I. Introduction

La méthode Waterfall, également connue sous le nom de modèle en cascade, est l'une des approches classiques de gestion de projets et de développement de logiciels. Elle se caractérise par une séquence linéaire et séquentielle d'étapes, où chaque phase du projet doit être complétée avant de passer à la suivante. Le nom Waterfall (cascade en français) fait référence à la façon dont le projet coule de manière fluide d'une étape à l'autre, comme une cascade d'eau.

Dans le modèle Waterfall, les phases du projet, telles que l'analyse des exigences, la conception, le développement, les tests, la validation et vérification et le déploiement, sont planifiées et exécutées de manière ordonnée. Chaque étape produit des livrables spécifiques, et il y a peu de retour en arrière une fois qu'une phase est terminée. Cette méthodologie est particulièrement adaptée aux projets avec des exigences bien définies dès le départ, ce qui est le cas de votre projet.

II. Avantages de la méthode Waterfall

1) Clarté et Structuration :

Clarté des étapes : Les phases du modèle Waterfall sont bien définies et ont des objectifs précis. Cela signifie que les parties prenantes, y compris les développeurs, les gestionnaires de projet et les clients, ont une compréhension claire de ce qui doit être accompli à chaque étape du projet.

Documentation solide : Chaque phase génère une documentation détaillée, y compris des spécifications, des plans de conception, des rapports de tests, etc. Cette documentation est essentielle pour assurer la traçabilité des exigences, faciliter la maintenance à long terme et permettre une communication claire entre les équipes.

2) Contrôle et Gestion :

Contrôle du projet : Waterfall permet un contrôle strict du projet. Chaque phase a des jalons clairement définis et des critères d'acceptation. Cela permet de mesurer le progrès par rapport aux objectifs et d'identifier rapidement les retards ou les problèmes potentiels.

Gestion du changement : Les modifications sont gérées de manière plus efficace dans Waterfall par rapport à certains modèles agiles. Puisque chaque phase doit être complétée avant de passer à la suivante, les changements sont identifiés tôt dans le processus et peuvent être traités avant qu'ils ne deviennent coûteux à mettre en œuvre.

3) Stabilité et Fiabilité :

Stabilité du code : Étant donné que la conception est complète avant le développement, il y a moins de risques de changements de dernière minute qui pourraient affecter la stabilité du code. Cela conduit souvent à des systèmes plus stables et fiables.

Adapté aux exigences définies : Waterfall est particulièrement adapté aux projets où les exigences sont bien comprises et peu susceptibles de changer significativement au cours du projet. Il fonctionne bien lorsque les objectifs sont clairement définis dès le départ.

4) Simplification et Formation :

Formation simplifiée : Les équipes nouvelles ou les membres de l'équipe qui rejoignent le projet en cours de route peuvent trouver plus facile de comprendre et de s'adapter au processus Waterfall en raison de sa structure séquentielle.

III. Les 6 phases du modèle Waterfall

1) Analyse des Exigences :

Dans cette phase initiale, l'équipe de projet travaille en étroite collaboration avec les parties prenantes, telles que les clients et les utilisateurs finaux, pour comprendre et documenter de manière exhaustive les exigences du projet. Cela comprend la collecte des besoins fonctionnels et non fonctionnels.

2) Conception :

Une fois les exigences clairement définies, l'équipe passe à la phase de conception. Pendant cette étape, les concepteurs et les architectes logiciels créent une architecture globale du système. Ils conçoivent également les détails de chaque composant logiciel, définissent les interfaces et établissent un plan de développement.

3) Développement :

Après la conception, les développeurs commencent à écrire le code source en suivant les spécifications de conception. Chaque composant logiciel est développé et testé individuellement. L'objectif est de mettre en œuvre le système conformément aux spécifications établies lors de la phase de conception.

4) Tests :

Dans cette phase, le système est soumis à une série de tests pour garantir qu'il fonctionne conformément aux exigences initiales. Les tests peuvent inclure des tests unitaires, des tests d'intégration, des tests de système et des tests d'acceptation. Les anomalies sont identifiées et corrigées.

5) Déploiement :

Une fois que le système a été testé, validé et vérifié avec succès, il est déployé dans l'environnement de production. Cela peut inclure l'installation du logiciel sur les serveurs du client, la configuration, la migration de données, et la formation des utilisateurs finaux.

6) Maintenance :

L'étape de maintenance dans la méthode Waterfall est essentielle pour garantir que le système reste opérationnel, sécurisé et efficace à long terme. Elle permet également d'assurer que le logiciel évolue en fonction des besoins changeants de l'entreprise et des commentaires des utilisateurs. C'est une phase continue qui peut durer aussi longtemps que le système est en service, et elle peut être cruciale pour maintenir la valeur du système sur le long terme.

