**CHAPITRE 2**

**Conception**

Table des matières

[**2.1. Introduction** 3](#_Toc73626861)

[**2.2. Cycle de vie** 4](#_Toc73626862)

[**2.3. Modèles de cycle de vie classique 2.3.1. Modèle en cascade** 4](#_Toc73626863)

[**2.3.2. Modèle en V**  Le modèle de cycle de vie en V part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception. 5](#_Toc73626864)

[**2.3.3. Modèle en spirale** 5](#_Toc73626865)

[**2.3.4. Modèle incrémental** 5](#_Toc73626866)

[**2.3.5.Prototypage** 5](#_Toc73626867)

[**2.4. Démarche à suivre** 6](#_Toc73626868)

[**2.5 Processus unifié UP** 6](#_Toc73626869)

[**2.5.1.Définition**  Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d’un utilisateur en système logiciel. L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétence et à différentes tailles de l'entreprise. 6](#_Toc73626870)

[**2.5.2. Caractéristiques du processus UP** 6](#_Toc73626871)

[**2.5.3. Vie du processus unifié**  UP est un ensemble de principes génériques adapté en fonctions des spécificités des projets. UP répond aux préoccupations suivantes:  **QUI** participe au projet ? 7](#_Toc73626872)

[**2.5.4 Avantage d’UP**  Pour le processus de développement UP, on peut citer les avantages suivants : 7](#_Toc73626873)

[**2.6. UML (*UnifiedModelingLanguage)*  2.6.1. Définition de l’UML** 7](#_Toc73626874)

[**2.6.2. Justification de la modélisation UML** 9](#_Toc73626875)

[**2.7. Spécifications initiales** 9](#_Toc73626876)

# **2.1. Introduction**

Après avoir connu la situation existante dans le chapitre précédent et compris le fonctionnement du système actuel, nous allons présenter dans ce chapitre les principaux concepts du langage UML « Unified Modeling Language » qui représente l’état de l’art des langages de modélisation objet. Il fournit les fondements pour spécifier, construire, visualiser et décrire les artefacts d’un système logiciel et ces concepts de base et offre également des mécanismes d’extension de ces concepts.

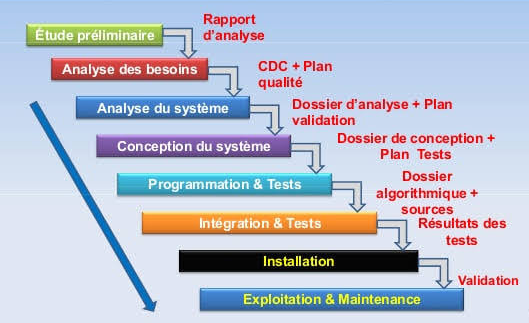
# **2.2. Cycle de vie**

Le cycle de vie d’un logiciel (en anglais *software life cycle*), désigne toutes les étapes du développement d’un logiciel, de sa conception à sa disparition. L’objectif d’un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel et la vérification du processus de développement. L’origine de ce découpage provient du constat que les erreurs ont un coût d’autant plus élevé qu’elles sont détectées tardivement dans le processus de réalisation.

Le cycle de vie permet de détecter les erreurs à temps et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés.  
 Il comprend généralement les activités suivantes :

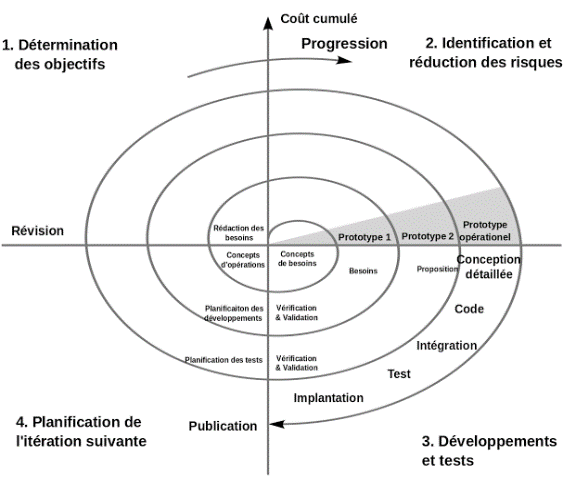
* Définition des objectifs.
* Analyse des besoins et faisabilité.
* Conception générale.
* Conception détaillée.
* Codage.
* Tests unitaires.
* Qualification.
* Documentation.
* Mise en production.
* Maintenance.

