Centre universitaire
Abdelhafid Boussouf
Mila

Faculté des sciences et de la technologie

Département math et informatique

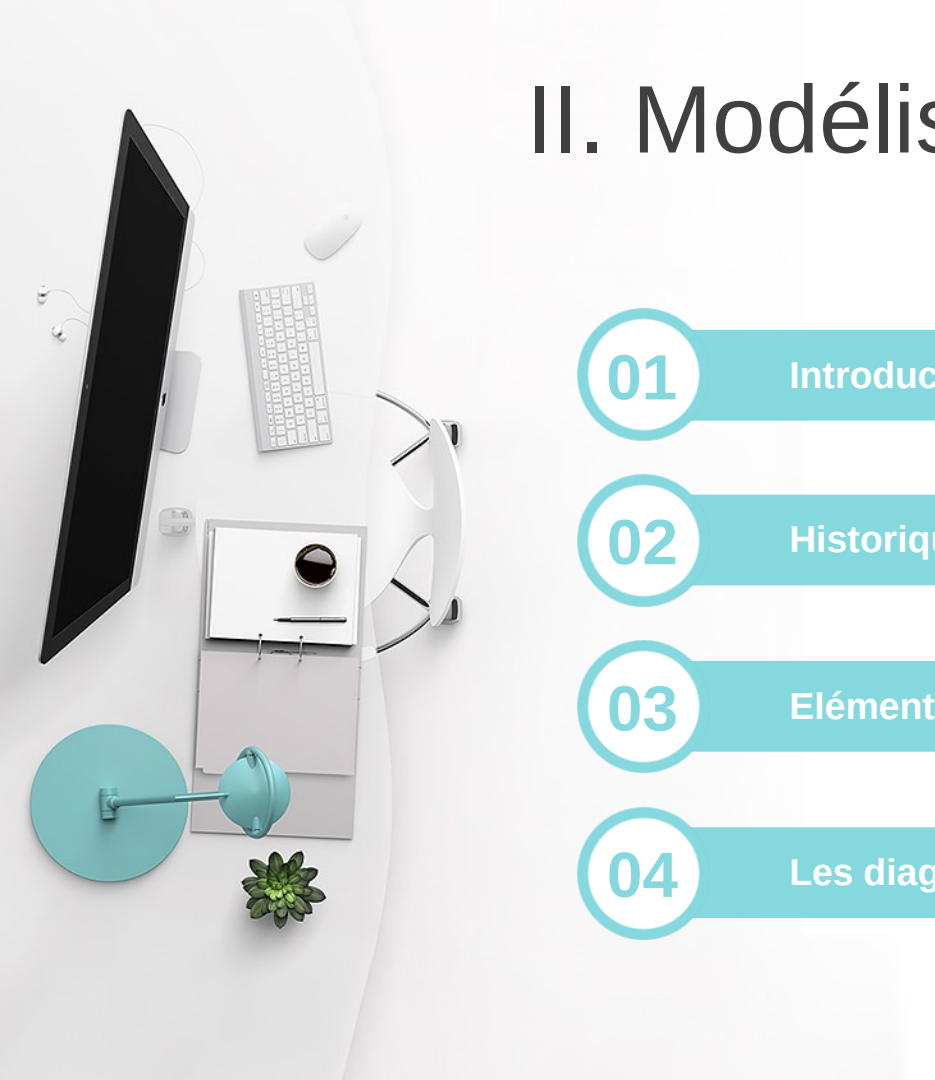
Génie logiciel

Chapitre 2

Modélisation avec UML

Mme. S.HEDJAZ

2023/2024



II. Modélisation avec UML

01 Introduction

02 Historique

03 Eléments et mécanismes généraux

04 Les diagrammes UML

01 Introduction

Modélisation:

L'action de concevoir **un modèle** dans un langage de modélisation dédié. Modéliser consiste à identifier les caractéristiques intéressantes ou pertinentes d'un système dans le but de pouvoir l'étudier du point de vue de ses caractéristiques.

