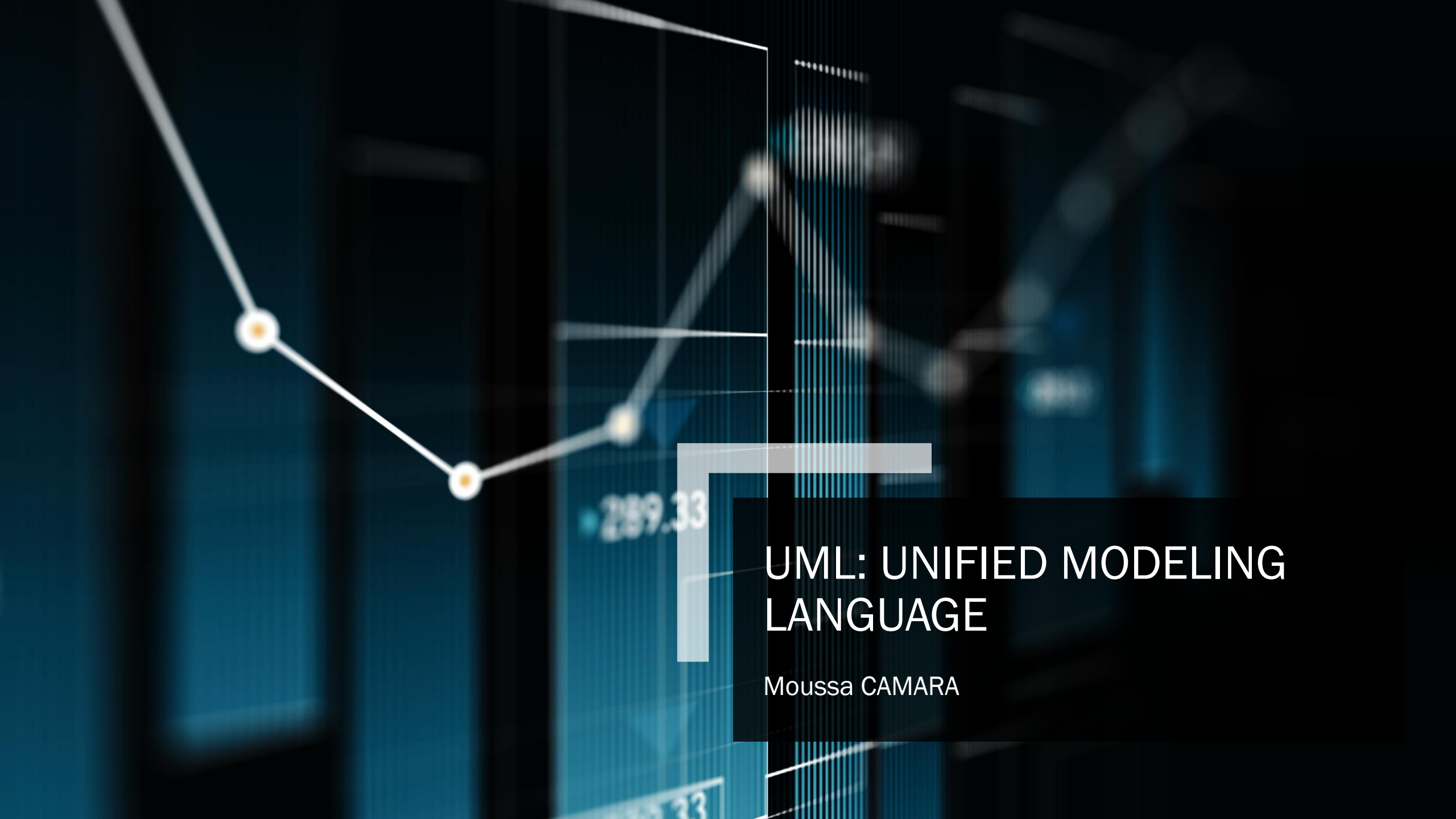


UML: UNIFIED MODELING LANGUAGE

Moussa CAMARA

UML: Introduction

Pour faire face à la complexité croissante des systèmes d'information, de nouvelles méthodes et outils ont été créées. La principale avancée des quinze dernières années réside dans la programmation orientée objet (POO).

Face à ce nouveau mode de programmation, les méthodes de modélisation classique (telle MERISE) ont rapidement montré certaines limites et ont dû s'adapter.

La méthode Merise est bien adaptée à l'automatisation de tâches séquentielles de gestion pure. En revanche, elle est mal adaptée aux environnements distribués, où de multiples applications externes à un domaine viennent interagir avec l'application à modéliser.

Dans ce contexte et devant le foisonnement de nouvelles méthodes de conception « orientée objet », l'Object Management Group (OMG) a eu comme objectif de définir une notation standard utilisable dans les développements informatiques basés sur l'objet.

UML: Définition

- Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement.

- L'UML est un outil permettant de spécifier les systèmes logiciels. Types de diagrammes standardisés pour vous aider à décrire et visuellement mapper la conception et la structure d'un système logiciel.
- En utilisant UML il est possible de modéliser à peu près tout type d'application, à la fois spécifiquement et indépendamment d'une plate-forme cible. Alors que UML est naturellement orienté vers la programmation orientée objet, mais il est tout aussi facile de modéliser les langages procéduraux tels que C, Visual Basic, etc

UML: Définition

- L'UML n'est pas un langage de programmation, mais il existe des outils qui peuvent être utilisés pour générer du code en plusieurs langages à partir de diagrammes UML.

- Les modélisations UML ne se dessinent bien sûr pas avec Paint. Ce sont les outils UML qui vous aident à utiliser le langage de modélisation. Mais trouver le bon outil n'est pas si facile. Il existe d'innombrables fournisseurs de programmes UML sur Internet, mais tous n'offrent pas les mêmes fonctions.

Mais pourquoi l’UML

- De la même façon qu'il vaut mieux dessiner une maison avant de la construire, il vaut mieux modéliser un système avant de le réaliser.

- UML pour :
  - ***Obtenir une modélisation de très haut niveau indépendante des langages et des environnements.***
  - ***Faire collaborer des participants de tous horizons autour d'un même document de synthèse.***
  - ***Faire des simulations avant de construire un système.***
  - ***Exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques****, spécifications, etc…*
  - *Documenter un projet.*
  - *Générer automatiquement la partie logiciel d'un système.*

Mais pourquoi l'UML

- Les auteurs de l'UML précisent néanmoins qu'une méthode basée sur l'utilisation UML doit être :

  - *Pilotée par les cas d'utilisation :*
    - la principale qualité d'un logiciel étant son utilité, c'est-à-dire son adéquation avec les besoins des utilisateurs, toutes les étapes, de la spécification des besoins à la maintenance, doivent être guidées par les cas d'utilisation qui modélisent justement les besoins des utilisateurs ;
  - *Centrée sur l'architecture :*
    - l'architecture est conçue pour satisfaire les besoins exprimés dans les cas d'utilisation, mais aussi pour prendre en compte les évolutions futures et les contraintes de réalisation. La mise en place d'une architecture adaptée conditionne le succès d'un développement. Il est important de la stabiliser le plus tôt possible ;
  - *Itérative et incrémentale :*
    - l'ensemble du problème est décomposé en petites itérations, définies à partir des cas d'utilisation et de l'étude des risques. Les risques majeurs et les cas d'utilisation les plus importants sont traités en priorité. Le développement procède par des itérations qui conduisent à des livraisons incrémentales du système. Nous avons déjà présenté le modèle de cycle de vie par incrément dans la section

Concepts de modélisation

- Le développement d'un système est axé sur trois modèles de systèmes globaux :
  - *Fonctionnel : ce sont des diagrammes de cas d'utilisation, qui décrivent la fonctionnalité du système du point de vue de l'utilisateur.*
  - *Objet : ce sont des diagrammes de classes qui décrivent la structure d'un système en termes d'objets, attributs, associations et opérations.*
  - *Dynamique : ce sont des diagrammes d'interaction, diagrammes états-transitions et diagrammes d'activités utilisés pour décrire le comportement interne du système.*

Concepts orientés objet

Les objets permettent la décomposition de systèmes complexes en éléments compréhensibles qui permettent de construire les pièces une par une.

Voici quelques concepts fondamentaux d'un monde orienté objet :

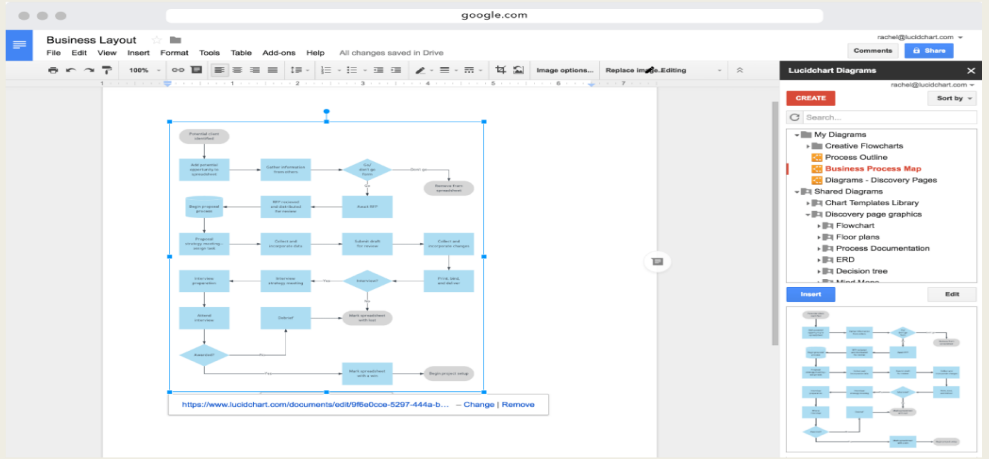
- *Objets Représentent une entité et le module de base*
- *Classe Plan d'un objet*
- *Abstraction Comportement d'une entité du monde réel*
- *Encapsulation Mécanisme qui consiste à relier les données et à les cacher du monde extérieur*
- *Héritage Mécanisme par lequel de nouvelles classes sont créées à partir d'une classe existante*
- *Polymorphisme Définit le mécanisme sous différentes formes.*

UML: Les outils

- Les modélisations UML ne se dessinent bien sûr pas avec Paint. Ce sont les outils UML qui vous aident à utiliser le langage de modélisation. Mais trouver le bon outil n'est pas si facile. Il existe d'innombrables fournisseurs de programmes UML sur Internet, mais tous n'offrent pas les mêmes fonctions.

- En voici quelques uns:
  1. Gliffy : un outil UML en ligne pour les débutants
  2. ArgoUML : le logiciel gratuit populaire pour les diagrammes simples
  3. MagicDraw : tout pour des diagrammes UML professionnels
  4. https://creately.com/fr/lp/outil-de-diagramme-uml/

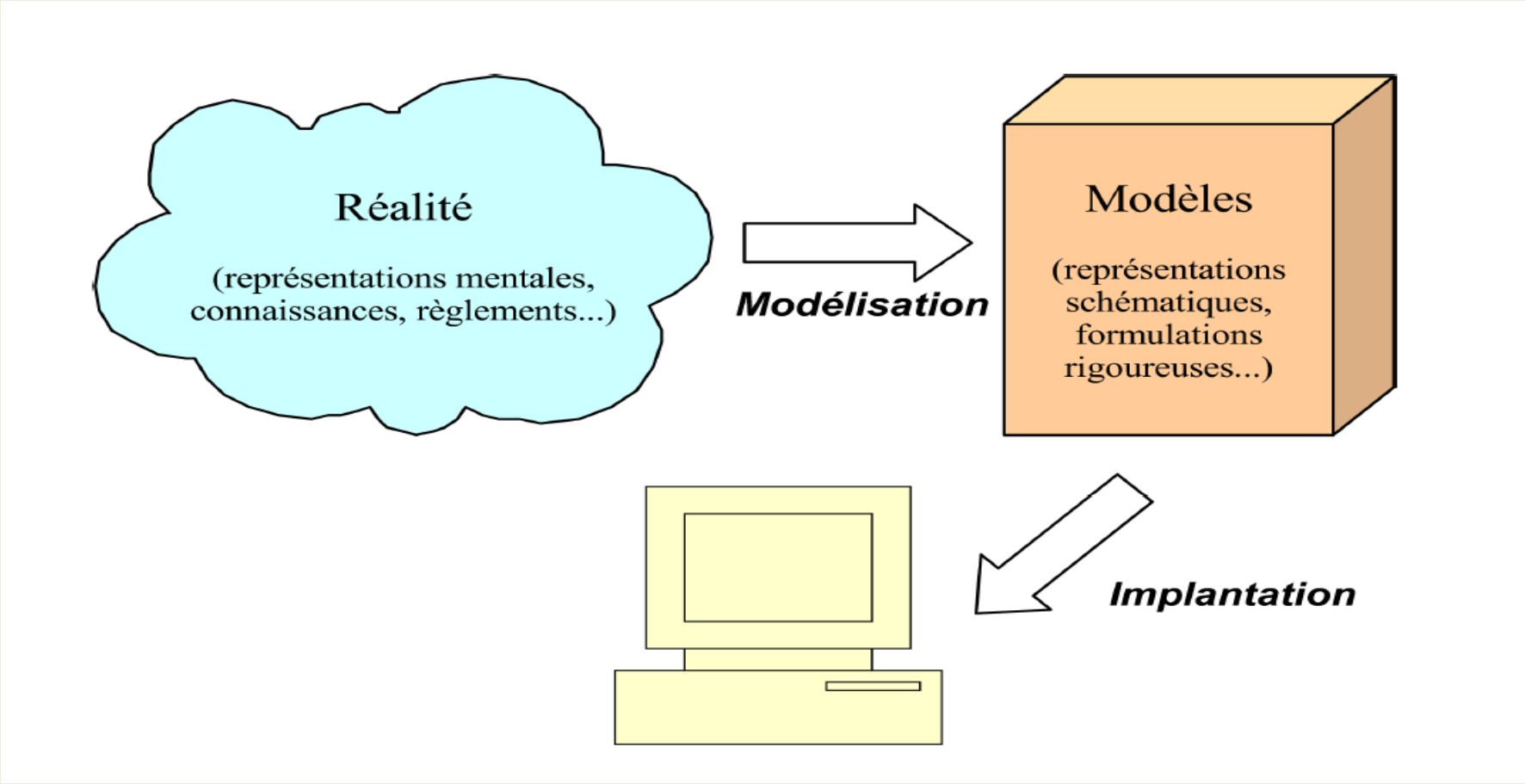
UML: Les outils



Types de diagrammes UML

- C’est quoi un diagramme ?
  - *Un diagramme fournit une représentation visuelle d'un aspect d'un système.*
- Il sert à quoi un diagramme ?
  - *Les diagrammes UML illustrent les aspects quantifiables d'un système qui peuvent être décrits visuellement, tels que les relations, le comportement, la structure ou la fonctionnalité.*
  - *Par exemple, un diagramme de classes décrit la structure du système ou les détails d'une implémentation, tandis qu'un diagramme de séquence montre l'interaction entre des objets dans le temps.*

