Conception, Projet

Cours 4 - Développement logiciel

Table des matières

Le besoin de qualité

Le besoin de qualité dans le développement de logiciels est crucial pour satisfaire les attentes des utilisateurs finaux. Ce besoin fondamental est pourtant ardu à assurer.

La qualité d'un logiciel est définie comme l'ensemble des mesures qui permettent d'évaluer si un logiciel répond aux exigences spécifiques des utilisateurs finaux, tout en respectant les normes internes aux entreprises, et les normes de l'industrie.

Les avantages d'un logiciel de qualité sont nombreux. Un logiciel de qualité offre une meilleure expérience utilisateur, améliore la satisfaction du client, augmente la productivité, réduit les coûts de maintenance, et augmente la rentabilité globale de l'entreprise sur le long terme.

Le besoin de qualité

La qualité est un élément essentiel tout au long du cycle de vie du développement de logiciels, de la planification à la mise en œuvre en passant par les tests et la maintenance. Les équipes de développement de logiciels doivent s'assurer que la qualité est contrôlée dans chaque étape du processus de développement.

Cela peut être atteint en adoptant des pratiques telles que la gestion de la qualité, les tests de qualité, la documentation des exigences, la collaboration étroite avec les clients et les parties prenantes, et l'utilisation de technologies et d'outils de pointe pour garantir la conformité des produits livrés.

Le besoin de qualité

Il est également important de noter que le besoin de qualité doit être équilibré avec les autres exigences du projet, telles que les délais et les coûts. Une bonne planification et une bonne gestion du projet peuvent aider à atteindre cet équilibre et à s'assurer que le système logiciel développé répond aux contraintes du projet.

Au fil du temps, l'industrie de l'informatique a connu beaucoup d'effervescence. Il y a eu des succès majeurs, mais aussi de nombreux échecs qui ont poussé l'industrie à se remettre en question.

Un échec connu est le désastre du système de contrôle des bagages de l'aéroport de Denver (Denver International Airport, DIA) en 1995.

Source : `https://www5.in.tum.de/~huckle/DIABaggage.pdf`

Le besoin de qualité - Aéroport de Denver

L'aéroport de Denver avait prévu un système de contrôle des bagages entièrement automatisé, révolutionnaire et ultramoderne pour gérer le flux des bagages entre les terminaux et les avions.

Ce système ambitieux devait réduire les délais de manutention des bagages, les erreurs de manipulation et les coûts de main-d'œuvre. Cependant, le projet a été confronté à de nombreux problèmes dès le début.

Le besoin de qualité - Aéroport de Denver

Parmi les problèmes rencontrés, on peut citer :

- La planification et la gestion du projet étaient insuffisantes. Les délais et les coûts ont rapidement dérapé, en partie à cause de l'absence de coordination entre les différentes parties prenantes et de la complexité du système.
- Les exigences n'étaient pas clairement définies et ont subi de nombreux changements en cours de projet, ce qui a entraîné des problèmes de conception, des retards et une augmentation des coûts.
- Les tests et la validation du système ont été insuffisants, et de nombreux problèmes techniques sont apparus lors de la mise en service. Par exemple, les tapis roulants ne fonctionnaient pas correctement, les bagages étaient endommagés ou perdus, et les chariots de transport se coinçaient fréquemment.

Le besoin de qualité - Aéroport de Denver

Au final, le système de contrôle des bagages de l'aéroport de Denver n'a jamais fonctionné comme prévu.

L'aéroport a finalement été ouvert avec un retard d'environ seize mois et un coût supplémentaire de plusieurs centaines de millions de dollars.

Le système automatisé a été partiellement abandonné au profit de systèmes conventionnels de manutention des bagages.

Cet échec majeur a servi de leçon à l'industrie du logiciel et a souligné l'importance d'une planification et d'une gestion de projet rigoureuses, d'une définition claire des exigences, d'une communication efficace entre les parties prenantes et d'un processus de test et de validation approfondi pour garantir la réussite des projets logiciels.

L'échec d'un projet

Le développement de logiciels peut parfois se solder par un échec en raison de divers facteurs, malgré les efforts consacrés au projet. Voici quelques raisons courantes pour lesquelles un projet logiciel peut échouer :

- Dépassement de budgets
- Dépassement de temps ou inadéquation avec les besoins du marché
- Le logiciel ne satisfait pas les besoins définis du client
- Moins bonne qualité que prévue
- Les performances ne correspondent pas aux attentes
- Le logiciel est trop difficile à utiliser

L'échec d'un projet - Dépassement de budgets

Les projets logiciels peuvent dépasser leur budget en raison d'une mauvaise estimation des coûts, de changements imprévus dans les exigences ou de problèmes techniques non anticipés.

