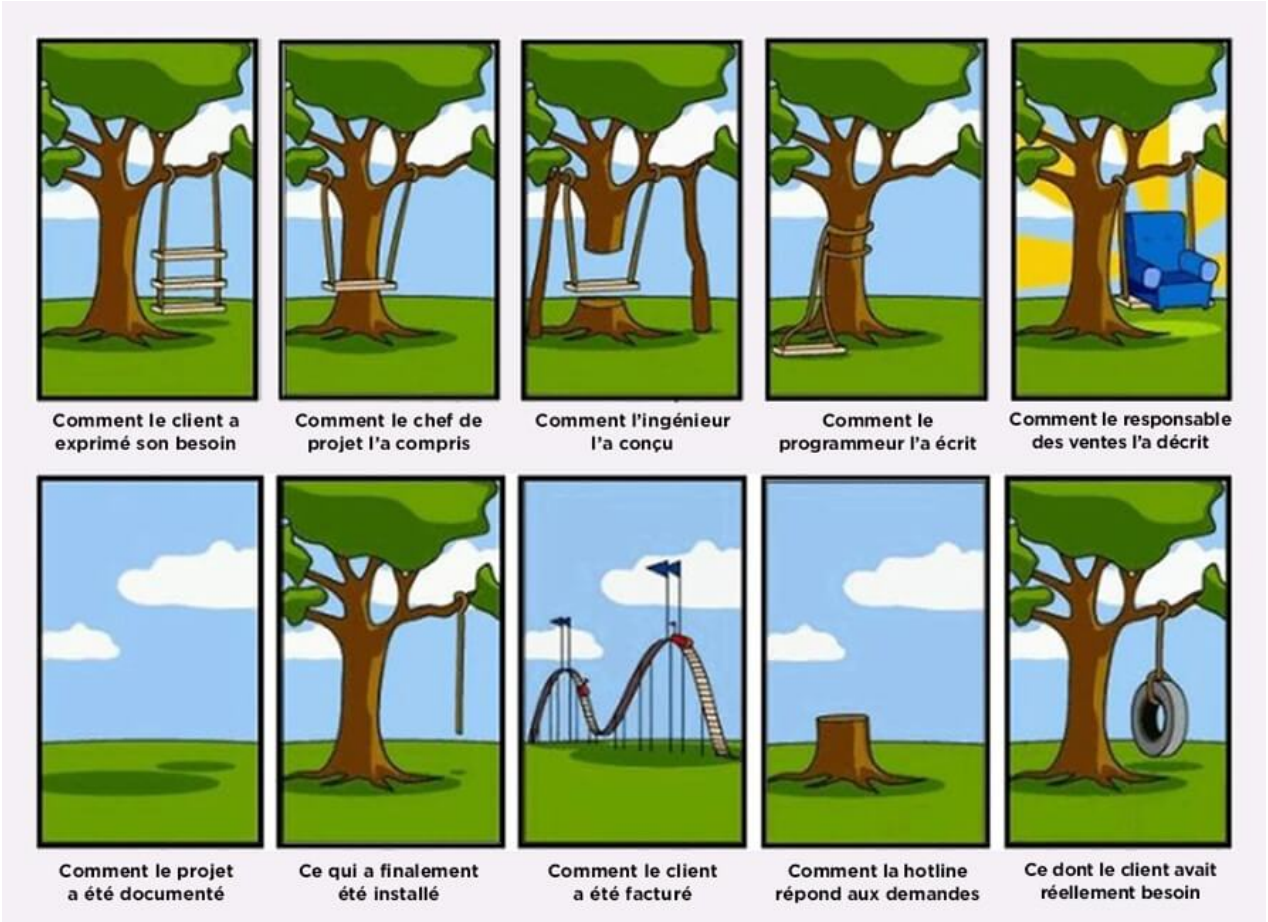
UML et POO

Xavier Nodet, xavier.nodet@gmail.com

Janvier 2022

Plan du cours

- Introduction
- UML
  - Étude fonctionnelle : acteurs et cas d'utilisation
  - Modélisation statique : classes et objets, attributs, opérations, etc
- Programmation orientée objet
  - Python, Java et C++


Comment le client a exprimé son besoin
Comment le chef de projet l'a compris
Comment l'ingénieur l'a conçu
Comment le programmeur l'a écrit
Comment le responsable des ventes l'a décrit
Comment le projet a été documenté
Ce qui a finalement été installé
Comment le client a été facturé
Comment la hotline répond aux demandes
Ce dont le client avait réellement besoin

Introduction

- Communication avec les clients
- Communication avec les developpeurs

- Textes et diagrammes UML
- Quelques notions de POO

Plan du cours

- Introduction
- UML
  - Étude fonctionnelle : acteurs et cas d'utilisation
  - Modélisation statique : classes et objets, attributs, opérations, etc
- Programmation orientée objet
  - Python, Java et C++

UML : les acteurs

UML : les acteurs

- Rôle joué par une entité
- Ne fait pas partie du système étudié
- Humain ou non

