Sommaire

I. INTRODUCTION AU LANGAGE UML 2

1. Le système d'information 2

2. Méthodologies de conception 2

II. LE DIAGRAMME DE CAS D'UTILISATION 5

1. Présentation des cas d'utilisation (cu) 5

2. Eléments constituants un diagramme de cu 6

3. Elaboration du diagramme de cu 8

III. LE DIAGRAMME DE CLASSE 9

1. Présentation des classes et des objets 10

2. Eléments constituants un diagramme de classe 10

3. Elaboration du diagramme de classe 13

IV. LE DIAGRAMME DE SEQUENCE 14

1. Introduction 14

2. Eléments constituants un diagramme de séquence 14

V. LE DIAGRAMME D'ACTIVITÉ 19

1. Introduction 19

2. Eléments constituants un diagramme d'activité 19

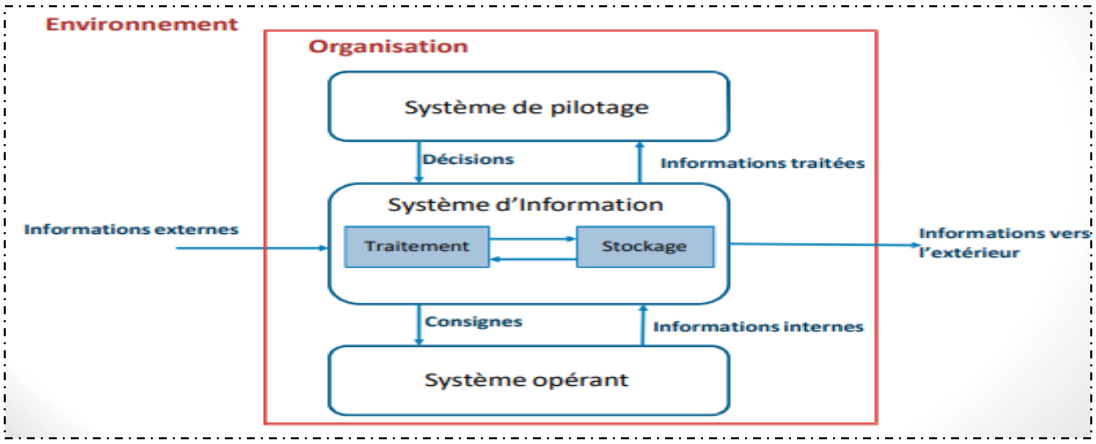
I.INTRODUCTION AU LANGAGE UML

1. Le système d'information

a. Définition

Un Système d'Informations (SI) représente l'ensemble des éléments (personnel, matériel, logiciel, données et procédures) participant au stockage, classification, traitement, et diffusion de l'information au sein d'une organisation.

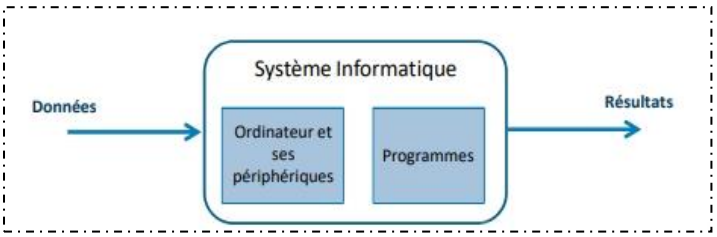
Place du SI dans une organisation :



b. Le système informatique

L'informatique est la science qui étudie le traitement automatique de l'information par un ordinateur.

Le système informatique est un environnement matériel et logiciel pour traiter automatiquement l'information



Le Système Informatique est un moyen pour mettre en œuvre le Système d'Information.

2. Méthodologies de conception

a. Modèle

Un modèle est une représentation d'une situation dans la réalité.

Dans une situation donnée, chaque personne peut avoir une compréhension différente de la situation réelle.

b. Modélisation

La modélisation avec un ensemble de règles et de notations permet de partager la compréhension et ainsi favorise la communication.

Modéliser un système avant sa réalisation permet de comprendre son fonctionnement mais on peut modéliser un système existant.

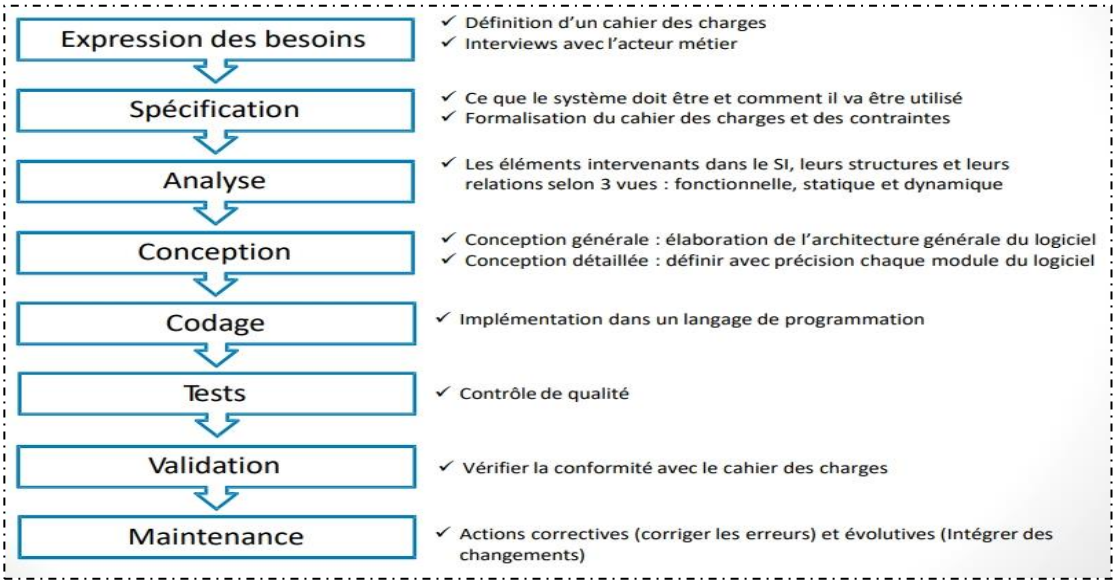
c. Conception

La conception est la construction de la solution à un problème donné.

La conception peut englober la modélisation ou pas: Sans modélisation le système conçu ne sera pas bien compris.

La conception consiste à répondre à la question « comment faire le système ? »

d. Cycle de vie du logiciel



3. Le langage UML:

a. Introduction

Pour programmer une application, il ne convient pas de se lancer tête baissée dans l'écriture du code : il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle modélisation ; son produit est un modèle.

UML UNIFIED MODELING LANGUAGE :

Proposer une méthode unifiée revient à proposer une autre méthode

Il est plus intéressant d'avoir un langage unifié qui sera associé à n'importe qu'elle démarche.

UML peut être associé à toute démarche de conception :

À n'importe quelle étape de la démarche

Avec différents environnements de développement.

L'Unified Modeling Language (UML) est un langage standardisé utilisé pour la modélisation et la conception des systèmes d'information. Il permet de représenter visuellement la structure et le comportement d'un logiciel à l'aide de divers diagrammes (cas d'utilisation, classes, séquence, etc.). UML facilite la communication entre les développeurs, analystes et parties prenantes en offrant une vision claire et structurée du système à concevoir.

