# Chapitre2 : Analyse

Introduction

1. Le choix de la méthodologie de conception :

2. Identification des acteurs et des cas des utilisations

3. Modèle d’analyse

Conclusion

## Introduction

Après avoir préciser l’objectif et les besoins attendus de ce projet dans le chapitre précèdent, La modalisation de notre application est basée sur le formalisme UML. Dans ce chapitre la présentation d’UML, leur définition, et leur cycle est étudié en premier lieu, puis la démarche conceptuelle sera présentée en deuxième lieu, ensuite les différents diagrammes permettant d’analyser le problème et de modéliser la solution seront décrits.

## Choix de la méthodologie de conception

### Adéquation d’UML et de modèle en cascade

Le développement de tout projet informatique nécessite l’utilisation d’une méthode de développement qui englobe, normalement, un modèle de conception et des formalismes de notations.

Face à la diversité des formalismes utilisés par les méthodes d’analyse et de conception objet, **UML** (Unified Modeling Language) représente un réel facteur de progrès par un effort de normalisation réalisé. Pour cette raison, on va utiliser ce langage de conception ou modélisations. En outre, on a choisi le modèle de la CASCADE comme démarche à suivre.

### Présentation du langage de modélisation : UML

UML (Unified Modeling Langage) n’est pas une méthode de développement mais un langage demodélisationqui définit des standards relatifs à la modélisation orientée objet. Ce langage permet de mettre en œuvre neuf (09) diagrammes différents.

Dans la mise en œuvre d’une méthode de développement, un diagramme est choisi en fonction des concepts que l’on veut représenter à une étape de développement donnée. Les concepts permettent de couvrir les étapes, depuis l’expression des besoins jusqu’au codage.

La Figure suivante résume UML avec ses vues et ses 9 diagrammes.

**La vue statique est caractérisée par**

* Les cas d’utilisation
* Diagramme d’objet
* Diagramme de classes
* Diagramme de composants
* Diagramme de déploiement

**La vue dynamique est caractérisée par**

* Diagramme de collaboration
* Diagramme de séquence
* Diagramme d’état-transition
* Diagramme d’activités

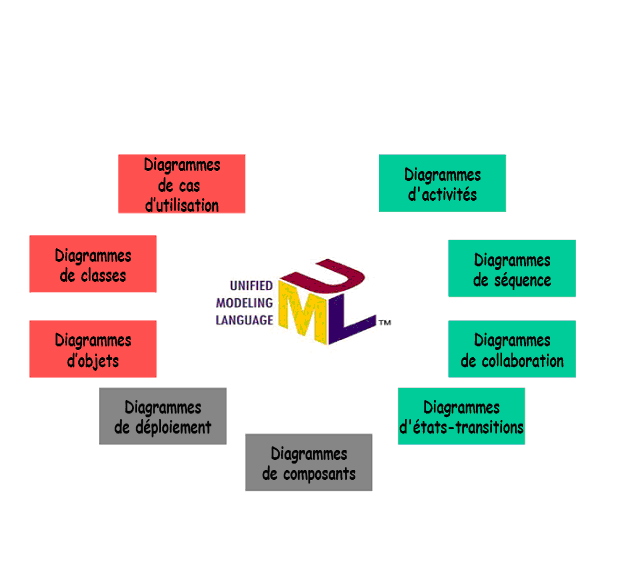
****

Figure 1: Les 9 Diagrammes en UML

**L’approche orientée objet :** La difficulté de cette modélisation consiste à créer une représentation abstraite, sous forme d’objets, d’entités ayant une existence matérielle (chien, voiture, ampoule, personne, . . .) ou bien virtuelle (client, temps, . . .).

* La Conception Orientée Objet (COO) est la méthode qui conduit à des architectures logicielles fondées sur les objets du système, plutôt que sur la fonction qu’il est censé réaliser.

### Avantage d’UML

UML cadre d’analyse objet, en offrant :

Différentes vues complémentaires d’un système, qui guide l’utilisation des concepts objets. Plusieurs niveaux d’abstractions, qui permettent de mieux contrôler la complexité dans l’expression des solutions objets.

UML est un langage formel et modélisé :

* Gain de précision.
* Gain de stabilité.
* Encourage l’utilisation d’outil

UML est un support de communication performant c'est-à-dire

* Qu’il facilite la compréhension des représentations abstraites et complexes.
* Un langage universel pouvant servir de support pour tout langage de programmation orientée objet.

