Programmation Orientée Objet en Python

#4 UML Structural Diagrams

par David Albert

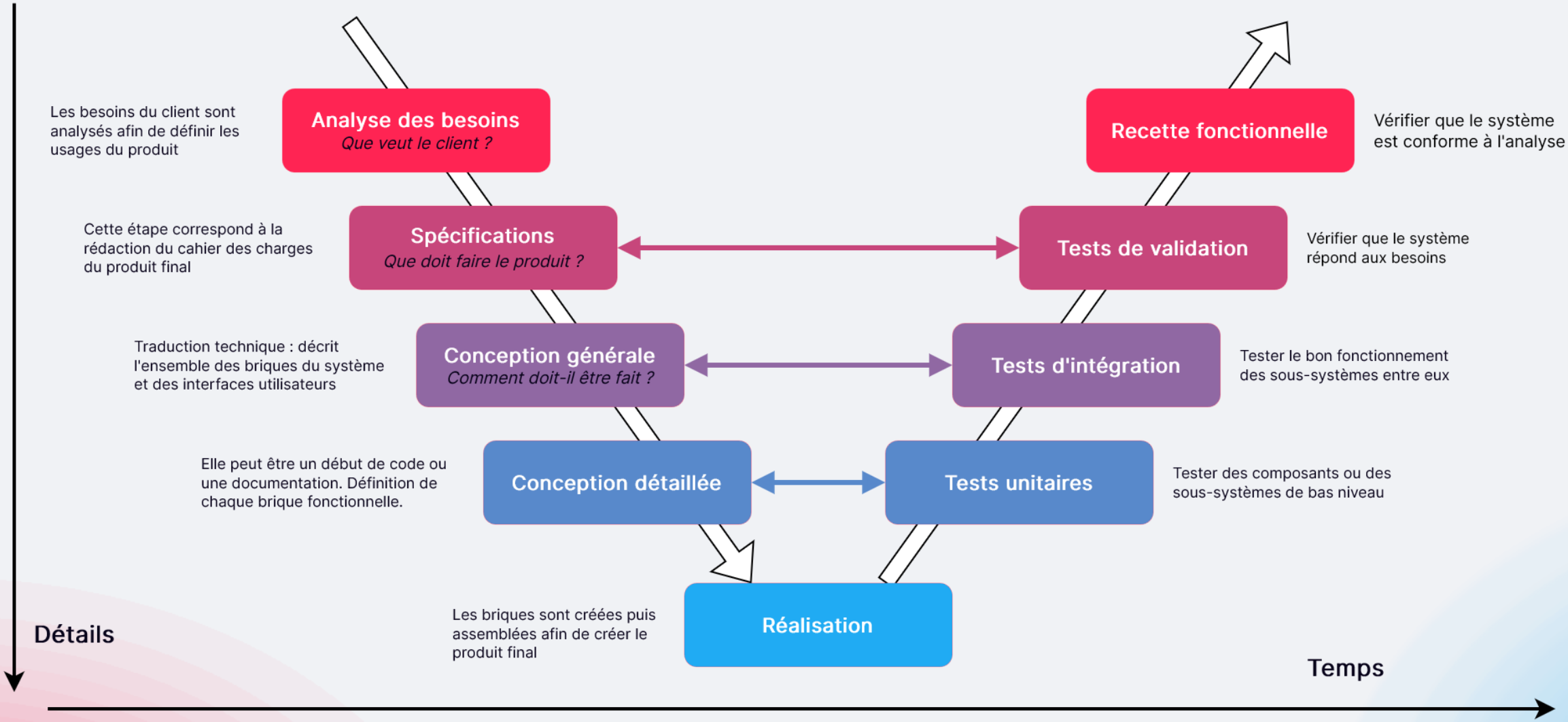
2023

Table des matières

01

Cycles de développement

Cycle en V


Les besoins du client sont analysés afin de définir les usages du produit
Analyse des besoins
Que veut le client ?
Recette fonctionnelle
Vérifier que le système est conforme à l'analyse
Cette étape correspond à la rédaction du cahier des charges du produit final
Spécifications
Que doit faire le produit ?
Tests de validation
Vérifier que le système répond aux besoins
Traduction technique : décrit l'ensemble des briques du système et des interfaces utilisateurs
Conception générale
Comment doit-il être fait ?
Tests d'intégration
Tester le bon fonctionnement des sous-systèmes entre eux
Elle peut être un début de code ou une documentation. Définition de chaque brique fonctionnelle.
Conception détaillée
Tests unitaires
Tester des composants ou des sous-systèmes de bas niveau
Les briques sont créées puis assemblées afin de créer le produit final
Réalisation
Détails
Temps

Vers une méthodologie AGILE

!

Le **cycle en V** a un **inconvénient majeur**. La vérification de la conformité aux besoins client attend la fin du développement du produit. S'il y a un soucis, on s'en rend compte **très tardivement**.

Méthologie AGILE

Pour pallier à cela, les entreprises privilégient de plus en plus des **cycles courts** et successifs. On répètera successivement les étapes de *spécifications*, *conception*, *développement*, *test et validation*.

