Génie Logiciel

Chapitre 2:

Modélisation avec UML

Niveau: 3ième année License informatique

Année: 2023/2024

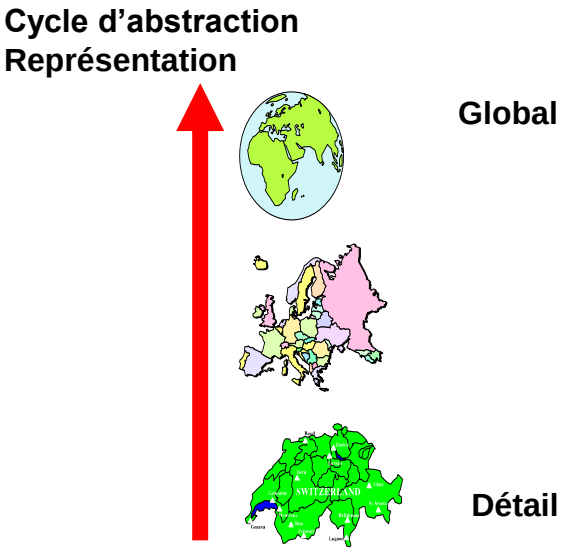
Introduction générale

Introduction générale

- **Qu'est qu'un modèle?**
  - **Un modèle est une simplification de la réalité**
- **Pourquoi modéliser?**
  - **Les modèles permettent de mieux comprendre le système que l'on développe**
- **Nous construisons des modèles pour les systèmes complexes parce que nous ne sommes pas en mesure d'appréhender de tels systèmes dans leur intégralité.**

Introduction générale

- **Le choix des modèles à créer a une forte influence sur la manière d'aborder un problème et sur la nature de sa solution.**
- **Tous les modèles peuvent avoir différents niveaux de précision.**
- **Les meilleurs modèles ne perdent pas le sens de la réalité.**
- **Parce qu'aucun modèle n'est suffisant à lui seul, il est préférable de décomposer un système important en un ensemble de petits modèles presque indépendants.**



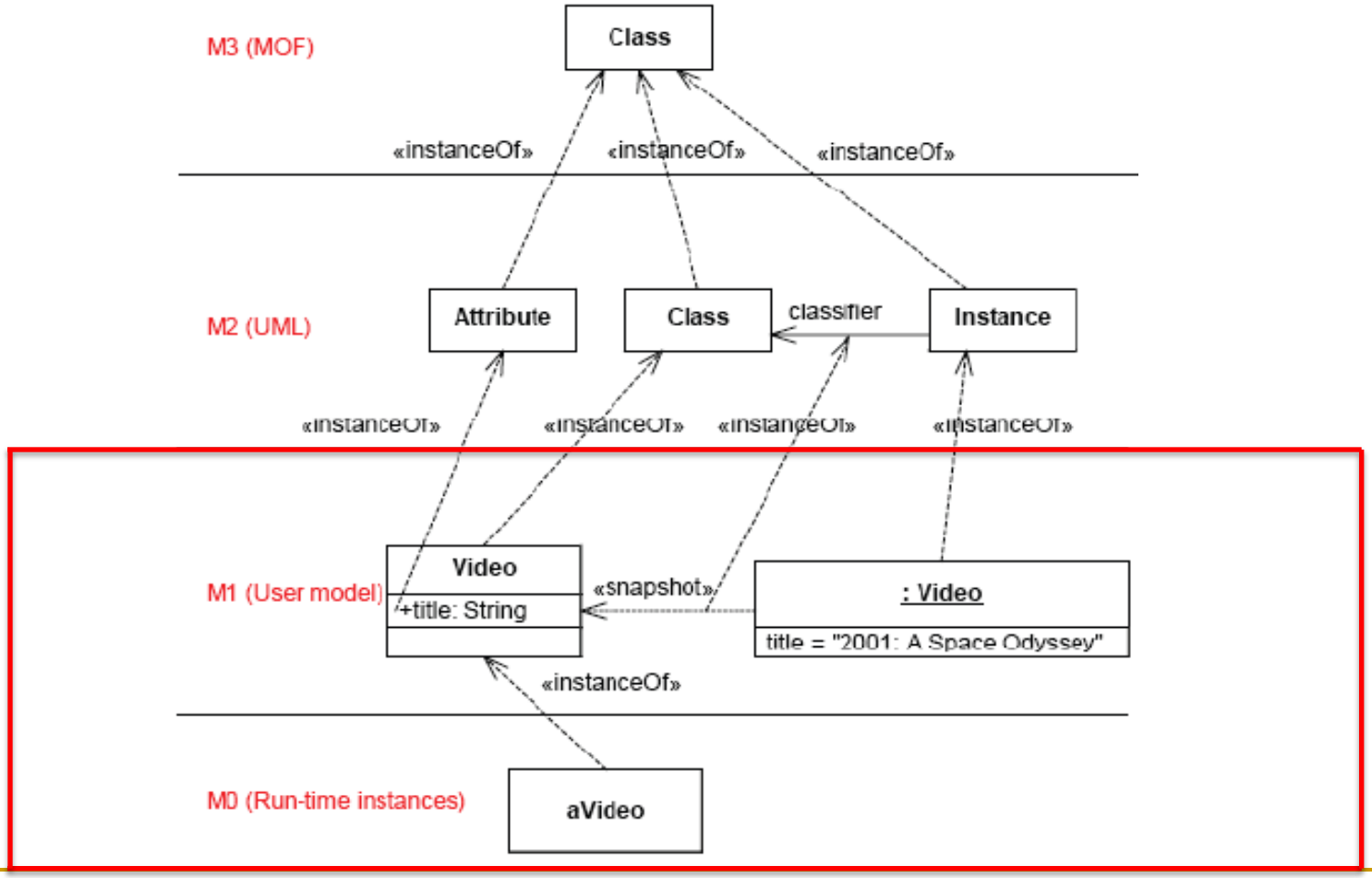
Introduction générale

- **L'OMG (Object Management Group) définit 4 niveaux de modélisation**
  - **M0 : système réel, système modélisé**
  - **M1 : modèle du système réel défini dans un certain langage**
  - **M2 : méta-modèle définissant ce langage**
  - **M3 : méta-méta-modèle définissant le méta-modèle**

- **Le niveau M3 est le MOF (Meta-Object Facility)**
  - **Dernier niveau, il est méta-circulaire : il peut se définir lui-même**

- **Le MOF est – pour l'OMG – le méta-méta-modèle unique servant de base à la définition de tous les métamodèles**
- **Seules les deux couches *user model* et *run time instance* sont destinées à l'utilisateur.**

Introduction générale



Introduction a l'UML

Introduction a l'UML

- **UML (Unified Modeling Language, "*langage de modélisation objet unifié*") est un langage pour *visualiser*, *spécifier*, *construire* et *documenter* tous les aspects et artefacts d'un système logiciel.**

