Modélisation Orientée Objet et UML

Fondamentaux et approches

Formation POO + BD

Veille 1

Introduction

Objectifs de la présentation

- Comprendre les fondamentaux de la modélisation dans le développement logiciel
- Distinguer l'analyse fonctionnelle de l'analyse orientée objet
- Maîtriser les concepts clés d'UML et ses différentes vues
- Explorer les principaux diagrammes UML (contexte, cas d'usage, classes)

Importance de la modélisation

- Facilite la communication entre les parties prenantes
- Permet de visualiser et comprendre des systèmes complexes
- Réduit les risques et les coûts de développement
- Sert de documentation et de référence tout au long du cycle de vie du logiciel

Modélisation : Notions d'analyse et conception

Analyse

Étude approfondie du problème et des besoins pour comprendre ce qui doit être construit.

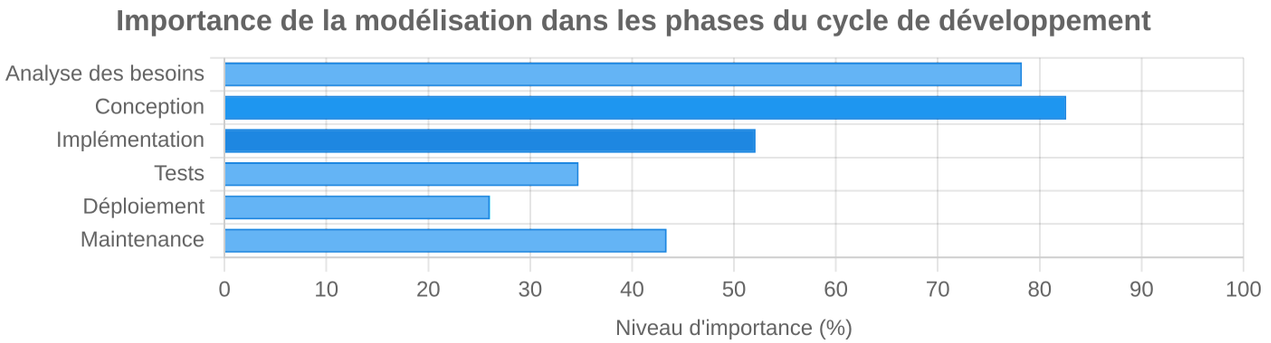
- Se concentre sur le "quoi" plutôt que le "comment"
- Identifie les exigences fonctionnelles et non-fonctionnelles
- Établit les limites du système

Conception

Élaboration de la solution technique qui répond aux besoins identifiés lors de l'analyse.

- Se concentre sur le "comment" implémenter
- Définit l'architecture du système
- Spécifie les composants et leurs interactions

Cycle de développement logiciel



Analyse Fonctionnelle vs Analyse Orientée Objet

Critère	Analyse Fonctionnelle	Analyse Orientée Objet
Approche	Décomposition du système en fonctions	Organisation du système en objets interagissant
Focus	Processus et flux de données	Entités, leurs attributs et comportements
Structure	Hiérarchie de fonctions	Hiérarchie de classes et d'objets
Réutilisabilité	Limitée	Élevée (héritage, polymorphisme)
Maintenance	Plus complexe pour les grands systèmes	Plus facile grâce à l'encapsulation

Avantages de l'analyse fonctionnelle

- Plus intuitive pour les systèmes procéduraux
- Meilleure pour les systèmes à flux de données complexes
- Facilite la compréhension des processus métier

Avantages de l'analyse orientée objet

- Meilleure modularité et réutilisabilité
- Représentation plus naturelle du monde réel
- Facilite l'évolution et la maintenance du système

Introduction à UML

UML (Unified Modeling Language) est un langage de modélisation graphique à base de pictogrammes, utilisé pour spécifier, visualiser, construire et documenter les artefacts d'un système logiciel.

