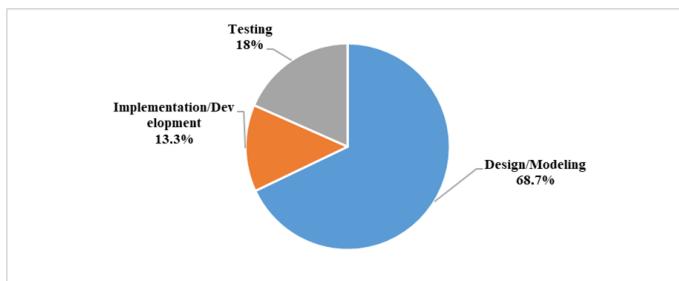


L'UML dans les grandes lignes



L'UML (Unified Modeling Language) est un langage de modélisation visuelle utilisé principalement dans le domaine du génie logiciel pour représenter les systèmes complexes. Voici les grandes lignes de l'UML :

1. Objectifs de l'UML



- **Standardisation** : Fournir une notation standardisée pour la modélisation des systèmes logiciels.
- **Visualisation** : Aider à visualiser la structure et le comportement d'un système.
- **Documentation** : Faciliter la documentation des architectures et des processus.
- **Spécification** : Définir clairement les besoins et les spécifications des systèmes.
- **Conception** : Aider à la conception et à l'analyse des systèmes avant leur développement.

2. Diagrammes UML

Classification	Types	Features
Structure Diagrams	Class Diagram	Structure of each class; relationships between classes
	Component Diagram	Components that make up the software and the dependencies between them
	Deployment Diagram	Physical layout of the system
	Package Diagram	Grouping of model elements such as classes and relationships between groups (packages)
Behavioral Diagrams	Use Case Diagram	Functions provided by the system, and relationships with external users and other systems
	Sequence Diagram	Interaction of objects along the time axis
	Collaboration Diagram	Objects interacting to implement some behavior within a context
	Statechart Diagram	Model life time of an object from creation to termination
	Activity Diagram	System operation flow

L'UML se compose de différents types de diagrammes, chacun ayant un objectif spécifique :

a. Diagrammes structurels

Ces diagrammes décrivent la structure statique du système.

- **Diagramme de classes** : Montre les classes du système, leurs attributs, méthodes et relations.
- **Diagramme d'objets** : Une instance du diagramme de classes montrant un exemple concret de classes et objets.
- **Diagramme de composants** : Montre l'organisation et les relations des composants physiques du système.
- **Diagramme de déploiement** : Représente la configuration des éléments matériels et logiciels au moment de l'exécution.
- **Diagramme de paquets** : Montre l'organisation et la hiérarchie des packages.

b. Diagrammes comportementaux

Ces diagrammes décrivent le comportement dynamique du système.

- **Diagramme de cas d'utilisation** : Décrit les interactions entre les utilisateurs (acteurs) et le système pour accomplir un but.
- **Diagramme de séquence** : Montre les interactions entre les objets dans le temps.

- **Diagramme de communication** : Illustre les interactions entre objets ou composants en termes de messages échangés.
- **Diagramme d'activités** : Décrit les flux de travail ou d'activités dans un processus.
- **Diagramme d'états** : Montre les états d'un objet et les transitions entre ces états.
- **Diagramme de synchronisation** : Combine des éléments des diagrammes de séquence et de communication pour montrer l'interaction et la collaboration.

3. Principes de base de la modélisation UML

Principles

- **Modularity:**
 - alleviate complexity of large-scale systems
- **Abstraction:**
 - separating the essential from the non-essential characteristics of an entity
- **Encapsulation:**
 - Information hiding
- **Polymorphism:**
 - Portability
- **Levels of Abstraction:**
 - Organization of classes and interfaces

