Conception des systèmes d'informations

Dr. Rania REBAÏ BOUKHRISS
Docteur en Informatique
rania.rebai@hotmail.fr

Plan

Plan

Introduction générale

Objectifs du cours

❑ Ce cours vise à :

- Maîtriser la notion d'information
- Savoir ce qu'est un système d'information et connaître ses principales fonctions.
- Expliquer le processus de développement de SI.
- Appliquer les principes d'analyse et de conception de SI.
- Appliquer les méthodologies d'analyse et de développement des SI

Pourquoi la conception des SI ?

- La conception des SI permet spécifiquement aux étudiants des filières 1ère GL et 1ère ST de :
  - Être capable de dialoguer avec des fournisseurs de solutions logicielles
  - Intégrer un logiciel au sein d'un SI existant
  - Savoir identifier des flux d'informations
  - Pouvoir collaborer à la mise en place d'un SI

Organisation du cours

- Volume horaire
  - 21h de cours
  - 21h de travaux dirigés
- Evaluation
  - Projet
  - Devoir supervisé
  - Examen

Définition d'un système

Un système: ensemble autoréglable et interagissant avec l'environnement qui fonctionne en vue d'un objectif précis. Un système peut se décomposer en plusieurs sous-systèmes.

- Exemple
  - Une entreprise de transport
  - Banque
  - Filiale d'une société d'assurance

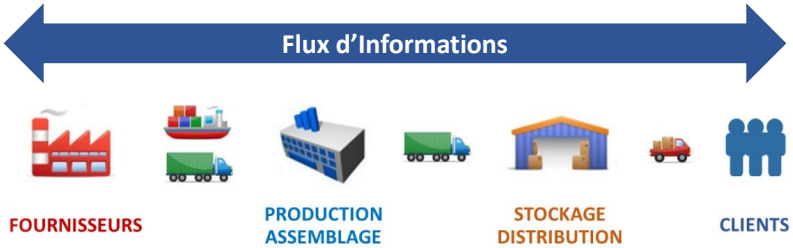
Définition d'une information

L'information: De point de vue d'un système de type entreprise, l'information est « un symbole qui fournit pour les acteurs d'une entreprise une connaissance utile à l'accomplissement de leur travail ».

- Exemple d'informations pour le système « Société Nationale des Chemins de Fers Tunisiens » (SNCFT) :
  - Gares, les voies, les trains, les wagons, les places, les classes, les horaires, les tarifs, les distances, les conducteurs, les contrôleurs, les réservations…
- Exemple d'acteurs de SNCFT :
  - Guichetiers, l'internaute, les écrans d'affichage des trains à l'arrivée et au départ…

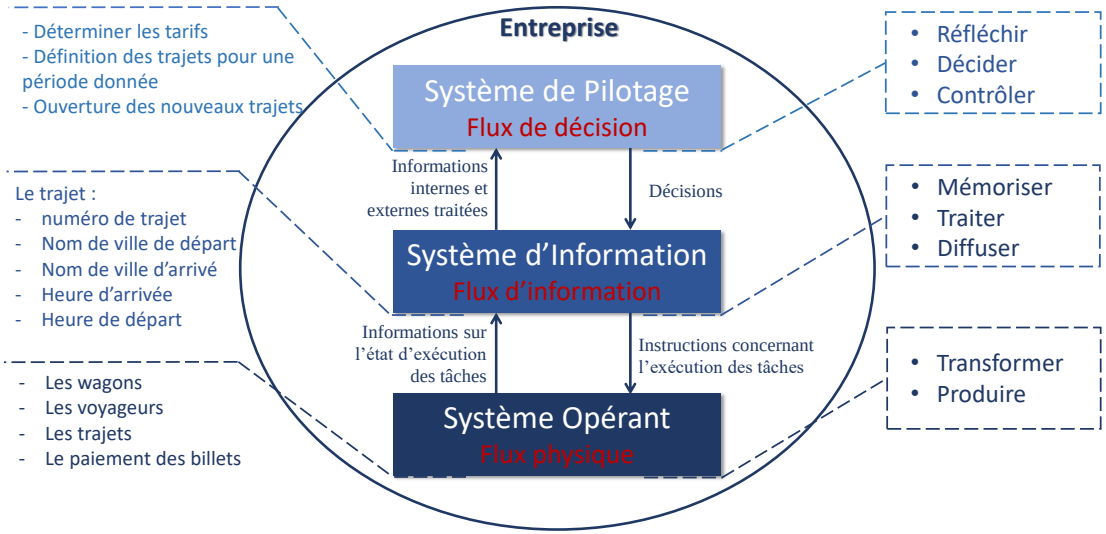
Définition d'un système d'information

Le Système d'Information (SI): ensemble organisé de ressources (matériel, logiciel, personnel, données, procédures…) permettant d'**acquérir**, de **stocker**, de **traiter**, de **communiquer** des informations de toutes formes dans une organisation.



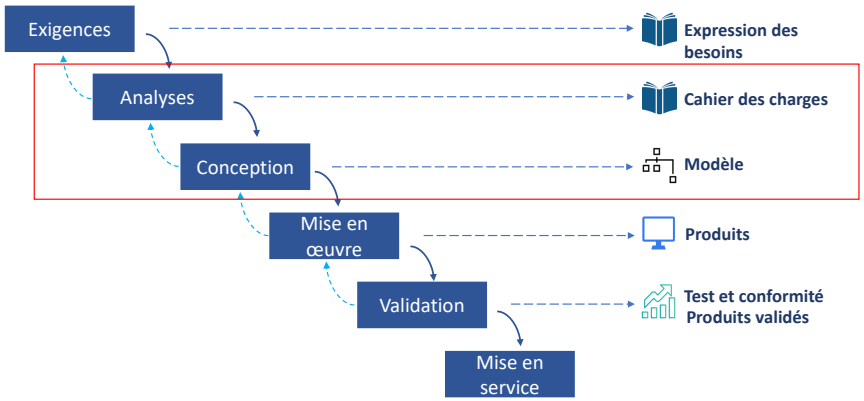
Système d'information de gestion de transport

Positionnement du SI dans l'entreprise



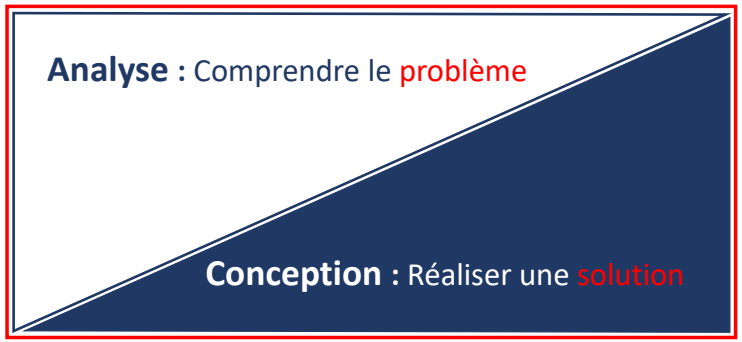
Etapes de réalisation d'un SI

❑ **Processus de développement**



Analyse Vs conception

Modélisation



La conception est souvent un compromis

La modélisation

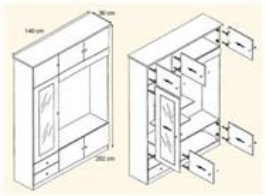
❑ Pourquoi modéliser ?

- Modéliser, c'est décrire de manière visuelle et graphique les besoins et, les solutions fonctionnelles et techniques d'un projet logiciel.

✓ l'utilisation de schémas et d'illustrations facilite la compréhension d'un sujet complexe.

➡ **C'est exactement à quoi sert la modélisation dans des projets de réalisation de logiciels !**

Exemple: Constitution d'un meuble en kit



☑ Schéma

☒ Texte

La modélisation

❑ Objectifs

- Aider à visualiser le système
- Spécifier la structure et le comportement
- Servir de plan pour la construction effective
- Permettre de documenter les choix

❑ Avantages

- Abstraction
- Compréhension : mise au point avec le client
- L'énergie déployée pour modéliser révèle les difficultés
- Les erreurs sur les modèles coûtent bien moins cher

Les différentes approches de modélisation

Approche fonctionnelle

- ➢ Première génération des méthodes d'analyse et de conception des SI (à partir des années 70)
- ➢ Approche traditionnelle utilisant des procédures et des fonctions : identification des fonction nécessaires à l'obtention du résultat
- ➢ Décomposer hiérarchiquement une application en un ensemble de sous applications
- ➢ Exemple de méthodes : SADT, Warnier, ...

Les différentes approches de modélisation

Approche Fonctionnelle : points forts

- ➢ Simplicité de processus de conception
- ➢ Affinage progressif
- ➢ Répond plus vite aux besoins ponctuels des utilisateurs (applications simples)

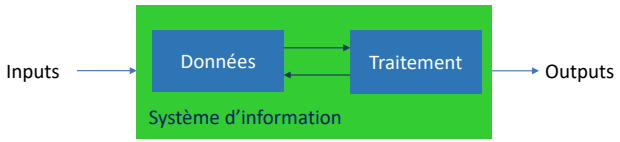
Approche Fonctionnelle : points faibles

- ➢ Concentration sur le traitement
- ➢ Redondance possible des données
- ➢ Problème de structuration des données
- ➢ Difficulté de fixer des limites pour les décompositions hiérarchiques

Les différentes approches de modélisation

Approche Systémique

- Deuxième génération des méthodes d'analyses et de conception de SI (à partir des années 80)
- Traiter le système en ensemble d'entités communicant entre elles et avec l'extérieur
- Une double approche de modélisation : des données et des traitements
- Exemple : MERISE, AXIAL, SSADM...



Les différentes approches de modélisation

Approche Systémique : points forts

- Approche globale qui prend en compte la modélisation des données et des traitements
- Introduction des niveaux d'abstraction dans le processus de conception : conceptuel, logique et physique
- Bonne adaptation à la modélisation des données et la conception des bases de données

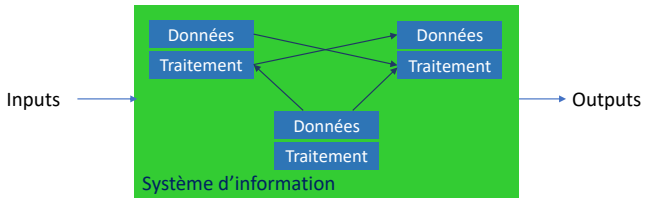
Approche Systémique : points faibles

- Redondance possible des données
- Séparation entre les données et les traitements

Les différentes approches de modélisation

Approche Orientée Objet

- Dernière génération des méthodes d'analyses et de conception de SI (à partir des années 90)
- Emergence de la philosophie Objet
- Inclure les concepts objets dans la démarche d'analyse et de conception
- Exemple : UML, O*, OMT ...



Les différentes approches de modélisation

Approche Orientée Objet : points forts

- Reflète plus finement les objets du monde réel
- Intégrer dans l'objet des données et des traitements
- Plus stable: un changement appliqué à un sous-système facile à identifier et isoler du reste du système
- Réduire la distance sémantique entre le langage des utilisateurs et le langage des concepteurs : meilleure communication entre utilisateurs et concepteurs

Approche Orientée Objet : points faibles

- Parfois moins intuitive que l'approche fonctionnelle
- Pas de fils conducteur, nécessite une expérience pour être mise en place

Unified Modelling Language (UML)

- Un besoin d'unification :

Unified …

- Pour la modélisation :

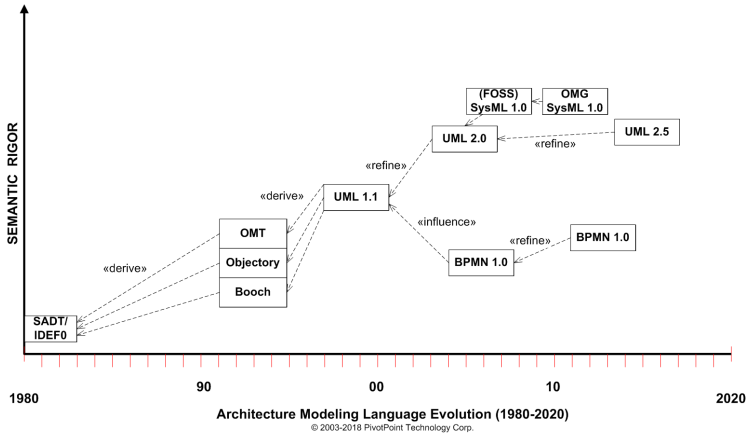
Modelling …

- Sous forme de langage :

Language …

Unified Modelling Language (UML)

Dernière version est UML 2.5.1 diffusé en 2017



Unified Modelling Language (UML)

❑ Unified Modelling Language (UML)

- Un standard des notations graphiques et du vocabulaire
- N'est pas une méthode de conception mais un ensemble de notation unifié
- UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes.
- Chaque diagramme donne une vision différente du projet à traiter.
  - Possède une structure
  - Transmettre une sémantique précise
- UML est utilisé lorsqu'on prévoit de développer des applications avec une démarche objet (développement en Java, en C++,etc).

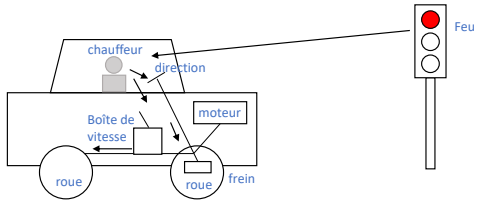
Unified Modelling Language (UML)

❑ La démarche Objet

- L'approche objet nécessite une démarche itérative et incrémentale, c'est-à-dire que le concepteur doit faire des allers-retours entre
  - ✓ les diagrammes initiaux
  - ✓ les besoins du client et des utilisateurs perçus au fur et à mesure de la conception du logiciel afin de le modifier si nécessaire.
- L'approche objet est guidée par les besoins du client.