## **2.3. Modèles de cycle de vie classique 2.3.1. Modèle en cascade**

 Le modèle en cascade (en anglais : waterfall model) est un modèle de gestion linéaire qui divise les processus de développement en phases de projet successives. Contrairement aux modèles itératifs, chaque phase est effectuée une seule fois. Les sorties de chaque phase antérieure sont intégrées comme entrées de la phase suivante. Le modèle en cascade est principalement utilisé dans le développement de logiciels.

## **2.3.2. Modèle en V** Le modèle de cycle de vie en V part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception.

## **2.3.3. Modèle en spirale**

 L’idée de ce modèle est de réduire les risques dans le cycle de vie du logiciel et de fournir le plus rapidement possible un prototype permettant de valider les concepts, parfois de répondre à un appel d’offre avec un prototype de type illustratif*.* Le principe est alors de fournir régulièrement une nouvelle version de l’application en se rapprochant progressivement de l’application finale.

## **2.3.4. Modèle incrémental**

Dans ce modèle chaque incrément est une construction partielle du logiciel. Trie les spécifications par priorités et les rassembles dans des groupes de spécifications. Chaque incrément implémente un ou plusieurs groupes jusqu’à ce que la totalité du produit soit finie.

## **2.3.5.Prototypage**

Le projet se fait sur plusieurs itérations, les développeurs construisent un prototype selon les attentes du client, le prototype est évalué par le client qui donne son feedback après avec quoi les développeurs adaptent le prototype selon les besoins du client. Quand le prototype satisfait le client, le code est normalisé selon les standards et aussi les bonnes pratiques.

# **2.4. Démarche à suivre**

Aucun modèle n’est meilleur que l’autre car le choix se fait selon certains critères tel que la nature du projet, sa taille, la nature du client et nos compétences à pouvoir réaliser le projet.  
 Nous nous basons dans notre travail sur le prototypage. Les raisons pour lesquelles on a choisi ce modèle sont:

* Implication active du client donc moins de risque d’avoir des retours en arrière.
* Le développeur est en contact direct avec le client.
* S’adapte rapidement aux changements des besoins.
* Progrès constant et visible.
* Une grande interaction avec le produit.
* Implication active du client donc moins de risque d’avoir des retours en arrière.
* Le développeur est en contact direct avec le client.
* S’adapte rapidement aux changements des besoins.
* Progrès constant et visible.
* Une grande interaction avec le produit.

# **2.5 Processus unifié UP**

## **2.5.1.Définition** Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d’un utilisateur en système logiciel. L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétence et à différentes tailles de l'entreprise.

## **2.5.2. Caractéristiques du processus UP**

**2.5.2.1 Le processus unifié est piloté par les cas d’utilisation**

L’objectif principal d’un système logiciel est de rendre service à ses utilisateurs. Le processus de développement sera donc accès sur l’utilisateur. Les cas d’utilisation permettent d’illustrer ces services. Ils détectent puis décrivent les besoins fonctionnels (du point de vue de l’utilisateur), et leur ensemble constitue le modèle de cas d’utilisation qui dicte les fonctionnalités complètes du système.

**2.5.2.2 Le processus unifié est centré sur l’architecture**

Une architecture adaptée est la clé de voûte du succès d’un développement. Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performance, fiabilité…).Ph. Kruchten propose différentes perspectives, indépendantes et complémentaires, qui permettent de définir un modèle d’architecture (publication IEEE, 1995). Cette vue (« 4+1 ») a fortement inspiré UML.

**2.5.2.3 Le processus unifié est itératif et incrémental**

L’itération est une répétition d’une séquence d’instructions ou d’une partie du programme un nombre de fois fixé à l’avance ou tant qu’une condition définie n’est pas remplie. Elle qualifie un traitement ou une procédure qui exécute un groupe d’opérations de façon répétitive jusqu’à ce qu’une condition bien définie soit remplie.

## **2.5.3. Vie du processus unifié** UP est un ensemble de principes génériques adapté en fonctions des spécificités des projets. UP répond aux préoccupations suivantes: **QUI** participe au projet ?