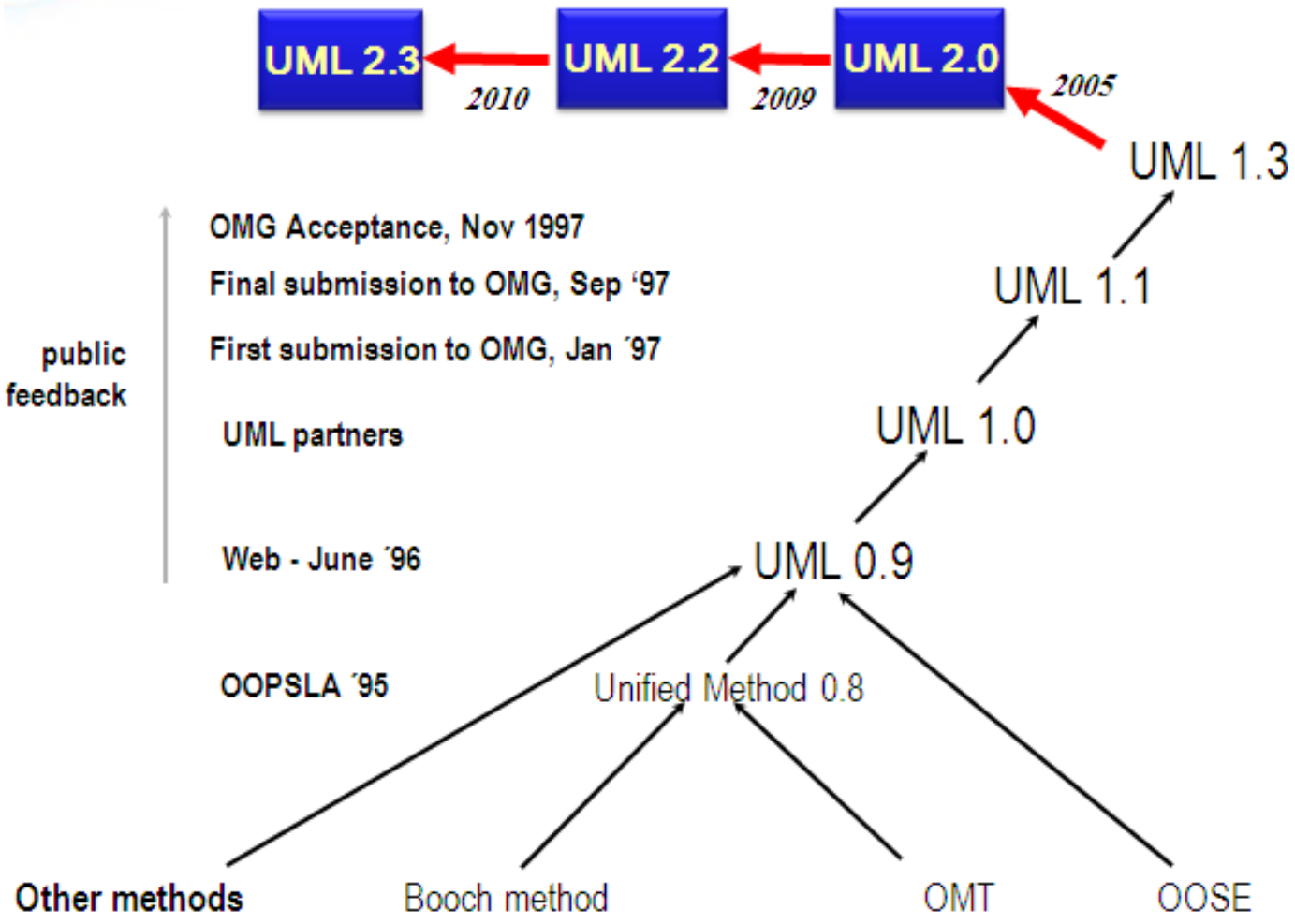
- **UML est le résultat de la fusion d'un ensemble d'autres langages de modélisation graphique orientés objets (OMT, Booch et OOSE)**

- **UML est rapidement devenu le standard incontournable de la modélisation orienté objet.**

Introduction a l'UML

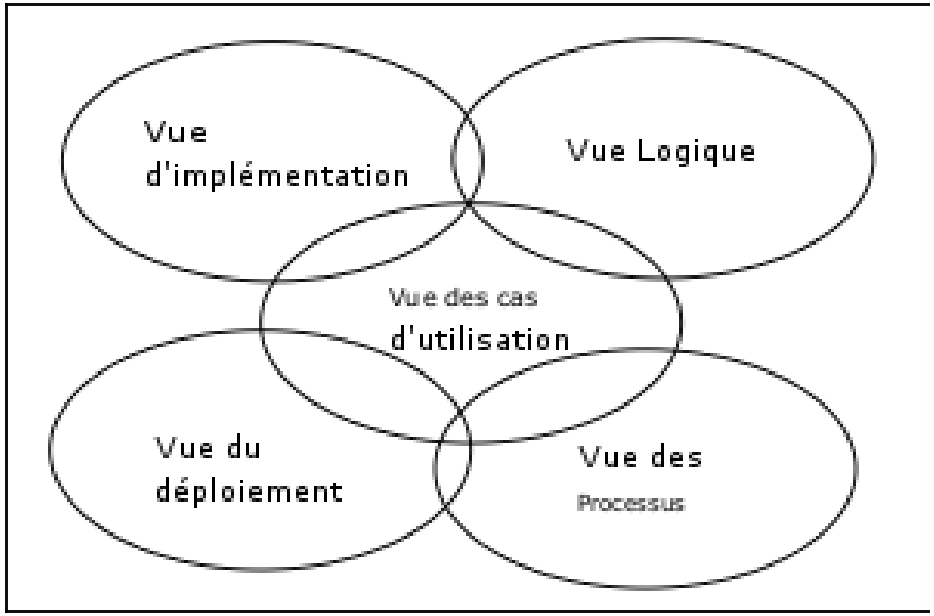
- **L'évolution du langage UML est contrôlée par l'OMG, un organisme international à but non lucratif, créé en 1989 à l'initiative de grandes entreprises (HP, Sun, Unisys, American Airlines, et Philips).**

Introduction a l'UML



Notion de vue

- **UML est fondé sur la notion de « vue », qui traduit le fait qu'il y a plusieurs manières de regarder un système. UML suit le modèle « 4+1 vues »**
- **Ces vues se concrétisent en 14 types de diagrammes.**



Notion de vue

- **La vue logique** s'attache aux aspects statiques du système en termes d'objets et de classes, enrichis de descriptions dynamiques en termes d'activités, de séquences et de communications.
- **La vue des processus** s'intéresse aux flots d'exécution et à leur synchronisation, en termes de taches, threads et processus, avec leurs états et leurs interactions.
- **La vue de la réalisation** décrit l'organisation des composants logiciels.
- **La vue du déploiement** spécifie les ressources matérielles et l'implantation des composants logiciels sur ces ressources.
- **La vue des cas d'utilisation** donne une vision d'ensemble des finalités du système en termes d'acteurs, de fonctionnalités et de scénarios d'utilisation.

Les points forts d'UML

- UML est un langage semi-formel et normalisé
  - gain de précision
  - gage (garantit) de stabilité
  - encourage l'utilisation d'outils
- UML est un support de communication performant
  - Il cadre l'analyse.
  - Il facilite la compréhension de représentations abstraites complexes.
  - Son caractère polyvalent et sa souplesse en font un langage universel.

