جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Université Abderrahmane Mira de Bejaïa
Faculté des sciences exactes
DEPARTEMENT D'INFORMATIQUE

Génie Logiciel

Niveau : 3ème année licence (MI)

Chapitre 1 : Introduction au Génie Logiciel

Chargés de cours :
Dr M. MOHAMMEDI
& Dr N. YESSAD

Année universitaire 2024/2025

Plan du chapitre 1

- Introduction
- Définition et objectifs
- Qualités exigées d’un logiciel
- Cycle de vie d'un logiciel
- Modèles de cycle de vie d'un logiciel

Introduction

Le concept de génie logiciel a été introduit à la fin des années 1960 lors d'une conférence organisée pour répondre à ce qui était alors connu sous le nom de « **crise du logiciel** ».

Les **diverses causes** à l'origine de cette crise sont les suivants :

- La qualité du produit logiciel livré était imparfaite et ne répondait pas aux besoins des utilisateurs. De plus, le logiciel produit consommait plus de ressources que prévu et était à l'origine de pannes;
- Les performances du logiciel telles que le temps de réponse, la disponibilité, le coût de stockage, etc., étaient très souvent médiocres;
- Les délais fixés pour la production des produits logiciels sont généralement dépassés et ne répondent pas aux spécifications ;
- La prédiction des coûts de développement de logiciels était presque infaisable et habituellement excessif;
- L'invisibilité du logiciel, ce qui signifie qu'il a souvent été constaté que le logiciel développé ne répondait pas à la demande au moment de la livraison;
- La maintenance du produit logiciel était difficile, coûteuse et souvent source de nouvelles erreurs.

Définition & Objectifs

Définition

Le génie logiciel englobe l'ensemble des méthodes, techniques et outils utilisés pour produire des logiciels de qualité tout en respectant les contraintes de coûts et de délais.

Objectifs

- Fournir un logiciel qui répond aux besoins des utilisateurs;
- Garantir que la qualité du logiciel est conforme aux exigences initiales du contrat de service;
- Maîtriser les coûts pour qu'ils restent dans les limites prévues initialement.
- Respecter les délais établis;
- Assurer la fiabilité, la performance et la sécurité du logiciel;
- Faciliter la maintenance et l'évolution du logiciel;
- Favoriser la réutilisation des composants logiciels;
- Optimiser l'efficacité et la productivité du processus de développement logiciel;
- Promouvoir la collaboration et la communication au sein de l'équipe de développement;
- Appliquer des bonnes pratiques pour minimiser les risques et les erreurs.

Qualités exigées d'un logiciel

Un logiciel est considéré de qualité s'il satisfait **au minimum** les critères suivants :

- **La fiabilité (ou robustesse) :** Accomplir sans défaillance l'ensemble des fonctionnalités spécifiées, dans un environnement opérationnel de référence et pour une durée d'utilisation donnée.

- **La maintenabilité :** C'est la facilité avec laquelle la maintenance d'un produit logiciel peut être effectuée.

- **L'efficacité :** est la capacité d'un système logiciel à utiliser le minimum de ressources matérielles, que ce soit le temps machine, l'espace occupé en mémoire externe et interne, ou la bande passante des moyens de communication, etc.

- **La facilité d'emploi :** Est la facilité avec laquelle des personnes présentant des formations et des compétences différentes peuvent apprendre à utiliser les produits logiciels et s'en servir pour résoudre des problèmes. Elle recouvre également la facilité d'installation, d'opération et de contrôle.

Cycle de vie du logiciel

Le cycle de vie d'un logiciel est une séquence de phases qui comprend sa conception, son développement, ses tests, son déploiement et sa maintenance jusqu'à sa fin de vie, en suivant une approche structurée pour assurer sa qualité et son évolution.

Le cycle de vie d'un logiciel permet de détecter les **erreurs** dès que possible, ce qui permet de **garantir la qualité** du produit, de **respecter les délais** de développement et de **maîtriser les coûts** associés.

