Modélisation orientée Objets et UML

Bobo DIALLO
ambobodiallo@gmail.com

Du fonctionnel à l'objet????

Le Challenge

Modifier, quand on veut

là où l'on veut (local)

sans se soucier d'ailleurs (global)

Du fonctionnel à l'objet????

Le Gain

Réutilisation et/ou extension

efficiente

et simple par contrat (API)

Votre exemple

- À vous de jouer
  - Imaginer un projet
    - Modéliser le avec Merise?
- Groupe de 5

Présenter vos projets

- Donnez les grandes fonctionnalités actuelles
- Comment les faire évoluer en fonction
  - Des nouveaux besoins
  - De l'évolution de l'environnement
- Evaluer le coût de la maintenance

Le découpage fonctionnel: l'inconvénient

coût de la maintenance onéreux

→ Séparation données et traitement: un piège

Ex: Passage d'une bibliothèque en médiathèque

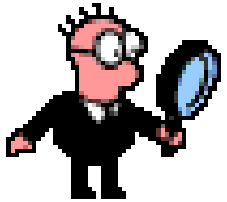
Prise en compte de nouveau type de médias

Des traitements spécifiques de plus à faire

→ Nécessite une révision totale du système

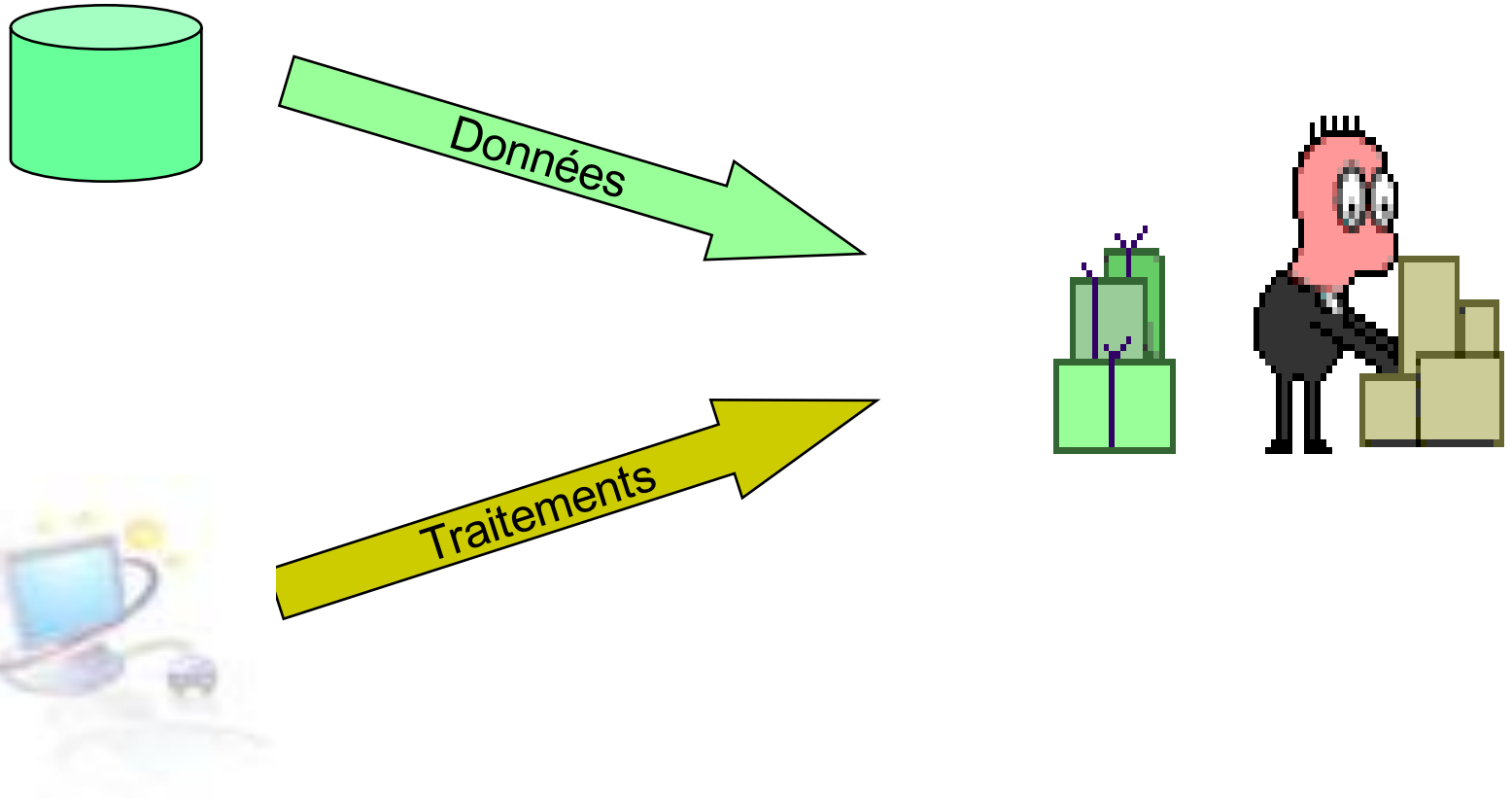
Changer la manière de PENSER ????

