UNIVERSITE ASSANE SECK DE ZIGUINCHOR
UFR Sciences et Technologies
****
DEPARTEMENT D'INFORMATIQUE

Modélisation Orientée Objet
Unified Modeling Language (UML)

L2 Informatique Ingénierie
2021-2022

© Bassirou DIENE

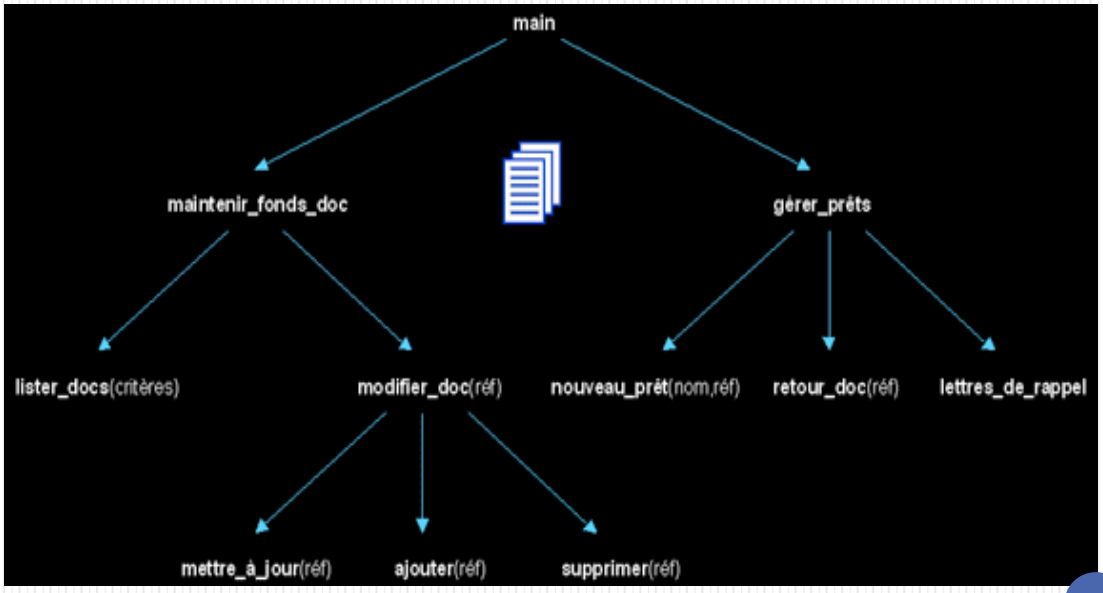
APPROCHE FONCTIONNELLE VS APPROCHE OBJET 1/8

- Découpe fonctionnelle (ou structuré) d'un problème informatique: approche intuitive
  - Le logiciel est composé:
    - d'une **hiérarchie de fonctions** qui fournissent les services désirés
    - de **données** qui représentent les éléments manipulés (livre, etc.)
  - Avantages
    - Approche logique cohérente et intuitive
    - Permet de factoriser les comportements

Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 2/8

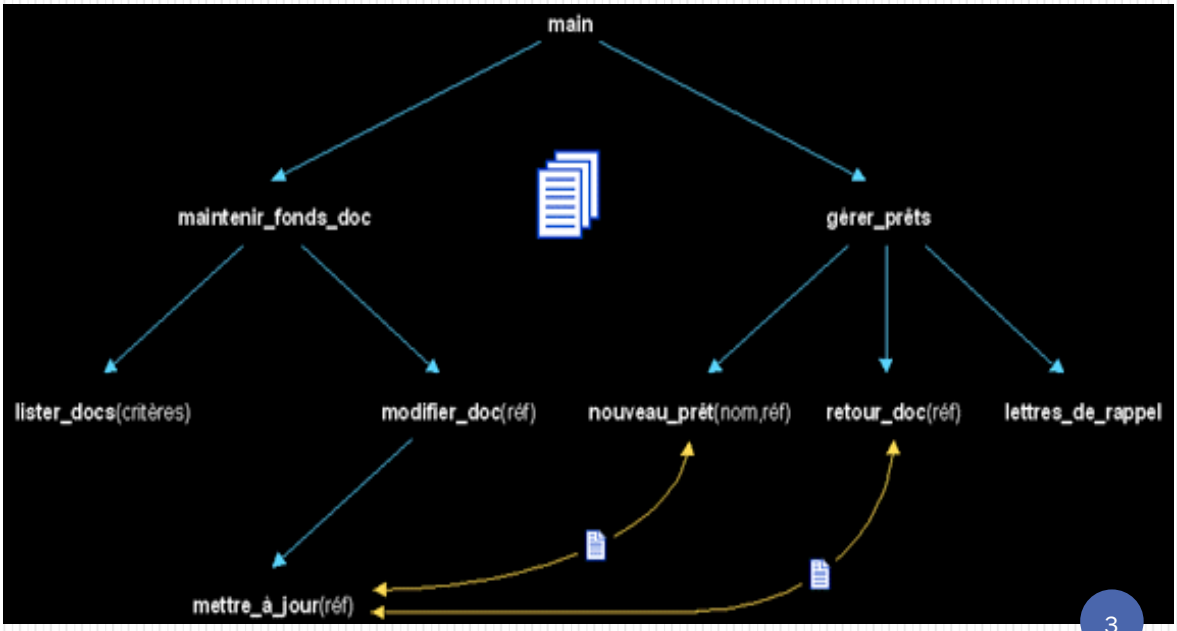
- Exemple de découpe fonctionnelle d'un logiciel dédié à la gestion d'une bibliothèque



Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 3/8

- Découpe fonctionnelle "intelligent" → Factorisation de comportement



Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 4/8

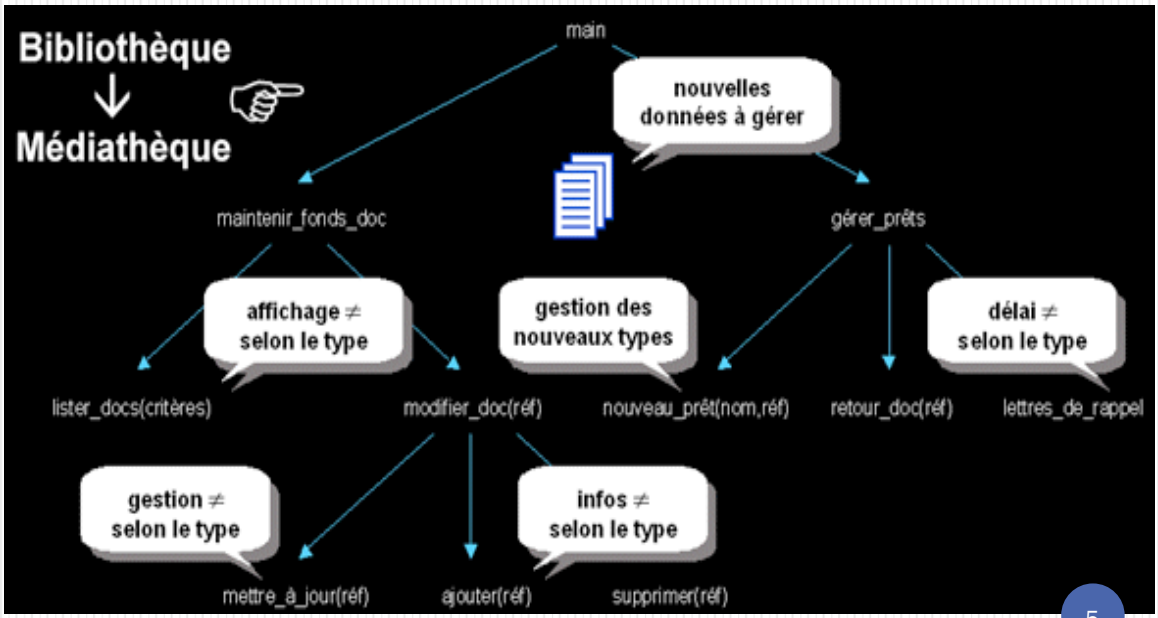
❑ Inconvénients de la factorisation de comportement

- Fonctions interdépendantes : une simple mise à jour du logiciel à un point donné, peut impacter en cascade une multitude d'autres fonctions.
  - Une modification des paramètres de la fonction mettre_à_jour entraîne une modification de la fonction modifier_doc.
- En cas d'évolution majeure du logiciel, la multiplication des points de maintenance, engendrée par le chaînage des fonctions, rend l'adaptation très laborieuse.

Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 5/8

- Exemple d'évolution du logiciel



Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 6/8

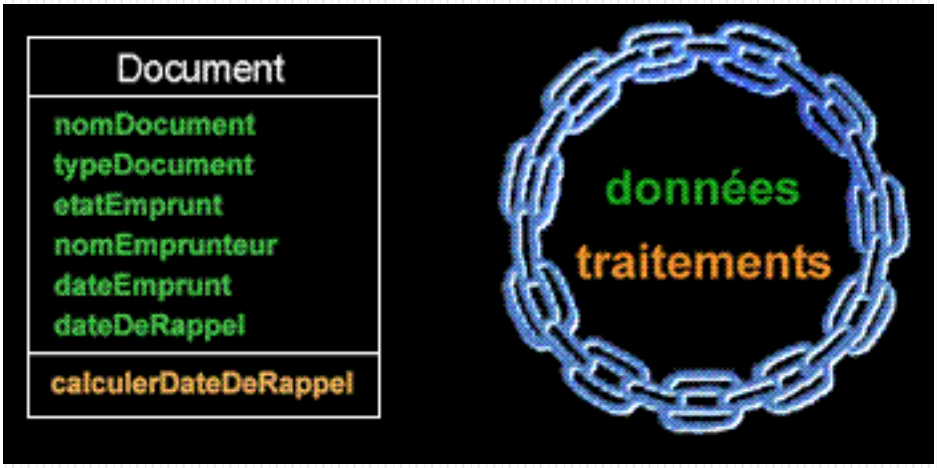
- **Solution** → approche objet
  - Rassembler les valeurs qui caractérisent un type, dans le type
  - Centraliser les traitements associés à un type, auprès du type

- **Avantages**
  - Réduire les points de maintenance
    - L'ajout d'un nouveau type n'impacte pas sur les données déjà existant ainsi que leur traitement
    - Un nouveau type vient avec ses propres traitements
    - Les fonctions à modifier sont celles qui utilisent le nouveau type
  - Retrouver plus rapidement le bout de code qui est chargé de traiter les données, puisqu'il se trouve au plus près de la structure de données concernée.

Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 7/8

- Exemple



Unified Modeling Language

APPROCHE FONCTIONNELLE VS APPROCHE OBJET 8/8

- Récapitulons
  - Approche fonctionnelle
    - Hiérarchie de fonctions fournissant des services + données
    - Le système a un état partagé, qui est centralisé et accessible par l'ensemble des fonctions
    - Maintenance laborieuse
  - Approche objet
    - Ensemble d'objets (données + traitements) qui interagissent
    - Chaque objet dispose d'un ensemble d'attributs décrivant son état et l'état du système est décrit de façon décentralisée par l'état de l'ensemble.
    - Maintenance plus facile

Unified Modeling Language

MODELISATION OBJET

- Principe:
  - Le système informatique est modélisé comme une collection d'**objets**, avec leurs propriétés et leurs comportements, qui collaborent entre eux.

- Objectif principal: réduire et gérer la complexité des logiciels:
  - Décomposition modulaire.
  - Regroupement des fonctions et des propriétés concernant un type donné dans un module.
  - Cacher la complexité des fonctions et celle de leurs actions.
  - Fournir une interface qui sera la partie visible du module.
  - Communication par envoi de messages.

Unified Modeling Language

CONCEPTS DE L'ORIENTÉ OBJET (1/5)

- **Objet:** représente une entité physique, logicielle, ou conceptuelle.
  - Exemples: un client, un logiciel, une voiture.

