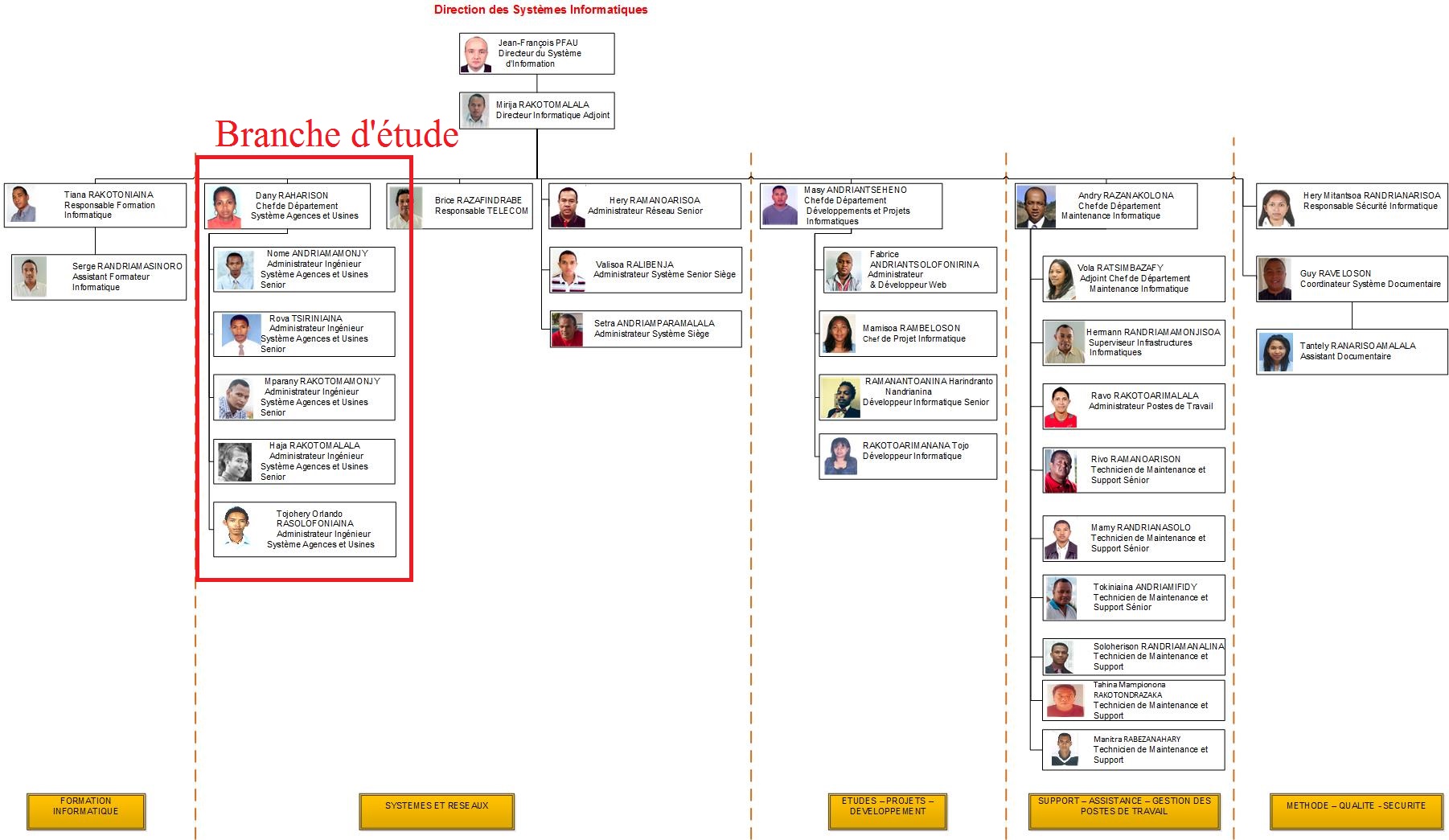
# **ANALYSE ET CONCEPTION**

## **ANALYSE PREALABLE**

### **Analyse de l’existant**

#### **Organisation actuelle**

La figure x montre l’organigramme du département Système Agences et Usines rattaché aux Systèmes et Réseaux.