UML **est idéal** pour :

* Concevoir et déployer une architecture logiciel développée dans un langage objet (Java, C++, VB.net). Certes UML, dans sa volonté "unificatrice" a proposé des Formalismes.
* Pour modéliser les données (le modèle de classe réduit sans méthodes et stéréotypé en entités), mais avec des lacunes que ne présentait pas l'entité relation de Merise.
* Pour modéliser le fonctionnement métier (le diagramme d'activité et de cas d'utilisation) qui sont des formalismes très anciens.

Pourquoi UML ?

Après une étude comparative, Il est certes que nous adoptons UML comme langage de modélisation puisque nous allons utiliser le concept de l’oriente objet, à travers le SDK androïde qui est basé sur JAVA, pour développer l’application au profit de SORETRAK.

Ainsi, la méthodologie de conception adoptée se base sur le choix de diagrammes UML adéquats. Nous avons utilisé quatre diagrammes : diagramme de cas d’utilisation, diagramme de séquence, diagramme de collaboration et diagramme de classes.

Le schéma suivant représente notre méthodologie de conception.











Figure 2: Méthodologie de conception adoptée

Notre outil de conception UML est le logiciel ***Rational Rose 2003 Enterprise Edition*** de Rational Software Corporation. C’est une référence pour la modélisation UML. Nous allons l’utiliser pour réaliser tous les diagrammes UML.

### Description du modèle de cycle de vie en cascade

Le modèle de cycle de vie **en cascade** a été mis au point dès 1966, puis formalisé aux alentours de 1970.

Le principe de modèle en cascade est de découper le projet en phases distinctes sur le principe de non retour. Dans ce modèle le principe est très simple : chaque phase se termine à une date précise par la production de certains document ou logiciels. Les résultats sont définis sue la base des interactions entre étapes, ils sont soumis a une revue approfondie et on ne passe à la phase suivante que s’ils sont jugés satisfaisants.

Le modèle en cascade est décrit par la figure suivante:

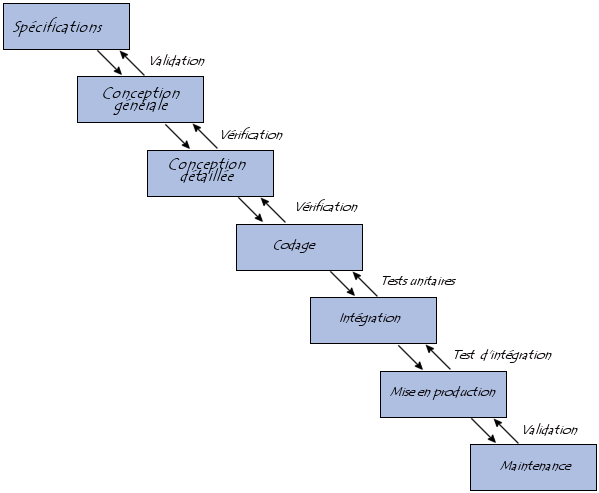


Figure 3: Modèle du cycle de vie en cascade

#### Avantages du modèle en CASCADE

* Production de documentation (grâce à cette documentation on peut revenir à l’étape précédente).
* Validation de l’étape précédente avant le passage à l’étape suivante.
* Largement utilisé.
* S’adapte aux projets simples dont les besoins sont clairement définis dés le début.

#### Inconvénients du modèle en CASCADE

* Cout très tardifs.
* Cout préventives élevé.
* Inadapté pour les gros projets.

### Rapport avec notre projet

Pour concevoir et implémenter notre Site web et notre application mobile nous avons tout d’abord choisi UML comme langage de modélisation.

En outre, nous avons choisi le modèle de la CASCADE comme un modèle de conception, grâce à sa simplicité car il nous permet la validation et la vérification pour le passage d’une étape à une autre.

## Identification des acteurs et des cas d’utilisation

### Identification des acteurs

Cette partie consiste à définir le périmètre du système, à définir le contour de l’organisation :

***Qu’est-ce qu’un acteur ?***

Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système. (La même personne physique peut donc être représentée par plusieurs acteurs en fonction des rôles qu’elle joue).

Par définition, un acteur est externe au périmètre de l’étude, qu’il appartienne ou pas à la société.

Enfin, un acteur n’est pas nécessairement une personne physique : il peut être un service, une société, un système informatique …



Figure 4: Diagramme d’acteurs

### Identification des cas d’utilisation

#### Définition

Il permet d’identifier les possibilités d’interaction entre le système et les acteurs (intervenants extérieurs au système), c’est-à-dire toutes les fonctionnalités que doit fournir le système. Lors de ces interactions, les acteurs génèrent des messages qui affectent le système informatique et appellent généralement une réponse de celui-ci .Nous allons isoler ces messages et les représenter graphiquement sur le diagramme de séquence.

