

UNIFIED
MODELING
LANGUAGE



MODÉLISATION LOGICIELLE

CHAPITRE 1A – **PRÉSENTATION GÉNÉRALE**

Modélisation Logicielle

1- Présentation générale

■ Qu'est ce que UML ?

- UML (**U**nified **M**odeling **L**anguage ou « langage de modélisation unifié ») est une notation permettant de modéliser un problème de façon standard.
- Point de convergence des notations utilisées dans le domaine de l'analyse et la conception objet.
- Fusion et synthèse des formalismes utilisés par les trois méthodes historiques OMT, BOOCH et OOSE qui ont marqué le début de l'approche objet.

1- Présentation générale

■ Pourquoi UML ?

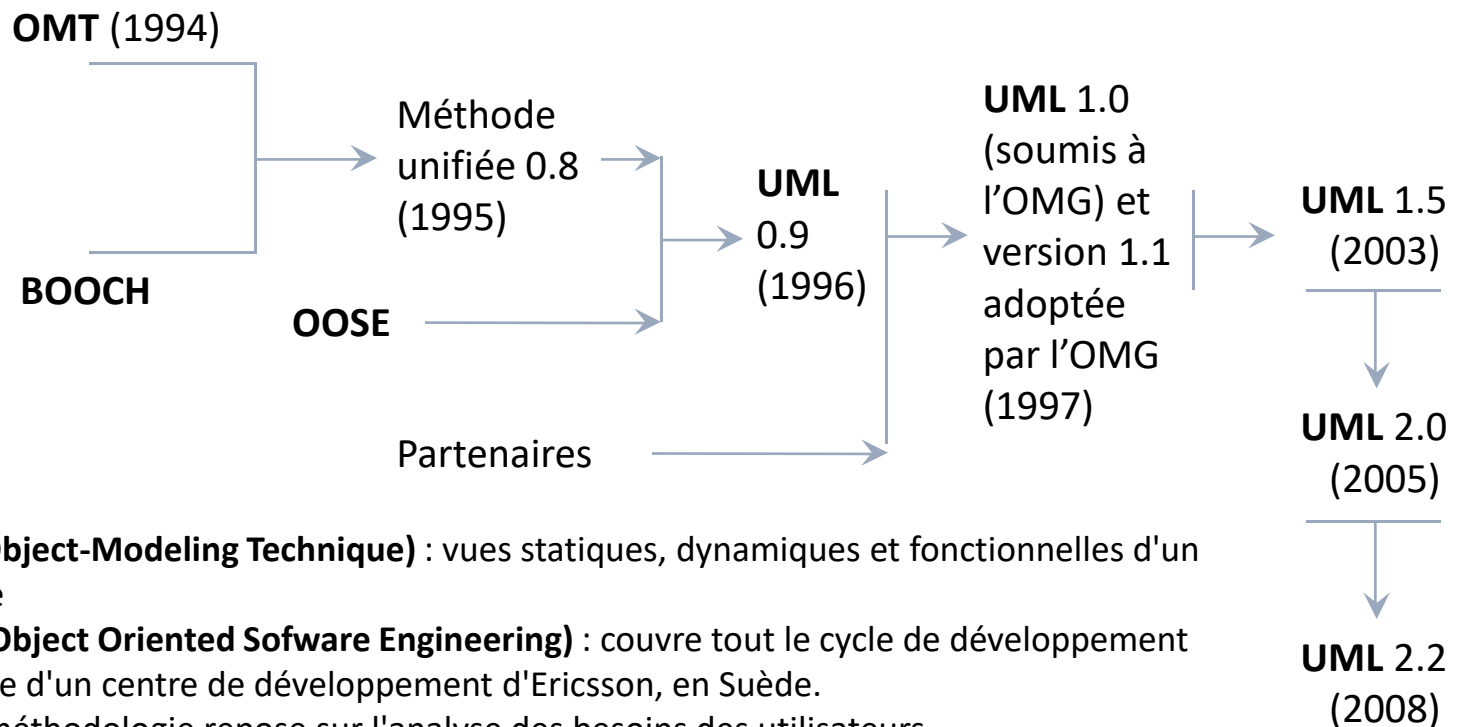
- De la même façon qu'il vaut mieux dessiner une maison avant de la construire, il vaut mieux modéliser un système avant de le réaliser.