Les concepts de l'orienté objet : l'objet

❑ **Le monde est composé d'entités qui « collaborent »**



- L'approche objet consiste à résoudre un problème en termes d'objets qui collaborent.
- Ces objets sont des abstractions des objets réel

Les concepts de l'orienté objet : l'objet

❑ L'objet est une entité (matérielle, immatérielle, …) du monde réel possédant :

- Une identité non explicitée dans le modèle : attribuée par le système (deux objets sont distincts même si toutes les valeurs de leurs attributs sont identiques)
- Un état : regroupant les valeurs de tous ses attributs à un instant donné
- Un comportement : regroupant l'ensemble d'opérations

➡ **Objet = identité + état + comportement**

Les concepts de l'orienté objet : l'objet

Objet = identité + état + comportement

Un objet **Client**

➡

Etat (informations)
nom= « Salah »
âge = 45
numCin=00985694

Comportements
ChangerNom()
AfficherAge()
AfficherCin()

Introduction à UML

❑ Le diagramme est un élément fondamental de modélisation UML

❑ Un diagramme est une représentation graphique d'objets et de relations entre eux.

❑ Il en existe plusieurs sortes de diagramme : **un seul modèle ne suffit pas !**

Introduction à UML

- UML 2.5.1 propose 14 diagrammes répartis en trois catégories :
  - Les diagrammes de structure: classes, objets, composants, déploiement, structure composite, déploiement et packages.
  - Les diagrammes de comportement: cas d'utilisation, états-transitions et activités.
  - Les diagrammes d'interactions (sous catégories des diagrammes de comportement): communication, séquence, diagramme globale d'interaction et diagramme de temps.

Introduction à UML

Spécification des besoins : ▫ Diagramme de cas d'utilisation ▫ Diagramme d'activités	**Modèles structurels :** ▫ Diagramme de classes ▫ Diagramme d'objets
Modèles dynamique : ▫ Diagrammes d'interaction (séquence/ collaboration) ▫ Diagramme d'état	**Architecture :** ▫ Composants ▫ Déploiement

Le diagramme de Cas d'utilisation

Objectifs de diagramme de Cas d'Utilisation (CU)

- La phase **d'analyse des besoins** nécessite :
  - de comprendre les besoins à couvrir par le système
  - **d'exprimer** et de **formaliser** ces **besoins**

➡ Le **moyen** offert par UML pour **représenter** ces besoins : **Diagramme de cas d'utilisation**

Objectifs de diagramme de Cas d'Utilisation (CU)

- Décrit le comportement du système du point de vue de son utilisateur
- Permet d'analyser et d'organiser les besoins des utilisateurs
- Représente un ensemble d'actions réalisées par le système et produisant un résultat observable pour un acteur (utilisateur)

Objectifs de diagramme de Cas d'Utilisation (CU)

- **Définir les besoins fonctionnels du système** : capture des fonctionnalités couvertes par le système
- **Définir le périmètre fonctionnel du système :** définir les frontières du système avec son environnement
- **Définir le dialogue entre l'utilisateur et le système :** définir comment l'utilisateur interagit avec le système
- **Servir de support de référence tout au long des phases de développement du système :** les cas d'utilisation seront consultés et référencés tout au long du processus de développement du système

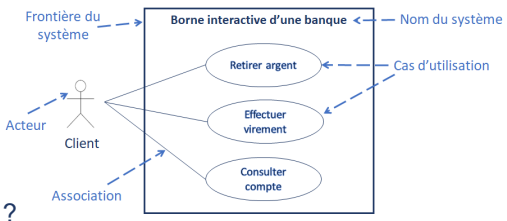
➡ Pour élaborer les cas d'utilisation, il faut se fonder sur des entretiens avec les utilisateurs

Eléments de diagramme des Cas d'Utilisation (CU)

- **Intérêt des cas d'utilisation :**
  - Recentrer l'expression des besoins sur les utilisateurs
  - Obliger les utilisateurs à définir la manière dont ils voudraient interagir avec le système

- **Un diagramme de cas d'utilisation définit :**
  - Le système
  - Les acteurs
  - Les cas d'utilisation (fonctionnalités)
  - Les liens entre acteurs et cas d'utilisation

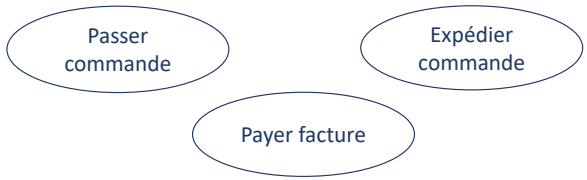
Quel acteur accède à quel cas d'utilisation ?



Cas d'Utilisation

- **Définition:**
  - Un cas d'utilisation décrit un ensemble d'actions réalisées par le système fournissant un résultat observable pour son utilisateur

- **Notation :**



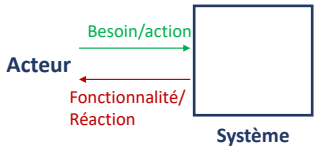
  - Dans la pratique, les noms des cas d'utilisation sont de petites phrases verbales actives qui décrivent le comportement existant dans le vocabulaire du système en cours de modélisation

Acteur

❑ **Définition :**

- Un acteur représente un rôle joué par une entité externe (une personne, un périphérique ou un autre système) qui agit sur le système étudié et interagit directement avec lui en :
  - Échangeant de l'information avec le système
  - Consultant ou modifiant l'état du système

- Un acteur est déterminé en examinant les :
  - Utilisateurs directs du système
  - Responsables de l'exploitation et/ou de la maintenance
  - Autres systèmes interagissant



Acteur

❑ Une même personne physique peut jouer le rôle de plusieurs acteurs

❑ On distingue 4 catégories d'acteurs:

- **Principal :** utilise les fonctions principales d'un système
- **Secondaire :** effectue des tâches administratives ou de maintenance
- **Matériel externe :** les dispositifs matériels : autres que les machines sur lesquelles s'exécute l'application et nécessaire aux fonctionnement du système (ex. carte magnétique)
- **Autres systèmes** avec qui le système doit interagir

Acteur

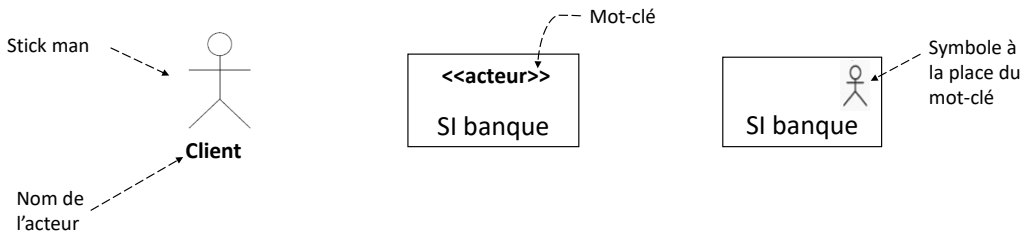
❑ **Exemple de CU :** traiter le passage d'un client à une caisse

- **Principal :** caissier
- **Secondaire :** gestionnaire des caisses
- **Matériel externe :** lecteur de carte bancaire
- **Autres systèmes :** système de gestion de stock, système d'autorisation de paiement

Acteur : représentation graphique

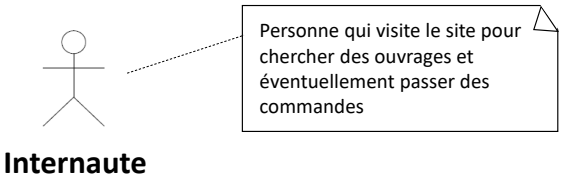
❑ Les acteurs se représentent sous la forme d'un petit personnage (stick man) ou sous la forme d'une case rectangulaire (appelé classeur) avec le mot clé << actor >>

❑ Chaque acteur porte un nom



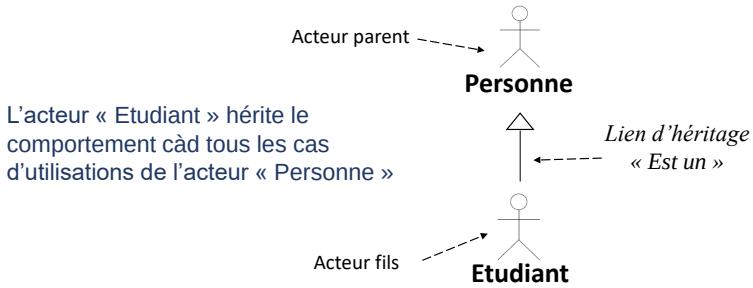
Acteur : représentation graphique

- Pour chaque acteur on doit choisir un **identificateur représentatif de son rôle**, éventuellement accompagné d'une brève description textuelle.

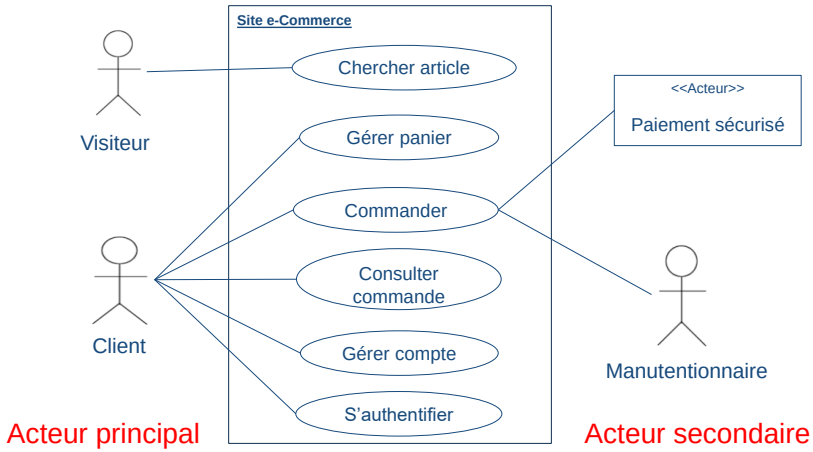


Acteur : représentation graphique

- Un acteur peut participer à des relations de généralisation avec d'autres acteurs.
  - Le seul lien qui peut être entre les acteurs est le lien d'héritage
  - L'acteur fils hérite les fonctionnalités de l'acteur parent.

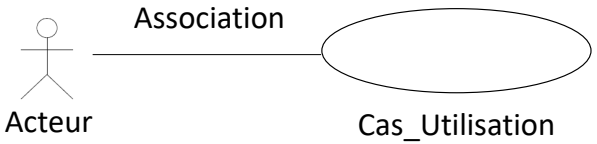


Acteur : représentation graphique



Associations

- L'interaction entre un acteur et un cas d'utilisation se représente comme une association
- Une association permet de décrire les échanges entre acteur et un cas d'utilisation.
- A chaque acteur est associé un ou plusieurs cas d'utilisations,



- Les acteurs sont connectés aux cas d'utilisation uniquement par association qui indique que l'acteur et le cas d'utilisation communiquent entre eux, chacun pouvant envoyer et recevoir des messages.

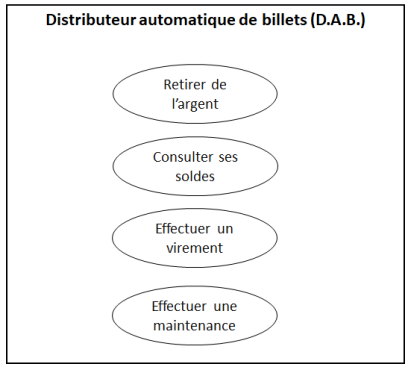
Etude de cas

❑ **Le Distributeur Automatique de Billet (DAB)**

- Un DAB permet à tout détenteur de carte bancaire de retirer de l'argent.
- Si le détenteur de carte est un client de la banque propriétaire du DAB, il peut en plus consulter les soldes de ses comptes et effectuer des virements entres ces différents comptes.
- Dans le cas du DAB, l'acteur Client banque est une spécialisation de l'acteur Porteur de carte.
- Les transactions sont sécurisées c'est-à-dire :
  - Le DAB consulte le Système d'Information de la banque (S.I. Banque) pour les opérations que désire effectuer un client de la banque (retraits, consultation soldes et virements).
  - Le DAB consulte le Système d'Autorisation Globale Carte Bancaire (Sys. Auto.) pour les retraits des porteurs de cartes non clients de la banque.
- Le DAB nécessite des opérations de maintenance tel que la recharge en billet, la récupération des cartes avalées, etc.