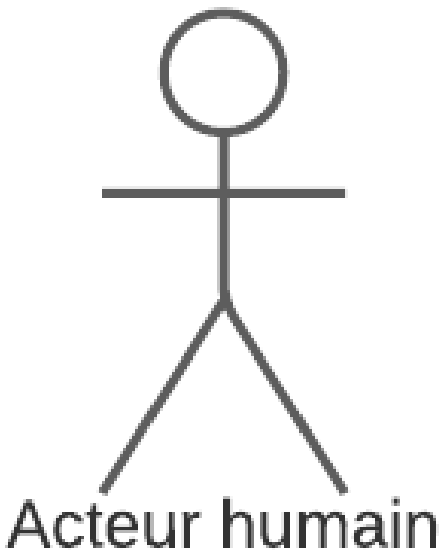
UML : les acteurs

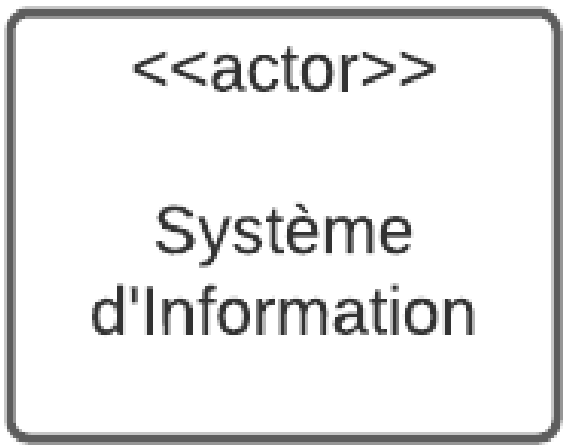
- Rôle joué par une entité
- Ne fait pas partie du système étudié
- Humain ou non

Exemples:

- L'utilisateur d'une carte de paiement lors d'une transaction sur Internet.
- Le système de gestion des stocks, dans l'étude d'une caisse de supermarché.

UML : les acteurs





https://forms.gle/z5gwPvn2E6UDomQZ9

Cas d’utilisation

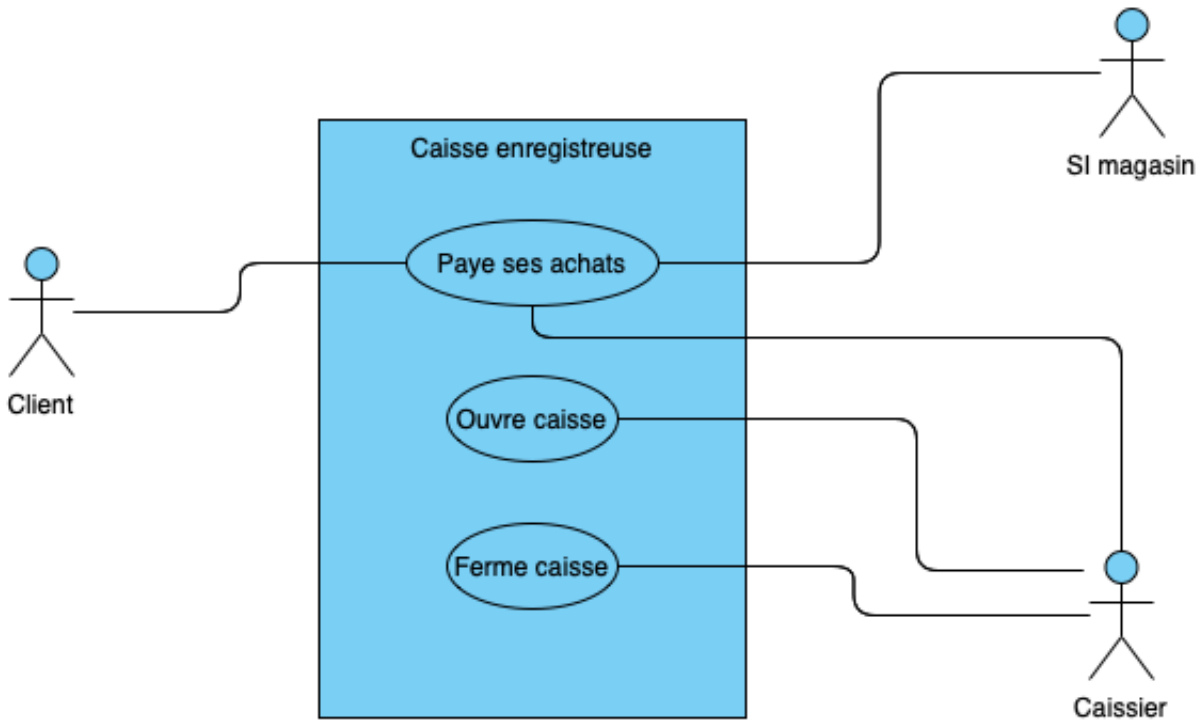
- Séquence d’évenements au cours de laquelle l’acteur principal intéragit avec le système
- Description du comportement attendu du système
- Description du *quoi*, et non pas du *comment*