Cela peut mettre en péril la viabilité financière du projet et affecter les autres activités de l'entreprise.

L'échec d'un projet - Dépassement de temps ou inadéquation avec les besoins du marché

Les retards dans le développement du logiciel peuvent entraîner une mise sur le marché tardive, ce qui peut réduire son attractivité et sa compétitivité.

De plus, le marché évolue rapidement et un logiciel peut devenir obsolète ou inadapté aux besoins des clients s'il n'est pas livré à temps.

L'échec d'un projet - Le logiciel ne satisfait pas les besoins définis du client

Un logiciel peut ne pas répondre aux attentes du client en raison d'une mauvaise compréhension des besoins, d'une communication insuffisante entre les parties prenantes ou d'une conception inadéquate.

Cela peut entraîner une insatisfaction du client et un échec commercial.

L'échec d'un projet - Moins bonne qualité que prévue

Si le logiciel présente de nombreux bugs, des problèmes de stabilité ou des défauts de conception, il peut ne pas atteindre le niveau de qualité attendu.

Cela peut affecter l'expérience utilisateur et nuire à la réputation de l'entreprise.

L'échec d'un projet - Les performances ne correspondent pas aux attentes

Un logiciel peut ne pas offrir les performances attendues en termes de rapidité, d'efficacité ou de fiabilité, ce qui peut limiter son utilité et sa valeur pour les clients et les utilisateurs.

L'échec d'un projet - Le logiciel est trop difficile à utiliser

Si un logiciel est mal conçu ou présente une interface utilisateur complexe et peu intuitive, il peut être difficile à utiliser pour les clients et les utilisateurs finaux.

Cela peut réduire l'adoption du logiciel, nuire à la satisfaction des clients et entraîner un échec du projet.

Causes fréquentes

Il n'y a pas que les erreurs de programmation qui vont causer l'échec d'un projet. De nos jours, les erreurs de programmation ne sont pas des causes majeures d'échec d'un projet.

Ainsi, plusieurs outils, méthodologies, techniques, principes, et politiques ont été développés pour assurer le succès d'un projet et limiter ses risques.

Causes fréquentes

Voici un exemple de causes majeurs pouvant mener à l'échec d'un projet :

- Projet non réaliste
- Gestion de projet de mauvaise qualité
- Mauvaise estimation des ressources
- Mauvaise définition des besoins du système
- Risques non gérés
- Mauvaise communication entre les clients, développeurs, utilisateurs
- Inaptitude à gérer la complexité du projet
- Mauvaise méthodologie de conception
- Mauvais outils de développement
- Mauvaise méthodologie de test
- Mauvaise couverture des tests
- Processus de développement inapproprié
- Coûts de maintenance trop élevés

La norme ISO 12207

La norme ISO/IEC 12207 est une norme internationale pour le cycle de vie du logiciel qui décrit les processus, activités et tâches nécessaires à la conception, au développement, à la livraison, et à la maintenance des logiciels. Elle est applicable à tous les types de logiciels, quel que soit leur domaine d'application.

La norme ISO 12207 a été publiée pour la première fois en 1995 et a depuis été révisée plusieurs fois pour s'adapter aux évolutions technologiques et aux pratiques de développement de logiciels. Elle est largement utilisée dans l'industrie du logiciel pour garantir la qualité et la fiabilité des produits logiciels.

Soucres :
`https://fr.wikipedia.org/wiki/ISO/CEI_12207`
`https://www.iso.org/standard/63712.html`

La norme ISO 12207

Voici la liste des acteurs :

- Fournisseur (Maître d'œuvre)
- Client (Maître d'ouvrage)

Puis des processus globaux :

- Processus de base
- Processus de support
- Processus organisationnels

Fournisseur (Maître d'œuvre)

Le fournisseur est l'organisation ou l'entité responsable du développement, de la livraison et, éventuellement, de la maintenance du logiciel ou du système.

En tant que maître d'œuvre, le fournisseur est responsable de la réalisation technique et opérationnelle du projet.

