Algorithmique et UML

CM6 : UML

Mickaël Martin Nevot

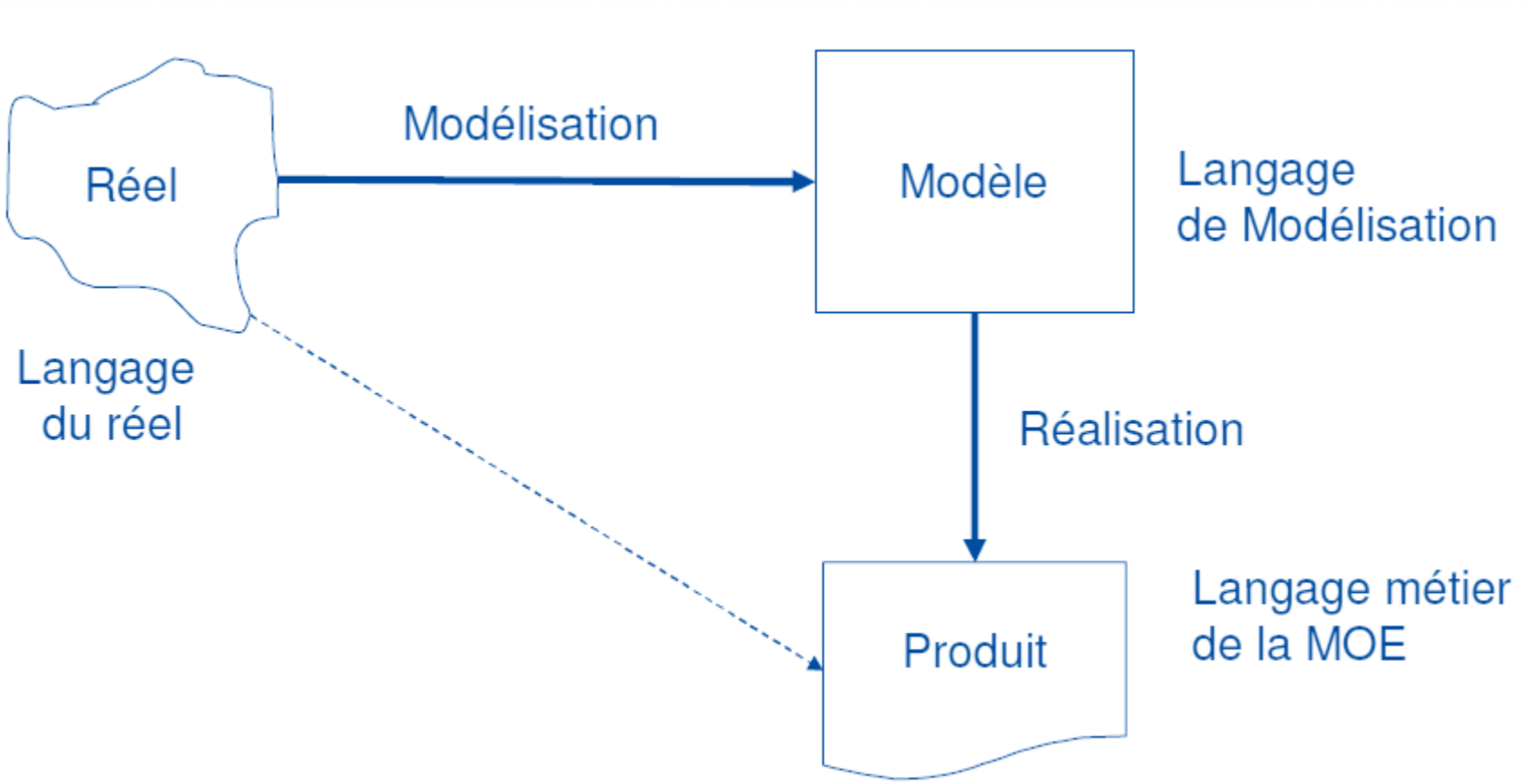
V1.6.0

Algorithmique et UML

Autres notions sur les objets

- Entité aux frontières précises :
  - **Insécable** (complet)
- **Agrégation** : construction d'objets avec d'autres objets
- **Composition** (agrégation particulière) :
  - Durée de vie : l'objet est créé et meurt avec l'agrégat
  - Exclusivité : l'objet n'est utilisé que par une seule classe
  - Fait partie (« physiquement ») de l'agrégat
- **Dépendance** :
  - Relation dans un **agrégat** logiciel

Problématique de la conception



Conception informatique

Base de données

- Approche systémique

Objet

- Approche pragmatique

UML

- **Méthodologie** du **génie logiciel**
- Langage de **modélisation** graphique

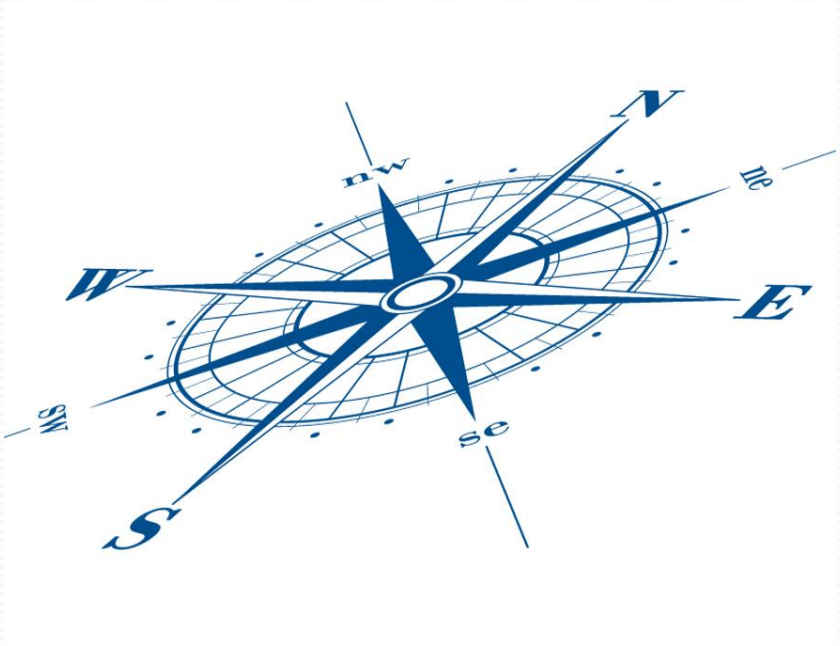
> Représentation abstraite d'un système qui facilite l'étude et la compréhension du système et permet de le simuler. Vue subjective, décomposée mais pertinente de la réalité. Représentation d'un système dans un autre monde que celui du système