02

Introduction à UML

Unified Modeling Language

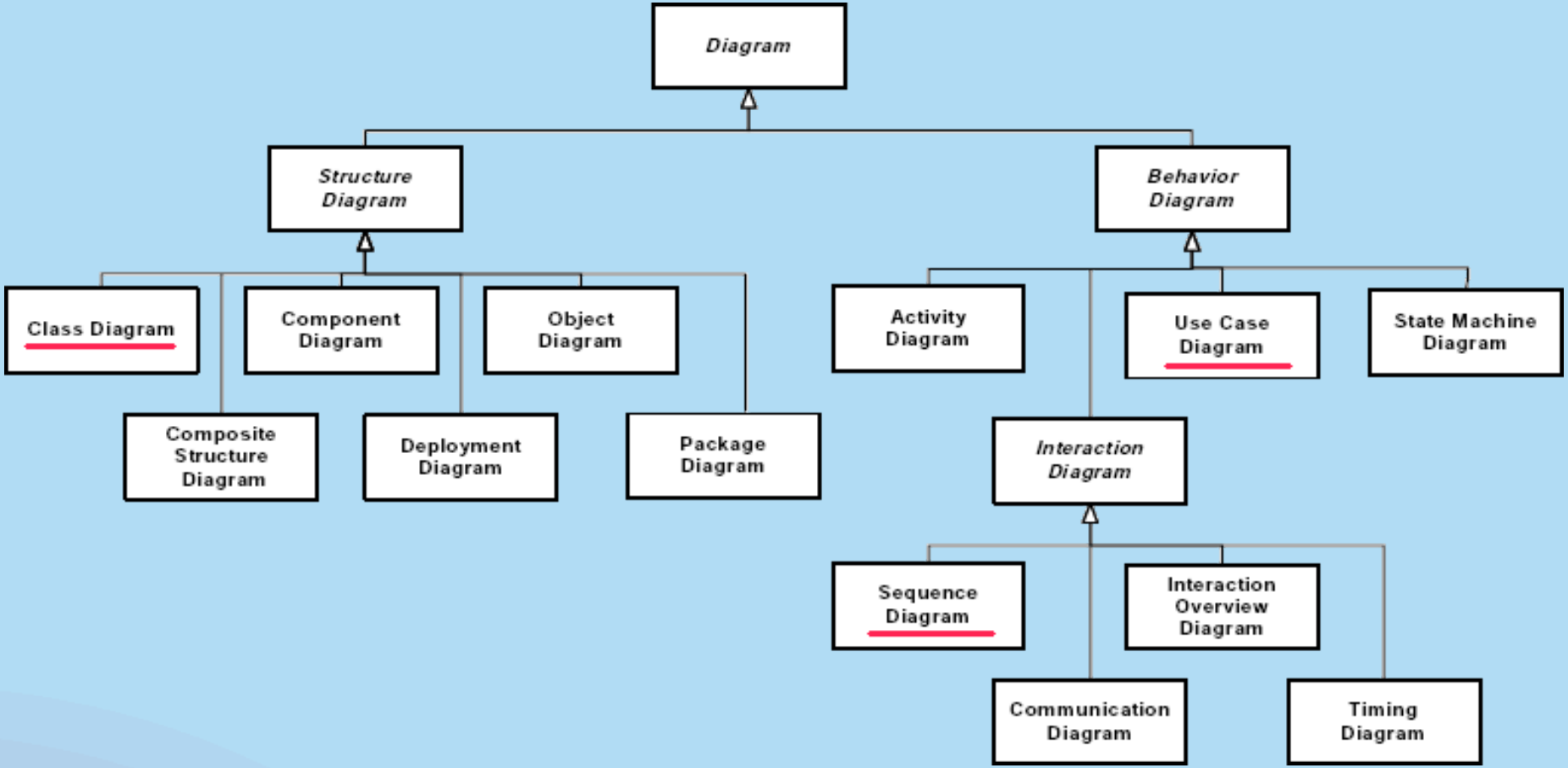
Motivations

- Besoin de se comprendre
- Besoin de conception pour réaliser une architecture complexe.

UML c'est quoi ?

- un langage de modélisation de systèmes informatiques
- modèle graphique (à base de pictogrammes)
- indépendant du langage de programmation
- intervient dans la phase de conception (générale et détaillée)

Fun fact : UML est décrit en UML.



Quelques diagrammes

Diagrammes structurels

Diagramme de classes
Définit l'ensemble des classes et de leurs relations

Diagramme de composants
Liste les composants logiciels

Diagramme de déploiement
Définit la répartition des composants sur une architecture matérielle

Diagrammes de comportement

Diagramme des cas d'utilisation
Définit les scénarios d'interaction entre les utilisateurs et le système