■ A quoi sert UML ?

- d'obtenir une **modélisation** de très **haut niveau indépendante** des **langages** et des **environnements**
- de faire **collaborer** des participants de tout horizon autour d'un même document de synthèse
- de **documenter** un projet
- de générer automatiquement la partie logiciel d'un système
- de **modéliser** des systèmes non logiciels.
 - Par exemple, le work flow d'un processus dans une unité de fabrication, etc.

1- Présentation générale

■ La genèse d'UML : Historique des méthodes d'analyse



OMT (Object-Modeling Technique) : vues statiques, dynamiques et fonctionnelles d'un système

OOSE (Object Oriented Software Engineering) : couvre tout le cycle de développement

- Issue d'un centre de développement d'Ericsson, en Suède.
- La méthodologie repose sur l'analyse des besoins des utilisateurs.

BOOCH : vues logiques et physiques du système

1- Présentation générale

Les points forts d'UML

- **UML est un langage formel et normalisé**
 - gain de précision
 - gage 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.

CHAPITRE 1B – DÉMARCHE GÉNÉRALE DE MODÉLISATION

Modélisation Logicielle

Le modèle

Qu'est ce qu'un modèle ?

- Abstraction de la réalité
- Frontière entre la réalité et la perspective de l'observateur. Ce n'est pas « la réalité », mais une vue très subjective de celle-ci.
- Reflet des aspects importants de la réalité, il en donne donc une **vue juste** et **pertinente**.

Le modèle

Caractéristiques

- Le caractère abstrait d'un modèle doit notamment permettre :
 - de **faciliter** la compréhension du système étudié : un modèle réduit la complexité du système étudié.
 - de **simuler** le système étudié : un modèle représente le système étudié et reproduit ses comportements.

Comment modéliser avec UML ?

- UML est un langage qui permet de représenter des modèles.
- UML n'est pas une méthode de modélisation. Il ne définit donc pas le processus d'élaboration des modèles.

Comment modéliser avec UML ?

- Cependant, dans le cadre de la modélisation d'une application informatique, la démarche préconisée est la suivante :
 1. Itérative et incrémentale
 2. Orientée besoins utilisateurs
 3. Orientée Architecture

Comment modéliser avec UML ?

1- Démarche itérative et incrémentale

- Pour comprendre et représenter un système complexe (le modéliser), il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes.
- Cette démarche doit aussi s'appliquer au cycle de développement dans son ensemble.

Comment modéliser avec UML ?

2- Démarche orientée besoins utilisateurs

- 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).

Comment modéliser avec UML ?

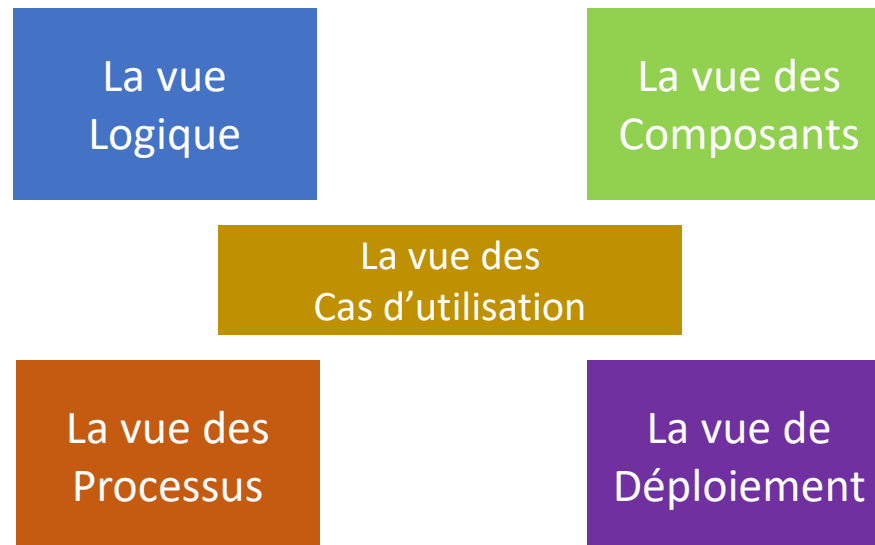
2- Démarche orientée besoins utilisateurs