#### **Mode de traitement des données**

Flux de transit des données

#### **Les applications existantes**

Script qui lance l’audit > crée un fichier log

Excel

#### **Inventaire des moyens matériels et logiciels**

Matériels utilisés par les responsables

### **Critique de l’existant**

Temps :

Connexion à distance.

Plus de tâches aux responsables.

### **Conception avant-projet**

#### **Proposition de solutions**

Face à ces contraintes, on a alors proposé les solutions suivantes :

1. Utiliser des logiciels spécifiques pour l’audit des tablespaces comme : **« Manage Engine Application Manager d’ORACLE ».**
2. Acheter un logiciel sur mesure auprès d’une SSII.
3. Automatiser les taches directement pour l’alimentation vers la base de donnée à l’aide d’un script et permettre de visualiser ces informations.

Choix 1 – Fanazavana kely anle étapes

#### **Avantages et inconvénients de chaque solution**

1. Utiliser des logiciels spécifiques pour l’audit des tablespaces comme : **« Manage Engine Application Manager d’ORACLE ».**

**Avantages :**

* Logiciel stable
* Vue globale et précis des informations des tablespaces

**Inconvénients :**

* Nécessite connexion à chaque site et exécution de script SQL spécifique
* Ouverture de plusieurs fenêtres pour chaque site
* Manque d’informations comme les détails des disques

1. Acheter un logiciel sur mesure auprès d’une SSII.

**Avantages :**

* Bénéfice des mises à jours
* Outil très robuste et opérationnel
* Gain de temps considérable
* Formation assurée par des professionnels

**Inconvénients :**

* Coût du logiciel considérable
* Coût de formation considérable

1. Automatiser les taches directement pour l’alimentation vers la base de donnée à l’aide d’un script et permettre de visualiser ces informations.

**Avantages :**

* Produit adapté aux besoins de l’utilisateur
* Vue globale de chaque site
* Logiciel pour le bénéfice de la Star sans dépense financière
* Code source modifiable

**Inconvénients :**

* Cout de temps considérable lors de la réalisation jusqu’à la mise en place
* Processus incrémental et itératif
* Difficulté de maintenir les mises à jours

#### **Choix de la solution et justification**

On a opté pour la dernière solution car la conception et la réalisation d’une application sur mesure répond pleinement aux besoins des futurs l’utilisateur. Pour ce faire, on a choisi de développer une application d’alimentation de données et d’un autre destinés à les afficher à partir d’un interface Web, nous avions aussi besoins d’une méthode de conception, d’un langage de programmation et d’un outil de développement. Dans le cadre de ce stage, cette dernière solution nous correspond parfaitement.

#### **Méthodologie**

Afin de procéder à la réalisation et la mise en place de cette solution, l’on doit tout d’abord se