- Un objet possède:
  - Une **identité**: permet de distinguer chaque objet par rapport aux autres.
  - un **état**: correspond aux valeurs de tous ses attributs à un instant donné.
  - un **comportement**: ensemble des opérations (méthodes) qui définissent les réactions de l'objet; opérations exécutées en réaction aux messages provenant des autres objets.

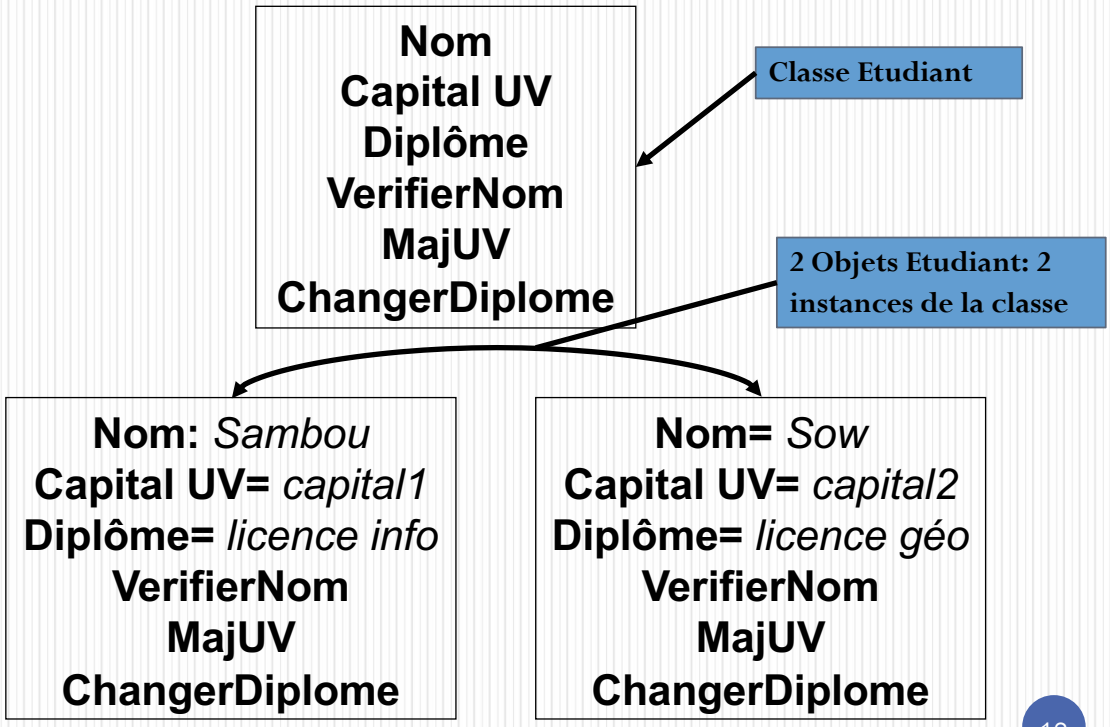
Unified Modeling Language

CONCEPTS DE L'ORIENTÉ OBJET (2/5)

- **Classe:** description abstraite d'un ensemble d'objets possédant une structure identique (liste des attributs) et un même comportement (liste des opérations).
  - Un objet est une **instance** de classe (une occurrence de type abstrait).
- L'**instanciation** est la création d'un objet d'une classe.
- 2 instances d'une même classe peuvent avoir des attributs avec des valeurs différentes mais partagent les mêmes méthodes.

Unified Modeling Language

CONCEPTS DE L'ORIENTÉ OBJET (3/5)



Unified Modeling Language

CONCEPTS DE L'ORIENTÉ OBJET (4/5)

- **Encapsulation:** technique consistant à regrouper les données et les méthodes d'un objet et à masquer les détails de leur implémentation.
- **Interface:** vue externe d'un objet, définit les services accessibles aux utilisateurs de l'objet.
- **Héritage:** mécanisme de transmission des propriétés d'une classe (attributs et méthodes) vers une sous classe.
  - Évite la duplication et favorise la réutilisation.

