Modélisation des Systèmes d ’information

Table des matières

I Modélisation des Systèmes d'information

1. UML : Unified Modeling Language

UML (Unified Modeling Language) est un langage de modélisation normalisé permettant de représenter graphiquement un système logiciel ou informationnel, avant sa mise en œuvre.

Norme : ISO/IEC 19505-1 et 19505-2 (OMG UML 2.5):

Créé dans les années 1990 par **Grady Booch, James Rumbaugh et Ivar Jacobson** (méthodes unifiées de modélisation).

Objectifs principaux :

- Faciliter la compréhension de systèmes complexes
- Améliorer la communication entre utilisateurs, concepteurs et développeurs
- Servir de base à la conception et à la génération de code2^2,3^3, 10^{10}

1.1. Diagrammes UML

UML peut être utilisé pour définir de nombreux modèles :

Modèles statiques :

- Décrivent la structure du système
- Classes, objets, composants, paquetages

Modèles dynamiques :

- Décrivent le comportement et les interactions
- Séquences, états, activité 1^1, 10^{10}

a) Diagrammes UML

Diagrammes structurels (7) :

- Diagramme de classes
- Diagramme d'objets
- Diagramme de paquetages
- Diagramme de composants
- Diagramme de déploiement
- Diagramme de structure composite
- Diagramme de profil

Diagrammes comportementaux (7) :

- Diagramme de cas d'utilisation
- Diagramme d'activités
- Diagramme de machines à états
- Diagramme de séquence

- Diagramme de communication
- Diagramme de vue d'ensemble d'interaction
- Diagramme de temps (timing)
- ... et un langage pour exprimer des contraintes : OCL*1*[1] , *10*[10]

