Langage de modélisation UML

Semaine 2

Généralités

Yaovi HOETOWOU

Objectifs

- décrire les concepts UML
- reconnaître et identifier les diagrammes UML

Contenu

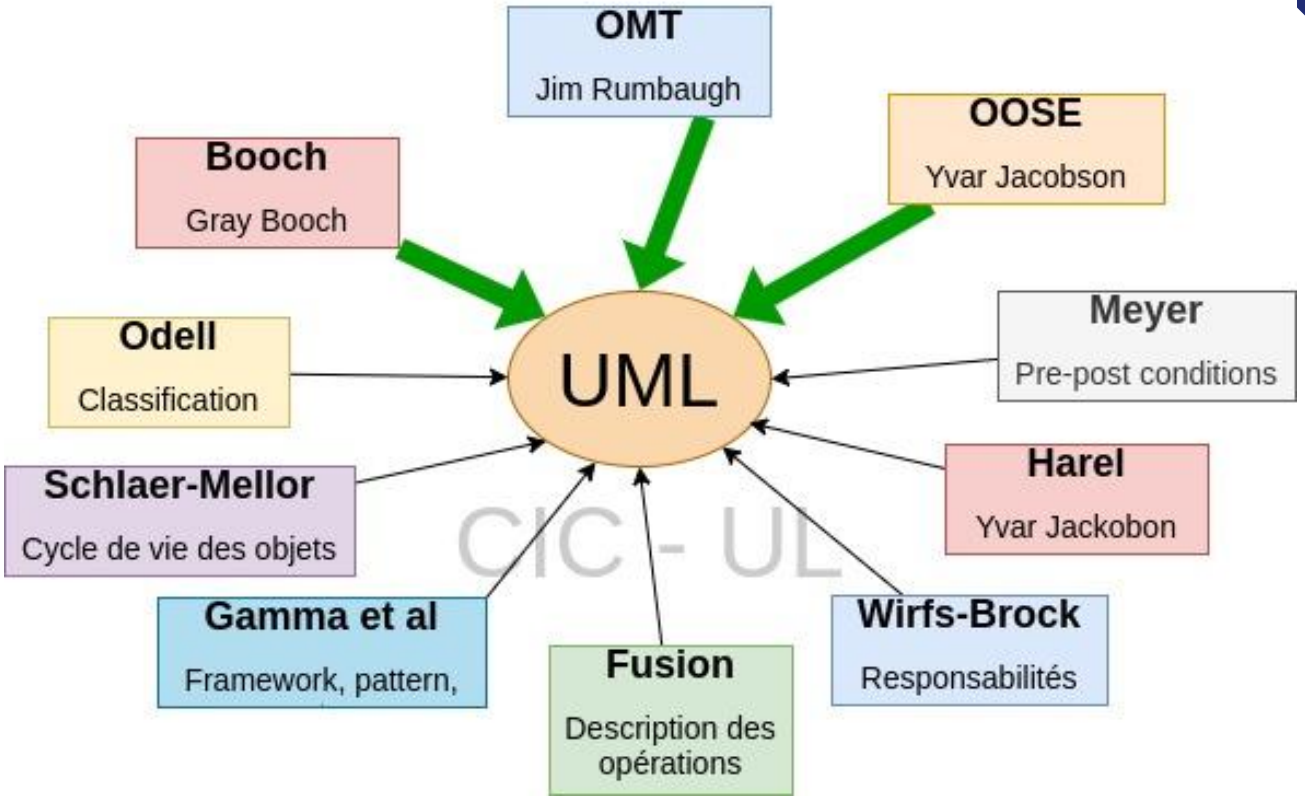
1. À propos du langage UML.
2. Notions de modèles
3. Modèles UML
4. Métamodèles et mécanismes d'extension du langage

1. A propos de UML

1.1- Genèse de UML

- Naissance de l'approche objets dans les années **1960** avec le langage **Simula** suivi par d'autres langages : **C++**, **Java**, **C#**, ...
- Description des objets réalisée de façon formelle selon une syntaxe rigoureuse pas très être lisible par des non-programmeurs.
- Naissance et multiplication des notations graphiques au début des années **1990.**

1.1- Genèse de UML



1.1- Genèse de UML

- Fusion des méthodes **OMT** (Object Modeling Technique) de **Jim Rumbaugh** et **Booch** de **Gray Booch** en **1994**, groupe rejoint par **Yvar Jacobson** en **1995** avec son langage **OOSE** (Object Oriented Software Engineering).
- Naissance du **UML 1.0** en **1997** adopté l'Object Management Group (**OMG**).
- Création d'une Task force chargée de l'évolution de **UML** au sein de l'**OMG.**

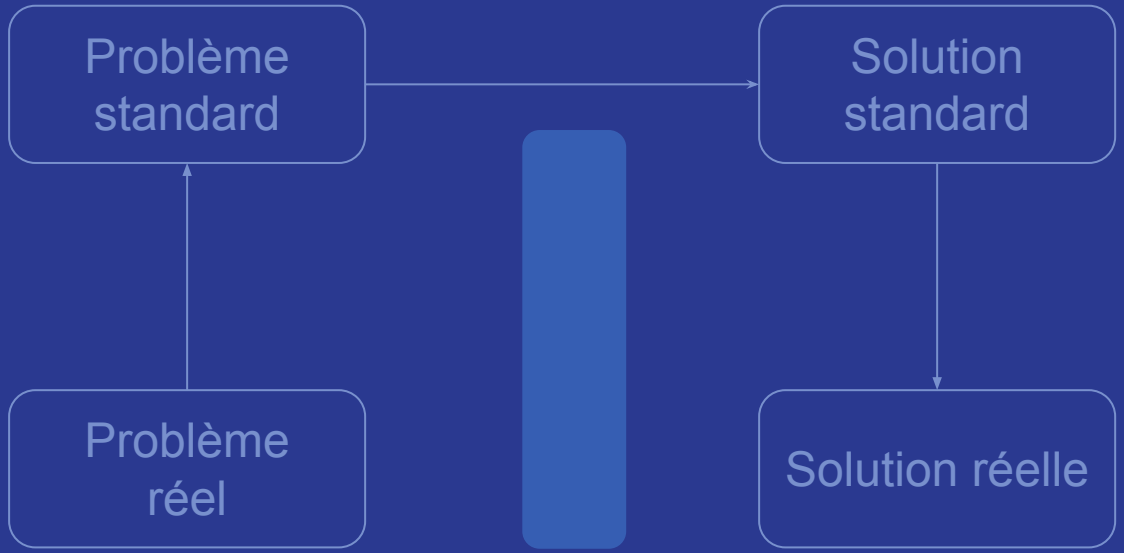
1.2- Les versions de UML

- **01-1997** : Normalisation de **UML 1.0** par **l'OMG**
- **11-1997** : Adoption de **UML 1.1** par **l'OMG**
- **06-1998** : Adoption de **UML 1.2** par **l'OMG**
- **10-1998** : Adoption de **UML 1.3** par **l'OMG**
- **09-2001** : Evolution vers de **UML 1.4**
- **03-2003** : Evolution vers de **UML 1.5**
- **07-2005** : Adoption de **UML 2.0** par **l'OMG**
- **11-2007** : Diffusion de **UML 2.1.2** par **l'OMG**
- **01-2009** : Diffusion de **UML 2.2** par **l'OMG**
- **05-2010** : Diffusion de **UML 2.3** par **l'OMG**
- **07-2011** : Diffusion de **UML 2.4.1** par **l'OMG**
- **12-2017** : Diffusion de **UML 2.5.1** par **l'OMG**

1.3- UML

- Langage de modélisation visuel le plus utilisé pour construire les systèmes orienté objet
- Un langage pour :
  - spécifier,
  - visualiser,
  - construire,
  - et documenter les éléments d'un système logiciel
- De l'anglais **Unified Modeling Language** (**UML**)
- Utilisé en développement logiciel et en conception orientée objet

2. Notion de modèle



source : http://triz-experience.blogspot.com/2011/06/la-modelisation-dun-probleme.html

2.1- Modèle

- Synthétise les deux sens symétriques et opposés de la notion de ressemblance
  - *Sens original* : donner une représentation d'un objet réel que l'on cherche à imiter
  - *Sens scientifique* : construire d'abord un prototype, concret ou conceptuel, qui servira de « modèle » à une construction réelle

2.1- Modèle

- Un modèle est une vue **abstraite** (simplifiée) d'un système, d'un problème
  - décrit le système par rapport à un point de vue spécifique
- "Un modèle informatique est une représentation simplifiée de la réalité en vue de réaliser un traitement avec un ordinateur "

2.2- Modélisation

- Modélisation = conception d'un modèle
- "Opération par laquelle on établit le modèle d'un système complexe, afin d'étudier plus commodément et de mesurer les effets sur ce système des variations de tel ou tel de ses éléments composants"

(Giraud-Pamart Nouv. 1974)

2.2- Modélisation

- Le développement d'un système logiciel industriel nécessite la création de modèles
- Un système complexe s'appréhende mieux à travers un petit ensemble de vues indépendantes.
- Chaque modèle peut être représenté à différents niveaux de fidélité ;
- UML fournit des outils pour concevoir des modèles : ce qui le classe au rang des **langages de modélisation**

2.2- Modélisation

➔ Un langage de modélisation rigoureux comporte:

- **Des éléments de modélisation** : les concepts de modélisation fondamentaux et leur sémantique ;
- **Une Notation** : le rendu visuel des éléments de modélisation ;
- **Des directives** : des idiomes pour le bon usage du langage.