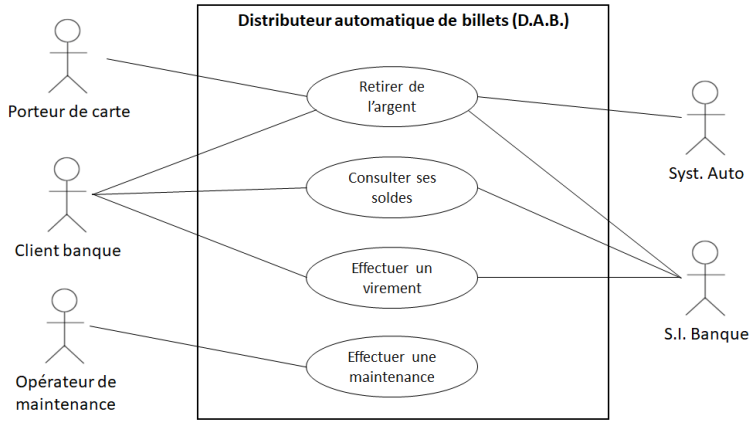
Etude de cas

❑ **Exemple de DAB :** Identification des cas d'utilisation



Etude de cas

❑ **Exemple de DAB :** Identification des acteurs et des associations



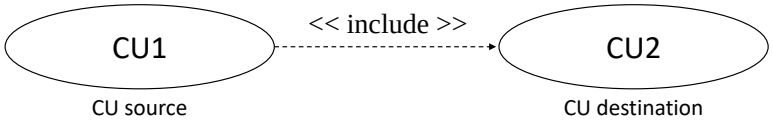
Relations entre les CU

❑ Trois relations peuvent être décrites entre cas d'utilisation:

- Une relation d'inclusion (« include »)
- Une relation d'extension (« extend »)
- Une relation de généralisation

Relations entre les CU : Relation d'inclusion

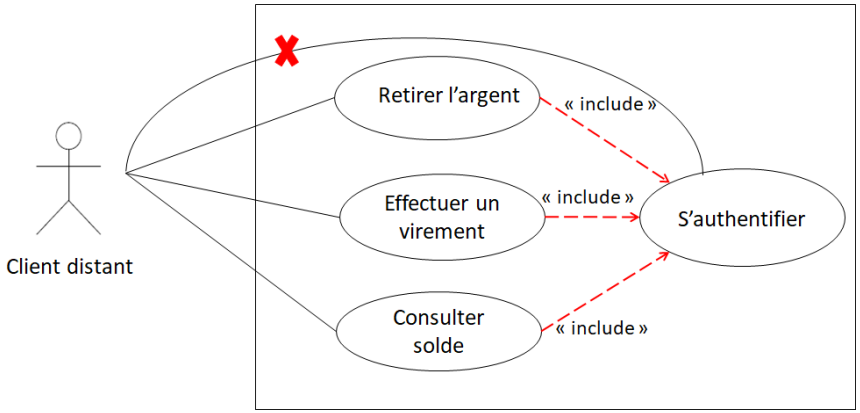
- Indique qu'un cas d'utilisation CU1 utilise systématiquement et obligatoirement un autre CU2
- Elle est représentée par une flèche pointillée étiquetée *« include »* **pointant vers le cas d'utilisation utilisé**



- La relation **include** permet d'éviter de décrire la même suite d'interactions plusieurs fois, et ce en rangeant le comportement commun à plusieurs CU dans un CU

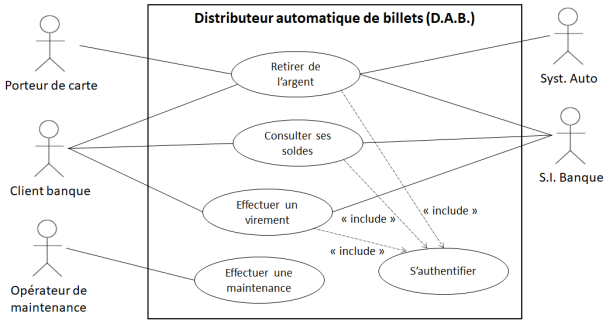
Relations entre les CU : Relation d'inclusion

- **Exemple :**



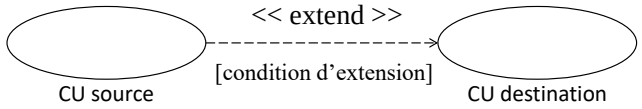
Relations entre les CU : Relation d'inclusion

- **Exemple de DAB :** Après discussion avec l'expert métier, il apparaît que l'une des sous fonctions importantes est l'authentification (systématique et commune au 3 cas d'utilisation Retirer de l'argent, Consulter ses soldes et Effectuer un virement).



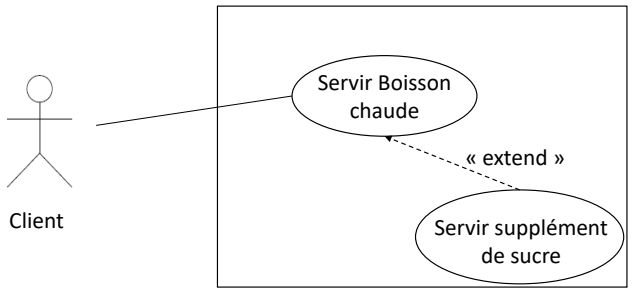
Relations entre les CU : Relation d'extension

- **La relation d'extension** enrichit un cas d'utilisation (CU destination) par un autre cas d'utilisation de sous fonction (CU source) qui est optionnel
- Cette relation est représentée par une flèche en pointillée étiquetée *« extend »* ayant comme source le CU optionnel et ciblant le CU enrichi
- L'extension peut être soumise à une condition spécifiée à côté du mot clé « extend »
  - La condition reflète le caractère optionnel de la relation d'extension (le CU source n'est exécuté que si la condition est vraie)
  - L'absence de condition dans la relation d'extension montre l'aspect obligatoire de cette relation (elle est exécuté dans tous les cas)



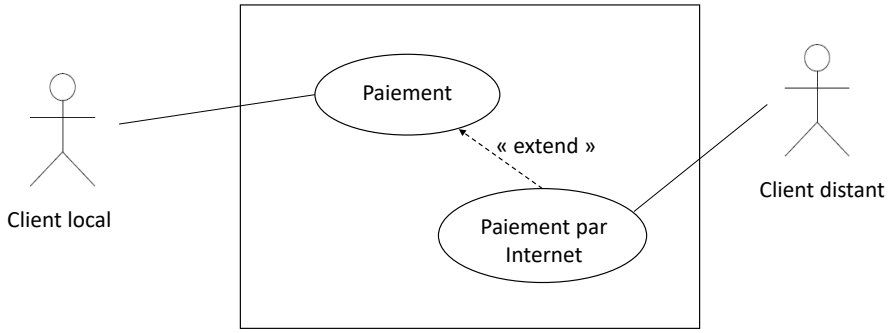
Relations entre les CU : Relation d'extension

❑ **Exemple 1 :**



Relations entre les CU : Relation d'extension

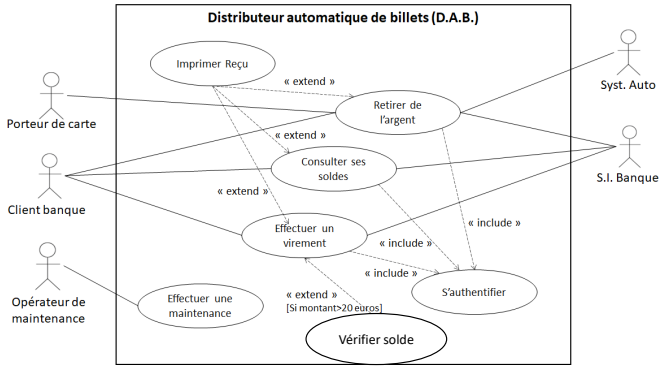
❑ **Exemple 2 :**



Relations entre les CU : Relation d'extension

❑ **Exemple de DAB :**

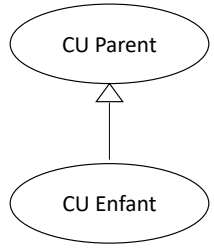
- Le DAB permet à son utilisateur d'imprimer un reçu s'il le désire.
- Une vérification du solde du compte éventuelle n'intervient que si la demande de retrait dépasse 20 euros.



Relations entre les CU : Relation de généralisation

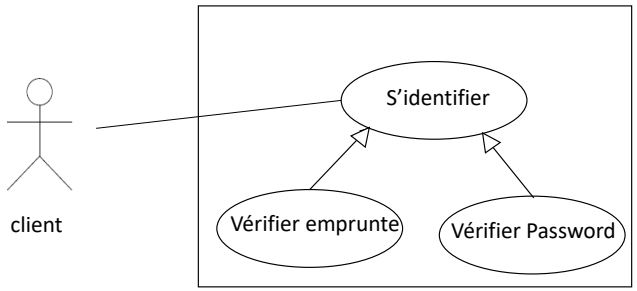
❑ Il est possible de spécialiser un cas d'utilisation par un autre cas d'utilisation
❑ La relation de généralisation est représentée par une flèche avec **une extrémité triangulaire**

- Le CU enfant hérite le comportement de CU parent
- Le CU enfant peut compléter ou remplacer le comportement du CU parent



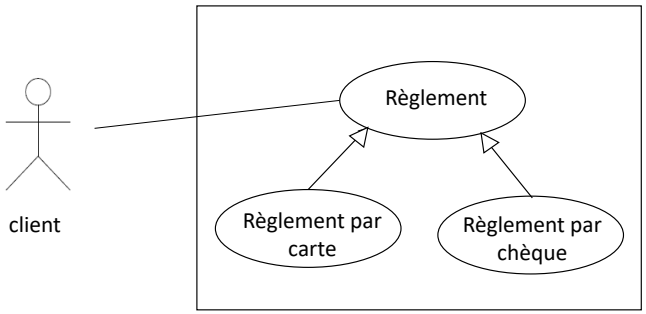
Relations entre les CU : Relation de généralisation

❑ **Exemple 1 :**



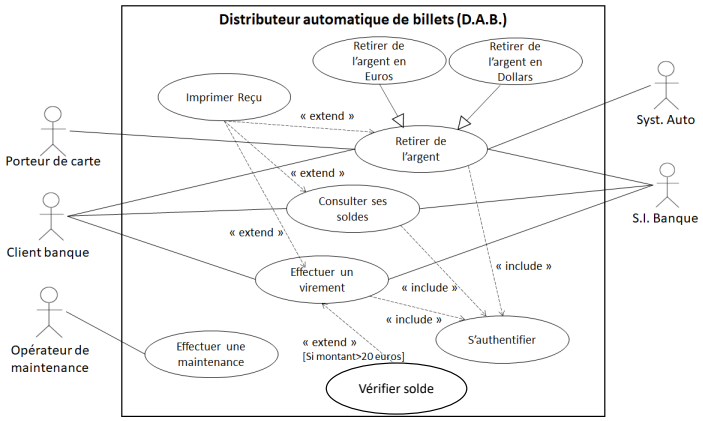
Relations entre les CU : Relation de généralisation

❑ **Exemple 2 :**



Relations entre les CU : Relation de généralisation

❑ **Exemple de DAB :** L'expert métier précise que le DAB sera situé dans une zone internationale et devra donc pouvoir fournir la somme d'argent en Dollars ou en Euros.



Exercice

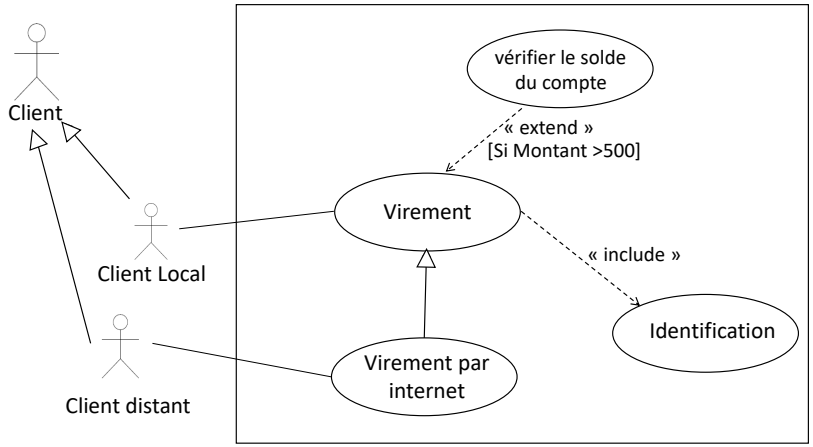
❑ **Etude de cas :** considérons les règles de gestion suivantes:

- Un client peut consulter la liste des produits
- Un client peut être distant ou local
- Un client distant effectue ses virements via Internet
- Les virements par Internet sont des cas particuliers de virements
- Tout type de virements nécessite une identification du client
- Dans le cas où le montant du virement dépasse 500, on est obligé de vérifier le solde du compte du client.

❑ **Question :** Donner le diagramme de CU correspondant à ces règles

Exercice

❑ **Solution:**



Conclusion

❑ Les cas d'utilisation sont de bons moyens pour modéliser les besoins des utilisateurs

❑ Avantages:

- Un formalisme simple
- Un bon moyen de communication: client/concepteur

Le diagramme de classe

Le diagramme de classe : introduction

❑ Diagramme central et obligatoire du modèle du SI.

❑ Le diagramme de cas d'utilisation montre le SI du point de vue des acteurs, alors que le diagramme de classes montre la structure interne du SI.

❑ Le plus riche en notations.

❑ Exprime la structure statique du système en termes de classes et de relations statique entre ces classes

Concept de classe

Une classe est une **description abstraite** d'un ensemble d'objets du domaine; elle définit **leur structure**, leur **comportement** et leurs **relations**.

- Une classe peut représenter des éléments variés comme:
  - Des éléments concrets (ex. : des voitures ou des personnes),
  - Des éléments abstraits (ex. : des achats de marchandises, des réclamations ou services),
  - Des composants d'une application (ex. : les boutons des boîtes de dialogue).