Le fournisseur doit respecter les exigences spécifiées par le client, les normes de qualité et les contraintes de coûts et de délais.

Fournisseur (Maître d'œuvre)

Les responsabilités du fournisseur incluent :

- Analyser et comprendre les besoins et les exigences du client.
- Proposer une solution technique et un devis.
- Négocier et conclure des contrats avec le client.
- Concevoir, développer, tester et valider le logiciel.
- Livrer le logiciel et la documentation associée.
- Former les utilisateurs et assurer le soutien technique.
- Assurer la maintenance et l'évolution du logiciel, le cas échéant.

Client (Maître d'ouvrage)

Le client est l'organisation ou l'entité qui acquiert et utilise le logiciel ou le système développé par le fournisseur.

En tant que maître d'ouvrage, le client est responsable de la définition des besoins, des exigences et des objectifs du projet.

Il doit également assurer la supervision et le contrôle du projet pour garantir sa réussite.

Client (Maître d'ouvrage)

Les responsabilités du client incluent :

- Définir et exprimer les besoins et les exigences.
- Sélectionner un fournisseur et négocier les contrats.
- Collaborer avec le fournisseur tout au long du projet.
- Participer aux revues conjointes et aux audits.
- Vérifier et valider que le logiciel répond aux exigences spécifiées.
- Accepter et déployer le logiciel dans l'organisation.
- Assurer l'exploitation et l'utilisation du logiciel et des services associés.
- Communiquer les problèmes et les besoins d'évolution au fournisseur.

Processus de base

Ce qui amène la velur directe au client :

- Acquisition pour l'organisme lui-même
- Activités du fournisseur
- Développement du logiciel
- Exploitation du système informatique et des services associés
- Maintenance du logiciel

Processus de base - Acquisition pour l'organisme lui-même

L'acquisition concerne le processus par lequel un organisme (le client) spécifie, sélectionne et acquiert un logiciel ou un système auprès d'un fournisseur.

Ce processus inclut la définition des besoins, l'établissement des critères de sélection, la communication avec les fournisseurs, la négociation des contrats, la réception et l'acceptation du logiciel.

L'objectif de ce processus est de s'assurer que le client obtient un logiciel qui répond à ses exigences et respecte les normes de qualité.

Processus de base - Activités du fournisseur

Ce processus traite des responsabilités et des activités du fournisseur lors de la livraison d'un logiciel ou d'un système au client.

Il inclut la réponse aux appels d'offres, la négociation et la conclusion des contrats, le développement ou l'adaptation du logiciel selon les exigences du client, la livraison de la documentation et la formation, et le support après-vente.

Le but est de satisfaire les besoins du client tout en respectant les contraintes de coûts, de qualité et de délais.

Processus de base - Développement du logiciel

Le processus de développement du logiciel concerne toutes les activités liées à la création d'un logiciel. Il inclut la définition des besoins, l'analyse des exigences, la conception de l'architecture et des composants, l'implémentation (codage), les tests, l'intégration et la validation du logiciel.

Ce processus vise à produire un logiciel qui répond aux exigences du client et respecte les normes de qualité et de performance.

Processus de base - Exploitation du système informatique et des services associés

L'exploitation du système informatique et des services associés englobe toutes les activités liées à l'utilisation, à la gestion et à la maintenance du logiciel dans un environnement opérationnel.

Il comprend la planification, l'installation, la configuration, l'administration, le support aux utilisateurs, la surveillance, et le contrôle des performances du système.

L'objectif de ce processus est d'assurer que le logiciel fonctionne de manière efficace et fiable, tout en répondant aux exigences des utilisateurs.

Processus de base - Maintenance du logiciel

La maintenance du logiciel est le processus qui traite de la correction des défauts, de l'adaptation, et de l'amélioration du logiciel après sa mise en service.

Il inclut l'analyse des problèmes signalés par les utilisateurs, la correction des erreurs, la mise à jour des fonctionnalités pour répondre aux nouveaux besoins, et l'optimisation des performances.

Le but de la maintenance est de garantir que le logiciel continue de répondre aux exigences des utilisateurs et de s'adapter aux évolutions technologiques et aux besoins de l'entreprise.