Cas d’utilisation

- Scénario nominal
- Enchainements alternatifs :
  - Le porteur de carte fait une ou deux (mais pas trois) erreurs de code.
  - Le client présente sa carte de fidélité à la caisse
- Enchainements d’erreur :
  - Pas d’autorisation de retrait
  - Livre déjà réservé

Cas d’utilisation

- Pré-conditions, post-conditions
- Exigences non fonctionnelles



https://forms.gle/6Ti8UEjQKR2H6c7G9

Diagramme d’activité

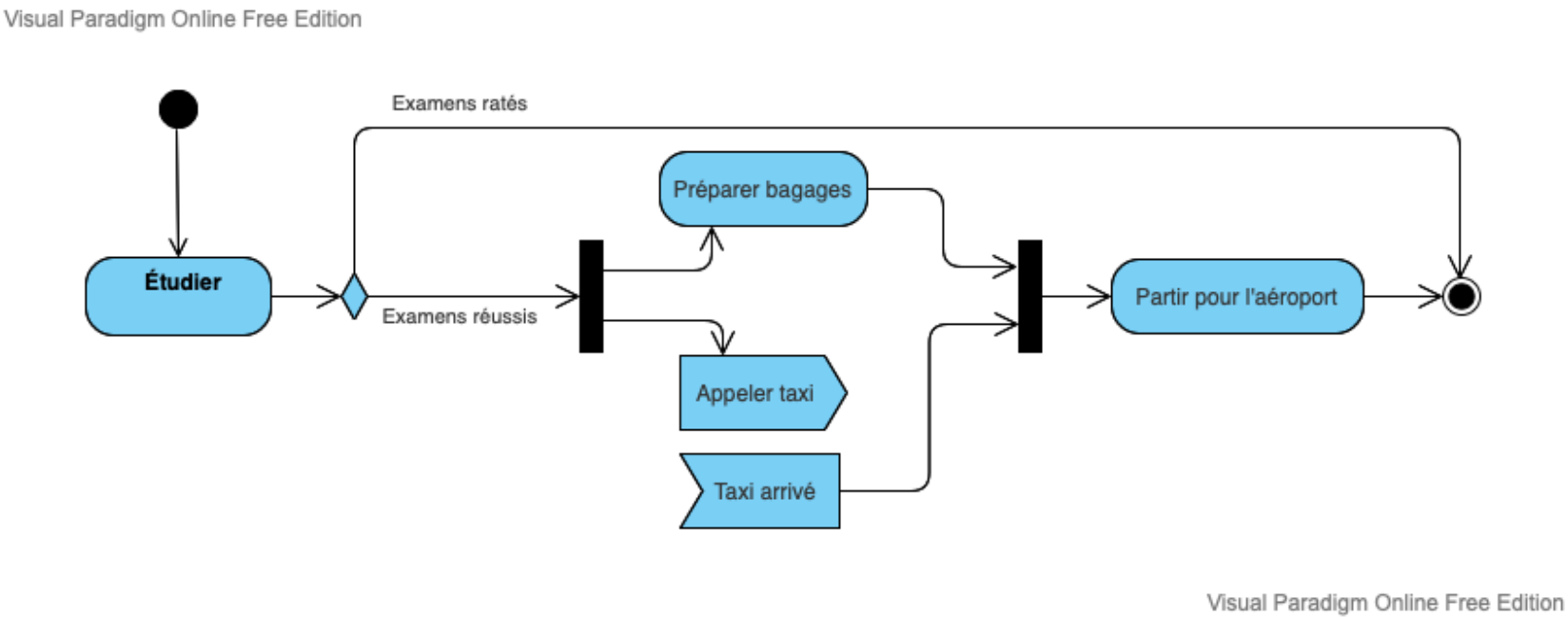
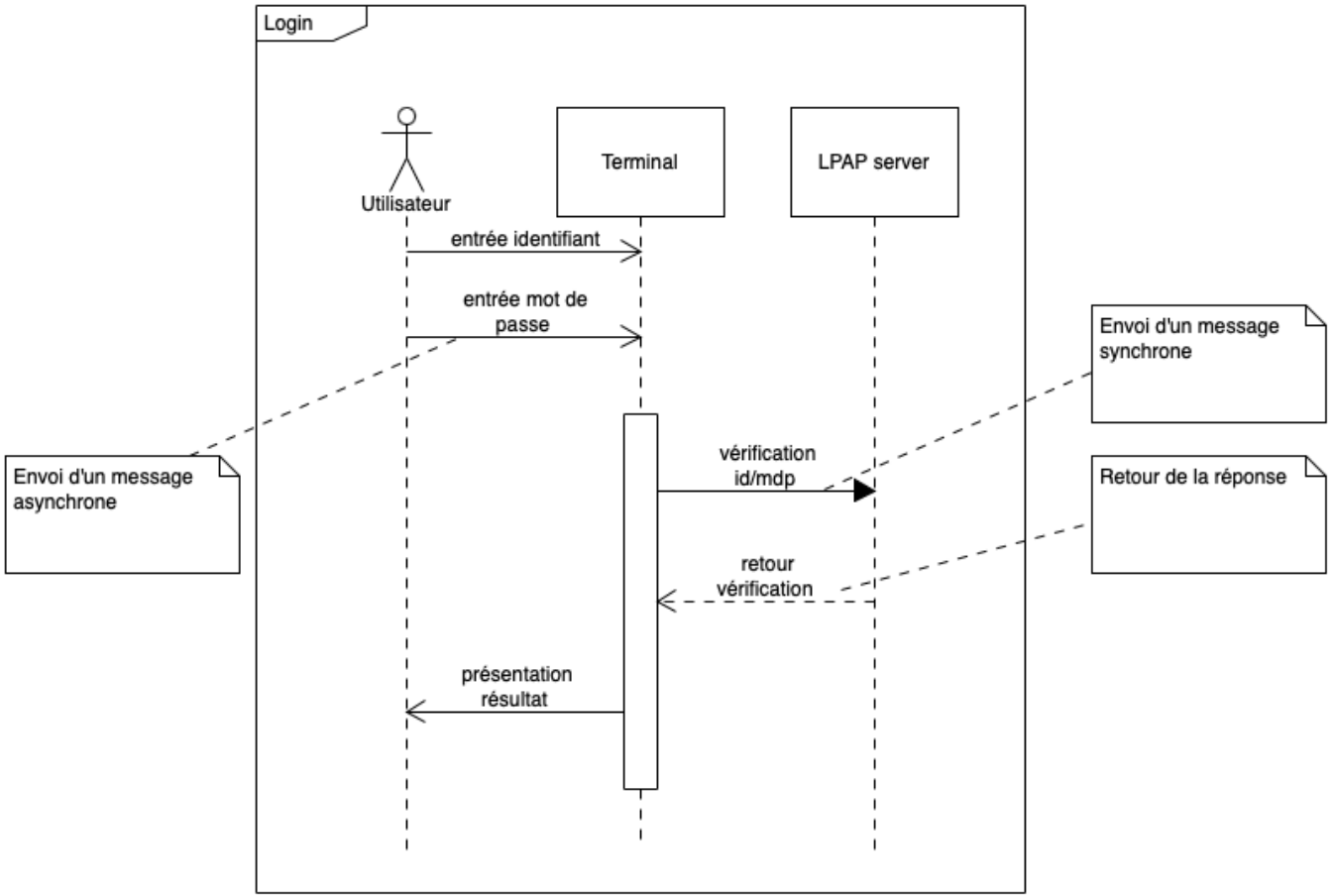
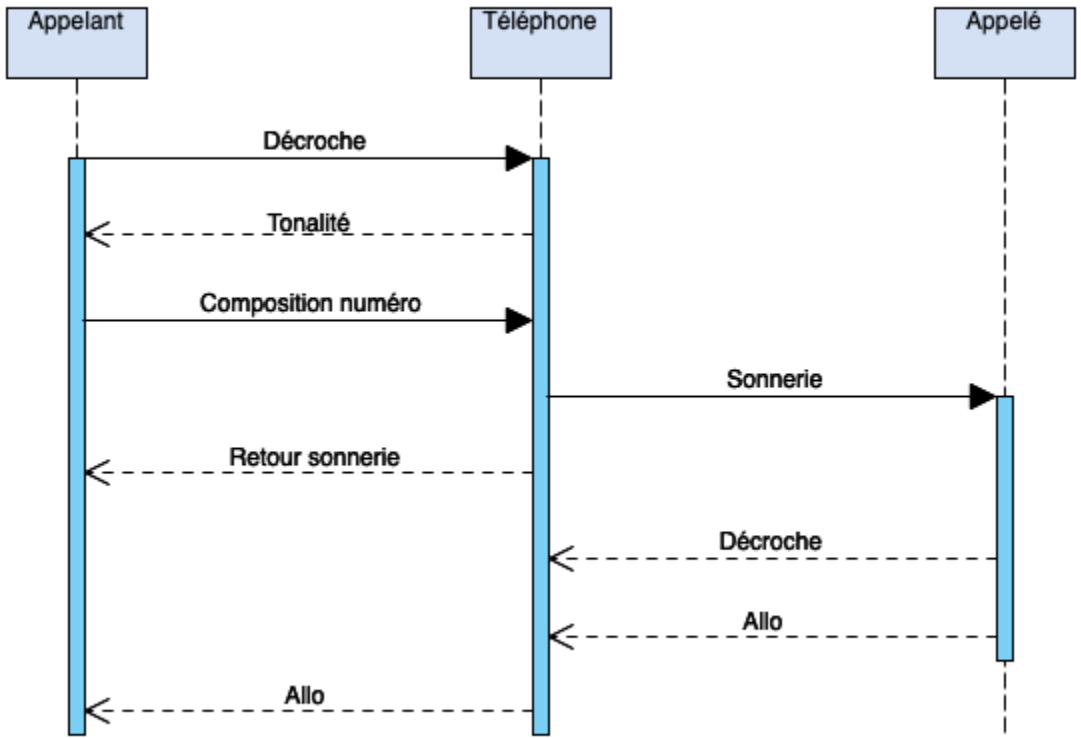