Qu'est ce qu'un modèle ?:

- Une abstraction permettant de mieux comprendre un objet complexe (représenter le système selon des degrés différents de détails)
- Une représentation simplifié d'une réalité (logiciel, bâtiment, … etc.)
- Les modèles sont souvent graphiques (un petit dessin c'est vaut mieux qu'un long discours)

Bon modèle:

- ✓ Il doit faciliter la compréhension du phénomène (système) étudié
- ✓ Il doit permettre de simuler le phénomène (système) étudié

01 Introduction

Pourquoi modéliser ?

- Faciliter la compréhension d'un système
- Permettent également une bonne communication avec le client
- Permettent d'écrire avec précision et complétude les besoins sans connaitre les détails du systèmes
- Permettent de se focaliser sur des aspects spécifiques d'un système (offres différentes visions)
- Permettent de tester une multitude de solutions à moindre coût et dans des délais réduits

01 Introduction

Méthode de modélisation:

Une méthode définit une démarche reproductible pour obtenir des résultats fiable. Donc, une méthode d'élaboration d'un logiciel décrit comment **modéliser** et construire des systèmes logiciels de manières fiable.

Les méthodes de modélisation fonctionnelles:

Les méthodes fonctionnelles et systémiques sont imposées les premiers (les années 70_80), ces méthodes s'inspirent directement de l'architecture des ordinateurs. La séparation entre les données et le traitement telle qu'elle existe physiquement dans le matériel à transposée vers les méthodes.

01 Introduction

Inconvénients:

- Une simple mise à jour du logiciel à un point données peut impacter en cascade une multitude d'autres fonctions
- L'évolution majeur du logiciel multiple les points de maintenance (le logiciel doit être retouché en globalité)

Solutions:

- Centraliser les données d'un type et les traitements associés dans une même entité physique permet de limiter les points de maintenance dans le code

Solution → Les méthodes orienté objet (OO)

01 Introduction

Concepts importants de l'approche objet:

L'objet: est une entité autonome, qui regroupe un ensemble de propriétés cohérentes et de traitements associés. Une telle entité constitue le concept fondateur de l'approche du même nom.

L'encapsulation: consiste à masquer les détails d'implémentation d'un objet, en définissant une interface. L'interface définit les services accessibles (offerts) aux utilisateurs de l'objet.

L'encapsulation facilite l'évolution d'une application car elle stabilise l'utilisation des objets : on peut modifier l'implémentation des attributs d'un objet sans modifier son interface.

01 Introduction

Concepts importants de l'approche objet:

Héritage: est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe. L'héritage évite la duplication et encourage la réutilisation.

Polymorphisme: représente la faculté d'une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve. Ainsi, une opération définie dans une superclasse peut s'exécuter de façon différente selon la sous-classe où elle est héritée. Le polymorphisme augmente la généricité du code.

L'agrégation: il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe. L'agrégation permet d'assembler des objets de base, afin de construire des objets plus complexes.

02 Historique

Méthode BOOCH:

Distingue 2 niveaux:

- **Logique**
  - ✓ Diagrammes de classes
  - ✓ Diagramme d'instances
  - ✓ Diagramme états/transitions
- **Physique**
  - ✓ Diagrammes de modules (principe des packages)
  - ✓ Diagramme de processus

Grady Booch

02 Historique

OMT (Object Modeling Techninque):

Trois axes:

- **Statique** : identifie les propriétés des objets et leurs liaisons avec les autres objets

- **Dynamique** : définit le cycle de vie des objets : comportement des objets, les différents états par lesquels ils passent, et les événements déclenchant ces changements d'états

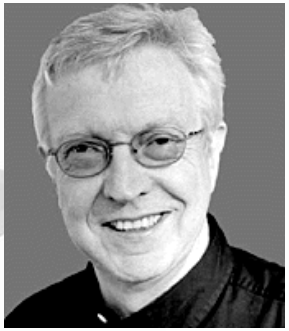
- **Fonctionnel** : précise les fonctions réalisées par les objets par l'intermédiaire des méthodes.