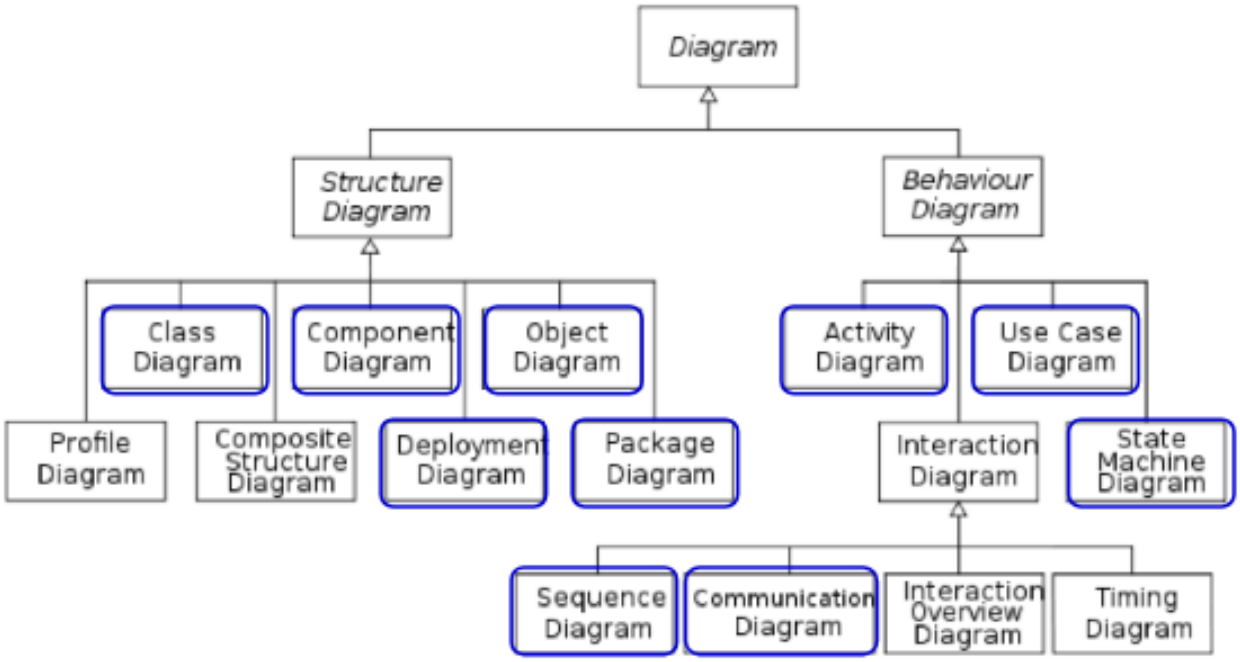

Image 1 Diagrammes UML

i) Modélisation de la structure du système: point de vue statique

Modéliser la structure avec UML:}Point de vue statique sur le système

- Décrire la structure du système en termes de :
- Composants du système: Objets, Classes, Paquetages, Composants, ...;
- Relations entre ces composants: Spécialisation, Association, Dépendance, ...;
- Pas de facteur temps

1 Modéliser le comportement avec UML

Modéliser le comportement avec UML

Décrire le comportement du système en termes d'interactions:

Système vu comme une boite noire :

- Qui interagit avec le système, et dans quel but?
- ---> Diagramme de cas d'utilisation:
- Comment interagit le système avec son environnement?
- ---->Diagramme de séquence système

Système vu comme une boite blanche :

- Comment interagissent les objets du système ?
- ---->Diagramme de séquence et/ou de communication

Décrire l'évolution du système dans le temps :

- Comment évoluent les états des objets ?
- ----->Diagrammes d'états transitions

1 Développer un modèle architectural

Décrire l'architecture avec UML

– Tous les diagrammes UML peuvent être utiles pour décrire les différents aspects du modèle architectural

– Trois des diagrammes UML sont particulièrement utile pour décrire une architecture logicielle

• Diagramme de packages;

• Diagramme de composants;

• Diagramme de déploiement

Les diagrammes de paquetages, composants et de déploiement permettent de générer la structure de l'application et son architecture. Le diagramme de classes permet la génération du code métier. Les outils UML comme Modelio et Enterprise Architect assurent une traçabilité complète entre conception et implémentation. »

1 Diagramme de paquetages (package diagram)

Modéliser la structure avec UML Diagrammes de paquetage

Diagrammes de paquetages.

Qu'est-ce qu'un paquetage (package) ?

C'est un élément de modélisation qui :

- Contient d'autres éléments de modélisation (classes, autres paquetages, ...);
- Possibilité de ne pas représenter tous les éléments contenus
- Définit un espace de nom (namespace)

Objectif :

- Organiser logiquement le code en sous-modules (MVC, couches métier / données / présentation).
- Relier à la structure du projet : controller, model, view, auth, dashboard, gestion utilisateurs, ... etc.

Dépendance : c.-à.-d.: un paquetage est dépendant d'un autre s'il l'utilise… *1*[1]

