# Méthode de Conduite de Projet en Informatique :Modèles et activités de développement

BORIBO KIKUNDA PHILIPPE

ISP BUKAVU

L2 IG,2024-2025

# Plan

# Rôles d'un modèle

## Rôles d'un modèle

- Organiser les différentes phases du cycle de vie pour l'obtention d'un logiciel fiable, adaptable et efficace ;
- Guider le développeur dans ses activités techniques ;
- Fournir des moyens pour gérer le développement et la maintenance (ressources, délais, avancement, etc.) ;

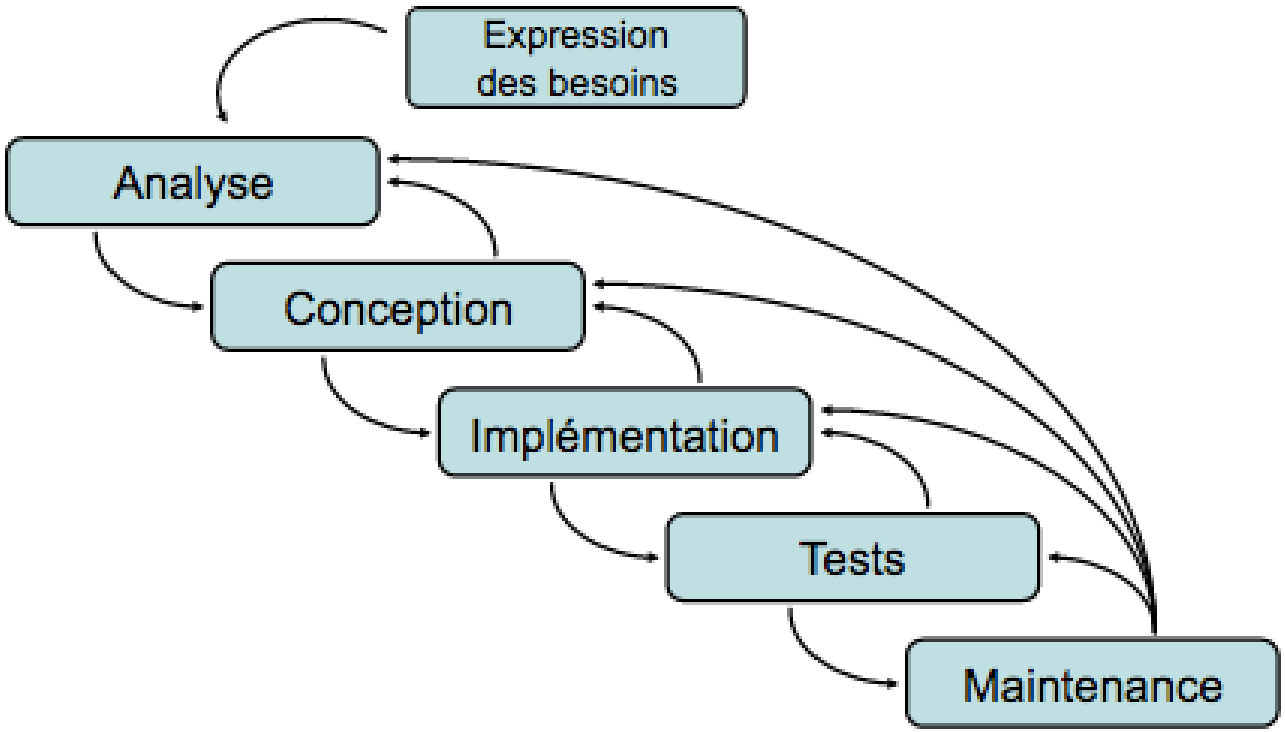
# Types de modèle

- Modèle code-and-fix ;
- Modèle (linéaire) en cascade.
- Modèle en V ;
- Modèle en spirale ;
- Processus unifié ;
- Agile ;

# Modèle en cascade

- Atteinte de l'objectif par atteinte ordonnée de sous objectifs ;
- Les activités sont représentées dans des processus séparés ;
- Processus séquentiel : Chaque étape doit être terminée avant que la suivante commence
- Livrables :
  - À la fin de chaque étape, le livrable est vérifié et validé ;
  - Vérification : le livrable est-il correct ?
  - Validation : est-ce le bon produit ? (Comparé à l'énoncé de l'étape).