Cycle de vie du logiciel

Les activités principales du processus de développement d'un logiciel, indépendamment de l'approche utilisée, comprennent les phases suivantes :

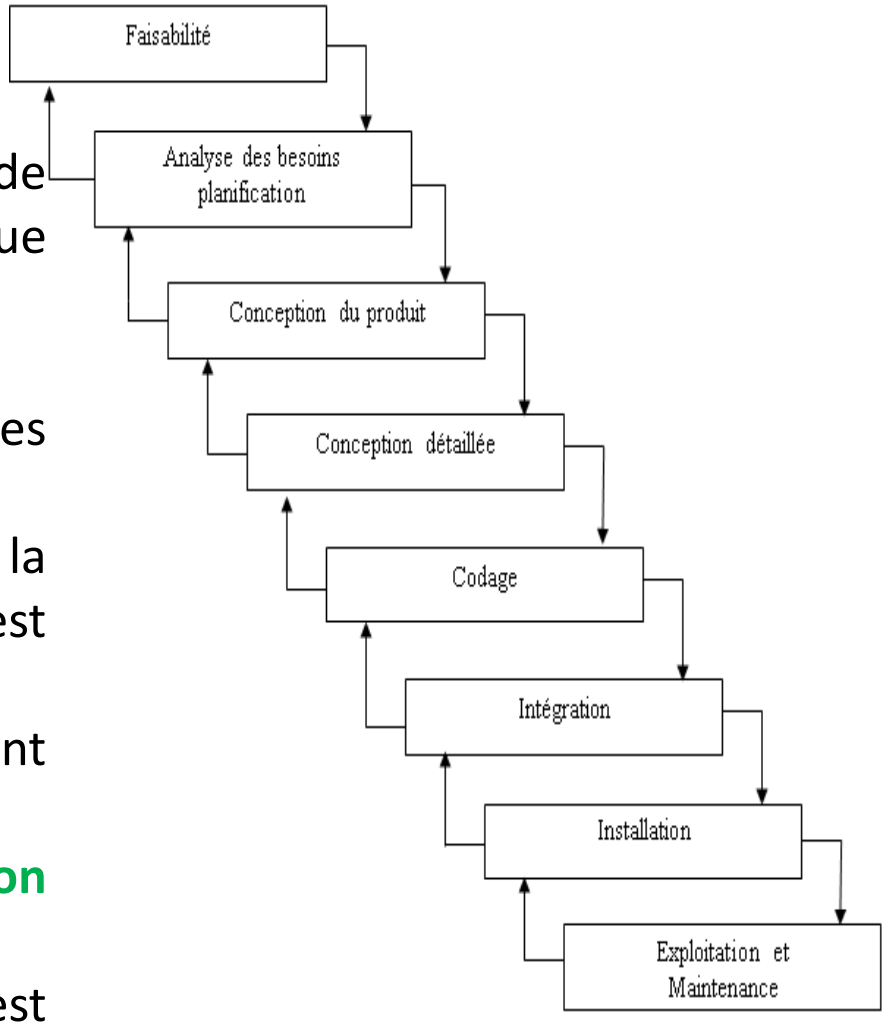
- L’analyse des besoins
- La spécification globale
- Conception architecturale et détaillée
- Programmation (Codage)
- Gestion de configuration et intégration
- Validation et vérification
- Prototypage

Dans le domaine du développement logiciel, il existe une variété de cycles qui sont utilisés pour mener à bien la création d'un logiciel. Ces cycles incluent toutes les étapes nécessaires à la conception du logiciel. Parmi les modèles couramment utilisés, on peut mentionner :

1) Le modèle en Cascade

- Le modèle de cycle de vie en Cascade est créé par Winston Royce en 1970.
- Le principe de ce modèle est d'avoir un certain nombre d'étapes chacune se terminant à une date précise par la production de certains documents ou logiciels;
- On ne passe à l'étape suivante que si l'étape en cours est terminée. Si une erreur critique est rencontrée, il est possible de revenir à la première étape.
- Le modèle en cascade d'origine ne permettait pas de faire des retours en arrière.

1) Modèle en cascade (suite)



Les versions actuelles du modèle en cascade intègrent la **validation** et la **vérification** à chaque étape. On observe ainsi successivement :

- Lors de la faisabilité et de l'analyse des besoins, une **validation** est effectuée.
- Pendant la conception du produit et la conception détaillée, une **vérification** est réalisée.
- Pendant le codage, des **tests unitaires** sont effectués.
- Lors de l'intégration, des **tests d'intégration** sont réalisés, suivis des **tests d'acceptation**.
- Durant l'installation, un **test système** est effectué.

Modèle de cycle de vie du logiciel

1) Modèle en cascade (suite)

Avantages

- Une structure simple grâce à des phases de projet clairement délimitées.
- Une bonne documentation du processus de développement produite par des étapes clairement définies.