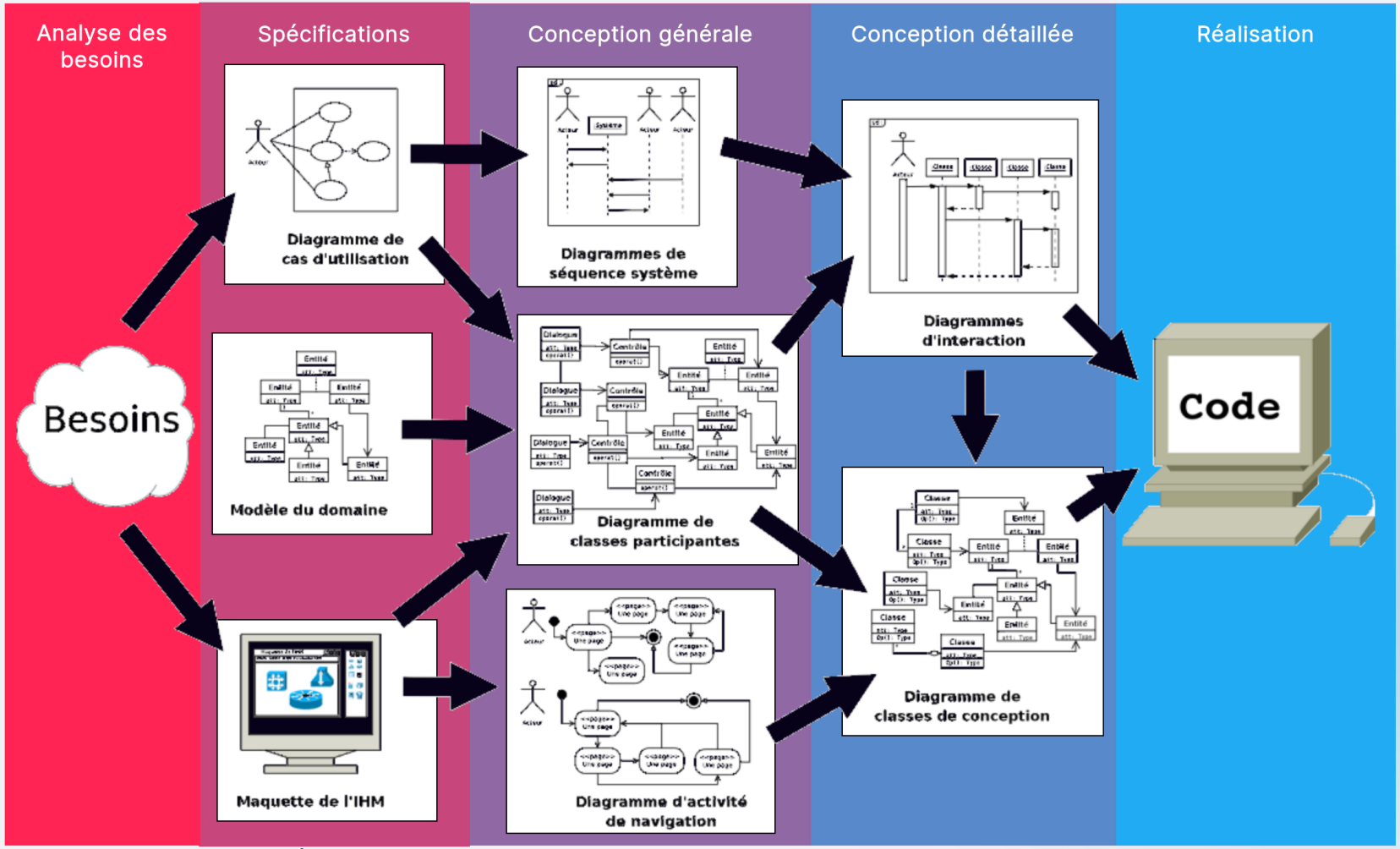
Diagramme d'activité
Représente les états du système et leurs transitions par événements

Diagramme de séquence
Représente les scénarios d'interactions entre entités du système

Référence: Laurent Vercouter, Cours UML, Insa Rouen

Chaîne de conception

Différents diagrammes arrivent à différents moments dans la chaîne de conception.



Référence:
Laurent Audibert

03

Diagramme de cas d'utilisation

Etude de cas

Gestion d'un restaurant

On souhaite réaliser une application pour un restaurant qui lui permettra de gérer les réservations et les commandes de ses clients.

> **On utilisera cet exemple comme fil rouge tout au long de ce cours.**

Enoncé détaillé

Le restaurant accueille des clients qui sont identifiés par leur nom, leur email et leur numéro de téléphone. Le restaurant est ouvert tous les jours de 19h et 23h30. Il réalise chaque soir 3 services de 1h30 et accueille jusqu'à 20 clients par service. Les clients peuvent réserver une table sur ces créneaux. S'il n'y a plus de place, ils peuvent également commander leur repas en ligne, payer via l'application et venir le récupérer dans la foulée. Sur place, les serveurs s'occupent des commandes et du paiement des clients.

Diagramme de cas d'utilisation

Résumé

Objectifs

- Premier diagramme réalisé pour définir les scénarios d’usage
- A réaliser avec le client
- À utiliser tout au long du développement