# Modèle en cascade



# Modèle en cascade :Exemple

Projet : développement d'une application de gestion des taches :
1. Analyse des besoins
Dans cette phase,l'équipe de projet rencontre le client pour recueillir des informations sur ce qu'il attend de l'application. Les exigences pourraient inclure

## Analyse des besoins

- Créer, modifier et supprimer des taches
- Organiser les taches par catégories
- Ajouter des dates d'échéance et des rappels
- Partager des taches avec d'autres utilisateurs

# Modèle en cascade :Exemple

2. Conception : L'équipe conçoit alors l'architecture de l'application. Cela peut inclure :

## Conception du système

- un diagramme d'architecture montrant les composants(base des données, serveurr, interface utilisateur)
- des maquettes d'interface utilisateur(UI) pour visualiser comment les utilisateurs interagiront avec l'application
- la definition des API pour la communication entre le front-end et le back-end

# Modèle en cascade :Exemple

3. Implémentation(codage) : Les développeurs commencent à écrire le code, Par exemple

## Implémentation(codage) :

- Développement de la base des données pour stocker les taches.
- création des interfaces utilisateurs pour ajouter et afficher les taches ;
- Ecriture du code pour gerer la logique m"tier, comme le tri des taches par date d'échéance

# Modèle en cascade :Exemple

4. Tests : Une fois le développement terminé, l'équipe effectue plusieurs types de tests :

## Tests

- tests unitaires : verification que chaque fonction individuelle fonctionne correctement(par exemple, la fonction qui ajoute une tache)
- tests d'integration : verification que les differents modules fonctionnent bien ensemble(par exemple, s'assurer que l'interface utilisateur peut communiquer avec la base des données) ;
- tests fonctionnels : verification que toutes les exigences sont respectées(par exemple, s'assurer qu'un utilisateur peut effectivement créer et partager une tache)

# Modèle en cascade :Exemple

5. Deploiement : après avoir corrigé les bugs découverts lors des tests, l'application est déployée sur un serveur accessible aux utilisateurs finaux.

**Deploiement :**

Les utilisateurs peuvent maintenant télécharger l'application ou y accéder via un navigateur

# Modèle en cascade :Exemple

5. Maintenance : Apres le deploiement, l'équipe continue à surveiller l'application pour detecter tout problème. Par exemple :

## Maintenance

Les utilisateurs signalent un bug qui empêche l'ajout de rappels. De nouvelles fonctionnalités sont demandées, comme la possibilité d'ajouter des sous-taches.

# Modèle en cascade : avantages

- Simplicité et clarté : la structure linéaire rend le processus facile à comprendre et à gérer
- Documentation complète : Chaque phase produit une documentation claire qui facilite le suivi et la revision
- Planification prévisible : les délais et les couts sont plus facilement estimables grace à la nature séquentielle du modèle

# Modèle en cascade : Exemple 2

Exemple 2 : Développement d'un site web vitrine pour un petit commerce

**Contexte**

Un petit commerçant souhaite avoir un site web simple pour présenter son activité, ses produits et ses coordonnées.

1. Analyse des besoins
   - Recueillir les attentes du commerçant :
   - Présenter la page d'accueil avec une description de l'activité.
   - Avoir une page « Produits » avec une liste des articles.
   - Une page « Contact » avec adresse, téléphone et formulaire.
   - Définir les contraintes : site simple, responsive (adapté mobile), délai de 1 mois.

# Modèle en cascade : Exemple 2

2. Conception
   - Élaborer la structure du site :
   - Arborescence des pages (Accueil, Produits, Contact).
   - Maquettes des pages (wireframes).
   - Choix des technologies (HTML/CSS/JavaScript simple).

# Modèle en cascade : Exemple 2

2.1 Conception détaillée

- Définir la structure globale des pages :
  - Page d'accueil
  - Page Produits
  - Page Contact
- Organiser les liens de navigation entre les pages (menu principal, liens internes).
- Prévoir éventuellement un pied de page avec mentions légales, réseaux sociaux, etc.

# Modèle en cascade : Exemple 2

2.1 Conception détaillée : Spécifications fonctionnelles

- Page d'accueil :
  - Présentation succincte de l'activité (texte + image).
  - Mise en avant d'une offre ou promotion éventuelle.
- Page Produits :
  - Liste des produits avec nom, photo, description courte, prix.
  - Possibilité de filtrer ou classer les produits (si besoin).
- Page Contact :
  - Formulaire avec champs : nom, email, message.
  - Informations de contact : adresse, téléphone, horaires d'ouverture.
  - Carte intégrée (ex : Google Maps) pour localisation.