Inconvénients

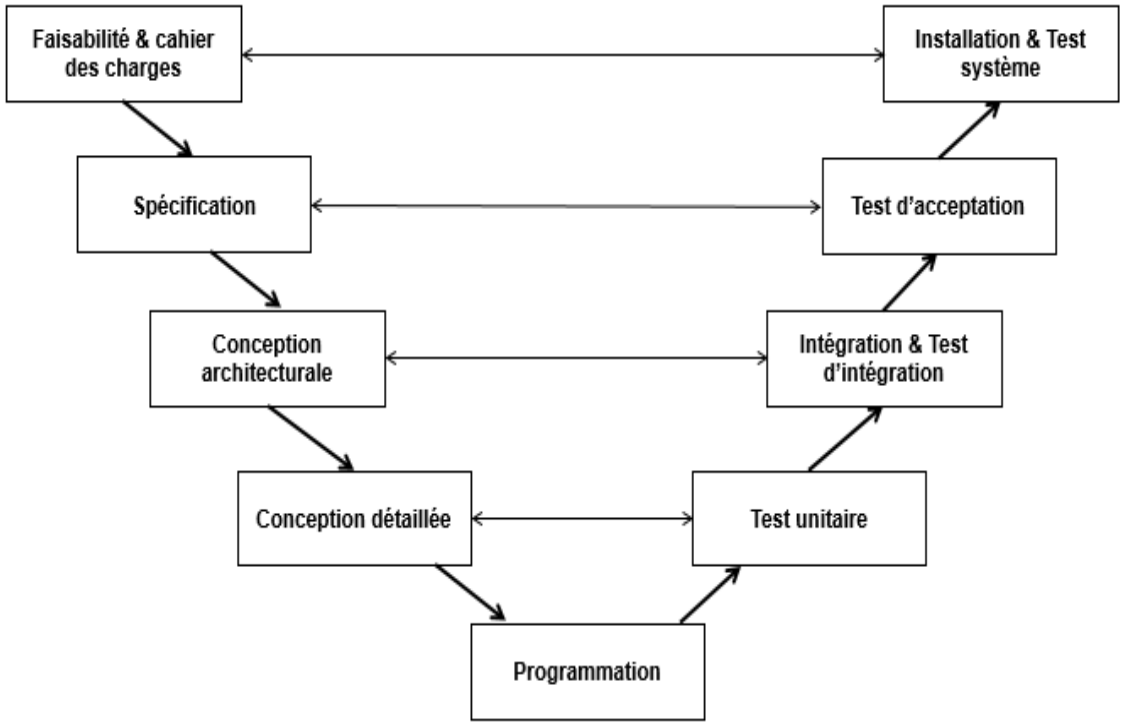
- Manque de transparence envers le client pendant le développement;
- Difficulté à obtenir toutes les spécifications du client;
- Détection tardive des erreurs.

Ce modèle est adopté dans des projets de développement logiciel avec des exigences stables et bien définies, où les changements majeurs sont peu probables une fois que le processus de développement a commencé.

2) Le modèle en V

Le modèle en V est une variation du modèle en cascade qui met également en évidence les **relations logiques** entre les **phases distantes**.

- On observe l'ajout de flèches horizontales en plus des flèches inclinés dans le modèle en cascade, ce qui reflète la séquence linéaire des étapes.
- Les flèches horizontales indiquent que certains résultats de l'étape de départ sont directement utilisés par l'étape d'arrivée.
- Le modèle en V est un cycle normalisé et largement utilisé dans les processus de développement.



2) Modèle en V (suite)

Avantages

- Contient des tests de bas niveau (tests unitaires) et des tests de haut niveau (tests de système) ;
- Identifier clairement les différentes étapes de développement et de test ;
- Affinement progressif de haut en bas, avec une division claire du travail à chaque étape, ce qui facilite le contrôle global du projet.
- Les éventuelles erreurs peuvent être détectées plus tôt.

2) Modèle en V (suite)