Garder en esprit la notion de fonctionnalité!!!



La nouveauté

Réunir données et traitements



Principe de POO

- Principe:
  - Système informatique modélisé comme une collection d'objets
- 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.

Concepts de l'objet (1)

- **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

Concepts de l'objet (2)

- **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)

- **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 le duplication et favorise la réutilisation

Concepts de l'objet (3)

- **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 n’est pas 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

Comment prendre en main l'objet?

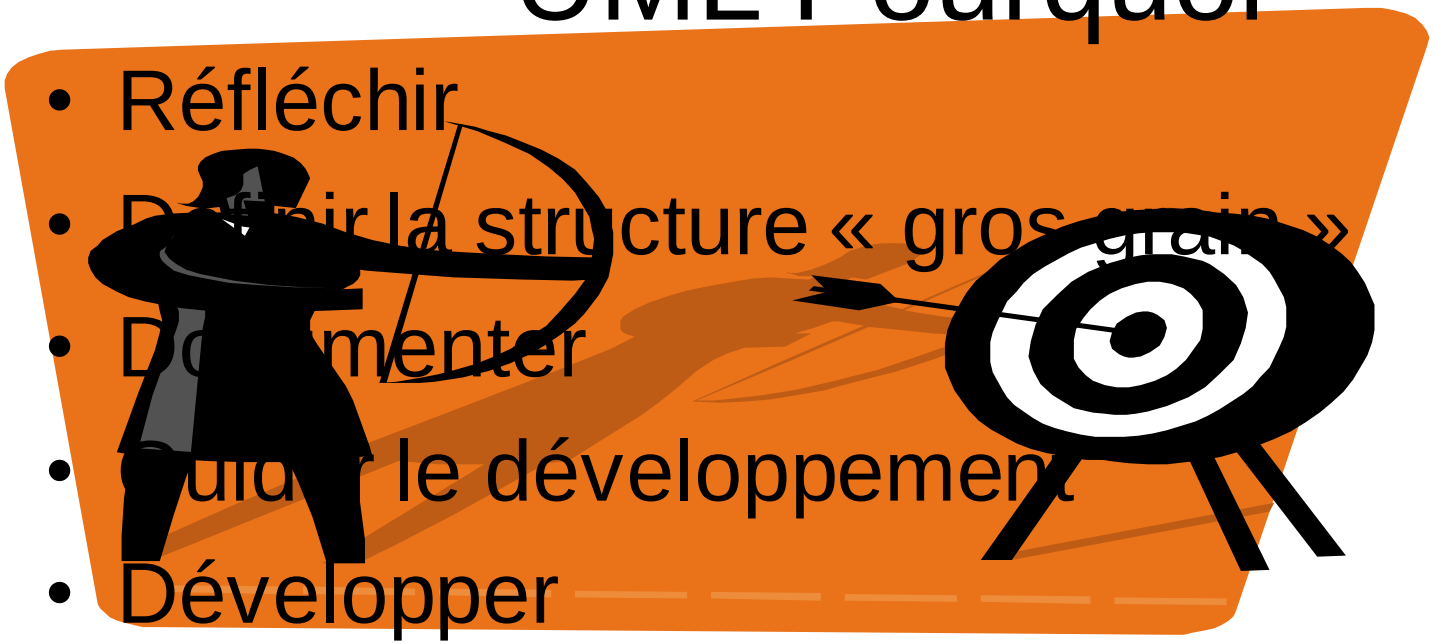
- **Un langage pour exprimer les concepts objet qu'on utilise, afin de pouvoir:**
  - Représenter des concepts abstraits (graphiquement par exemple…).
  - Limiter les ambiguïtés (parler un langage commun).
  - Faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions).