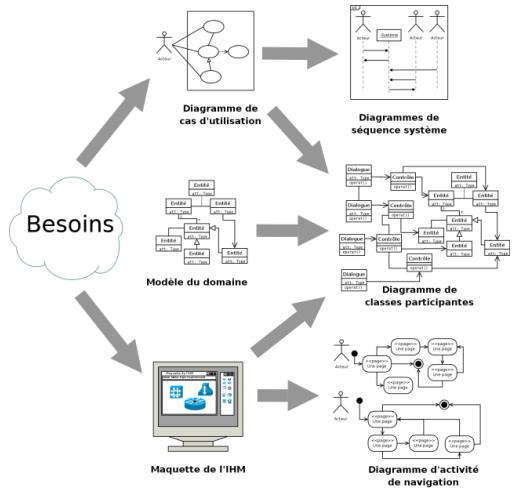
CS 3300 Object-Oriented Concepts

Object-Oriented Modeling Using UML

32

- **Abstraction** : Réduction de la complexité en ne montrant que les aspects essentiels.
- **Encapsulation** : Dissimuler les détails internes des classes.
- **Modularité** : Décomposition du système en sous-systèmes ou modules indépendants.
- **Hiérarchie** : Utilisation d'héritage pour organiser les classes et objets en une hiérarchie.

4. Outils et méthodologies

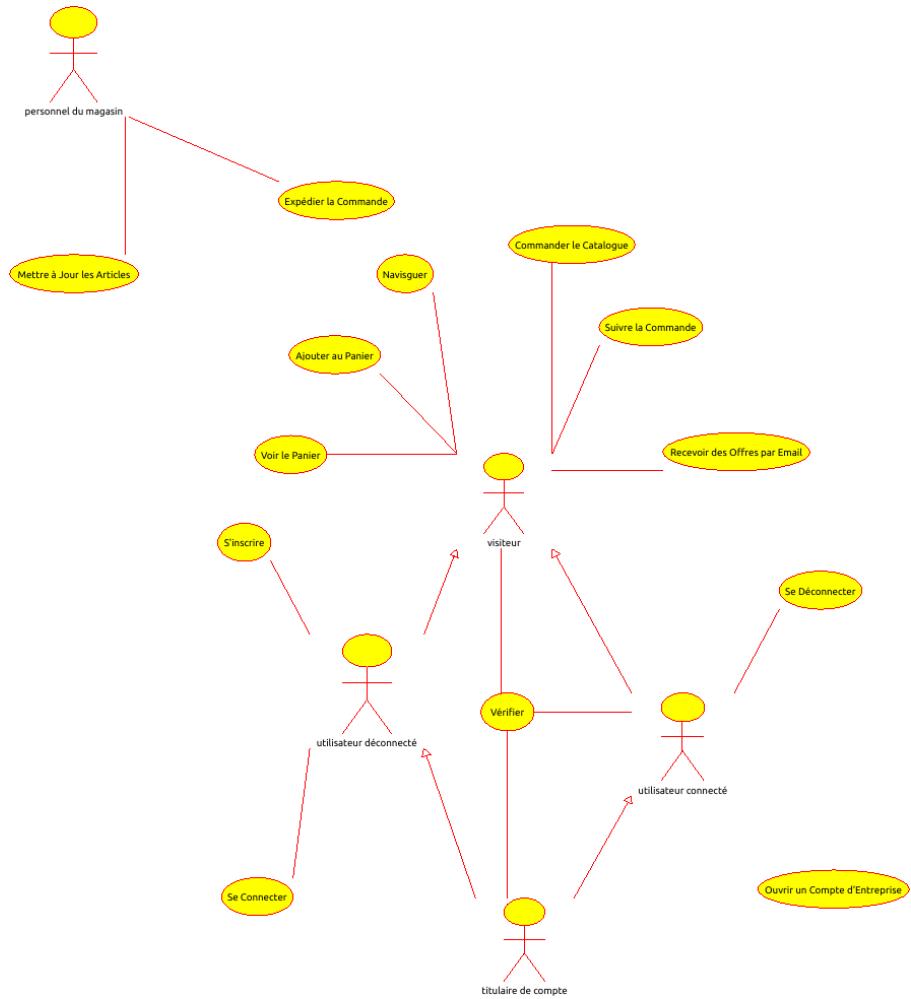


L'UML est souvent utilisé avec des outils de modélisation comme Rational Rose, Enterprise Architect, ou Visual Paradigm. Il s'intègre bien dans des méthodologies de développement comme Agile, Scrum, ou le modèle en V.