CONCEPTS DE L'ORIENTÉ OBJET (5/5)

- **Généralisation:** factorisation des éléments communs de classes (attributs, méthodes)
- **Spécialisation:** adaptation d'une classe générale à un cas particulier.
- **Polymorphisme:** faculté d'une méthode de s'exécuter différemment suivant le contexte et le type de la classe où elle se trouve.
  - Augmente la généricité du code.
- **Agrégation**: relation entre plusieurs classes, spécifiant qu'une classe est composée d'une ou plusieurs autres classes.

L'APPROCHE OBJET: PAS TOUJOURS FACILE

- L'approche objet est moins intuitive que l'approche fonctionnelle
  - Quels moyens utiliser pour faciliter l'analyse objet?
  - Quels critères identifient une conception objet pertinente?
  - Comment comparer deux solutions de découpe objet d'un système?
- L'application des concepts objets nécessite une grande rigueur!
  - Le vocabulaire est précis (risques d'ambiguïtés, d'incompréhensions).
  - Comment décrire la structure objet d'un système de manière pertinente?
- Pas facile à prendre en main
  - Période d'adaptation assez longue pour un esprit cartésien

Unified Modeling Language

LES REMÈDES AUX INCONVÉNIENTS DE L'APPROCHE OBJET

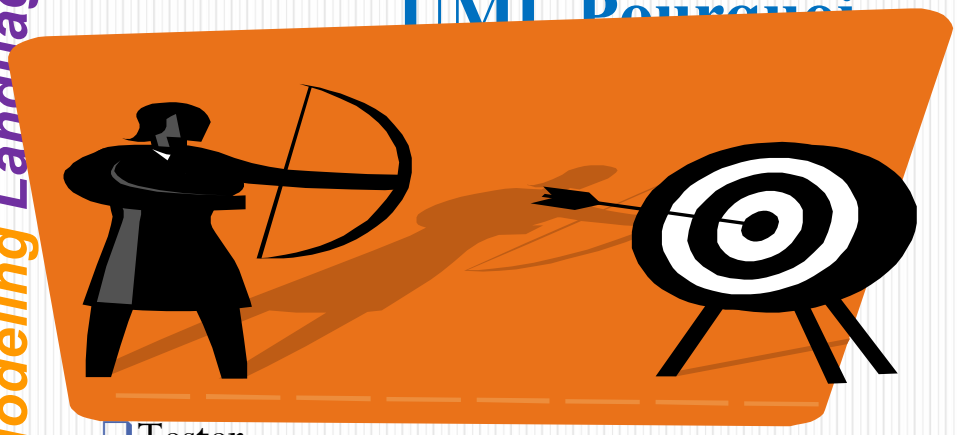
- **Un langage pour exprimer les concepts objet qu'on utilise, afin de pouvoir:**
  - Représenter des concepts abstraits (graphiquement par exemple).
  - Limiter les ambiguïtés (parler un langage commun).
  - Faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions).

- **Une démarche d'analyse et de conception objet, pour:**
  - Ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, mais penser objet dès le départ.
  - Définir les vues qui permettent de couvrir tous les aspects d'un système, avec des concepts objets.

Unified Modeling Language (UML)

Unified Modeling Language

UML Pourquoi



- Tester
- Auditer

18

Unified Modeling Language

Comment modéliser avec UML

- UML est un langage permettant de représenter des modèles
  - Mais ne définit pas le processus d'élaboration des modèles!
- Cependant, dans le cadre de la modélisation d'une application informatique, les auteurs d'UML préconisent d'utiliser une démarche:
  - **itérative et incrémentale,**
  - **guidée par les besoins des utilisateurs du système,**
  - **centrée sur l'architecture logicielle.**
- D'après les auteurs d'UML, un processus de développement qui possède ces qualités devrait favoriser la réussite d'un projet.

19

Unified Modeling Language

itérative et incrémentale ?

- **Idée très simple:** pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s 'y prendre en plusieurs fois, en affinant son analyse par étapes.
- Cette démarche devrait aussi s'appliquer au cycle de développement dans son ensemble, en favorisant le prototypage.
- Le but est de mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes.