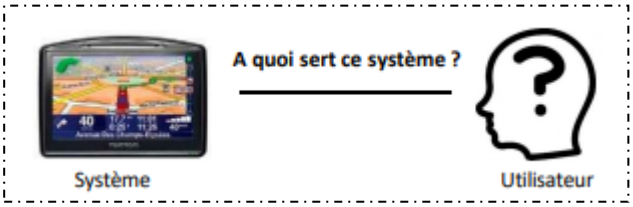
b. Diagrammes d'UML

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 répartissent en deux grands groupes :

Diagrammes structurels ou diagrammes statiques	Diagrammes comportementaux ou diagrammes dynamiques
- diagramme de classes	- diagramme de cas d'utilisation
- diagramme d'objets	- diagramme d'activités
- diagramme de composants	- diagramme d'états-transitions
- diagramme de déploiement	- Diagrammes d'interaction
- diagramme de paquetages	- diagramme de séquence
- diagramme de structures composites	- diagramme de communication
	- diagramme global d'interaction
	- diagramme de temps

II.LE DIAGRAMME DE CAS D'UTILISATION

1. Présentation des cas d'utilisation (cu)



- ✓ Un système est destiné à être utilisé par des utilisateurs (humains ou machines)
- ✓ Le point de vue des utilisateurs est important pour comprendre :
  - Le fonctionnement du système
  - Les services rendus par le système
- ✓ UML permet de décrire d'un point de vue utilisateur, le système étudié (existant ou futur) par un diagramme spécial : le diagramme de Cas d'Utilisation.
- ✓ Les fonctionnalités du système peuvent être exprimés sous forme de phrases simples, tels que :

- Retirer d'argent
- Acheter recharge

La modélisation des fonctions du système en utilisant des notations graphiques permet de mieux comprendre le système d'un point de vue fonctionnel et favorise la communication.

En UML, un Cas d'Utilisation modélise un service (ou une fonctionnalité) rendu par le système.

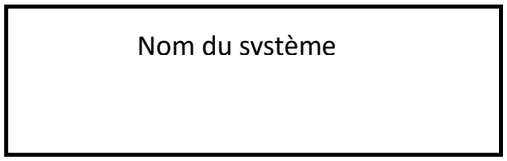
2. Eléments constituants un diagramme de cu

a. Système

Le système est représenté sous forme d'un rectangle qui contient un libellé qui correspond au nom du système à modéliser

Le système sert à délimiter le système par rapport à son environnement et aux autres systèmes.

Le système est un ensemble de CU. De cette manière, UML modélise le système et les services rendus par le système.



b. Cas d'Utilisation

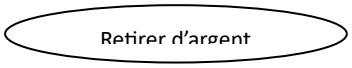
Un Cas d'Utilisation représente une fonctionnalité fournie par un système.

Graphiquement, un CU est représenté sous forme d'une ellipse.

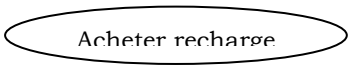
La fonction réalisée par le CU est représentée sous forme d'un verbe à l'infinitif.

Les fonctionnalités du système exprimées textuellement seront modélisés à l'aide des cas d'utilisations.

Retirer d'argent :



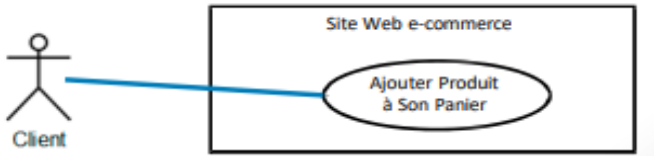
Retirer d'argent :



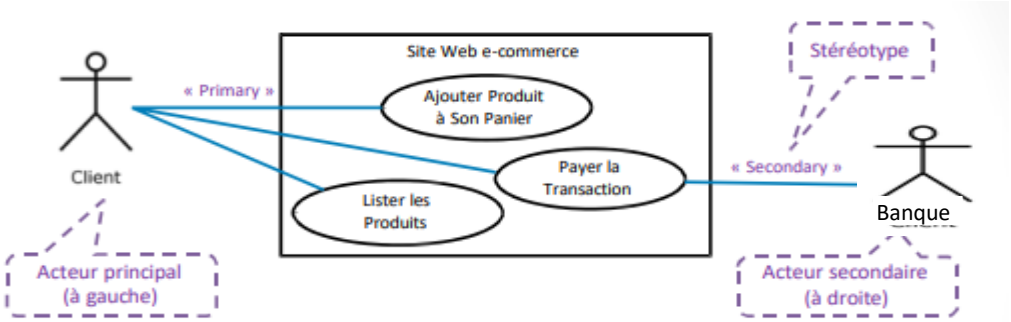
c. Acteur

Un acteur est un ensemble cohérent de rôles joués par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent avec le système.

- ✓ Les acteurs sont représentés par un stick man :
- ✓ Il ne faut pas confondre la notion d'acteur avec la notion d'utilisateur.
- – L'acteur décrit le rôle qu'un utilisateur joue par rapport au système.
- – Un utilisateur c'est une personne utilisant le système.
- ✓ La relation d'association entre acteur et CU représente la possibilité pour l'acteur de déclencher le cas



Acteur principal et secondaire



L'acteur principal obtient un résultat observable du système. Il déclenche le CU par son initiation de la communication. Il utilise le système comme outil pour réaliser son but.

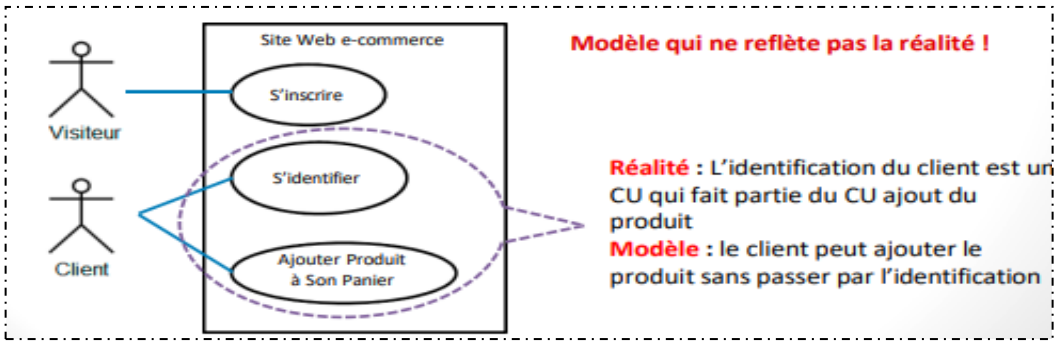
L'acteur secondaire ou auxiliaire est sollicité pour des informations complémentaires. Il intervient suite à l'intervention de l'acteur primaire. Il offre généralement ses services au système.

d. Relations :

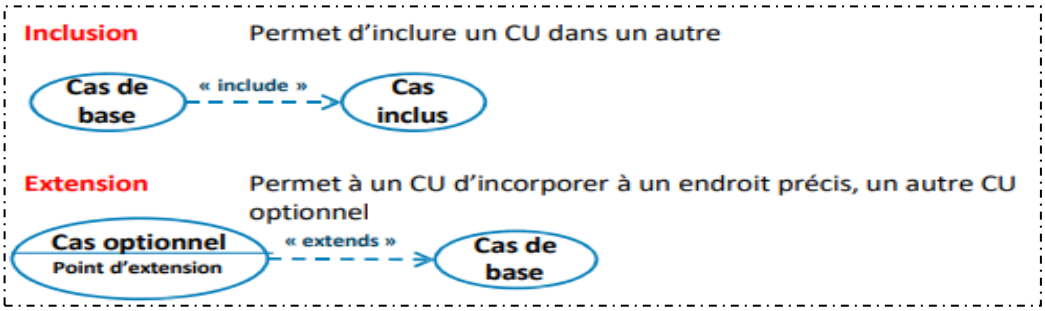
Les relations entre CU

Le visiteur qui veut gérer son panier (entre autres l'ajout du produit) doit d'abord devenir un client, en remplissant un formulaire d'inscription

Après l'inscription, le visiteur devient un client doté de paramètres d'identifications (login et mot de passe).