James rumbaugh

02 Historique

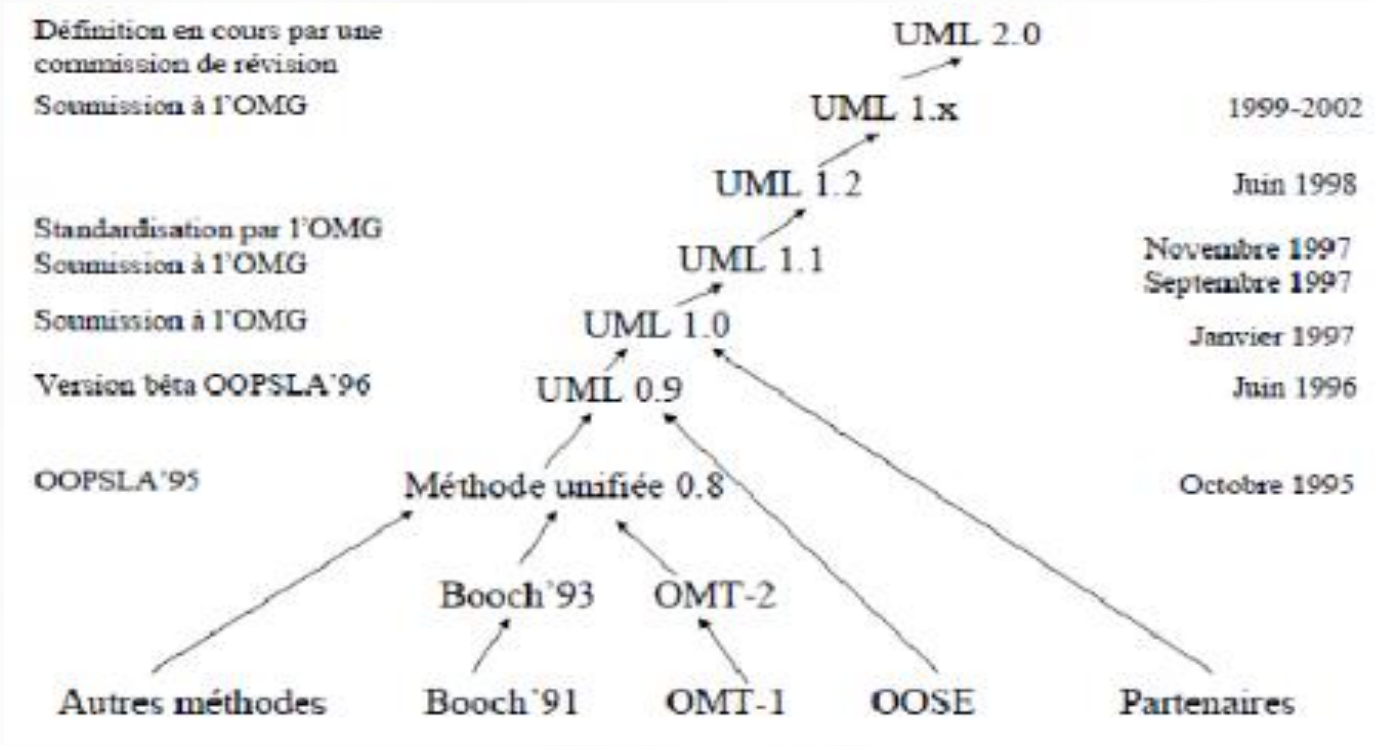
OOSE (Object Oriented Software Engineering):

- ✓ **Cinq modèles:**
  - Description des besoins
  - Analyse
  - Conception
  - Implantation
  - Test
- ✓ **3 types d ’objets**
  - Entités
  - Contrôles
  - Interfaces
- ✓ **Notion de Cas d’Utilisation: Use Cases**



Ivar Jacobson

02 Historique



03 Elements et mécanismes généraux

Pourquoi UML?