Concept de classe

- La classe est représentée par un rectangle avec attributs et opérations

NOM DE CLASSE
Attributs
Opérations

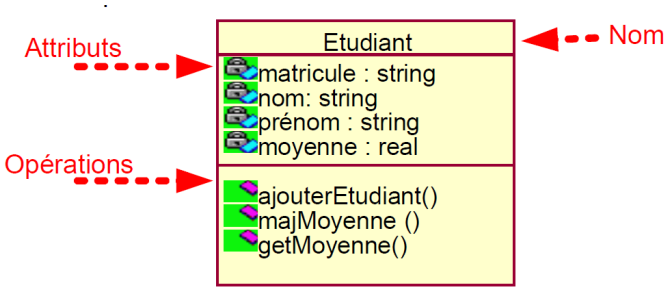
- Autres représentations

NOM DE CLASSE
Attributs

NOM DE CLASSE

➢ Les deux derniers compartiments d'une classe peuvent être supprimés lorsque leur contenu n'est pas pertinent dans le contexte du diagramme établit

Concept de classe



- Le nom de la classe est toujours **au singulier**

Exemple : Fichier, Client, Chat, …

Concept de classe : Attribut

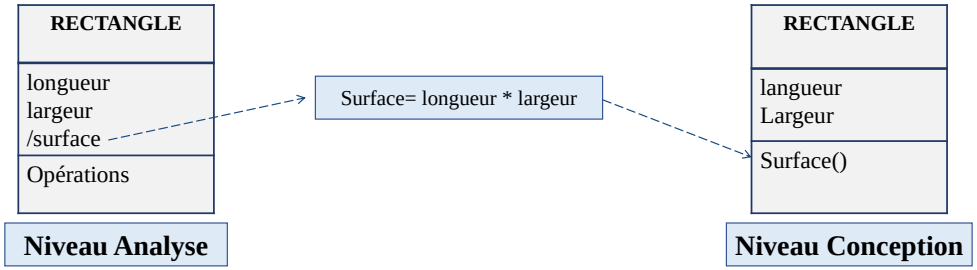
Attribut :
Un attribut est une propriété nommée de classe qui porte un ensemble de valeurs que les objets de la classe vont prendre.

Exemple :

Matricule est un attribut de la classe Etudiant

Concept de classe : Attribut

- Un attribut peut être dérivé:
  - Peut être déduit par application d'une formule sur d'autres attributs
  - Qui conduit en implémentation à une opération



➢ Les attributs dérivés sont symbolisés par l'ajout d'un « / » devant leur nom.

Concept de classe : Opération

Opération :
Les opérations modélisent le comportement de ses objets.

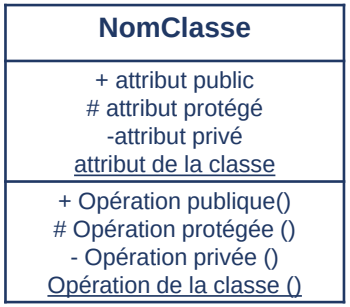
- Une opération possède une signature
- La signature est constituée du **nom** de l'opération, de **son type de retour** et **de la liste de ses paramètres (salaire() : float).**

Concept de classe : Visibilité

- Visibilité des attributs et des opérations
  - ✓Public **+** : l'élément est visible par tous
  - ✓Protégé **#** : l'élément est visible par les sous-classes
  - ✓Privé **-** : l'élément est visible à la classe seule

Concept de classe : Visibilité

- Visibilité des attributs et des opérations



- Les attributs/opérations soulignés sont visibles globalement dans toute la portée de la classe

Relations

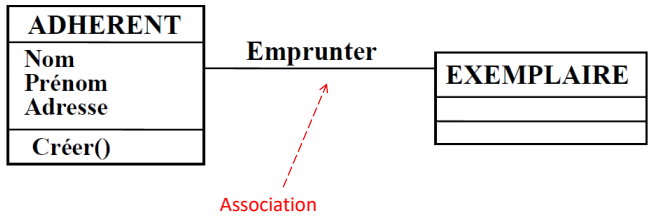
Les relations entre classes :

- ❑ Association
- ❑ Agrégation
- ❑ Composition
- ❑ Héritage

Concept de relation : les associations

❑ **Les associations:**

- Une association exprime une connexion bidirectionnelle durable entre n classes (n>=1).
- Les associations se présentent en traçant une ligne entre les classes associées



Concept de relation : les associations

❑ **Nommage des associations :**

❑ Une association peut être nommée : c'est optionnel.

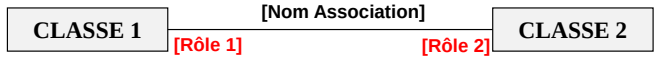
❑Les noms peuvent être en forme verbale active ou passive

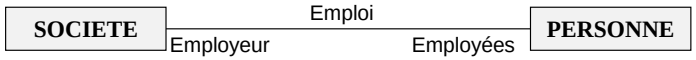
❑Le sens de lecture d'une association **peut être** précisé lorsqu'il est ambigu

Concept de relation : les associations

Les rôles :

- ❑ L'extrémité d'une association peut avoir un nom appelé rôle
- ❑ Rôle 1 : le rôle joué par Classe 1 dans l'association
- ❑ Rôle 2 : le rôle joué par Classe 2 dans l'association





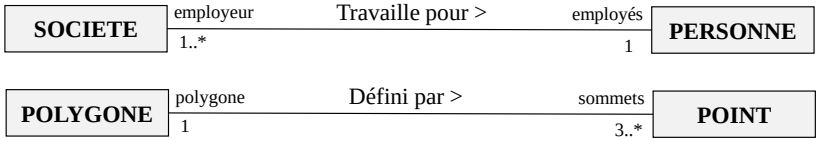
➢ L'indication des rôles est nécessaires dans les associations ambiguës

Concept de relation : les associations

❑ **Les multiplicités ou cardinalités :** précisent les nombres **min** et **max** d'objets d'une classe qui peuvent être liés à un objet de l'autre

Valeurs de cardinalité conventionnelles

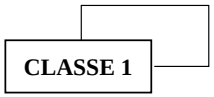
1	**Un et un seul**
0..1	Zéro ou un
N	N (entier naturel)
M..N (3..7)	De M à N (entiers naturels)
*****	De 0 à plusieurs
0..*	De 0 à plusieurs
1..*	De 1 à plusieurs



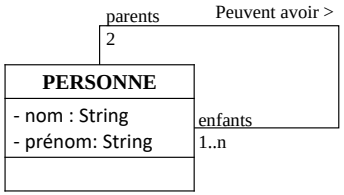
Concept de relation : les associations

❑ **Arités des associations**

- Association **réflexive** : qui relie une classe à elle-même



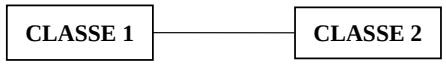
- Exemple :



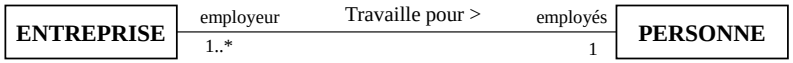
Concept de relation : les associations

❑ **Arités des associations**

- Association **binaire :** qui relie deux classes



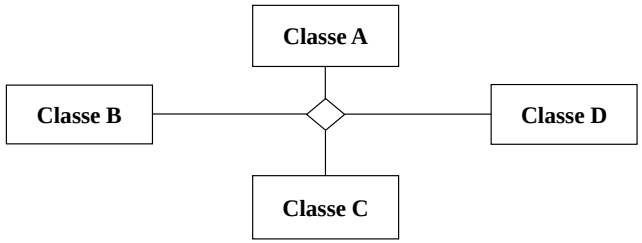
- Exemple :



Concept de relation : les associations

❑ **Arités des associations**

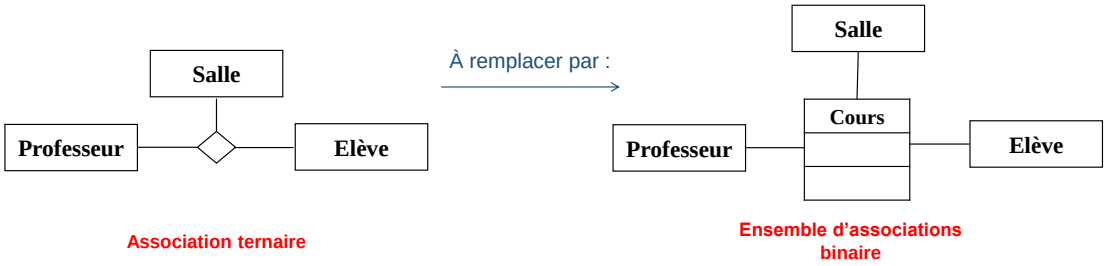
- Association **n-aire :** qui lie plus de 2 classes



Concept de relation : les associations

❑ Arités des associations

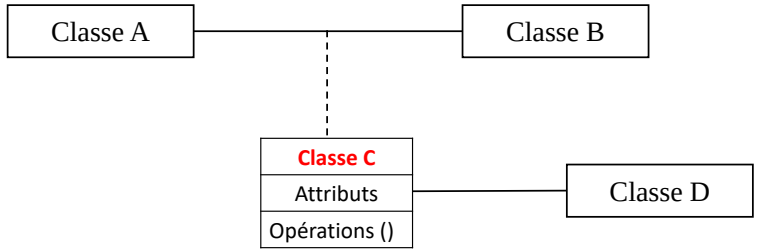
- Association **n-aire :** L'association n-aire est **imprécise**, et souvent **source d'erreur**. Elle peut être donc remplacée par un ensemble d'associations binaires



Concept de relation : les associations

❑ Une classe-association ou classe associative:

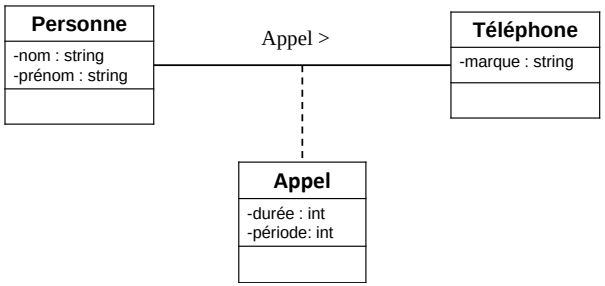
- Permet de représenter une association par une classe pour définir des attributs et/ou des opérations dans l'association
- Possède les caractéristiques d'une association et d'une classe



Concept de relation : les associations

❑ Une classe-association ou classe associative:

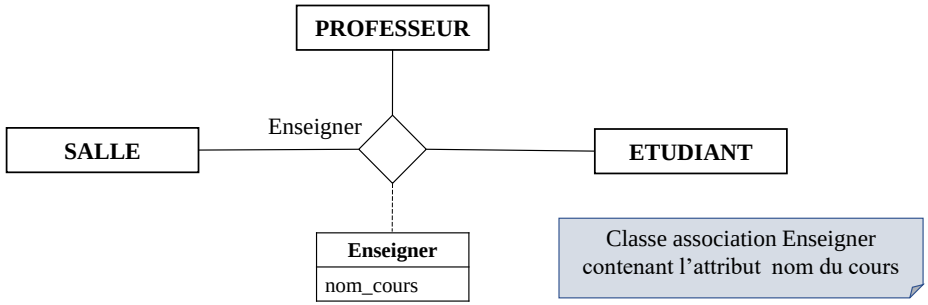
- Exemple : Quand une personne fait un appel par un téléphone, il faut savoir à quel moment il a lieu et calculer la durée de l'appel pour mesurer le tarif de l'appel. Pour cela il faut ajouter deux attributs durée et période tarifaire qui n'appartiennent ni à la classe Personne ni à la classe Téléphone. Ces deux attributs sont mis dans une nouvelle classe **Appel** qui est attachée à l'association



Concept de relation : les associations

❑ Une classe-association ou classe associative:

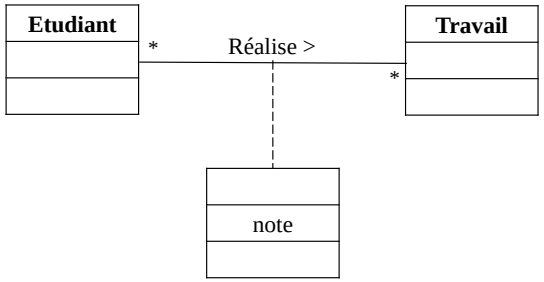
- Exemple : on désire représenter la relation enseigner un cours entre un Professeur, une Salle et des étudiants.



