Chapitre 3 - GL

Langage UML

Présenté par : Dr. Lamia BERKANI

Section :L3 ACAD « B »
Année: 2024-2025

Pourquoi et comment modéliser ?

- Modèle : représentation abstraite et simplifiée d'une entité du monde réel en vue de le décrire, de l'expliquer

- Cette représentation peut être sous forme physique, graphique, mathématique ou verbale.

Intérêt "informatique" de la modélisation

- Les modèles aident à visualiser un système existant ou futur (tel que l'on souhaite qu'il devienne)
- Les modèles permettent de spécifier la structure et le comportement d'un système
- Les modèles documentent les choix effectués

Langages de modélisation

Un langage de modélisation doit définir :

- La sémantique des concepts (sens);
- Une notation pour la représentation de concepts;
- Des règles de construction et d'utilisation des concepts.

Langages de modélisation

Des langages à différents niveaux de formalisation (modélisation):

- Langages formels (Z,B,VDM) : le plus souvent mathématiques, au grand pouvoir d'expression et permettant des preuves formelles sur les spécifications;

- Langages semi-formels (MERISE, UML…) : le plus souvent graphiques, au pouvoir d'expression moindre mais plus faciles d'emploi.

Langages de modélisation

Le développement du logiciel dispose de nombreux langages de modélisation :

- Adaptés aux systèmes procéduraux (MERISE…);
- Adaptés aux systèmes temps réel (ROOM, SADT…);
- Adaptés aux systèmes à objets (OMT, Booch, **UML**, …).

Historique des modélisations par objets

- Années 80: méthodes fondées sur la modélisation séparée des données et des traitements;
- 1990-1995 : plus de 50 méthodes apparaissent (Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE, etc.) mais aucune ne parvient à s'imposer
- En 1994, un **consensus** se fait autour de trois méthodes dont la fusion a donné naissance au langage **UML (Unified Modelling Language)**:
  - OMT (Object Modeling Technique -Technique de Modélisation Objet);
  - Booch (par Grady Booch) qui introduit le concept de package (élément d'organisation des modèles);
  - OOSE (Object Oriented Software Engineering) inventée par Ivar Jacobson fonde l'analyse sur la description des besoins des utilisateurs

