimension de l'entreprise et modes d'organisation

UML est aujourd'hui un standard incontournable. Les raisons de son succès sont multiples :

Il est le résultat d'un large consensus (industriels, méthodologistes...).

Il couvre toutes les phases d'un cycle de développement.

Les outils qui supportent UML se multiplient (GDPro, ObjectTeam, Objecteering, OpenTool, Rational Rose, Rhapsody, STP, Visio, Visual Modeler, WithClass...).