Processus de support

Les processus de support sont l'ensemble des produits et processus qui vont permettre d'accomplir un projet informatique.
Ils comprennent les aspects suivants :

- Documentation du logiciel et de son cycle de vie
- Gestion de configuration
- Assurance qualité (PAQ) avec les revues, audit et vérification
- Vérification
- Validation
- Revue conjointe
- Audit pour la vérification de la conformité avec les exigences
- Résolution de problèmes et de non-conformités en cours de développement et à l'exploitation

Processus de support - Documentation du logiciel et de son cycle de vie

Ce processus concerne la création, la maintenance et la diffusion de la documentation tout au long du cycle de vie du logiciel.

La documentation doit couvrir les besoins, la conception, les spécifications, les tests, la maintenance et les autres aspects pertinents du logiciel.

Une bonne documentation facilite la communication entre les parties prenantes, améliore la traçabilité et soutient la maintenance et l'évolution du logiciel.

Processus de support - Gestion de configuration

La gestion de configuration est un processus qui assure le contrôle et la traçabilité des modifications apportées au logiciel et à sa documentation tout au long du cycle de vie.

Il comprend l'identification et la gestion des versions, la gestion des modifications, la gestion des livraisons et la gestion des baselines (versions de référence).

Ce processus permet d'assurer la cohérence et la qualité du logiciel en évitant les problèmes liés aux modifications non maîtrisées.

Processus de support - Assurance qualité (PAQ) avec les revues, audit et vérification

L'assurance qualité est un processus visant à garantir que le logiciel répond aux exigences de qualité spécifiées.

Il inclut la planification de l'assurance qualité (PAQ), la mise en place de procédures et de normes, la réalisation de revues, d'audits et de vérifications pour évaluer la conformité du logiciel et l'identification et la résolution des problèmes de qualité.

Ce processus permet d'assurer la satisfaction des clients et des utilisateurs en termes de qualité et de fiabilité du logiciel.

Processus de support - Vérification

La vérification est un processus d'évaluation qui permet de s'assurer que les produits intermédiaires du cycle de vie du logiciel (tels que les documents de conception, le code source, etc.) sont conformes aux exigences spécifiées.

La vérification vise à détecter les erreurs et les écarts par rapport aux exigences le plus tôt possible dans le cycle de vie, afin de réduire les coûts et les délais de correction.

Processus de support - Validation

La validation est un processus qui consiste à évaluer si le logiciel final répond aux besoins et aux exigences des utilisateurs.

Elle inclut des activités telles que les tests d'acceptation, les tests de performance, les tests de sécurité et les tests de compatibilité.

La validation permet de confirmer que le logiciel est prêt à être déployé et utilisé dans un environnement réel.

Processus de support - Revue conjointe

La revue conjointe est un processus d'évaluation collaborative qui implique les différentes parties prenantes du projet (telles que les développeurs, les testeurs, les clients, et les utilisateurs) pour examiner et discuter des aspects clés du logiciel, tels que les exigences, la conception, les tests et les problèmes de qualité.

Les revues conjointes permettent d'identifier et de résoudre les problèmes, d'améliorer la communication et de faciliter la prise de décision.

Processus de support - Audit pour la vérification de la conformité avec les exigences

L'audit est un processus d'évaluation indépendante qui vise à vérifier la conformité du logiciel et de ses processus de développement et de maintenance avec les exigences spécifiées, les normes et les réglementations applicables.

Les audits peuvent être internes (réalisés par l'organisation elle-même) ou externes (réalisés par un organisme tiers).

Les audits permettent d'identifier les écarts, les risques, et les opportunités d'amélioration, et d'assurer la qualité et la conformité du logiciel.

Processus de support - Résolution de problèmes et de non-conformités en cours de développement et à l'exploitation

Ce processus traite de la détection, de l'analyse et de la résolution des problèmes et des non-conformités qui surviennent pendant le développement et l'exploitation du logiciel.

Il comprend la gestion des incidents, la gestion des problèmes, la gestion des demandes de changement et la gestion des correctifs.

La résolution des problèmes et des non-conformités contribue à améliorer la qualité et la fiabilité du logiciel, à réduire les coûts et les délais de maintenance, et à assurer la satisfaction des clients et des utilisateurs.

Processus organisationnels

Les processus organisationnels jouent un rôle essentiel dans la mise en œuvre réussie du cycle de vie des logiciels selon la norme ISO 12207.
Ils comprennent les aspects suivants :

- Management de toutes les activités, y compris la gestion de projet
- Infrastructure nécessaire à un processus
- Pilotage et amélioration du cycle de vie
- Formation du personnel

Processus organisationnels - Management de toutes les activités, y compris la gestion de projet

Ce processus concerne la planification, l'organisation, la direction et le contrôle de toutes les activités liées au développement, à la maintenance et à l'évolution du logiciel.