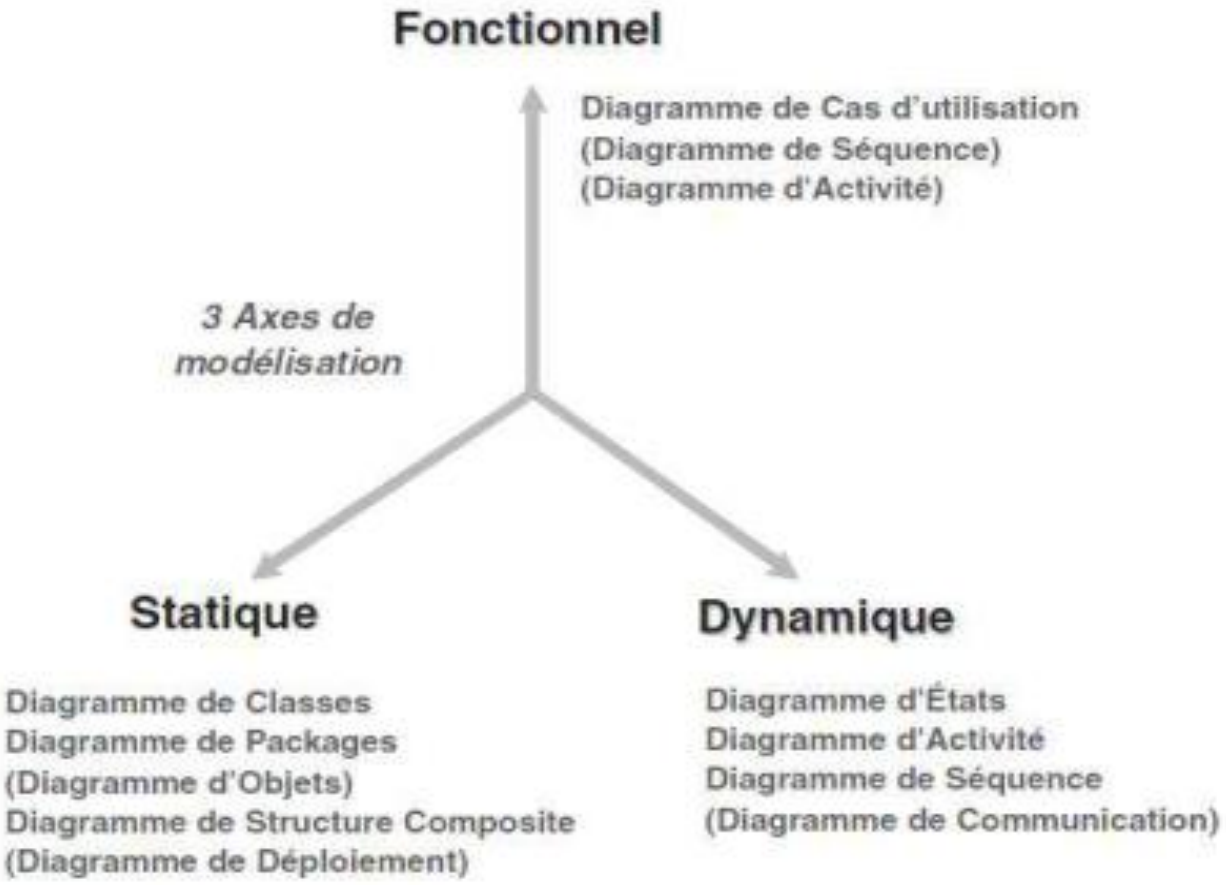
Historique des modélisations par objets

- 1997 : l'OMG (Object Management Group) accepte UML, proposé par Rational, comme standard de modélisation objet.
- 2001 : révision par l'OMG d'UML 1.
- 2004 : adoption d'UML 2.0
- Dernière version validée par OMG est celle de 2017