Exemple

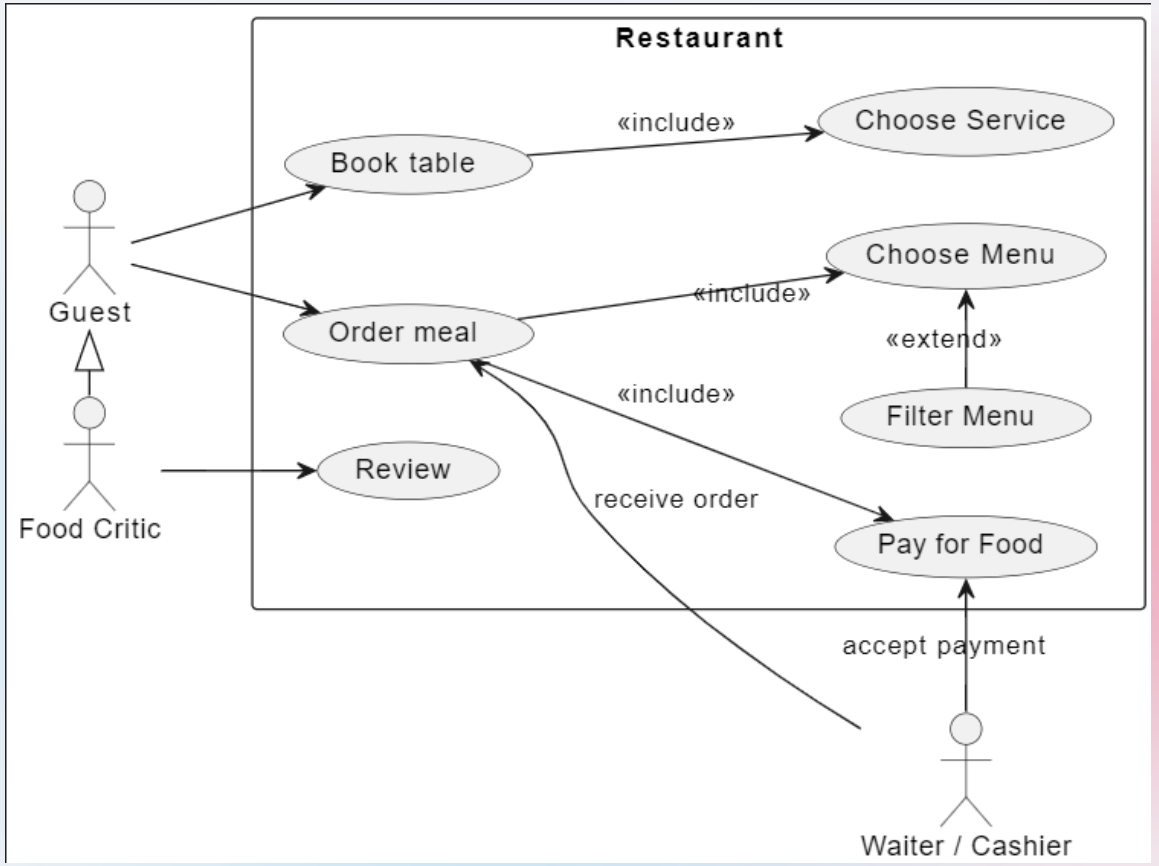
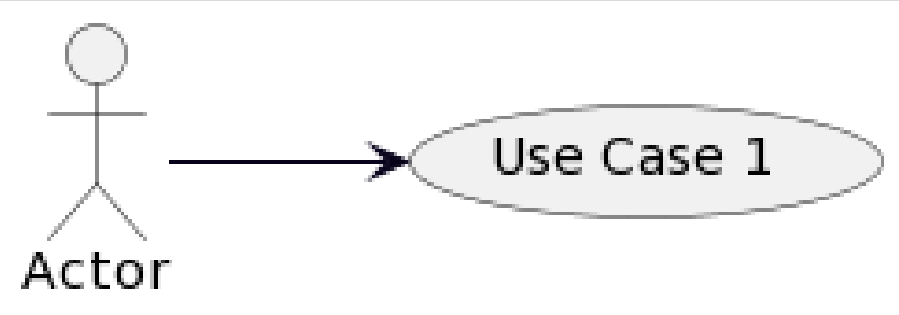


