

Génie logiciel

Notes du cours de 14/10 , partie 1

L3 Informatique appliquée 2022-2023

MABROUK Fayez

8 novembre 2022

1 Initiation à UML

1.1 Introduction à UML

- * Langage de modélisation graphique.
- * Objectifs :
 - * Fournir une description d'un logiciel.
 - * Permettre la visualisation des différents aspects d'un logiciel.
 - * Analyse du logiciel.
 - * Permettre la communication à l'intérieur et à l'extérieur d'un projet, avec des personnes techniques et non techniques.
 - * Vérification de l'exhaustivité, de la cohérence et de l'exactitude.
- * Langage de modélisation à usage général.
 - * Indépendant du processus.
 - * Peut être utilisé pour représenter des informations sur **la structure, le comportement ou l'interaction**.

1.2 Vues

- * Un modèle est composé de plusieurs vues.
- * Une vue décrit un système sous différents angles.
- * Exemple de vues :
 - * Vue structurelle : donne des informations sur la structure du modèle.
 - * Vue comportementale : donne des informations sur le comportement du modèle.
 - * Vue interactionnelle : donne des informations sur la manière dont les différentes parties du modèle se comportent **les unes par rapport aux autres**.

1.3 Types de diagrammes

- * UML définit 13 diagrammes en 3 catégories qui permettent de définir un système selon selon différents points de vue.
- * Diagrammes de structure :
 - * Diagramme de classes, diagramme d'objets, diagramme de composants, diagramme de structure composite, Diagramme de paquetage et Diagramme de déploiement
- * Diagrammes de comportement :
 - * **Diagramme de cas d'utilisation** ,Diagramme d'activité et diagramme de machine à états
- * Diagrammes d'interaction :
 - * **Diagramme de séquence, diagramme de communication**, diagramme de synchronisation et aperçu des interactions Diagramme.

1.4 Rappels sur l'approche par objet

- * UML est basé sur une approche par objet.
- * Définition d'un **objet** : Un objet est une entité référencée par un identifiant. Il est souvent tangible.
- * Un objet possède un ensemble d'attributs (structure) et de méthodes (comportement).
- * Définition d'une **classe** : ensemble d'objets similaires (c'est-à-dire ayant les mêmes attributs et les mêmes méthodes).
- * Un objet d'une classe est une instance de cette classe.
- * Définition de l'**abstraction** : principe de sélection des propriétés pertinentes d'un objet pour un problème donné.
- * Aspect important d'UML : l'objet réel est simplifié par son abstraction pour ne garder que ce qui est pertinent pour le modèle.
- * Définition de l'**encapsulation** : cacher certains attributs ou méthodes à d'autres objets. Notez qu'il s'agit d'une abstraction.
- * **Spécialisation** : une nouvelle classe A peut être créée comme sous-classe d'une autre classe B, dans ce cas, la classe A spécialise la classe B.
- * La **généralisation** est l'inverse (la superclasse B est une généralisation de la sous-classe A).
- * **Héritage** : le fait qu'une sous-classe obtienne le comportement et la structure de la super-classe.
- * C'est une **conséquence** de la spécialisation.
- * Classes **abstraites** et **concrètes** : les classes abstraites sont des classes qui n'ont pas de d'instances (par exemple, Mammal). Les classes concrètes en ont (par exemple, Human).
- * Les classes **abstraites** permettent de hiérarchiser les classes et de regrouper les attributs et les méthodes. Elles doivent avoir des sous-classes.
- * **Polymorphisme** : le comportement des objets d'une même classe (en général abstraite) peut être différent car il s'agit d'instances de classes différentes. être différent car ils sont des instances de différentes sous-classes.
- * **Composition** : les objets complexes peuvent être composés d'autres objets.
- * Elle est définie au niveau de la classe, mais nous ne composons que des instances réelles.
- * Elle peut être :
 - * une relation forte : les composants ne peuvent pas être partagés ; la destruction de l'objet composé implique la destruction des composants.
 - * une relation faible (aussi appelée agrégation) : les composants peuvent être partagés.

1.5 Qu'est-ce qu'un modèle logiciel ?

- * Formalisé sous forme de document.
- * Pas seulement des diagrammes !

- * Le document doit indiquer :
 1. Les informations pratiques (auteurs, date, version).
 2. Le contexte du projet.
 3. Introduction au modèle (choix, quelles vues, discussion).
 4. Les diagrammes, centrés sur les cas d'utilisation.

1.6 Conclusion

- * UML permet de modéliser les logiciels.
- * UML est un standard :
 - * Des gens y ont pensé.
 - * Permet une bonne communication.
 - * Communauté forte.
 - * Evolution.
 - * Ne disparaîtra pas demain.
- * UML est difficile à maîtriser (et nous ne le maîtriserons pas dans ce cours).