Objectifs d'UML

- Fournir un langage de modélisation visuel expressif et facile à comprendre
- Permettre l'échange de modèles entre différents outils
- Intégrer les meilleures pratiques de l'industrie
- Supporter le développement orienté objet

Historique et évolution

- **1994-1995** : Fusion des méthodes de Booch, Rumbaugh (OMT) et Jacobson
- **1997** : UML 1.0 adopté par l'OMG (Object Management Group)
- **2005** : UML 2.0 avec améliorations majeures
- **2015** : UML 2.5 (version actuelle)

Domaines d'application

- Développement logiciel
- Modélisation de processus métier
- Ingénierie des systèmes

Les Aspects UML : Fonctionnel vs Architecture

Aspect Fonctionnel

- Se concentre sur ce que le système doit faire
- Décrit les fonctionnalités du point de vue de l'utilisateur
- Représente les interactions entre le système et les acteurs
- Diagrammes principaux : cas d'usage, activité, séquence

Aspect Architectural

- Se concentre sur comment le système est construit
- Décrit la structure interne et l'organisation du système
- Représente les composants et leurs relations
- Diagrammes principaux : classes, composants, déploiement

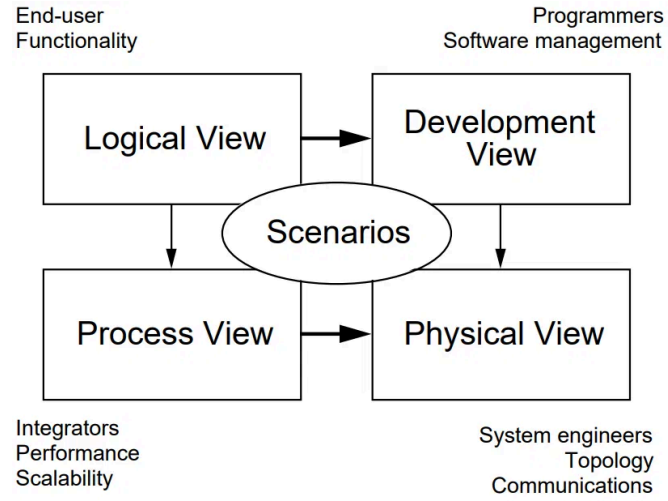
Complémentarité des aspects

- Les deux aspects sont nécessaires pour une modélisation complète
- L'aspect fonctionnel guide l'aspect architectural
- L'aspect architectural implémente l'aspect fonctionnel
- Ensemble, ils couvrent le "quoi" et le "comment" du système

Approche des 4+1 Vues

Le modèle d'architecture 4+1 vues est un cadre pour décrire l'architecture d'un système logiciel selon différentes perspectives.

- Chaque vue aborde un aspect spécifique du système
- La vue des cas d'utilisation est centrale et relie les autres vues



Vue des cas d'utilisation (centrale)

Décrit les scénarios et cas d'usage qui capturent les exigences fonctionnelles.

Vue d'implémentation

Décrit l'organisation des composants logiciels et artefacts.

Vue logique

Représente les abstractions clés sous forme de classes et d'objets.

Vue de déploiement

Illustre la distribution physique du système sur les nœuds matériels.

Vue de processus

Montre les aspects de concurrence et synchronisation du système.

Vue des Besoins

> La vue des besoins est la perspective qui capture les exigences fonctionnelles du système du point de vue des utilisateurs et des parties prenantes.