2.3- Appréciation d'un modèle

- Un modèle est une abstraction
  - une simplification de la réalité
  - ce n'est pas la réalité
  - il n'est jamais complètement fidèle par construction
- Un modèle doit être simple expressif d'un point de vue du système
- Le seul modèle complètement fidèle à la réalité est la réalité elle-même, et ce n'est donc pas un modèle

3. Les modèles UML

Diagramme UML

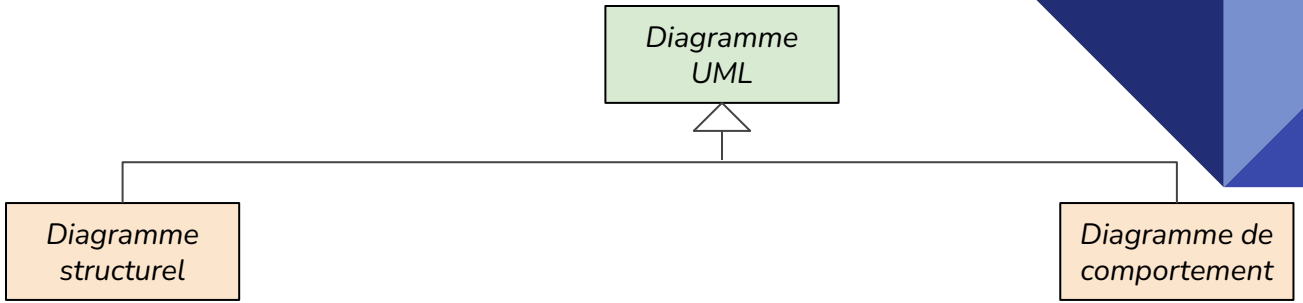


Diagramme UML
Diagramme structurel
Diagramme de comportement

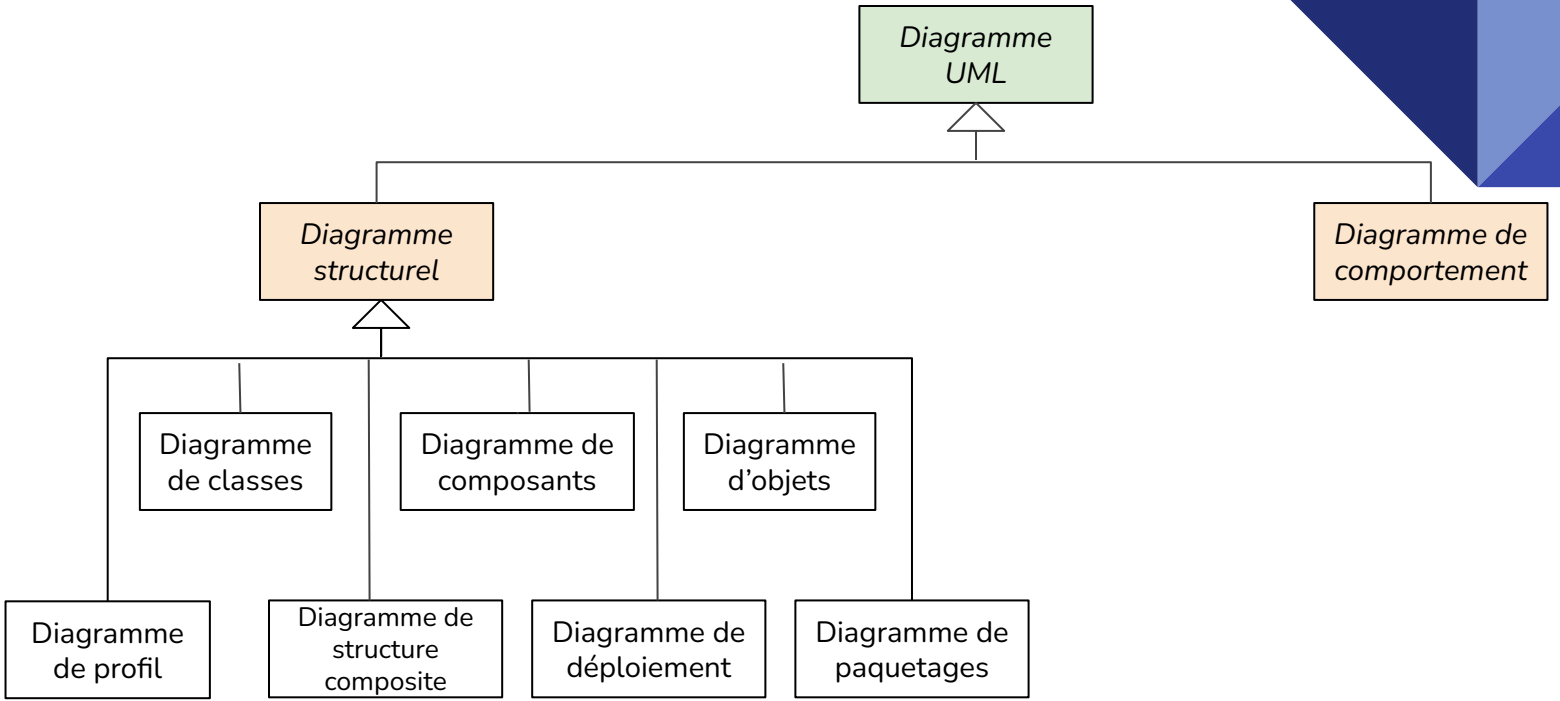


Diagramme UML
Diagramme structurel
Diagramme de comportement
Diagramme de classes
Diagramme de composants
Diagramme d'objets
Diagramme de profil
Diagramme de structure composite
Diagramme de déploiement
Diagramme de paquetages