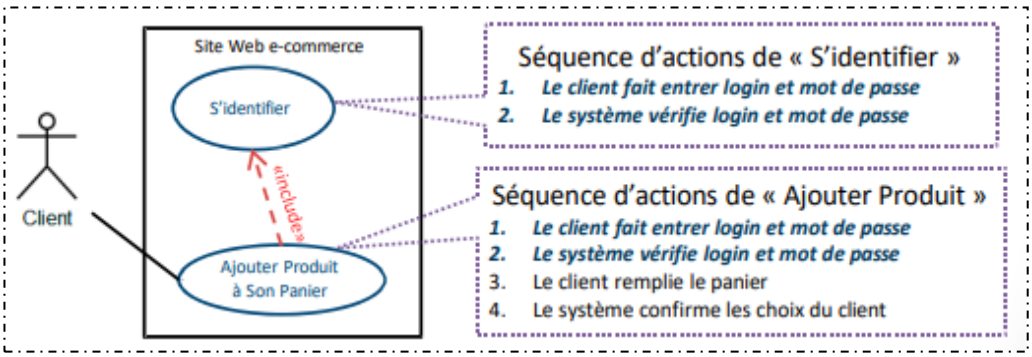


UML propose deux types de relations entre les cas d'utilisation :



- **Inclusion**

Pour vérifier qu'un cas est inclus dans un autre ou pas, il faut vérifier que les actions et les interactions réalisés entre l'acteur et le système dans le cas inclus sont dupliqué entièrement dans le cas de base.

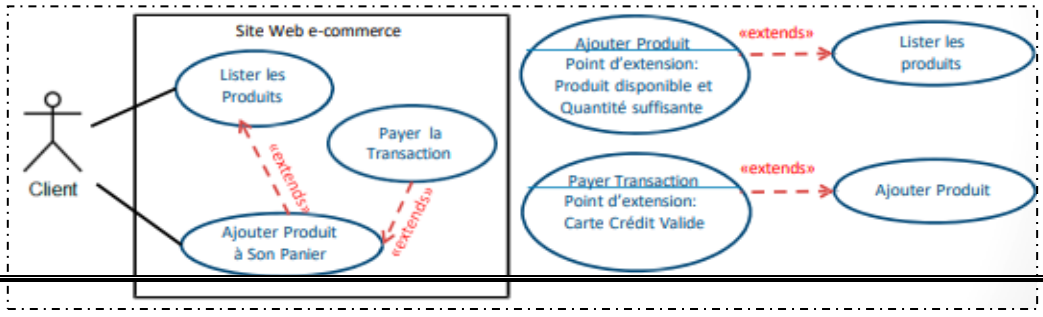


L'inclusion permet d'enlever la redondance (factorisation) dans la description textuelle et de mieux se concentrer sur les autres fonctionnalités.

- **Extension**

En listant les produits, le client peut décider d'ajouter un produit à son panier.

Après avoir ajouté un ou plusieurs produits à son panier, le client peut décider de payer ses achats en ligne en réalisant la transaction nécessaire



L'extension permet de séparer le comportement optionnel ou rare du comportement obligatoire.

Avec l'extension, les CU s'exécutent indépendamment les uns des autres.

3. Elaboration du diagramme de cu

La construction d'un diagramme de cas d'utilisation en UML suit une démarche progressive qui permet d'identifier et de modéliser les interactions entre les acteurs et le système. Voici les étapes détaillées :

1. Identifier les acteurs

- ✓ Un acteur représente un utilisateur ou un autre système interagissant avec l'application.
- ✓ Il peut être un utilisateur humain (client, employé) ou un système externe (base de données, service web).
- ✓ On distingue les **acteurs principaux** (qui initient des actions) et les **acteurs secondaires** (qui participent mais ne déclenchent pas d'actions principales).

2. Identification des cas d'utilisation pour chaque acteur

- ✓ Un cas d'utilisation (CU) représente une fonctionnalité du système répondant aux besoins d'un acteur.
- ✓ Pour chaque acteur, on liste les actions qu'il peut effectuer avec le système.
- ✓ On veille à utiliser des noms clairs et significatifs, souvent sous forme de verbe + complément (ex. : "Passer une commande", "Consulter un solde").

3. Réaliser le diagramme de cas d'utilisation préliminaire

- ✓ On représente les acteurs sous forme de stickmans (bonhommes) et les cas d'utilisation par des ovales.
- ✓ Les relations entre les acteurs et les CU sont illustrées par des lignes.
- ✓ À ce stade, le diagramme reste simple et brut, sans optimisation.

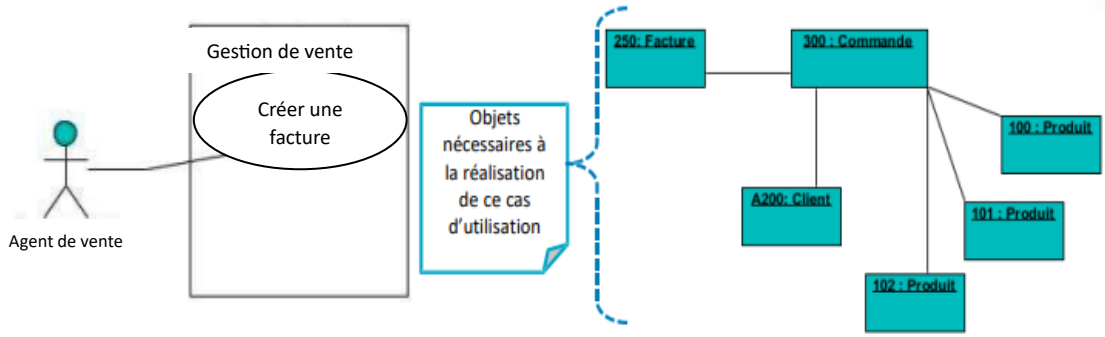
4. Améliorer en éliminant la redondance

- ✓ On affine le diagramme en identifiant les similitudes entre cas d'utilisation.
- ✓ On applique les relations UML :
  - o **« include »** pour les fonctionnalités communes (ex. : "Authentification" incluse dans "Commander un produit").
  - o **« extend »** pour les variantes optionnelles (ex. : "Ajouter un commentaire" peut étendre "Passer une commande").
- ✓ On s'assure que le diagramme est clair, cohérent et non redondant.

III.LE DIAGRAMME DE CLASSE

1. Présentation des classes et des objets

La vue logique a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments.



- ✓ Le diagramme de Classe permet de fournir une représentation abstraite de ces objets.
- ✓ Le diagramme de Classe montre la structure interne du système.
- ✓ En phase d'analyse, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs.
- ✓ En phase de conception, il représente la structure d'un code orienté objet, ou à un niveau de détail plus important, les modules du langage de développement.

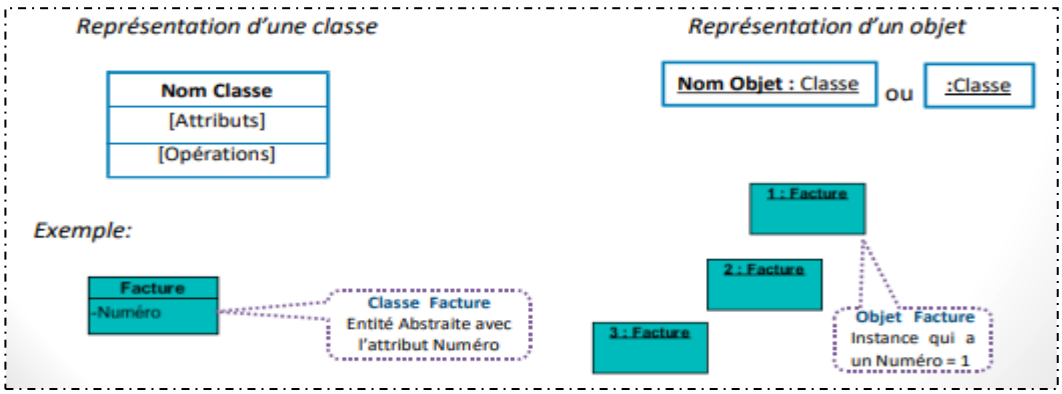
Classe :

Une classe représente la description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques. On parle également de type.