Les points faibles d'UML

- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.
  - Même si l'espéranto est une utopie, la nécessité de s'accorder sur des modes d'expression communs est vitale en informatique.
- Le **processus** (non couvert par UML) est une autre clé de la réussite d'un projet.
  - Or, **l'intégration d'UML dans un processus** n'est pas triviale et améliorer un processus est une tâche complexe et longue.
  - Les auteurs d'UML sont tout à fait conscients de l'importance du processus, mais l'acceptabilité industrielle de la modélisation objet passe d'abord par la disponibilité d'un langage d'analyse objet performant et standard.
- Manque de sémantique précise.

Les diagrammes UML

Les diagrammes UML

- **Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis de modèle ; c'est une perspective du modèle.**

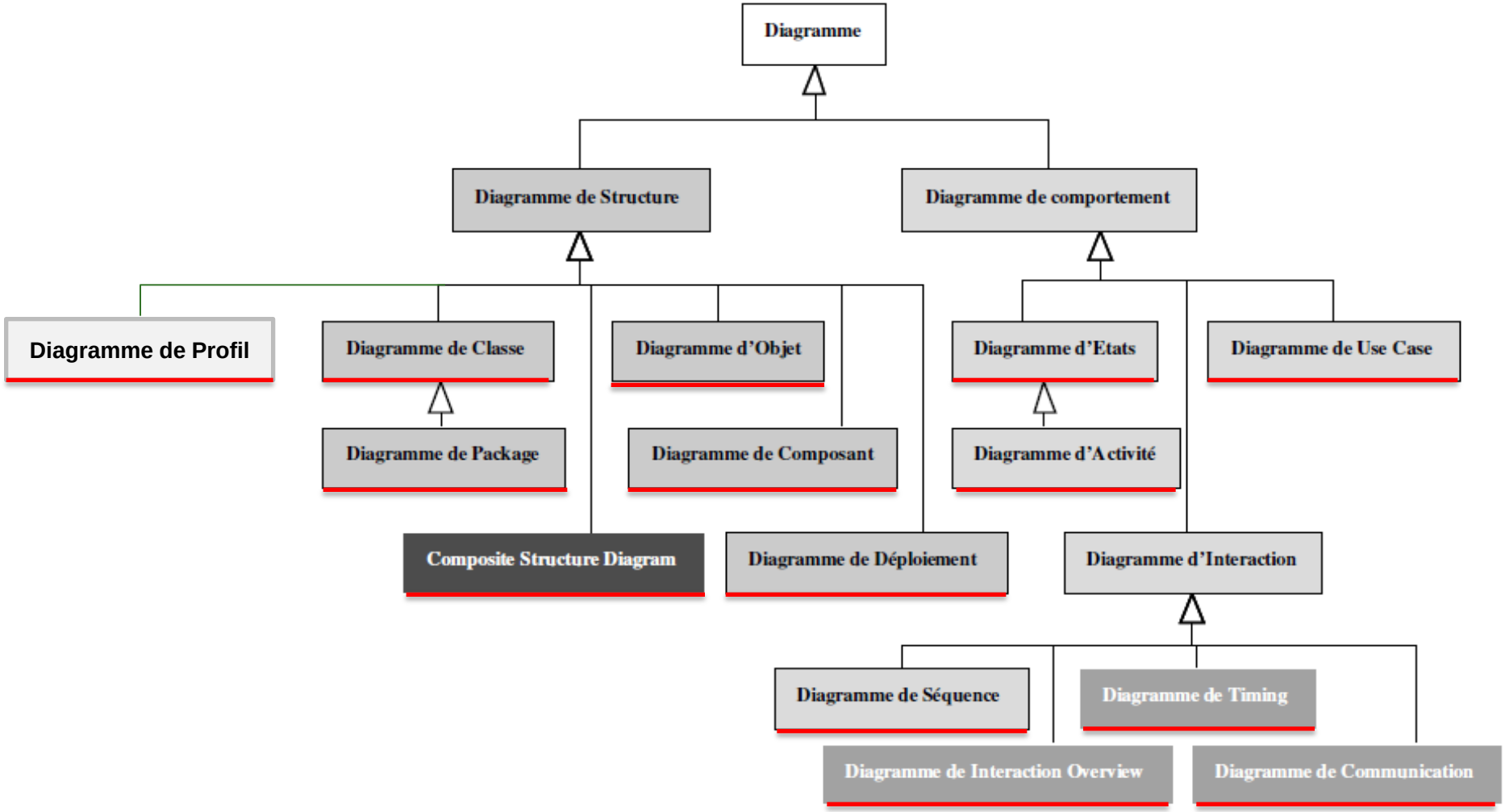
- **Chaque type de diagramme d'UML possède une structure et véhicule une sémantique précise.**

- **Combinés, les différents types de diagramme d'UML offrent une vue complète des aspects statiques et dynamiques d'un système.**

Les diagrammes UML

- **UML 2 comporte ainsi treize (14) types de diagrammes représentant autant de vues distinctes pour représenter des concepts particuliers du système d'information. Ils se répartissent en deux grands groupes comme illustré dans la figure ci-dessous:**

Les diagrammes UML



Diagrammes structurels (statiques)

Diagramme de classes

- **Il représente la description statique de système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système.**

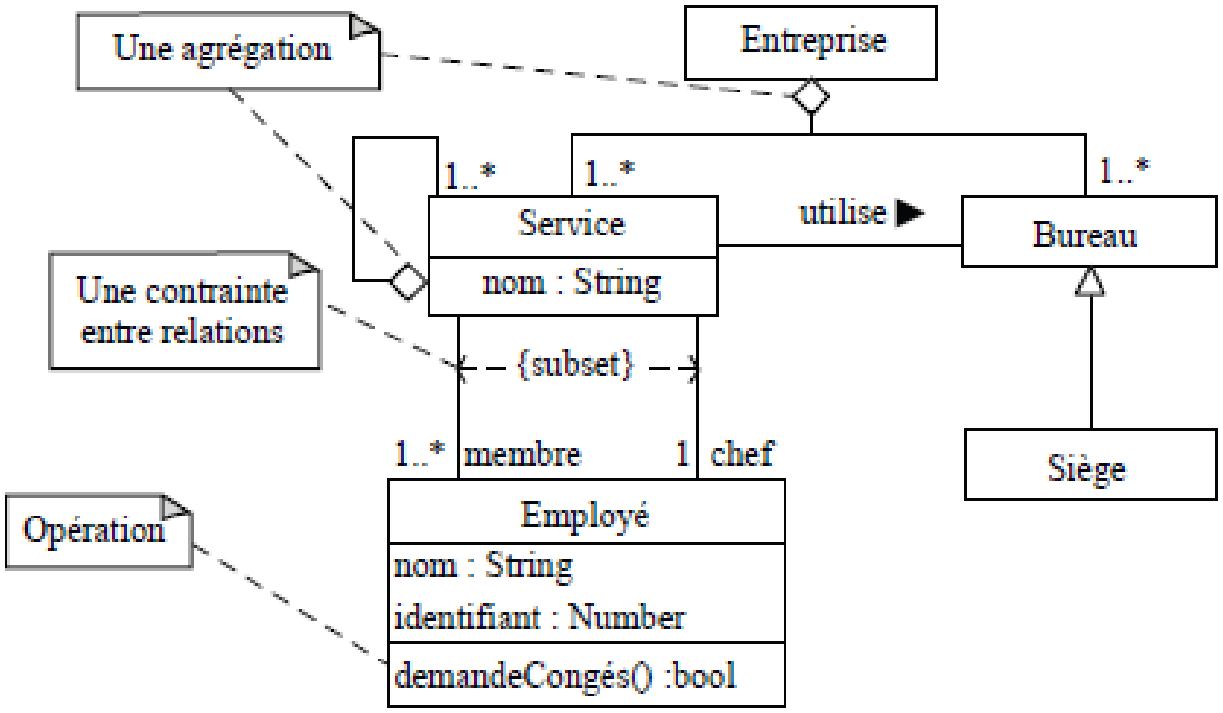


Diagramme d'objets

- **Le diagramme d'objets sert à illustrer des structures de classes compliquées en montrant des exemples d'instances. Ce diagramme est utilisé en analyse pour vérifier l'adéquation d'un diagramme de classes à différents cas possibles.**