Unified Modeling Language

Démarche pilotée par les besoins des utilisateurs ?

- **Avec UML, ce sont les utilisateurs qui guident la définition des modèles :**
  - Le périmètre du système à modéliser est défini par les besoins des utilisateurs (les utilisateurs définissent ce que doit être le système).
  - Le but du système à modéliser est de répondre aux besoins de ses utilisateurs (les utilisateurs sont les clients du système).
- **Les besoins des utilisateurs servent aussi de fil rouge, tout au long du cycle de développement (itératif et incrémental) :**
  - A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
  - A chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs.
  - A chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

Unified Modeling Language

Démarche centrée sur l'architecture ?

- Une architecture adaptée est la clé de voûte du succès d'un développement.
- Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité…).

Unified Modeling Language

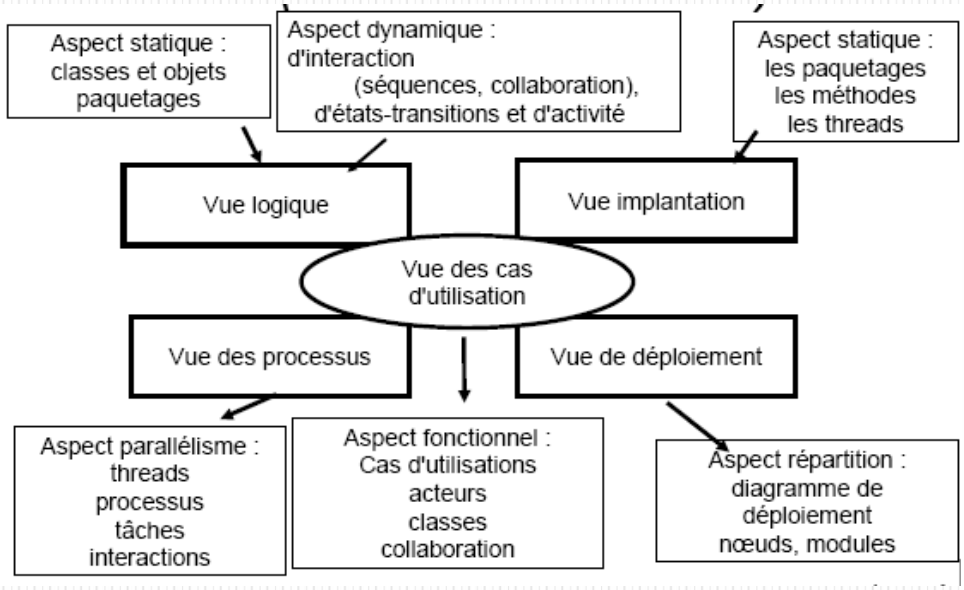
Modéliser avec UML

- UML permet de définir et de visualiser un modèle, à l'aide de diagrammes.
- Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle ; c'est une perspective du modèle, pas « le modèle ».
- Chaque type de diagramme UML possède:
  - une structure (les types des éléments de modélisation qui le composent sont prédéfinis).
  - une sémantique précise (un type de diagramme offre toujours la même vue d'un système).
- Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.

Unified Modeling Language

UML: vues

- Il existe 5 façons de voir le système



Unified Modeling Language

UML: vue logique

- Décrit comment les fonctionnalités du système sont réalisées.
- Décrit la structure statique (classes, objets, relations, etc.).
- Décrit la collaboration dynamique (échange de messages).
- Concerne essentiellement le concepteur et les développeurs.

Unified Modeling Language

UML: vue cas d'utilisation

- Vue la plus importante car le but final est de réaliser les fonctionnalités qui sont spécifiées par cette vue.
- Décrit les fonctionnalités attendues du système telles que perçues par les acteurs externes (utilisateurs, autres systèmes, etc.).
- personnes impliquées dans cette vue : clients, concepteurs, développeurs et testeurs.
- Diagramme essentiel: cas d'utilisation

Unified Modeling Language

UML: vue implantation

- Identifie les modules qui réalisent les classes de la vue logique.
- Montre les dépendances entre modules.
- Montre l'organisation des modules en sous systèmes (paquetages) et leurs dépendances (avec d'autres sous systemes ou modules).
- Concerne les développeurs.