UML: Modélisation

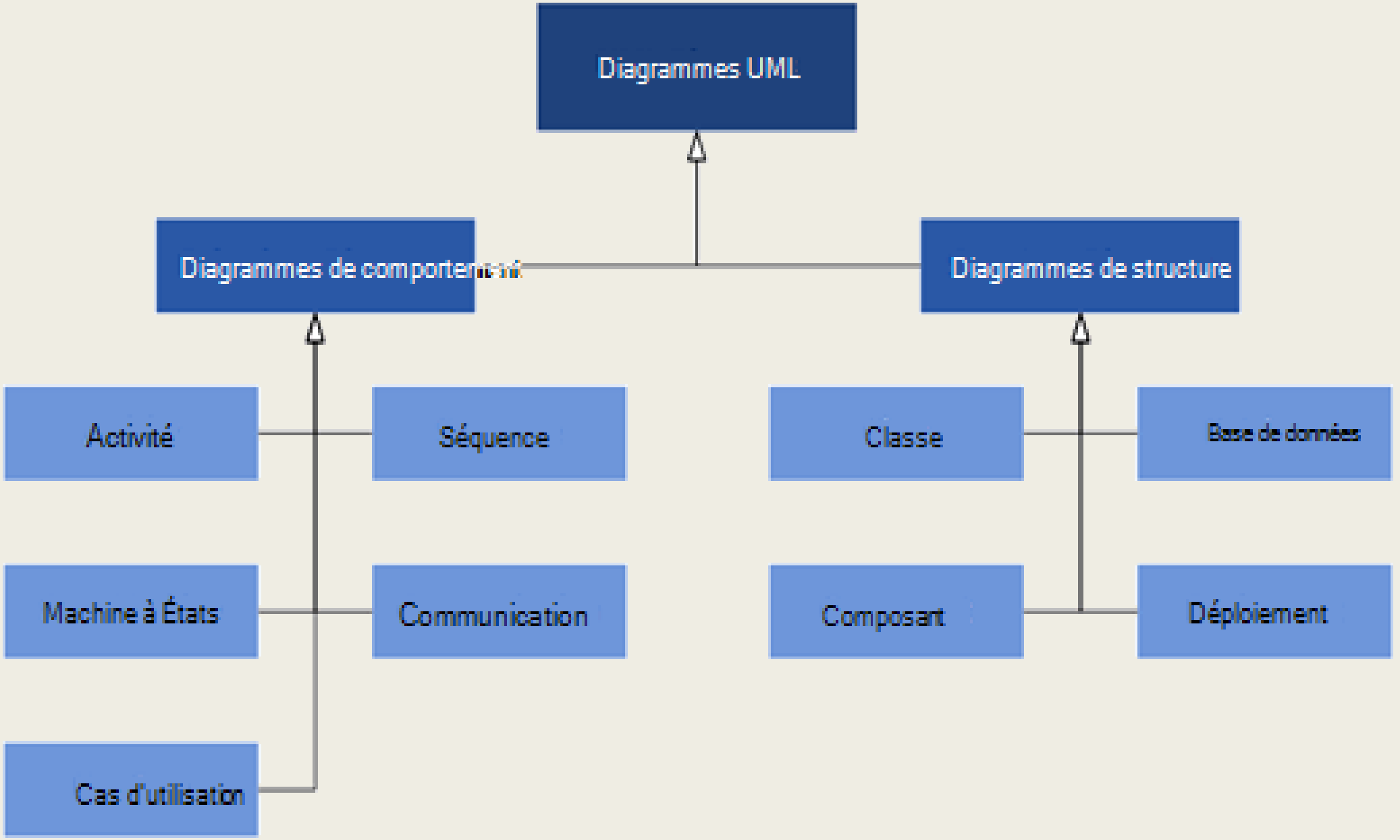


Types de diagrammes UML

- Ainsi, UML définit 9 types de diagrammes dans deux catégories de vues, les vues statiques et les vues dynamiques.

  - *Vues statiques:*
    - Les diagrammes de cas d'utilisation décrivent le comportement et les fonctions d'un système du point de vue de l'utilisateur
    - Les diagrammes de classes décrivent la structure statique, les types et les relations des ensembles d'objets
    - Les diagrammes d'objets décrivent les objets d 'un système et leurs relations
    - Les diagrammes de composants décrivent les composants physiques et l'architecture interne d'un logiciel
    - Les diagrammes de déploiement décrivent la répartition des programmes exécutables sur les différents matériels

  - *Vues dynamiques :*
    - Les diagrammes de collaboration décrivent les messages entre objets (liens et interactions)
    - Les diagrammes d'états-transitions décrivent les différents états d'un objet
    - Les diagrammes d'activités décrivent les comportements d'une opération (en termes d'actions)
    - Les diagrammes de séquence décrivent de manière temporelle les interactions entre objets et acteur


Diagrammes UML
Diagrammes de comportement
Diagrammes de structure
Activité
Séquence
Machine à États
Communication
Cas d'utilisation
Classe
Base de données
Composant
Déploiement

1- Diagramme de cas d'utilisation – use case

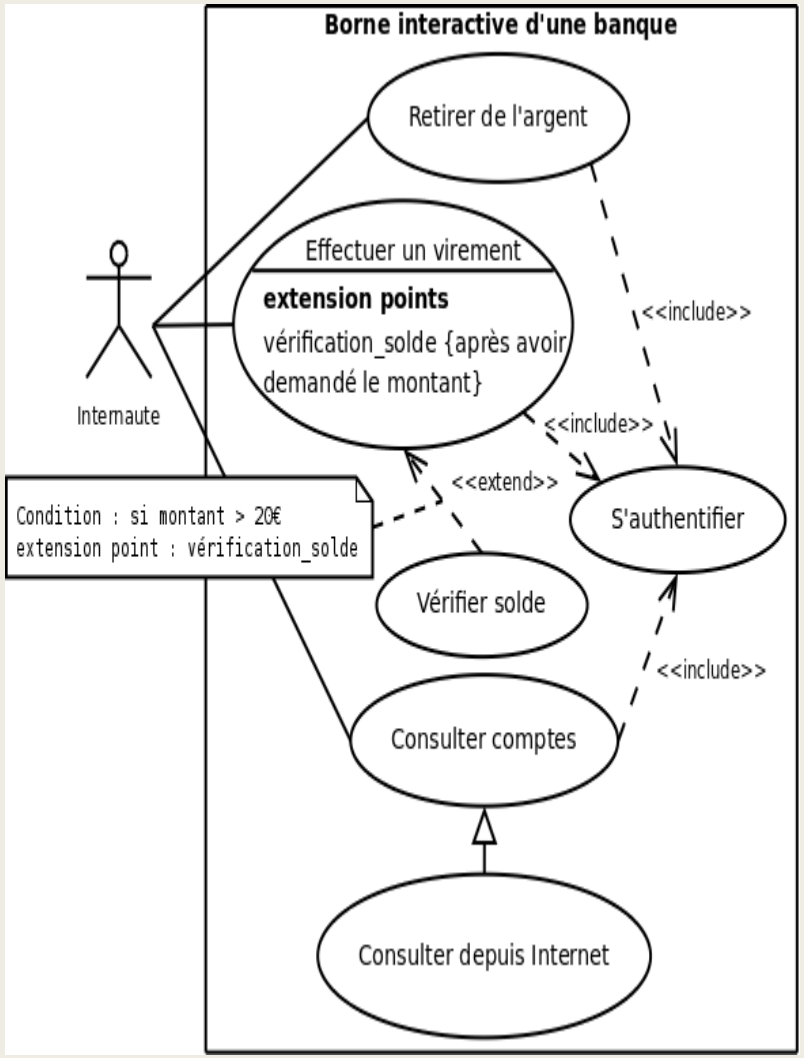
Les cas d'utilisation sont utilisés tout au long du projet.

Dans un premier temps, on les crée pour identifier et modéliser les besoins des utilisateurs. Ces besoins sont déterminés à partir des informations recueillies lors des rencontres entre informaticiens et utilisateurs. Il faut impérativement proscrire toute considération de réalisation lors de cette étape.

Dans le cadre d'une approche itérative et incrémentale, il faut affecter un degré d'importance et un coefficient de risque à chacun des cas d'utilisation pour définir l'ordre des incréments à réaliser.

La détermination et la compréhension des besoins sont souvent difficiles car les intervenants sont noyés sous de trop grandes quantités d'informations : il faut clarifier et organiser les besoins des clients (les modéliser).

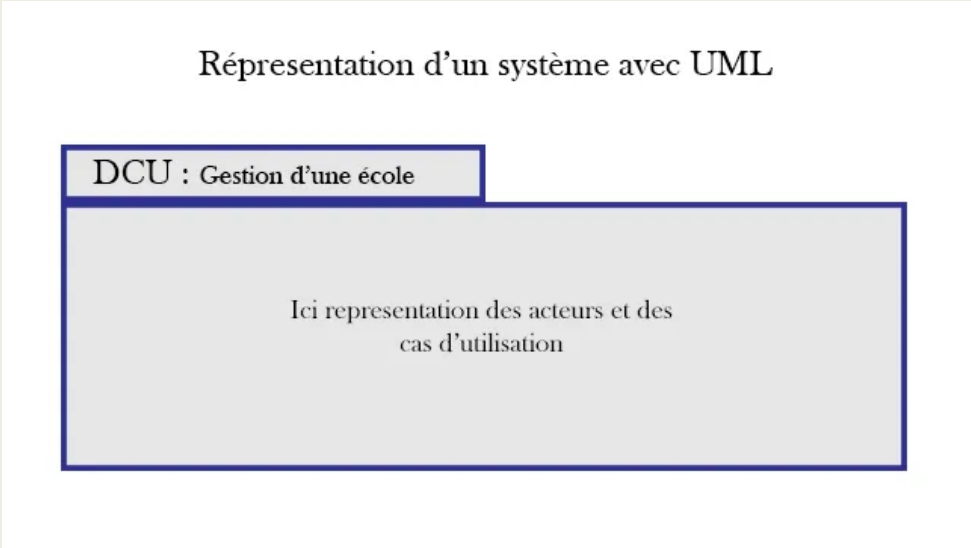
Pour cela, les cas d'utilisation identifient les utilisateurs du **système** (**acteurs**) et leurs **interactions** avec le système. Ils permettent de classer les acteurs et structurer les objectifs du système.



Le systéme

- Il represente notre besoin (Ce que l'on souhaite developer).
  Il peut être:
  - *Un site internet*
  - *Un processus business*
  - *Un composant Logiciel*
  - *Une application*

  <u>Forme graphique: un rectangle</u>



Pour résumé:

Avant de développer un système, il faut savoir précisément à **QUOI** il

devra servir, cad à quels besoins il devra répondre.

Modéliser les besoins permet de :

Faire l'inventaire des fonctionnalités attendues ;

Organiser les besoins entre eux, de manière à faire apparaître des relations (réutilsations possibles, ...).

Un cas d'utilisation est l'expression d'un service réalisé de bout en bout, avec un déclenchement, un déroulement et une n, pour l'acteur qui l'initie.

Les acteurs

Un acteur est un type stéréotypé représentant une abstraction qui réside juste en dehors du système à modéliser.

Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système. (la même personne physique peut donc être représentée par plusieurs acteurs en fonction des rôles qu'elle joue).

Pour identifier les acteurs, il faut donc se concentrer sur les rôles joués par les entités

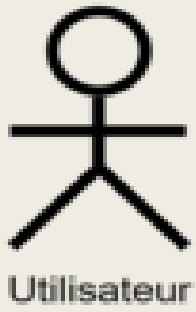
extérieures au périmètre.

un acteur est externe au périmètre de l'étude

Les acteurs

Il existe 4 catégories d’acteurs :

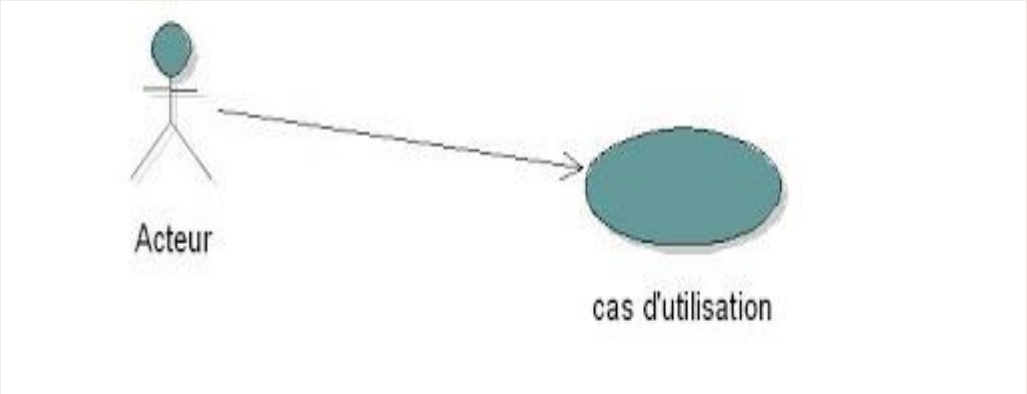
- les acteurs principaux : les personnes qui utilisent les fonctions principales du système
- les acteurs secondaires : les personnes qui effectuent des tâches administratives ou de maintenance.
- le matériel externe : les dispositifs matériels incontournables qui font partie du domaine de l’application et qui doivent être utilisés.
- les autres systèmes : les systèmes avec lesquels le système doit interagir.



Les cas d’utilisations

- Le cas d’utilization est une action qui accomplit une tâche de notre systéme.

- Forme graphique: ballon oval

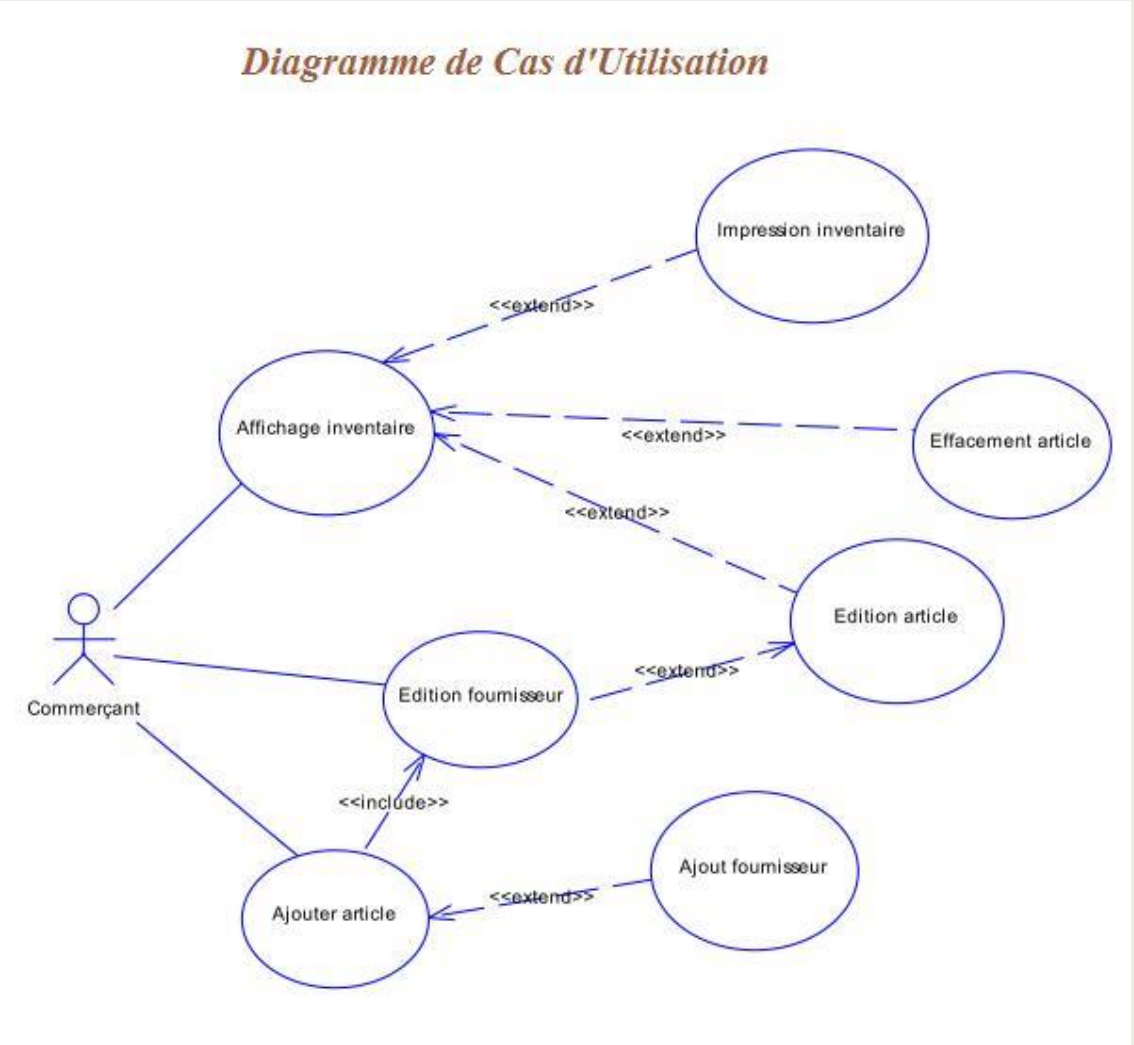


La relation :

Elle exprime l'interaction existant entre un acteur et un cas d'utilisation.