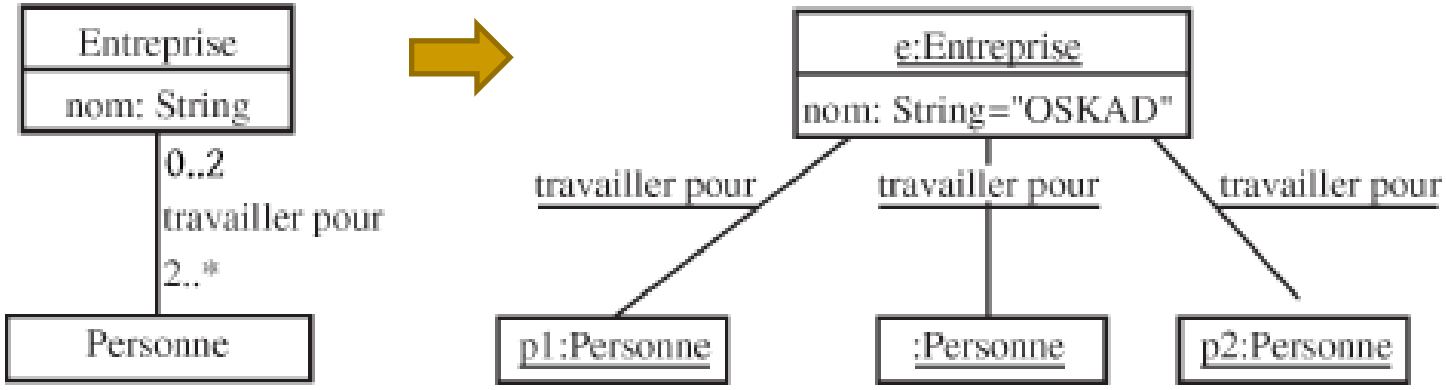


Diagramme de paquetages

- **montre les paquetages et, éventuellement, les relations entre eux.**

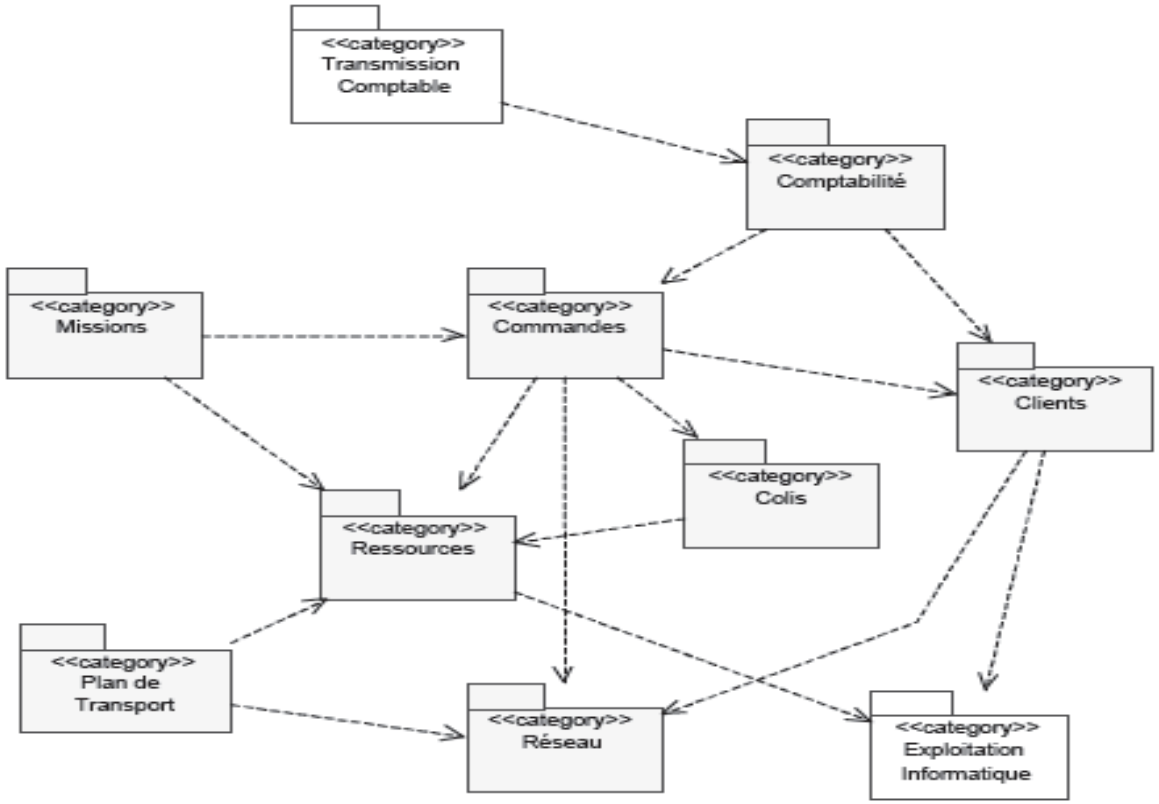


Diagramme de composants

- **Le diagramme de composants représente les concepts connus de l'exploitant pour installer et dépanner le système. Il s'agit dans ce cas de déterminer la structure des composants d'exploitation que sont les librairies dynamiques, les instances de bases de données, les applications, les progiciels, les objets distribués, les exécutables, etc.**