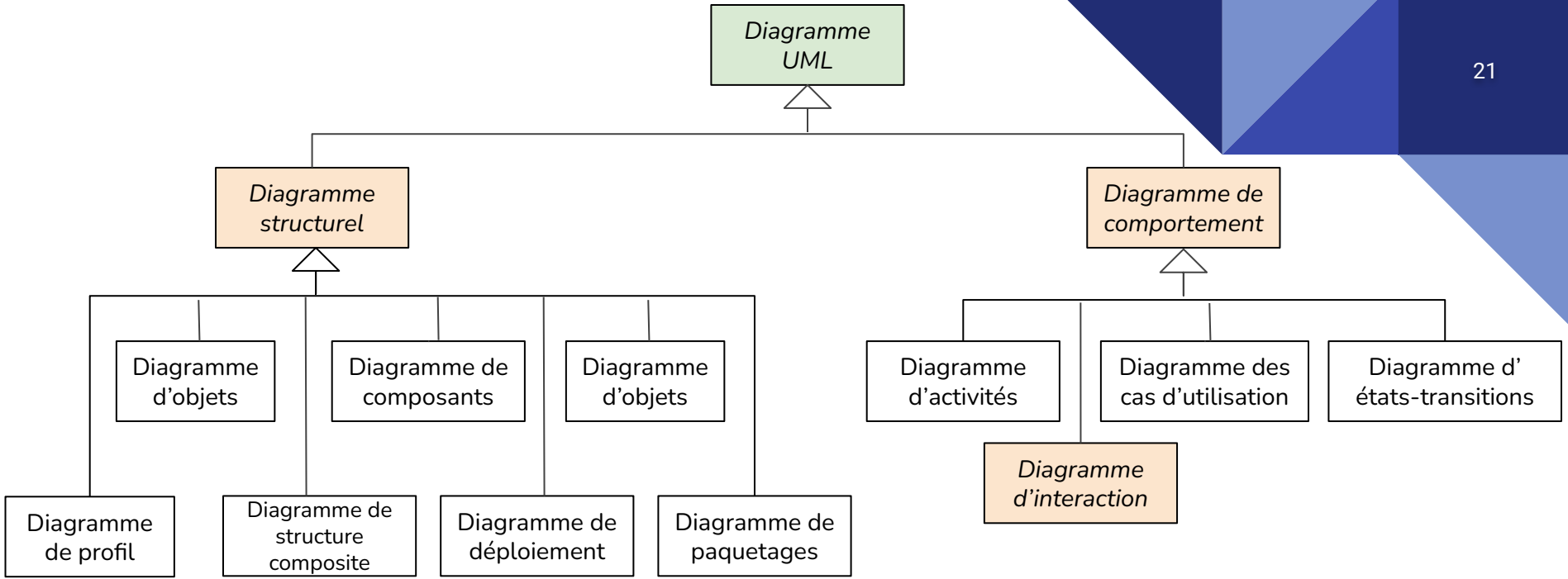


Diagramme UML
Diagramme structurel
Diagramme de comportement
Diagramme d'objets
Diagramme de composants
Diagramme d'objets
Diagramme d'activités
Diagramme des cas d'utilisation
Diagramme d' états-transitions
Diagramme de profil
Diagramme de structure composite
Diagramme de déploiement
Diagramme de paquetages
Diagramme d'interaction

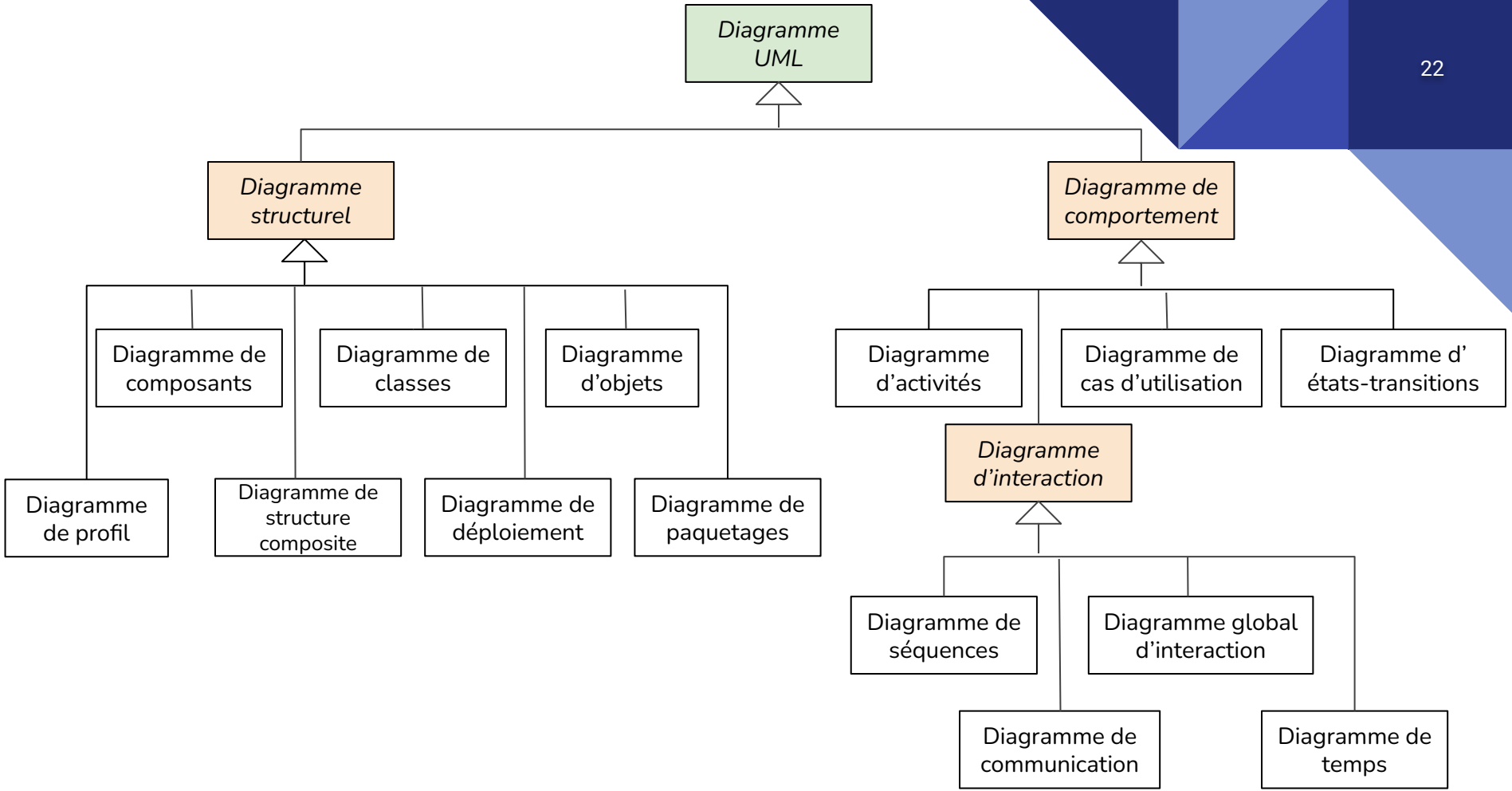
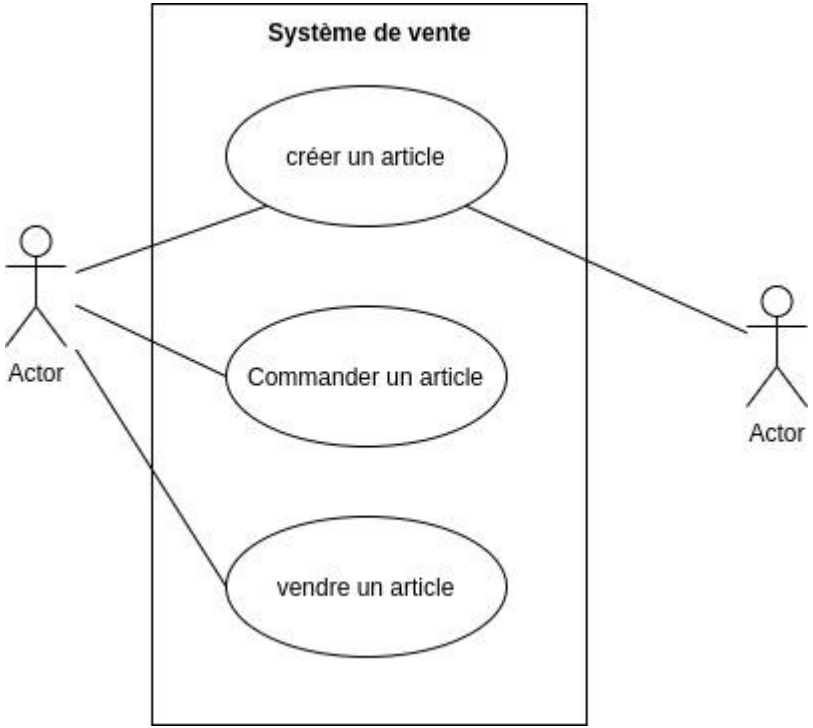


Diagramme UML
Diagramme structurel
Diagramme de comportement
Diagramme de composants
Diagramme de classes
Diagramme d'objets
Diagramme d'activités
Diagramme de cas d'utilisation
Diagramme d' états-transitions
Diagramme de profil
Diagramme de structure composite
Diagramme de déploiement
Diagramme de paquetages
Diagramme d'interaction
Diagramme de séquences
Diagramme global d'interaction
Diagramme de communication
Diagramme de temps