- Standard industriel de la modélisation **orientée objet**
- Formel et **normalisé**
- Support de communication performant
- Utilisé en informatique mais pas limité à ce domaine
- Utilisé dans toutes les étapes d'un projet

> UML n'est pas une méthode

Vision utilisateur

- Certains utilisateurs ne connaissent pas le standard UML :
  - Vision outillée d'UML (**Vision utilisateur**) :
    - 05% : forte compréhension
    - 45% : faible compréhension
    - 50% : aucune compréhension

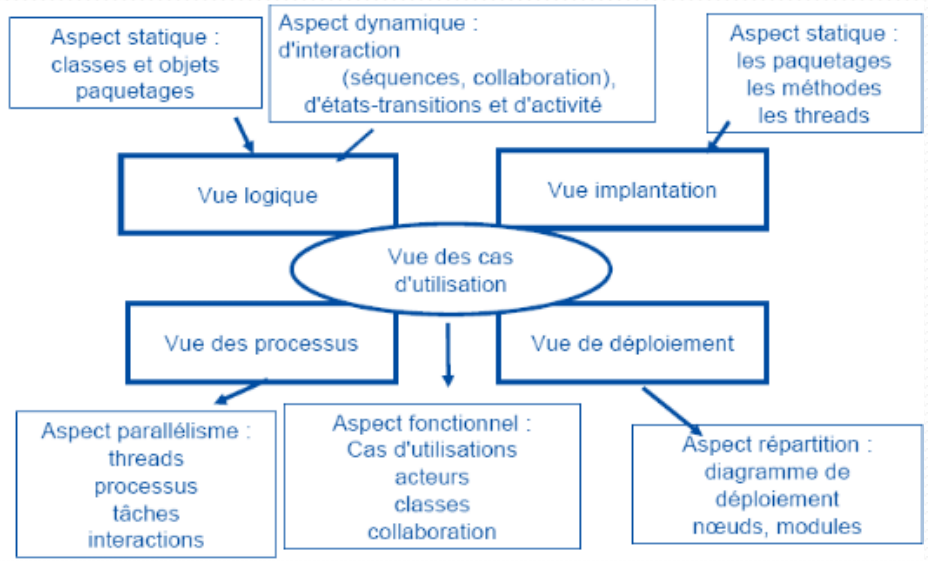


Formalisme d'UML

- **Vue** :
  - Observable du système
  - Description d'un point de vue donné
  - L'ensemble des vues définissent le système
- **Diagramme** :
  - Décrit le contenu des vues
  - Un diagramme peut faire partie de plusieurs vues
- **Modèles d'élément** :
  - Brique des diagrammes UML
  - Peuvent souvent être utilisés dans plusieurs diagrammes

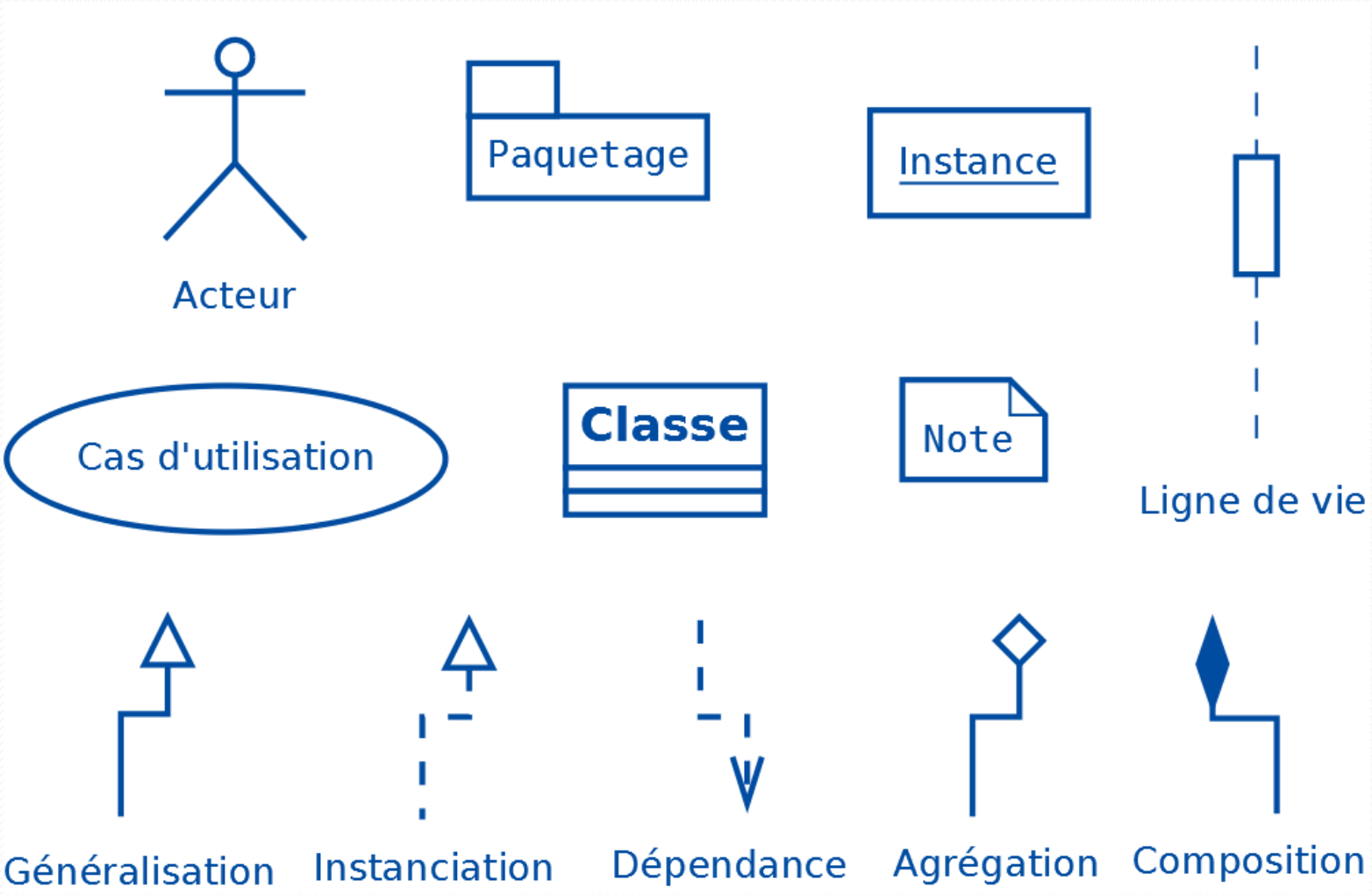
Les vues

- Vue logique : **comment ?**
- Vue d'implémentation : **dépendance** inter-modules ?
- Vue des processus : **technique ?**
- Vue des déploiements : **où** ?
- Vue des cas d'utilisation : **quoi**/**qui ?**