Diagramme de séquence

- Décrit les actions et messages échangés entre les acteurs
- Une *ligne de vie* verticale pour chaque acteur
- Messages synchrones ou asynchrones


Login
Utilisateur
Terminal
LPAP server
entrée identifiant
entrée mot de passe
Envoi d'un message synchrone
Envoi d'un message asynchrone
vérification id/mdp
Retour de la réponse
retour vérification
présentation résultat



Plan du cours

- Introduction
- UML
  - Étude fonctionnelle : acteurs et cas d'utilisation
  - Modélisation statique : classes et objets, attributs, opérations, etc
- Programmation orientée objet
  - Python, Java et C++

Modélisation statique

Décomposition

- Un système complexe sere décomposé pour faciliter son étude.
- Les composants d'un système deviennent acteurs pour l'étude d'un sous-système.

Fin de l'étude fonctionnelle

- En théorie, une fois l'étude fonctionnelle terminée, parler au client ne devrait plus être nécessaire.
- En pratique, ce n'est pas aussi simple…

Cycle en V

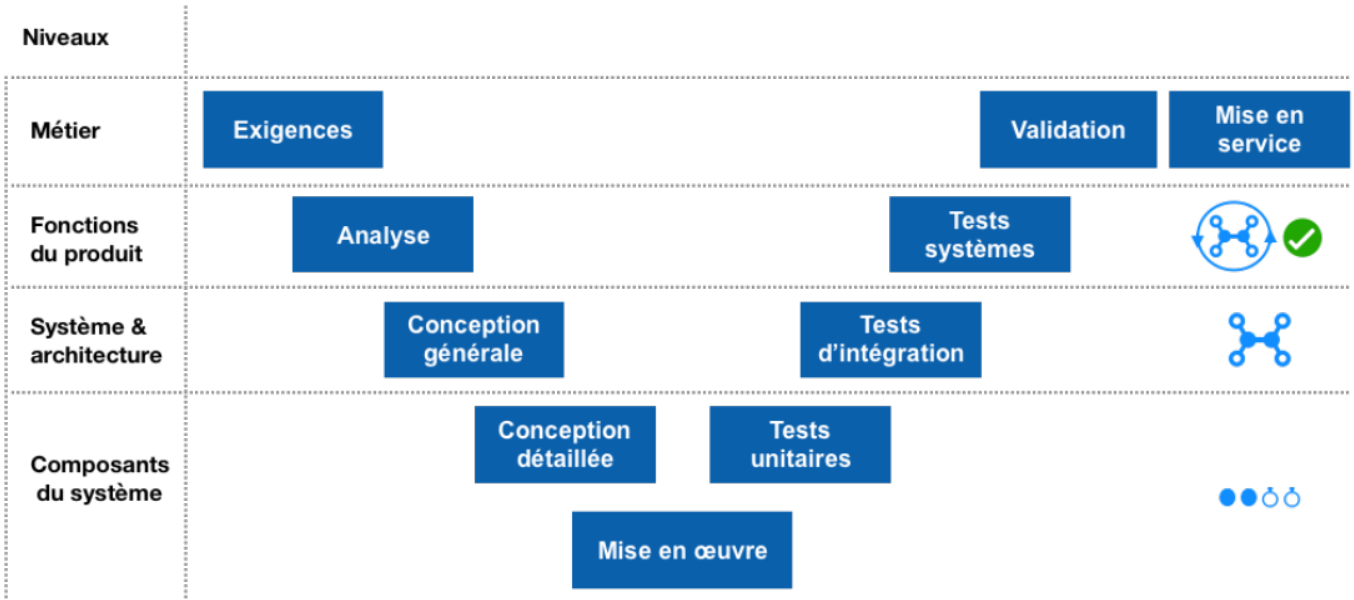


Figure 1. Le cycle en V, par Cth027 — Travail personnel, CC BY-SA 4.0, link

Méthodes agiles

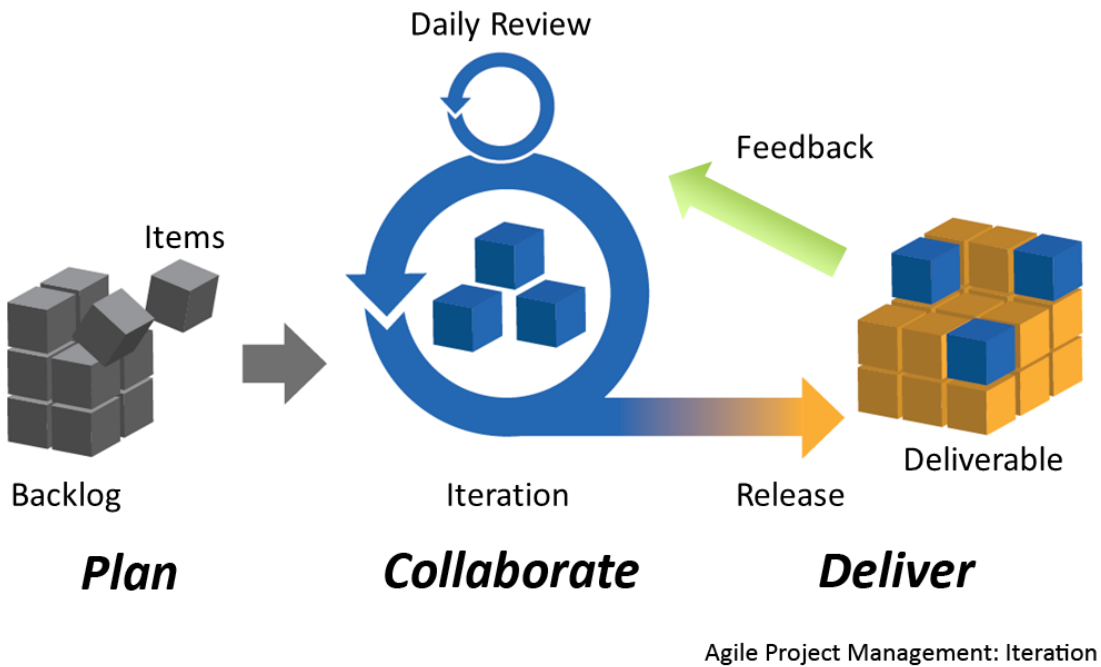


Figure 2. Les méthodes agiles, par Planbox - Travail personnel, CC BY-SA 3.0, link

Classes et objets

- Chaque type d'acteur est représenté par une *classe*.
- Les éléments manipulés dans le système étudié seront également représentés par des classes :