Concept de relation : les associations

❑ **Une classe-association ou classe associative:**

- **Association attribut**: c'est une association qui contient des attributs sans participer à des relations avec d'autres classes.
- La classe rattachée à cette association ne porte pas de nom spécifique

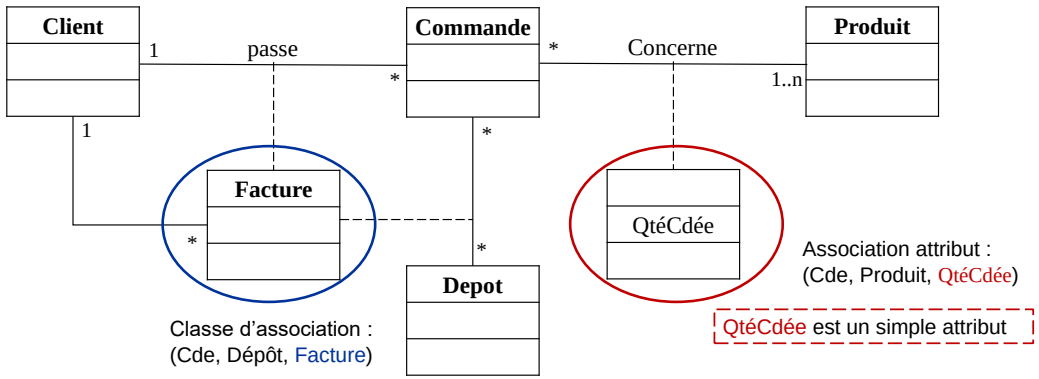


Concept de relation : les associations

❑ **Exemple:** considérons les règles de gestion suivantes:

- Un client passe une ou ++ commandes
- Une commande est passée par un seul client et concerne un ou ++ produits
- Un produit peut être commandé par ++ commandes
- Chaque commande est envoyée à plusieurs dépôts pour être satisfaite
- Chaque dépôt ayant satisfait une partie d'une commande, génère une facture correspondant à la partie satisfaite
- Toute facture générée sera envoyée au client correspondant

Concept de relation : les associations

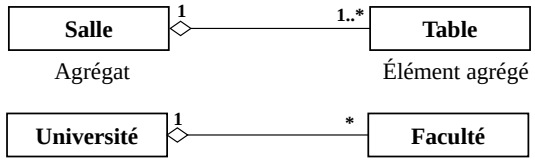


Concept de relation : Agrégation

❑ **L'agrégation**

- Une agrégation est une association qui représente une relation d'inclusion structurelle ou comportementale d'un élément (élément agrégé) dans un ensemble (agrégat).
- Elle représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité.
- Les cycles de vie de l'agrégat et de ses éléments agrégés peuvent être indépendants: la création (ou la suppression) de l'un n'implique pas celle de l'autre.
- Elle se représente par un losange vide du côté de l'agrégat (l'élément conteneur).

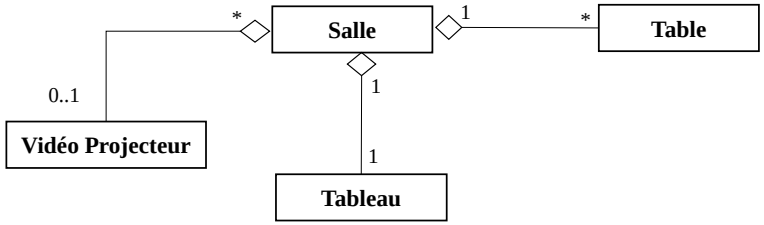
- Exemple :



Concept de relation : Agrégation

❑ L'agrégation

- Exemple :



Concept de relation : Composition

La composition

- ❑ La composition, également appelée agrégation composite, est une agrégation forte.
- ❑ Décrit une contenance structurelle entre instances .
- ❑ La destruction de l'objet composite implique la destruction de ses composants
- ❑ Une instance de la partie composant appartient toujours à au plus une instance de l'élément composite: la multiplicité du côté composite ne doit pas être > à 1 (i.e. 1 ou 0..1)
- ❑ Les Cycles de Vie (CV) des composants et du composé coïncident: si le composé est détruit (ou copié), ses composants le sont aussi.
- ❑ Elle se représente par un losange plein du côté du conteneur

Concept de relation : Exemple

❑ Exemple

- Une personne peut posséder des immeubles
- Dans un immeuble, on peut trouver des ascenseurs.
- Un immeuble est composé d'étages.
- Une personne peut posséder des comptes et une adresse

Concept de relation : Exemple

❑ Explication

- Une personne peut posséder des immeubles.
  - ➢ Le lien conceptuel: les objets ont des CV indépendants.

  ➡ Association

- Dans un immeuble, on peut trouver des ascenseurs.
  - ➢ Le lien: ensemble/élément, les CV des objets ne sont pas forcément dépendants.
  - ➢ La suppression d'un immeuble n'entraîne pas obligatoirement celle d'un ascenseur.
  - ➢ Un ascenseur ne peut être utilisé (au même temps) par plus qu'un immeuble. Mais, dans le temps, le même ascenseur peut être utilisé par différents immeubles.

  ➡ Agrégation

Concept de relation : Exemple

❑ Explication

- Un immeuble est composé d'étages.
  - ➢ Le lien: composé/composants: les CV des objets coïncident.
  - ➢ La création (ou la suppression) de l'immeuble entraine la création (ou la suppression) de ses étages.
  - ➢ Un étage ne peut pas être partagé par différents immeubles.

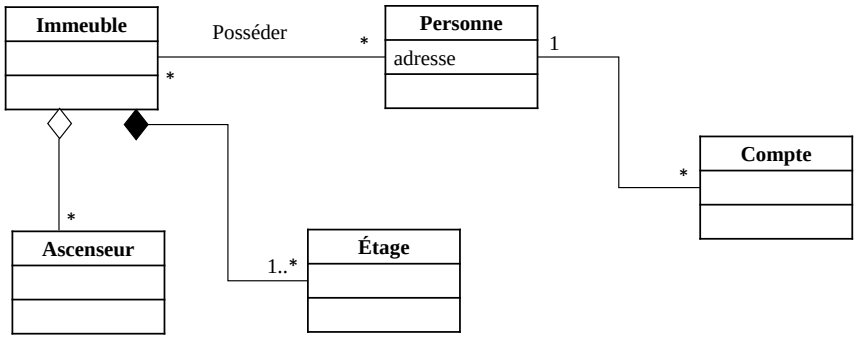
➡ Composition

- Une personne peut posséder des immeubles.
  - ➢ Le lien conceptuel: les objets ont des CV indépendants.

➡ Association

Concept de relation : Exemple

❑ Solution

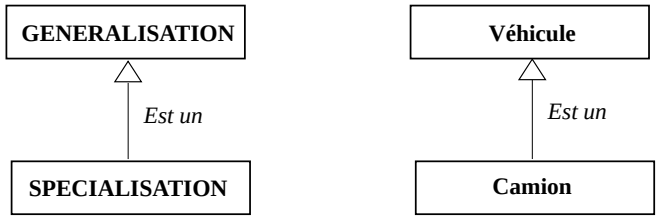


Concept de relation : Héritage

L'héritage

- ❑ Un mécanisme de transmission des caractéristiques (méthodes, attributs) d'une classe à une sous classe
- ❑ L'héritage peut être simple ou multiple
- ❑ Une sous classe hérite les attributs, les méthodes et les références de ses parents
- ❑ Une sous classe peut ajouter des attributs, des méthodes des références et redéfinir des méthodes héritées

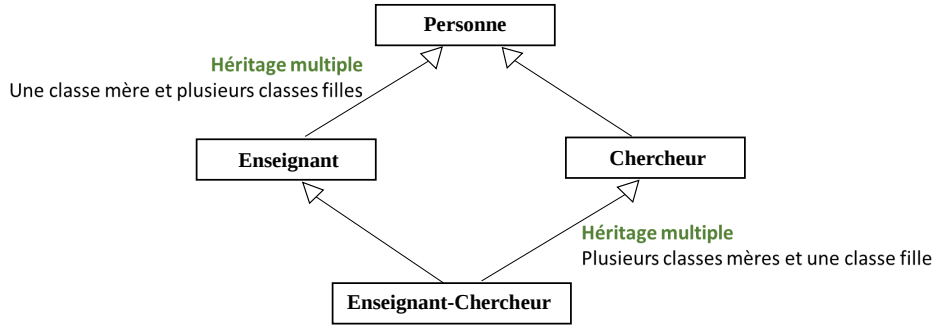
- Simple :



Concept de relation : Héritage

L'héritage

- Multiple: la spécialisation a plus qu'une généralisation



Comment construire un diagramme de classe ?

Démarche

1. Trouver les classes
2. Les associations entre elles
3. Trouver ensuite les attributs de chaque classe
4. Organiser et simplifier le diagramme en utilisant notamment le principe de généralisation

➢ Il faut réitérer le processus afin d'améliorer la qualité du modèle

Comment construire un diagramme de classe ?

Consignes (les classes)

- ❑ Après avoir établi la liste des classes possibles, éliminez les classes redondantes (celles qui modélisent des concepts similaires) ainsi que les classes superflues (celles qui ne sont pas en rapport direct avec le domaine étudié)
- ❑ Le nom des classes est important : il ne doit pas être vague

Comment construire un diagramme de classe ?

Consignes (les associations)

- ❑ Les associations correspondent souvent à des verbe mettant en relation plusieurs classes comme « pilote » ou « travaille pour »
- ❑ Les relations entre classes doivent être modélisées par une association et non par des attributs (par exemple mettre l'attribut enseignant dans la classe Matière).

QCM

1) L'agrégration est-elle un type d'association? : Oui Non
2) Une composition est-elle un type d'agrégration?: Oui Non
3) Que signifie la multiplicité 1..*? :
   1. Plusieurs incluant la possibilité d'aucun
   2. Exactement 1
   3. Au plus un
   4. Au moins un
4) Une action qu'un objet peut réaliser s'appelle :
   1. Une opération
   2. Une classe
   3. Un attribut
   4. Une formule

QCM

5) Entre une classe Vehicule et une classe Roue, quel type de relation est adéquate ?
   1. Composition
   2. Association
   3. Héritage
   4. Agrégation
6) Entre une classe Vehicule et une classe Conducteur, quel type de relation est adéquate ?
   1. Composition
   2. Association
   3. Héritage
   4. Agrégation

QCM

7) Entre une Classe Véhicule et une classe Bâteau, quel type de relation est adéquate ?
   1. Composition
   2. Association
   3. Héritage
   4. Agrégation
8) Considérons une association entre une classe Client et une classe Commande. Quelle multiplicité mettriez-vous du côté de Commande ?
   1. 0..1
   2. 0..*
   3. 1..*
   4. 1..1

QCM

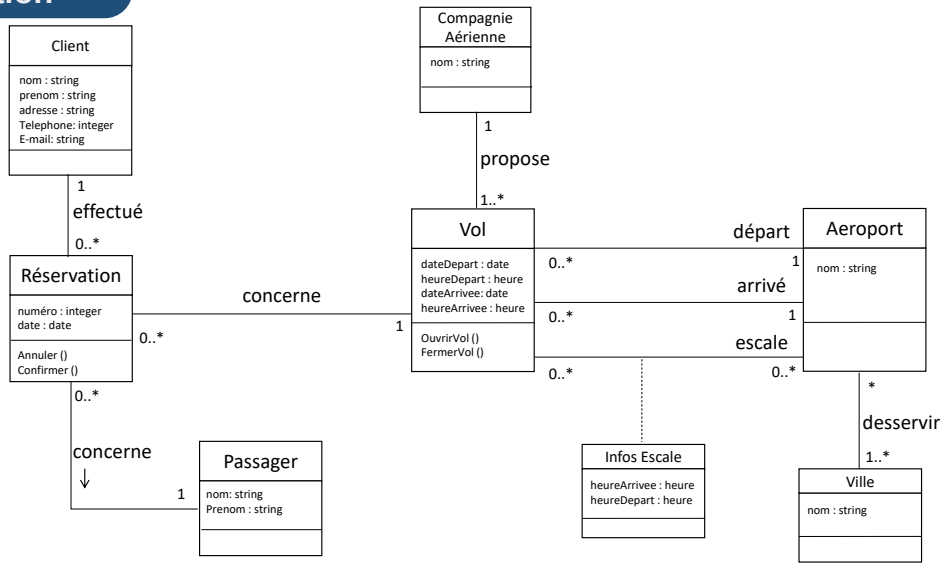
9) Considérons une association entre une classe Client et une classe Commande. Quelle multiplicité mettriez-vous du côté du Client ?
   1. 0..1
   2. 0..*
   3. 1..*
   4. 1..1
10) Grâce à une relation d'héritage, de quoi hérite la classe enfant ? (plusieurs choix possibles)
   1. des opérations
   2. des associations
   3. des relations d'héritage
   4. des attributs

Exercice 1:

❑ On souhaite gérer les réservations de vols effectués dans une agence. D'après les interviews réalisées avec les membres de l'agence, on sait que:

- Des compagnies aériennes proposent différents vols.
- Un vol est ouvert à la réservation et refermé sur ordre de la compagnie.
- Un client peut réserver un ou plusieurs vols, pour des passagers différents.
- Une réservation concerne un seul vol et un seul passager.
- Une réservation peut-être annulée ou confirmée.
- Un vol a un aéroport d'arrivée et un aéroport de départ.
- Un vol a une heure de départ et une heure d'arrivée.
- Un vol peut comporter des escales dans plusieurs aéroports.
- Une escale a une heure de départ et une heure d'arrivée
- Chaque aéroport dessert une ou plusieurs villes.