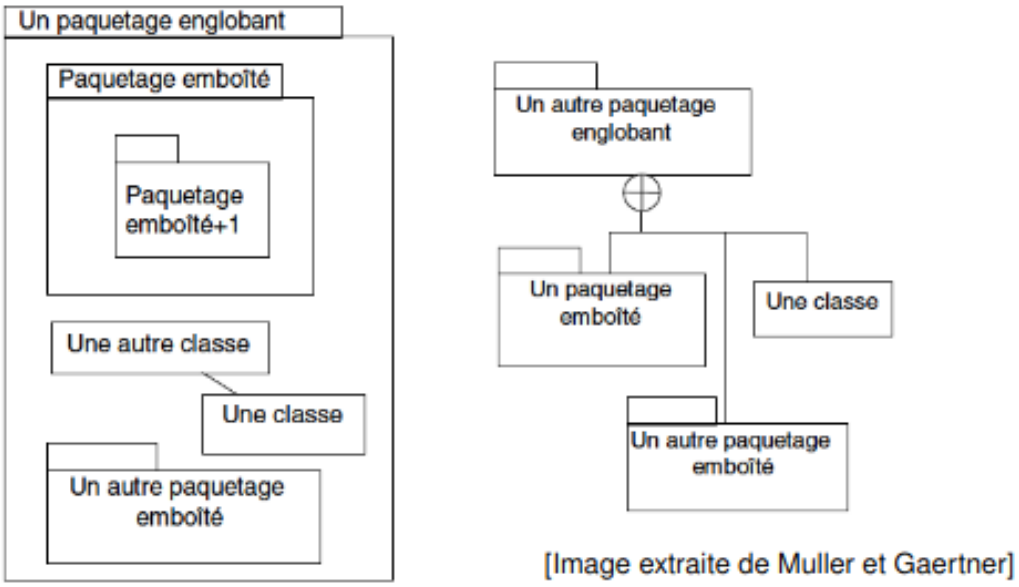

Image 2 Diagramme de paquetage

1 Diagramme de composants

Il montre les dépendances entre les parties compilées ou modules du système.

- Permet de modéliser le déploiement du système sur une architecture physique
- Disposition des artefacts sur les nœuds physiques

Artefacts : instances de composants, processus, ...

Nœuds physiques : Ordinateur, Téléphone, Imprimante, ...

Moyens de communication entre les nœuds *1*[1]

Exemple **:**

- Composant Web Front-End (React)
- Composant API (Node.js ou Django)
- Composant Base de données (MySQL, MongoDB)

1 Diagramme de déploiement

Montre où et comment les composants sont installés :

- Serveur Web
- Serveur BD
- Navigateur client
- Réseau interne / cloud *1*[1]

2. Génération de Code: Code generation

Définition **: Génération de Code**

La génération de code est une opération qui consiste à créer automatiquement du code source à partir d'une spécification de haut niveau, souvent un modèle formel ou un diagramme, afin de produire une implémentation exécutable, réduire les erreurs humaines et améliorer la productivité du développement logiciel.

Autrement dit : au lieu d'écrire manuellement les classes, méthodes et structures, un outil de génération de code produit automatiquement ces éléments à partir d'un modèle abstrait (ex. UML, PIM/MDA).

3. L'Ingénierie dirigée par les Modèles: IDM

Définition **: Ingénierie Dirigé par les Modèles (IDM) ou en Anglais: Model-Driven Engineering:**

le processus automatisé de transformation d'un modèle logiciel (comme UML) en code source dans un langage de programmation cible, tout en maintenant la cohérence entre le modèle et le code pour réduire les erreurs et faciliter la maintenance. *9*[9]

Principe: générer le code à partir des modèles UML (modèles CIM → PIM → PSM).

La transformation se fait généralement d'un modèle abstrait (par exemple, UML ou PIM) vers du code concret (Java, C++, C#, etc.).

Outils :

- Eclipse Modeling Framework (EMF)
- Acceleo (basé sur MOFM2T)
- Papyrus (modeleur UML open source)

- Langages cibles : Java EE, Spring Boot, Django, Laravel, etc.

3.1. Avantages et Inconvénients de IDM

a) Avantages

- Réduire les erreurs de programmation
- Améliorer la cohérence entre modèle et implémentation
- Accélérer le développement logiciel
- Faciliter la maintenance du logiciel automatisation forte, maintenabilité, traçabilité.

i) Limites

- Problème d'apprentissage: il faut former les ingénieurs des lanagegs et outils (apprentissage).
- Changements fréquents: Mise à jour continue des modèles dépendance à des outils spécifiques.
- Limites des outils: Support partiel de langages et plateformes
- Difficile de modéliser performance, sécurité…: non-fonctionnels

3.2. B. Approche Basée sur les patrons de conception (Design Patterns)

Principe : relier directement les diagrammes UML à des patrons de conception connus (MVC, DAO, Factory, Observer…).

Utilisation typique :

Diagramme de classes → classes concrètes (avec héritage, encapsulation)

Diagramme de séquence → logique métier (contrôleurs)

Diagramme de déploiement → architecture web (serveur, client, BD)

Exemple d'utilisation :

UML + MVC → code Spring, Django, Flask, etc.

Outils : Visual Paradigm, StarUML, GenMyModel

Avantages : très concret pour les étudiants, compréhension claire du lien entre design et code.