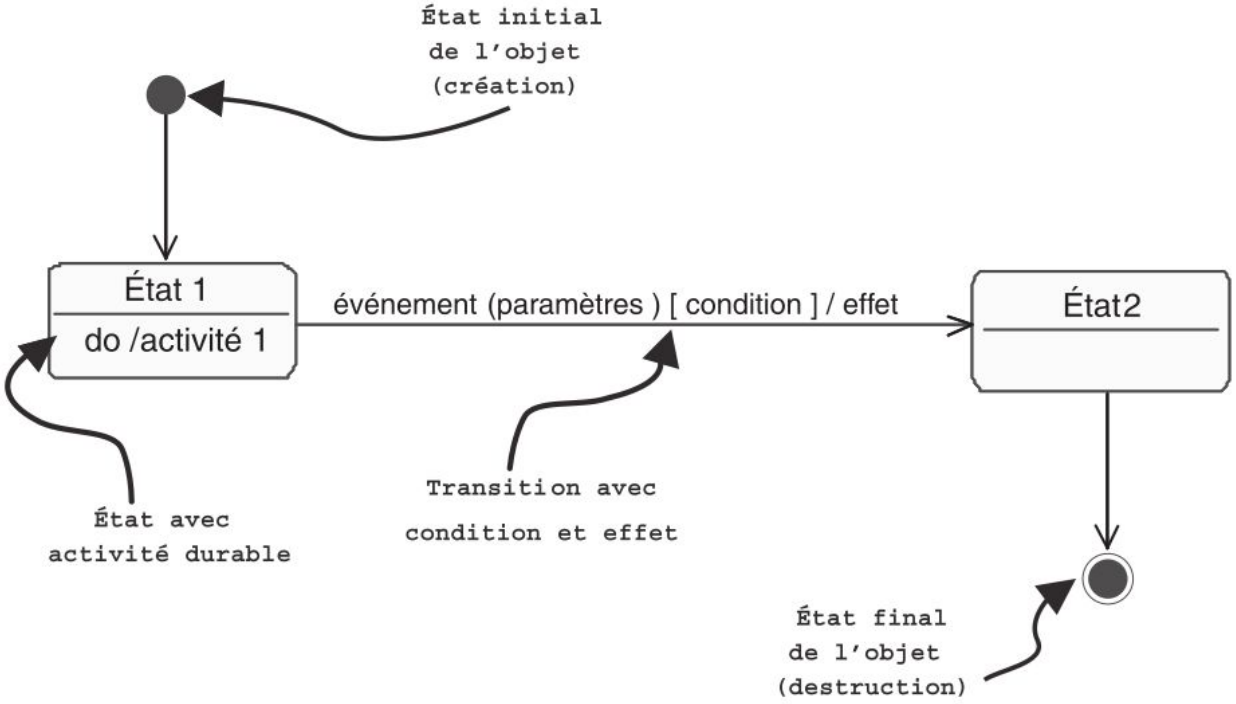
3.1- Diagramme de cas d'utilisation



3.1- Diagramme de cas d'utilisation

- Utilisé pour donner une vision globale du **comportement fonctionnel** d'un système logiciel.
- composé de cas d'utilisation (use case en anglais)
- **Cas d'utilisation** : représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système
- Les utilisateurs:
  - sont appelés **acteur** (**actor** en anglais)
  - ils interagissent avec les cas d'utilisation

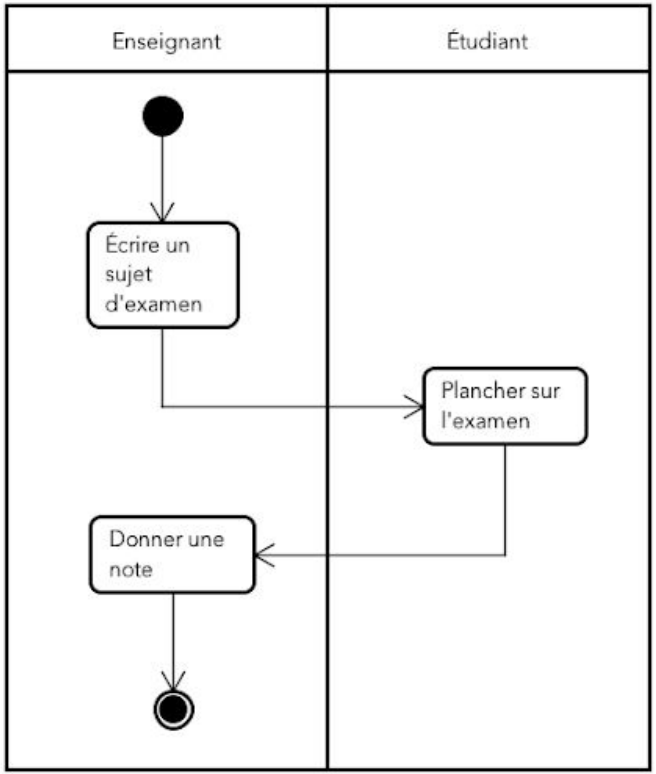
3.2- Diagramme d'états-transitions



3.2- Diagramme d'états

- Permettent de décrire précisément le comportement d'une instance d'une classe en terme d'états et d'événements au moyen d'un automate (stateschart) associé à la classe de l'objet.
- Intéressant pour les objets qui ont un comportement réactif marqué.
- Cycle de vie d'un objet
  - Évolution de l'état d'un objet,
  - Comportement face à l'arrivée d'évènement.
- Une forme spéciale du DET est utilisée pour modéliser les processus métiers et opérations complexes

3.3- Diagramme d'activité

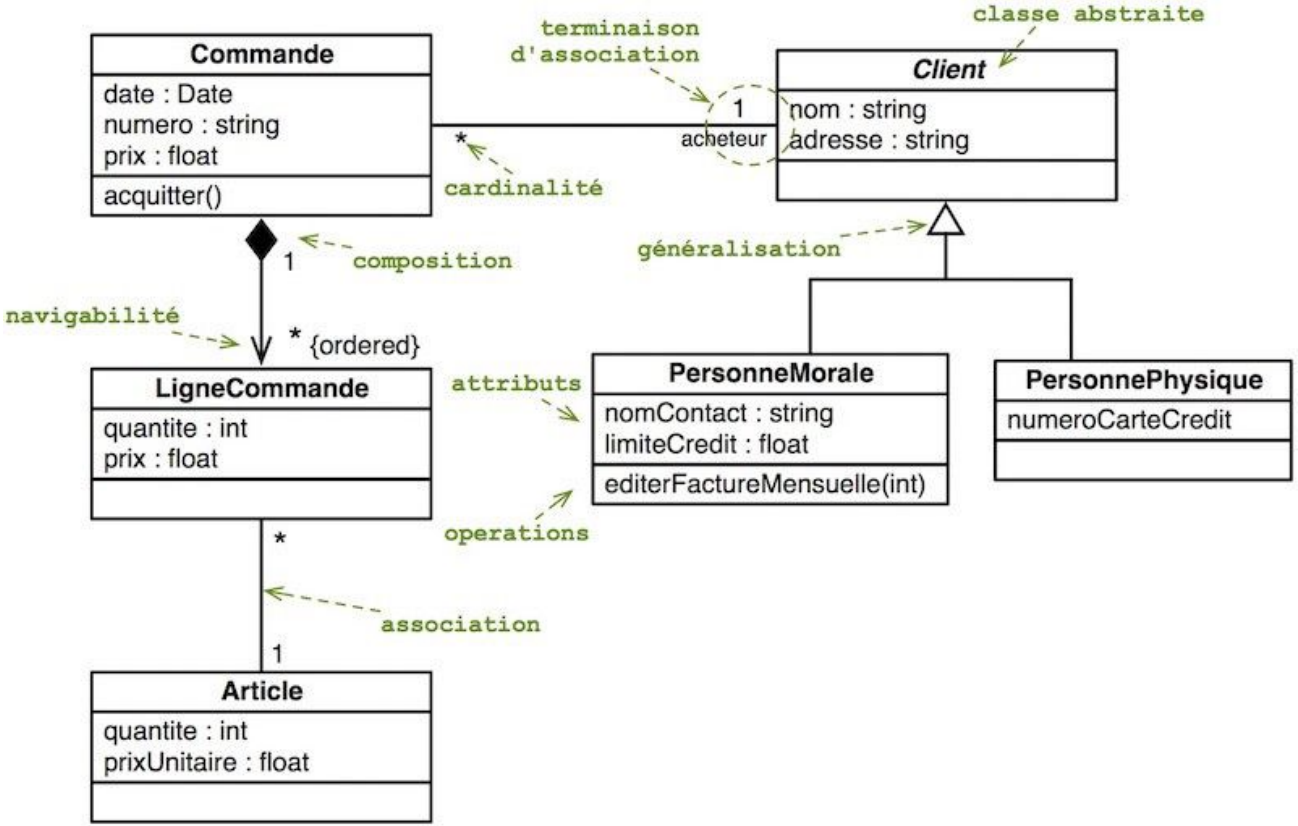


Source : https://fr.wikipedia.org/wiki/UML_(informatique)

3.3- Diagramme d'activité

- Un diagramme d'activités représente la suite d'étapes qui constituent un processus complexe (processus métier, algorithme, méthode, etc.) et les contraintes de séquencement.
- Il exprime le flux de contrôle en se concentrant sur les opérations plutôt que sur les objets.
- Les nœuds sont principalement les activités.
- Certaines activités se déroulent en continu jusqu'à ce qu'un évènement extérieur ne les interrompe, mais la plupart finissent par achever leur travail et s'interrompent d'elles mêmes.