##### **Méthode utilisé**

##### **Système de Gestion de Base de Données**

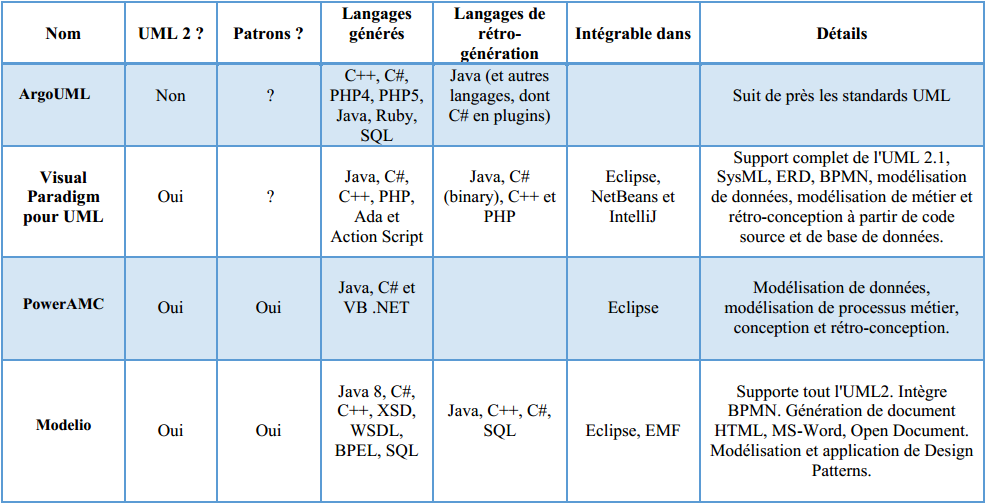
##### **Langage de programmation**

##### **Outil de modélisation**

Pour bien implémenter la phase de modélisation, il nous est utile durant le cours de notre projet d’utiliser un outil de modélisation pour permettre la création des diagrammes UML et des modèles qui en sont à l’origine.

Le tableau x présente la comparaison entre quelques outils de modélisation.

Tableau x: Comparaison de quelques outils de modélisation



**Choix et justification**

Nous avons opté pour Visual Paradigm durant notre projet car celui-ci permet de générer du code dans un langage de programmation déterminé afin de faciliter le développement. Il propose également la création d’autres types de diagrammes, comme celui qui permet la modélisation des bases de données pouvant, lui aussi, générer des canevas d’applications basé sur des Framework et Pattern mais en plus, générer du code SQL qu’il peut ensuite déployer automatiquement dans différents environnements.

## **ANALYSE CONCEPTUELLE**

### **Présentation de la méthode utilisée**

2TUP

### **Dictionnaire des données**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom** | **Signification** | **Type** | **Longueur** | **Règle de calcul** | **Commentaires** |
| **DIVISION** | Code du Site | AN | 2 |  |  |
| **DIVISION\_TYPE** | Type du Site | A | 6 |  |  |
| **DIVISION\_NAME** | Nom du site | AN | 64 |  |  |
| **DATE\_AUDIT\_JOURNALIER** | Date de l’audit | DD-MM-YYYY HH24 :MI :SS | 8 |  |  |
| **CACHE\_HIT\_RATIO** | Valeur de la cache hit ratio | N |  | **R1** |  |
| **ID\_OBJET\_DETAIL** | Identifiant de l’objet détail de l’audit journalier | N | 10 expo 127 |  |  |
| **TYPE\_OBJET** | Type de l’objet de la base de données | A | 12 |  |  |
| **NOM\_OBJET** | Nom de l’objet de la base de données | A | 64 |  |  |
| **TAILLE\_OBJET** | Taille de l’objet de la base de données | N |  |  |  |
| **NB\_EXTENT** | Nombre d’Extent | N |  |  |  |
| **TAILLE\_UTILISE** | Espace utilise par l’objet de la BD | N |  |  |  |
| **SERVEUR** | Catégorie du serveur | AN | 2 |  |  |
| **NOM\_DISQUE** | Nom de la disque | A | 3 |  |  |
| **ESPACE\_UTILISE** | Espace utilisé pour un disque | N |  | **R2** |  |
| **ESPACE\_TOTAL** | Espace total du disque | N |  | **R2** |  |
| **ESPACE\_LIBRE** | Espace libre du disque |  |  | **R2** |  |
| **TABLESPACES\_AVG\_FREE** | Moyenne de tous les espaces libres des tablespaces | N |  | **R** |  |
| **TABLESPACES\_AVG\_USED** | Moyenne de tous les espaces utilisées des tablespaces | N |  | **R3** |  |
| **TABLESPACES\_SUM\_TOTAL\_KO** | Somme du total en Ko des tablespaces | N |  | **R4** |  |
| **TABLESPACES\_SUM\_LIBRE\_KO** | Somme du total en Ko libre des tablespaces | N |  |  |  |
| **TABLESPACES\_SUM\_USED\_KO** | Somme du total en Ko utilise des tablespaces | N |  |  |  |
| **TAILLE\_TABLE\_KO** | Taille total des tables en Ko | N |  | **R5** |  |
| **TAILLE\_INDEX\_KO** | Taille total des index en Ko | N |  | **R6** |  |