Objectifs

- Identifier les fonctionnalités requises par les utilisateurs
- Définir les limites du système
- Établir les interactions entre le système et son environnement
- Servir de base pour la validation du système

Acteurs et parties prenantes

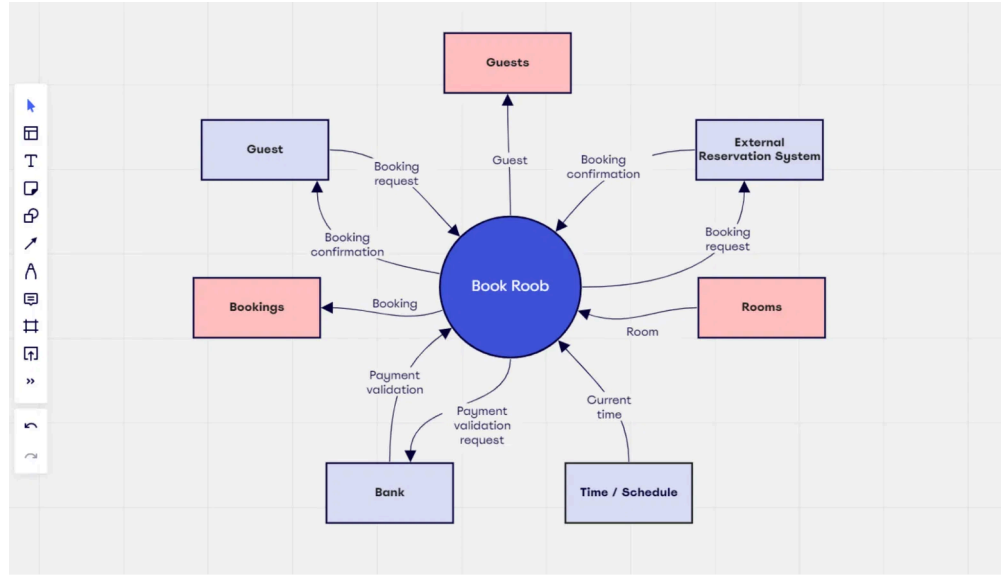
- Utilisateurs directs du système
- Systèmes externes en interaction
- Administrateurs et personnel de maintenance
- Décideurs et financeurs

Diagrammes associés

- **Diagramme de contexte** : montre le système dans son environnement
- **Diagramme de cas d'usage** : décrit les fonctionnalités du système
- Ces diagrammes servent de fondation pour les autres vues
- Ils facilitent la communication avec les parties prenantes non techniques

Diagramme de Contexte

> **Définition** : Le diagramme de contexte est une représentation de haut niveau qui montre le système comme une seule entité et ses interactions avec les éléments externes.



Objectifs

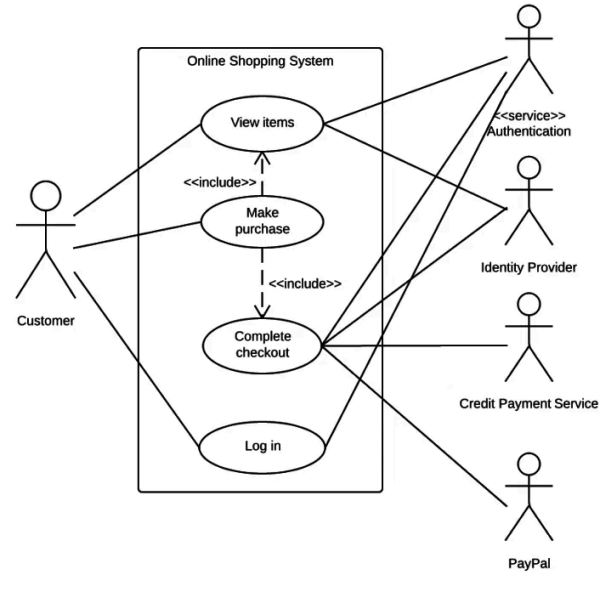
- Définir les frontières du système
- Identifier les acteurs et systèmes externes
- Visualiser les flux d'information entrants et sortants
- Établir une vision globale avant d'entrer dans les détails

Éléments constitutifs

- **Système central** : représenté par un cercle ou un rectangle
- **Acteurs externes** : personnes, organisations ou systèmes
- **Flux de données** : échanges d'information entre le système et les acteurs
- **Frontières du système** : délimitation claire du périmètre

Diagramme de Cas d'Usage

> **Définition** : Le diagramme de cas d'usage représente les interactions entre les acteurs et le système pour accomplir des objectifs spécifiques.