  - les livres d'une bibliothèque
  - les voitures d'un concessionnaire automobile

Classes et objets

- La classe est le patron, le modèle, à partir duquel les objets sont *instanciés*.
- Chaque objet est construit à partir d'une et une seule classe.

- Chaque classe n'est pas nécessairement instanciée plusieurs fois.
- Exemples :
  - l'IHM d'un programme
  - le serveur de base de donnée auquel le système est connecté

Attributs

- Attribut : propriété d'une classe qui associe une *variable* à chaque instance de cette classe.
- Exemples :
  - Les prénoms, noms et date de naissance d'un utilisateur
  - Le titre et le nombre de mots d'un livre

Attributs

- Un attribut peut avoir une type simple (entier, chaîne de caractères, date, etc).
- Les liens avec d'autre objets ne sont pas des attributs, mais des relations.
- Un attribut peut être dérivé, déduit d'informations présentes ailleurs dans le modèle. Il est noté */attribut*.

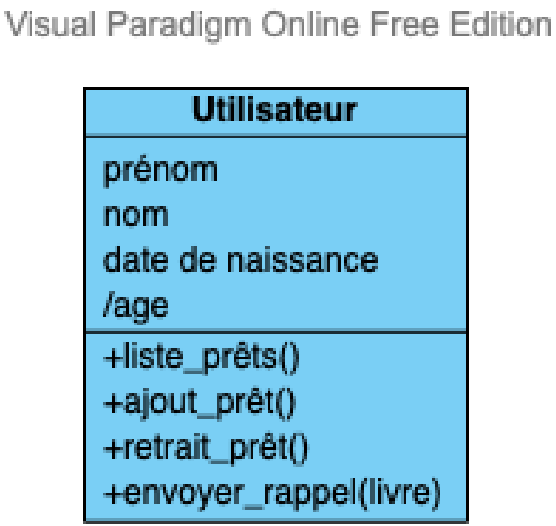
Opérations

- Une classe peut aussi définir des *opérations*. Ces opérations représentent des services que peuvent rendre les instances de la classe
- Exemples :
  - *nombre_emprunts_en_cours*
  - *rendre(livre)*
  - *envoyer_rappel(utilisateur, livre)*

Opérations

- Trois types de services :
  - demande d’information
  - enregistrement d’information
  - traitements sans échange d’informations