- Les besoins des utilisateurs servent aussi de fil rouge, tout au long du cycle de développement (itératif et incrémental) :
 1. à chaque itération de la **phase d'analyse**, on clarifie, affine et valide les besoins des utilisateurs.
 2. à chaque itération de la **phase de conception et de réalisation**, on veille à la prise en compte des besoins des utilisateurs.
 3. à chaque **itération de la phase de test**, on vérifie que les besoins des utilisateurs sont satisfaits.

Comment modéliser avec UML ?

3- Démarche orientée architecture

- Une architecture adaptée garantit le succès du développement d'un système
- Elle décrit des **choix stratégiques** qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...)
- Les 5 différentes perspectives (indépendantes et complémentaires) qui permettent de définir un modèle d'architecture sont :



Comment modéliser avec UML ?

3- Démarche orientée architecture

La vue Logique

- Cette vue de haut niveau se concentre sur l'abstraction et l'encapsulation, elle modélise les éléments et mécanismes principaux du système.
- Elle identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments « notions de classes et de relations »
 - les éléments du domaine *sont liés au(x) métier(s) de l'entreprise*,
 - ils sont *indispensables* à la mission du système,
 - ils sont *réutilisables* (ils représentent un savoir-faire)

Comment modéliser avec UML ?

3- Démarche orientée architecture

La vue des Composants

- Cette vue met l'accent sur l'organisation du code en termes de modules, de composants et surtout des concepts du langage ou de l'environnement d'implémentation.
- Dans cette perspective, l'architecte est surtout concerné par les aspects de gestion du code, d'ordre de compilation, de réutilisation, d'intégration et d'autres contraintes de développement pur.

Comment modéliser avec UML ?

3- Démarche orientée architecture

La vue des Composants

- Cette vue de bas niveau montre également :
 - *l'allocation des éléments de modélisation dans des modules (fichiers sources, bibliothèques dynamiques, bases de données, exécutable, etc...).*
 - *les modules qui réalisent (physiquement) les classes de la vue logique.*
 - *l'organisation des composants, leurs interdépendances ...*
 - *les contraintes de développement (bibliothèques externes...)*
 - ...

Comment modéliser avec UML ?

3- Démarche orientée architecture

La vue des Processus

- Cette vue montre :
 - *la décomposition du système en terme de processus (tâches).*
 - *les interactions entre les processus (leur communication).*
 - *la synchronisation et la communication des activités parallèles (threads).*
- Cette vue est très importante dans les environnements multitâches ; elle exprime ainsi la perspective sur les **activités concurrentes** et **parallèles**.

Comment modéliser avec UML ?

3- Démarche orientée architecture

La vue de Déploiement

- Elle exprime la répartition du système à travers un réseau d'ordinateurs (de calculateurs) et de nœuds logiques de traitements.
- Elle est particulièrement utile pour décrire la distribution d'un système réparti.
- Elle met en évidence :
 - *la disposition et la nature physique du matériel utilisé, ainsi que leurs performances.*
 - *l'implantation des modules principaux sur les nœuds du réseau.*
 - *les exigences en terme de performances (temps de réponse, tolérance aux fautes et pannes...).*

Comment modéliser avec UML ?