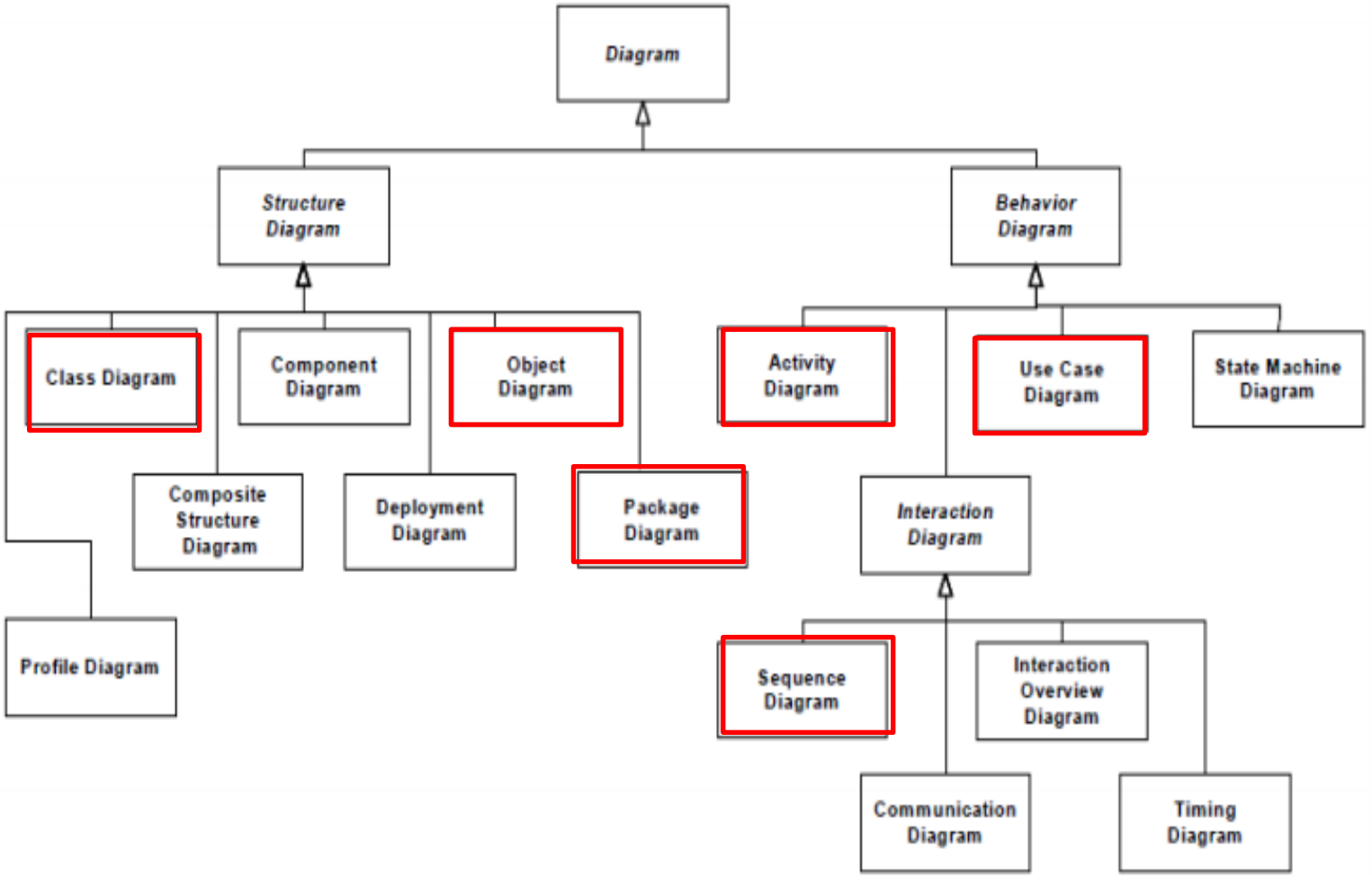
  https://www.omg.org/spec/UML/

  Dernière version UML 2.5.1

Modélisation UML



Diagrammes UML



Diagrammes comportementaux

Diagramme de cas d'utilisation: (Use case diagram)

- utilisé dans l'activité de spécification des besoins.
- présente les interactions fonctionnelles entre les acteurs et le système
  - présente les fonctions du système du point de vue des utilisateurs).

Diagramme d'états (state machine diagram):

- représente le cycle de vie commun aux objets d'une même classe;
- montre les différents états et transitions possibles des objets d'une classe à l'exécution.

Diagrammes comportementaux

Diagramme d'activité:

- adapté à la modélisation du cheminement de flots de contrôle et de flots de données.

- utilisé pour afficher la séquence des activités, en présentant les règles d'enchaînement des actions et décisions au sein d'une activité ou le comportement d'un cas d'utilisation.

Diagrammes comportementaux

Diagrammes d'interactions - Diagrammes de séquence: modélisation des scénarios

- représente la succession chronologique des opérations réalisées par un acteur,

- indique les objets qu'il va manipuler et les opérations qui sont passées d'un objet à l'autre.

Diagrammes structurels

Diagramme de classes:

- permet en analyse de décrire la structure statique des entités manipulées par les utilisateurs (classes, associations, attributs, opérations, interfaces, etc.); et

- en conception, permet de représenter la structure d'un code orienté objet.

Diagrammes structurels

Diagramme d'objets:

- permet de représenter les instances des classes, i.e. des objets.

- exprime les relations qui existent entre les objets, mais aussi l'état des objets, ce qui permet d'exprimer des contextes d'exécution.
  - i.e. représente une photographie à un instant précis des attributs et objet existants.