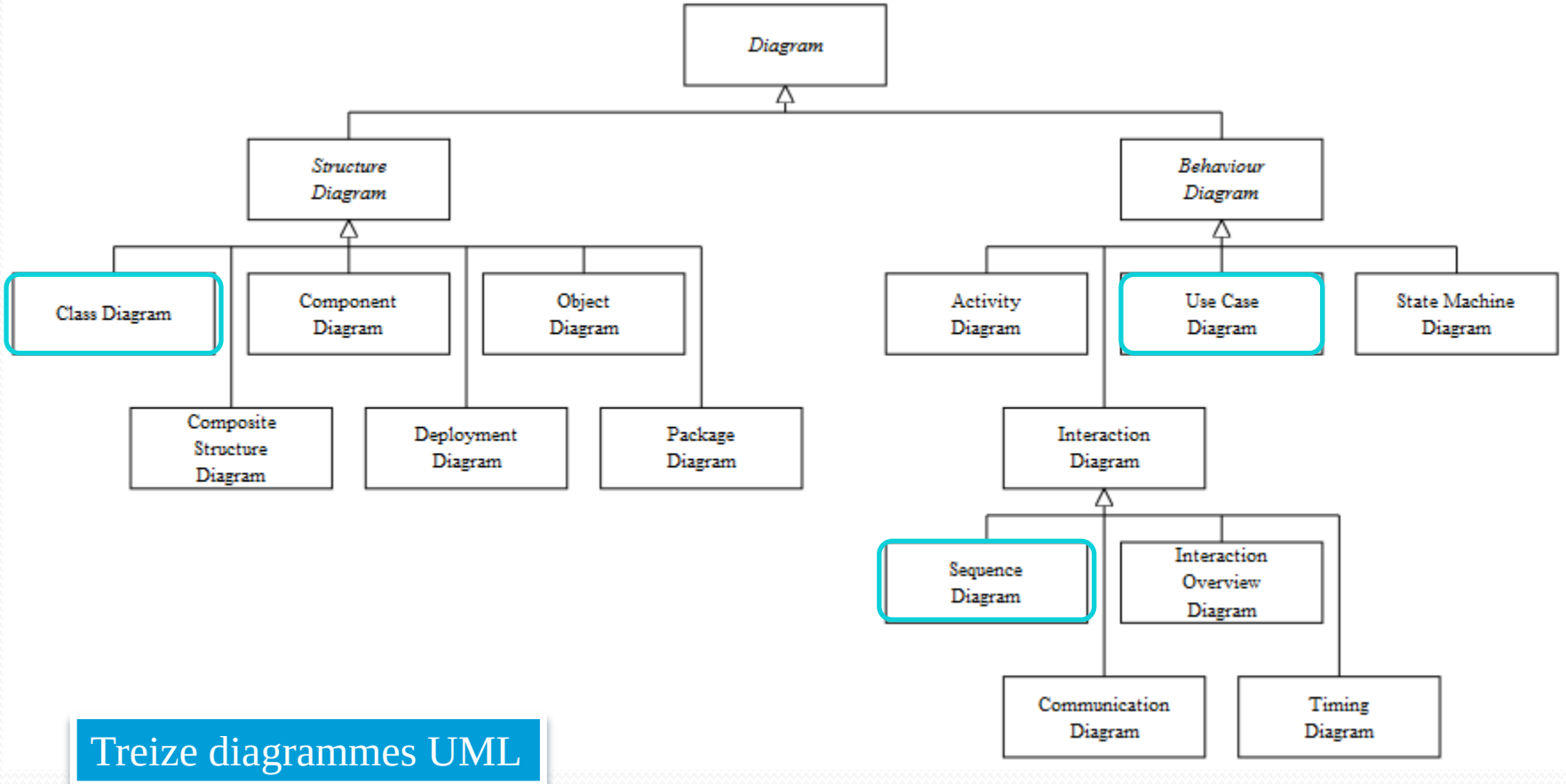


Pas de « pourquoi ?» en UML

Modèle d'élément



Hiérarchie des diagrammes



Treize diagrammes UML

Exemple d'ordre de création

Diagramme	Étape de développement
1. Diagramme de cas d'utilisation	Cahier des charges, spécification
2. Diagramme de séquence	
3. Diagramme d'activité (processus métiers)	
4. Diagramme d'activité (cinématique et/ou processus applicatifs)	
5. Diagramme de classe	Conception architecturale
6. Diagramme d'objet	
7. Diagramme de communication	
8. Diagramme de déploiement	
9. Diagramme de composant	

Diagramme de cas d'utilisation

- Expression du **comportement fonctionnel du système**
- Selon le **point de vue de l'utilisateur**
- Décrit le **système** et les **relations** système/environnement
- Intérêts :
  - Permet de délimiter les frontières du système
  - Constitue un moyen d'exprimer les besoins d'un système
  - Permet d'exprimer les attentes/besoins des utilisateurs finaux
  - Permet d'impliquer les utilisateurs finaux dès les premiers stades du développement
  - Constitue une base pour les tests fonctionnels