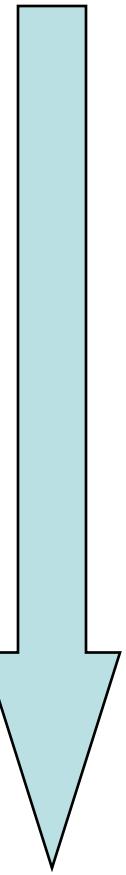
- **Une démarche d'analyse et de conception objet, pour:**
  - Ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, mais penser objet dès le départ.
  - Définir les vues qui permettent de couvrir tous les aspects d'un système, avec des concepts objets.

Unified Modeling language

UML Pourquoi

- Réfléchir
- Définir la structure « gros grain »
- Documenter
- Guider le développement
- Développer
- Tester
- Auditer





Comment modéliser avec UML

- UML langage permettant de représenter des modèles
  - Pas de définition du 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.

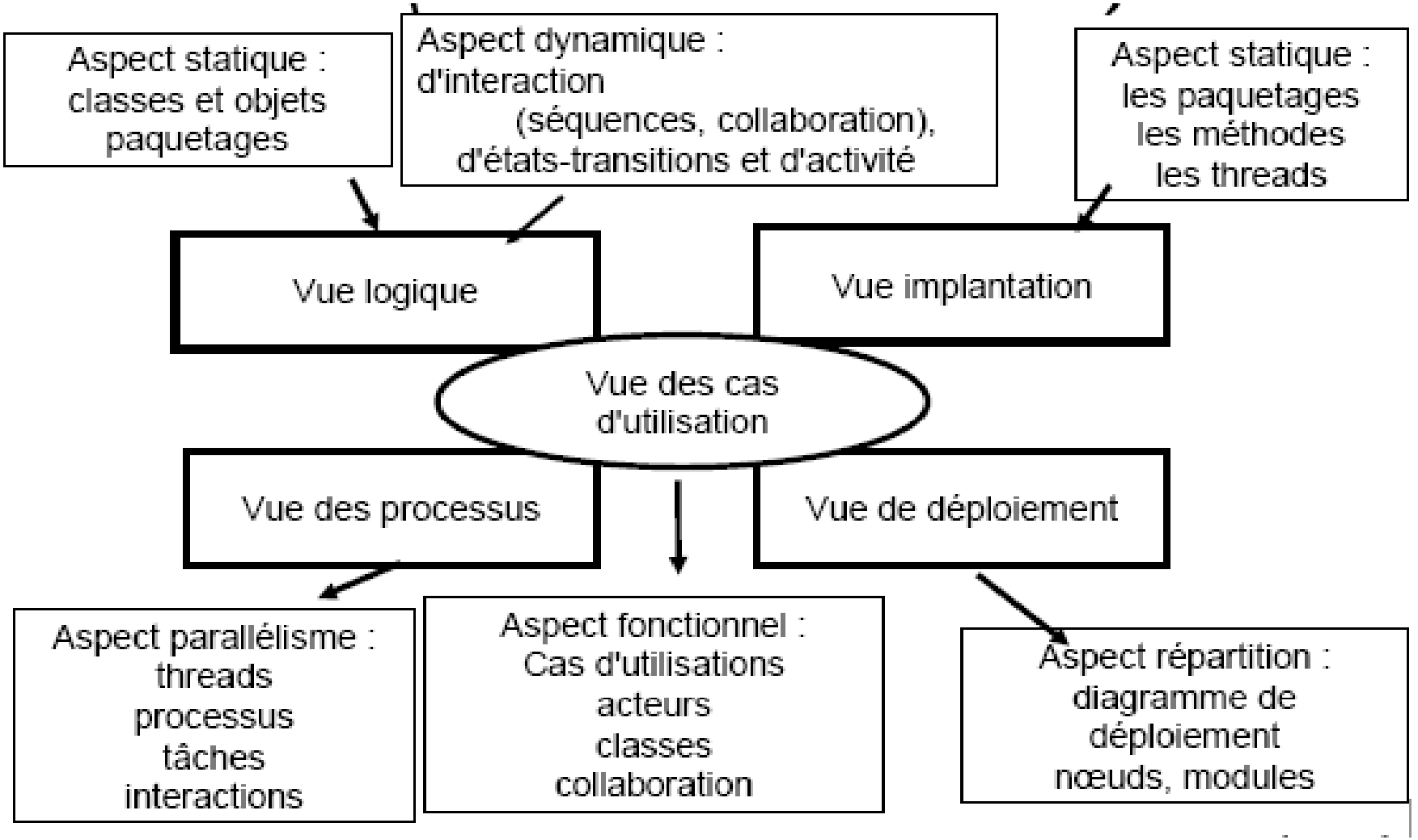
Modèles avec UML

- 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).