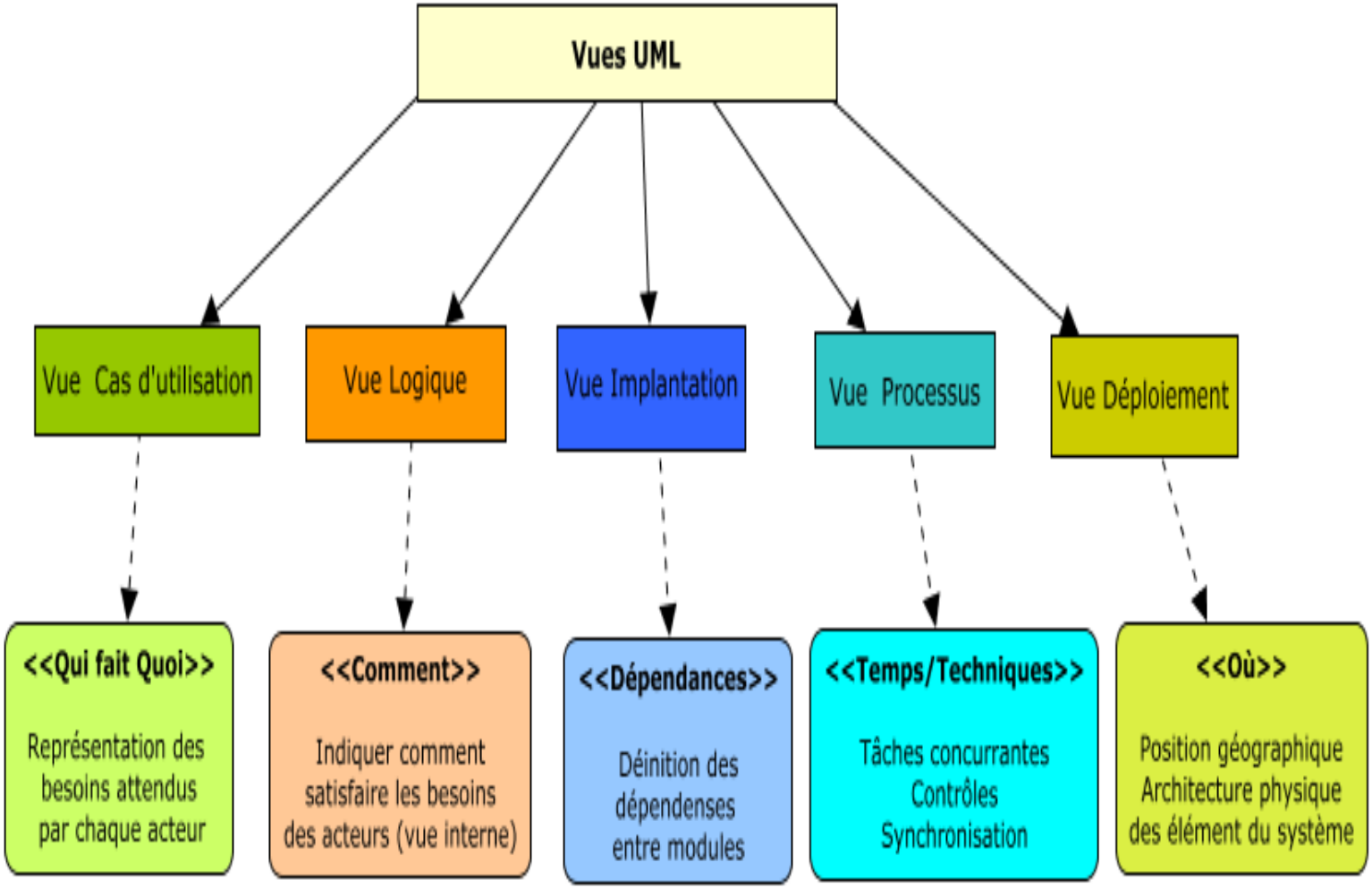
Diagrammes structurels

Diagramme de packages:

- montre l'organisation logique du modèle et les relations entre packages;

- permet de structurer les classes d'analyse et de conception, ainsi que les cas d'utilisation.

VUES D'UML


Vues UML
Vue Cas d'utilisation
Vue Logique
Vue Implantation
Vue Processus
Vue Déploiement
<<Qui fait Quoi>>
Représentation des besoins attendus par chaque acteur
<<Comment>>
Indiquer comment satisfaire les besoins des acteurs (vue interne)
<<Dépendances>>
Déinition des dépendenses entre modules
<<Temps/Techniques>>
Tâches concurrantes
Contrôles
Synchronisation
<<Où>>
Position géographique
Architecture physique
des élément du système