### **Règles de gestion**

**R1 :**

**R2 :**

**R3 :**

**R4 :**

**R5 :**

**R6 :**

**R7 :**

**R8 :**

**R9 :**

### **Représentation et spécification des besoins**

Le diagramme de cas d’utilisation est le diagramme principal du modèle UML. Il décrit la succession des opérations réalisé par un acteur (personne qui assure l’exécution d’une activité). Par conséquent, il assure la relation entre l’utilisateur et les objets que le système met en œuvre.

Le diagramme des cas d’utilisation permet de structurer les besoins des utilisateurs et les objectifs correspondants d’un système. Leur but est justement d’éviter de tomber dans la dérive d’une approche fonctionnelle, où l’on liste une litanie de fonctions que le système doit réaliser.

Un cas d’utilisation est donc une représentation d’un ensemble de séquence d’actions qui sont réalisées par le système et produisent un résultat observable intéressant pour un acteur particulier.

Pour représenter un diagramme de cas d’utilisation, on a besoin les éléments de bases suivant :

* **Acteur** : entité externe (utilisateur humain, dispositif matériel ou autre) qui agit sur le système étudié. Il peut consulter et/ou modifier directement l’état du système, en émettant et/ou en recevant des messages éventuellement porteurs de données.

**Formalisme :**

La figure 04 représente le formalisme d’un acteur

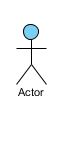


Figure 4: Formalisme d'un acteur dans UML

* **Cas d’utilisation** : suite d’interactions entre un acteur et le système. Il doit définir exhaustivement les exigences fonctionnelles du système où chaque cas d’utilisation correspond à une fonction métier du système, selon le point de vue d’un de ses acteurs.

**Formalisme :**

La figure 5 représente le formalisme d’un cas d’utilisation



Figure 5: Formalisme d'un cas d'utilisation dans UML

UML définit trois types de relations standardisées entre cas d’utilisation :

* **Relation d’inclusion** : Une relation représentée par un prototype « include » permet d’enrichir un cas d’utilisation (cas de base) par un autre cas d’utilisation (cas inclus). Le cas inclus est ajouté obligatoirement au cas de base. On utilise fréquemment cette relation pour éviter de décrire plusieurs fois le même enchainement, en factorisant le comportement commun dans un cas d’utilisation.

**Formalisme :**

La figure 6 représente le formalisme d’un cas d’utilisation.

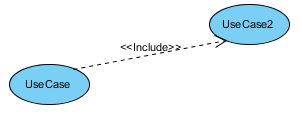


Figure 6: Formalisme d'une relation d'inclusion

* **Relation d’extension** : une relation représentée par prototype « extends » permet d’enrichir un cas d’utilisation par un autre, cependant, cet enrichissement est optionnel. Le cas de base peut fonctionner tout seul, mais il peut également compléter par un autre. On utilise principalement cette relation pour séparer le comportement optionnel du comportement obligatoire.

**Formalisme :**

La figure 7 représente le formalisme d’une relation d’extension.

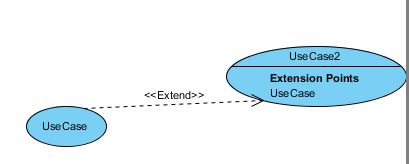


Figure 7: Formalisme d'une relation d'extension

* **Relation de généralisation/spécification** : Une relation d’héritage. La relation de généralisation entre deux entités exprime le fait que l’entité spécialisée est un cas particulier de l’entité général. L’entité spécialisée peut réaliser tout ce que l’entité général peut réaliser.

**Formalisme :**

La figure 8 représente le formalisme d’une relation de généralisation/spécialisation.