# Modèle en cascade : Exemple 2

2.1 Conception détaillée : Spécifications techniques

- Choix des technologies : HTML5, CSS3, JavaScript basique (sans framework complexe).
- Responsive design : le site doit s'adapter aux mobiles et tablettes (utilisation de media queries CSS).
- Accessibilité : respecter les bonnes pratiques (balises sémantiques, contrastes suffisants).
- Optimisation SEO basique : titres, méta-descriptions, URLs propres.

# Modèle en cascade : Exemple 2

2.1 Conception détaillée : Maquettes graphiques (wireframes / prototypes)

- Réaliser des croquis ou maquettes numériques pour chaque page :
- Positionnement des éléments (header, menu, contenu principal, footer).
- Choix des couleurs, typographies et styles graphiques simples et cohérents avec l'identité visuelle du commerçant.
- Validation des maquettes avec le client avant développement.

# Modèle en cascade : Exemple 2

2.1 Conception détaillée : Planification

- Estimation du temps nécessaire pour chaque tâche de développement.
- Identification des ressources (personnes impliquées, outils).
- Définition des jalons et dates de validation intermédiaires.

# Modèle en cascade : Exemple 2

2.1 Conception détaillée : Documentation Rédaction d'un document de conception regroupant :

- L'arborescence complète.
- Les spécifications fonctionnelles et techniques.
- Les maquettes validées.
- Le planning prévisionnel.

Ce document servira de référence tout au long du projet et permettra au client comme à l'équipe technique d'avoir une vision claire des attentes.

# Modèle en cascade : Exemple 2

3. Implémentation
   - Développer les pages web selon la conception validée.
   - Intégrer le contenu fourni par le commerçant.

# Modèle en cascade : Exemple 2

4. Tests
   - Vérifier la navigation entre les pages.
   - Tester le formulaire de contact.
   - Contrôler l'affichage sur différents navigateurs et mobiles.

# Modèle en cascade : Exemple 2

5. Déploiement
   - Mettre en ligne le site sur un hébergement web choisi.
   - Informer le commerçant que le site est accessible.

# Modèle en cascade : Exemple 2

6. Maintenance
   - Corriger d'éventuels bugs remontés par le commerçant.
   - Mettre à jour les informations si nécessaire (ex : nouveaux produits).

# Modèle en cascade : Inconvenients

- Rigide : une fois qu'une phase est terminée, il est difficile de revenir en arrière pour apporter des modifications, ce qui peut poser des problèmes si les exigences changent.
- Non adapté aux projets complexes : pour des projets plus complexes, où les exigences peuvent évoluer, ce modèle peut être trop restrictif ;
- Retard dans la direction des problèmes : les erreurs peuvent ne pas être découvertes avant la phase de test, rendant leur correction plus couteuse

NB : Ce modèle est un bon choix pour des projets simples avec des exigences bien définies et stables. Cependant non adapté pour des projets plus complexes ou dynamiques.

# Modèle en cascade : exercice d'application

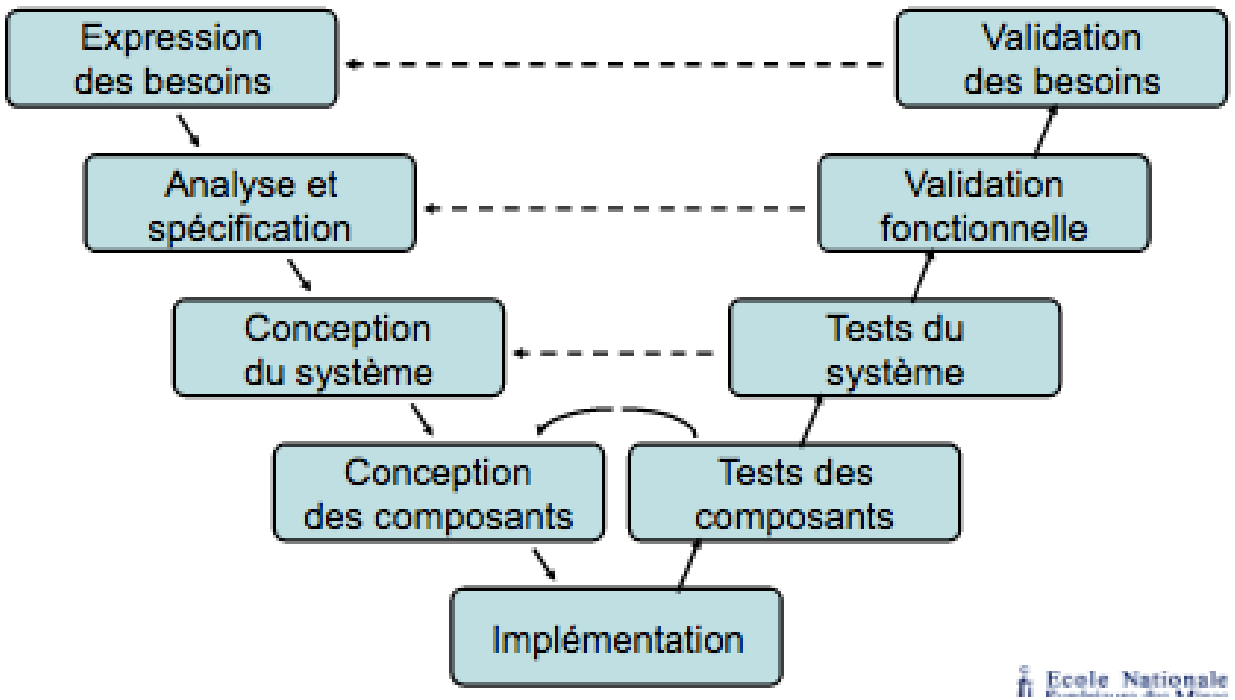
Élaborer l'analyse de besoin ainsi que la conception de votre projet de mémoire en vous inspirant sur l'exemple ci haut selon le modèle de cascade. Élaborer en estimant avec un plan prévisionnel le nombre de jours de chaque tâche.