IV. Livraisons à chaque phase et leur importance

- Analyse des Exigences

Cette première étape tient une importance capitale pour garantir le succès du projet. Il s'agit notamment d'**analyser et de spécifier les besoins du client** afin de répondre convenablement aux exigences. Cette phase découle souvent sur **l'établissement d'un document des exigences** clair et précis. De cette manière, chaque partie prenante prend en compte sa responsabilité vis-à-vis du développement du projet.

Le document des exigences est la livrable central de cette phase. Il remplit diverses fonctions cruciales pour le développement d'un système ou d'un produit logiciel. Son objectif principal est de définir de manière claire et exhaustive les besoins et les attentes du client ou de l'utilisateur final.

Il s'agit d'un document structuré, organisé en sections ou chapitres, pour faciliter la gestion des exigences. Chaque exigence est décrite en détail, y compris des déclarations précises et, si nécessaire, des exemples d'utilisation pour une meilleure compréhension. De plus, ce document permet de prioriser les exigences, indiquant ainsi leur niveau d'importance. Il peut également spécifier des exigences dérivées, résultant de la décomposition des exigences de haut niveau en détails plus spécifiques. La traçabilité des exigences est assurée, liant chaque exigence à d'autres composants du projet pour permettre un suivi tout au long du cycle de développement. Enfin, l'approbation formelle des parties prenantes, notamment le client, est essentielle pour valider le cahier des charges, garantissant ainsi un consensus sur les besoins et les attentes du projet.

C’est un contrat implicite entre l'équipe de développement et le client, car il définit ce qui doit être construit. Une fois le document des exigences approuvé, il sert de référence pour le reste du projet, guidant la conception, le développement, les tests et la validation pour s'assurer que le système est construit conformément aux spécifications convenues.

- Conception

Pour garantir la réussite de cette phase, il vaut mieux la diviser en 2 sous-étapes interdépendantes. Il s’agit de la **conception logique** et de la **conception physique**. D’un côté, la conception logique permet à l’équipe projet de déterminer et de **théoriser toutes les solutions réalisables**. De l’autre côté, la conception physique permet de **transformer ces informations et schémas conceptuels en spécifications concrètes.**

Le plan de conception est une livrable essentielle de la phase de conception. Il décrit en détail la manière dont l'équipe de développement abordera la phase de conception d'un projet logiciel. Ce plan vise à assurer une compréhension claire de la portée de la conception et des démarches qui seront entreprises pour la réaliser.

Dans ce document, l'équipe de conception est identifiée, avec une mention des rôles clés tels que les architectes logiciels, les concepteurs d'interface utilisateur et les experts en base de données. La méthodologie de conception qui sera suivie est également définie, tout comme les outils de conception et de modélisation qui seront utilisés, tels que les diagrammes UML. L'architecture globale du système est décrite, y compris l'approche architecturale choisie, qu'il s'agisse d'une architecture en couches, orientée services (SOA), ou autre.

Le plan de conception spécifie quels modèles de conception seront créés pour représenter la structure et le comportement du système. Ces modèles peuvent inclure des diagrammes de classe, des diagrammes de séquence, des diagrammes de flux de données, etc. Si le système comprend des interfaces utilisateur, le plan de conception détaille comment elles seront conçues en termes d'apparence, d'ergonomie et de flux de travail. Pour les systèmes comportant une base de données, le plan précise la structure de la base de données, notamment les tables, les relations, les contraintes et les index.

Les aspects de sécurité sont également abordés, avec une identification des vulnérabilités potentielles et des mesures de sécurité à mettre en place. La performance est prise en compte, avec des considérations telles que l'optimisation de la base de données, la gestion de la mémoire et la gestion de la charge.

Il spécifie également comment les tests de conception seront planifiés et exécutés pour garantir que la conception répond aux exigences. Enfin, le document peut inclure un calendrier prévisionnel indiquant les dates clés de la phase de conception, ainsi qu'une estimation des coûts associés à cette phase, si nécessaire.

Le plan de conception du système est essentiel pour assurer que la conception technique répond aux besoins et aux exigences définis lors de la phase d'analyse des exigences, tout en fournissant une feuille de route claire pour l'équipe de développement. Une fois approuvé, il guide la mise en œuvre technique du système.

- Développement

Au moment où la conception est validée, il est temps de passer à la **mise en œuvre technique**. Il s’agit certainement de la phase la plus courte de la méthode Waterfall. En effet, **les développeurs procèdent à la fabrication** en suivant une documentation bien définie. Les exigences et spécifications relevées favorisent **un processus de développement approfondi et rapide**.