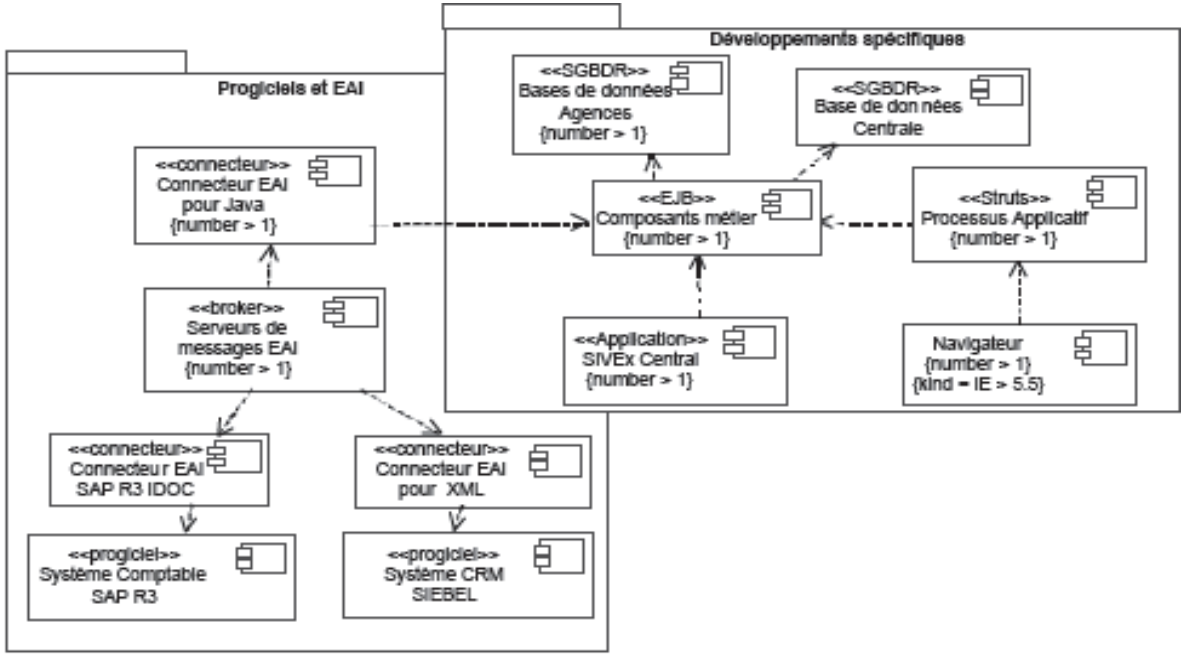


Diagramme de déploiements

- **Le diagramme de déploiement correspond à la fois à la structure du réseau informatique qui prend en charge le système logiciel, et la façon dont les composants d'exploitation y sont installés.**

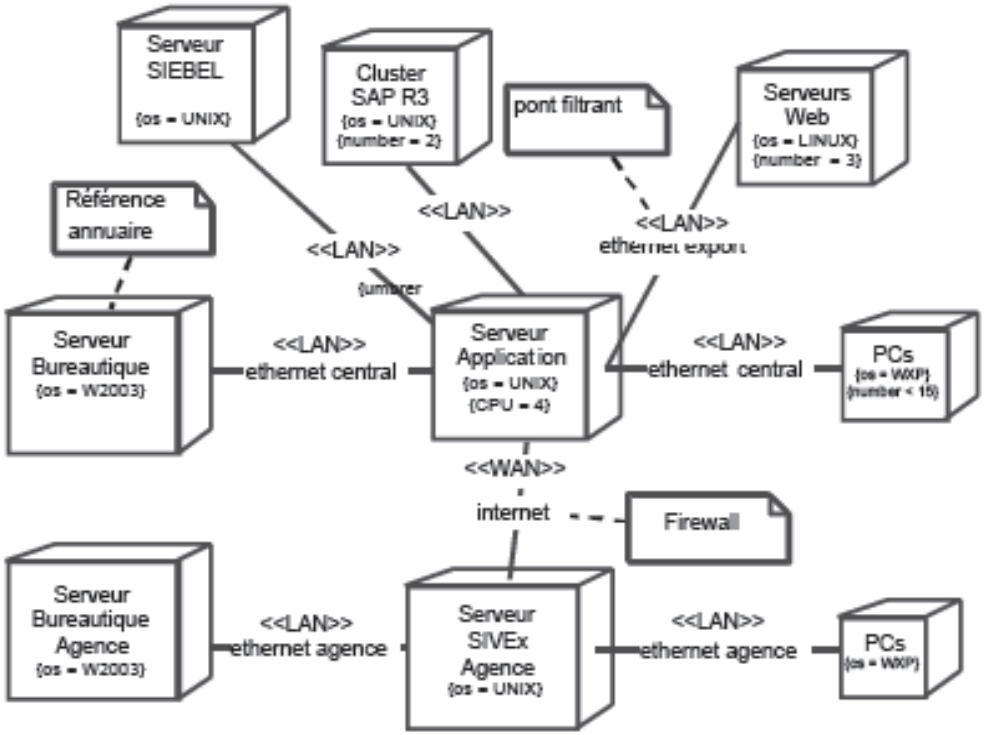


Diagramme de structures composites

- **Le diagramme de structure composite décrit la composition d'un objet complexe lors de son exécution. Ce diagramme est propre à UML 2 ; il introduit la notion de structure d'un objet complexe, tel qu'il se présente en phase de run-time.**

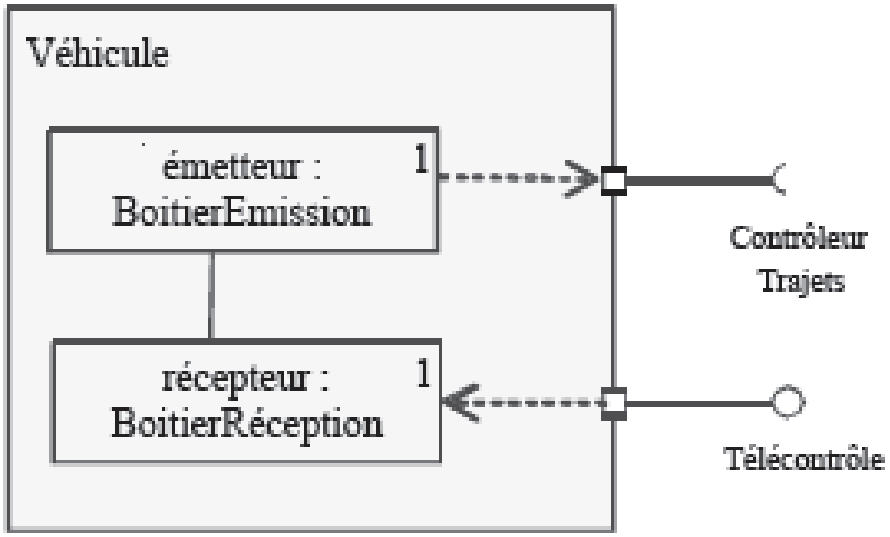
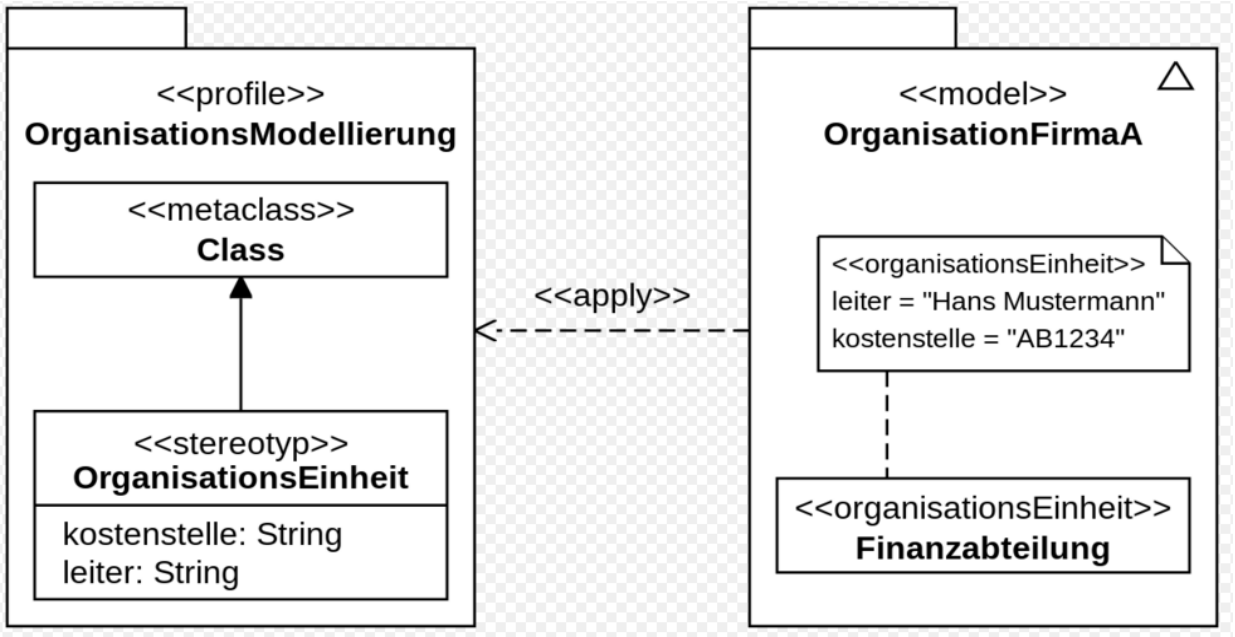