Limites : génération partielle, pas de transformation automatique complète.

3.3. C. Approche Agile / itérative}

Principe : UML est utilisé pour prototyper et documenter des itérations rapides.

Modèles UML légers (souvent cas d'utilisation + classes + séquences)

Génération manuelle assistée : on part d'un modèle UML pour structurer le code, mais la génération est semi-automatique.

Outils : StarUML, PlantUML, Lucidchart ...etc.

Avantages : flexibilité, bon équilibre entre théorie et pratique.

Limites : pas de génération automatique intégrale.

3.4. D. Approche Low-code / Model-to-application

Principe : l'UML ou le modèle conceptuel alimente une plateforme low-code ou no-code (ex. Mendix, OutSystems, WebRatio).

But : générer automatiquement une application web fonctionnelle à partir d'un modèle.

Avantages : rapide, visuel, utilisable sans beaucoup de programmation.

Limites : dépendance à la plateforme, code parfois fermé.

4. Outils de l'ingénierie de modèles

4.1. Modelio

Modelio :(développé par Modeliosoft) est un outil modulaire, open source, compatible UML2 / BPMN2, et extensible via modules :

Java Designer → génération de code Java à partir du modèle UML.

SQL Designer → génération de script SQL à partir du modèle conceptuel.

Web Modeler / Analyst → conception orientée métier (SI, processus).

- MDA Designer → transformation de modèles (PIM → PSM). [4]

a) Approche MDE avec Modelio : génération de code web

Étape 1 —: **Créer le modèle PIM (Platform Independent Model)**

Dans Modelio :

Crée un diagramme de cas d'utilisation pour modéliser les fonctionnalités de l'application web.

Exemple : Gérer les utilisateurs, Publier un article, Consulter les statistiques.

Ajoute un diagramme de classes pour les entités principales (ex. Utilisateur, Article, Commentaire).

Crée un diagramme de séquence pour décrire les interactions utilisateur / système (utile pour le futur contrôleur MVC).

Étape 2 — Transformer vers un modèle PSM (Platform Specific Model)

Activer le module Java Designer (via Configuration → Modules → Install a module).

Sélectionner les classes à transformer → clic droit → Java → Generate Java Code.

Cela crée une arborescence de classes dans un projet Java

Il faut relier ce code à un framework web (par exemple, Spring Boot pour Java ou Laravel pour PHP).

⚙ Le module Java Designer génère la structure objet (modèle métier), ensuite on complète avec le code web (contrôleurs, vues).

Étape 3 —Génération ou intégration web

□ *Exemple* : "Gestion des articles avec Modelio

créer une petite application web "Gestion des articles".

Modèle UML :

Classes :

Article (id, titre, contenu)

Utilisateur (id, nom, email)

Commentaire (id, texte, date)

Relations :

Utilisateur 1.. →Article

Article 1.. →Commentaire

4.2. B. Eclipse Modeling Framework (EMF) + Acceleo

C'est la référence académique et industrielle du MDE complet.

Avantage :

- Approche très formelle, soutenue par l'OMG et l'Eclipse Foundation.

Permet de montrer le cycle complet MDA :

UML → Ecore → Java → Web app déployée. *5*[5]

Étape	Outil
CIM/PIM	Papyrus (UML)
Transformation M2M	ATL (Atlas Transformation Language)
Transformation M2T	Acceleo (Model-to-Text)
Génération du code	Acceleo templates (Java, PHP, etc.)
Déploiement	Maven / Gradle intégrés à Eclipse

Image 3 L'ingénierie de Modèles avec Acceleo

4.3. C. WebRatio (Low-code MDE orienté Web)

C'est une approche 100 pour cent MDE orientée application Web et BPMN, on peut:

- Modèliser le processus métier (CIM/PIM).
- Définir le modèle de navigation et contenu UML.
- Le système génère automatiquement une application web complète (J2EE ou Spring).*7*[7]

Étape	WebRatio
CIM/PIM	BPMN + UML
PSM	Mapping vers Java EE
Code	Génération automatique
Déploiement	Exécutable web complet

Image 4 MDE avec WebRatio

4.4. D. Integranova MDA Platform

Une solution industrielle utilisée dans certains projets académiques.