**QUOI,** qu’est-ce qui est produit durant le projet ?

**COMMENT** doit-il être réalisé ?

**QUAND** est réalisé chaque livrable ?

## **2.5.4 Avantage d’UP** Pour le processus de développement UP, on peut citer les avantages suivants :

* Méthodologie complète ;
* Identification rapide des risques ;
* Largement adopté en entreprise ;
* Intégration avec UML ;

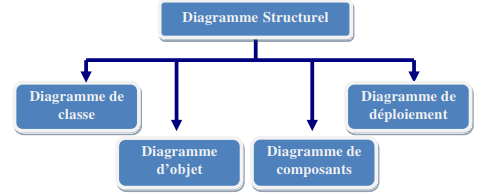
# **2.6. UML (*UnifiedModelingLanguage)* 2.6.1. Définition de l’UML**

Langage de modélisation unifié est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

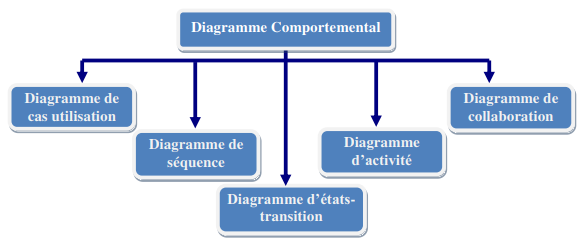
UML est à présent un standard défini par l'Object  
Management Group (OMG). La dernière version diffusée par l'OMG est UML 2.4.1 depuis aout 2011. Il est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet.Il offre un standard de modélisation, pour représenter l'architecture logicielle et propose 13 types de diagramme (9 en UML 1.3).

Diagrammes structurels et diagrammes comportemental :

⮚ **Vue Statique :**



⮚ **Vue Dynamique :**



Dons notre projet, nous avons utilisé 3 digrammes qui sont les plus utilisés lors de la modélisation (cas d'utilisation, séquence, classes).

## **2.6.2. Justification de la modélisation UML**

* Obtenir une modélisation de très haut niveau indépendante des langages et des environnements.
* Faire collaborer des participants de tous horizons autour d’un même document de synthèse.
* Faire des simulations avant de construire un système.
* Exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécifications etc.
* Documenter un projet.
* Générer automatiquement la partie logicielle d’un système.
* Valider le modèle vis à vis des clients.
* Spécifier les structures de données et le comportement du système.
* Fournir un guide pour la construction du système.
* Documenter le système et les d´excisions prises.

# **2.7. Spécifications initiales**

L’objectif de cette partie est d’analyser et définir les besoins de haut niveau et les caractéristiques du futur logiciel. Il se focalise sur les fonctionnalités requises par les utilisateurs, et sur la raison d’être de ces exigences. Le détail de la description des besoins se trouve dans les spécifications des cas d’utilisations.

**Identification des messages échangés**Un message représente la spécification d’une communication unidirectionnelle entre objets qui transporte de l’information avec l’intention de d´enclencher une activité chez le récepteur .  
Ci-dessous un résume des messages ´échangés entre les acteurs et le système :  
*•* Messages entrants (acteur vers système) :  
– M1 : Demande d’authentification ;  
– M2 : Demande de consultation ;  
– M3 : Etablir un bon d’entrée ;  
– M4 : Etablir un bon de sortie ;  
– M5 : Etablir un bon de consommation ;  
– M6 : Etablir un bon de transfert ;  
– M7 : Etablir une fiche de stock ;  
– M8 : Mettre `a jour les utilisateurs ;  
– M9 : Mettre `a jour les produits ;  
– M10 : Etablir l’inventaire ;  
– M11 : Etablir la demande de fourniture.  
*•* Messages sortants (système vers acteur) :  
– M12 : Interface d’authentification ;  
– M13 : Interface de consultation adéquate ;  
– M14 : Interface pour ´établir un bon d’entrée ;  
– M15 : Interface pour ´établir un bon de sortie ;  
– M16 : Interface pour ´établir un inventaire ;  
– M17 : Interface de mise `a jour adéquate ;  
– M18 : Interface pour remplir la demande de fourniture.