Diagramme de profils

- un **diagramme de profils** permet l'utilisation de profils pour un métamodèle donné. Apparu avec UML 2.2, ce diagramme fournit une représentation des concepts utilisés dans la définition des profils (packages, stéréotypes, application de profils, etc.)



Diagrammes comportementaux (dynamiques)

Diagramme de cas d'utilisation (use case)

- **il est destiné à représenter les besoins des utilisateurs par rapport au système.**

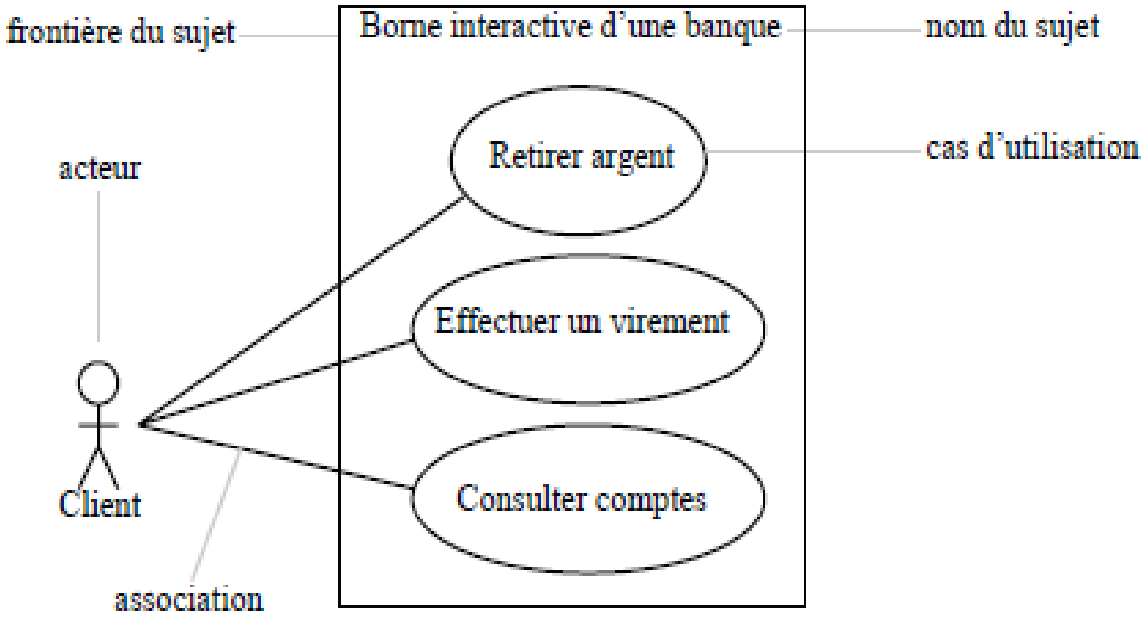


Diagramme d'activités

- **Le diagramme d'activité donne une version des enchaînements des activités propre à une méthode ou à un cas d'utilisation.**

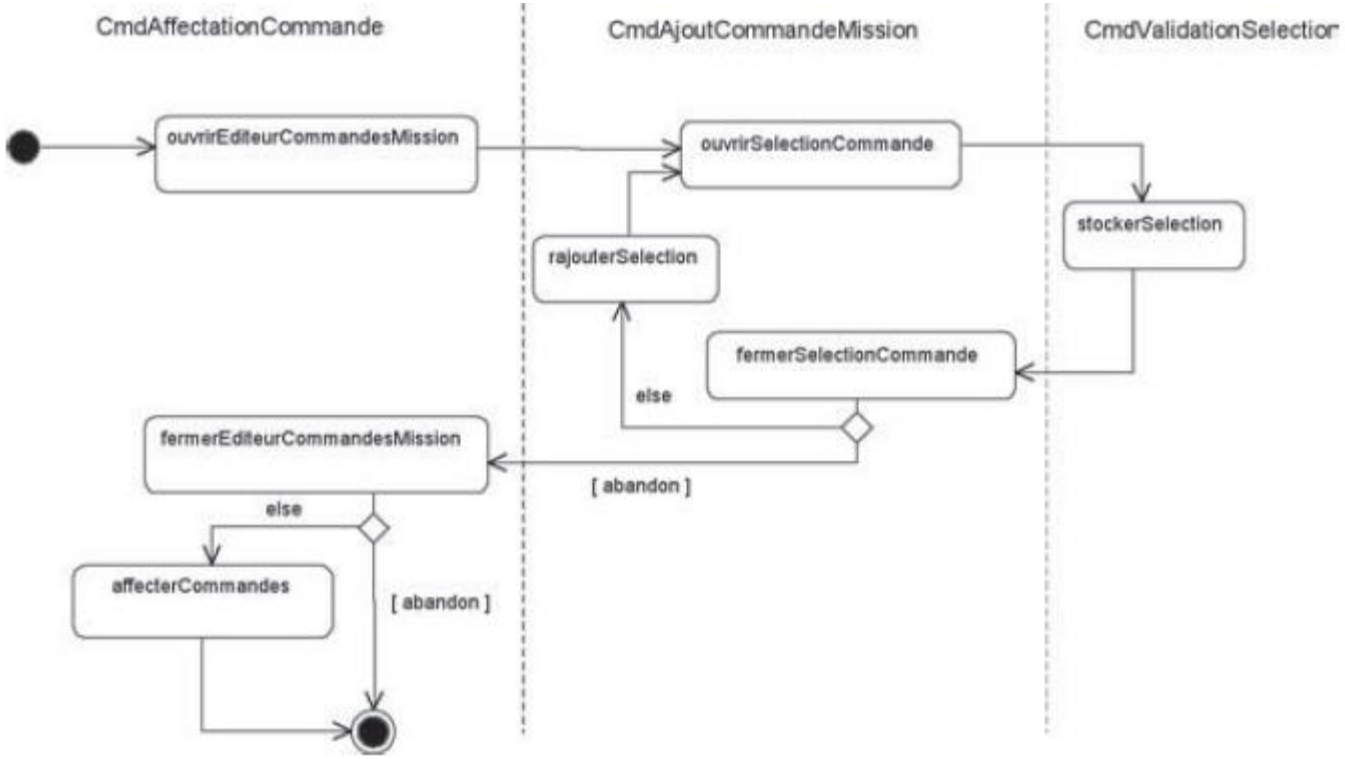


Diagramme d'états-transitions

- **Le diagramme d'états représente le cycle de vie commun aux objets d'une même classe. Ce diagramme complète la connaissance des classes en analyse et en conception.**