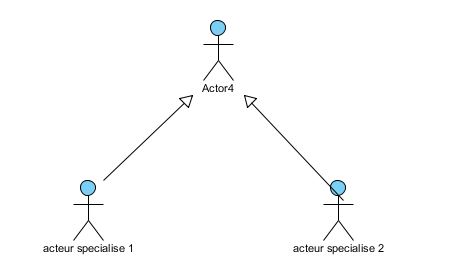


Figure 8: Formalisme d’une relation de généralisation/spécialisation

#### **Diagrammes des cas d’utilisation**

#### **Priorisation des cas d’utilisation**

#### **Diagramme de séquences système pour chaque cas d’utilisation**

### **Spécification des besoins techniques**

Les besoins du maître d’ouvrage étant modélisés de façon générale, des spécifications techniques nous ont été aussi données :

### **Modèle du domaine**

Un modèle du domaine est une visualisation des concepts d’un domaine du monde réel. L’élaboration du modèle des classes du domaine permet d’opérer une transition vers une véritable modélisation objet. L’analyse du domaine est une étape totalement séparée de l’analyse des besoins. Elle peut être menée avant, en parallèle ou après cette dernière. La phase d’analyse du domaine permet d’élaborer la première version du diagramme de classe appelée modèle du domaine. Ce modèle doit définir les classes qui modélisent les entités ou concepts présents dans le domaine de l’application. Il s’agit donc de produire un modèle des objets du monde réel dans un domaine donné. Ces entités ou concepts sont identifiés directement à partir de la connaissance du domaine ou par des entretiens avec des experts du domaine. [4]

Pour établir le diagramme, on suit des étapes :

* Identifier les entités ou concepts du domaine
* Identifier et ajouter les associations et les attributs
* Organiser et simplifier le modèle en éliminant les classes redondantes et en utilisant l’héritage
* Le cas échéant, structurer les classes en paquetage selon les principes de cohérence et d’indépendance.

## **CONCEPTION DETAILLEE**



### **Architecture du système**

REST Architecture

### **Diagramme de séquence de conception pour chaque cas d’utilisation**

Le diagramme de séquence de conception est la représentation graphique des interactions entre les objets manipulés par le système selon un ordre chronologique.

### **Diagramme de classe de conception pour chaque cas d’utilisation**

La modélisation des concepts ou des domaines permet d'identifier les objets importants dans une application. Ce processus nous permettra d'identifier les futurs problèmes et de mieux comprendre le fonctionnement de l'application. Ces concepts sont représentés dans le diagramme de classes. Le diagramme de classes est la clé de la conception orientée objet. Ce diagramme représente la structure du code à développer. Certaines applications UML permettent même d'exporter du code à partir de diagrammes de classes. Cela permet d'unifier le travail de plusieurs programmeurs au sein d'une même équipe, en plus de sauver du temps.

Le diagramme de classes se base sur les concepts suivants :

* **Classe :** description formelle d’un ensemble d’objets ayant une sémantique, des propriétés et un comportement commun.
* **Association :** relation sémantique entre deux ou plusieurs classes.
* **Propriété (attribut) :** élément permettant de décrire une classe ou une association.

### **Diagramme de classe de conception global**

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu’un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les cas d'utilisation ne réalisent donc pas une partition des classes du diagramme de classes. Un diagramme de classes n'est donc pas adapté (sauf cas particulier) pour détailler, décomposer, ou illustrer la réalisation d'un cas d'utilisation particulier. Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier. Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

### **Diagramme de paquetages**

Un paquetage permet de regrouper sous une même appellation un ensemble d'élément de modélisation UML tels que:

* Des classes, des composants, des nœuds, des collaborations, des cas d’utilisation, ...
* Des diagrammes de classes, de collaboration, de séquence, de cas d’utilisation, ...

Paquets alimentations et paquets consommations.

### **Diagramme de déploiement**

Un diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système et montre la répartition des composants sur ces matériels. Chaque ressource étant matérialisée par un nœud, le diagramme de déploiement précise comment les composants sont répartis sur les nœuds et quelles sont les connexions entre les composants ou les nœuds. Les diagrammes de déploiement existent sous deux formes : spécification et instance. [3].

Selon déploiement dans bloc note.