Un objet :

Un objet est une entité possédant une identité et encapsulant un état et un comportement. Un objet est une instance d'une classe. ❑ La classe est le modèle, l'objet est sa réalisation.

Représentation UML :



2. Eléments constituants un diagramme de classe

a. Classe

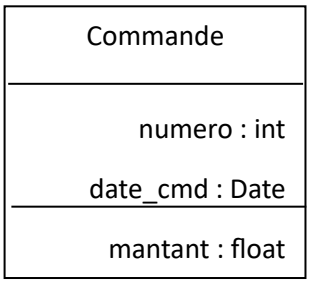
La Classe est un concept Abstrait qui permet de représenter toutes les entités d'un système.

Elle est définie par :

Un nom, ses attributs et ses opérations comme suit :

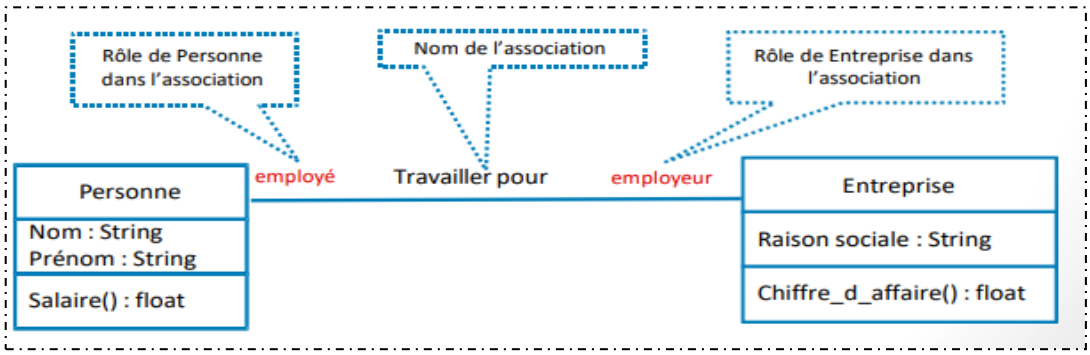
Un attribut est une propriété de classe intéressante pour l'analyste. Les attributs correspondent à des variables associées aux objets de la classe.

Une opération est un service rendu par la classe. Les opérations correspondant à des fonctions associées aux objets de la classe.



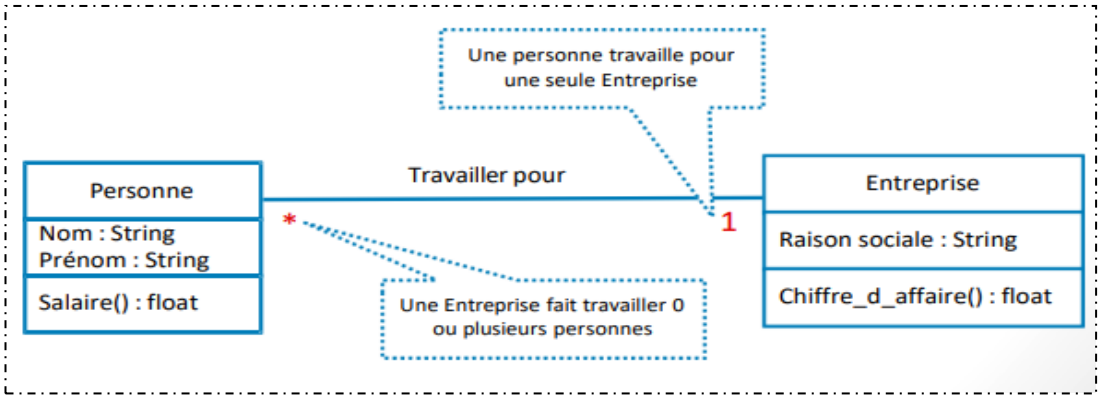
b. Associations

- ✓ Une association est une relation statique entre plusieurs classes.
- ✓ Elle représente une relation possible entre les objets des classes.
- ✓ On peut donner à une classe un rôle dans une association.
- ✓ Un rôle précise la signification de l'entité à proximité du rôle dans l'association.



c. Multiplicité

- ✓ La multiplicité spécifie l'ensemble des cardinalités possibles (parmi les entiers naturels) sur un rôle.
- ✓ Les multiplicités permettent de contraindre le nombre d'objets intervenant dans les instanciations des associations. On en place de chaque côté des associations.
- ✓ Une multiplicité d'un côté spécifie combien d'objets de la classe du côté considéré sont associés à un objet donné de la classe de l'autre côté

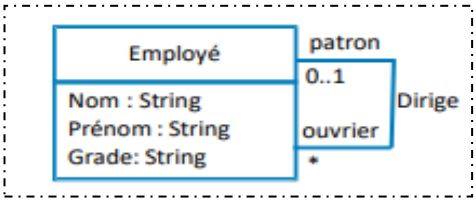


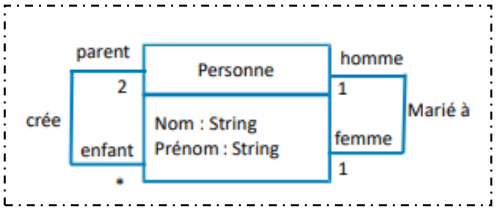
On peut préciser :

- ✓ **0 . . 1** ➔ 0 ou 1
- ✓ **0. . *** ➔ au moins 1
- ✓ **0 ..* ou *** ➔

d. Association réflexive

C'est une association entre une classe donnée et elle-même.

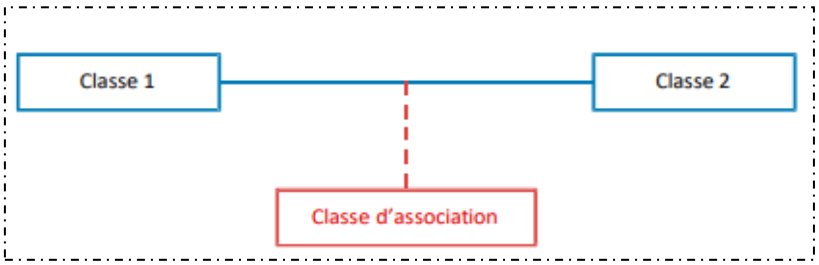




e. Classe d'Association

✓ Une association peut parfois avoir des propriétés. Dans ce cas, ces propriétés sont encapsulées dans une classe nommée classe d'association.

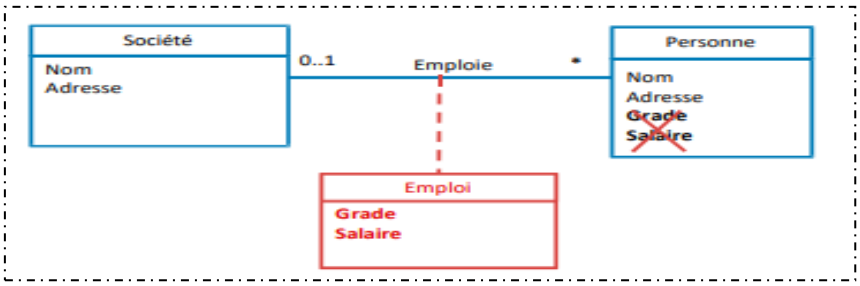
✓ Une classe d'association est représentée par un trait discontinu qui relie la classe avec l'association qu'elle caractérise



Généralement, une association de cardinalité plusieurs à plusieurs est une classe d'association.

