Waterfall

Objectifs

- ✓ Comprendre les principes de la méthode Waterfall.
- ✓ Apprendre à planifier un projet en suivant les phases de la méthode Waterfall.
- ✓ Analyser les avantages et inconvénients de la méthode Waterfall.
- ✓ Appliquer la méthode Waterfall à un projet via une étude de cas.

Historique

- ✓ 1970 Winston W. Royce
  "Managing the Development of Large Software Systems"
- ✓ Très populaire dans les années 80-90


Étape 1 : La planification
Étape 2 : La conception
Étape 3 : L'implémentation
Étape 4 : La vérification
Étape 5 : La livraison
Étape 6 : La maintenance

Principe

- ✓ Chaque étape découle de la précédente
- ✓ Le projet est donc une cascade (waterfall) de phases séquentielles

1 Planification

- ✓ Analyse des besoins du client
- ✓ Spécification de la solution
- ✓ Cahier des charges

Livrables

- ✓ **Spécifications fonctionnelles :** Ce document contient toutes les exigences du projet, y compris les besoins fonctionnels (ce que le système doit faire) et non fonctionnels (performance, sécurité, etc.). Il sert de référence pour toutes les autres phases.

- ✓ **Cahier des charges :** Signé par les parties prenantes (client, prestataire et utilisateurs finaux) pour s'assurer que tous les besoins sont correctement compris et pris en compte.

Exemple

Imaginons un projet pour développer un logiciel de gestion d'inventaire pour une entreprise de distribution. Pendant cette phase, les parties prenantes (client, utilisateurs finaux, équipe technique) se réunissent pour discuter des fonctionnalités attendues, telles que :

- ✓ Suivi en temps réel des stocks.
- ✓ Alertes automatiques pour les réapprovisionnements.
- ✓ Génération de rapports de vente.

Toutes ces exigences seront documentées et validées par les parties prenantes avant de passer à l'étape suivante.

2 Conception

- ✓ Conception système
  - ✓ La stack technique
  - ✓ Les structures de données
  - ✓ L'architecture globale de la solution
- ✓ Conception détaillée
  - ✓ Les composants logiciels
  - ✓ Le découpage en modules
  - ✓ Les patterns utilisés

Livrables

- ✓ **Document de conception système** : décrit l'architecture globale du système, comment les différents modules interagiront entre eux, et les choix technologiques.
- ✓ **Document de conception détaillée** : présente les spécifications détaillées de chaque composant du système, y compris les schémas de base de données, les diagrammes d'interface utilisateur, les algorithmes et les flux de données.
- ✓ **Maquettes (Wireframes et/ou prototypes)** : Visuels ou prototypes des interfaces utilisateur (UI, IHM) pour valider le design avec le client.

Exemple

Pour le logiciel de gestion d'inventaire, cette phase inclurait :

- ✓ Le choix du serveur de messagerie, du serveur d'application et du protocole de communication entre ces deux serveurs
- ✓ La création d'un schéma de base de données
- ✓ Une conception d'interface utilisateur
- ✓ La définition des interactions entre le système de réapprovisionnement et les notifications par e-mail

3 Implémentation

- ✓ Codage de l'application selon le plan de conception
- ✓ Mise en oeuvre de l'architecture
  - ✓ installation des serveurs
  - ✓ Paramètrage

Livrables

- ✓ **Code source** : Le code développé pour chaque module du projet. Ce livrable inclut les fonctionnalités programmées telles que définies dans la phase de conception.
- ✓ **Documentation technique** : Cette documentation explique comment le code est structuré, comment les développeurs peuvent l'étendre ou le modifier, ainsi que des guides d'installation et d'utilisation du logiciel.
- ✓ **Journal des versions (versioning)** : Ce livrable trace l'évolution du développement, avec les versions de chaque module et les changements apportés.

Exemple

Dans le cadre du projet de gestion d'inventaire,
les développeurs créeront :

- ✓ Un module pour la gestion des produits, où les employés peuvent ajouter, supprimer ou modifier des informations sur les stocks.
- ✓ Un autre module pour générer automatiquement des rapports sur les ventes et l'état des stocks.