#### Remarques

* Il s’agit de la solution UML pour représenter le modèle conceptuel.
* Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d’un système.
* Ils centrent l’expression des exigences du système sur ses utilisateurs : ils partent du principe que les objectifs du système sont tous motivés.
* Ils se limitent aux préoccupations « réelles » des utilisateurs ; ils ne présentent pas de solutions D’implémentation et ne forment pas un inventaire fonctionnel du système.

#### Eléments de base des cas d’utilisation

* Acteur : entité externe qui agit sur le système (opérateur, autre système...).
* L’acteur peut consulter ou modifier l’état du système.
* En réponse à l’action d’un acteur, le système fournit un service qui correspond à son besoin.
* Les acteurs peuvent être classés (hiérarchisés).
* Use case : ensemble d’actions réalisées par le système, en réponse à une action d’un Acteur.
* Les uses cases peuvent être structurés.
* Les uses cases peuvent être organisés en paquetages (packages).
* L’ensemble des use cases décrit les objectifs (le but) du système

### Description détaillé de cas d’utilisation

#### Diagramme de cas d’utilisation initiale

Figure 5: Diagramme de cas d’utilisation initiale

#### 2.3.2 Raffinement des cas d’utilisations

Au niveau de cette phase nous procéderons au raffinement de chaque cas d’utilisation, en effet on va présenter les scénarios d’utilisations possibles.

* **Administrateur**

##### Cas d’utilisation < Gérer catégorie >



Figure 6: Diagramme de Cas d’utilisation < Gérer catégorie >

Description textuelle de cas d’utilisation  < Gérer catégorie >

|  |  |
| --- | --- |
| Titre | Gérer catégorie |
| Acteur | Administrateur |
| Pré condition | l’administrateur doit s’identifier afin de pouvoir gérer les catégories |
| Post condition | L’administrateur pourra :   * Ajouter une nouvelle catégorie * Consulter la liste des catégories. * Modifier les informations concernant une catégorie * Supprimer une catégorie |

##### Cas d’utilisation < Gérer produit >



Figure 7: Cas d’utilisation < Gérer produit >

##### Description textuelle de cas d’utilisation < Gérer produit > :

|  |  |
| --- | --- |
| Titre | Gérer produit |
| Acteur | Administrateur |
| Pré condition | l’administrateur doit s’identifier afin de pouvoir gérer les produits |
| Post condition | L’administrateur pourra :   * Ajouter un nouveau produit * Consulter la liste des produits. * Modifier les informations concernant un produit * Supprimer un produit |

##### Cas d’utilisation  < gérer techniciens >



Figure 8: Cas d’utilisation < Gérer les techniciens >

##### Description textuelle de cas d’utilisation <  gérer les techniciens >

|  |  |
| --- | --- |
| Titre | Gérer techniciens |
| Acteur | Administrateur |
| Pré condition | l’administrateur doit s’identifier afin de pouvoir gérer les techniciens |
| Post condition | L’administrateur pourra :   * Ajouter un nouveau technicien * Consulter la liste des techniciens * Modifier les informations concernant un technicien * Supprimer un technicien |

##### Cas d’utilisation  < gérer agence >



Figure 9: Cas d’utilisation <gérer agence >

##### Description textuelle de cas d’utilisation < gérer agence >

|  |  |
| --- | --- |
| Titre | Gérer agences |
| Acteur | Administrateur |
| Pré condition | l’administrateur doit s’identifier afin de pouvoir gérer les agences |
| Post condition | L’administrateur pourra :   * Ajouter une nouvelle agence * Consulter la liste des agences * Modifier les informations concernant une agence * Supprimer une agence |

##### Cas d’utilisation <  consulter boite de réception >



Figure 10: Cas d’utilisation < Consulter boite de réception >

##### **Description textuelle de cas d’utilisation < Consulter boite de réception > :**

|  |  |
| --- | --- |
| Titre | Consulter boite de réception |
| Acteur | Administrateur |
| Pré condition | L’administrateur doit s’identifier afin de pouvoir consulter sa boite de réception |
| Post condition | l’administrateur pourra :   * Consulter la liste des msgs * Supprimer un message |

#### Cas d’utilisation finale

##### Cas d’utilisation finale pour l’acteur < Administrateur >



Figure 11: Cas d’utilisation finale pour l’acteur < Administrateur >

##### Cas d’utilisation finale pour l’acteur < Client >

****

Figure 12: Cas d’utilisation finale pour l’acteur <client>

2.3.3.2 Cas d’utilisation finale pour l’acteur < technicien >



Figure 13: Cas d’utilisation finale pour l’acteur < technicien >