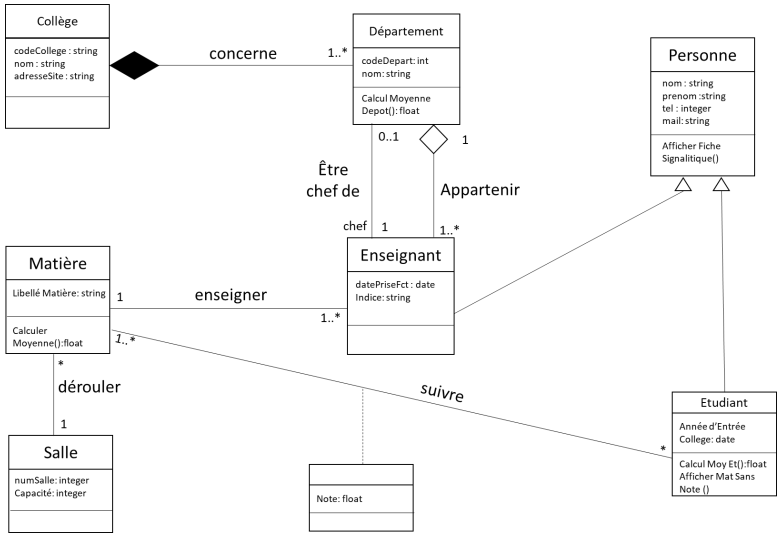
Solution



Exercice 2:

- Une académie souhaite gérer les cours dispensés dans plusieurs collèges. Pour cela, on dispose des renseignements suivants :
  - Chaque collège possède une adresse d'un site Internet
  - Chaque collège est structuré en départements, qui regroupent chacun des enseignants spécifiques. Parmi ces enseignants, l'un d'eux est responsable du département.
  - Un enseignant se définit par son nom, prénom, tél, mail, date de prise de fonction et son indice.
  - Chaque enseignant ne dispense qu'une seule matière.
  - Les étudiants suivent quant à eux plusieurs matières et reçoivent une note pour chacune d'elle.
  - Pour chaque étudiant, on veut gérer son nom, prénom, tél, mail, ainsi que son année d'entrée au collège.
  - Une matière peut être enseignée par plusieurs enseignants mais a toujours lieu dans la même salle de cours (chacune ayant un nombre de places déterminé).
  - On désire pouvoir calculer la moyenne par matière ainsi que par département
  - On veut également calculer la moyenne générale d'un élève et pouvoir afficher les matières dans lesquelles il n'a pas été noté
  - Enfin, on doit pouvoir imprimer la fiche signalétique (nom, prénom, tél, mail) d'un enseignant ou d'un élève.

Solution



Le diagramme d'état-transition

Le diagramme d'état-transition

❑ Jusqu'à présent on a étudié le système comme « **un tout** »:

- Cas d'utilisation **du système**
- Diagramme de classes **du système**

❑ Pour analyser le comportement de chaque classe du système selon des points de vue différents : Le **D**iagramme d'**E**tats–**T**ransitions (DET)

❑ Un **DET** permet d'étudier l'aspect dynamique d'une classe, compte tenu de l'importance de son comportement.

Le diagramme d'état-transition

❑ Les objets d'une classe ne sont pas figés :

- Ils peuvent évoluer et changer d'états au cours de leur cycle de vie (CV:intervalle de temps entre la création et la suppression de l'objet)

❑ Un DET est une description des changements d'états d'un objet (ou d'un composant) au cours du fonctionnement du systèmes:

- modifications des attributs ;
- exécutions des méthodes ;
- réactions à des sollicitations externes au système,…

❑ L'ensemble des DET forme une partie du modèle dynamique du SI modélisé.

Le diagramme d'état-transition

❑ Un DET d'une classe est une description des évolutions possibles de ses objets. Il donne :

- la liste des **états** que peut prendre un objet durant son CV ;
- les **événements** déclenchant les changements d'états ;
- les éventuelles **conditions** qu'il doit vérifier avant de changer d'état ;
- les **opérations** qui le font passer d'un état à un autre.

Notions de base : Etat

❑ **État d'un objet**

- L'état d'un objet est une situation donnée durant la vie de cet objet pendant laquelle :
  - Il satisfait à des conditions
  - réalise des actions
  - Ou bien, il attend un certain évènement
- Un objet passe par une succession d'états durant son existence.
- Un état se caractérise par **sa durée** et **sa stabilité**.

Notions de base : Etat

❑ Exemple:

➢Une personne peut être dans les trois états suivants

En activité **À la retraite** **Au chômage**

Notions de base : Etat

❑ Dans un DET, on distingue deux états particuliers:

- L'état **initial** correspond à l'état dans lequel se trouve l'objet avant sa **création**.
- L'état **final** correspond à la destruction de l'objet càd l'état à partir duquel l'objet ne peut plus évoluer.
- Un diagramme d'états a toujours **un et un seul état initial**. Il peut n'avoir **aucun état final** ou **plusieurs**.
- Après sa création, un objet passe par une série d'états « **normaux** »

Etat initial

Etat Final

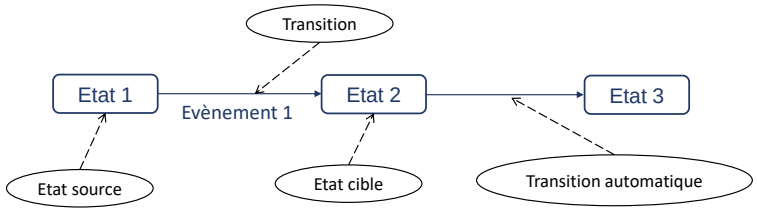
Etat intermédiaire

Etat normal

Notions de base : Transition

❑ Les transitions

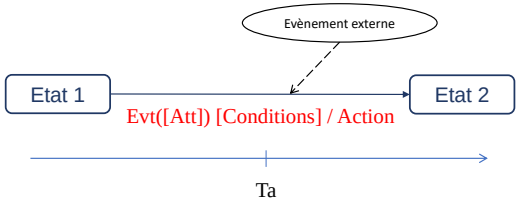
- Une transition décrit la **réponse** d'un objet lorsqu'un événement se produit provoquant le passage de l'objet d'un état (état source) dans un autre (état cible)
- La transition est représentée par une flèche orientée de l'état source vers l'état cible.
- Elle est déclenchée par un événement: c'est l'arrivée d'un événement qui conditionne la transition.
- Si aucun événement n'est spécifié, alors li s'agit d'une transition automatique



Notions de base : Transition

❑ Événements

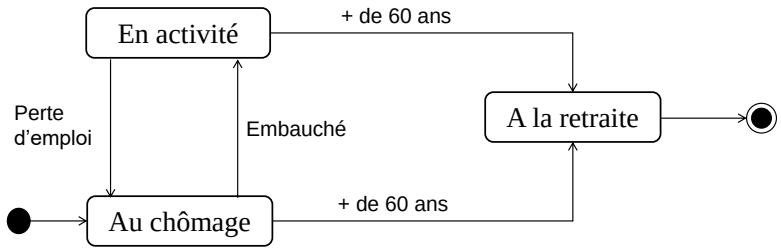
- Un événement est un stimulus qui arrive à un moment précis et pouvant déclencher une transition entre états.
- La transition n'est qu'un évènement qui s'est produit avec une condition vérifiée et déclenchant une action effectuée.



➢ À l'instant Ta, suite à l'arrivée d'un événement « **EVT** », ayant les attributs « **Att** », et sous certaines conditions « **Conditions** », l'objet passe de l'état 1 à l'état 2 par l'activation de l'action « **Action** ».

Notions de base : Transition

❑ **Exemple :**



Notions de base : Transition

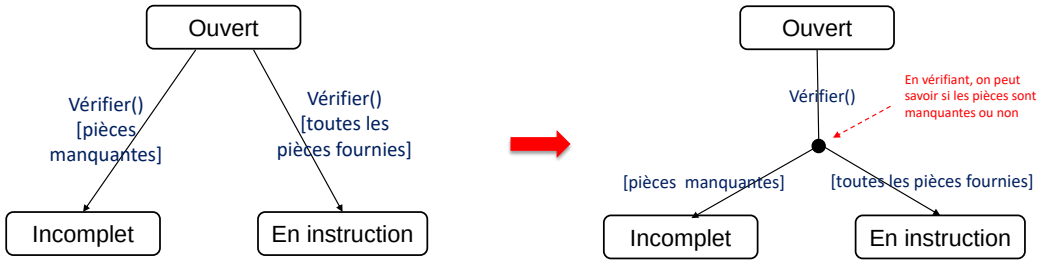
❑ **Les actions**

- Les **actions** spécifiées dans une transition sont les actions à exécuter lors du déclenchement de la transition par l'événement.
- Les actions correspondent à une opération déclarée dans la classe de l'objet destinataire.
- Une action peut comporter des appels d'opération, la création ou la destruction d'un objet, etc.

Notions de base : Transition

❑ **Les transitions composites**

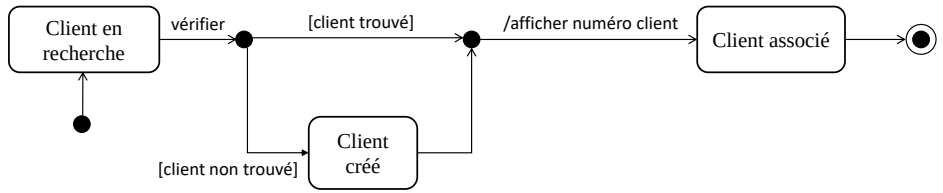
- factorisent et partagent des connexions
- Plusieurs transitions peuvent se rejoindre puis partager des actions
- Une transition peut se séparer en des connexions mutuellement exclusives



Notions de base : Transition

❑ **Les transitions composites**

- factorisent et partagent des connexions
- Plusieurs transitions peuvent se rejoindre puis partager des actions
- Une transition peut se séparer en des connexions mutuellement exclusives



Notions de base : Etat composite

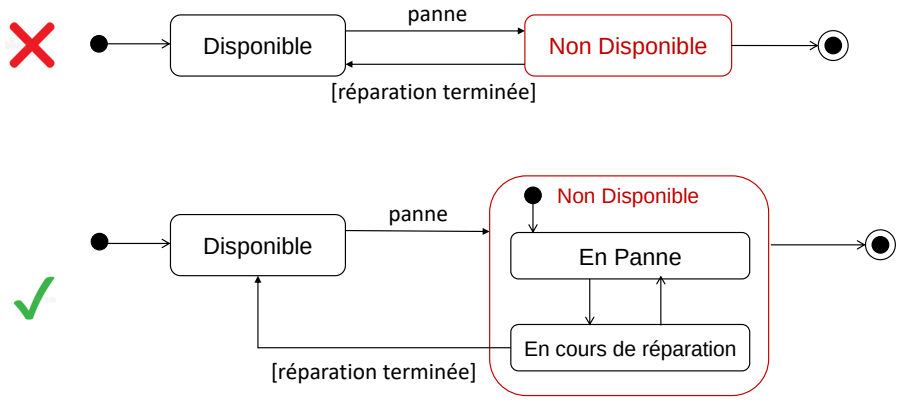
❑ Etat composite (ou composé)

- Un état composite est décomposé en sous état
- Un sous-état est un état emboîté dans un état composite.
- Les sous-états peuvent être emboîtés à n'importe quel niveau.

➡ Plus de clarté apportée aux DET

Notions de base : Etat composite

❑ Exemple:



Exercice

❑ Gestion commerciale

- Quand on gère les stocks de produits, il est nécessaire de prévoir, à tout moment, les différents états possibles de chaque stock de produit.
- Généralement, quand on crée un nouveau produit, il est automatiquement mis "en rupture de stock".
- Il ne sera disponible que s'il y a une entrée (une livraison d'une commande de ce produit).
- Pour bien gérer les approvisionnements, on se fixe une quantité minimale (QteMin) au dessous de laquelle on commande systématiquement le produit.

Exercice

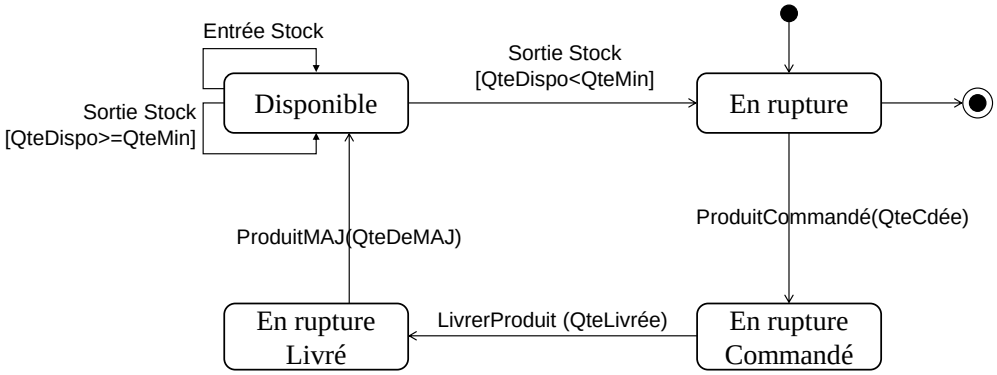
❑ Gestion commerciale

- QteMin servira à comparer la quantité disponible (QteDispo) du produit et à lancer la commande du produit si elle est inférieure à QteDispo.
- Une fois commandé, on doit attendre la livraison du produit pour qu'il redevient disponible.
- Quand un produit est disponible, toute opération d'ajout ne le fait pas changer d'état.

➢ Donner le diagramme d'états-transitions de l'objet Produit

Solution

❑ DET d'un Produit



Conclusion

- ❑ Dans le diagramme de cas d'utilisation et le diagramme de classes, nous avons étudié le système comme « un tout ».
- ❑ Le diagramme d'états – transitions (DET) permet d'analyser le comportement de chaque classe du système selon des points de vue différents.
- ❑ Un DET permet d'étudier l'aspect dynamique d'une classe, compte tenu de l'importance de son comportement.