3.4- Diagramme de classes

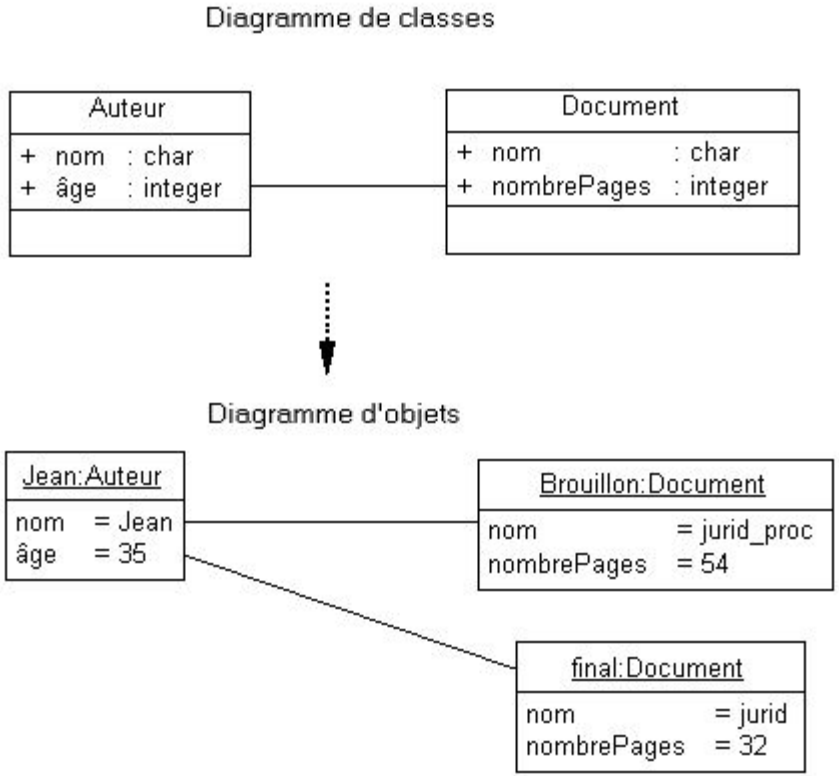


Source : https://medium.com/@manurnx/

3.4- Diagramme de classes

- Schéma utilisé en génie logiciel pour présenter
  - les classes et les interfaces des systèmes
  - les différentes relations entre elles
- Fait abstraction des aspects temporels et dynamiques.
- **Classe** : ensemble de fonctions (méthodes) et de données (attributs) qui sont liées ensemble par un champ sémantique

3.5- Diagramme d'objets

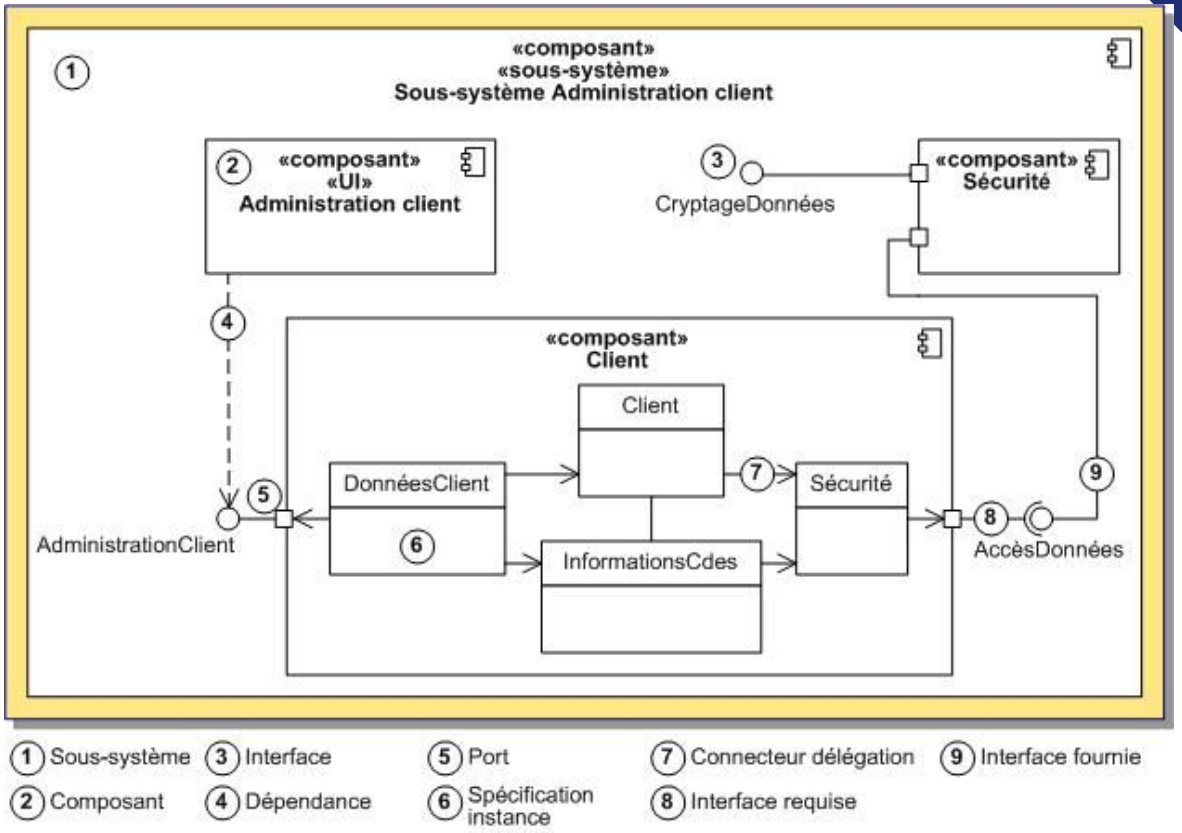


Source : http://infocenter.sybase.com

3.5- Diagramme d’objets

- Permet de représenter les instances des classes
- Exprime les relations qui existent entre les objets
- Montre l'état des instances d'objet avant et après une interaction