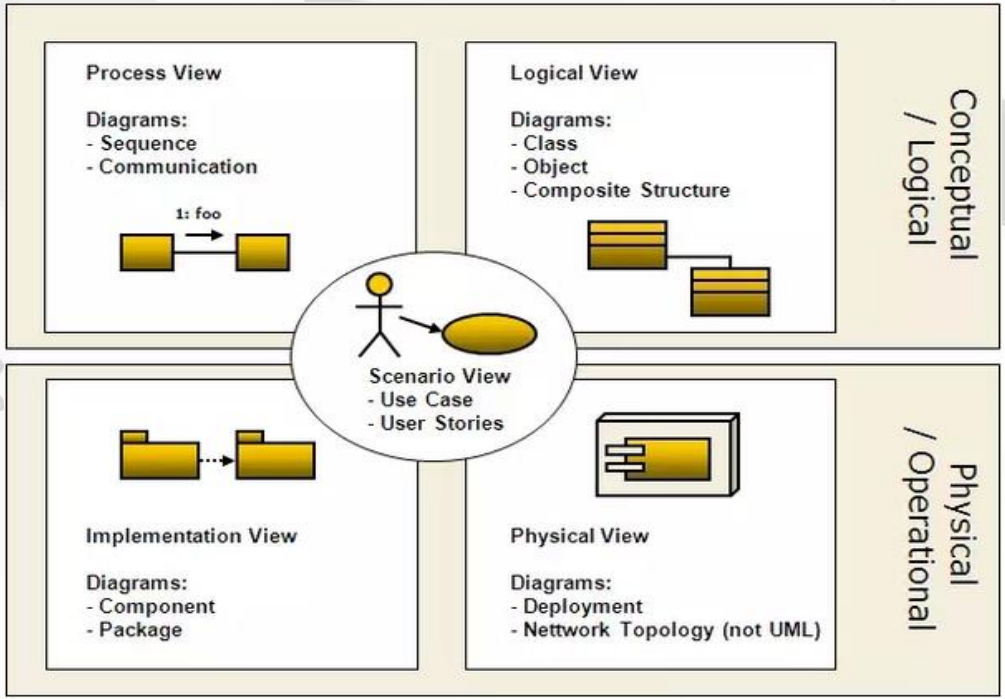
- Graphique et simple
- Uml est un standard
- Représenter des systèmes entiers
- Langage commun qui est utilisable par toutes les méthodes et adapté à touts les phases de développement

UML n’est pas une méthode, c’est un langage de modélisation objet

- Itératif et incrémental
- Guidé par les besoins des utilisateurs
- Centré sur l’architecture