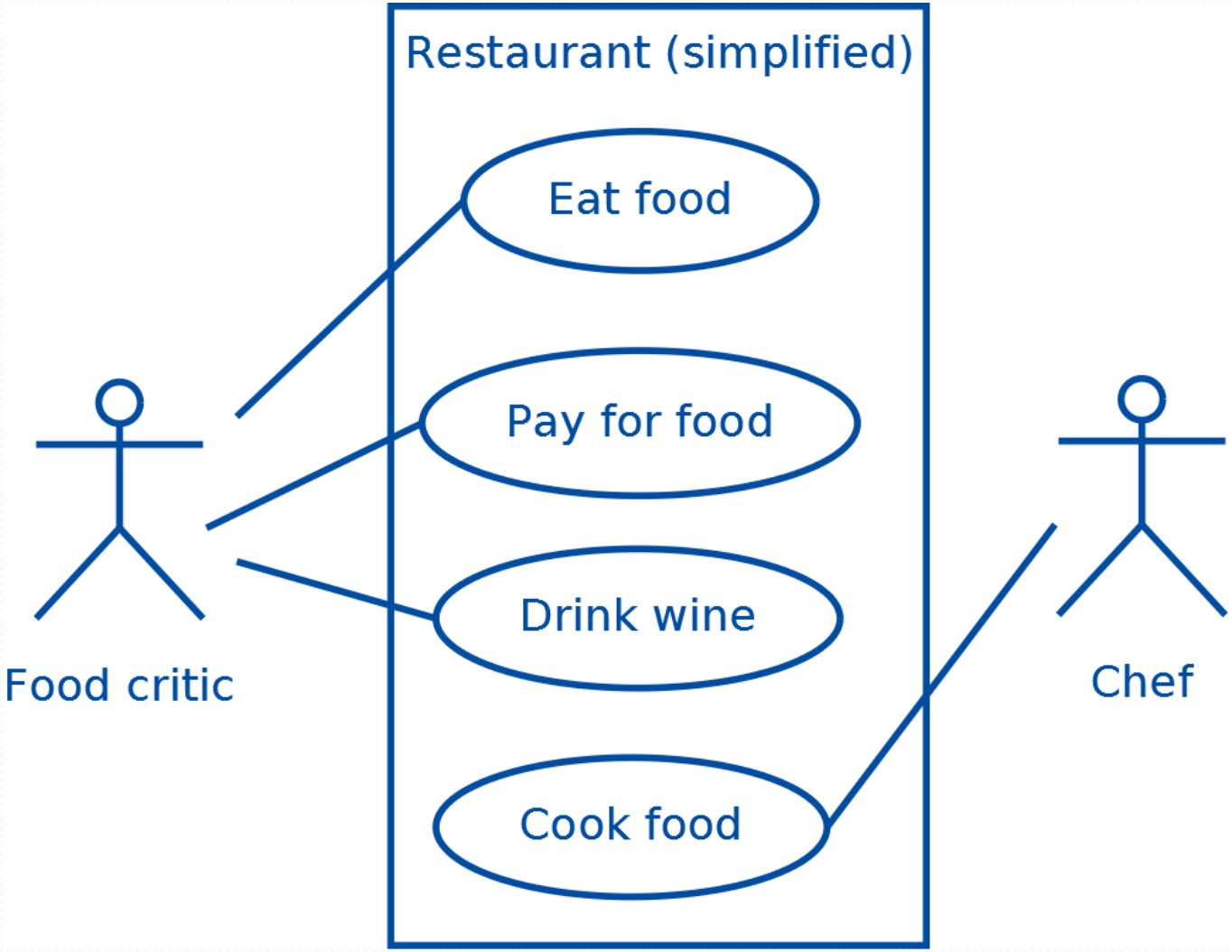
Éléments de cas d'utilisation

- **Acteur** :
  - Rôle (personne(s) ou système(s)) externe qui échange de l'information (entrée/sortie) :

    - L'acteur peut consulter ou modifier l'état du système
    - En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin
    - Les acteurs peuvent être **hiérarchisés** en utilisant l'héritage
- **Cas d'utilisation** :
  - Ensemble d'actions réalisées par le système, en réponse à une action d'un acteur :

    - Les cas d'utilisation peuvent être structurés
    - Les cas d'utilisation peuvent être organisés en paquetages
    - L'ensemble des cas d'utilisation décrit les objectifs du système

Cas d'utilisation (exemple 1)



Relations entre cas d'utilisation

- Relation d'utilisation : <<include>>
  - Le cas d'utilisation **contient** un autre cas d'utilisation
- Relation d'extension : <<extend>>
  - Le cas d'utilisation **étend** (précise) les objectifs (le comportement) d'un autre cas d'utilisation

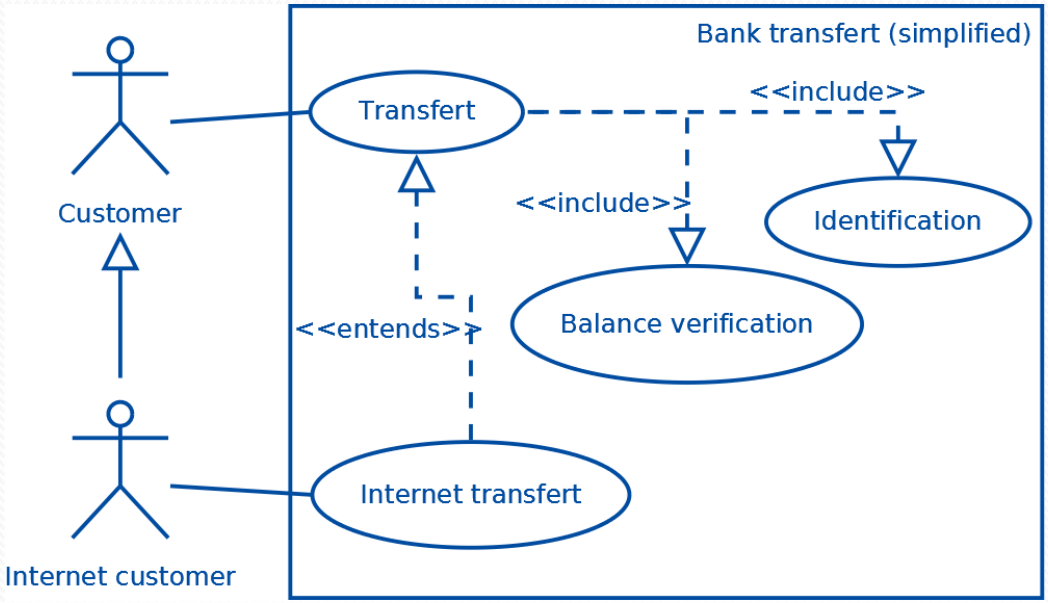


Diagramme de séquence

- Représentation graphique des interactions entre les acteurs et le système selon un **ordre chronologique**
- Dans le cadre d'un scénario d'un diagramme des cas d'utilisation
- Types de messages :
  - Simple :
  - Avec durée de vie (attend une réponse) :
  - Synchrone (bloquant) :
  - Asynchrone (non bloquant) :
  - Dérobant (en attente) :