# Modèle en V

- Amélioration du modèle en cascade ;
- Met en évidence la symétrie et la relation qu'il y a entre les phases du début du cycle de vie et celles de fin ;
- Les phases du début doivent être accompagnées d'une planification des phases de fin ;
- Lors de la planification, on développe et documente les plans de test.

NB : particulièrement adapté aux projets où la qualité est primordial et où les exigences sont bien comprises dès le depart car il offre une approche systémique pour s'assurer que chaque phase de développement est suivie d'une phase de test correspondante.

# Modèle en cascade



# Modèle en V : avantages

- détection précoce des défauts : les tests sont intégrés dès le le debut, ce qui permet de detecter et de corriger les problèmes rapidement ;
- clarté des exigences : la structure en V aide à s'assurer que toutes les exigences sont prises en compte et validées ;
- amélioration de la qualité : l'accent mis sur les tests et la validation conduit généralement à un produit final de meilleur qualité.

# Modèle en V : inconvenients

- Rigidité : comme le modèle en cascade, il peut être rigide face aux changements. Les modifications tardives peuvent être couteuses.
- Temps et couts élevés : le processus de test détaillé peut allonger le temps de développement et augmenter les couts.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
1. Analyse des besoins

## Analyse des besoins et validation

- Recueillir les besoins des utilisateurs (fonctionnalités souhaitées comme l'ajout, la modification, la suppression et l'affichage des tâches).
- Documenter les exigences fonctionnelles et non fonctionnelles (performance, sécurité, etc.).
- Organiser des ateliers avec les utilisateurs pour valider que toutes les exigences sont correctement comprises et documentées.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
2. Conception système :

## Conception système

- Élaborer l'architecture de l'application (choix des technologies, structure de la base de données).
- Définir les composants principaux (interface utilisateur, logique métier, accès aux données).
- Présenter l'architecture à l'équipe technique pour validation et ajustements éventuels.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
3. Conception detaillée :

## Conception detaillée :

- Créer des spécifications détaillées pour chaque module (ex. gestion des utilisateurs, gestion des tâches).
- Concevoir l'interface utilisateur (maquettes et prototypes).
- Réaliser des revues des spécifications avec les développeurs et les parties prenantes pour s'assurer qu'elles répondent aux exigences.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
4. Implémentation :

## Implémentation

- Développer le code pour chaque module en suivant les spécifications détaillées.
- Utiliser des pratiques de codage standard et effectuer des revues de code.
- Effectuer des revues de code pour s'assurer que le code respecte les spécifications.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
5. Tests unitaires :

## Tests unitaires

- Écrire et exécuter des tests unitaires pour chaque fonction développée (ex. ajout d'une tâche).
- Vérifier que chaque module fonctionne indépendamment selon les spécifications définies.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
6. Tests d'integration :

## Tests unitaires

- Tester l'interaction entre différents modules (ex. intégration entre la gestion des tâches et la gestion des utilisateurs)..
- S'assurer que les modules interagissent correctement et que les données circulent comme prévu.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
7. Tests systèmes :

## Tests systèmes

- Effectuer des tests fonctionnels sur l'application complète pour vérifier que toutes les fonctionnalités sont opérationnelles.
- Vérifier que l'application répond aux exigences initiales documentées lors de l'analyse.