Exemple :

On veut caractériser chaque personne travaillant dans une société par ses grades et ses salaires.



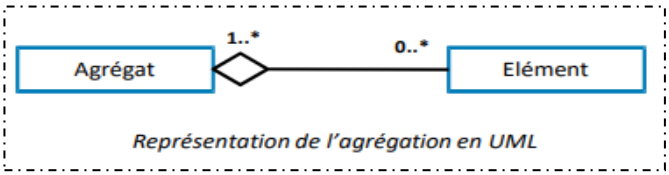
La société emploie plusieurs personnes avec des grades et des salaires différents.

Une personne peut avoir plusieurs grades et salaires dans la société.

Les attributs grade et salaire n'appartiennent ni à la classe Société ni à la classe Personne.

f. Relation d'agrégation

- ✓ C'est un cas particulier d'une association qui exprime la contenance
- ✓ N'a pas besoin d'être nommée (elle signifie « contient », « est composé de
- ✓ Un élément peut appartenir à plusieurs agrégats(agrégation partagée)
- ✓ La destruction de l'agrégat n'entraine pas la destruction de tous ses éléments.

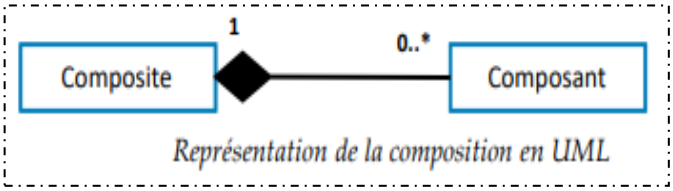


Représentation de l'agrégation en UML

- ✓ Une agrégation peut exprimer :
- – Qu'une classe « élément » fait partie d'une autre « agrégat »
- – Un changement d'état d'une classe entraine un changement d'état d'une autre.
- – Une action sur une classe entraine une action sur une autre.

g. Relation de composition

- ✓ C'est une relation qui exprime une agrégation plus forte.
- – Un élément ne peut appartenir qu'à un seul agrégat composite (agrégation non partagée)
- – La destruction de l'agrégat composite entraine la destruction de tous ses éléments. (le composite est responsable du cycle de vie des composants).



Représentation de la composition en UML

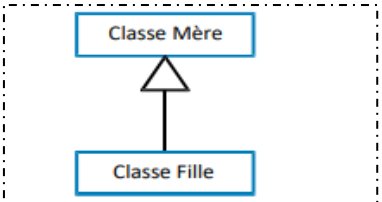
- ✓ Les objets composants sont des instances de la classe composite.

h. Relation de généralisation

L'association de généralisation définit une relation de classification entre une classe plus générale et une classe plus spécifique.

La classe spécifique contient par héritage tous les attributs, les opérations et les de la classe générale et peut en contenir d'autres.

- ✓ Une hiérarchie d'héritage ne doit pas contenir de cycle.
- ✓ Non réflexive : une classe ne peut pas dériver d'elle-même.



Représentation de l'héritage en UML

- ✓ Non symétrique : si B dérive de A alors A ne peut pas dériver de B.

3. Elaboration du diagramme de classe

L'élaboration d'un diagramme de classes suit une démarche méthodique en plusieurs étapes :

- ✓ Analyse du domaine : Identifier les concepts clés et les entités du système à modéliser.
- ✓ Identification des classes : Déterminer les classes principales en fonction des besoins fonctionnels.
- ✓ Définition des attributs et méthodes : Associer à chaque classe ses propriétés et ses comportements.
- ✓ Établissement des relations : Spécifier les associations, agrégations, compositions et héritages entre classes.
- ✓ Validation du modèle : Vérifier la cohérence et l'exhaustivité du diagramme avec les exigences.

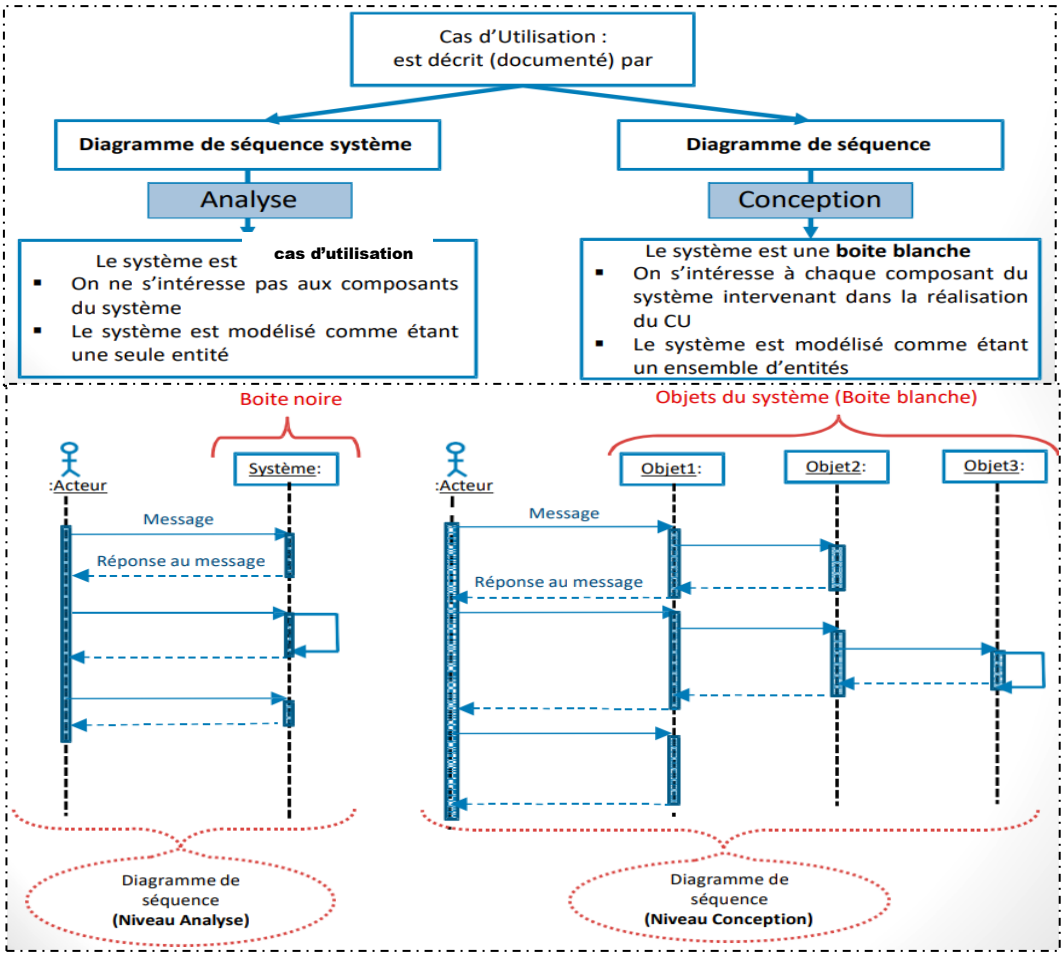
  Affinage et optimisation : Simplifier, normaliser et améliorer le diagramme pour faciliter l'implémentation.

IV.LE DIAGRAMME DE SEQUENCE

1. Introduction

Le diagramme de séquence modélise l'aspect dynamique du système.

Il s'agit d'une séquence d'interaction d'un point de vue temporel entre le système et les acteurs.