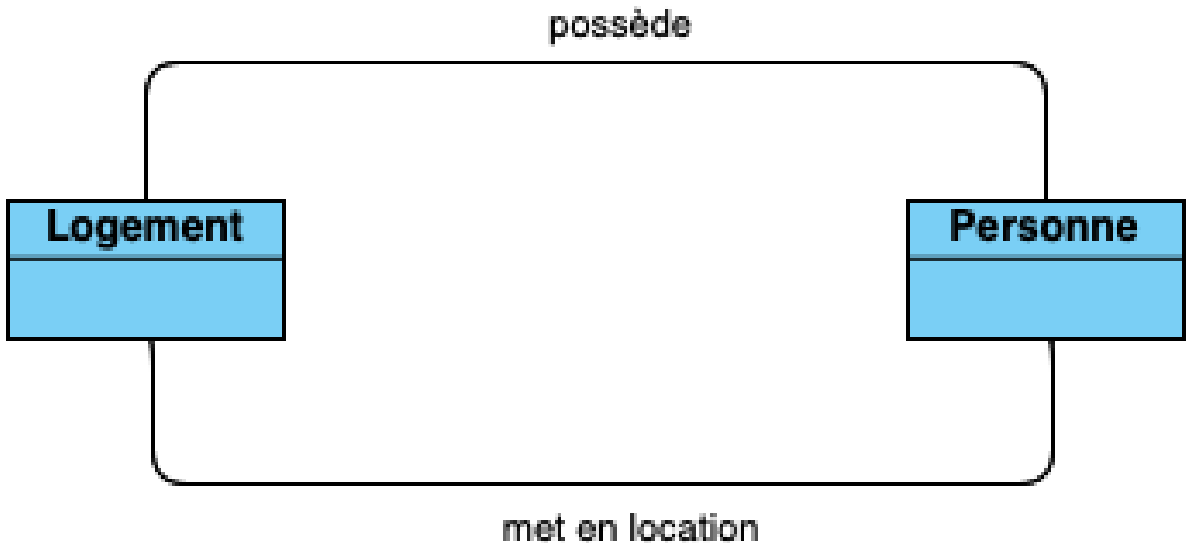
Classe


Visual Paradigm Online Free Edition
Utilisateur
prénom
nom
date de naissance
/age
+liste_prêts()
+ajout_prêt()
+retrait_prêt()
+envoyer_rappel(livre)
Visual Paradigm Online Free Edition

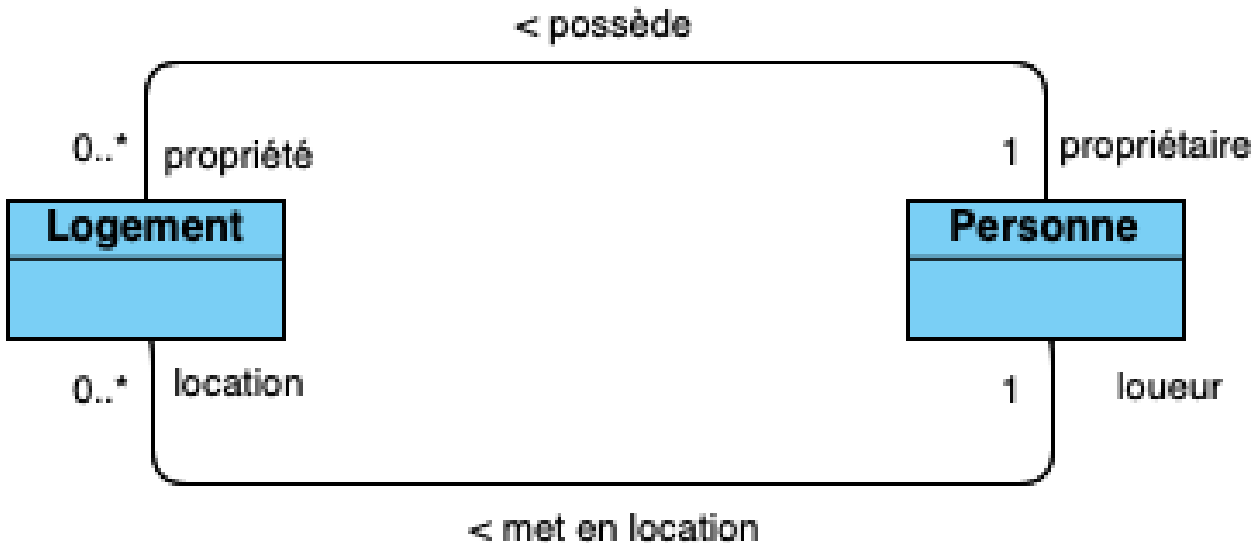
Associations

- Association : relation sémantique durable entre deux classes.
- Exemples :
  - Une bibliothèque possède des livres. La relation *possède* est une association entre la classe *Bibliothèque* et la classe *Livre*.
  - Un utilisateur emprunte des livres. La relation *emprunte* est une association entre la classe *Utilisateur* et la classe *Livre*.