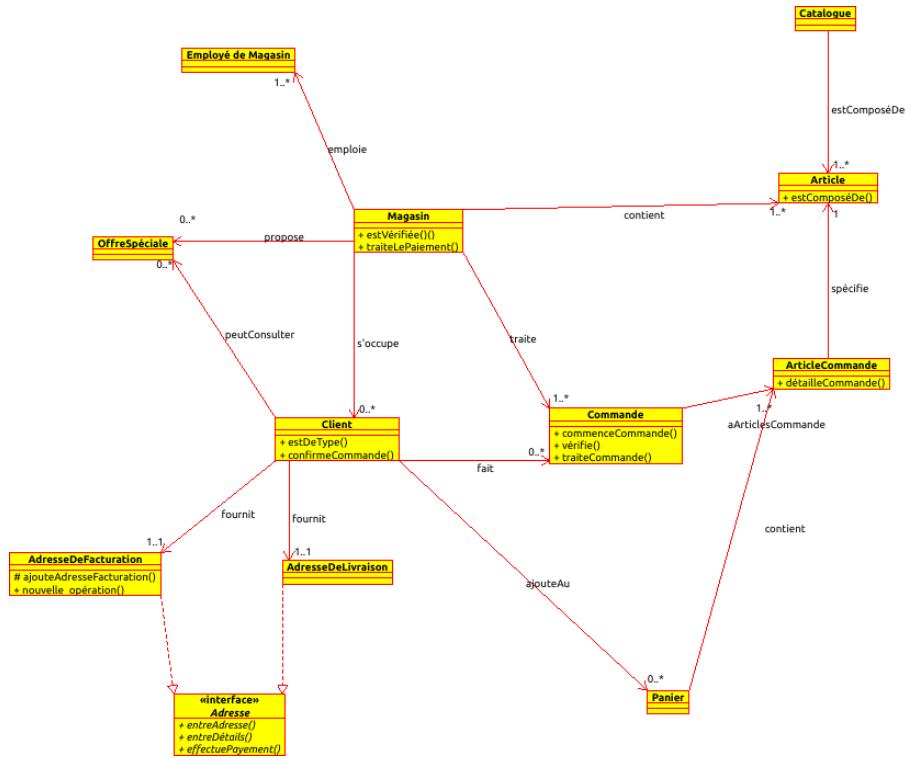
4. Exemple concret de Lab

Parce qu'il sagit d'un logiciel libre, bien qu'imparfait, nous utiliserons Umrello. Premier aperçu d'un Lab en UML:

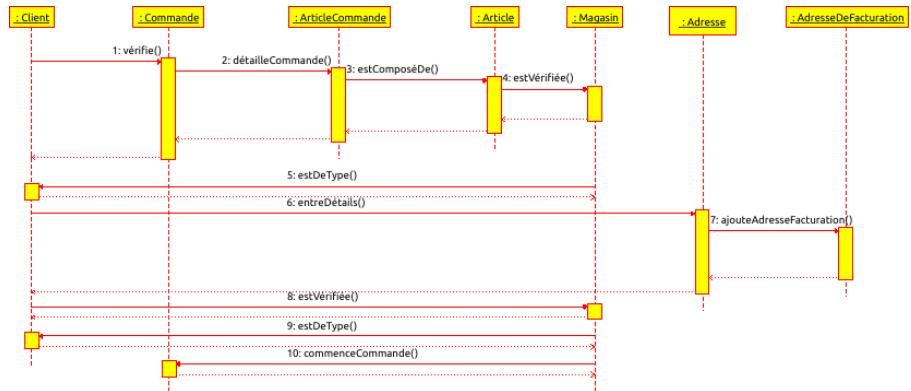
1ère étape: Le Diagramme de Cas Utilisateur