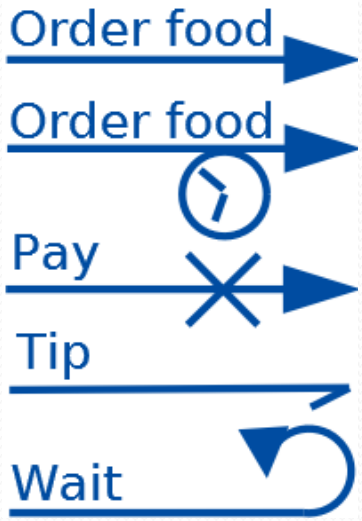


Diagramme de séquence

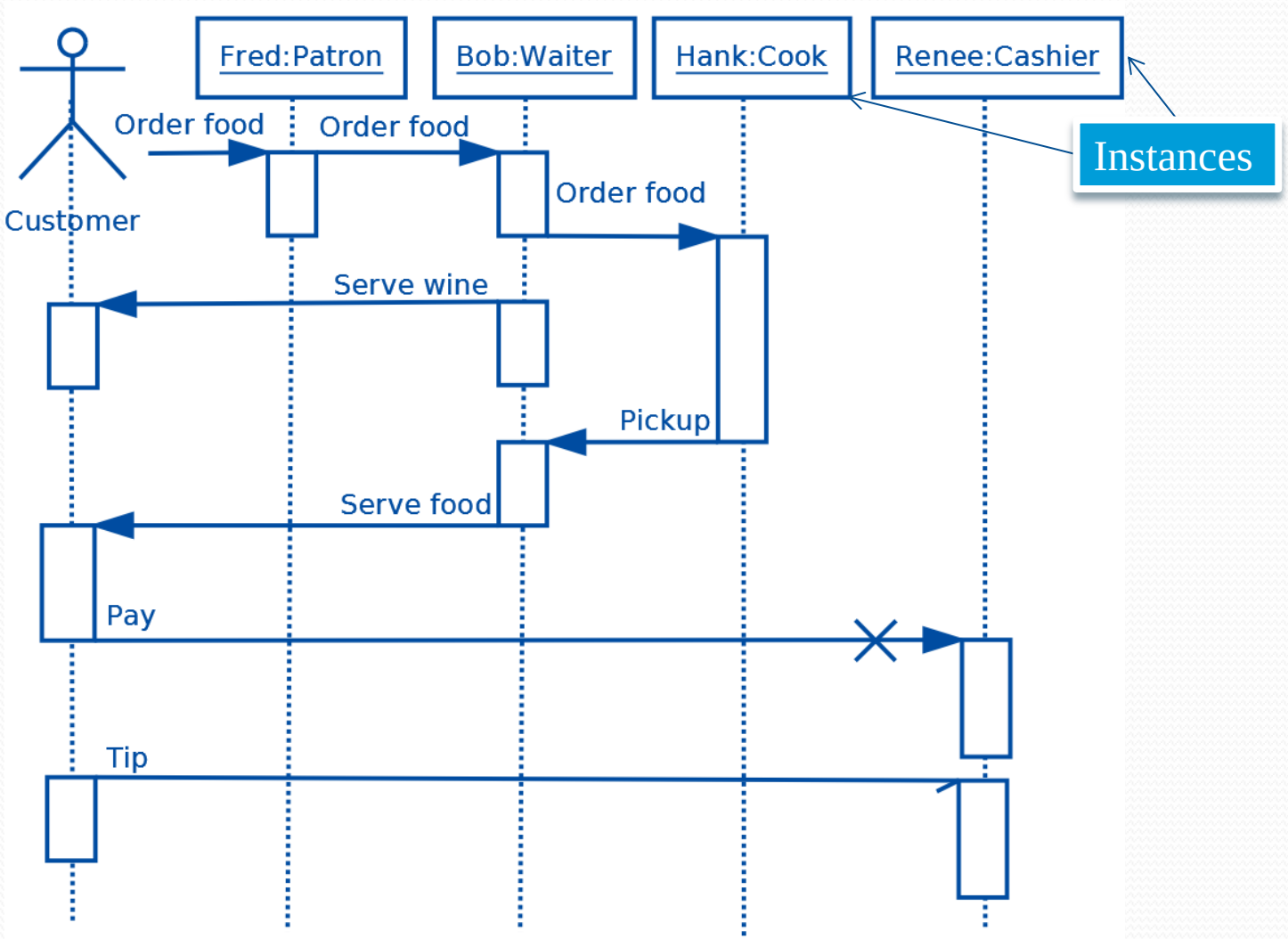


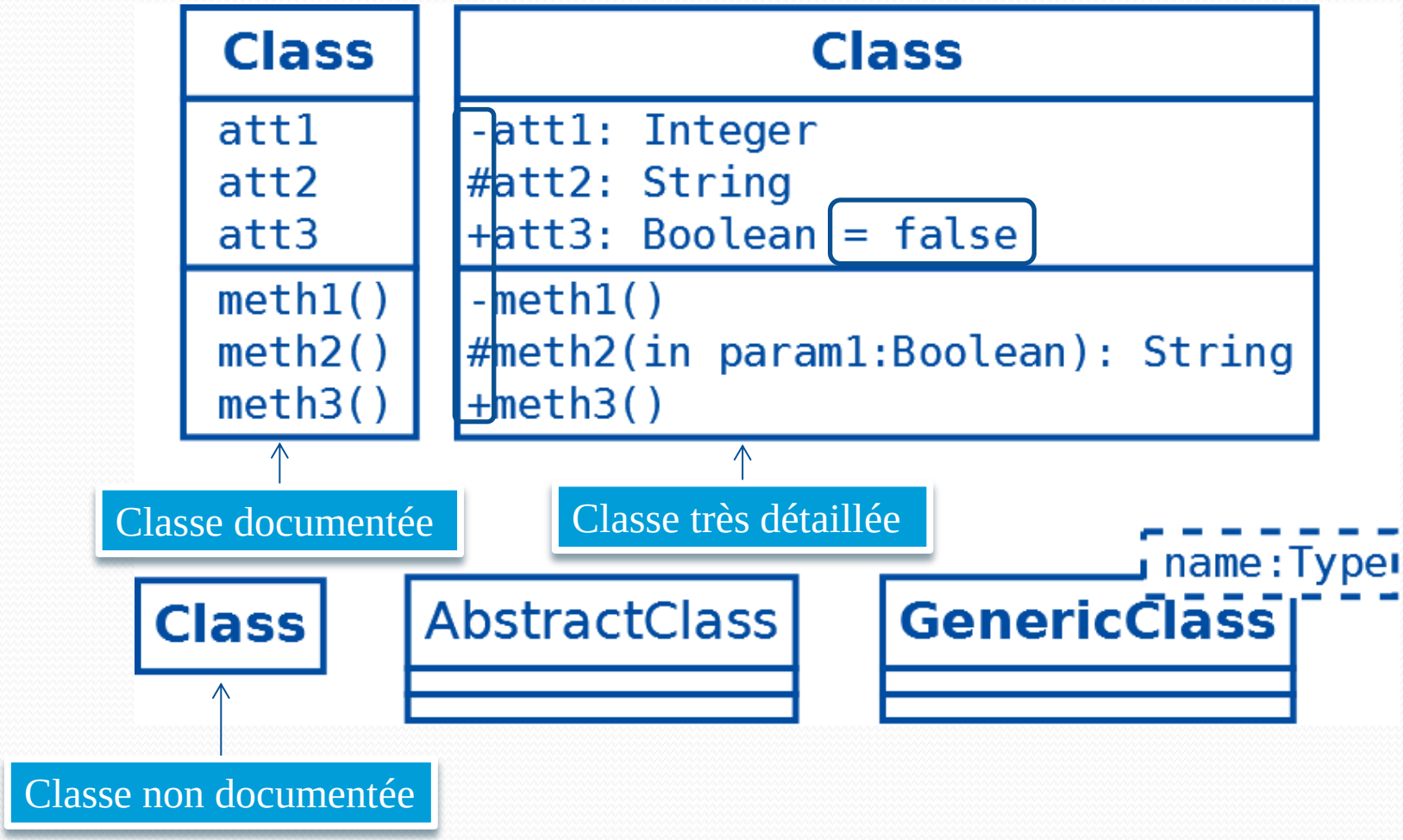
Diagramme de classes

- Présente les **classes**, les **interfaces** et les relations entre eux :
  - Le **nom** :
    - En normal pour les classes classiques
    - En *italique* (souvent) pour les classes abstraites
  - Les **attributs** :
    - Multiplicité : nombre de fois où l'attribut peut exister (entre `[]`)
  - Les **méthodes** :
    - Direction du paramètre : `in` (rentrant) / `inout` (rentrant/sortant)

Visibilité :
+ : public
# : protégé
– : privé