# Modèle en V :Exemple

Projet : développement d'une application de gestion des taches :
8. Tests d'acceptation :

## Tests d'acceptation

- Réaliser des sessions de tests avec des utilisateurs finaux pour recueillir leurs retours.
- Obtenir l'approbation finale des utilisateurs finaux avant le déploiement.

# Modèle en V : Conclusion

## Conclusion

Le modèle en V permet une approche structurée et systématique du développement logiciel, où chaque phase est clairement définie et validée. Cela garantit que l'application de gestion des tâches répond aux besoins des utilisateurs tout en respectant les exigences techniques et fonctionnelles.

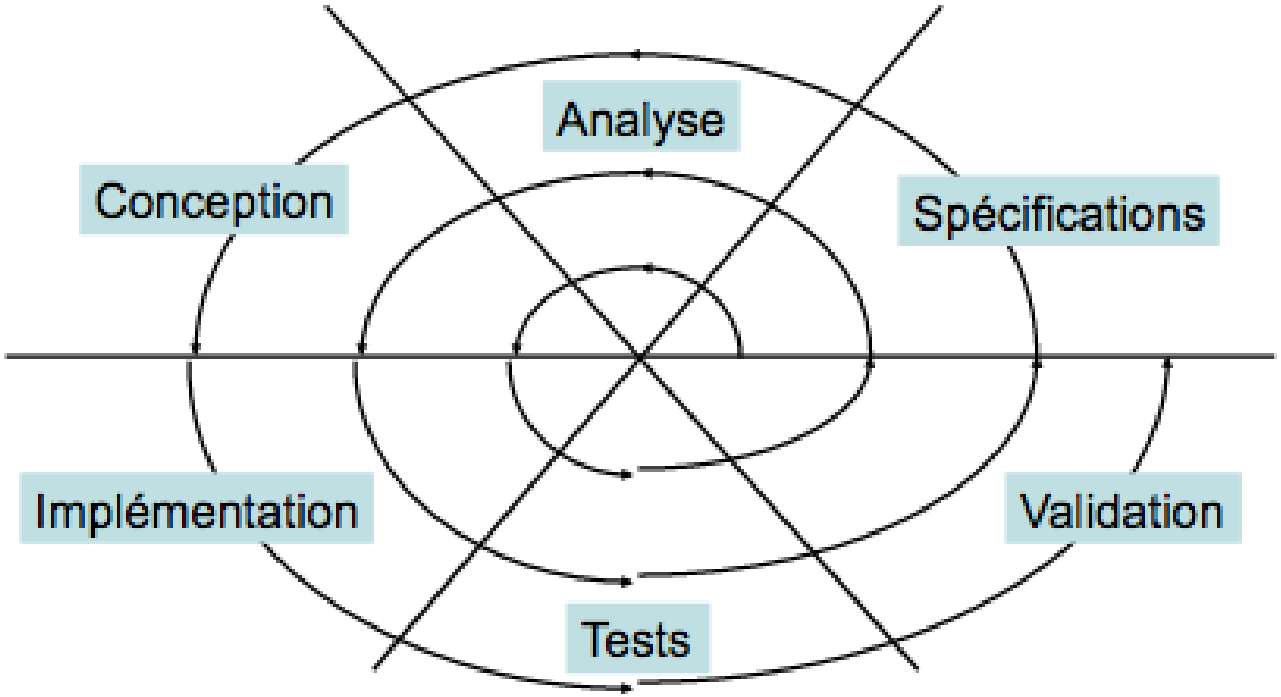
# Modèle en Spirale

Le développement de nouveaux logiciels pose un grand défi à toutes les parties prenantes. Plus l'application à développer est complexe, plus il est difficile d'organiser clairement le processus de développement et le rendre maîtrisable dans sa complexité.

# Modèle en Spirale

- Mise de l'accent sur l'évaluation des risques ;
- A chaque étape, après avoir défini les objectifs et les alternatives, celles-ci sont évaluées par différentes techniques (prototypage, simulation, ...), l'étape est réalisée et la suite est planifiée ;
- Le nombre de cycles est variable selon que le développement est classique ou incrémental ;

# Modèle en Spirale



# Modèle en spirale : avantages

- Modèle générique flexible ;
- Implication précoce du client et des utilisateurs possible ;
- Contrôle périodique dû aux risques ;
- Coordination parfaite entre exigences techniques et conception ;
- Maîtrise maximale des coûts, ressources et qualité du projet logiciel
- Adapté aux environnements techniques novateurs .