3.6- Diagramme de composants

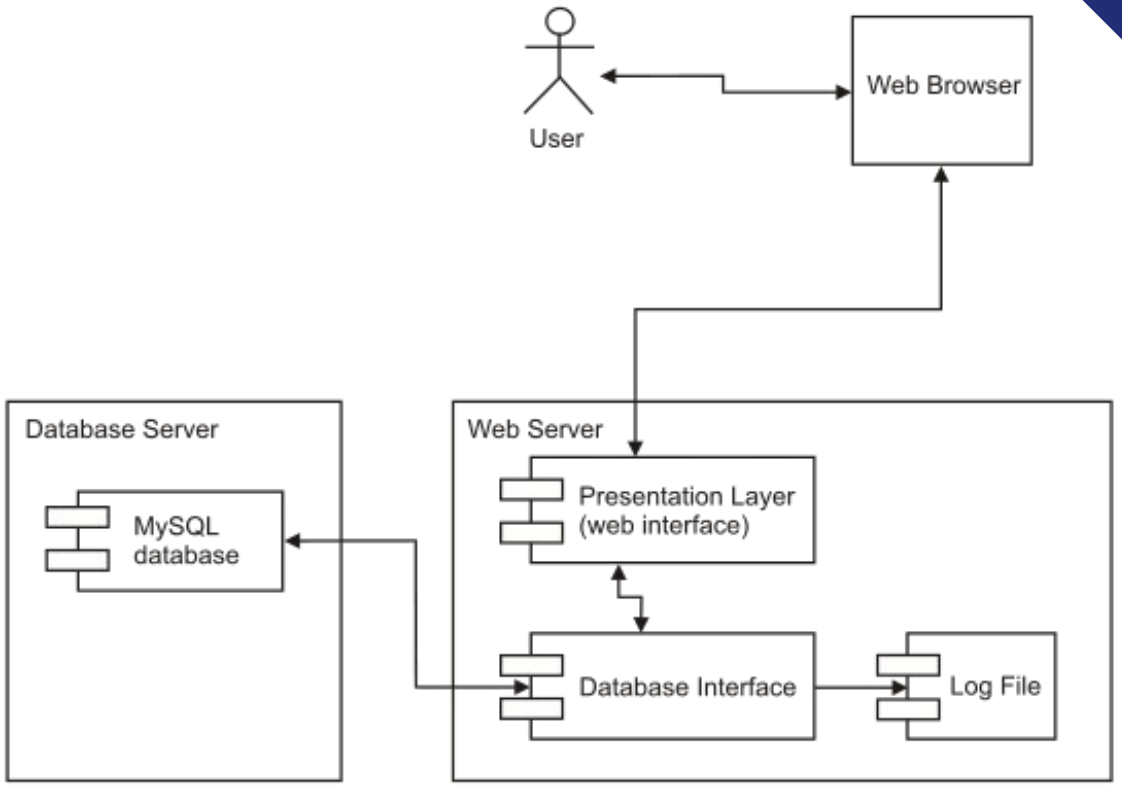


Source : http://docs.embarcadero.com/products/rad_studio/radstudio2007

3.6- Diagramme de composants

- Décrit l'organisation du système du point de vue des éléments logiciels
  - les modules (paquetages, fichiers sources, bibliothèques, exécutables)
  - données (fichiers, bases de données)
  - éléments de configuration (paramètres, scripts, fichiers de commandes)
- permet de mettre en évidence les dépendances entre les composants (**qui utilise quoi**)

3.7- Diagramme de déploiement



Source : https://fr.wikipedia.org/wiki/UML_(informatique)

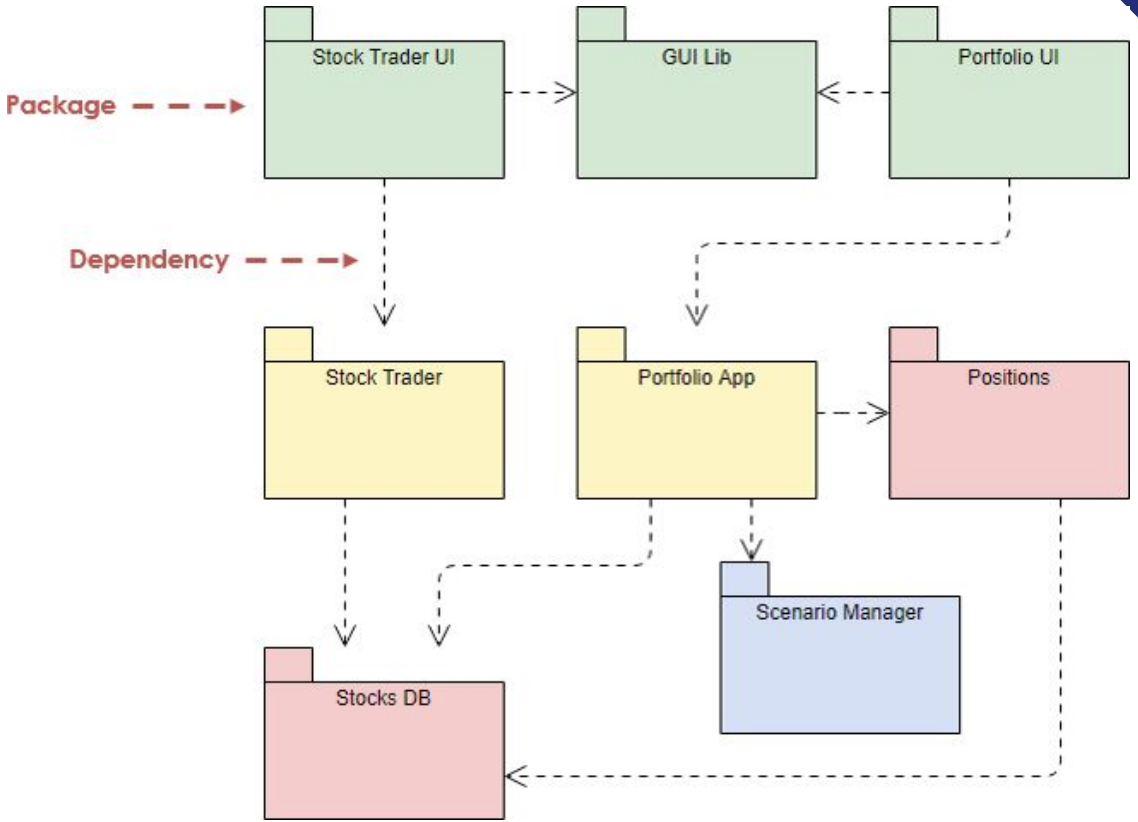
3.7- Diagramme de déploiement

- Vue statique qui décrit
  - l'utilisation de l'infrastructure physique par le système,
  - la répartitions des composants du système,
  - les relations entre les composants.

3.8- Diagramme de profil

- S'inscrit dans la rubrique des possibilités d'extension de UML.
- Permet de spécialiser chaque application suivant son contexte.
- Est caractérisé principalement par :
  - une liste des stéréotypes
  - une liste des valeur marquées
  - un ensemble des contraintes

3.9- Diagramme de paquetage



Source : https://online.visual-paradigm.com/diagrams/tutorials/package-diagram-tutorial/

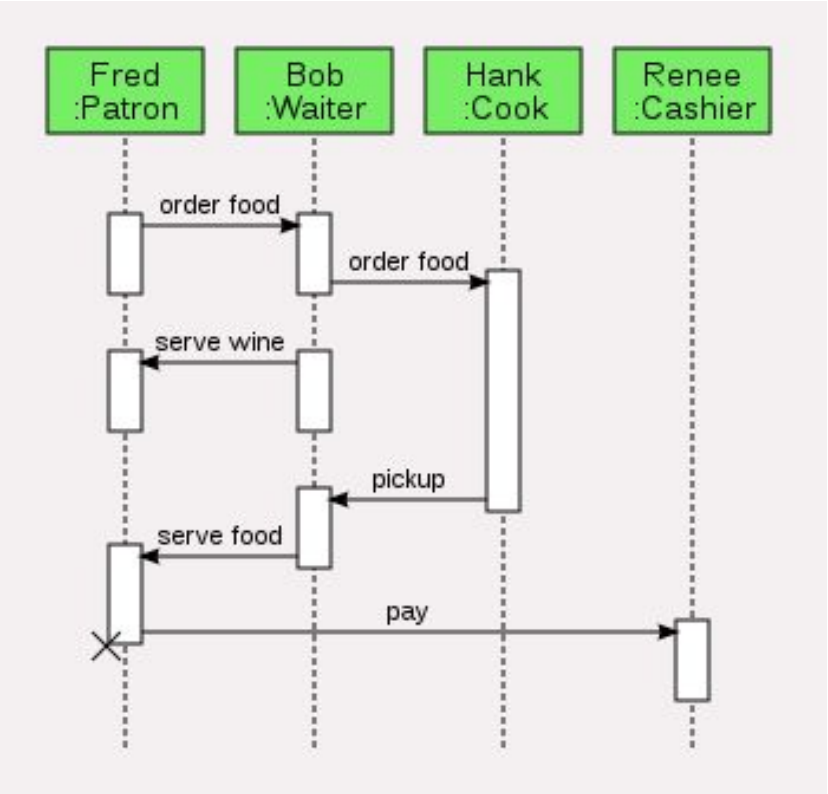
3.9- Diagramme de paquetage

- Les Paquetages servent à organiser les éléments de modélisation en groupes avec pour objectif de rendre les diagrammes plus simples et faciles à comprendre.
- Utilisés principalement sur les diagrammes de classe et les diagrammes de CU.
- Peuvent structurer n'importe quel type de diagramme