3- Démarche orientée architecture

La vue des Cas d'utilisation

- Cette vue permet :
 - *de trouver le « bon » modèle*
 - *les cas d'utilisation permettent de guider la modélisation.*
 - *l'utilisation des scénarios et des cas d'utilisation s'avère plus rigoureuse et plus systématique que les entretiens et l'analyse des documents pour découvrir les abstractions du domaine.*
 - *d'expliquer et de justifier ses choix*
 - *Il est en effet nécessaire d'expliquer le système, de justifier les choix qui ont guidé sa conception et son fonctionnement pour pouvoir le construire, le maintenir et le tester.*

CHAPITRE 1C – **RÈGLES GÉNÉRALES**

Modélisation Logicielle

1- Les règles générales

- Mise en place de règles d'écriture et de représentations graphiques normalisées.
- Mise en place des mécanismes et concepts communs applicables à l'ensemble des diagrammes.
- Ceci, dans un souci d'assurer un bon niveau de cohérence et d'homogénéité sur l'ensemble des modèles.
- UML propose, en outre, un méta modèle de tous les concepts et notations associées utilisés dans les treize diagrammes du langage de modélisation

1- Les règles générales

- Les principaux éléments généraux d'UML :
 1. Le stéréotype
 2. La note
 3. La contrainte
 4. Les règles d'écriture des noms et des expressions
 5. Le paquetage
 6. La relation de dépendance

1- Les règles générales

- **Le stéréotype**

- Moyen de classer les éléments de la modélisation.
- Facilité d'élaboration du méta modèle d'UML
- Possibilité d'ajouter à ceux existant dans UML pour la prise en compte des situations particulières propres aux entreprises
- Applicable principalement aux classes
- Possibilité d'identification d'une typologie de classes

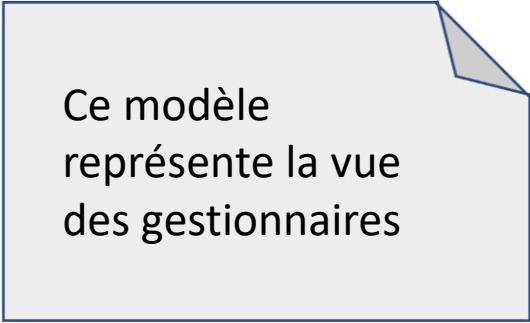
1- Les règles générales

- Le **stéréotype** (formalisme)
 - Le nom du stéréotype est indiqué entre **guillemets**.
 - Un acteur peut être vu comme un stéréotype particulier d'une classe appelée Client.

Client
« *acteur* »

1- Les règles générales

- **La note**
 - *Commentaire explicatif d'un élément d'UML.*

A light gray rectangular box with a blue border and a folded top-right corner, representing a UML note.