La phase de développement dans la méthodologie Waterfall génère plusieurs éléments essentiels pour la création du système ou du produit logiciel. Tout d'abord, le "Code Source" constitue le cœur du logiciel, mettant en œuvre de manière concrète les spécifications techniques établies au cours de la phase de conception. Rédigé par les développeurs, il revêt une importance majeure et doit être accompagné d'une documentation détaillée pour faciliter la maintenance future.

En outre, la "Documentation du Code" revêt une importance cruciale. Elle va au-delà du code en lui-même, en expliquant le fonctionnement des fonctions, des classes, des méthodes, des variables, et en fournissant des commentaires utiles

pour que d'autres développeurs puissent comprendre et collaborer efficacement sur le code.

Si le système intègre des "Interfaces Utilisateur", il est impératif de livrer les écrans, les formulaires, et les fonctionnalités visuelles développés, en incluant également les éléments de conception tels que les maquettes et les éléments d'interface utilisateur.

Lorsqu'une "Base de Données" est utilisée, les scripts de création de la base de données, les requêtes SQL, et les schémas doivent être fournis pour permettre une configuration adéquate de la base de données.

Les Tests Unitaires constituent une étape clé, impliquant de petits tests automatisés qui vérifient le bon fonctionnement de chaque composant logiciel de manière isolée. Il est essentiel de documenter les résultats de ces tests unitaires.

Les "Rapports de Tests" sont également cruciaux, détaillant les résultats des tests effectués pour garantir que le logiciel est en conformité avec les spécifications. Ces rapports englobent les résultats des tests liés aux exigences fonctionnelles et non fonctionnelles.

Si le logiciel requiert une "Documentation d'Utilisation", un "Manuel d'Utilisation" doit être élaboré pour guider les utilisateurs sur la manière d'utiliser le logiciel, ses fonctionnalités, et pour fournir des instructions étape par étape.

Enfin, un "Rapport de Développement" offre une vue d'ensemble de la phase de développement, mettant en lumière les défis rencontrés, les modifications apportées par rapport au plan initial, et la conformité aux délais et au budget définis pour le projet. Ces éléments garantissent la qualité et la cohérence du système développé avant sa livraison à l’équipe de tests.

- Tests

La phase de test dans la méthode Waterfall génère un ensemble de livrables cruciaux pour assurer la qualité du système avant son déploiement. Tout d'abord, le "**Plan de Test**" est élaboré en amont et détaille la stratégie de test, les objectifs, les ressources requises, les scénarios de test et les critères d'acceptation. Il constitue une feuille de route essentielle pour l'ensemble de la phase de test.

En parallèle, les "**Cas de Test**" sont élaborés pour décrire les procédures et les données nécessaires à l'exécution de tests spécifiques. Chaque cas de test a pour objectif de valider une fonctionnalité ou une exigence particulière du système.

Les scénarios de test viennent compléter l'ensemble en décrivant des séquences d'actions qui simulent des utilisations réelles du système. Ils peuvent inclure des cas de test individuels ainsi que des combinaisons de cas de test pour vérifier la cohérence du système dans diverses situations.

Les Données de Test sont soigneusement préparées pour servir d'entrées spécifiques lors de l'exécution des cas de test et des scénarios de test. Elles couvrent différents aspects du système, incluant les situations normales et exceptionnelles.

Les résultats des tests sont consignés dans les "Rapports de Test", documentant ainsi les problèmes identifiés, les anomalies détectées, ainsi que les réussites. Ces rapports fournissent une preuve tangible quant à la conformité du système aux exigences spécifiées.

En cas de non-conformité, un "Rapport de Non-conformité" est établi pour détailler les problèmes et les défauts relevés lors des tests. Ce rapport indique leur gravité et leur criticité, permettant ainsi à l'équipe de développement de mettre en œuvre les mesures correctives nécessaires.

Enfin, une fois que les tests sont menés avec succès et que le système répond aux critères d'acceptation préalablement définis, un "Rapport de Validation et d'Acceptation" est créé. Ce rapport est soumis à l'approbation des parties prenantes, confirmant ainsi que le système est prêt pour le déploiement. Ces livrables de la phase de test garantissent la qualité du système et fournissent une documentation essentielle pour la phase de maintenance future.

- Déploiement

Au cours de cette phase, les bugs identifiés pendant la phase de test sont corrigés. Il est maintenant temps de **mettre le logiciel ou projet web à la disposition du client final**. Pour les projets de développement de produit et autres, cette étape consiste particulièrement à réaliser la phase de lancement et à **déployer tous les livrables**.