3.10- Diagramme de structure composite

- Ensemble d'éléments interconnectés collaborant dans un but commun lors de l'exécution d'une tâche
- Expose la structure interne d'une classe ainsi que les collaborations que cette dernière rend possible
- Chaque élément se voit attribuer un rôle dans la collaboration

3.11- Diagramme de séquence

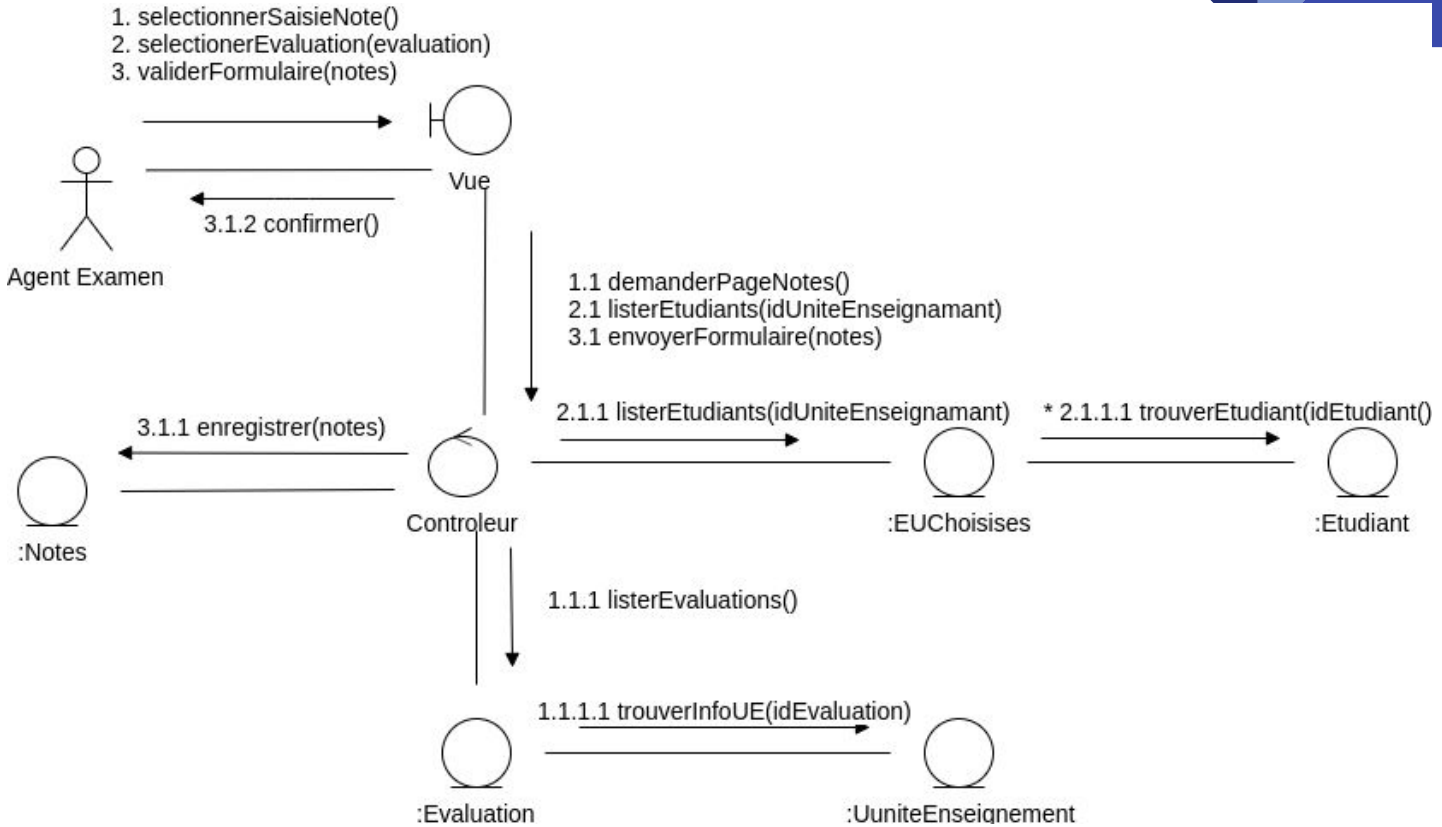


Source : https://fr.wikipedia.org/wiki/UML_(informatique)