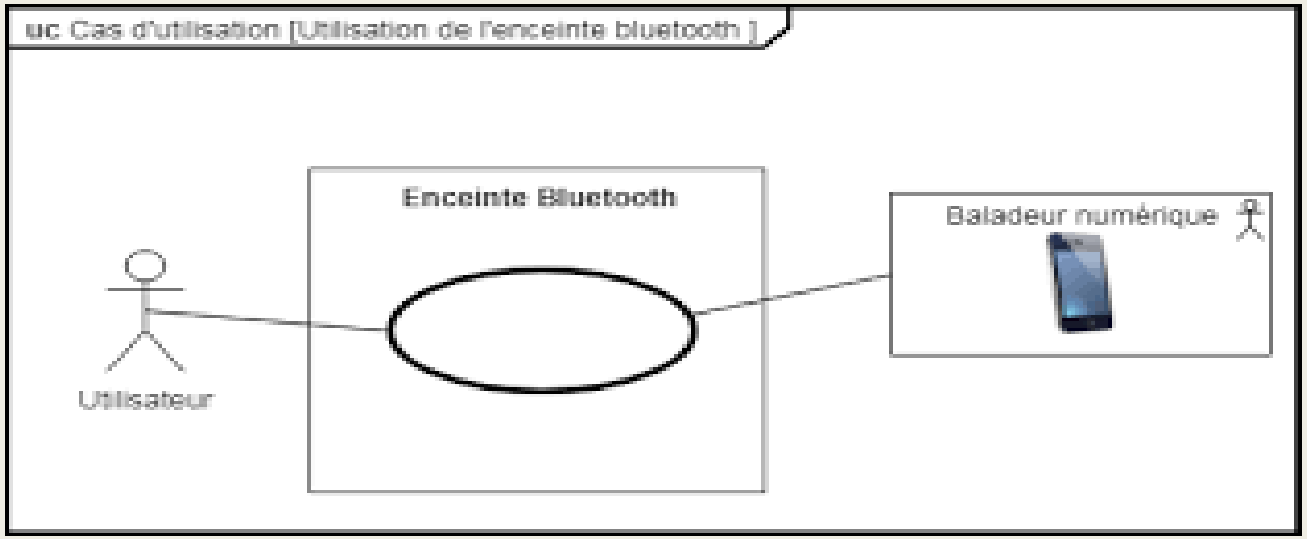
Il existe 3 types de relations entre cas d'utilisation :

- la relation de généralisation
- la relation d'extension
- la relation d'inclusion



Relation association

- L'association est une connexion sémantique entre deux élements (relation logique). Elle peut être binaire, dans ce cas elle est représentée par un simple trait



Relation d'extension

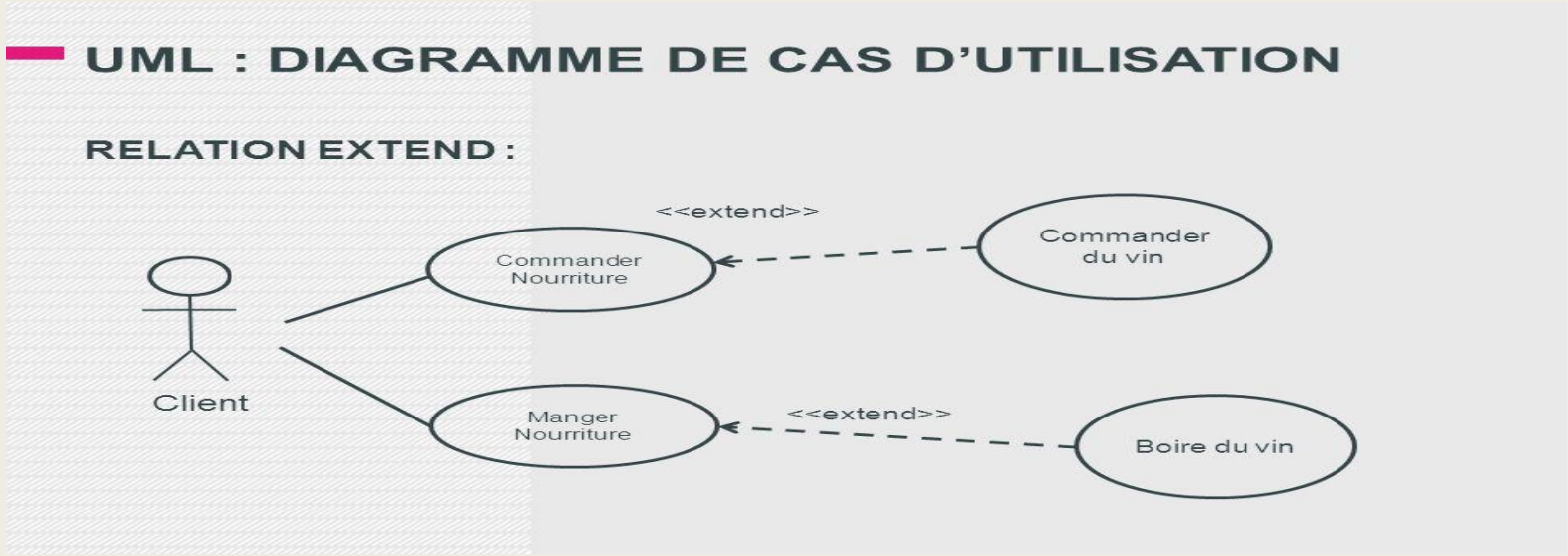
Cette relation indique que le cas d'utilisation source ajoute son comportement au cas d'utilisation

destination. L'extension peut être soumise à condition. Le comportement ajouté est inséré au niveau d'un point d'extension défini dans le cas d'utilisation destination.

Elle permet de modéliser les variantes de comportement d'un cas d'utilisation (selon les interactions des acteurs et l'environnement du système).

C'est **optionnelle**: dans notre exemple 'commande du vin' est optionnelle par rapport à la 'commande de nourriture'

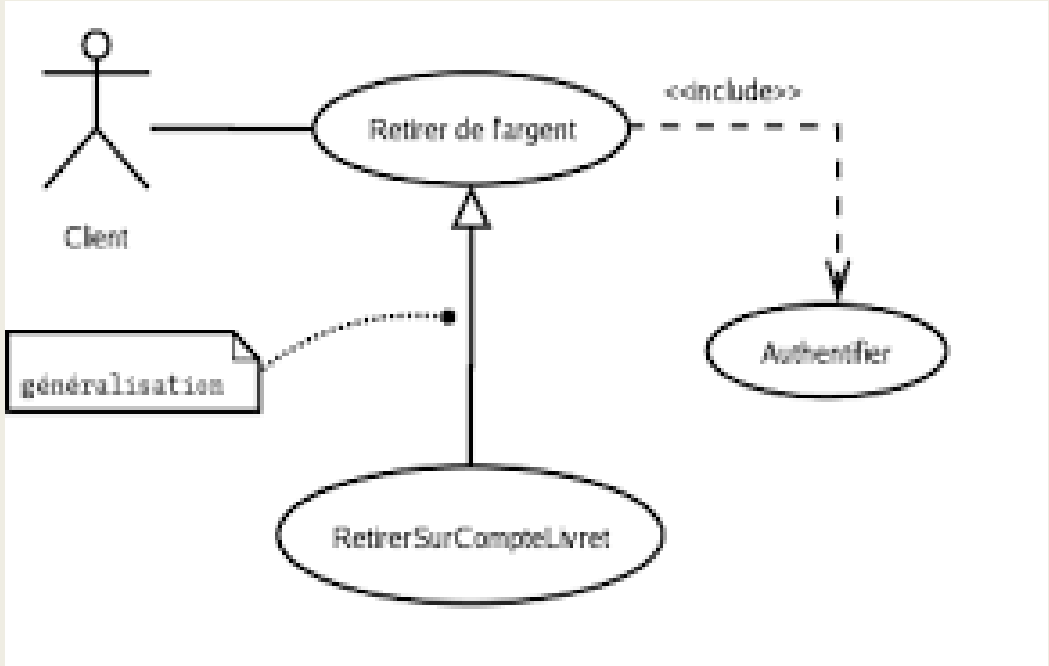
NB: FAIRE ATTENTION AU SENS DE LA FLECHE



Relation d’inclusion

Elle indique que le cas d’utilisation source contient aussi le comportement décrit dans le cas d’utilisation destination. L’inclusion a un caractère obligatoire, la source spécifiant à quel endroit le cas d’utilisation cible doit être inclus.

NB: FAIRE ATTENTION AU SENS DE LA FLECHE



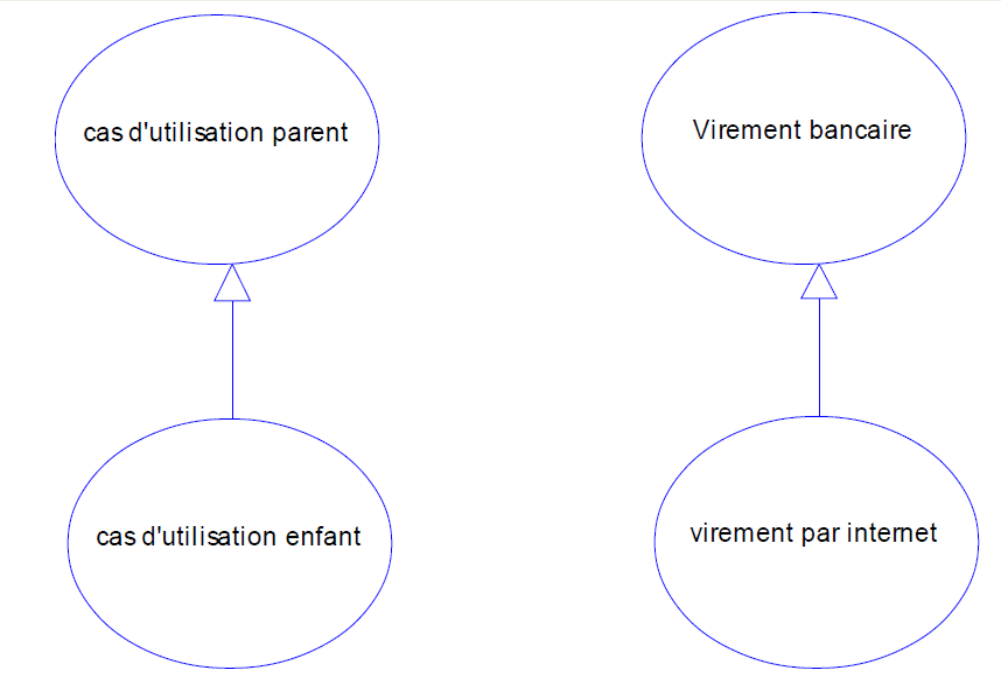
En résumé

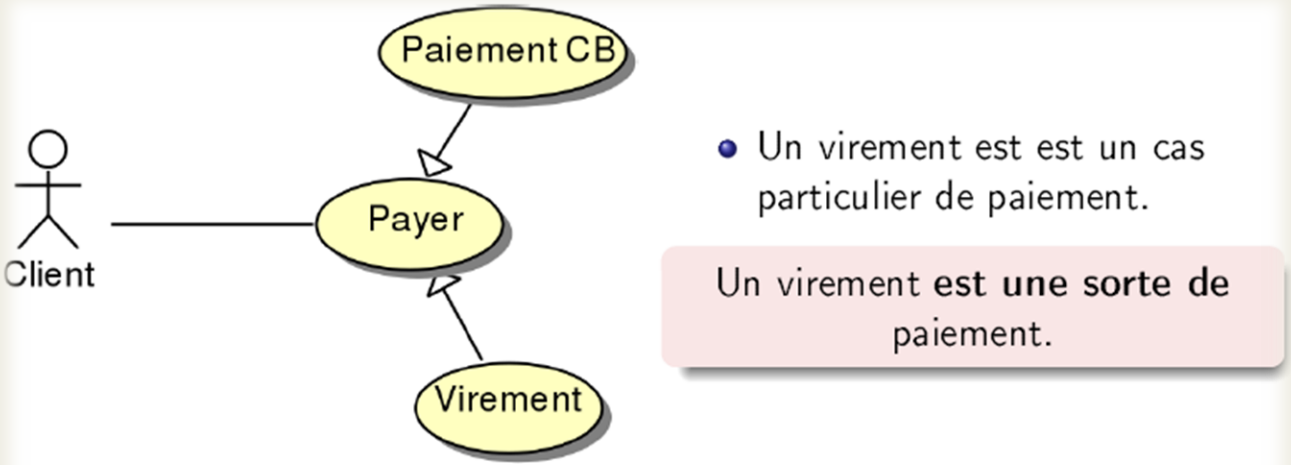
- Les inclusions et les extensions sont représentées par des dépendances.
  - Lorsqu'un cas B inclut un cas A, B dépend de A.
  - Lorsqu'un cas B étend un cas A, B dépend aussi de A.

- On note toujours la dépendance par une èche pointillée B ---> A qui se lit B dépend de A .
- Lorsqu'un élément A dépend d'un élément B, toute modification de B sera susceptible d'avoir un impact sur A.
- Les incude et les extend sont des stéréotypes (entre guillements) des relations de dépendance.

Relation de généralisation

Dans une relation de généralisation entre 2 cas d'utilisation, le cas d'utilisation enfant est une spécialisation du cas d'utilisation parent.





- Un virement est est un cas particulier de paiement.

Un virement **est une sorte de** paiement.

- La flèche pointe vers l'élément général.
- Cette relation de généralisation/spécialisation est présente dans la plupart des diagrammes UML et se traduit par le concept d'**héritage** dans les langages orientés objet.

Exercice

- Definir un cas d'utisation d'un systeme de paiement en ligne.
  - *Acteur primaire: Le client*
  - *Acteur Secondaire: La banque*
  - *Systeme: Compte bancaire*
  - *Cas d'utilisations:*
    - Connecter
      - *Verifier mot de passe*
      - *Afficher message d'erreur*
    - Verifier le solde
    - Faire un paiement
      - *Payer avec compte courant*
      - *Payer avec un paypal*

Exercice

- Le joueur se connecte
- Une fois connecté, il choisit un personnage apres avoir éventuellement consulté le catalogue des personnes.
- Il peut alors entrer dans l'aréne et jouer (se déplacer, attaquer, gagner, perdre).
- Il peut aussi pendant le jeu, dialoguer avec les autres joueurs.
- A tout moment (connecté ou pas), il peut consulter l'aide du jeu

TP – Gestion de blog d'articles

- Cahier des charges:
  - *Pour son blog, notre client a besoin d'un système de gestion des utilisateurs.*
  - *Les utilisateurs non identifiés pourront lire les articles (Post)*
  - *Les utilisateurs identifiés et autorisés pourront poster des articles (Post)*
  - *Les utilisateurs identifiés pourront commenter des articles (Post)*

- **FAIRE LE DIAGRAMME DE CAS D'UTILISATION DU BLOG**

TP – Use case

- Voir use-case.pdf

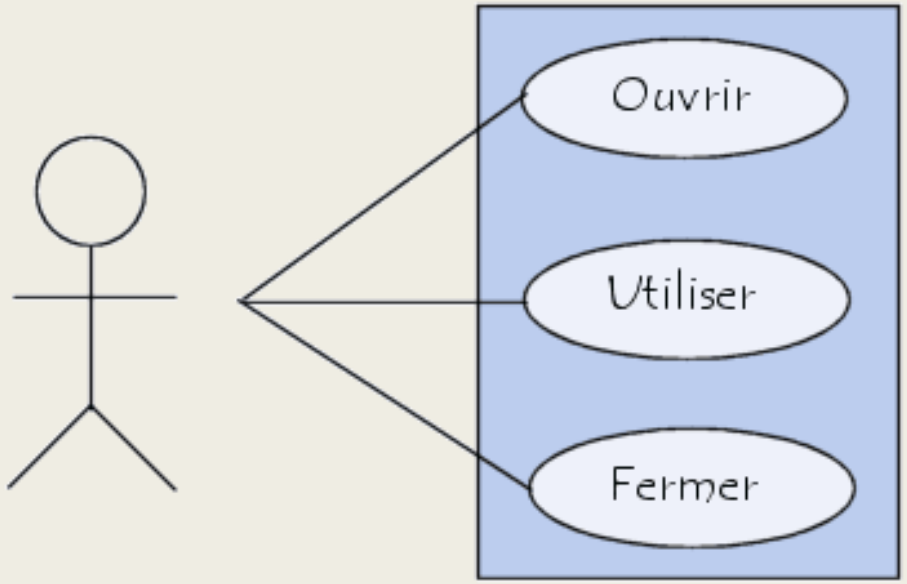


Diagramme de classe

- Le diagramme de classe est un diagramme faisant partie des diagrammes structurels et est un des diagrammes d'UML le plus utilisé du fait de sa notation syntaxique riche.