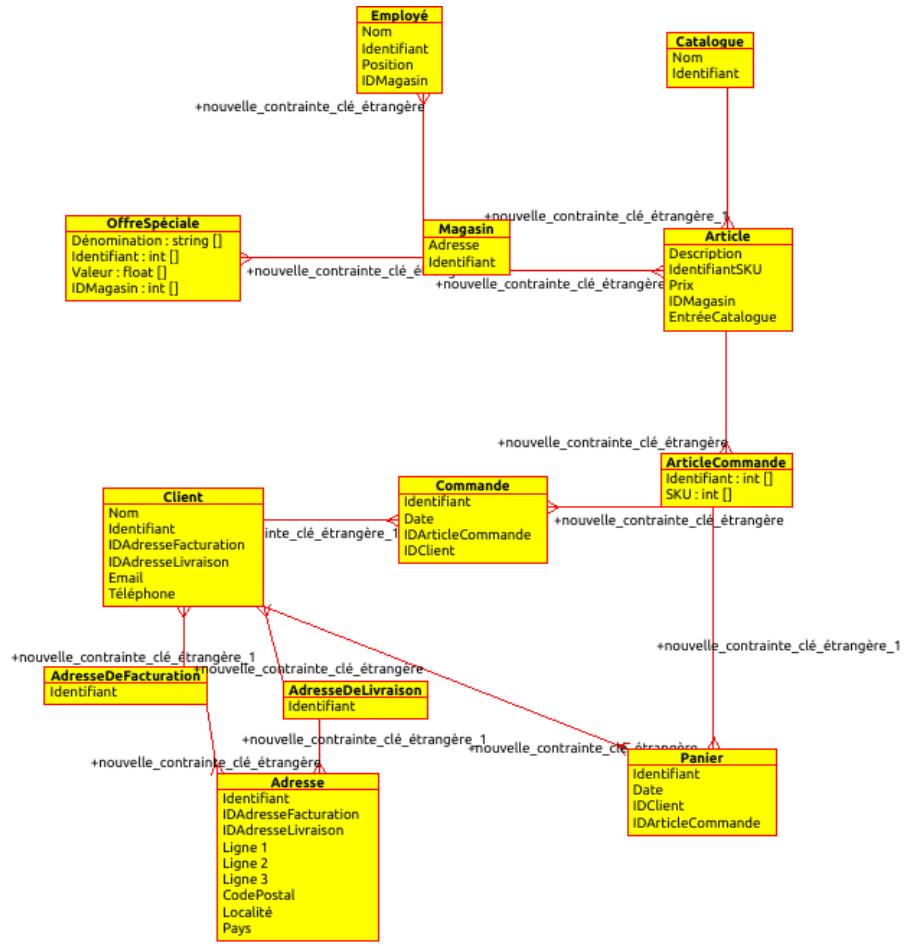
2ème étape: Le Diagramme de Classe



3ème étape: Le diagramme de Séquence



4ème étape: Le diagramme de Relation d'Entité



Nous tirerons de ces diagrammes une première mouture de génération code source et nous abordons la démarche inverse qui produit des diagrammes à partir d'un code source (rétro-conception).

6. Avantages de l'UML

Avantages et Limites d 'UML

Avantages d'UML	Limites d'UML
Prise en compte des besoins directs des utilisateurs (diag. UC)	Absence de démarche
Norme internationale de modélisation (communication facilitée)	Absence de modèles pour représenter l'IHM
Large palette de modèles	Trop de concepts proposés (complexité du langage)
Personnalisation possible de la modélisation (stéréotype)	Ambiguité possible de certains modèles (utilisation à différents niveaux d'abstraction)

111

NONNE Laurent

- **Communication** : Facilite la communication entre les parties prenantes.
- **Clarté** : Fournit une vue claire et compréhensible des systèmes complexes.
- **Réutilisabilité** : Encourage la réutilisation des composants grâce à une meilleure compréhension du système.
- **Analyse et conception** : Aide à identifier les problèmes potentiels tôt dans le cycle de vie du développement.

Conclusion

En résumé, l'UML est un outil puissant pour la modélisation, la documentation, et la conception des systèmes logiciels, offrant un ensemble riche et standardisé de notations pour représenter les différentes facettes d'un système.