La gestion de projet inclut l'identification des objectifs, la définition des tâches et des responsabilités, l'estimation des ressources, l'élaboration du planning, le suivi des progrès et la gestion des risques et des problèmes.

Processus organisationnels - Infrastructure nécessaire à un processus

L'infrastructure englobe l'ensemble des ressources matérielles et logicielles, des services et des outils nécessaires pour mettre en œuvre, maintenir et améliorer les processus du cycle de vie des logiciels.

Elle peut inclure des serveurs, des postes de travail, des réseaux, des logiciels de développement, des systèmes de gestion de configuration, des outils de test et de validation, et des plateformes de collaboration et de communication.

Processus organisationnels - Pilotage et amélioration du cycle de vie

Ce processus vise à surveiller, évaluer et améliorer en continu les processus du cycle de vie des logiciels, afin d'optimiser leur efficacité, leur qualité et leur performance.

Il implique la collecte et l'analyse des données de processus, l'identification des écarts, des risques et des opportunités d'amélioration, et la mise en œuvre de mesures correctives et préventives.

Le pilotage et l'amélioration du cycle de vie contribuent à la maîtrise des coûts, des délais et des risques, et à la satisfaction des clients et des utilisateurs.

Processus organisationnels - Formation du personnel

La formation du personnel est un élément clé pour garantir la compétence et l'efficacité des équipes impliquées dans le développement, la maintenance et l'évolution des logiciels.

Elle englobe la formation initiale, la formation continue et le développement professionnel dans les domaines techniques, méthodologiques, managériaux et organisationnels.

La formation du personnel favorise l'acquisition et le partage des connaissances, des compétences et des meilleures pratiques, et stimule l'innovation et la performance des équipes.

Conclusion

Les avantages de la norme ISO/IEC 12207 sont nombreux.

Tout d'abord, elle fournit une méthode standard pour gérer le développement de logiciels, permettant aux organisations de planifier, suivre et contrôler les activités de développement de manière cohérente et efficace.

Elle permet également une meilleure communication entre les différents acteurs impliqués dans le développement de logiciels et facilite la collaboration.

De plus, la norme ISO 12207 définit les exigences de qualité pour chaque phase du cycle de vie du projet et du logiciel.

Conclusion

Cependant, la norme ISO/IEC 12207 peut également présenter des inconvénients.

Elle peut être complexe à mettre en œuvre, nécessitant une formation et des ressources importantes pour les organisations qui souhaitent l'utiliser.

Elle peut également être rigide, ne permettant pas toujours une adaptation facile aux besoins spécifiques des projets et des organisations. Enfin, elle peut être coûteuse à mettre en place, car elle nécessite souvent des audits et des certifications externes pour être pleinement appliquée.

Conclusion

En fin de compte, la décision d'utiliser la norme ISO 12207 dépendra des besoins et des objectifs spécifiques de chaque organisation.

Si la qualité et la standardisation sont des priorités élevées, cette norme peut être une excellente option. Cependant, si l'adaptabilité et la flexibilité sont plus importantes, il peut être préférable d'utiliser des méthodes plus agiles pour gérer le développement de logiciels.

Finalement, certains organismes vont adapter sur mesure cette norme, mais sans avoir une certaine expertise, cela va créer une gestion de projet boiteuse.

Les méthodes de gestion

Voici trois (3) méthodes de gestion répendues dans l'industrie :

- Gestion de projet en cascade
- Gestion de projet en itération
- Les méthodes AGILE

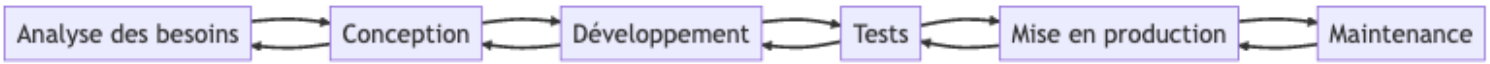
Gestion de projet en cascade

La gestion de projet en cascade (ou "Waterfall") est une approche séquentielle et linéaire du développement logiciel. Elle divise le projet en différentes phases, chacune devant être achevée avant de passer à la suivante.