3.11- Diagramme de séquence

➔ permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation

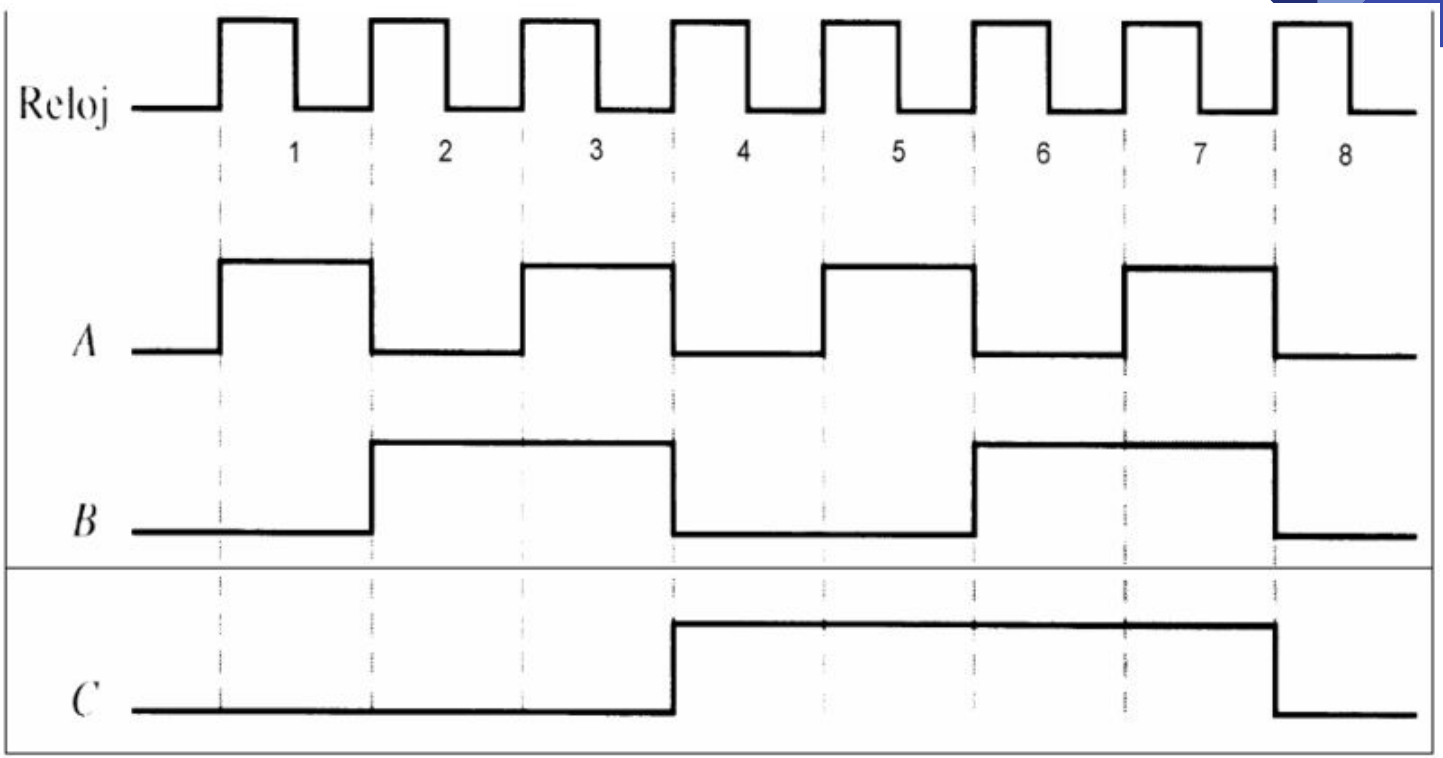
3.12- Diagramme de communication



3.12- Diagramme de communication

➔ Représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets

3.13- Diagramme de temps

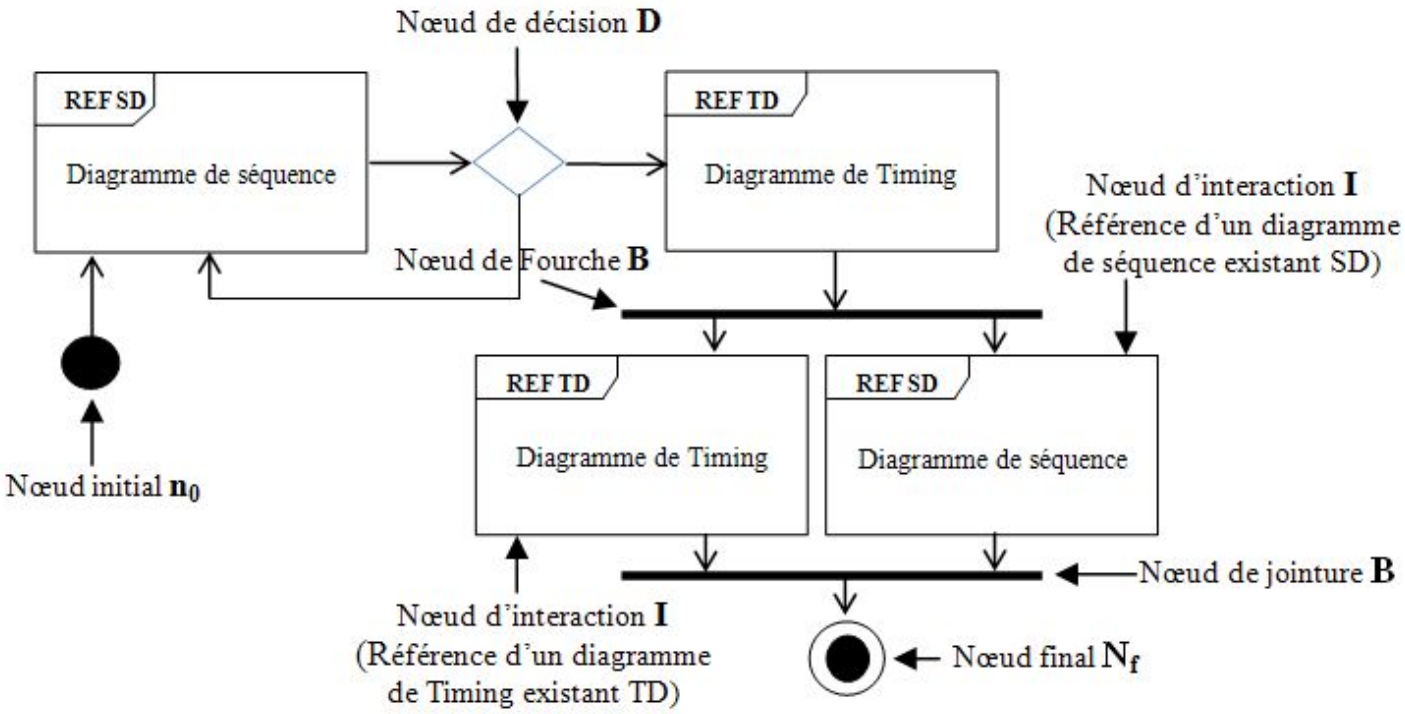


Source : https://fr.wikipedia.org/wiki/UML_(informatique)

3.13- Diagramme de temps

➔ Utilisé pour explorer le comportement des objets d'un système à travers une période déterminée

3.14- Diagramme global d'interaction



Source : www.researchgate.net/figure/Exemple-dun-diagramme-global-dinteraction-Notament-les-memes-auteurs-ont-formellement_fig6_320194055

3.14- Diagramme global d'interaction

➔ Utilisé pour rendre compte de l'organisation spatiale des participants à l'interaction.

4. Métamodèles et d'extension du langage UML

4.1- Méta-modèle

- Méta-modèle = modèle de modèle
- L'OMG définit 4 niveaux de modélisation
  - **M0** : système réel, système sujet à une modélisation
  - **M1** : modèle du système réel défini dans un certain langage
  - **M2** : méta-modèle définissant ce langage
  - **M3** : méta-méta-modèle définissant le méta-modèle

4.1- 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

4.2- Stéréotype

Classe « control »	Classe « boundary »	Classe « acteur »

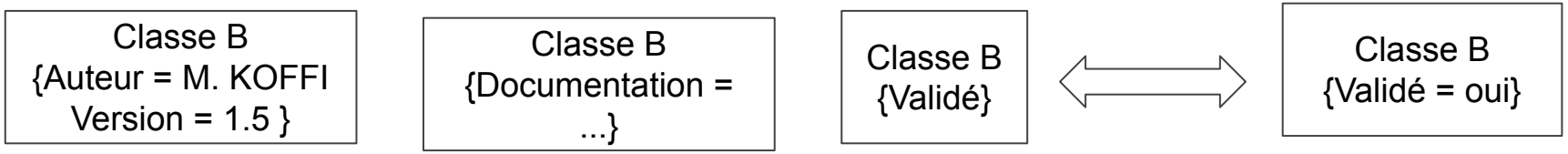
- Formalisme : nom du stéréotype indiqué entre guillemets
- Constitue un moyen de classer les éléments d'un modèle
- S'applique à n'importe quel concept d'UML
- Un certain nombre de stéréotypes sont déjà définis dans UML (voir annexe C de la norme)

4.2- Stéréotype

- Possibilité de rajouter d'autres stéréotypes
  - évolution général de UML
  - prise en compte de situations particulières propres aux entreprises
- Dans un diagramme de classe
  - permet de définir de nouveaux types de classe

4.2- Valeur marquée / étiquette

- Une valeur marquée est une paire (Nom, Valeur) qui ajoute une nouvelle propriété à un élément de modélisation.
- La spécification d'une valeur marquée prend la forme
  - Nom = Valeur
- Une valeur marquée est indiquée entre accolades: { }
- Il est possible d'utiliser des valeurs marquées prédéfinies (ex: documentation) ou personnalisées (ex: Auteur, Version ...)



4.3- Profil

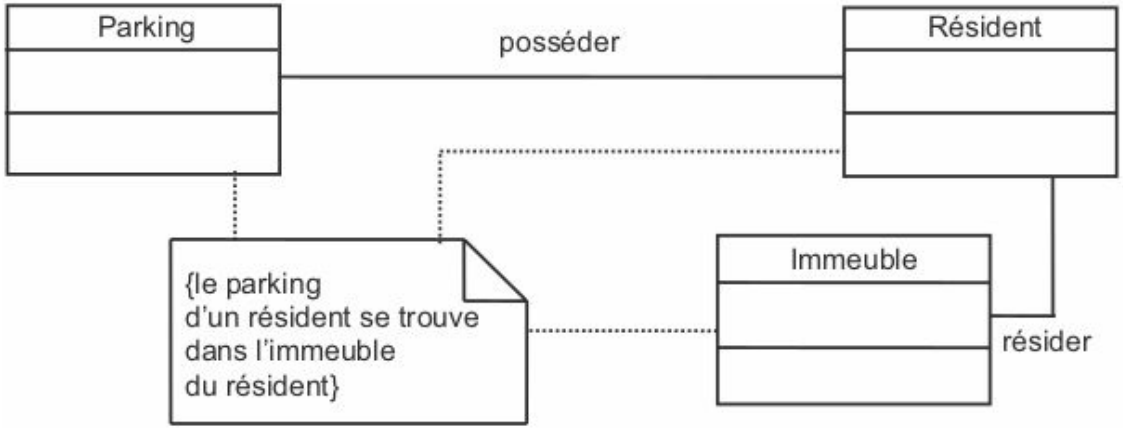
- Mécanisme d'extension de métamodèles défini dans le standard Meta-Object Facility (MOF)
- caractérisé principalement par :
  - la liste des stéréotypes,
  - la liste des valeurs marquées
  - et les contraintes spécifiées
- **Objectif :** permet de définir de nouveaux éléments du métamodèle, mieux adaptés à la modélisation de domaines d'application particuliers.

4.4- Note

- Correspond à un commentaire explicatif d'un élément d'UML
- Exemple :

4.4- Contrainte

- **Contrainte :** Note ayant une valeur sémantique particulière pour un élément du modèle
- Première forme d'écriture d'une contrainte : **{ceci est une contrainte}**.
- Deuxième forme d'écriture : à l'intérieur d'une note



4.4- Contrainte

- L'utilisateur peut définir de nouvelles contraintes dans un langage libre ou en **OCL** (Object Constraint Language) .