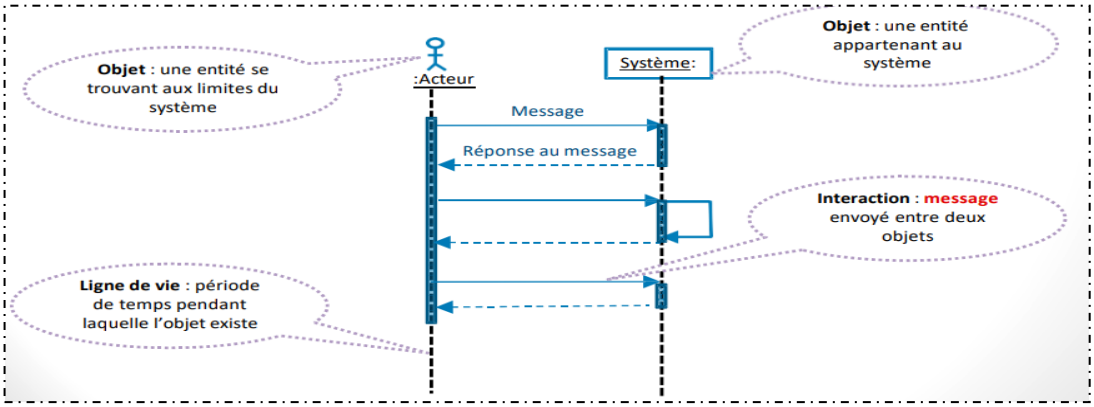


Les diagrammes de séquence sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML »

2. Eléments constituants un diagramme de séquence

Le diagramme de séquence permet de faire apparaitre:

- ✓ Les intervenants dans l'interaction (objets du système ou acteurs)
- ✓ La description de l'interaction (messages)
- ✓ Les interactions entre les intervenants



Objet :

Les objets sont identifiés par l'intermédiaire des cas d'utilisation ou par le diagramme de classe.

Les objets sont représentés comme suit : 

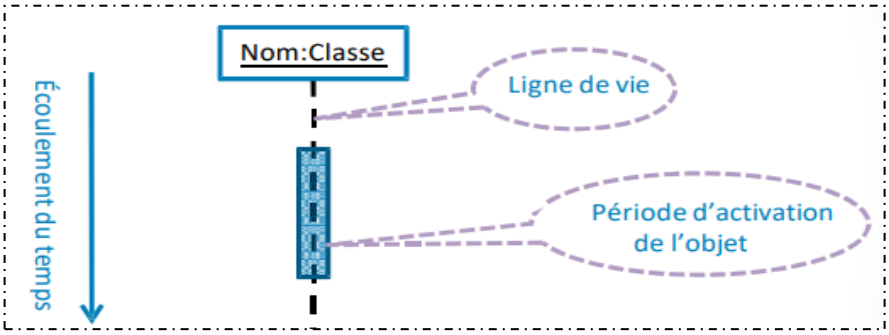
a. Ligne de vie :

✓ La ligne de vie est représentée par une ligne verticale pointillée en dessous de l'objet.

✓ La dimension verticale représente l'écoulement du temps.

✓ La période d'activité d'un objet est représentée par une bande rectangulaire superposée à la ligne de vie de l'objet.

✓ Un objet peut être actif plusieurs fois au cours de son existence.

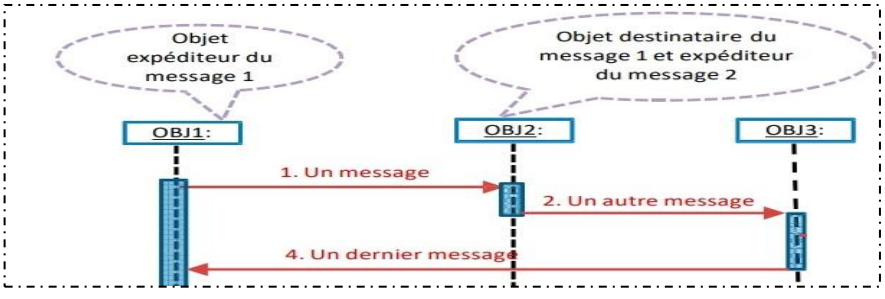


b. Messages :

✓ Les messages sont représentés sous forme de flèches.

✓ Ils sont étiquetés par le nom de l'opération ou du signal invoqué.

✓ L'ordre d'envoi d'un message est déterminé par sa position sur la ligne de vie; le temps s'écoule « de haut en bas »



Il existe différents types de messages :

Message synchrone :

Représenté par une flèche pleine ou complète et signifie que l'objet expéditeur envoie le message et reste bloqué tant que le destinataire n'a pas fin de traiter le message reçu.

Message asynchrone :

Représenté par une flèche vide ou incomplète et signifie que l'objet expéditeur envoie le message et ne reste pas bloqué pendant le traitement du message par le destinataire.

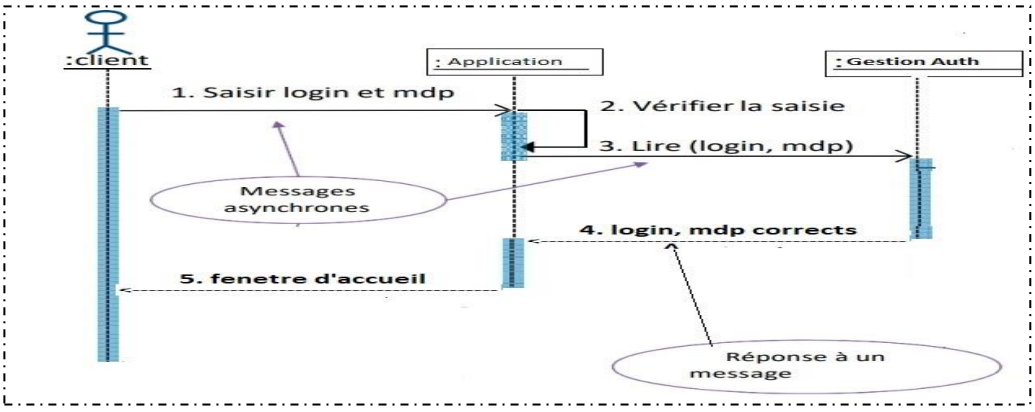
Message réflexif :

L'objet s'envoie un message à lui-même. L'expéditeur est lui-même le destinataire.

Message de retour :

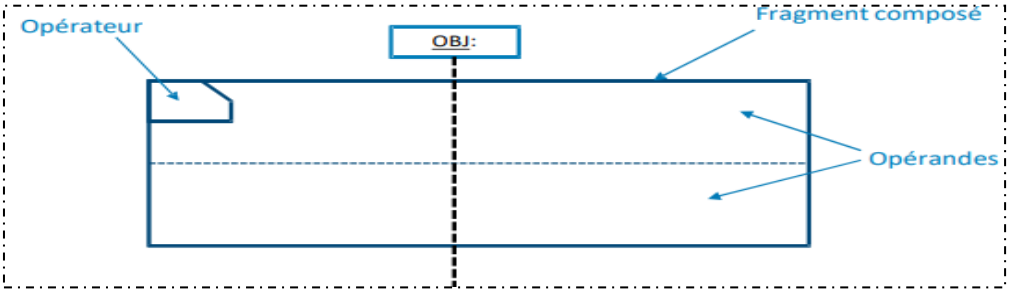
Représenté en pointillés. Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour

Exemple



c. Fragments composés :

- ✓ Les fragments composés doivent couvrir au moins une ligne de vie à tout moment, afin d'avoir une signification.
- ✓ Il est représenté par un rectangle dont le coin supérieur gauche contient un pentagone.
- ✓ La signification du fragment composé dépend fortement de l'opérateur d'interaction utilisé.