- Diagrammes complémentaires si modèle complexe :
  - Les classes qui participent à un cas d'utilisation précis
  - Les classes associées dans la réalisation d'un scénario précis
  - Les classes qui composent un paquetage
  - La structure hiérarchique d'un ensemble de classes

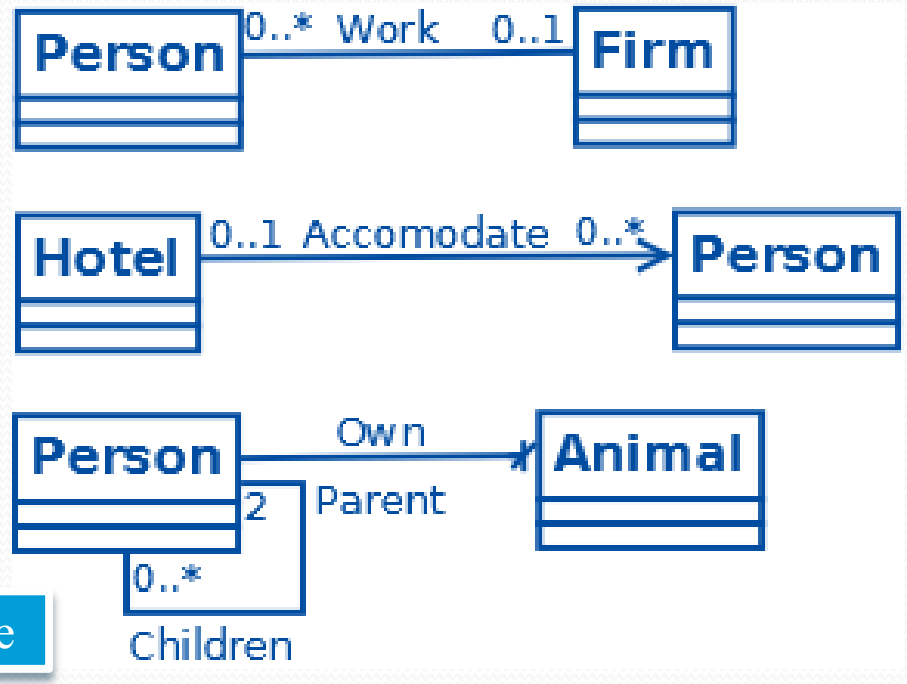
Diagramme de classes 1



Association

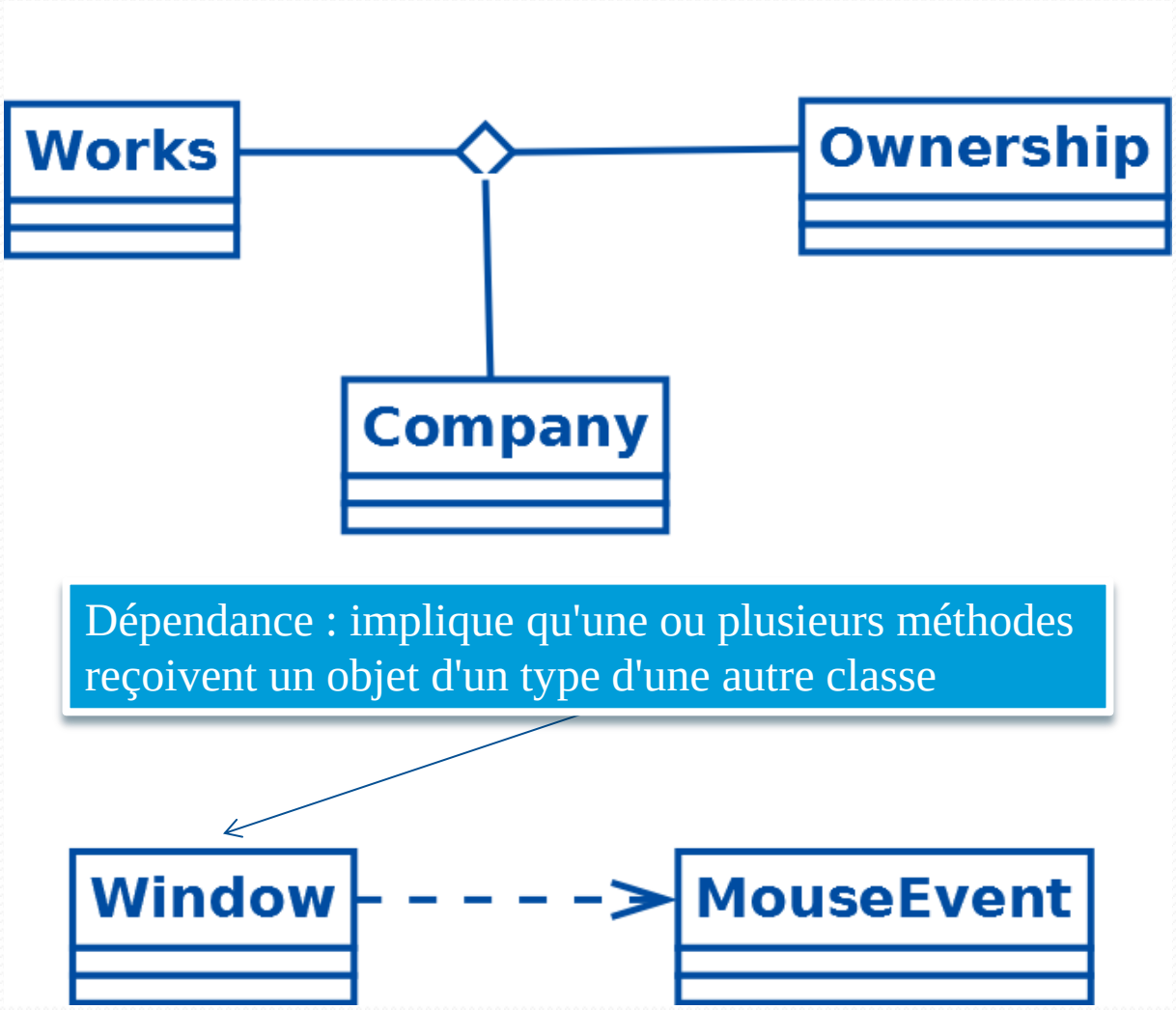
- Connexion sémantique entre deux classes
- **Multiplicité** : nombre minimum et maximum d'instances de chaque classe dans la relation entre classes
- **Navigabilité** :
  - Bidirectionnelle
  - Mono-directionnelle