Diagramme de cas d'utilisation

Syntaxe

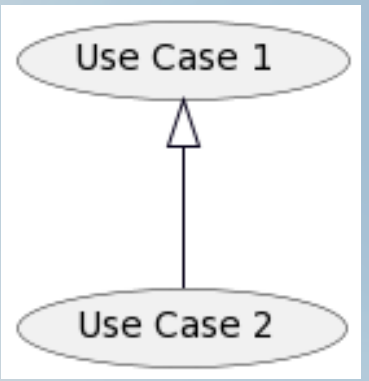
Déclenchement



Pré-requis

Prolongement

Héritage



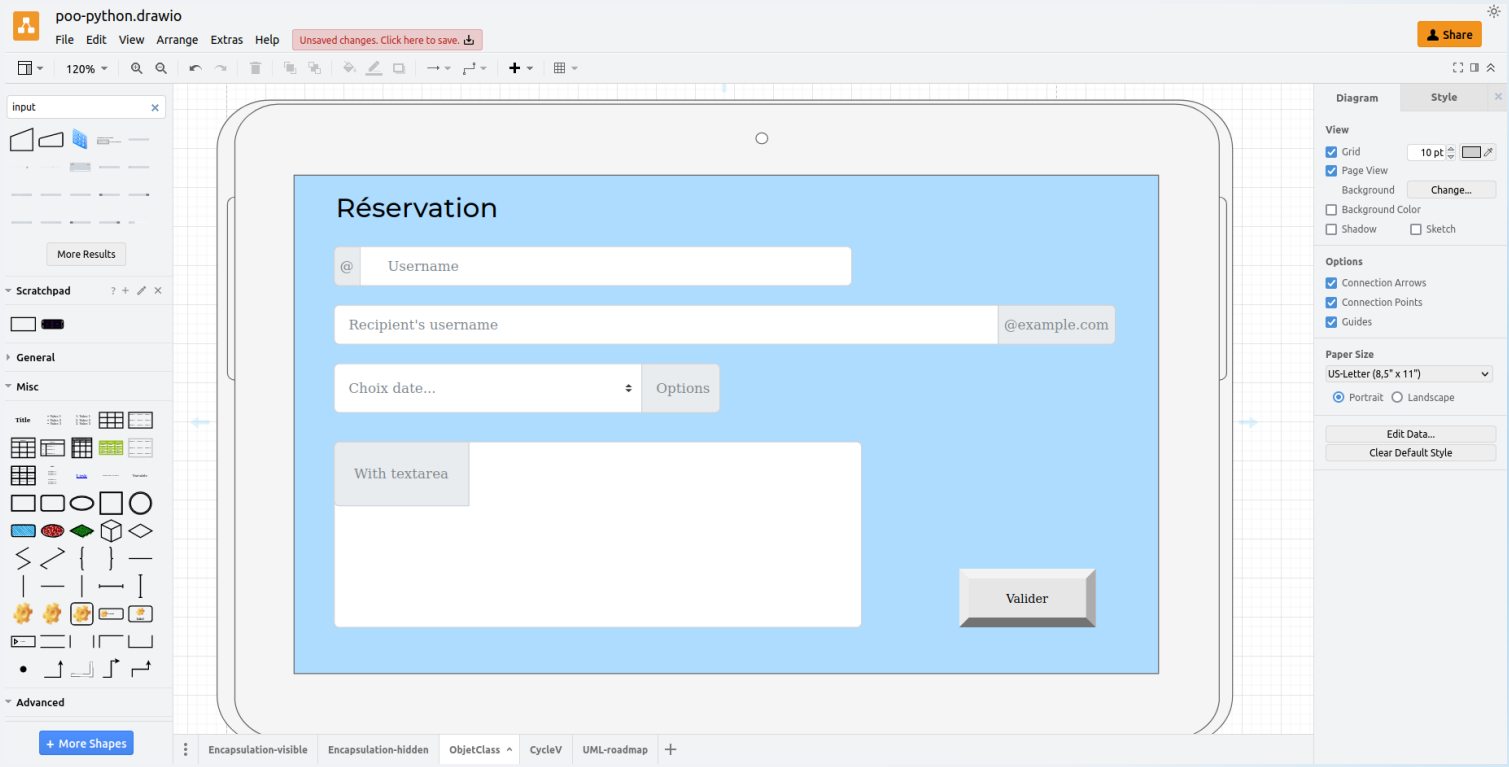
04

Maquettes

Réaliser les premières maquettes

Rien de tel que quelques maquettes pour mettre tout le monde d'accord sur l'interface homme-machine et ses interactions.