- Il représente la structure d'une application orientée objet en montrant les classes et les relations qui s'établissent entre elles.

- En résumé:

  Le système est composé d'objets qui interagissent entre eux et avec les acteurs pour réaliser ces cas d'utilisation.

  Les diagrammes de classes permettent de spécier la structure et les liens entre les objets dont le système est composé.

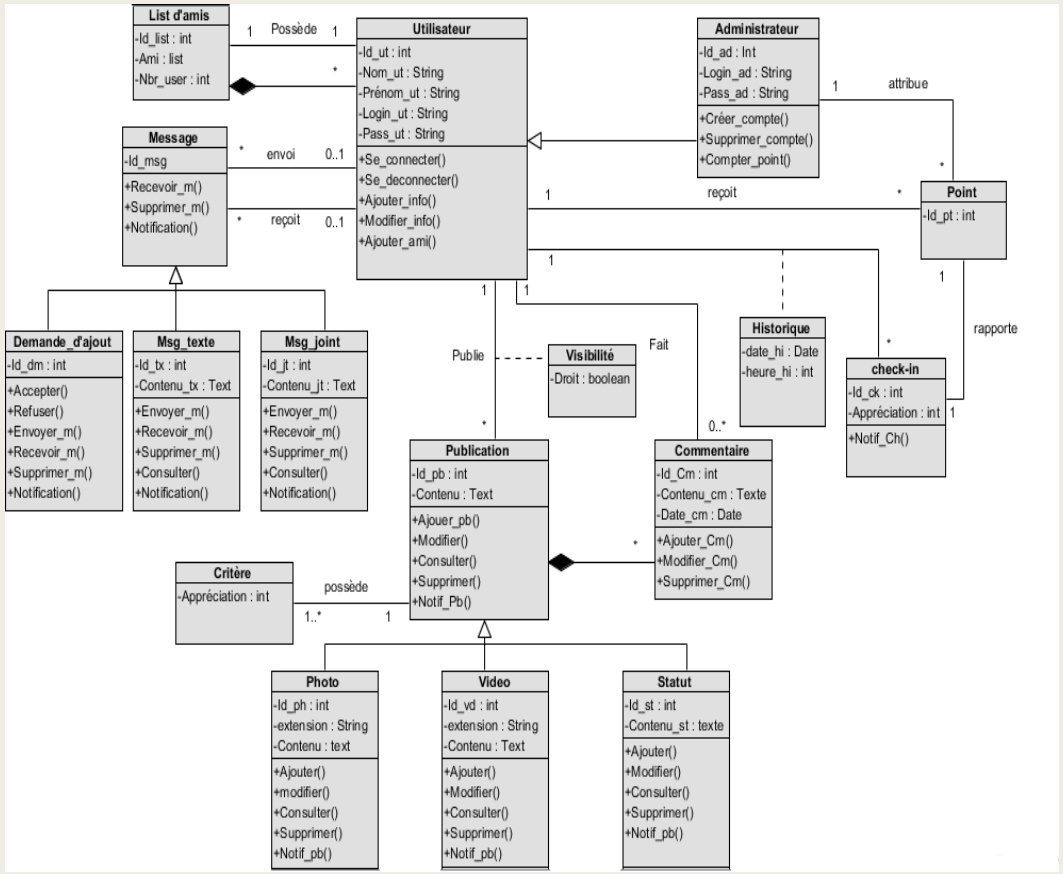
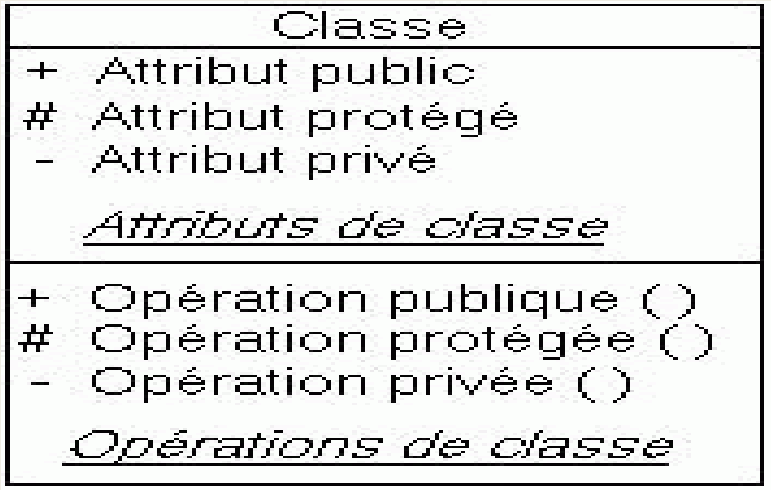


Diagramme de classe

Comme pour les attributs, on retrouve 3 niveaux de visibilité pour les opérations :

- public (+) : l'opération est visible pour tous les clients de la classe
- protégé (#) : l'opération est visible pour les sous-classes de la classe
- privé (-) : l'opération n'est visible que par les objets de la classe dans laquelle elle est déclarée.



La notion de relation

S’il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

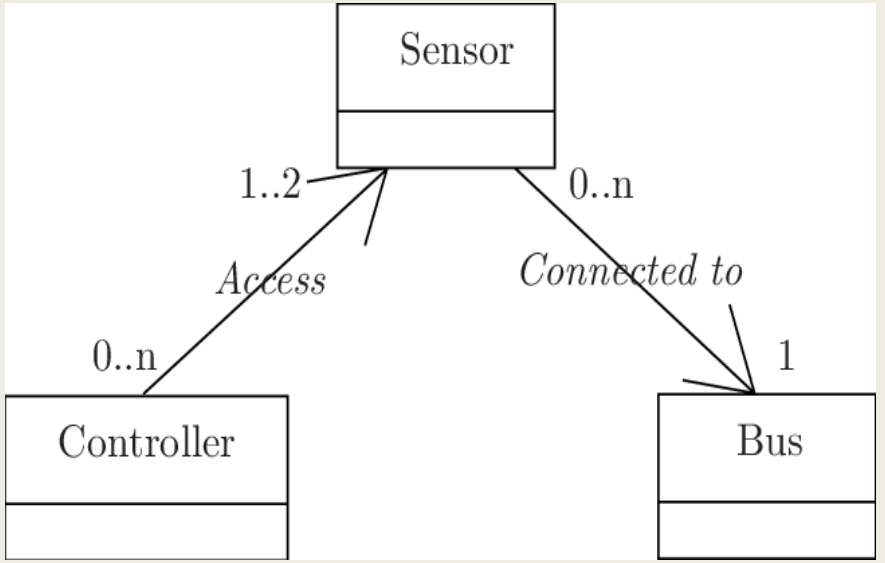
Il existe plusieurs types de relations entre classes :

- l’association
- la généralisation/spécialisation (Heritage)
- la dépendance
- L’aggrégation

Association

Il existe deux types d'associations: **association binaire et association n-aire**.

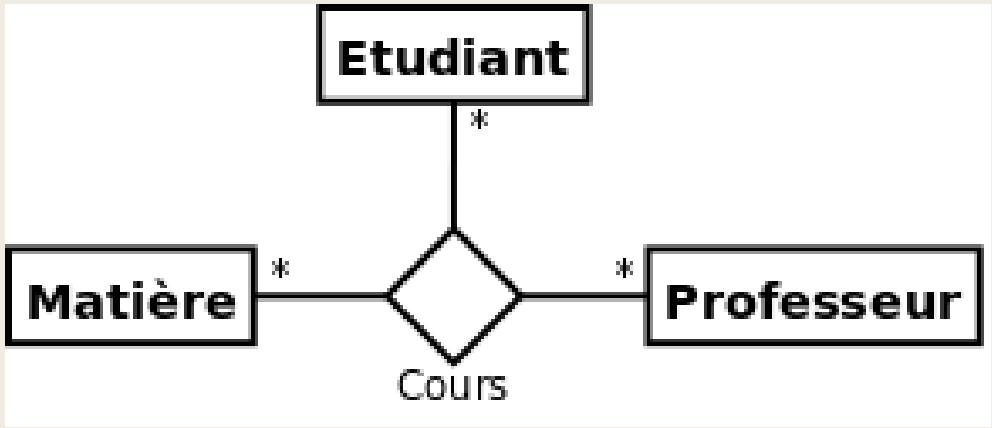
Une **association binaire** est matérialisée par un trait plein entre les classes associées. Elle peut être ornée d'un nom, avec éventuellement une précision du sens de lecture (▸ ou ◂).



Association n-aire

Une **association n-aire** lie plus de deux classes. C’est une association qui est très mal représentée dans certains outils.

On représente une association n-aire par un grand losange avec un chemin partant vers chaque classe participante. Le nom de l'association, le cas échéant, apparaît à proximité du losange.



Exemple

La notion de multiplicité permet le contrôle du nombre d'objets intervenant dans chaque instance d'une association.

Exemple :

un article n'appartient qu'à une seule catégorie (1)

une catégorie concerne plus de 0 articles, sans maximum (*).

Exercice 1

Une librairie vend des livres, caractérisés par leur auteur et leur nombre de pages .

certains livres possèdent également d’autres caractéristiques : une fourchette des âges pour les livres pour enfants, et la discipline et le niveau pour les livres scolaires.

Cardinalités ou Multiplicités

- La multiplicité associée à une terminaison d'association, d'agrégation ou de composition déclare le nombre d'objets susceptibles d'occuper la position définie par la terminaison d'association.

1	Un et un seul
0..1	Zéro ou un
N ou *	N (entier naturel)
M..N	De M à N (entiers naturels)
0..*	De zéros à plusieurs
1..*	De 1 à plusieurs

Exercice: les cardinalités

1. Exo1
    1. On considère une entreprise, et on suppose qu'un chef dirige plusieurs salariés (les subordonnés) et que le chef est lui-même un salarié.

2. Exo2:
    1. On considère une université, et les personnes y travaillant qui peuvent être des étudiants ou des enseignants.

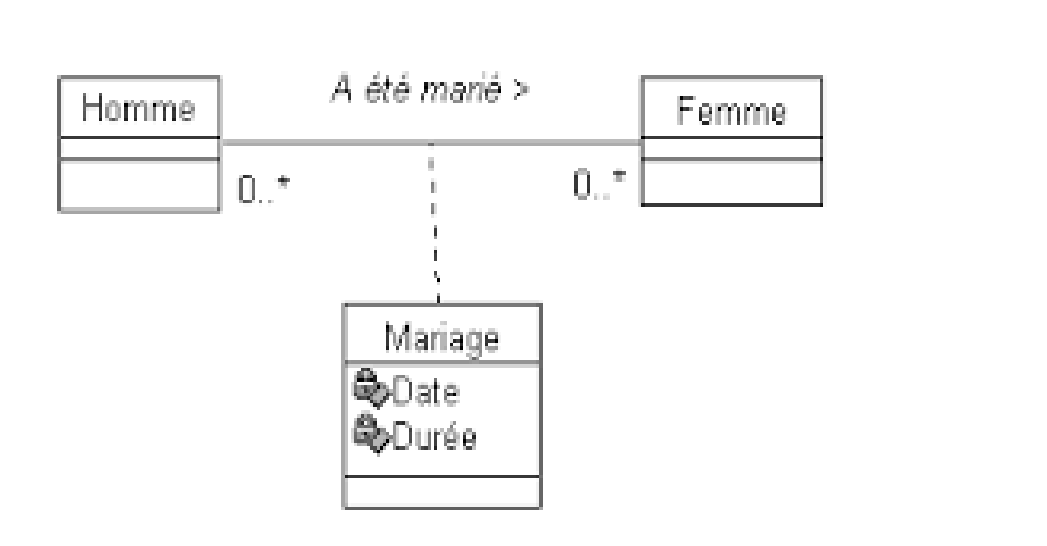
classe-association

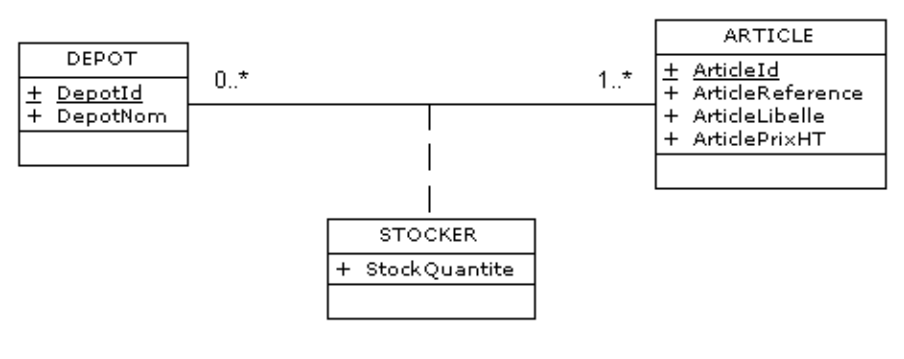
- On peut avoir une classe-association:

  Les attributs d'une classe dépendent fonctionnellement de l'identifiant de la classe. Parfois, un attribut dépend fonctionnellement de 2 identifiants, appartenant à 2 classes différentes.

  Par exemple, l'attribut « Date ou Durée » dépend fonctionnellement de Homme et de la femme . On va donc placer l'attribut « Date ou Durée » dans l'association « a été marié ».

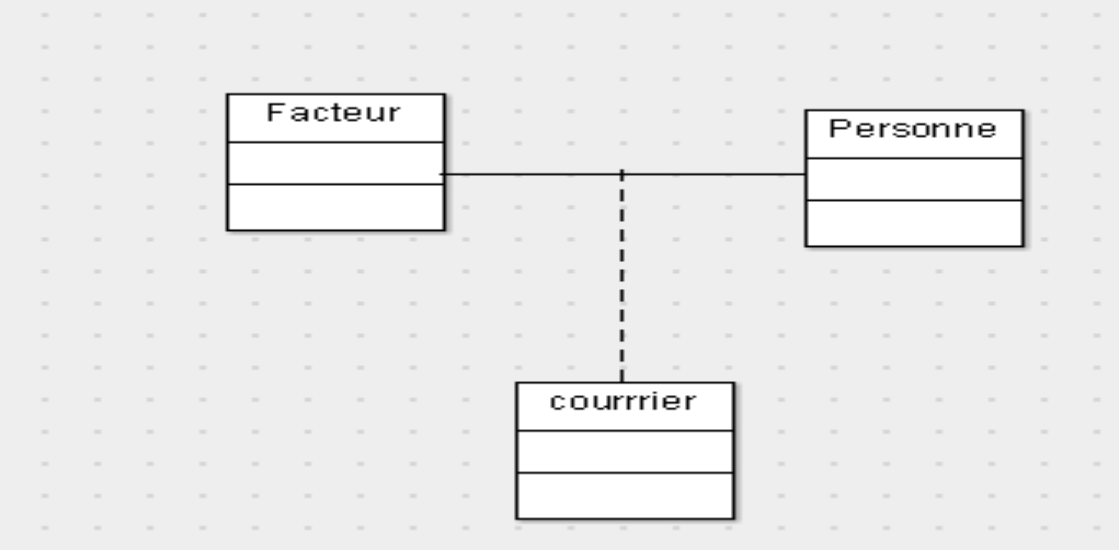
  Dans ce cas, l'association est dite « porteuse d'attributs ».





- Il est parfois nécessaire de compléter une association par des attributs qui caractérisent la nature de la relation existant entre 2 classes. Cela peut être représenté par une classe d'association, à laquelle on ajoute des attributs

- On peut prendre l'exemple de la poste avec les classes: Facteur, Personne ainsi on peut faire une classe associatio



L’agrégation

- Une agrégation est une forme particulière d'association. Elle représente la relation d'inclusion d'un élément dans un ensemble.