Unified Modeling Language

UML: vue processus

- Concerne la division du système en processus dans un environnement multi tâche.
- Montre les interactions entre les processus.
- Montre la synchronisation et la communication des activités parallèles (threads).
- Destinée aux développeurs et aux intégrateurs du système.

Unified Modeling Language

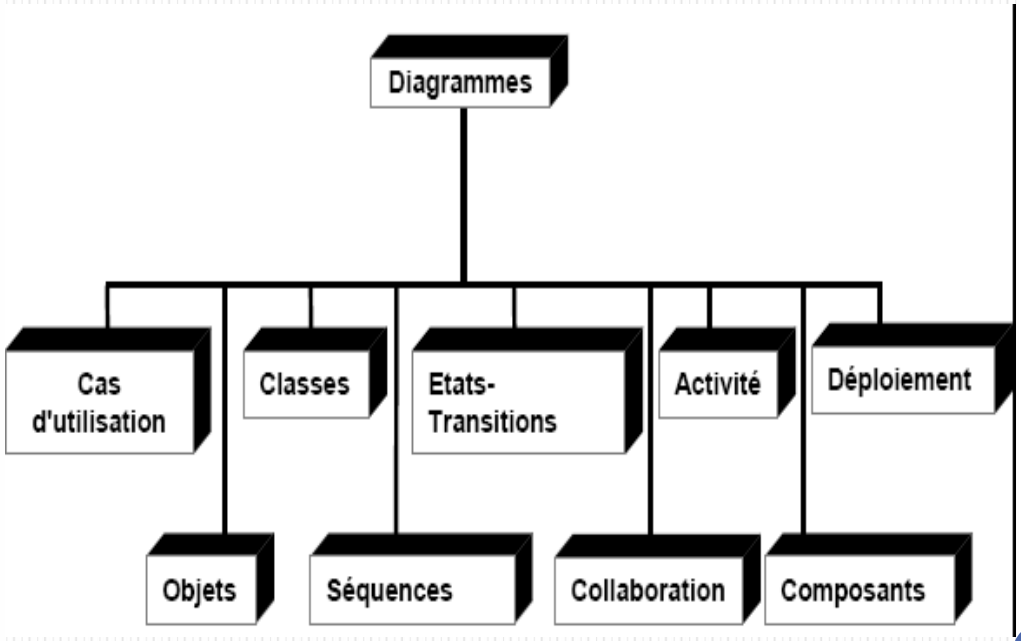
UML: vue déploiement

- Montre le déploiement physique du système
  - Décomposition en noeuds d'exécution
  - Rôle d'un noeud
  - Inter-connectivite, topologie
- Concerne les développeurs, les intégrateurs et les testeurs.

Unified Modeling Language

UML: Diagrammes

- Il existe 9 diagrammes dans UML



Unified Modeling Language

UML: Diagrammes

- **Diagramme de cas d'utilisation**
  - Décrit les fonctionnalités du système telles que perçues par les acteurs externes (utilisateurs, autres systèmes, etc.).
- **Diagramme de classes**
  - Montre la structure statique des classes dans le système.
- **Diagramme d'objets**
  - Montre comment le système est vu à un instant donné dans le temps.

Unified Modeling Language

UML: Diagrammes

- **Diagramme d'états**
  - Montre les états possibles qu'un objet peut avoir en réaction aux événements (envoi/réception d'un message, condition satisfaite, etc.).
- **Diagramme de séquence**
  - Décrit les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.
- **Diagramme de collaboration**
  - Représente des scénarios de cas d'utilisation en mettant plus l'accent sur les objets et les messages échangés.

Unified Modeling Language

UML: Diagrammes

- **Diagramme d'activité**
  - Décrit les activités qui sont exécutées dans une opération ou un cas d'utilisation.
- **Diagramme de composants**
  - Montre la structure physique du code.
- **Diagramme de déploiement**
  - Montre l'architecture physique du matériel et du logiciel dans le système.

FIN