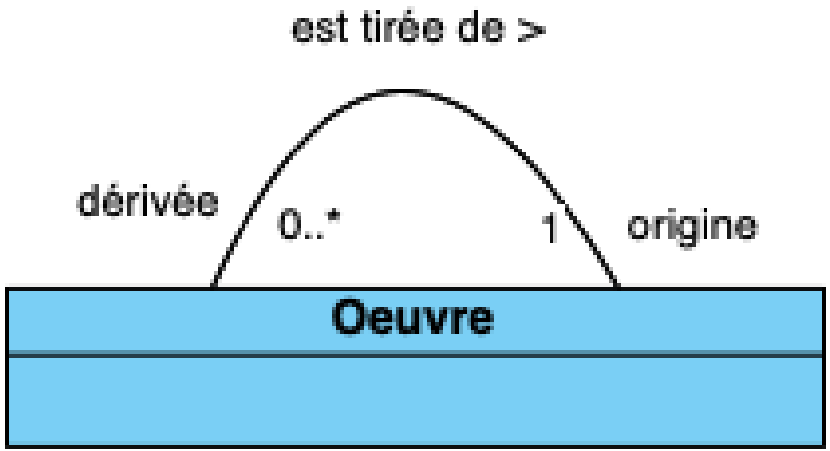
Associations



Associations

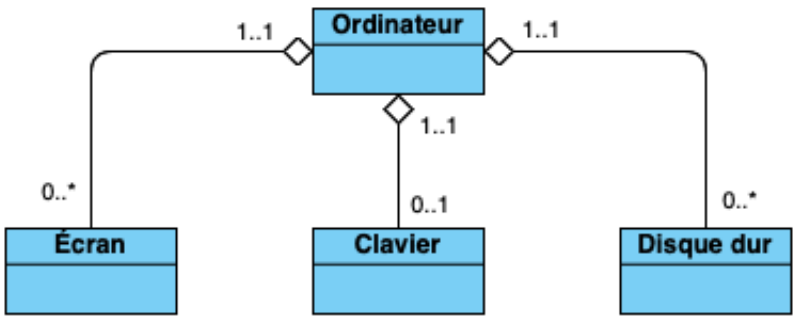


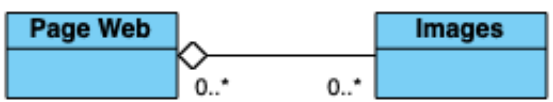
Associations



Agrégation

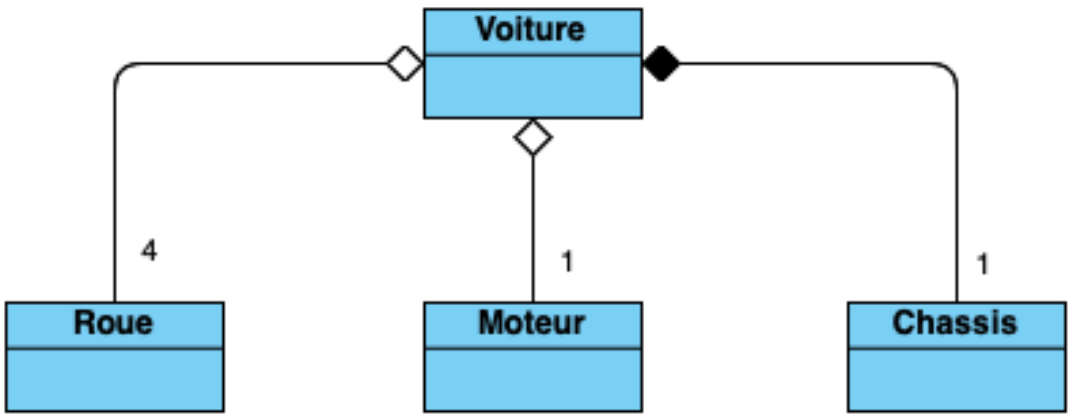
Visual Paradigm Online Free Edition



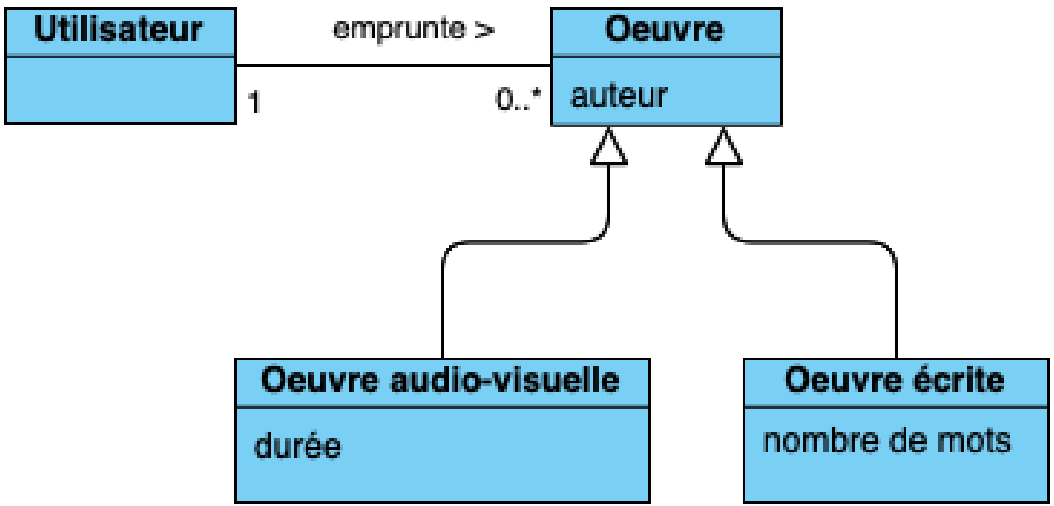


Visual Paradigm Online Free Edition

Composition



Généralisation



Généralisation

- *Oeuvre* est la *généralisation* de *Oeuvre écrite* et *Oeuvre audio-visuelle*.
- Réciproquement, on parle de *spécialisation*.
- Les instances d'une classe spécialisée sont aussi des instances de la *classe de base*, ou *super-classe*.
- La classe spécialisée *hérite* de tous les attributs et méthodes.
- Elle peut rajouter ses propres propriétés.
- Elle respecte le contrat de la classe de base : pré-conditions, post-conditions, invariants.