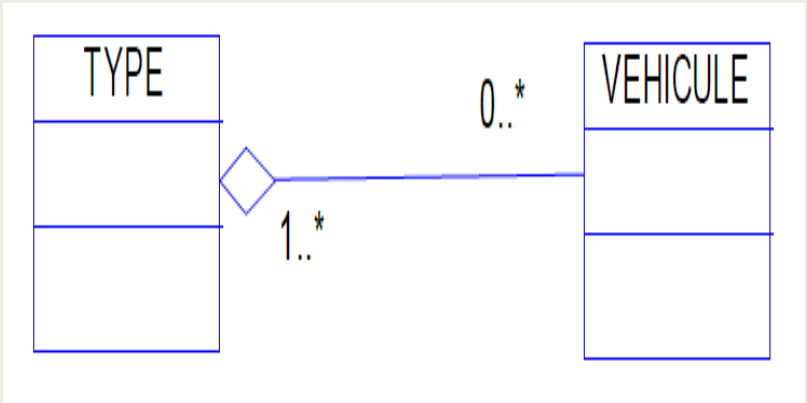
  - Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l’autre extrémité.
  - L’agrégation ne peut concerner qu’un seul rôle d’une association.
  - L’agrégation se représente toujours avec un petit losange du côté de l’agrégat.
  - Le choix d’une association de type agrégation traduit la volonté de renforcer la dépendance entre classes. C’est donc un type d’association qui exprime un couplage plus fort entre les classes.
  - L’agrégation permet de modéliser des relations de type maître et esclaves.
  - L’agrégation permet de modéliser une contrainte d’intégrité et de désigner l’agrégat comme
  - contrainte.

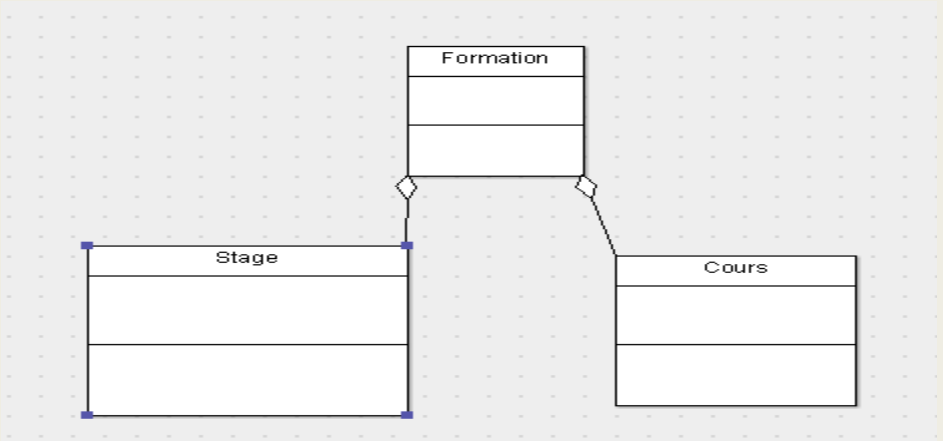
L’agrégation

L’exemple ci-dessus montre que l’on veut gérer une classification de véhicules. Chaque véhicule est classifié selon son type. En conséquence, il sera possible de prendre connaissance pour un véhicule de l’ensemble des caractéristiques du type de véhicule auquel il est associé.

NB : un agrégat peut être multiple. Dans l’exemple ci-dessous, un véhicule peut appartenir à plusieurs types.

Ici un pays est composé de région





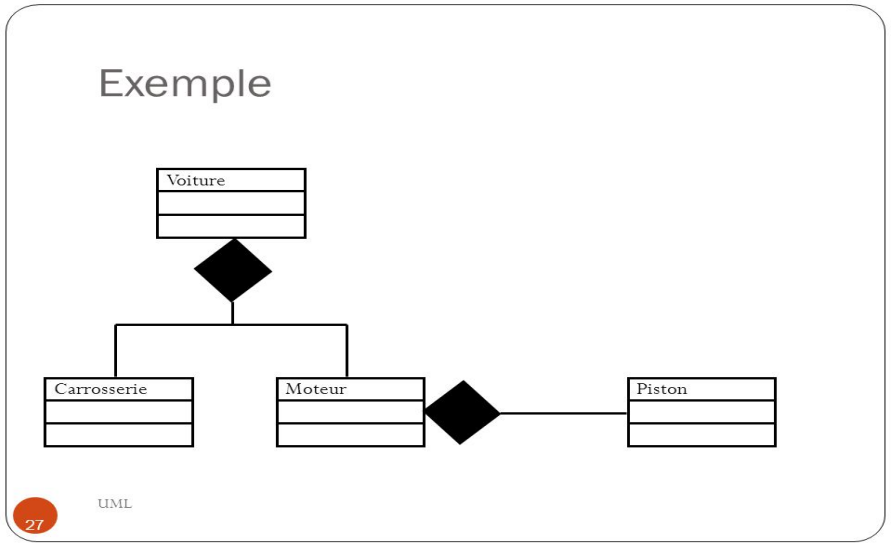
La composition

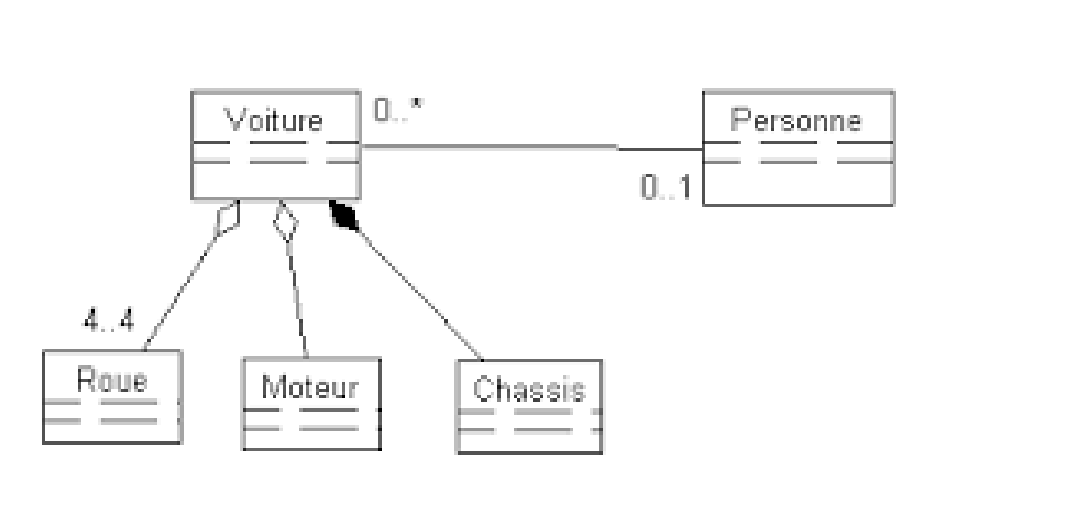
La composition est un cas particulier de l'agrégation dans laquelle la vie des composants est liée à celle des agrégats. Elle fait souvent référence à une contenance physique.

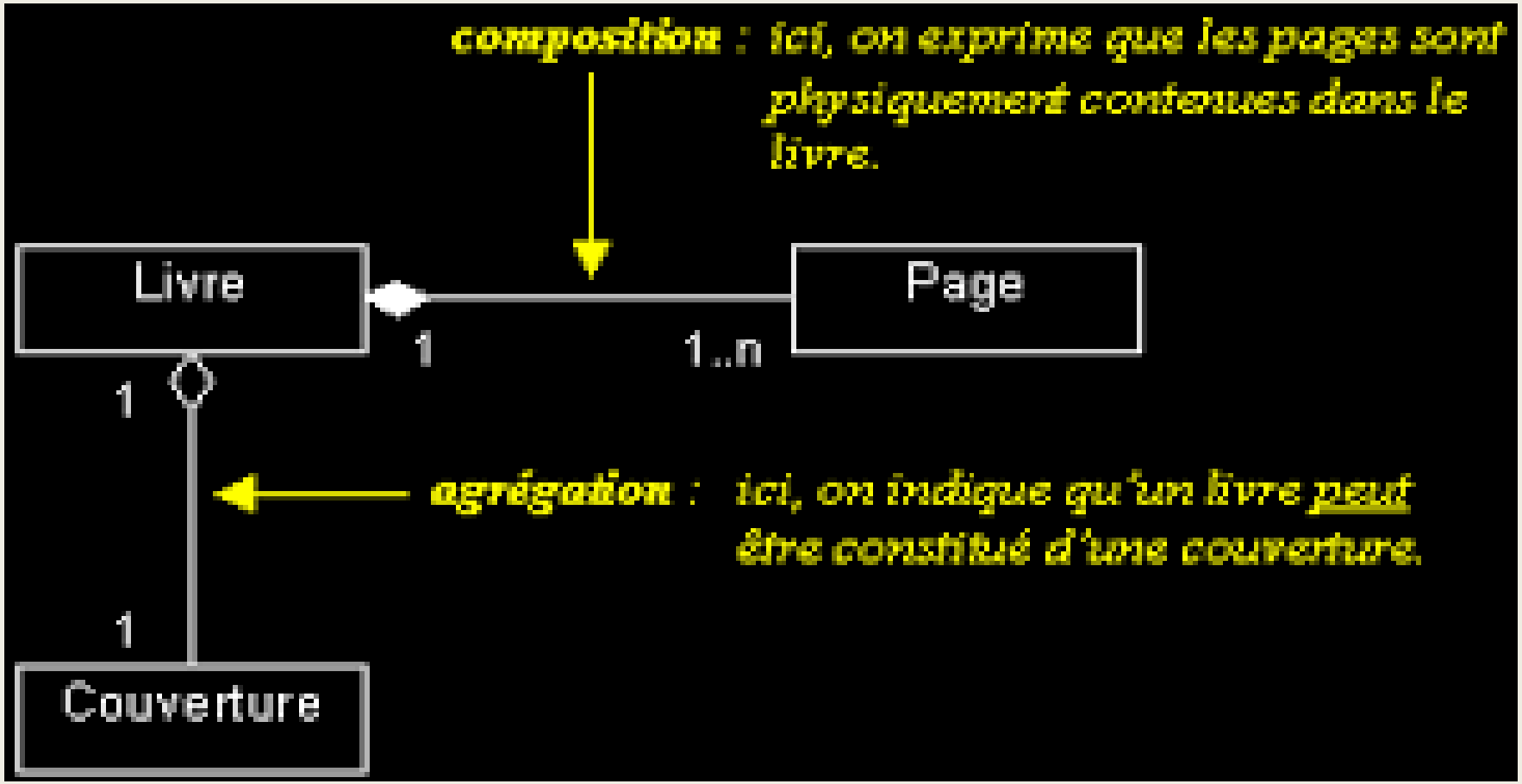
Ainsi, la destruction de l'objet composite implique la destruction de ses composants.

NB: La composition est aussi dite agrégation forte .

Graphiquement, on ajoute un losange plein





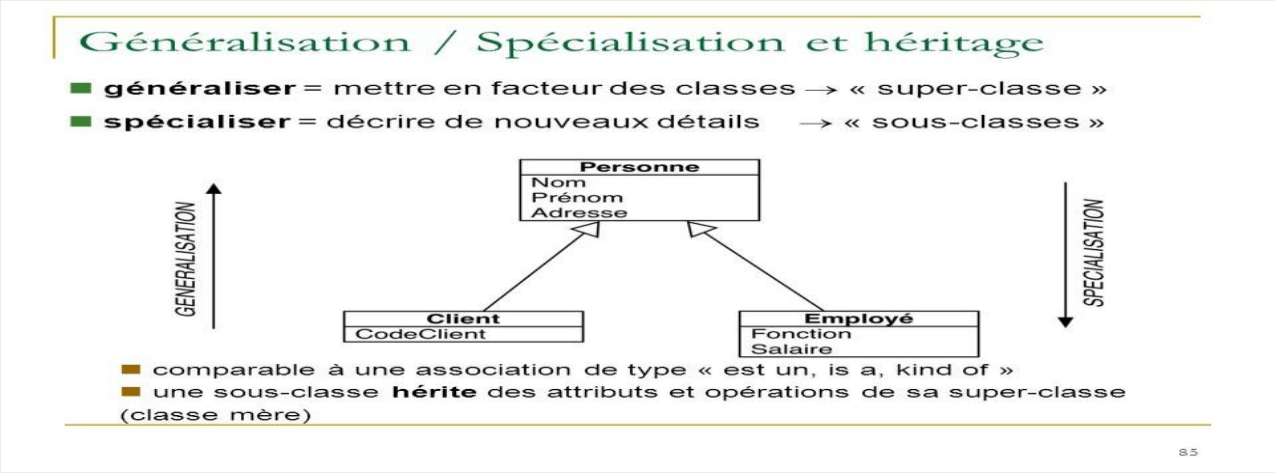

composition : ici, on exprime que les pages sont physiquement contenues dans le livre.
Livre
Page
1
1..n
1
agrégation : ici, on indique qu'un livre peut être constitué d'une couverture.
1
Couverture

La généralisation / spécialisation

Relation d'héritage, dans laquelle les objets de l'élément spécialisé (classe enfant) peuvent remplacer les objets de l'élément général (classe parent).

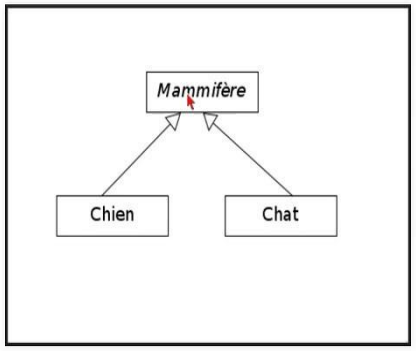
Les relations de généralisation peuvent être découvertes de 2 manières :

- la **généralisation** : il s'agit de prendre des classes existantes (déjà mises en évidences) et de créer de nouvelles classes qui regroupent leurs parties communes ; il faut aller du plus spécifique au plus général.
- La **spécialisation** : il s'agit de sélectionner des classes existantes (déjà identifiées) et d'en dériver des nouvelles classes plus spécialisées, en spécifiant simplement les différences.

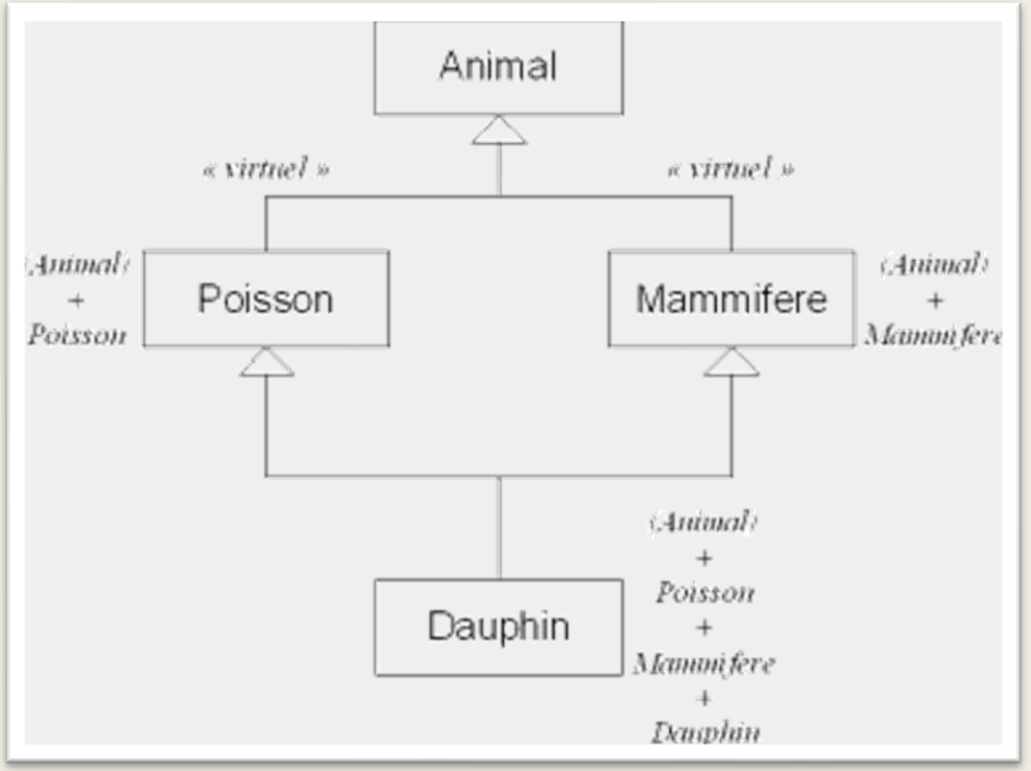


La généralisation / spécialisation / L'héritage

- Example: simple / Multiple



Héritage



Exercice:

Dans une société de transport, on voudrait gérer les bus de ramassage scolaire et les conducteurs. Un lycéen est un enfant, il est caractérisé par son nom, son âge et son sexe. Les informations qui caractérisent le conducteur sont les mêmes que pour le lycéen, avec en plus le numéro de son permis. Quant au bus, on a besoin de connaître son numéro d'immatriculation, sa date de mise en service, nombre d'années de service, et le poids total.