- UML → Java EE / .NET / SQL
- Gestion complète du cycle (modélisation, génération, test, déploiement)
- Supporte MDA, OCL, transformations ATL. *6*[6]

4.5. E. JetBrains MPS (Meta Programming System)

Approche plus moderne : création de langages spécifiques au domaine (DSL) dans un cadre MDE complet.

- Modéliser PIM
- Transformer en PSM (DSL)
- Générer du code
- Déployer via Gradle/Maven *5*[5]

5. Reverse-Engineernig: Rétro-ingénierie

□ *Définition* :

"The process of analyzing a software system to identify the system's components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction."

Traduction : Rétro-ingénierie « Le processus consistant à analyser un système logiciel pour identifier ses composants et leurs interrelations, et pour créer des représentations du système sous une autre forme ou à un niveau d'abstraction supérieur. »

2. IEEE Standard 1063 / 1044 :

IEEE Std 1063-2001 (Software Documentation) : Reverse Engineering est défini comme : "The process of analyzing a system to identify the system's components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction."

3. IEEE Std 1471 / ISO/IEC/IEEE 42010 (Architecture logicielle) :

"Reverse engineering is used to understand, re-document, or improve an existing system."

Traduction : « L'ingénierie inversée est utilisée pour comprendre, redocumenter ou améliorer un système existant. »

□ *Définition* : Définition générale: Reverse Engineering: Rétro-ingénierie

C'est le processus consistant à analyser un système existant (logiciel, application, matériel ou base de données) pour en extraire des modèles abstraits, la structure, le comportement et les fonctionnalités, sans disposer des spécifications ou du code source original de manière détaillée.

On utilise souvent le reverse engineering pour :

- Extraire un modèle UML à partir du code existant (classes, séquences, composants).
- Analyser la structure d'un système pour la maintenir ou l'améliorer.
- Migrer un ancien système vers une nouvelle technologie (ex. refonte d'un SI Web).
- Documenter une application lorsque la documentation originale est absente ou obsolète.

□ *Exemple* : Exemple

On prend un projet Java ou une application Web existante.

Avec un outil comme Modelio, Visual Paradigm ou Enterprise Architect, on génère des diagrammes UML : diagramme de classes, diagramme de séquence, diagramme de composants.

Ces diagrammes permettent aux développeurs ou analystes de comprendre le fonctionnement et planifier des modifications.

5.1. Outils Reverse-Engineering: Rétro-ingénierie

- Modelio: revser-engineering: JAVA
- Visual Paradigm: revsere-engineering JAVA, c, c#, , C++, PHP
- StarUML: revsere-engineering JAVA, C++
- Enterprise Architect: reverse-engineering multi-langages
- Eclipse Papyrus + EMF
- ObjectAid UML Explorer (Eclipse)

6. Application web simple de gestion d'utilisateurs

Objectifs pédagogiques

Montrer que :

Le diagramme de classes UML décrit les objets du domaine (User, Role).

Le code source peut être généré automatiquement à partir de ce modèle.

Le code généré correspond à la structure du futur site ou API web.

6.1. Étapes dans Modelio

a) Créer le modèle

Nouveau projet : CodeGen_WebUML

- Créer les classes User et Role dans le diagramme UML.
- Ajouter les attributs et les opérations.

b) Activer la génération de code

Menu :Java Designer → Generate Java code

Choisir un dossier de sortie (/generated_code)

Modelio crée automatiquement :

User.java

Role.java

Avec getters/setters, attributs, et structures de base.

```java
package com.webapp.model;

import java.util.List;

public class User {
    private int id;
    private String username;
    private String email;
    private List<Role> roles;

    public void login() {
        System.out.println(username + " logged in.");
    }

    public void logout() {
        System.out.println(username + " logged out.");
    }

    // Getters \& Setters générés
}
```