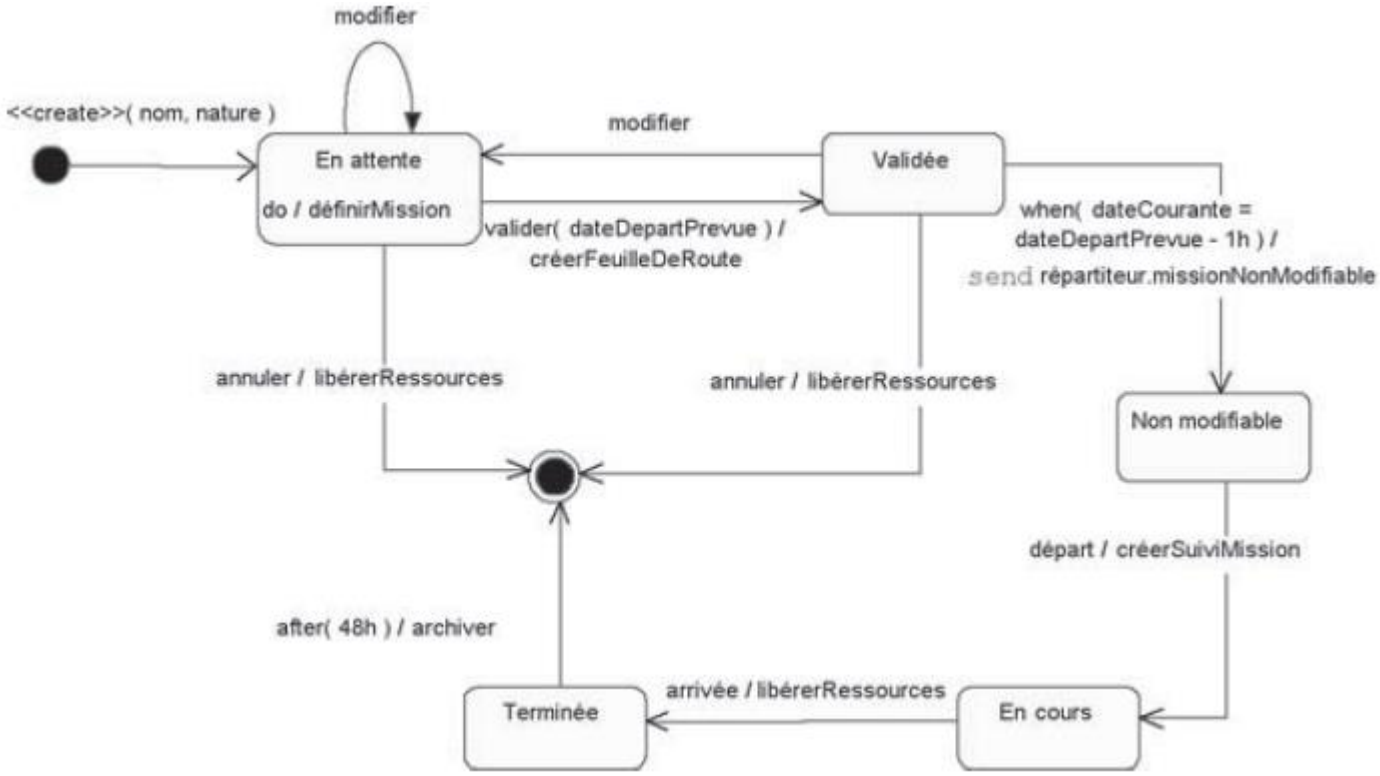
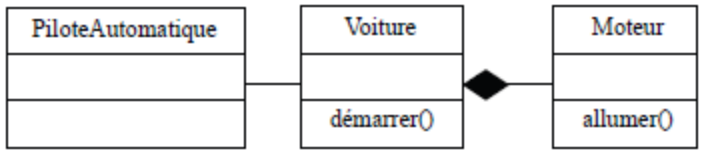