Outils: drawio et Figma



05

Diagrammes de classes

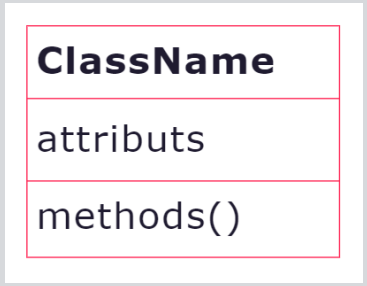
Diagramme de classes

Syntaxe 1

Classe

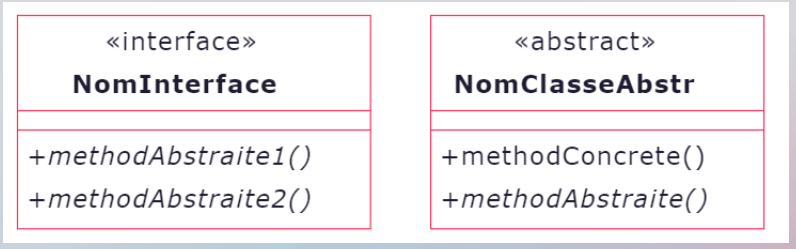
Attributs
[+/#/-] attr : Type

Méthodes
[+/#/-] method(param: Type): ReturnType



+ attributs **publics**
attributs **protégés**
- attributs **privés**

Interface et classes abstraites



Méthodes abstraites en *italic* (ou soulignée)

Héritage

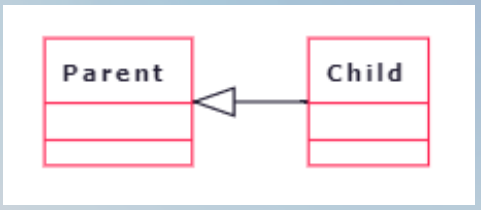


Diagramme de classes

Syntaxe 2

Association

Si deux classes sont en intéractions dans le système on les associent.

- On peut dire: "objet de la classe 1 utilise objet(s) de la classe 2"

On peut préciser la multiplicité de chaque côté de la branche.

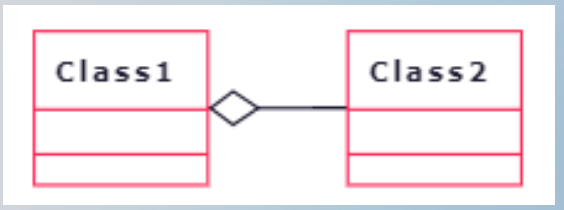
- `1`, `0..1`, `*`, `1..*`
- `n`, `0..n`, `1..n`

Aggrégation / Composition

Ce sont des associations particulières. Les deux classes ne sont pas au même niveau : une classe contient l'autre.

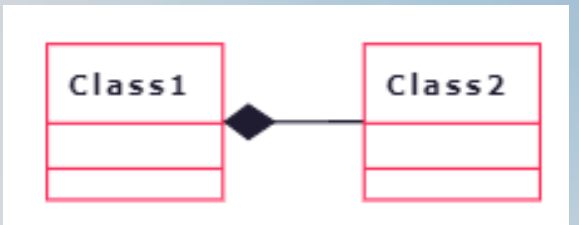
- On peut dire: "objet de la classe 1 contient objet(s) de la classe 2"

Aggrégation



Class1 détruite → *Class2* persiste

Composition



Contient physiquement *Class1* détruite → *Class2* détruite

Diagramme de classes

Aller plus loin

Pour mieux comprendre les relations entre classes et leur traduction en programmation, vous pouvez lire ce billet.