    (invocation de méthode)
  - Interdit une association



Attention : navigabilité UML ≠ cardinalité Merise

Association n-aires et dépendance



Classe d'association et contrainte

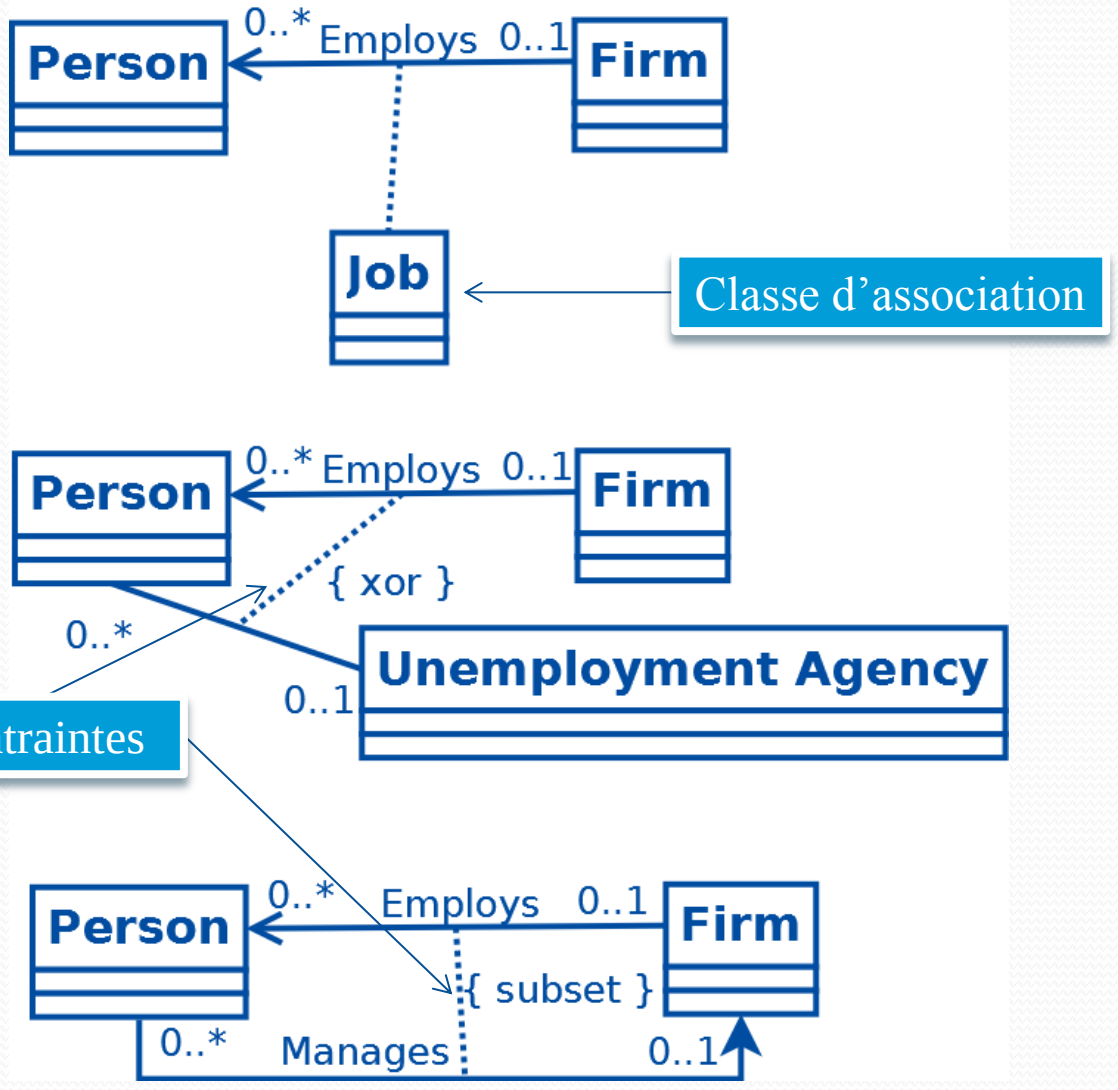
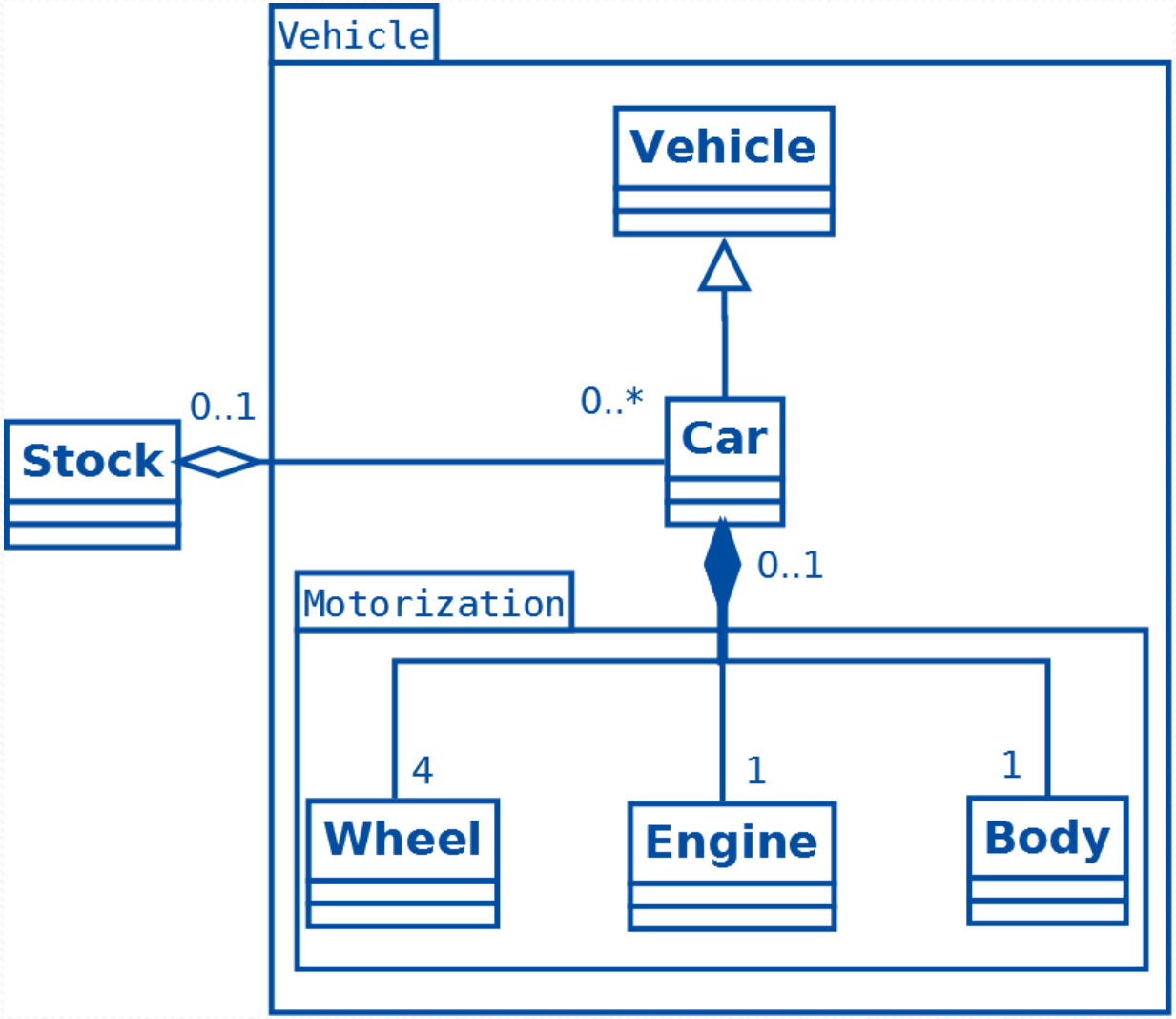