Ce modèle
représente la vue
des gestionnaires

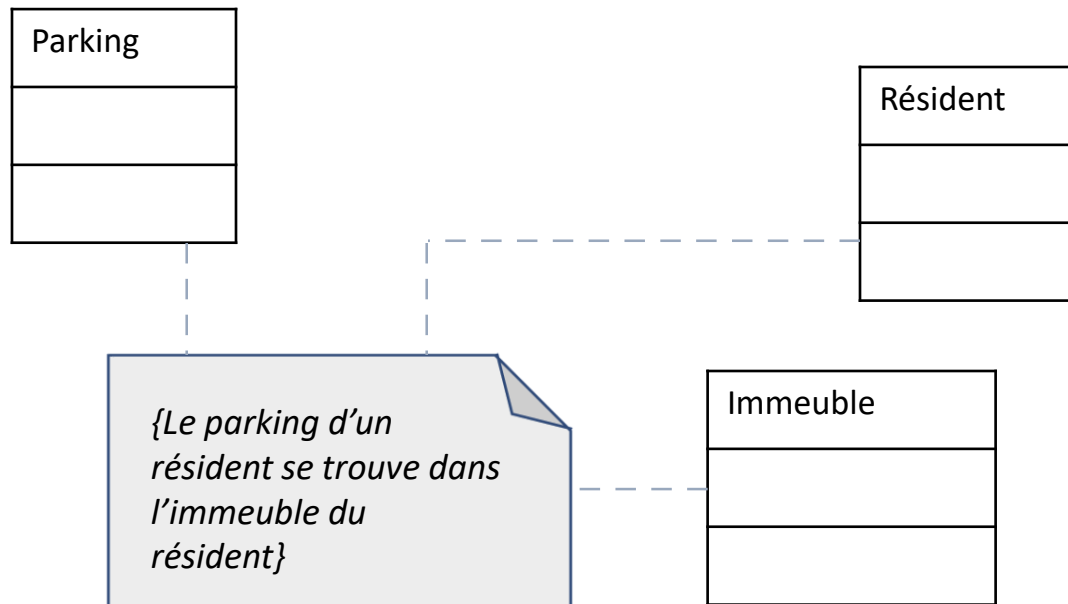
1- Les règles générales

- **La contrainte**

- Note ayant une valeur sémantique particulière pour un élément de la modélisation.
- Présentation : entre accolades {}
- Dans le cas où la contrainte concerne deux classes ou plus, celle-ci s'inscrit à l'intérieur d'une note.

1- Les règles générales

- La **contrainte** (formalisme)
 - **Première** forme d'écriture : *{ceci est une contrainte}*
 - **Deuxième** forme d'écriture : à l'intérieur d'une note



1- Les règles générales

- **Le nom**
 - Simple ou composé.
 - **Simple** : représenté par une chaîne de caractères
 - **Composé** : formé d'un nom simple suivi d'un point et du complément de dénomination propre au nom composé

1- Les règles générales

- Le **nom** (formalisme)

NomChambre

ou

NomHotel.NomChambre

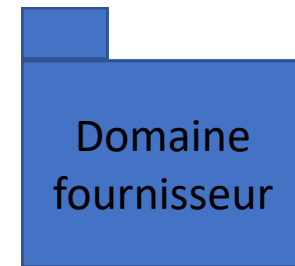
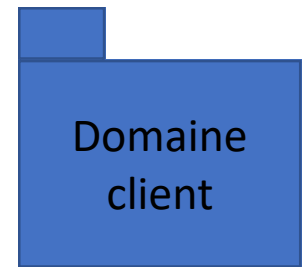
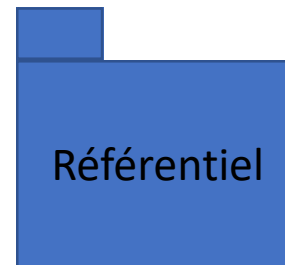
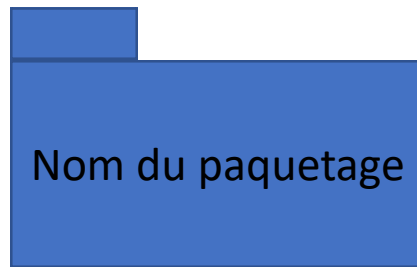
1- Les règles générales

- **Le paquetage**

- Regroupement des éléments de modélisation (les classes par exemple) portant sur un sous-ensemble du système dans le but de former un ensemble cohérent.
- Traduction d'un découpage logique du système
- Contient la plupart des éléments UML : classes, objets, cas d'utilisations, etc.
- Contient également des paquetages, créant une hiérarchie complète.