03 Elements et mécanismes généraux

Architecture 4+1



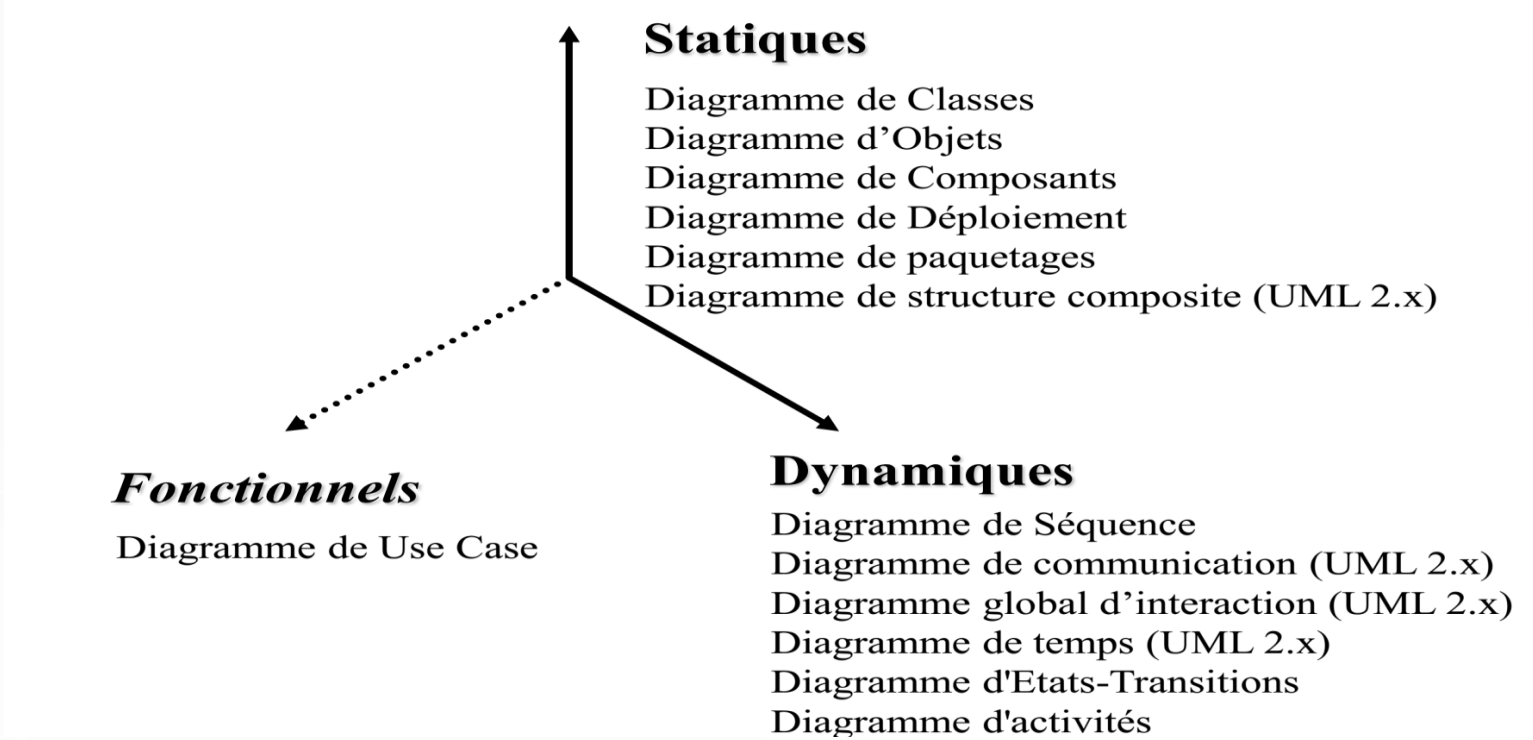
03 Elements et mécanismes généraux

Architecture 4+1

- **La vue des cas d'utilisation** : c'est la description du modèle « vue » par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur
- **La vue logique** : cette vue exprime la perspective abstraite de la solution en termes de classes, d'objets, de relations, de machine à états de transition etc..
- **La vue d'implémentation ou composant**: cette vue définit les dépendances entre les modules
- **La vue de comportement ou des processus** : c'est la vue temporelle et technique, qui met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation, etc...
- **La vue de déploiement** : cette vue décrit la position géographique et l'architecture physique de chaque élément du système

03 Elements et mécanismes généraux

Les axes de modélisation avec UML:



04 Les diagrammes UML

1. Diagramme de cas d’utilisation:

Il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.

2. Diagramme de classe:

Il représente les classes intervenant dans le système.

3. Diagramme d’objet:

Il sert à représenter les instances de classes (objets) utilisées dans le système.

04 Les diagrammes UML

4. Diagramme de composant:

Il permet de montrer les composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données...)

5. Diagramme de déploiement:

Il sert à représenter les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.

6. Diagramme de paquetages:

Un paquetage étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML. Le diagramme de paquetage sert à représenter les dépendances entre paquetages, c'est-à-dire les dépendances entre ensembles de définitions.

04 Les diagrammes UML

7. Diagramme de structure composite :

Permet de décrire sous forme de boîte blanche les relations entre composants d'une classe.

8. Diagramme de séquence:

Représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs

9. Diagramme de communication:

Représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets

04 Les diagrammes UML

10. Diagramme global d'interaction:

Permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activités)

11. Diagramme de temps:

Permet de décrire les variations d'une donnée au cours du temps

12. Diagramme d'états-transitions :

Permet de décrire sous forme de machine à états finis le comportement du système ou de ses composants

13. Diagramme d'activité :

Permet de décrire sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants

Bibliographies

- **Uml 2 pratique de la modélisation**, Benoît Charroux, Yann Thierry-Mieg, Aomar Osmani https://fr.slideshare.net/nassimamine3994/uml-2-pratique-de-la-modlisation
- **Uml 2 par la pratique, Pascal roques**
- **Les cahiers du programmeur, Pascal roques**
- …