Vues d'UML

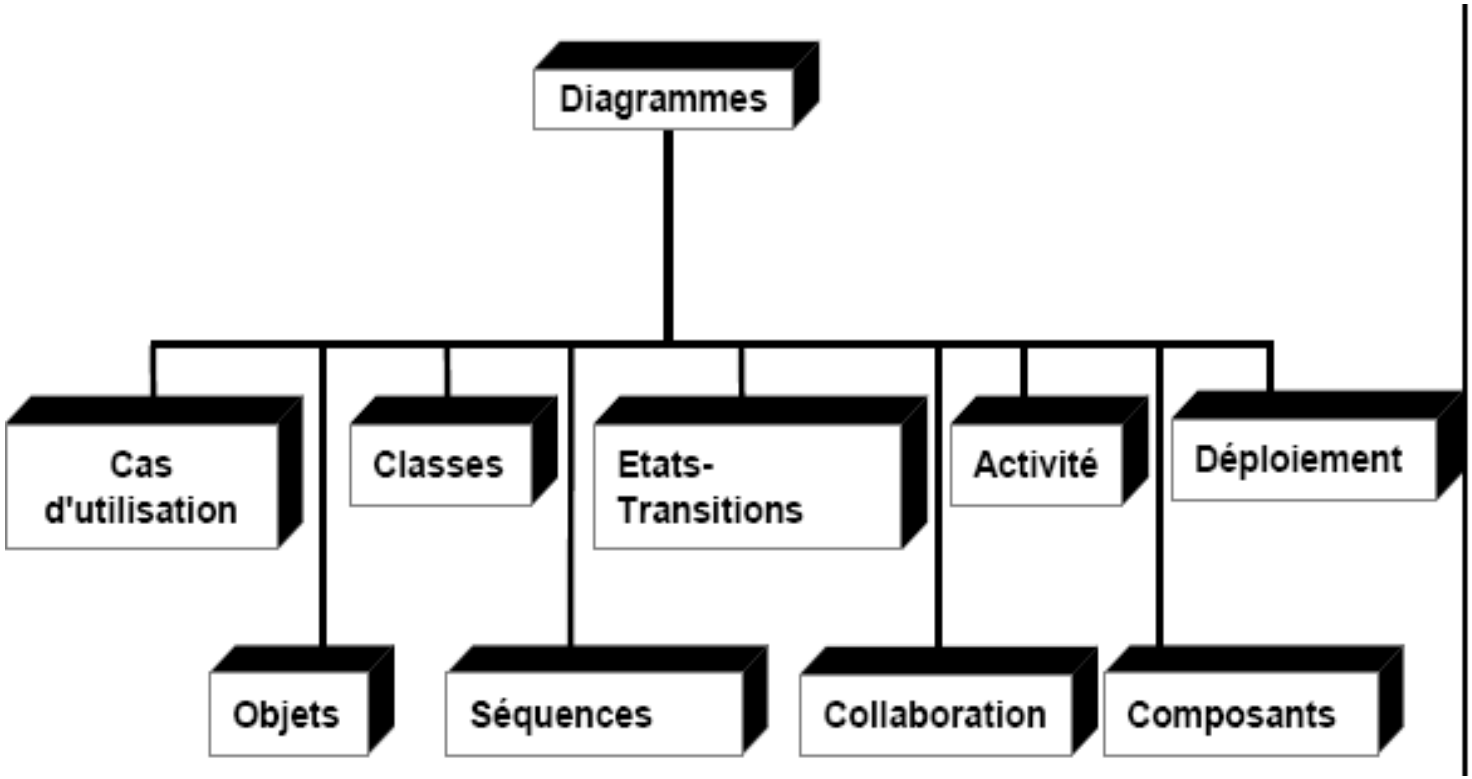
- 5 visions possibles
  - Logique
  - Cas d'utilisation
  - Processus
  - Vue implémentation
  - Vue déploiement

Vues d'UML: Contenu



UML: diagrammes

- Il existe 9 diagrammes dans UML



UML: Diagrammes

- **Diagramme de cas d'utilisation**
  - Décrit les fonctionnalités du système telles que perçues par les acteurs externes.
- **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.

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.

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