Un bus est composé d'une carrosserie (poids, couleur), de 6 roues (pression, diamètre), de plusieurs sièges (couleur) pour passagers, plusieurs vitres (épaisseur, poids).

Faire le diagramme de classes correspondant

La relation de dépendance

Une dépendance est une relation unidirectionnelle exprimant une dépendance sémantique entre des éléments du modèle.

Relation entre éléments du modèle ne nécessitant pas forcément un lien entre objets. Lorsque cette relation est réalisée par des liens entre objets, ces derniers sont limités dans le temps, contrairement à d'autres relations plus structurelles (cas d'une association - voir au-dessus).

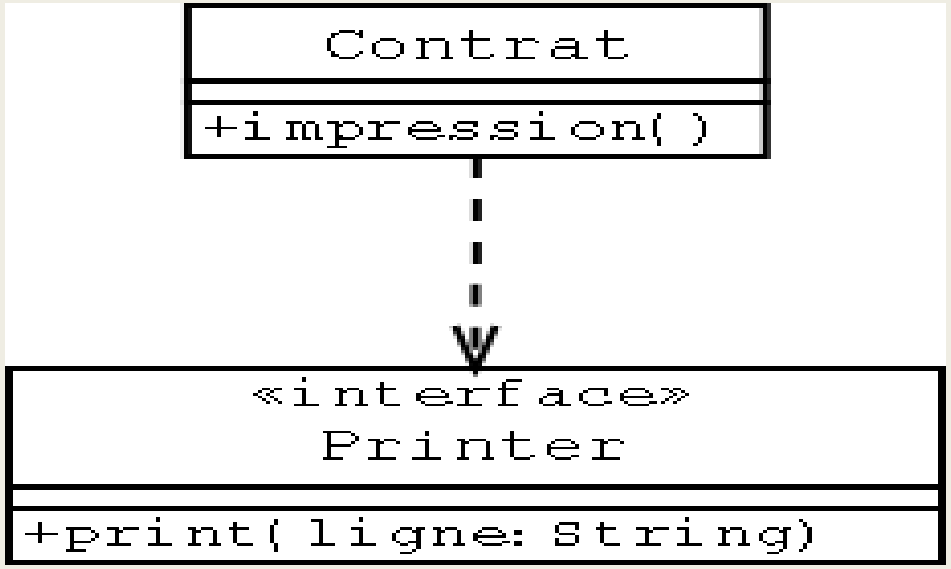
Un élément A dépend d'un élément B, lorsque A utilise des services de B.

De ce fait, tout changement dans B peut avoir des répercussions sur A.

Frome graphique: **trait discontinu** partant de la classe dépendante et pointant vers la classe proposant les services sollicités, se terminant par une **pointe de flèche ouverte**.

La relation de dépendance

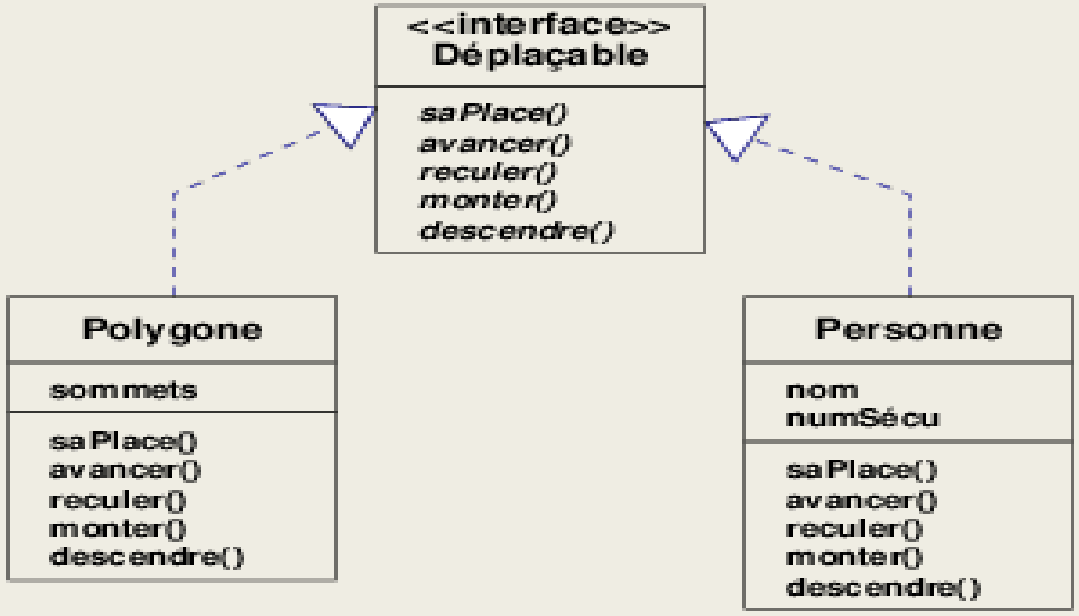
Exemple : Un contrat dispose d'un service d'impression (méthode impression), qui utilise une méthode (print), dont la spécification est déclarée par l'interface Printer.



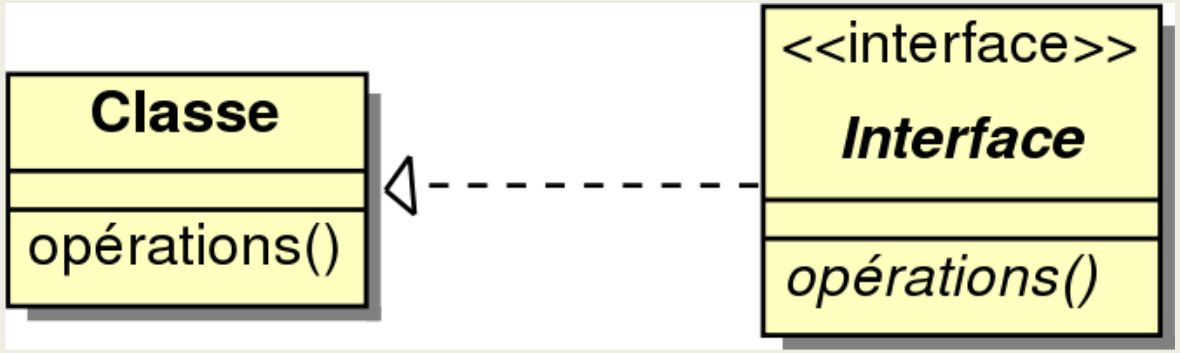
Les interfaces

- Une interface permet de décrire le comportement d'une classe, c'est-à-dire un savoir-faire sous la forme d'une liste de déclarations d'opérations sans leur définition. Une interface ne peut donner lieu à aucune instance. Toutes les opérations d'une interface sont abstraites.

- Graphiquement, cela est représenté par un trait discontinu terminé par une flèche triangulaire et le stéréotype « realize »



- Le rôle d'une interface est de regrouper un ensemble d'opérations assurant un service cohérent offert par un classeur et une classe en particulier.
- Une interface est dénie comme une classe, avec les mêmes compartiments. On ajoute le stéréotype interface avant le nom de l'interface.
- On utilise une relation de type réalisation entre une interface et une classe qui l'implémente.



Exercice - interface

```
public interface Observer {
  public void update(Observable o);
}

public class Observable {
  Collection observateurs;
  public void notify() {
    Iterator it = this.iterator();
    while (it.hasNext()) {
      ((Observer) it.next()).update(this);
    }
  }
  public void addObserver(Observer o) { observateurs.add(o); }
    ...
  }

public class Bilan extends Observable {
  void setChange() { notify(); }
  ...
}

public class UIGraphe implements Observer {
  public void update(Observable o) {
    Bilan unbilan = (Bilan) o;
    double compteResultat = unbilan.getCompteResultat();
    ...
  }
 ...
}
```

Les contraintes sur les associations

Les contraintes sont des expressions qui précisent le rôle ou la portée d'un élément de modélisation (elles permettent d'étendre ou préciser sa sémantique).

Sur une association, elles peuvent par exemple restreindre le nombre d'instances visées (ce sont alors des "expressions de navigation").

Les contraintes peuvent s'exprimer en langage naturel. Graphiquement, il s'agit d'un texte encadré d'accolade

TP – Series d’exercices

- **FAIRE LE DIAGRAMME DE CLASSE DU BLOG**

Diagramme d'objets

Un diagramme d'objets se concentre sur les attributs d'un ensemble d'objets et sur la façon dont ils interagissent les uns avec les autres.

Un diagramme d'objets peut être utilisé pour :

- *illustrer le modèle de classes en montrant un exemple qui explique le modèle ;*
- *préciser certains aspects du système en mettant en évidence des détails imperceptibles dans le diagramme de classes ;*
- *exprimer une exception en modélisant des cas particuliers ou des connaissances non généralisables qui ne sont pas modélisés dans un diagramme de classe ;*
- *prendre une image (snapshot) d'un système à un moment donné.*

Le diagramme de classes modélise des règles et le diagramme d'objets modélise des faits.

Exemple

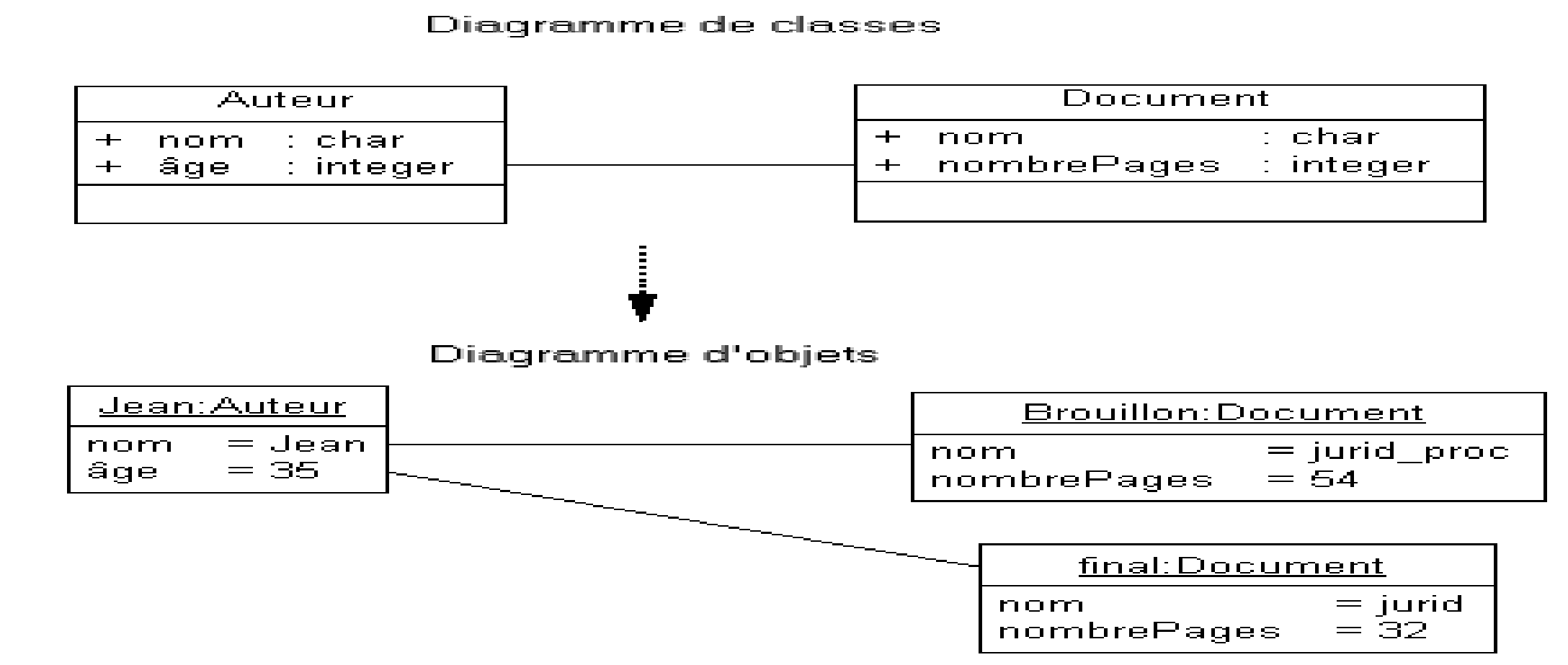


Diagramme de classes
Auteur
+ nom : char
+ âge : integer
Document
+ nom : char
+ nombrePages : integer
Diagramme d'objets
Jean:Auteur
nom = Jean
âge = 35
Brouillon:Document
nom = jurid_proc
nombrePages = 54
final:Document
nom = jurid
nombrePages = 32

TP –

- Reprendre l’exercice 1 de la serie d’exercice et
- **FAIRE LE DIAGRAMME D’OBJET DU BLOG**

TP- **FAIRE LE DIAGRAMME D'OBJET**

- Reprendre le tp sur la gestion d'une petite bibliothèque municipale
- Faire le diagramme d'objet

Diagramme de séquence

Les diagrammes de séquences permettent de décrire **COMMENT** les éléments du système interagissent entre eux et avec les acteurs :

- *Les objets au cœur d'un système interagissent en s'échangent des messages.*
- *Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).*

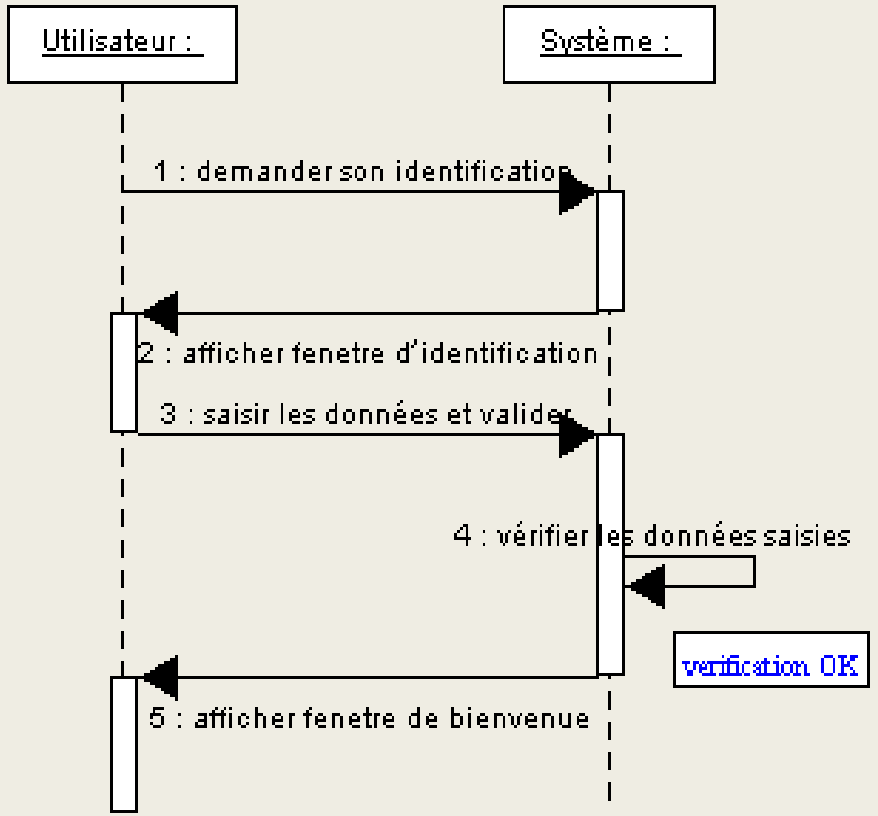


Diagramme de séquence

Le diagramme de séquences est :

- *la représentation temporelle des messages entre objets*
- *le cycle de vie d'un ensemble d'objets liés dans un cas d'utilisation*
- *la représentation des méthodes qui agissent entre objets*
- *Les messages entre objet:*
  - Type:
    - *synchrone, asynchrone, réflexif*