La phase de déploiement, dans la méthodologie Waterfall, engendre un ensemble de livrables cruciaux pour garantir un déploiement réussi du système ou du produit logiciel. Tout d'abord, le "Système Déployé" constitue le résultat majeur de cette phase, représentant la version complète et fonctionnelle du logiciel, prête à être utilisée par les utilisateurs finaux.

De plus, la "Documentation de Déploiement" joue un rôle clé en expliquant en détail la manière d'installer et de configurer le logiciel dans l'environnement de production. Elle fournit des instructions exhaustives pour garantir un déploiement correct.

En parallèle, le "Rapport de Déploiement" offre un récapitulatif complet de la manière dont la phase de déploiement s'est déroulée. Il inclut des informations sur les problèmes rencontrés, les délais respectés, et d'éventuelles déviations par rapport au plan initial.

Enfin, l'"Acceptation Finale" du système est formellement réalisée par les parties prenantes. Cette étape confirme que le logiciel déployé répond pleinement à leurs attentes et est prêt pour une utilisation en production. Dans l'ensemble, ces livrables sont essentiels pour garantir une transition fluide vers l'environnement de production, minimiser les risques, et assurer que le logiciel fonctionne de manière fiable pour les utilisateurs finaux.

- Maintenance

La **phase de maintenance** débute dès que le projet se déploie sur le marché. Pour les projets de développement de logiciels, les nouveaux bugs de fonctionnalités sont courants. Le développeur doit continuellement **proposer une nouvelle version pour mettre à jour le logiciel**. Concernant les autres secteurs d'activités, l'équipe de production applique des correctifs afin de solutionner la problématique identifiée.

Dans la phase de maintenance, plusieurs livrables jouent un rôle essentiel dans le maintien de la qualité et de la fiabilité du système ou du produit logiciel tout au long de sa durée de vie opérationnelle. Tout d'abord, les "Demandes de Maintenance" sont soumises par les utilisateurs finaux ou les parties prenantes pour signaler des problèmes, des anomalies ou des besoins d'amélioration du système. Ces demandes servent de base pour l'identification des travaux de maintenance à effectuer.

Ensuite, le "Plan de Maintenance" définit la stratégie globale de gestion de la maintenance, y compris la manière dont les demandes de maintenance seront gérées, priorisées et planifiées. Il établit également les responsabilités au sein de l'équipe de maintenance.

Les "Ordres de Travail" sont créés pour chaque tâche de maintenance à réaliser. Ils spécifient les détails de la tâche, tels que la description du problème, les ressources nécessaires, les délais et les responsables de la tâche.

Les "Rapports de Maintenance" documentent les activités de maintenance effectuées, y compris les détails des corrections apportées, des mises à jour ou des améliorations réalisées. Ces rapports servent de référence pour suivre l'historique des interventions de maintenance.

En cas de modifications importantes apportées au système lors de la maintenance, des "**Tests de Maintenance**" sont effectués pour s'assurer que les changements n'ont pas introduit de nouveaux problèmes ou de nouvelles erreurs.

Une fois que les travaux de maintenance sont terminés, une "**Validation de Maintenance**" est effectuée pour confirmer que les problèmes signalés ont été résolus conformément aux exigences spécifiées.

La "**Documentation de Maintenance**" est mise à jour pour refléter les changements apportés au système lors de la maintenance. Cela peut inclure des mises à jour du manuel d'utilisation, des guides de dépannage, ou d'autres documents pertinents.

Un "**Suivi des Demandes de Maintenance**" est maintenu de manière rigoureuse pour s'assurer que toutes les demandes sont traitées en temps opportun et que les utilisateurs sont informés de l'état de leurs demandes.

Enfin, un "**Rapport de Suivi de la Maintenance**" peut être généré pour fournir une vue d'ensemble des activités de maintenance sur une période donnée. Il inclut des informations sur les performances de l'équipe de maintenance et les délais de résolution. Ces livrables sont essentiels pour maintenir la qualité, la fiabilité et la pertinence du système ou du produit logiciel tout au long de son cycle de vie opérationnel, tout en répondant aux besoins changeants des utilisateurs pour assurer une expérience utilisateur optimale.

Sources :

- https://le-consultant-digital.com/gestion-de-projet/methode-waterfall
- https://reussir-son-management.com/la-methode-waterfall/
- https://bubbleplan.net/pedagogie-projet/methode-waterfall/etapes-methode-waterfall
- https://blog-gestion-de-projet.com/modele-en-cascade/
- https://www.bitstudios.com/blog/waterfall-software-development-model/
- https://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/
- https://www.researchgate.net/publication/30498645_The_Waterfall_Model_in_Large-Scale_Development