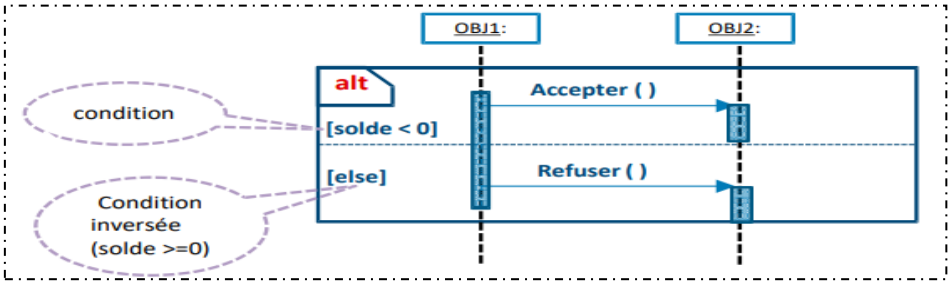


- **Le fragment « alt »**

L'opérateur alt désigne un choix ou une alternative: équivalent à SI … ALORS … SINON … composés

L'utilisation de l'opérateur else permet d'indiquer que la branche est exécutée si la condition du alt est fausse.

Une seule des deux branches sera réalisée dans un scénario donné.



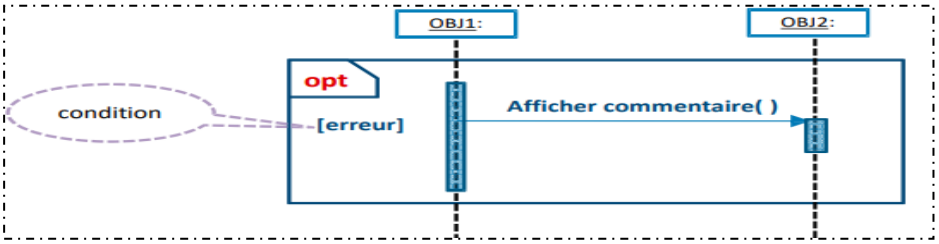
- **Le fragment « opt »**

L'opérateur opt désigne un choix de comportement où:

▪ Soit l'opérande seule s'exécute

▪ Soit rien ne s'exécute

Il est équivalent à SI … ALORS … | ➔ il est équivalent à un « alt » sans[else]

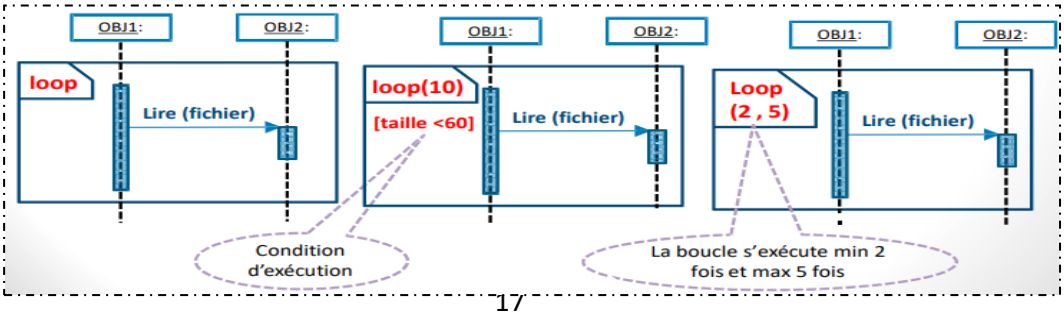


- **Le fragment « loop »**

✓ L'opérateur loop est utilisé pour décrire un ensemble d'interaction qui s'exécute en boucle.

✓ La condition spécifiée entre crochets indique la condition pour rester dans la boucle. Si la condition n'est plus satisfaite, alors la boucle est quittée.

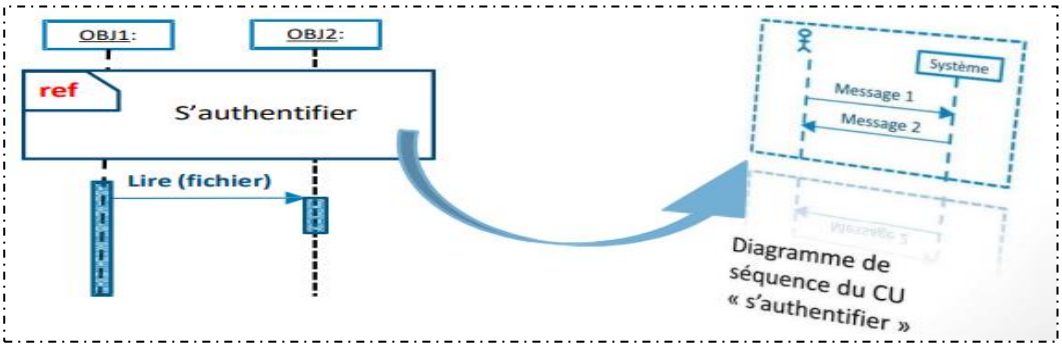
✓ On peut spécifier le nombre de répétition exacte ou l'intervalle de répétition de l'exécution de la boucle entre parenthèses.



- **Le fragment « ref »**

✓ L'opérateur ref est utilisé pour indiquer une référence vers un autre diagramme de séquence composés existant. Il peut être considéré comme un pointeur ou un raccourci vers un autre diagramme de séquence.

✓ Son rôle est de factoriser des parties de comportement utilisés dans plusieurs scénarios



V.LE DIAGRAMME D'ACTIVITE

1. Introduction

- ✓ Les diagrammes d'activités permettent de mettre l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.
- ✓ Dans la phase de conception, les diagrammes d'activités sont particulièrement adaptés à la description des cas d'utilisation. Plus précisément, ils viennent illustrer et consolider la description textuelle des cas d'utilisation

2. Eléments constituants un diagramme d'activité

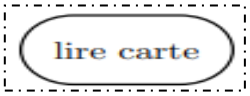
d. Activité

Comportement défini par un groupe d'instructions (action en UML)

Notation UML :

Exemple :

dans un processus de gestion d'emprunt (d'une bibliothèque)



- ✓ Nœud (activité) initial d'un diagramme d'activité

- ✓ Nœud (activite) final d'un diagramme d'activité

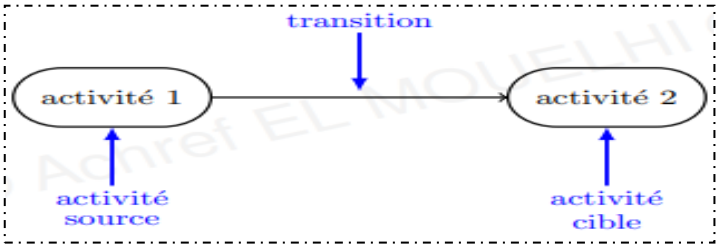
e. Transition

Le passage d'une activité vers une autre

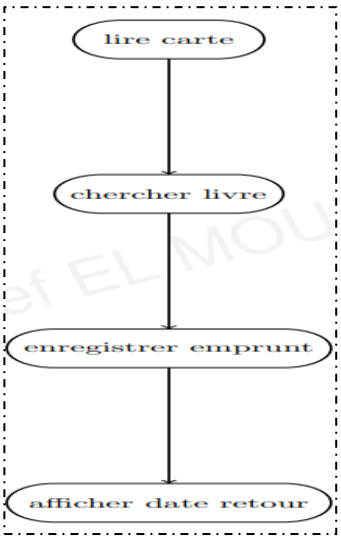
Déclenchées automatiquement à la fin de l'activité source provocant ainsi le début immédiat de l'activité cible

Représentée en ´ UML par une flèche (arc)

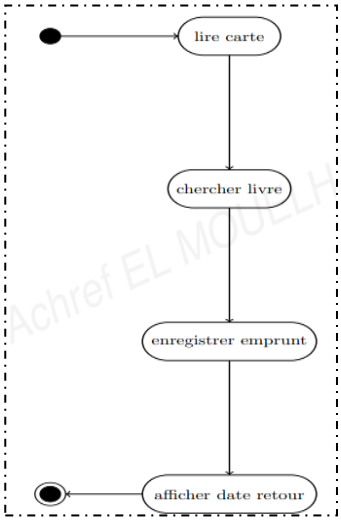
Activité + transition



Exemple



En ajoutant les nœuds initial et final, le diagramme devient :



Défauts de la solution précédente :

D'après notre système, on n'a pas besoin que la carte soit valide pour faire le reste.