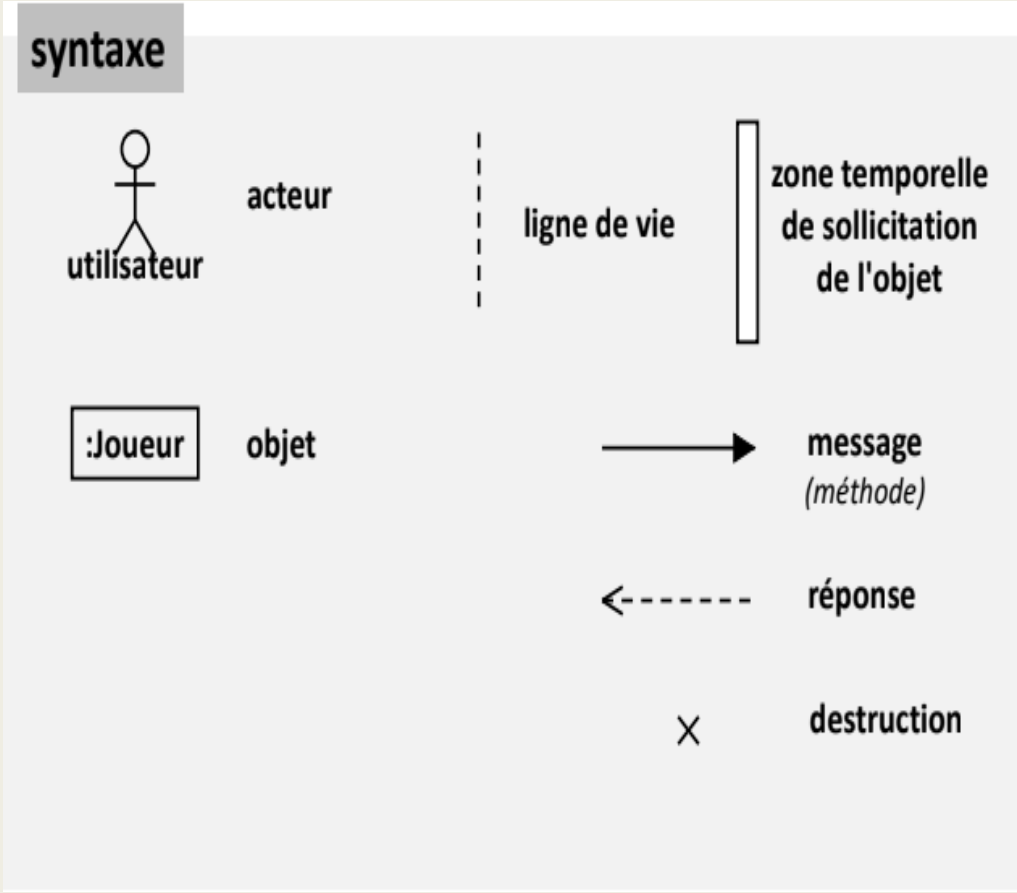
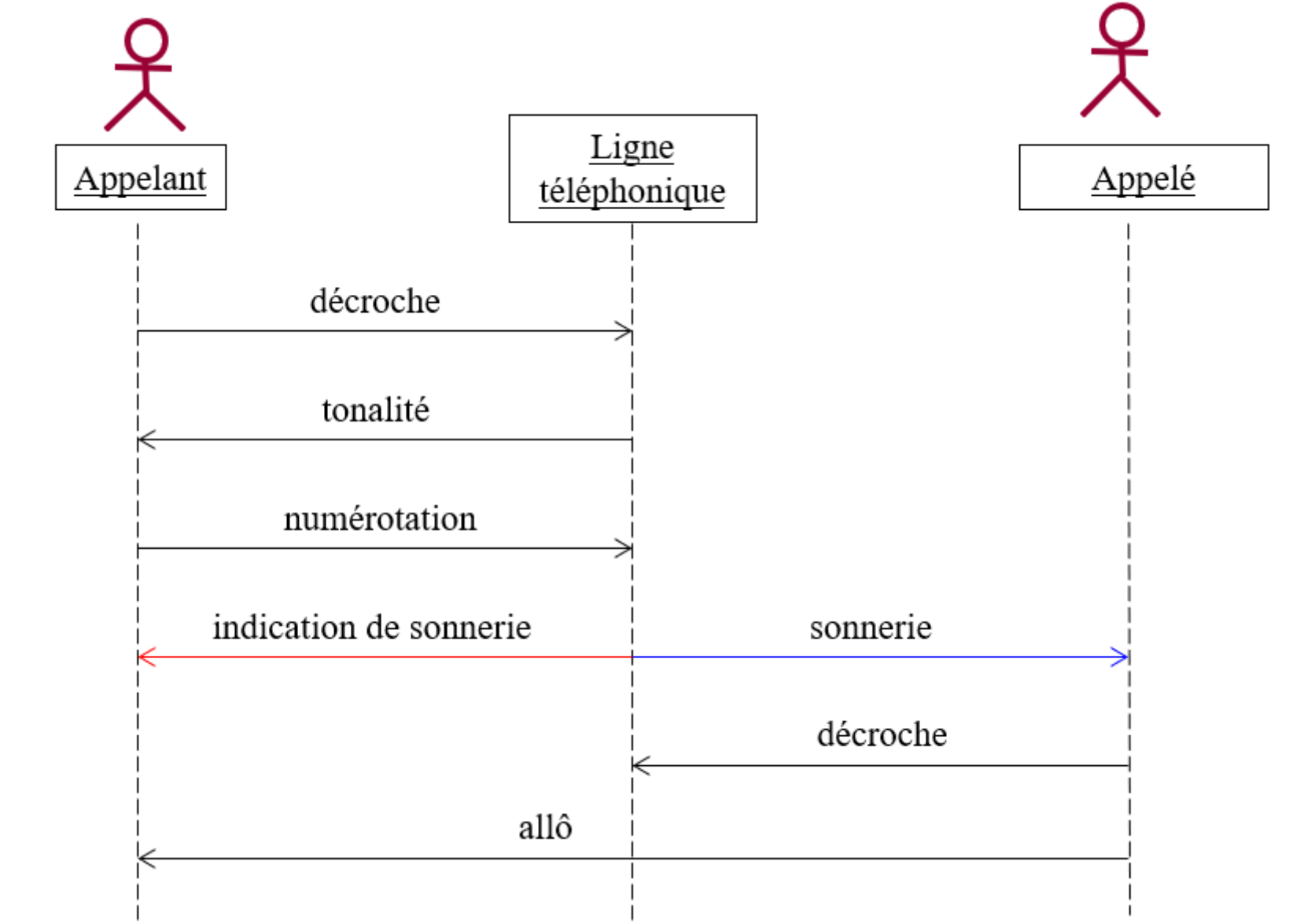


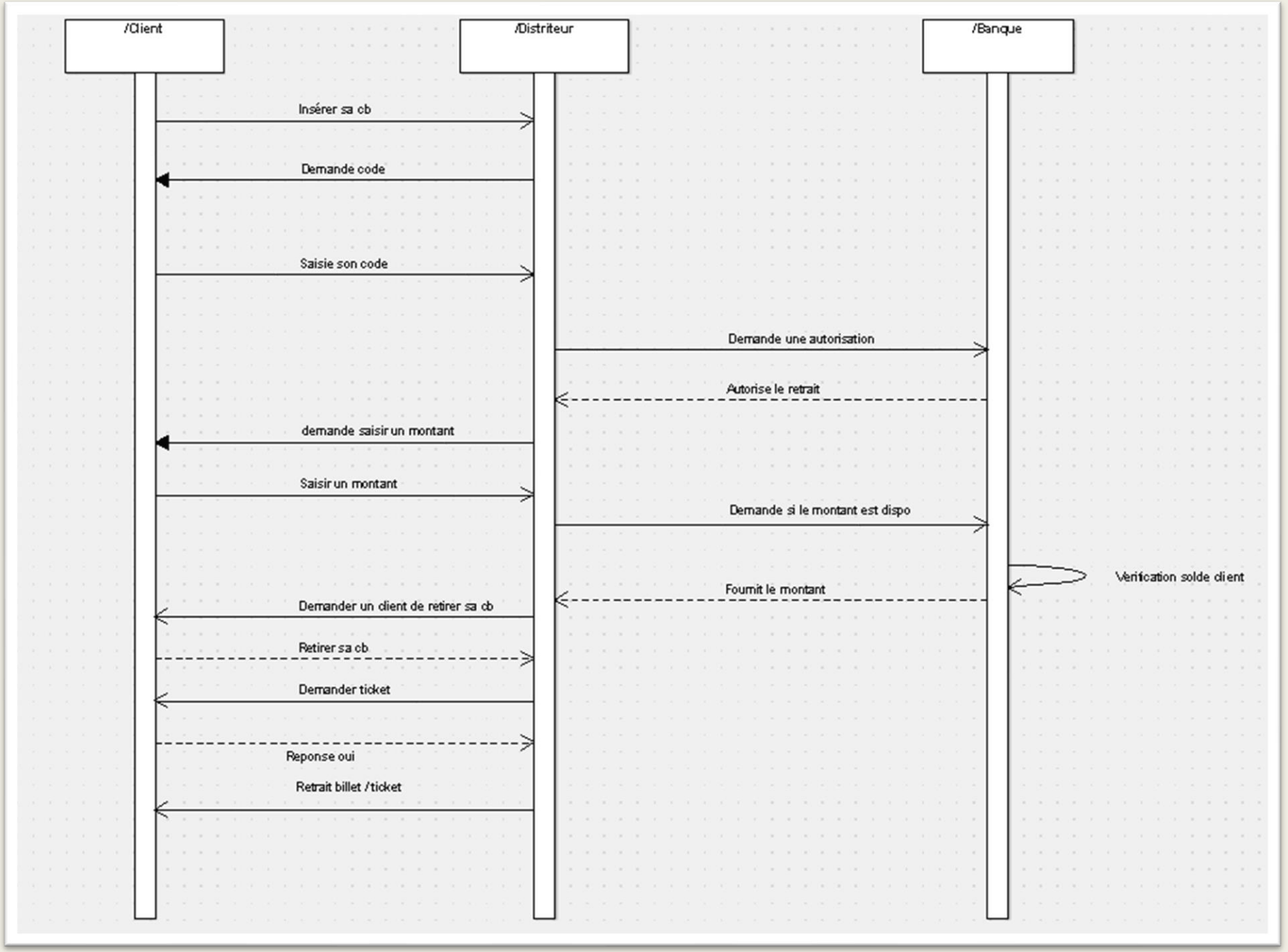
Diagramme de séquence

- Un message synchrone:
  - *l'expéditeur est bloqué pendant le traitement.*
    - Le retour d'appel est optionnel (implicite)
- Un message asynchrone:
  - *continue son exécution pendant le traitement du message*
- Un message réflexif:
  - *Message envoyé d'un objet vers lui-même*


Appelant
Ligne téléphonique
Appelé
décroche
tonalité
numérotation
indication de sonnerie
sonnerie
décroche
allô

Exercice - Retrait de billet au distributeur

- *le client introduit sa carte bancaire*
- *la machine vérifie alors la validité de la carte et demande le code au client*
- *si le code est correct, elle envoie une demande d'autorisation de prélèvement au groupement de banques. Ce dernier renvoie le solde autorisé à prélever.*
- *le distributeur propose alors plusieurs montants à prélever*
- *le client saisit le montant à retirer*
- *après contrôle du montant par rapport au solde autorisé, le distributeur demande au client s'il désire un ticket*
- *Après la réponse du client, la carte est éjectée et récupérée par le client*
- *les billets sont alors délivrés (ainsi que le ticket)*
- *le client récupère enfin les billets et son ticket*


/Client
/Distributeur
/Banque
Insérer sa cb
Demande code
Saisie son code
Demande une autorisation
Autorise le retrait
demande saisir un montant
Saisir un montant
Demande si le montant est dispo
Verification solde client
Fournit le montant
Demander un client de retirer sa cb
Retirer sa cb
Demander ticket
Reponse oui
Retrait billet /ticket

TP -

Caisse de supermarché :

- *un client arrive à la caisse avec ses articles à payer*
- *le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité si elle est supérieure à 1*
- *la caisse affiche le prix de chaque article et son libellé*
- *lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente*
- *la caisse affiche le total des achats*
- *le caissier annonce au client le montant total à payer*
- *le client choisit son mode de paiement :*
- *liquide : le caissier encaisse l'argent, la caisse indique le montant à rendre au client*
- *chèque : le caissier note le numéro de pièce d'identité du client*
- *carte de crédit : la demande d'autorisation est envoyée avant la saisie*
- *la caisse enregistre la vente et l'imprime*
- *le caissier donne le ticket de caisse au client*

Faire le diagramme de séquence

Diagramme de composants

- Un diagramme de composants a pour objectif d'illustrer la relation entre les différents composants d'un système.
- Dans le cadre de l'UML 2.0, le terme « composant » fait référence à un module de classes qui représentent des systèmes ou des sous-systèmes indépendants ayant la capacité de s'interfacer avec le reste du système.

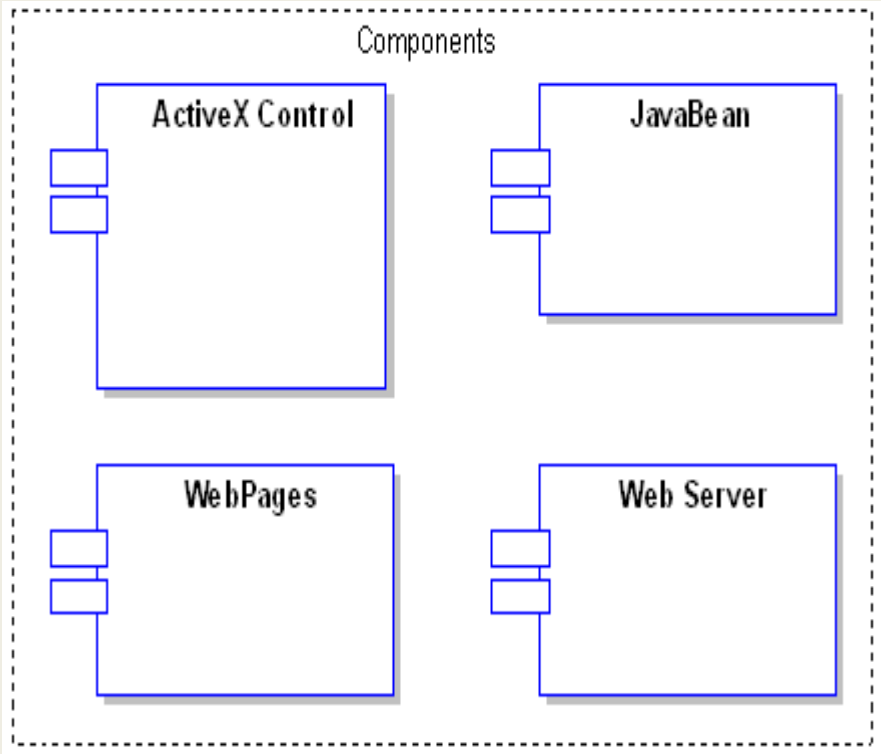
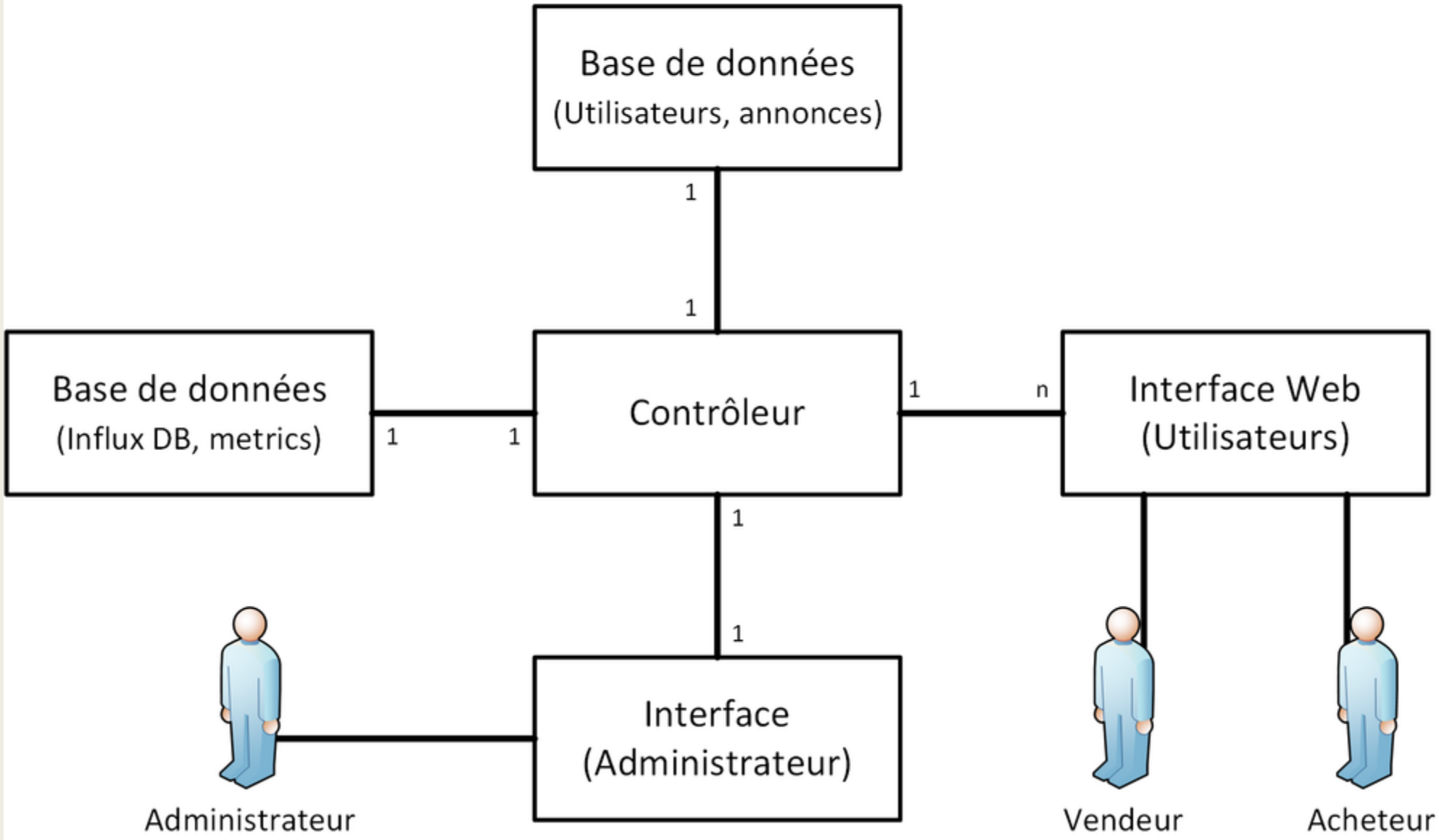