Éléments principaux

- **Acteurs** : utilisateurs ou systèmes externes qui interagissent avec le système
- **Cas d'usage** : fonctionnalités ou services fournis par le système
- **Relations** : liens entre acteurs et cas d'usage ou entre cas d'usage
- **Frontière du système** : délimitation du périmètre du système

Relations entre cas d'usage

Include

Un cas d'usage en inclut un autre comme partie de son comportement

Extend

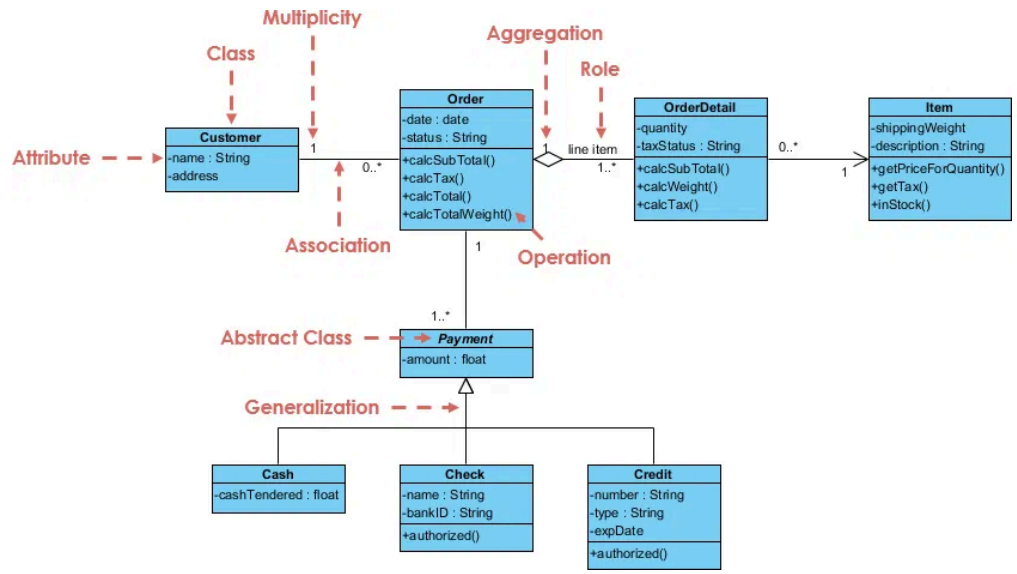
Un cas d'usage étend le comportement d'un autre dans certaines conditions

Généralisation

Un cas d'usage spécialise un cas d'usage plus général

Vue Logique et Diagramme de Classes

> Le diagramme de classes est le pilier de la vue logique, représentant la structure statique du système en termes de classes et de leurs relations.



Éléments d'une classe

- **Nom** : identifiant unique de la classe
- **Attributs** : propriétés ou données de la classe
- **Méthodes** : comportements ou opérations de la classe
- **Visibilité** : public (+), privé (-), protégé (#)

Relations entre classes

Association

Relation structurelle entre classes (ex: Client utilise Compte)

Agrégation/Composition

Relation tout-partie avec différents niveaux de dépendance

Héritage

Relation de généralisation/spécialisation entre classes

Conclusion

Points clés à retenir

- La modélisation est essentielle pour comprendre et concevoir des systèmes complexes
- L'analyse orientée objet offre une meilleure modularité et réutilisabilité que l'analyse fonctionnelle
- UML fournit un langage standardisé pour la modélisation orientée objet
- L'approche 4+1 vues permet d'aborder un système sous différentes perspectives
- Les diagrammes de contexte, de cas d'usage et de classes sont fondamentaux
- La modélisation UML facilite la communication entre toutes les parties prenantes

Prochaines étapes

- Mise en pratique avec l'outil StarUML
- Application à l'exercice "Gestion Dettes"
- Exploration des autres diagrammes UML
- Lien entre modélisation et implémentation