Modèle du domaine

Phase

Début de conception générale. Intervient juste après les premières maquettes et cas d'utilisation.

Objectifs

- Premier diagramme de classes à réaliser
- Indépendant des fonctionnels de l'application
- Représente le domaine métier

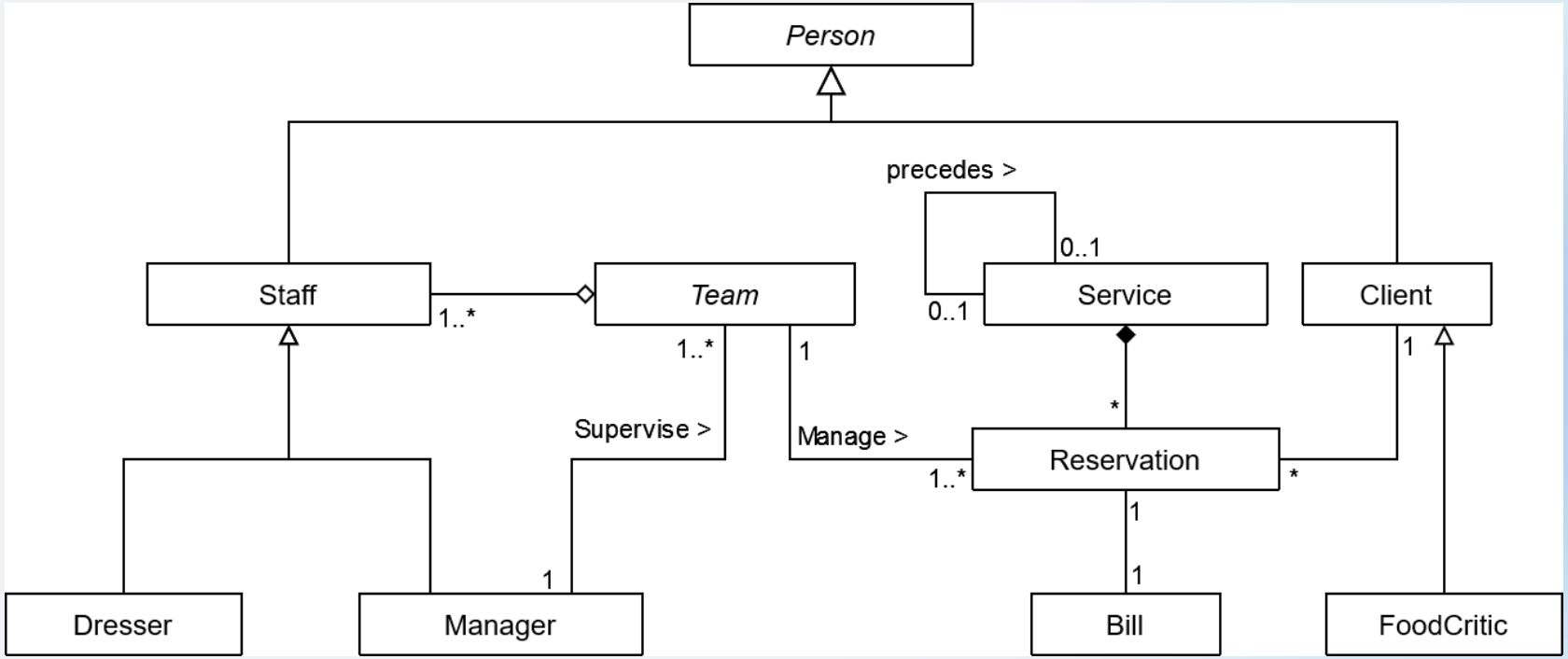


Diagramme de classes participantes

Phase

Fin de conception générale. Intervient dans la dernière phase de la conception générale en même temps que les diagrammes de séquence et d'activité.

Objectifs

- Enrichissement du modèle de domaine
- Modélisation guidée par les besoins

Diagramme de classes participantes