Diagramme de composants



Diagrammes de déploiement

un diagramme de déploiement fait partie de la catégorie des diagrammes structurels, car il décrit un aspect du système même.

Dans le cas présent, le diagramme de déploiement décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. On appelle artefact l'information qui est générée par le logiciel.

Les diagrammes de déploiement sont utiles dans plusieurs domaines. Vous pouvez les utiliser pour :

- *Montrer quels éléments logiciels sont déployés par quels éléments matériels.*
- *Illustrer le traitement d'exécution du point de vue matériel*
- *Visualiser la topologie du système matériel*

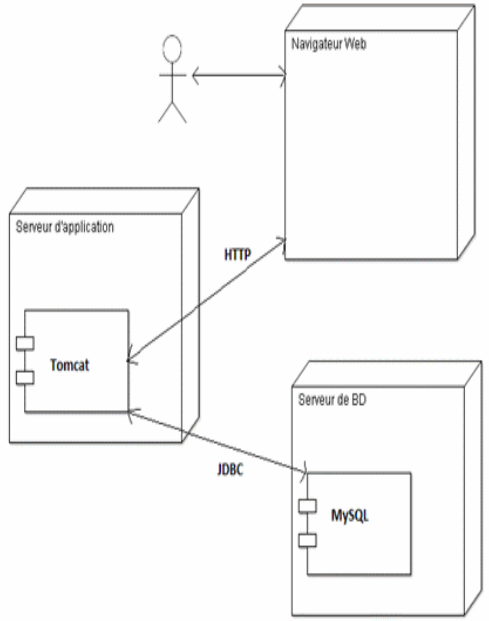


Diagramme états-transitions

Les diagrammes états-transitions ont plusieurs usages. Leurs principales applications sont les suivantes :

- Représenter des objets liés à un événement dans un système réactif
- Illustrer des cas d'utilisation dans un contexte d'entreprise
- Décrire comment un objet change d'état au cours de son existence
- Montrer le comportement global d'un automate ou le comportement d'un ensemble connexe d'automates.

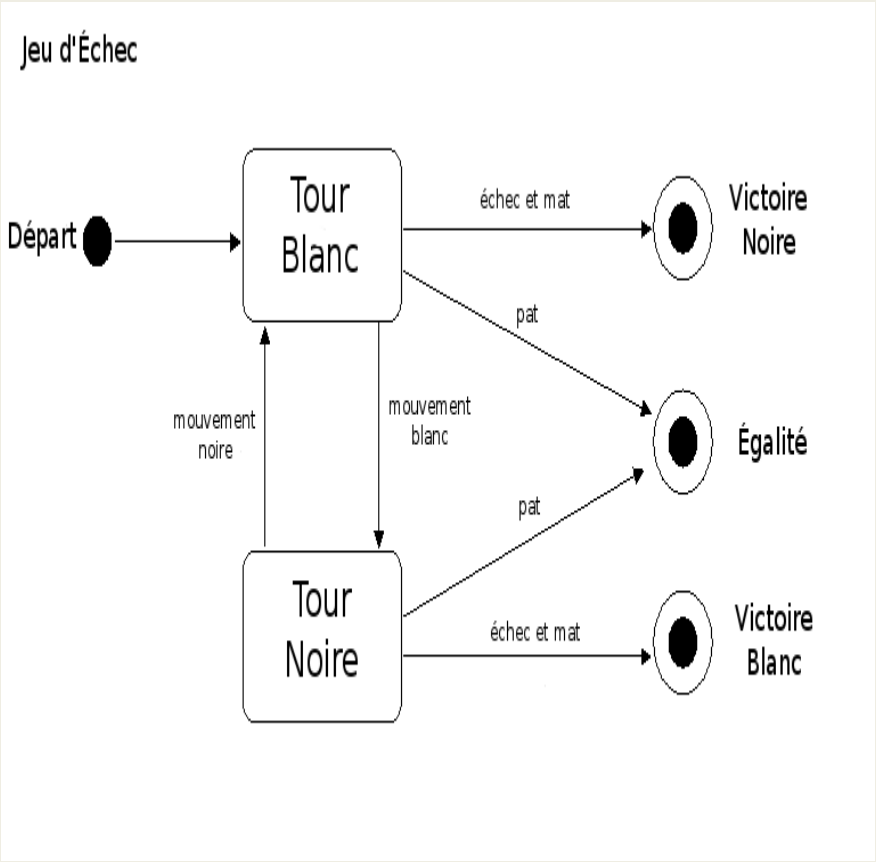


Diagramme d’activité

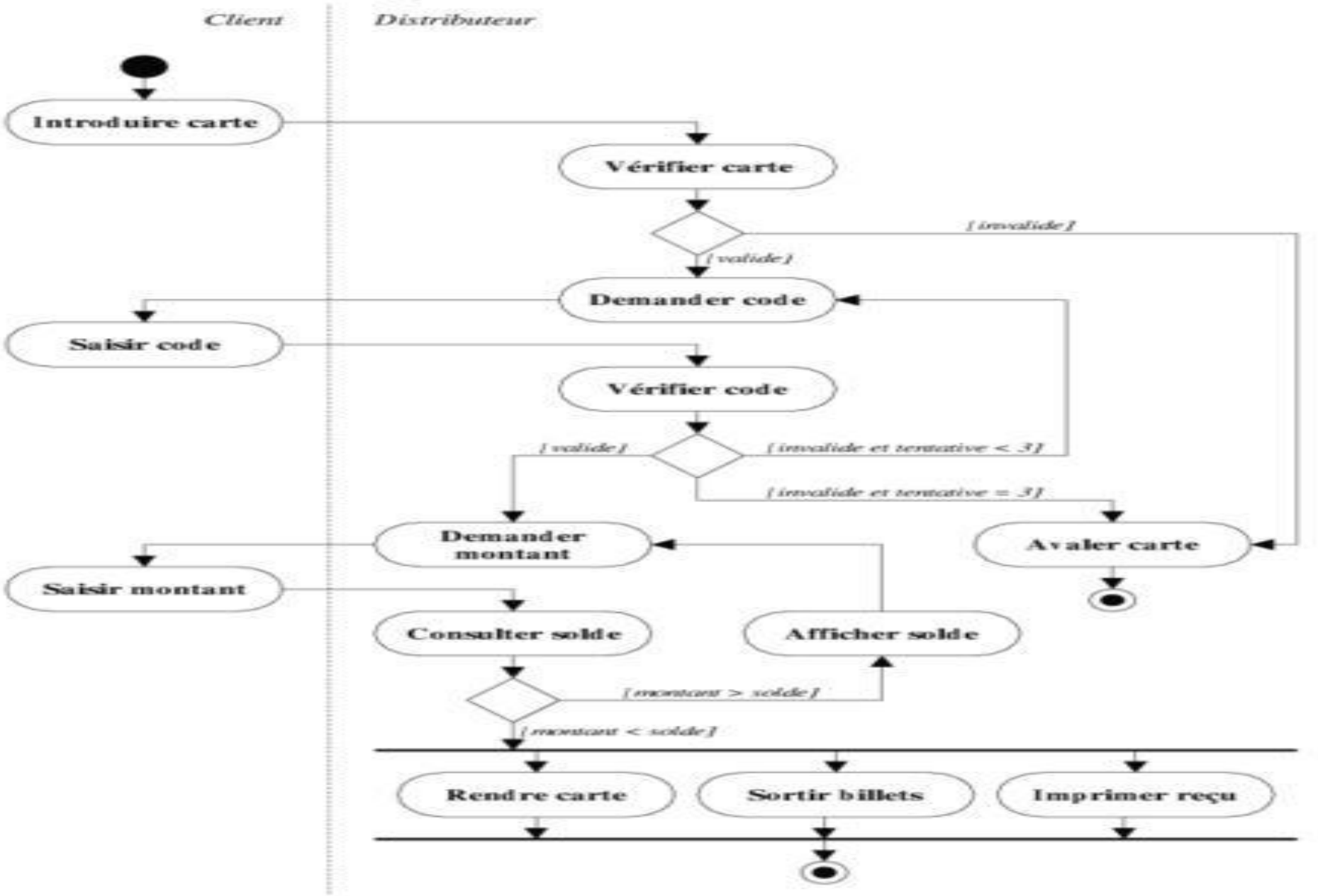
- un diagramme d’activité est utilisé pour afficher la séquence des activités. Les diagrammes d’activité représentent le flux de travail à partir d’un point de départ au point d’arrivée. Détaillant les nombreux sentiers de décision, qui existent dans la progression des événements contenus dans l’activité. Ils peuvent être utilisés à des situations de détail, où le traitement parallèle peut survenir dans l’exécution de certaines activités.
- Les diagrammes d’activités sont utiles pour la modélisation d’entreprise où ils sont utilisés pour détailler les processus impliqués dans des activités commerciales.

Diagramme d’activité - Composants

Avant de commencer à créer un diagramme d'activités, vous devez d'abord comprendre de quoi il est constitué. Voici quelques-uns des composants les plus courants d'un diagramme d'activités :

- **Action** : étape dans l'activité où les utilisateurs ou le logiciel exécutent une tâche donnée.
- **Nœud de décision :** embranchement conditionnel dans le flux, qui est représenté par un losange. Il comporte une seule entrée et au moins deux sorties.
- **Flux de contrôle :** autre nom donné aux connecteurs qui illustrent le flux entre les étapes du diagramme.
- **Nœud de départ :** élément symbolisant le début de l'activité, que l'on représente par un cercle noir.
- **Nœud de fin :** élément symbolisant l'étape finale de l'activité, que l'on représente par un cercle noir avec un contour.

Diagramme d'Activité



Exercice-

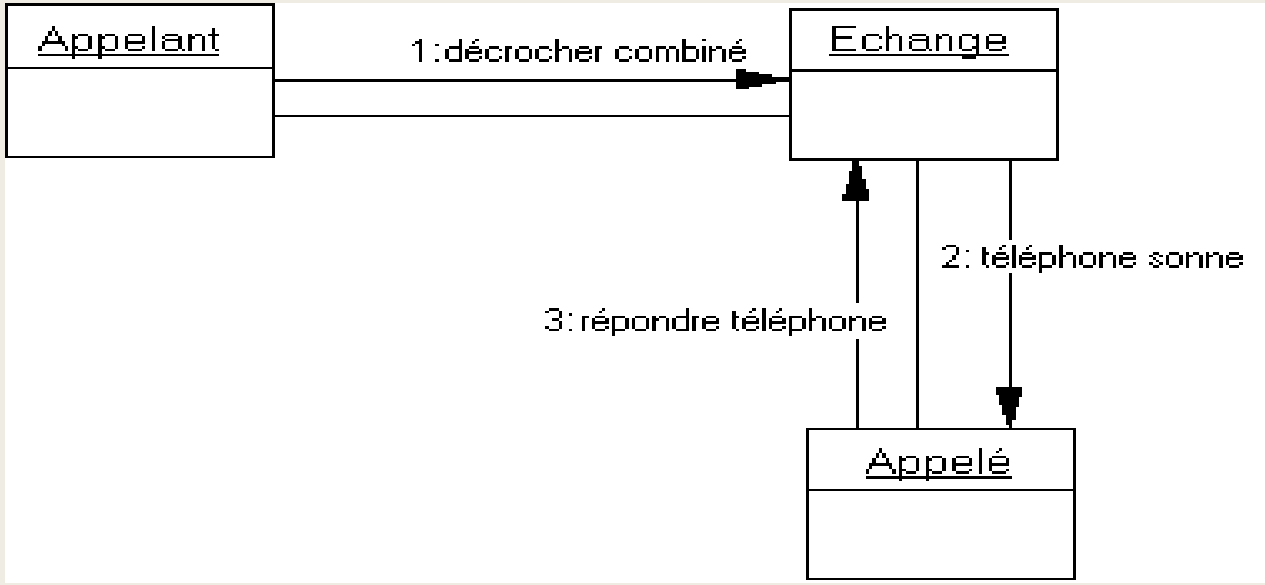
- Construire un diagramme d'activité pour modéliser le processus de commander d'un produit.
- Le processus concerne les acteurs suivants:
  - *Client: qui commande un produit et qui paie la facture*
  - *Caisse: qui encaisse l'argent du client*
  - *Vente: qui s'occupe de traiter et de facturer la commande du client*
  - *Entrepôt: qui est responsable de sortir les articles et d'expédier la commande.*

Correction exercice

Diagramme de communication

Un diagramme de communication est un diagramme d'interactions (appelé diagramme de collaboration), représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets.

En fait, le diagramme de séquence et le diagramme de communication sont deux vues différentes mais logiquement équivalentes (on peut construire l'une à partir de l'autre) d'une même chronologie.



Comparaisons entre MERISE et UML

Il existe des équivalences entre les modèles MERISE et les modèles UML.

• ***Le diagramme de cas d'utilisation*** ne montre que les acteurs, les cas d'utilisation et leur relation : il ne matérialise pas les flux et les échanges. Il ne faut donc pas le confondre avec le diagramme des flux.

• ***Le diagramme de classes et le diagramme d'objets*** sont proches des MCD et MOD. Les

différences fondamentales résident dans l'intégration des méthodes dans les classes et les objets.

De plus, le diagramme des classes ne se limite pas aux seules entités métier comme le MCD

et contient également des paquetages et des interfaces.

Comparaisons entre MERISE et UML

- ***Le diagramme de collaboration*** n'a pas d'équivalence
- ***Le diagramme d'états-transitions*** n'a pas d'équivalence dans MERISE. Il s'apparente au CVO (cycle de vie des entités et objets) dans MERISE 2
- ***Le diagramme d'activités*** présente des similitudes avec le diagramme des flux et se rapproche du MCT et MOT (MCTA et MOTA dans MERISE 2) pour fournir une vue globale de l'organisation.
- ***Le diagramme de composants*** montre la mise en œuvre physique des modèles logiques avec l'environnement de développement. Elle tient compte des problématiques de programmation, compilation, réutilisation…
- ***Le diagramme de déploiement*** est spécifique à UML.

Questions