Chaque fonctionnalité sera programmée et testée de manière isolée avant d'être intégrée dans le système global.

4 Vérification

- ✓ Tests
  - ✓ unitaire
  - ✓ d'intégration
  - ✓ fonctionnel
  - ✓ de comportement ou d'acceptance

Livrables

- ✓ **Plan de test** : détaille les stratégies de test, y compris les tests unitaires, d'intégration, de système et les tests d'acceptation par l'utilisateur (UAT). Il décrit également les scénarios de test et les critères d'acceptation. Inclus les jeux de données pour les tests
- ✓ **Rapports de test** : Résultats des tests effectués sur le système. Ce rapport liste les bugs ou anomalies détectés, les actions correctives, et les résultats finaux après correction.
- ✓ **Validation d'acceptation** : document signé par le client ou les utilisateurs finaux confirmant que le système répond aux exigences définies et est prêt pour le déploiement. Généralement donne lieu à une facturation partielle.

Exemple

Pour le logiciel de gestion d'inventaire :

- ✓ Des tests unitaires seraient réalisés sur chaque module (ex. : ajout d'un produit, génération de rapports).
- ✓ Des tests d'intégration vérifieraient que les modifications dans la base de données sont correctement reflétées dans l'interface utilisateur et les notifications par e-mail.
- ✓ Le test fonctionnel consisterait à jouer des scenarii d'utilisation du logiciel (automatisés ou non)
- ✓ Le client final pourrait effectuer des tests d'acceptation pour s'assurer que le logiciel répond à ses besoins avant le déploiement.

5 Livraison

- ✓ Installation de l'architecture technique de production
- ✓ Déploiement de la solution
- ✓ Formation des utilisateurs

Livrables

- ✓ **Version déployée du logiciel** : Le logiciel final, compilé et installé sur l'infrastructure de production.
- ✓ **Manuel utilisateur** : Un document fournissant des instructions pour utiliser le système, destiné aux utilisateurs finaux.
- ✓ **Documentation de formation** : Guides ou supports de formation pour accompagner les utilisateurs dans la prise en main du système.
- ✓ **Plan de déploiement** : Un document qui décrit les étapes de déploiement, les dates, et les équipes responsables du déploiement. Il inclut également des informations sur les procédures de sauvegarde et de basculement en cas d'échec ainsi que le plan de migration des données.

Exemple

Dans le cadre du logiciel de gestion d'inventaire :

- ✓ Le logiciel client serait installé sur les ordinateurs de l'entreprise.
- ✓ Le serveur de base de données serait mis en production avec peut-être une migration des données de l'entreprise
- ✓ L'équipe pourrait former les employés sur l'utilisation des différentes fonctionnalités
- ✓ Une période de soutien technique immédiat serait souvent prévue pour résoudre rapidement tout problème inattendu après le lancement.

5 Maintenance

- ✓ Maintenance corrective (sus aux bugs)
- ✓ Maintenance préventive (optimisation, refactoring, amélioration de la qualité du code)
- ✓ Maintenance évolutive (nouvelles fonctionnalités)
- ✓ Maintenance adaptative (mise en conformité avec les nouvelles technos, les nouvelles obligations juridiques, les changements externes)
  ex : RGPD, Cloud, passage à la norme SEPA

Livrables

- ✓ **Journal des incidents et correctifs** : Un document répertoriant tous les incidents signalés après le déploiement et les actions correctives apportées.
- ✓ **Mises à jour logicielles** : Les nouvelles versions du logiciel, corrigées ou améliorées selon les besoins des utilisateurs ou pour résoudre des bugs.
- ✓ **Rapports de maintenance** : Rapports périodiques sur les interventions effectuées pour la correction des bugs, les mises à jour du système, ou les modifications apportées en réponse aux changements des besoins du client.