# Modèle en spirale : incovenients

- Effort de gestion important ;
- Les décisions régulières peuvent retarder le processus de développement ;
- À cause de la subdivision du processus de développement, des erreurs et incohérences de conception peuvent facilement se retrouver dans le produit final ;
- Connaissance en analyse et gestion des risques indispensable, mais souvent manquante ;
- Inadapté aux petits projets aux risques raisonnables

# Modèle en spirale : Exemple

Phase 1 : Planification

- Objectif : Définir les fonctionnalités de l'application de gestion des tâches.
- Activités :
  - Organiser des réunions avec les utilisateurs potentiels pour identifier leurs besoins (exemples : création de tâches, gestion des priorités, notifications).
  - Établir les grandes lignes du projet, incluant le budget et le calendrier de développement.

# Modèle en spirale : Exemple

Phase 2 : Analyse des risques

- Objectif : Identifier et évaluer les risques du projet
- Activités :
  - Analyser les risques liés à l ?interface utilisateur (difficulté de navigation), à la technologie choisie (stabilité de la plateforme), et aux retours des utilisateurs (réactions à l'application).
  - Élaborer des stratégies pour atténuer ces risques, comme des séances de feedback fréquentes et une documentation claire.

# Modèle en spirale : Exemple

Phase 3 : Développement et prototypage

- Objectif : : Créer un prototype de base.
- Activités :
  - Développer une version rudimentaire de l'application incluant des fonctionnalités comme la création, l ?édition et la suppression de tâches.
  - Présenter ce prototype aux utilisateurs sélectionnés pour obtenir des retours.

# Modèle en spirale : Exemple

Phase 4 : Validation

- Objectif : Recueillir des retours sur le prototype.
- Activités :
  - Faire tester le prototype par un groupe d'utilisateurs pour recueillir des commentaires sur l'expérience utilisateur et les fonctionnalités.
  - Identifier des problèmes potentiels, par exemple des difficultés à retrouver des tâches ou un manque de fonctionnalités (comme le classement par date d'échéance).

# Modèle en spirale : Exemple

Instruction de retour Les utilisateurs suggèrent d'ajouter des fonctionnalités telles que des rappels de tâches, des sous-tâches et la possibilité de marquer des tâches comme complètes.

# Modèle en spirale : Exemple

Phase 1 : Planification

- Objectif : Intégrer les retours d'expérience des utilisateurs et ajuster la planification..
- Activités :
  - Mettre à jour le cahier des charges pour inclure les nouvelles fonctionnalités proposées.
  - Réajuster le calendrier et le budget en tenant compte des nouvelles exigences..

# Modèle en spirale : Exemple

Repeter le cycle.

1. Nouvelle itération
   - Nouvelle itération
   - Activités :
     - Développement d'un produit amélioré : Ajouter les nouvelles fonctionnalités comme les rappels et les sous-tâches.
     - Tests utilisateur : Recueillir de nouveaux commentaires sur les modifications apportées.

# Modèle en spirale : Exemple

2. Analyse des risques : Identifier de nouveaux risques (ex. : complexité accrue de l'interface), et trouver des mesures pour les atténuer.

# Modèle en spirale : Exemple

3. Validation : Tester l'application mise à jour avec un plus grand nombre d'utilisateurs et ajuster selon leurs retours.

**NB**

Ce processus se répète jusqu'à ce que l'application de gestion des tâches réponde aux attentes des utilisateurs et soit prête pour une mise sur le marché. À chaque phase, le modèle en spirale assure une amélioration continue et une gestion proactive des risques.