Inconvénients

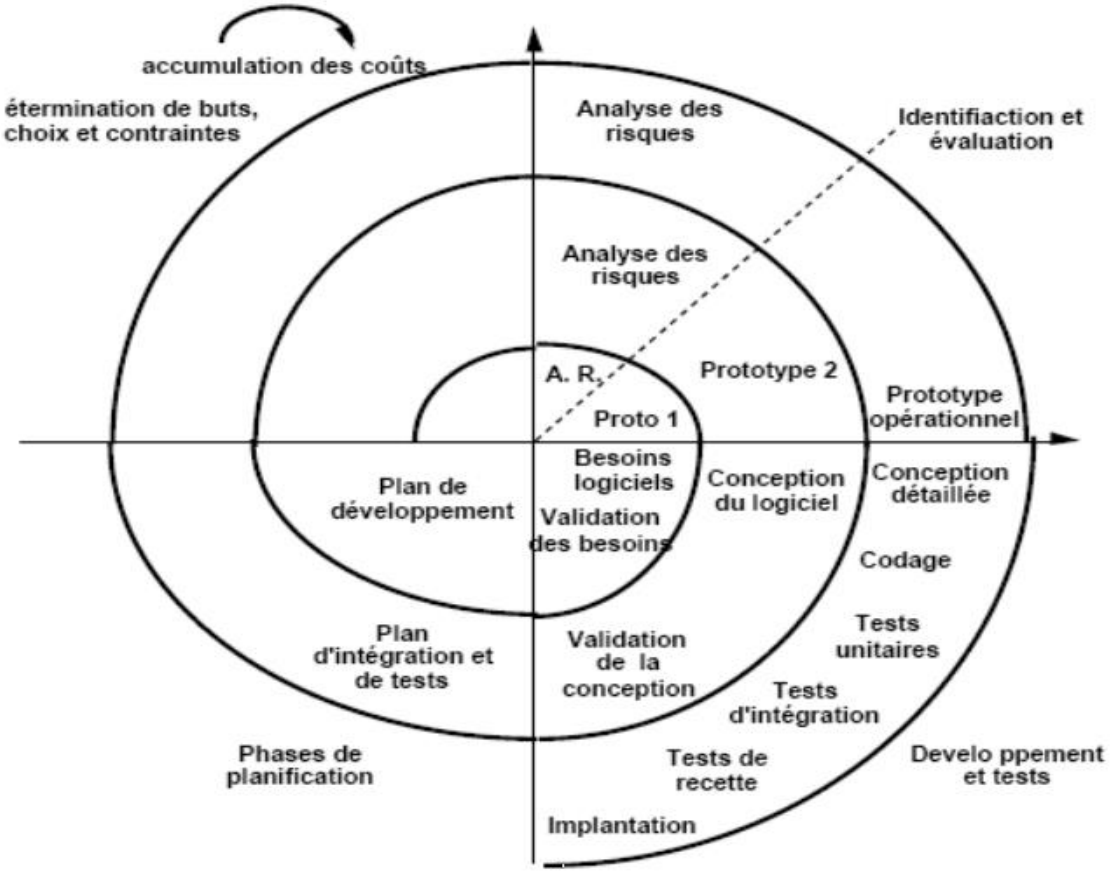
- La séquence descendante permet de tester le travail qui ne peut pas être modifié à temps après le codage ;
- Dans le travail réel, les exigences changent souvent, ce qui entraîne une exécution répétée des étapes du modèle V, une grande quantité de retouches et une faible flexibilité.

Ce modèle est adopté dans des projets de développement logiciel où l'accent est mis sur les activités de test et de validation pour garantir la conformité aux exigences spécifiées.

3) Le modèle en spirale

Le modèle de cycle de vie en spirale a été proposé par Barry BOEHM en 1988

Dans le modèle en spirale, les **risques** sont continuellement pris en compte et gérés à travers des **itérations successives**.



3) Modèle en spirale (suite)

Le modèle en spirale se déroule en cycles comprenant **quatre phases** distinctes :

Première phase : consiste à déterminer les objectifs du cycle, les alternatives pour les atteindre et les contraintes, en se basant sur les résultats des cycles précédents ou, en l'absence de ceux-ci, sur une analyse préliminaire des besoins ;

Deuxième phase : implique l'analyse des risques, l'évaluation des alternatives et éventuellement la création de maquettes ou de prototypes ;

Troisième phase : se concentre sur le développement et la vérification de la solution retenue ;

Quatrième phase : englobe la revue des résultats obtenus et la planification du cycle suivant.

3) Modèle en spirale (suite)

Avantages

- Flexibilité de conception, qui peut être modifiée à toutes les étapes du projet.
- Au fur et à mesure de l'avancement du projet, le client dispose toujours des dernières informations sur le projet afin de pouvoir interagir efficacement avec la direction.

Inconvénients

- L'utilisation de ce modèle nécessite une expérience et une expertise considérables en matière d'évaluation des risques : dans le développement de projets risqués, si le risque n'est pas identifié à temps, il entraînera inévitablement des pertes importantes.
- Trop d'itérations augmenteront les coûts de développement et retarderont le temps de subordination.

3) Modèle en spirale (suite)

Ce modèle est généralement adopté dans des projets de développement logiciel innovants, complexes et à haut risque, où l'approche itérative et une gestion rigoureuse des risques sont nécessaires.

Fin du Chapitre 1.