Exemple

Pour le logiciel de gestion d'inventaire, la maintenance pourrait inclure :

- ✓ La correction de bugs qui apparaissent lors de l'utilisation quotidienne.
- ✓ Des mises à jour pour intégrer de nouvelles fonctionnalités, comme un module d'analyse plus poussé des ventes.
- ✓ Des améliorations de performance pour traiter un nombre croissant de produits dans la base de données.

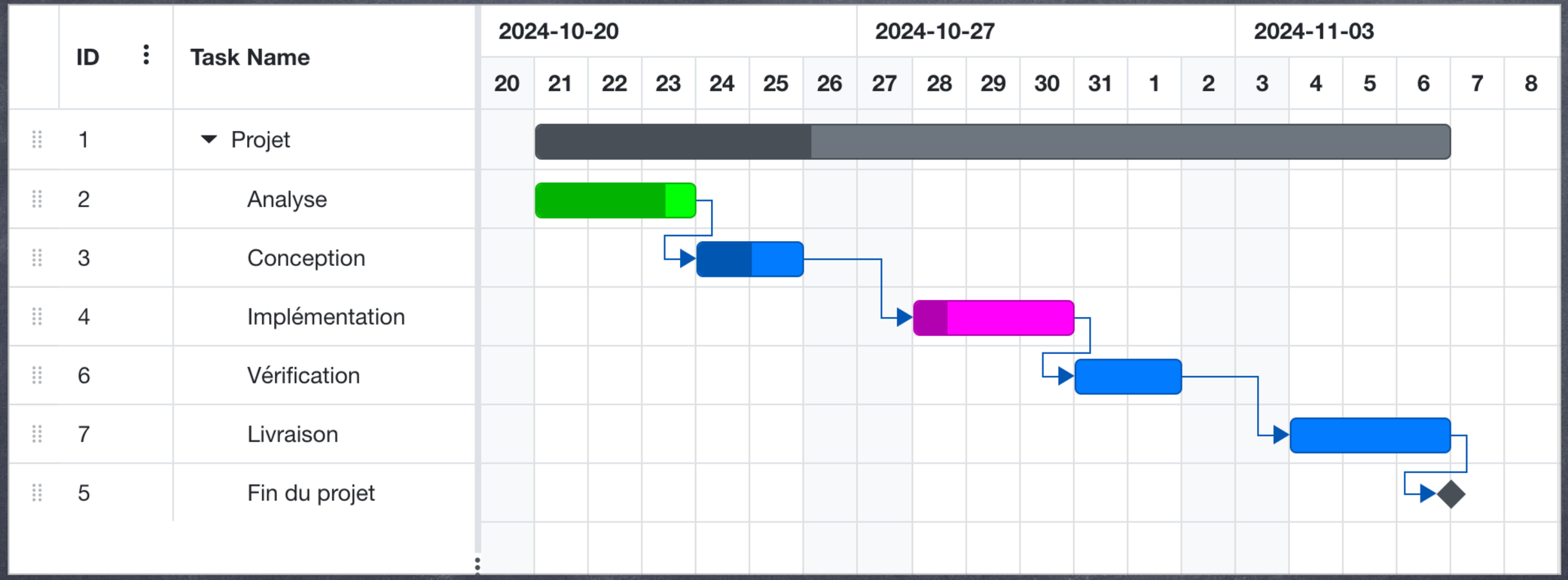
Le diagramme de Gantt

- ✓ Concept qui vient du Taylorisme
- ✓ Les tâches en abscisse (axe des x)
- ✓ Le temps en ordonnée (axe des y)
- ✓ Les tâches ont des dépendances
- ✓ Les tâches sont associées à des ressources (humaines, matérielles, financières)
- ✓ On peut définir des jalons (milestones)

Objectifs

- ✓ Planifier les tâches à exécuter
- ✓ Offrir une vue d'ensemble du projet
- ✓ Communiquer sur la planification du projet
- ✓ Identifier les dépendances entre les tâches
- ✓ Suivre l'avancement du projet