1- Les règles générales

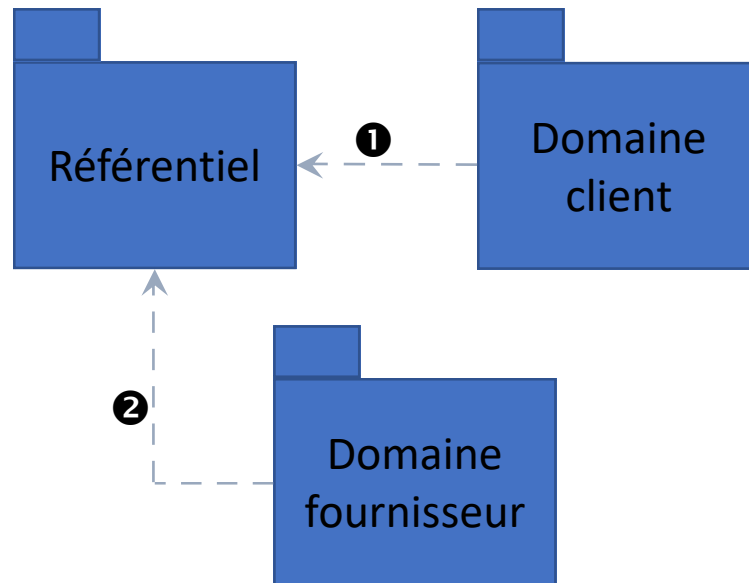
- Le **paquetage** (formalisme)



Système de l'entreprise

1- Les règles générales

- Le **paquetage** (relation de dépendance)



- ❶ Dans le domaine client il est fait appel à des éléments du Référentiel
- ❷ Dans le domaine fournisseur il est fait appel au Référentiel

Relation de dépendance

Lien de dépendance entre deux éléments de la modélisation représenté par un trait en pointillé entre les deux éléments.

CHAPITRE 1D – **LES DIAGRAMMES UML**

Modélisation Logicielle

LES DIAGRAMMES D'UML

13 diagrammes regroupés dans deux grands ensembles.

- Les diagrammes **structurels** : Ces diagrammes, au nombre de six, représentent l'aspect statique d'un système (classes, objets, composants, ...).
- Les diagrammes de **comportement** : Ces diagrammes représentent la partie dynamique d'un système réagissant aux événements et permettant de produire les résultats attendus par les utilisateurs.

■ LES DIAGRAMMES STRUCTURELS

Diagramme de classes (DCL)	●	Représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements.
Diagramme d'objets (DOB)	●	Représente les instances des classes et des liens entre instances.
Diagramme de composants (DCP)	●	Représente les différents constituants du logiciel au niveau de l'implémentation d'un Système.
Diagramme de déploiements (DPL)	●	Décrit l'architecture technique d'un système avec une vue centrée sur la répartition des composants dans la configuration d'exploitation.
Diagramme de paquetages (DPA)	●	Donne une vue d'ensemble du système structuré en paquetage. Chaque paquetage représente un ensemble homogène d'éléments du système (classes, composants...).
Diagramme de structure composites (DSC)	●	Décrit la structure interne d'un ensemble complexe composé par exemple de classes ou d'objets et de composants techniques.

■ LES DIAGRAMMES DE COMPORTEMENT

Diagramme des cas d'utilisations (DCU)	<ul style="list-style-type: none">● Ce diagramme est destiné à représenter les besoins des utilisateurs par rapport au système.
Diagramme d'état-transitions (DET)	<ul style="list-style-type: none">● Représente les différents états des objets en réaction aux événements.
Diagramme d'activités (DAC)	<ul style="list-style-type: none">● Représente une vision des enchaînements des activités propres à une opération ou à un cas d'utilisation.
Diagramme de séquences (DSE)	<ul style="list-style-type: none">● 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 de communications (DCO)	<ul style="list-style-type: none">● Ce diagramme est une autre représentation des scénarios des cas d'utilisation qui met plus l'accent sur les objets et les messages échangés.
Diagramme global d'interactions (DGI)	<ul style="list-style-type: none">● Ce diagramme fournit une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activités.
Diagramme de temps (DTP)	<ul style="list-style-type: none">● Ce diagramme permet de représenter les états et les interactions d'objets dans un contexte où le temps a une forte influence sur le comportement du système à gérer.

Outils

- L'Atelier de Génie Logiciel « **Modelio 3.8** » sera utilisé durant ce cours pour :
 - ✓ La construction des différents diagrammes

- Il existe plusieurs autres outils dans le domaine :
 - ✓ StartUML
 - ✓ ArgoUML
 - ✓ Rational Rose
 - ✓ Enterprise Architect

FIN