Si on ne trouvait pas le livre recherche, alors l'emprunt pourrait être enregistré.

Solution

Définir des conditions de franchissement.

f. Condition de franchissement et nœud de décision

- **Condition de franchissement (ou de garde)**

✓ une transition peut avoir une condition

✓ expression booléenne exprimée en langage naturelle ´ (mathématique, logique...).

✓ placée entre crochet [condition(s)]

✓ accompagnée souvent d'un nœud de décision

- **Nœud de décision**

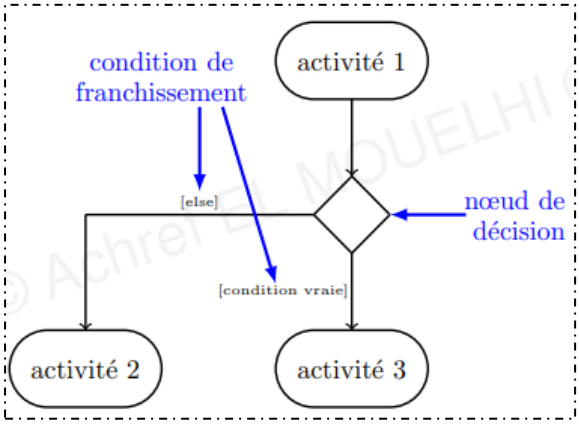
✓ Ou branchement conditionnel

✓ Schématisée en UML par un losange

✓ Permettant de faire un choix entre plusieurs sorties (au moins une entrée + plusieurs sorties)

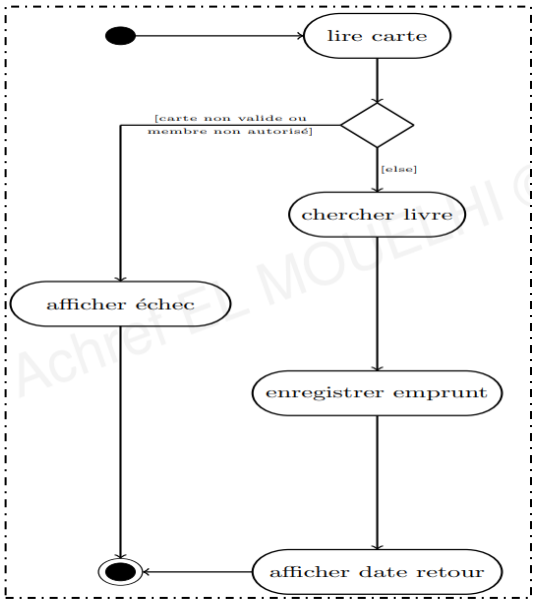
✓ Accompagnée souvent d'une condition de franchissement

Activité + transition + condition de franchissement + nœud de décision

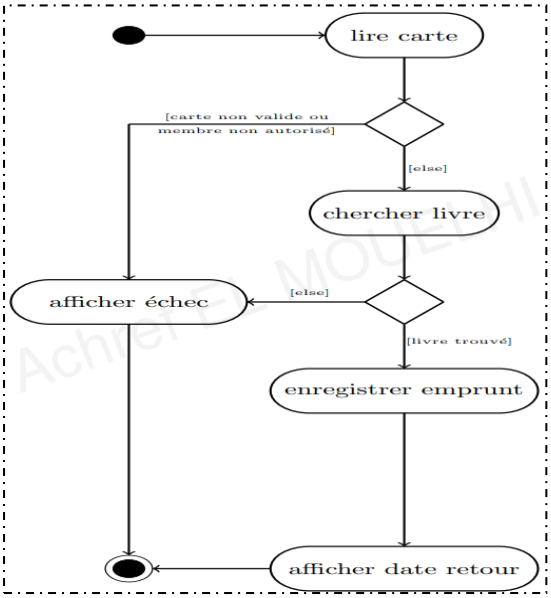


Exemple

Une seule condition



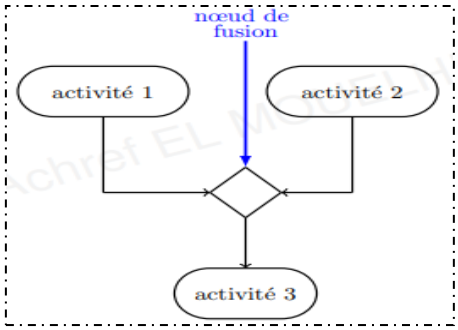
plusieurs conditions



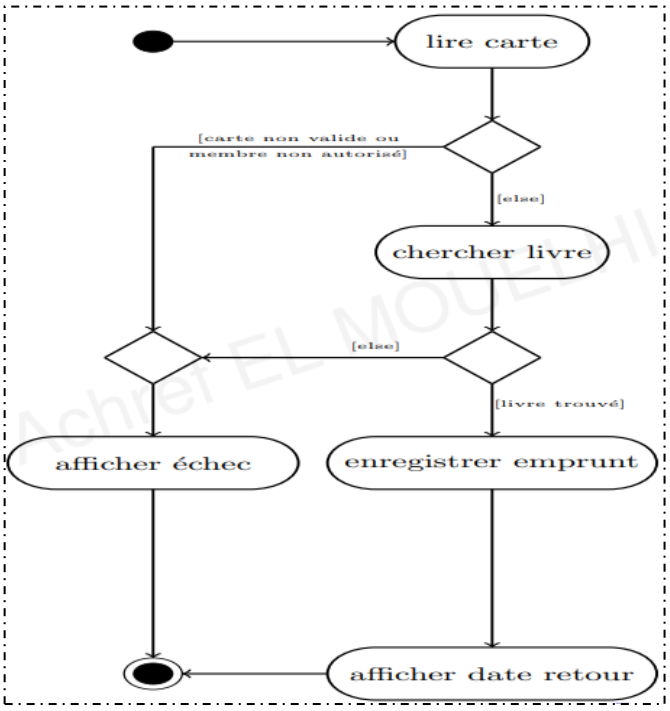
g. Nœud de fusion

Schématisée aussi en UML par un losange permettant de fusionner plusieurs entrées (plusieurs entrées + une sortie)

Nœud de fusion



Exemple avec un nœud de fusion



h. Nœud de bifurcation et d'union

- **Nœuds de bifurcation**

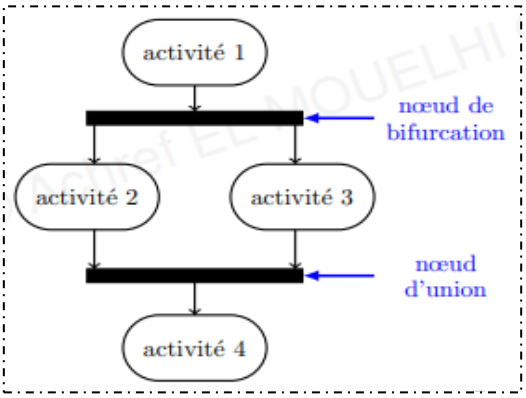
permettant de lancer des activités concurrentes (parallèles) schématisée en UML par un trait plein possédant une entrée et plusieurs sorties

- **Nœuds d'union**

Permettant de synchroniser des activités concurrentes (dites aussi parallèles) Schématisée en UML par un trait plein

Possédant plusieurs entre et une sortie.

Activités + transition + nœuds de bifurcation et d'union



Exemple avec nœuds de bifurcation et d'union