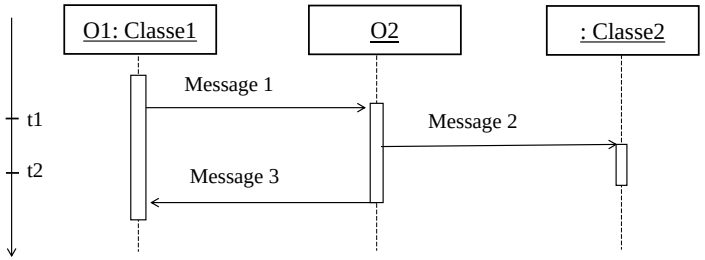
Le diagramme de Séquence

Définition

- ❑ Le diagramme de séquence permet de représenter des **échanges** entre les différents **objets** et **acteurs** du système en fonction du **temps.**
- ❑ Décrit la réalisation des cas d'utilisation sur le système décrit par le diagramme de classes.
- ❑ Se concentre sur la séquence des messages envoyés entre les objets (c.à.d. quand et comment les messages sont envoyés et reçus par les objets qui participent à une interaction)

Concepts de diagrammes de séquences

❑ **Représentation générale :**



- Chaque objet est matérialisé par un rectangle et une barre verticale appelée **ligne de vie**.
- L'objet qui débute l'interaction se place à gauche.
- La dimension verticale peut être graduée afin de représenter l'écoulement du temps
- L'ordre d'envoie des messages est donné par la position de ces messages sur les lignes de vie des objets.

Concepts de diagrammes de séquences

- ❑ L'objet
- ❑ La ligne de vie
- ❑ Les messages
- ❑ Les fragments combinés

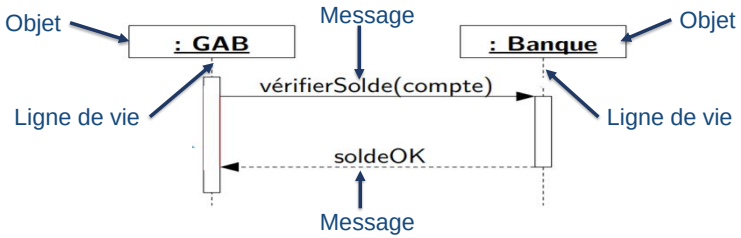


Diagramme de séquence : L'objet

L'objet

- ❑ Les diagrammes de séquences représentant les **échanges** entre les objets mais aussi les échanges avec les acteurs (la représentation du stickman)
- ❑ Notation **des noms** des objets :

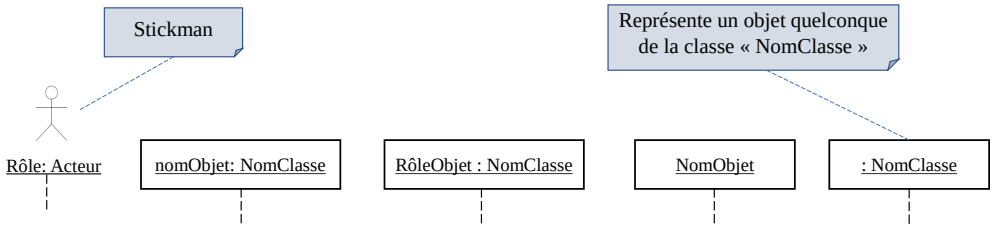


Diagramme de séquence : L'objet

Création et destruction d'objet

- ❑ Lorsqu'un objet est crée au cours de l'exécution d'un scénario, celui-ci n'apparaît qu'au moment où il est créé.
- ❑ Lorsque l'objet est détruit dans un scénario, la destruction se présente par l'achèvement de son ligne de vie par un croix.

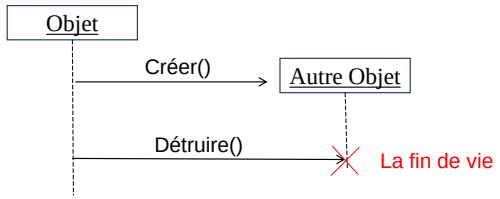


Diagramme de séquence : La ligne de vie

La ligne de vie

- A chaque objet est associé une **ligne de vie** (en trait pointillés à la verticale de l'objet) qui peut être considéré comme un axe temporel
- La ligne de vie indique les périodes d'activité de l'objet
- Une période d'activité correspond au temps pendant lequel un objet effectue une action directement ou par l'intermédiaire d'un autre objet.

Diagramme de séquence : La ligne de vie

La ligne de vie :

- Exemple

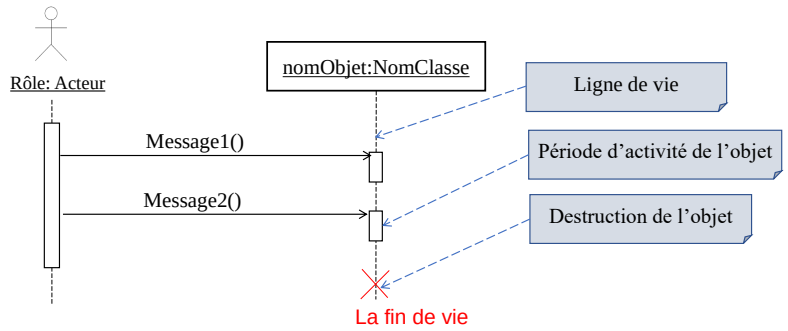


Diagramme de séquence : Les messages

Les messages :

- Un message définit **une communication particulière** entre la ligne de vie d'un objet vers celui d'un autre objet.
- Possède un nom (méthode, signal...)
- Peut avoir des paramètres
- Plusieurs types de messages existent, dont les plus courants :
  - l'envoi d'un signal ;
  - l'invocation d'une opération (appel de méthode) ;
  - la création ou la destruction d'un objet.

Diagramme de séquence : Les types de message

Les messages synchrones :

- Bloque l'émetteur
- Le récepteur rend la main à l'émetteur par **un message de retour** (sa représentation est optionnelle)
- Peut spécifier le résultat de la méthode invoquée
- Graphiquement, les messages synchrones sont représentées par une flèche avec un triangle plein à son extrémité

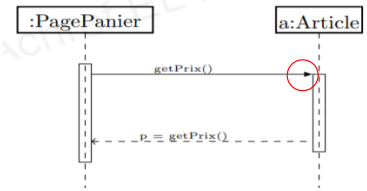
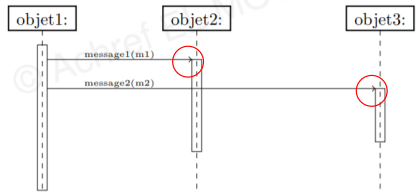


Diagramme de séquence : Les types de message

Les messages asynchrones :

- L'expéditeur n'attend pas la fin de l'activation de la méthode invoquée chez le destinataire.
- Graphiquement, ils sont représentés par une flèche simple.



- Objet1 a envoyé un premier message à Objet2 qui a déclenché une activité chez ce dernier.
- Ensuite, il a envoyé un deuxième message à Objet3 sans attendre ni la réponse de Objet2 ni la fin de son activité

Diagramme de séquence : Les types de message

Les messages de retour (réponse) :

- Si une méthode qui a été activée (par un message) doit retourner des valeurs à la fin de son activation, cela se fait par un message retour.
- Le message de retour n'est donc pas un appel de méthode (il ne provoque donc pas l'activation d'un objet)
- Le message retour porte souvent le nom de l'élément retourné.
- Graphiquement, ils sont représentés par une simple **flèche en pointillés**

Diagramme de séquence : Les types de message

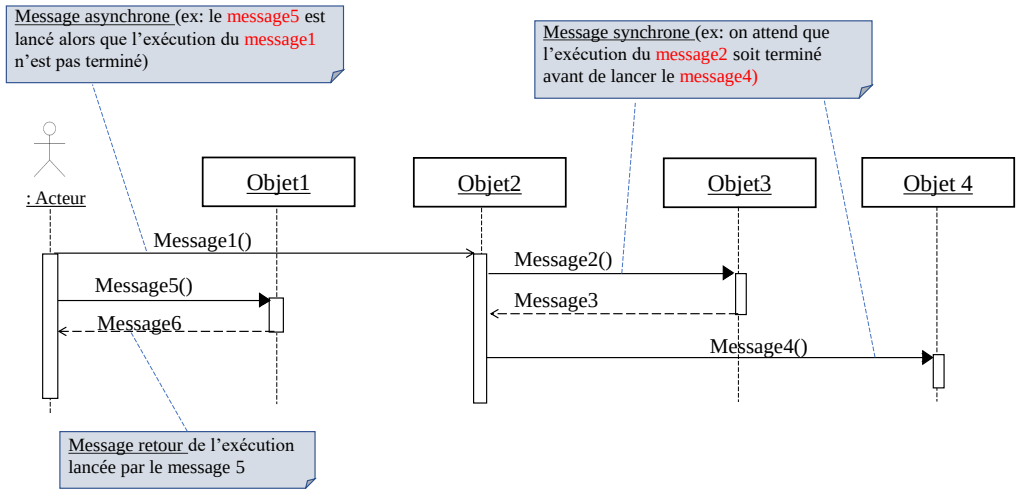
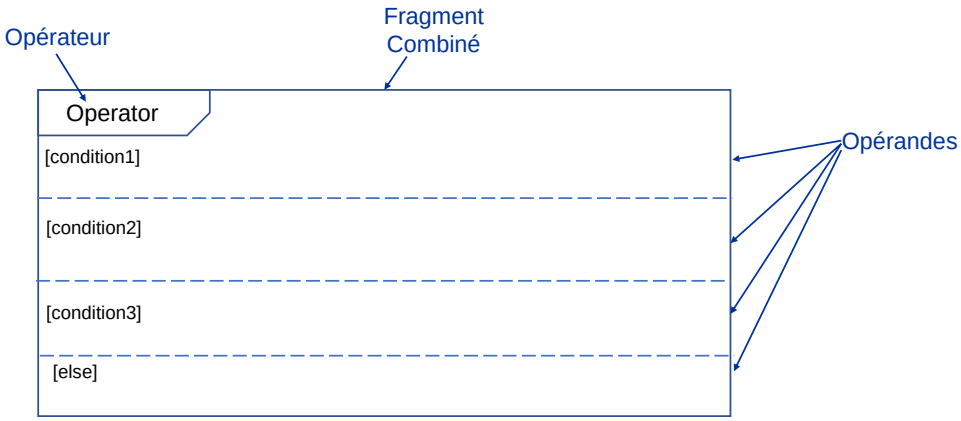


Diagramme de séquence : Les fragments combinés

- Comment faire pour représenter une partie du diagramme de séquence ou pour ajouter un bloc conditionnel ?
  - Utiliser les **fragments combinés**
- Un fragment combiné appelé aussi fragment d'interaction qui se compose d'un ou plusieurs opérateurs d'interaction et d'un ou de plusieurs opérandes d'interaction
- Exemple d'operateurs :
  - **alt** : opérateur conditionnel à plusieurs opérandes (équivalent d'un bloc if ... else if ... else en programmation)
  - **opt** : opérateur conditionnel à une seule opérande (if sans else)
  - **loop** : opérateur répétitif acceptant au max deux valeurs min et max
  - **ref** : opérateur indiquant une référence vers un autre diagramme de séquence existant

Structure des fragments combinés

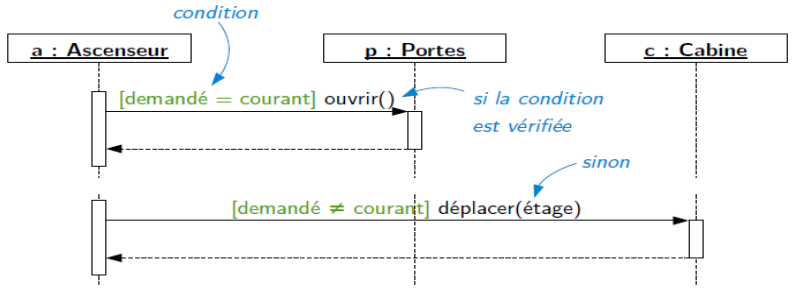


➢ Certains fragments n'ont pas besoin d'une condition ni de plusieurs opérandes.

Structure des fragments combinés : alt

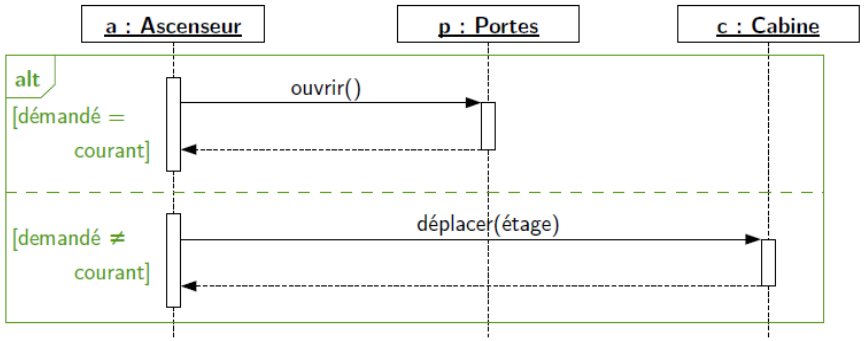
❑Alternative: Condition à l'envoi d'un message

- Exemple:



Structure des fragments combinés : alt

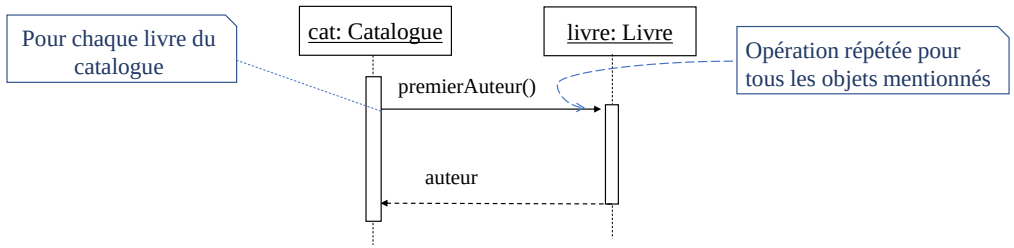
❑ Bloc alternative: alt



Structure des fragments combinés : loop

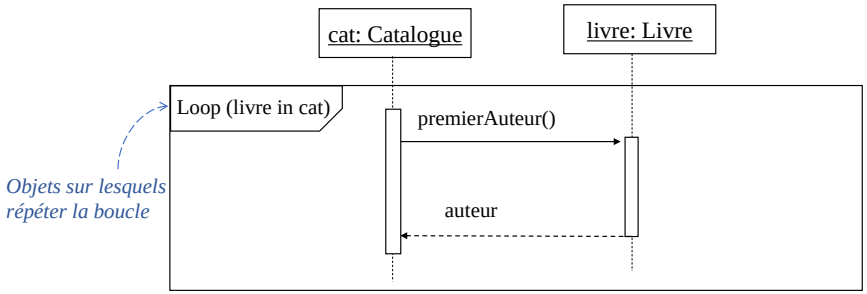
❑ Boucle: Répéter un enchaînement de messages