Un exemple



https://www.onlinegantt.com

Comment faire

- ✓ Lister les tâches
- ✓ Lister les ressources
- ✓ Assigner les tâches
- ✓ Planifier le projet et son calendrier
  Il est possible de partir de la fin (rétro-planning)
  pour savoir quand commencer le projet
- ✓ Corréler les tâches
- ✓ Fixer des jalons

Exercice

Vous planifiez un projet de rénovation de bureau avec les tâches suivantes :

1. Planification du design - 3 jours
2. Commande de fournitures - 2 jours (commence après la tâche 1)
3. Préparation du site - 2 jours (commence après la tâche 1)
4. Installation électrique - 4 jours (commence après la tâche 3)
5. Peinture - 3 jours (commence après la tâche 4)
6. Installation des meubles - 2 jours (commence après la tâche 5)
7. Nettoyage final et inspection - 1 jour (commence après la tâche 6)

Créez un diagramme de Gantt représentant ces tâches.
Identifiez les tâches qui peuvent être exécutées en parallèle pour optimiser le temps total de projet.
Indiquez la durée totale du projet avant et après optimisation.

Etude de cas rétrospective

Dans votre expérience personnelle étudiez un projet que vous avez réalisé (pas forcément informatique)
Appliquez la méthode Waterfall à ce projet et partagez avec le groupe

Etude de cas fictive

Une petite entreprise, "ModeExpress", souhaite développer un site web e-commerce pour vendre ses produits de mode en ligne. L'entreprise veut une plateforme robuste et fiable qui permet à ses clients de consulter ses produits, de passer des commandes en ligne, et de gérer leur profil client. Elle a décidé d'utiliser la méthode Waterfall pour mener à bien ce projet.

Lister ce que vous feriez à chaque phase :

Phase 1 : Liste des exigences fonctionnelles
Phase 2 : Des idées sur l'architecture technique de la solution
Phase 3 : Liste des modules développés
Phase 4 : Liste des tests réalisés
Phase 5 : Liste des actions misent en oeuvre
Phase 6 : Liste des possibles opérations que vous seriez amené à effectuer

Vous pouvez travailler en groupe de deux ou trois personnes et vous ferez une restitution orale au groupe, prévoyez un support de présentation

Débat

Quels sont les avantages et inconvénients de cette méthode ?

Divisez-vous en deux groupes, l'un pro Waterfall et l'autre anti Waterfall, listez vos arguments et restituez au groupe

Pro

- ✓ Le projet est bien documenté
- ✓ Il est possible d'accueillir de nouveaux membres
  voire de changer l'équipe de dev sans mettre en péril le projet
- ✓ Il est facile de suivre l'avancement du projet
- ✓ La méthode est très simple, inutile de former les équipes
- ✓ Contrôle rigoureux à chaque phase

Anti

- ✓ Rigidité, il n'est pas prévu de revenir en arrière
- ✓ Tests tardifs
- ✓ Manque d'interaction avec le client après la première phase
- ✓ Pas du tout adapté aux projets évolutifs.
  On suppose que le client sait ce qu'il veut et que les besoins n'évolueront pas

Conditions pour l'utiliser

- ✓ **Clarté des exigences dès le début** : Les projets avec des spécifications claires et stables qui n'évolueront pas pendant le développement.
- ✓ **Risque de modification limité** : Les projets où il est essentiel de suivre un plan bien défini et où les changements imprévus sont coûteux et difficiles à intégrer.
- ✓ **Livrables bien définis à chaque phase** : Les projets nécessitant des livrables spécifiques et formalisés à chaque étape du cycle de vie.
- ✓ **Délais et budgets fixes** : Les projets où le respect des échéances et du budget est critique.
- ✓ **Industries ou contextes réglementés** : Les projets soumis à des règles strictes nécessitant une documentation formelle à chaque étape.
- ✓ **Périmètre connu et maîtrisé** : Les projets "classiques" sur lesquels on peut se baser sur des expériences passées
- ✓ **Périmètre restreint** : Les projets de petite ou moyenne importance où les éventuels petits retards seront épongés par une surcharge de travail