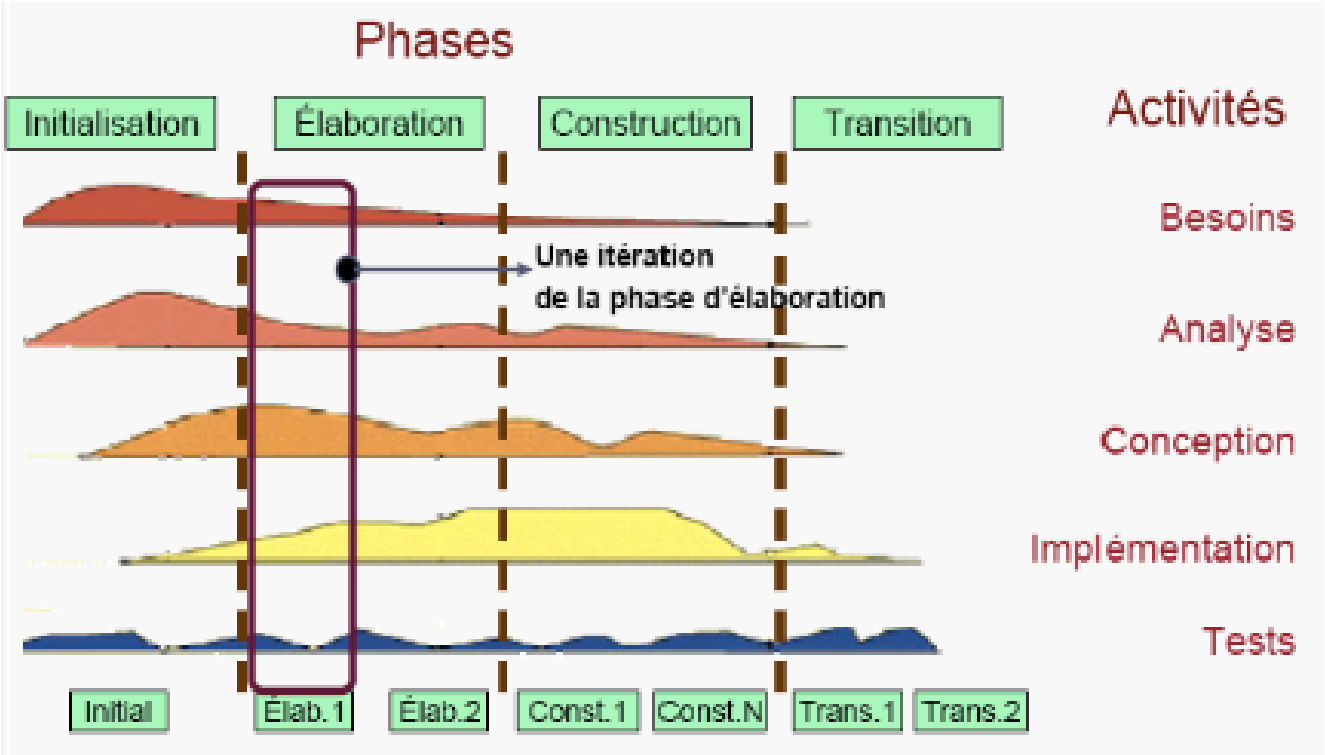
# Processus Unifié

- Regroupement des activités à mener pour le développement d'un système logiciel, basé sur la notion d'objets ;
- Piloté par les cas d'utilisation (bien comprendre les désirs et les besoins de ses futurs utilisateurs)
  - Un cas d'utilisation est une fonctionnalité du système produisant un résultat satisfaisant pour l'utilisateur. Les cas d'utilisation saisissent les besoins fonctionnels et leur ensemble forme le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système.
- Centré sur l'architecture (les différentes vues du système qui doit être construit)

# Processus Unifié

- Itératif et incrémental
  - Itératif : croissance et l'affinement successifs d'un système par le biais d ?itérations multiples, retours en arrière et adaptation cycliques
  - Incrémental : découpage du travail en plusieurs parties qui sont autant de mini-projets. Chaque mini-projet représente une itération ou étape de courte durée (1 mois) qui donne lieu à un incrément. Le résultat de chaque itération est un système testé, intégré et exécutable.

# Processus Unifié



# Activités de développement

- les sont décrites de façon indépendante en indiquant leur rôle, utilisent et produisent des ?artefacts ?
- Selon le modèle, une activité peut jouer un rôle plus ou moins important et parfois ne pas exister du tout ;
- Elles concernent :

# Activités de développement

- Planification (Étude de la faisabilité) ;
- Spécification des besoins (Requirement analysis) ;
- Analyse (Spécification formelle) ;
- Conception (Spécification technique) ;
- Implémentation (Codage) et tests unitaires ;
- Intégration et tests d'ensemble ;
- Livraison ;
- Maintenance .

# Planification

- Objectifs :
  - identification de plusieurs solutions et évaluation des coûts et bénéfices de chacune d'elles
- Activités :
  - simulation de différents scénarios de développement
- Résultats :
  - Rapport d'analyse préliminaire et un schéma directeur contenant : la définition du problème et les différentes solutions étudiées, avec, pour chacune d'elles, les bénéfices attendus, les ressources requises (délais, livraison, etc.