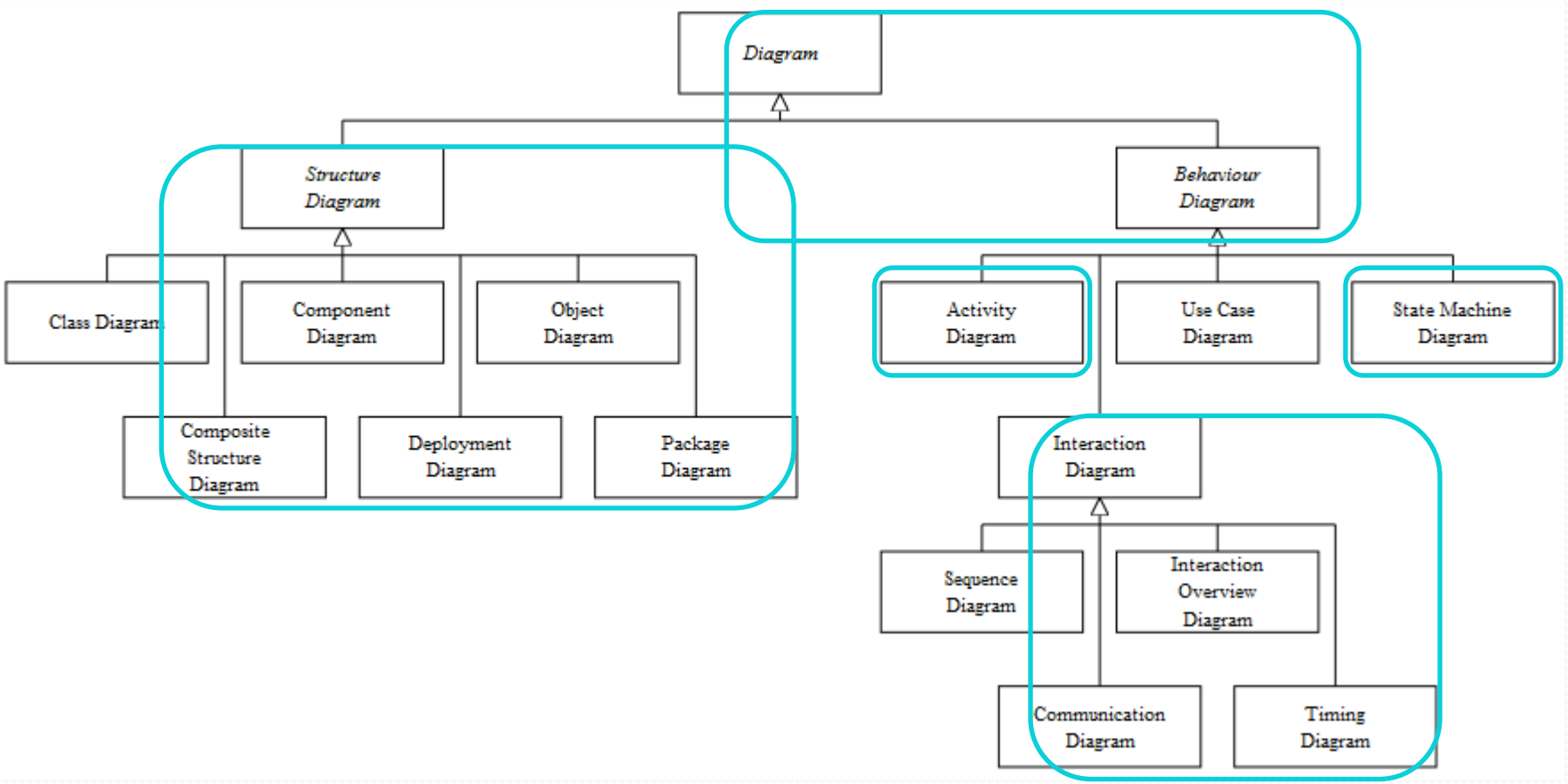
Diagramme de classes 2



Logiciels de modélisation UML

- Payants :
  - Together
  - **Rationnal Rose**
  - Acceleo
- Gratuits :
  - **Dia** ← Minimaliste (utiliser dans Wikipedia et dans le cadre de ce cours)
  - Eclipse
- Java :
  - Poseidon
  - ArgoUML

Aller plus loin



Liens

- Document classique :
  - Livre :
    - Craig Larman. *UML2 et les design patterns.*

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com