Diagramme de communication

- **Graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre objets.**



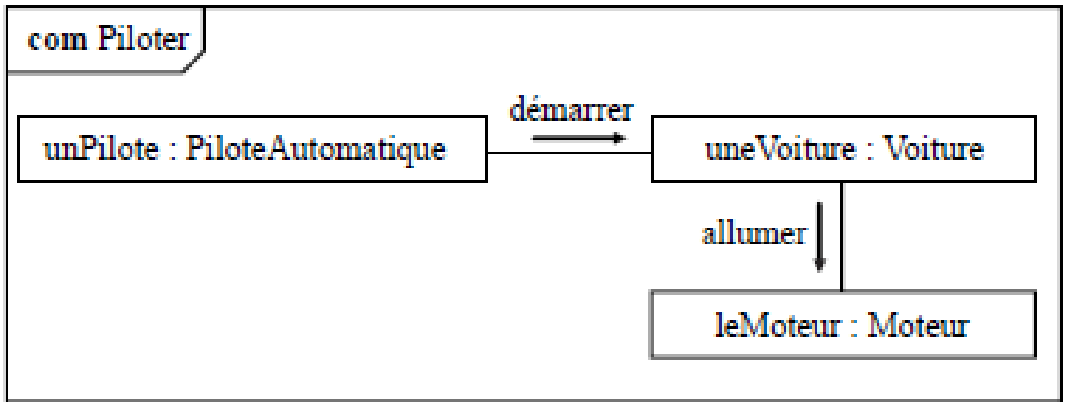


Diagramme de séquence

- **il permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.**

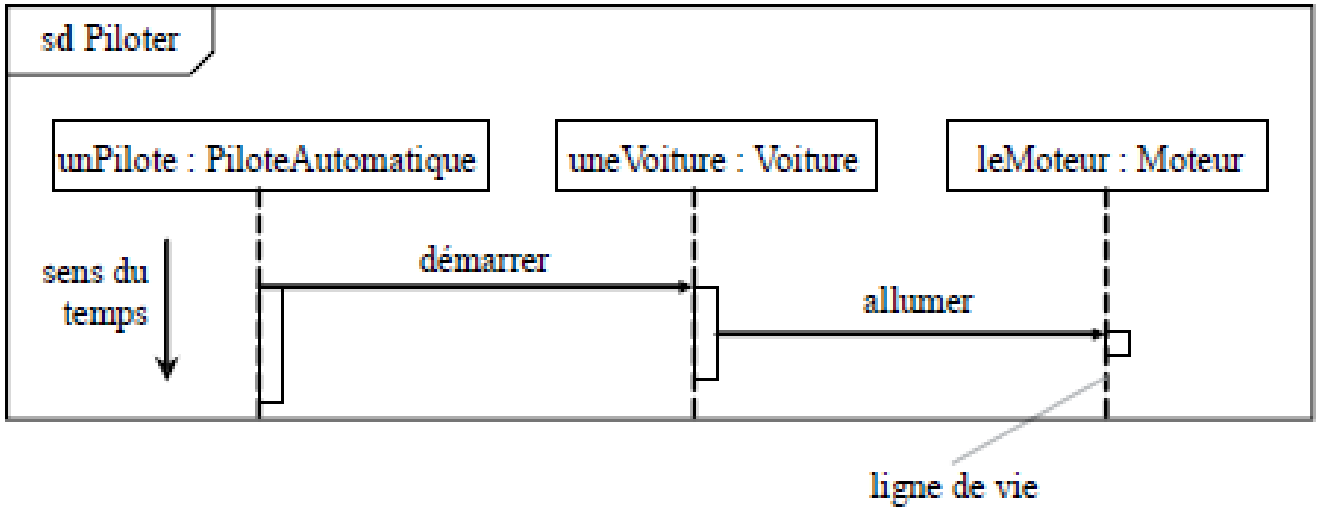


Diagramme global d'interaction

- **Le diagramme global d'interactions (*overview interaction*) a été introduit par UML 2.0. Il propose d'associer les notations du diagramme de séquence avec celles du diagramme d'activité. À ce titre, il peut être aussi bien utilisé en phase d'analyse qu'en phase de conception pour la description d'une méthode complexe.**

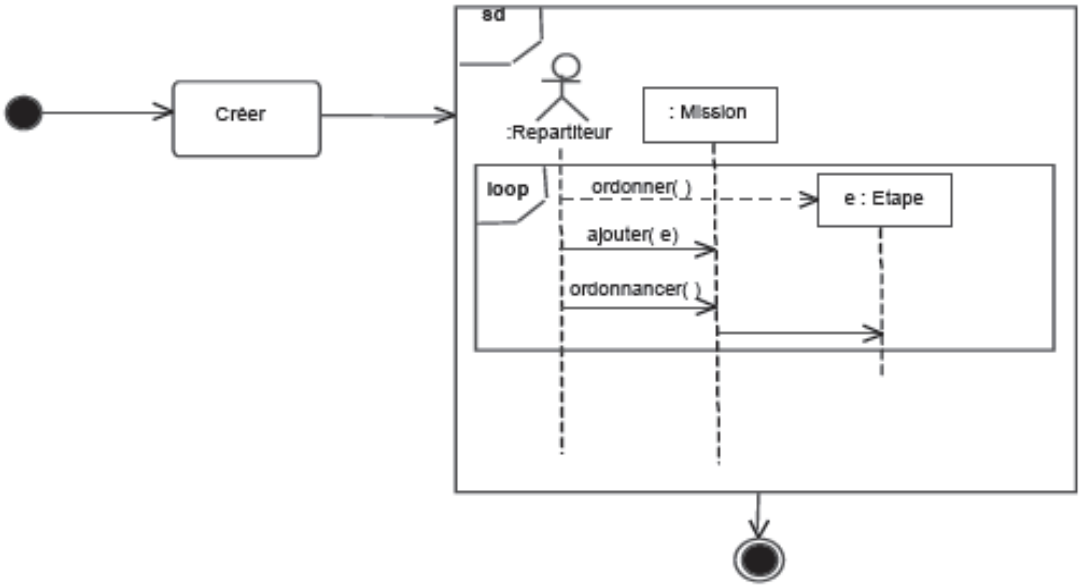
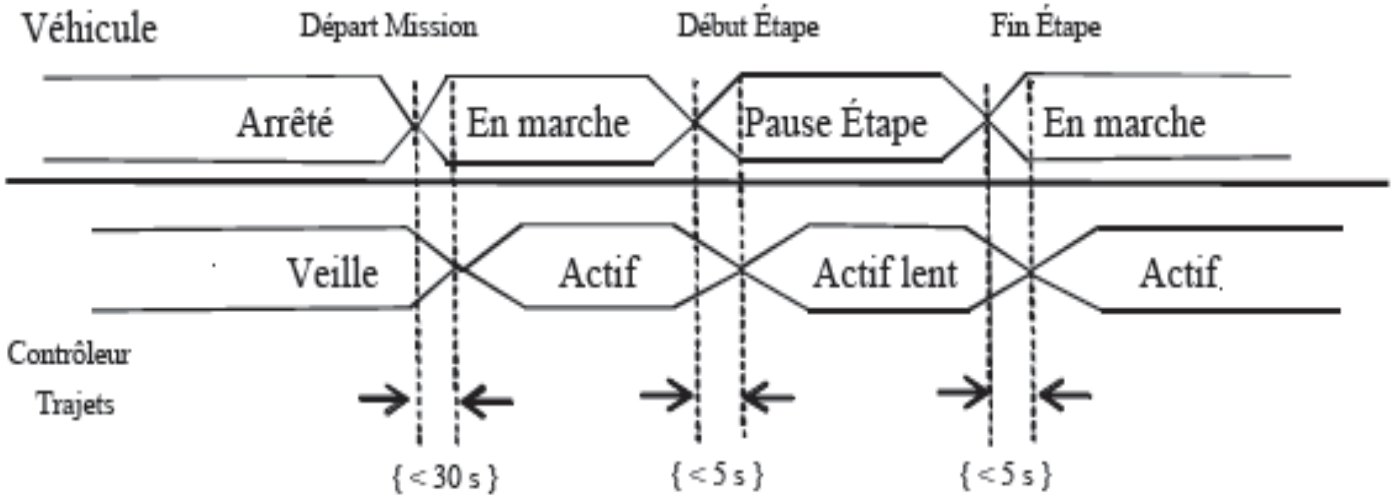


Diagramme de temps (timing)

- **Le diagramme de temps (*timing diagram*) est un nouveau type de formalisme apporté par UML 2.0. Ce dernier provient de techniques connues de l'ingénierie système et répond à des besoins de modélisation très spécifiques lorsque l'interaction entre plusieurs objets exige des contraintes temps-réel extrêmement précises et non équivoques (certain).**



Aller plus loin avec UML

Mécanismes d'extension UML

- **Stereotype (Stéréotype)**
  - **Méta-classe spécialisée (ex: « real-time »)**
  - **Ajout de nouveaux stéréotypes à extension**
- **Tagged value (valeur étiquetée)**
  - **méta-attribut (ex: {abstract})**
  - **Ajout d'un nouveau méta-attribut à extension**
- **Constraint (Contrainte)**
  - **Règle de formation d'expression (ex: {ordered})**
  - **Nouvelles contraintes sur le méta-modèle à extension**

Notion de profil UML

- **Standardisation d'un méta-modèle étendu d'UML**
- **Adapté à un domaine métier ou middleware**
- **Un profil UML peut contenir**
  - **Les éléments sélectionnés dans le méta-modèle de référence**
  - **Des extensions utilisant les différent mécanismes d'extension**
  - **Descriptions sémantiques des extensions**
  - **Notations supplémentaires**
  - **Règles de validation, présentation, transformation**