# Spécification des besoins

- Objectifs :
  - définition de ce que doit faire le logiciel
- Activités :
  - Description du problème à traiter afin d ?identifier les besoins de l'utilisateur, de spécifier ce que doit faire le logiciel : informations manipulées, services rendus, interfaces, contraintes
  - Mise en $\frac{1}{2}$uvre des principes : abstraction, séparation des problèmes, séparation des besoins fonctionnels
- Résultats : cahier des charges et plan qualité
  - un dossier d'analyse comprenant les spécifications fonctionnelles et non fonctionnelles du produit
  - une ébauche du manuel utilisateur pour les non informaticiens
  - une première version du glossaire contenant les termes propres au projet
  - un plan de test du futur système (cahier de validation)

# Analyse

- Objectifs :
  - Analyse détaillées de toutes les fonctions et autres caractéristiques que le logiciel devra réaliser pour l'usager, telles que vues par l'usager
- Activités :
  - Répondre au « Que fait le système ? », Modélisation du domaine d'application
  - Analyse de l'existant et des contraintes de réalisation
  - Abstraction et séparation des problèmes, séparation en unités cohérentes
- Résultats : Dossier d'analyse et plan de validation
  - Modèle du domaine
  - Modèle de l'existant (éventuellement)
  - Définition du modèle conceptuel.
  - Plan de validation, dossier de tests d'intégration

# Conception

- Objectifs :
  - Définition de l'architecture générale du logiciel. Spécification de la manière dont chacun des composants du logiciel sera réalisé et comment ils interagiront
- Activités :
  - Répondre au « Comment réaliser le système »
  - Décomposition modulaire, définition de chaque constituant du logiciel : informations traitées, traitements effectués, résultats fournis, contraintes à respecter
- Résultats : dossier de conception + plan de test global et par module
  - proposition de solution au problème spécifié dans l'analyse
  - organisation de l'application en modules et interface des modules (architecture du logiciel),
  - description détaillée des modules avec les algorithmes essentiels (modèle logique)
  - structuration des données.

# Implémentation

- Objectifs :
  - Réalisation des programmes dans un (des) langage(s) de programmation
  - Tests selon les plans définis lors de la conception
- Activités :
  - traduction dans un langage de programmation,
  - Mise au point (déboguage)
- Résultats : dossiers de programmation et codes sources.
  - Collection de modules implémentés, non testés
  - Documentation de programmation qui explique le code

# Tests unitaires

- Objectifs :
  - test séparé de chacun des composants du logiciel en vue de leur intégration
- Activités :
  - réalisation des tests prévus pour chaque module
  - les tests sont à faire par un membre de l'équipe n'ayant pas participé à la fabrication du module
- Résultats :
  - résultats des tests avec les jeux d ?essais par module selon le plan de test.

# Integration et Test Système

- Objectifs :
  - Intégration des modules et test de tout le système
- Activités :
  - Assemblage de composants testés séparément
  - Démarche d ?intégration (ascendante, descendante ou les deux)
  - Conception des données de tests
  - Tests Alpha : l'application est mise dans des conditions réelles d'utilisation, au sein de l'équipe de développement (simulation de l'utilisateur final)
  - Documentation des éléments logiciels
- Résultats :
  - Rapports de test
  - Manuel d ?utilisation

# Livraison, Maintenance et Evolution

- Objectifs :
  - Livraison du produit final à l'utilisateur,
  - Suivi, modifications, améliorations après livraison.
- Activités :
  - Tests Bêta : distribution du produit sur un groupe de clients avant la version officielle,
  - Livraison à tous les clients,
  - Maintenance : corrective, adaptative, perfective.
- Résultats : la version finale du manuel utilisateur, les traces d'évolution du système, les rapport d'exploitation
  - Produit et sa documentation
  - Trace d'exploitation et d'évolution

# TP 1

1.Vous travaillez sur un projet de développement d'une application de gestion de bibliothèque. Décrivez les étapes du modèle en cascade pour ce projet et donnez un exemple d'activité pour chaque étape.

# TP2

2. Vous devez développer un système de gestion des reservations pour un hotel, décrivez les différentes étapes du modèle en V et donnez un exemple d'activité pour chaque étape.

# TP3

3. Vous devez développer un système de gestion des reservations pour un hotel, décrivez les différentes étapes du modèle en Spirale et donnez un exemple d'activité pour chaque étape.

# TP4

Elaborer les différents documents obtenus à chaque étape du développement concernant les deux applications
A Remettre la semaine prochaine(du 26/08/2024)

Table de matière