- Exemple:



Structure des fragments combinés : loop

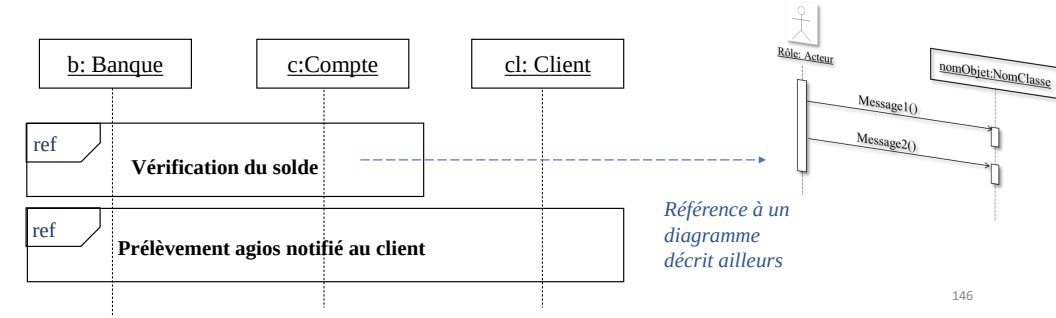
❑ Bloc de boucle loop



Structure des fragments combinés : Ref

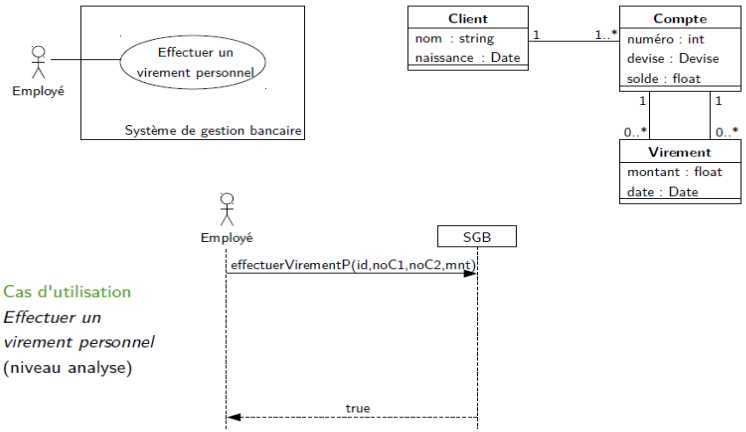
❑ **L'opérateur ref :** sert comme une référence ou un raccourci vers un autre diagramme de séquence existant

❑Le rôle de l'opérateur **ref** est de réutiliser une partie de diagramme de séquence commune.



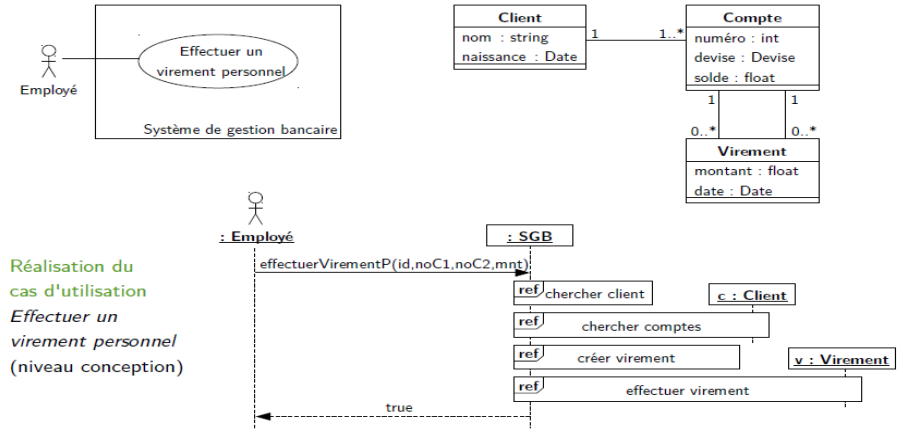
Exemple

❑ Exemple: Analyse



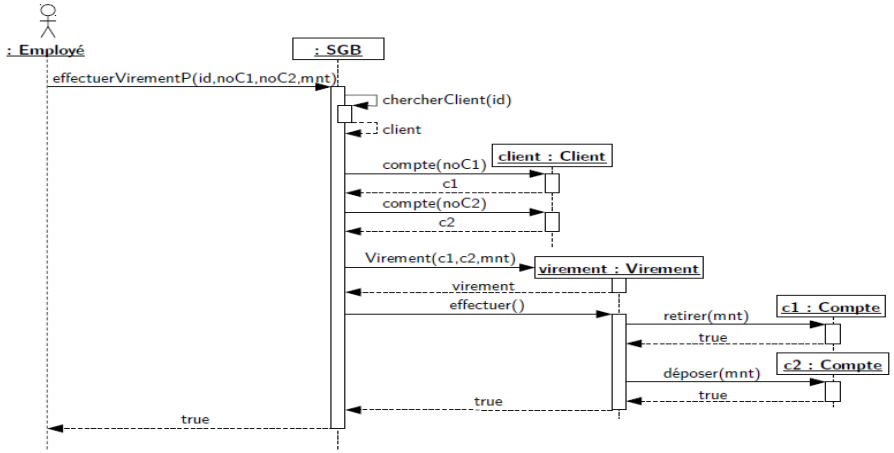
Exemple

❑ Exemple: Conception



Exemple

❑ Exemple: Conception



QCM

1) Un diagramme de séquence se lit?
   1. De haut en bas
   2. De bas en haut
2) Comment s'appelle les lignes verticales sortant d'un objet/acteur?
   1. Ligne de vie
   2. Ligne de temps
   3. Message

3) Des messages peuvent s'envoyer en même temps
   1. Vrai
   2. Faux
4) Un message asynchrone a un retour
   1. Vrai
   2. Faux

Conclusion

- ❑ Les diagrammes de CU expriment les besoins des utilisateurs d'une manière abstraite.
- ❑ Le diagramme de séquence explicitent la dimension temporelle et permettent d'exprimer les contraintes temporelles dans les envois de messages.
- ❑ C'est un bon moyen pour représenter les interactions, organisées dans le temps, entre les objets.
- ❑ Il s'approche davantage de l'implémentation avec l'utilisation des boucles, des structures de contrôles, etc.

Méthodes de développement
Agile - Scrum

Introduction

- Les méthodes classiques de développement de logiciel présentent de nombreux inconvénients:
  - Enorme effort fournit pendant le phase de planification
  - Mauvaise conversion des exigences dans un environnement en évolution rapide
  - Traitement du personnel en tant que facteur de production

➡ **Nouvelles Méthodes :** Développement logiciel **Agile**

Méthodologie : Méthode Agile

- Définir la méthodologie du travail est une étape décisive pour l'élaboration d'une application informatique.

- **Définition [5] :**

« Une méthode **Agile** est une approche itérative et incrémentale pour le développement de logiciel, réalisé de manière très collaborative par des équipes responsabilisées, appliquant un cérémonial minimal, qui produisent, dans un délai contraint, un logiciel de grande qualité répondant aux besoins changeants des utilisateurs. »

Méthode Agile

- **Le processus Agile:**
  - Permet de s'orienter vers l'opérationnel plutôt qu'accumuler une masse d'informations difficile à traiter et qui fait perdre de vue l'essentiel.
  - Il cherche la collaboration avec le client.
  - Le développement Agile tend à s'adapter aux changements qui s'opèrent durant la réalisation du projet plutôt que suivre un plan préétabli, qui ne prend pas en compte l'évolution de la situation.

Méthode Scrum

- **Définition [6] :**
  - Scrum signifie mêlée au rugby.
  - Scrum utilise les valeurs et l'esprit du rugby et les adapte aux projets de développement.
  - Comme le pack lors d'un ballon porté au rugby, l'équipe chargée du développement travaille de façon collective, soudée vers un objectif précis.
  - Comme un demi de mêlée, le Scrum Master guide les membres de l'équipe, les repositionne dans la bonne direction et donne le tempo pour assurer la réussite du projet.

Méthode Scrum

- **Scrum** définit **trois rôles** dans **l'équipe** du travail :
  - **Product Owner :** C'est le propriétaire du produit, une personne qui porte la vision du produit à réaliser, généralement c'est un expert dans le domaine
  - **Scrum Master :** C'est la personne qui doit assurer le bon déroulement des différents sprints, et qui doit impérativement maitriser Scrum
  - **Le Scrum Team :** L'équipe de Scrum est constituée des personnes qui seront chargées d'implémenter les différents besoins du client.

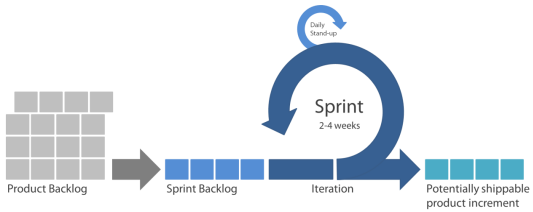
Méthode Scrum

- **Scrum** se caractérise par deux notions principales : le **backlog** et les **sprints**
- Les Sprints sont des boites de temps durant lesquelles des fonctionnalités sont réalisées.
- Le backlog du produit est le carnet contenant les fonctionnalités à accomplir pendant la période du travail

➡ Le **backlog du sprint** désigne les tâches qui doivent être **achevées** pendant un sprint bien déterminé

Mécanisme de fonctionnement du Scrum

- Sprint Planning Meeting : Réunion de planification de sprint
- Sprint
- Daily Scrum: Scrum quotidien
- Sprint Review Meeting : Réunion d'examen du Sprint



Sprint Planning Meeting

- Une **réunion collaborative** au début de chaque sprint entre le Product Owner, le Scrum Master et l'équipe
- Prend 8 heure et se compose de 2 parties :
  - Partie 1 :
    - Création de backlog de produit
    - Déterminer l'objectif du sprint
    - Participants : Product Owner, Scrum Master, Scrum Team
  - Partie 2 :
    - Participants : Scrum Master et Scrum Team
    - Créer un backlog de Sprint

Daily Scrum

- Il s'agit d'une courte réunion (15 minutes) qui se tient tous les jours avant le début des travaux de l'équipe
- Participants : Scrum Master et Scrum Team
- Chaque membre de l'équipe doit répondre à 3 questions ;
  - Q u'as-tu fais hier?
  - Que vas-tu faire aujourd'hui?
  - Quel sont les obstacles qui te freinent dans mon travail?
- C'est un bon moyen pour un Scrum Master de suivre le progrès de l'équipe
- N'est pas une session de résolution de problème
- N'est pas un moyen de collecter des informations sur les retardataires

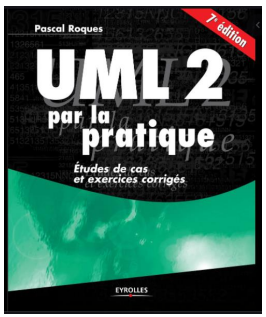
Sprint Reviewer Meeting

- Se tient à la fin de chaque sprint
- La fonctionnalité métier créée lors du sprint est présentée au Product Owner
- Informel, ne doit pas distraire les membres de l'équipe de leur travail

Bibliographie

Ce cours a été réalisé en se basant sur les références suivantes :

[1] Livre « **UML 2 Par la Pratique: Études de Cas Et Exercices Corrigés** » par **Pascal Roques**



Bibliographie

Ce cours a été réalisé en se basant sur les références suivantes :

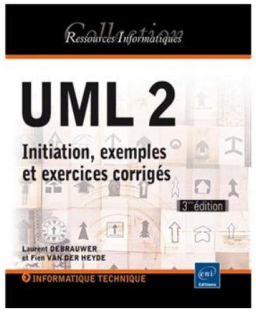
[2] Cours « **Modélisation UML**» par **Christine Solnon,** INSA de Lyon

[3] Cours « **Introduction à UML 2 Modélisation Orientée Objet de Systèmes Logiciels »** par **Pierre Gérard**, Université de Paris 13 IUT Villetaneuse.

Bibliographie

- Ce cours a été réalisé en se basant sur les références suivantes :

[4] « UML 2 : initiation, exemples et exercices corrigés » par Laurent Debrauwer et Fien Van Der Heyde



Bibliographie

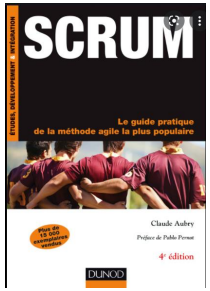
- Ce cours a été réalisé en se basant sur les références suivantes :

[5] Le livre « Scrum guide » par Ryan Smith

Bibliographie

- Ce cours a été réalisé en se basant sur les références suivantes :

[6] Le livre « SCRUM » par Claude Aubry



Merci pour votre attention