Les phases typiques incluent l'analyse des besoins, la conception, le développement, les tests, la mise en production et la maintenance.

Cette méthode repose sur une planification rigoureuse et une documentation détaillée, mais elle peut être peu flexible face aux changements de besoins ou aux imprévus.

Gestion de projet en cascade



Gestion de projet en itération

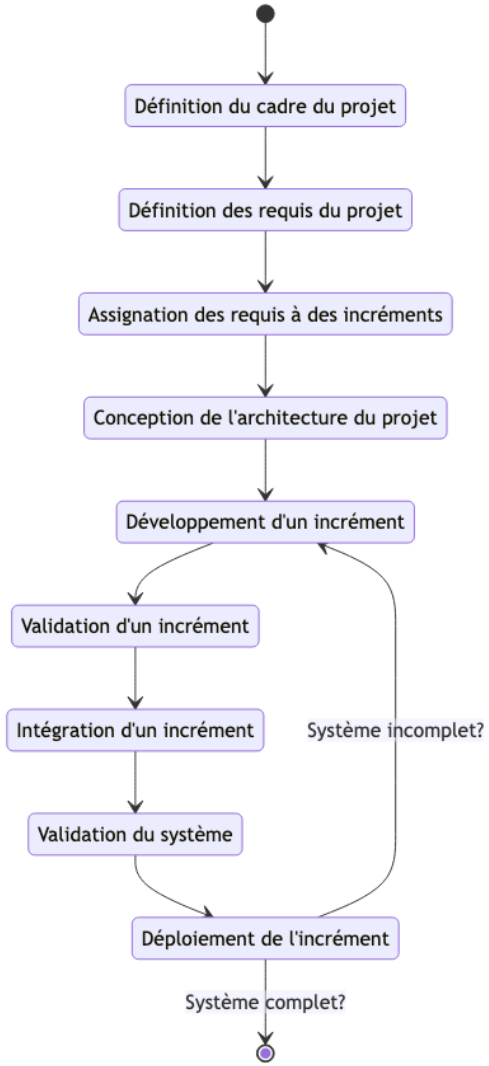
La gestion de projet en itération, proposée par [David Lorge Parnas], est une approche incrémentale du développement logiciel.

Au lieu de réaliser toutes les phases de manière linéaire, le projet est divisé en plusieurs itérations, chacune comprenant un ensemble de fonctionnalités à développer.

À la fin de chaque itération, un prototype ou une version partielle du logiciel est livrée, permettant d'obtenir des retours d'expérience et d'ajuster les besoins et les exigences avant de passer à l'itération suivante.

Cette méthode offre une meilleure flexibilité et une meilleure gestion des risques par rapport à l'approche en cascade.

Gestion de projet en itération



Les méthodes AGILE

Les méthodes Agile sont un ensemble de pratiques de gestion de projet qui visent à améliorer la flexibilité, l'efficacité et la collaboration dans le développement logiciel.

Elles mettent l'accent sur des cycles de développement courts et itératifs, une communication étroite entre les parties prenantes, l'adaptabilité aux changements de besoins et l'amélioration continue.

Parmi les méthodes Agile les plus connues, on peut citer Scrum, Kanban, eXtreme Programming (XP) et Feature-Driven Development (FDD).

Ces méthodes permettent aux équipes de développement de mieux répondre aux besoins des utilisateurs et d'assurer une livraison plus rapide et de meilleure qualité des logiciels.

Génie logiciel - Définition

Le génie logiciel est une discipline de l'ingénierie qui concerne la conception, le développement, la maintenance, les tests et l'évaluation des logiciels.

Il englobe un ensemble de techniques, méthodes et processus visant à créer des logiciels de haute qualité, fiables et efficaces, tout en répondant aux besoins et aux attentes des utilisateurs.

Génie logiciel - Logiciel

Un logiciel est un ensemble de programmes, de procédures et de documents qui permettent à un système informatique d'exécuter des tâches spécifiques.

Il est composé de code source, qui est écrit dans un langage de programmation, et de données, qui sont utilisées par les programmes pour fonctionner.

Génie logiciel - Métriques

Les métriques du génie logiciel sont des mesures quantitatives qui permettent d'évaluer la qualité, la performance, la complexité et d'autres aspects des logiciels et des processus de développement.

Elles sont essentielles pour suivre les progrès, identifier les problèmes potentiels et améliorer la qualité et l'efficacité du développement logiciel.

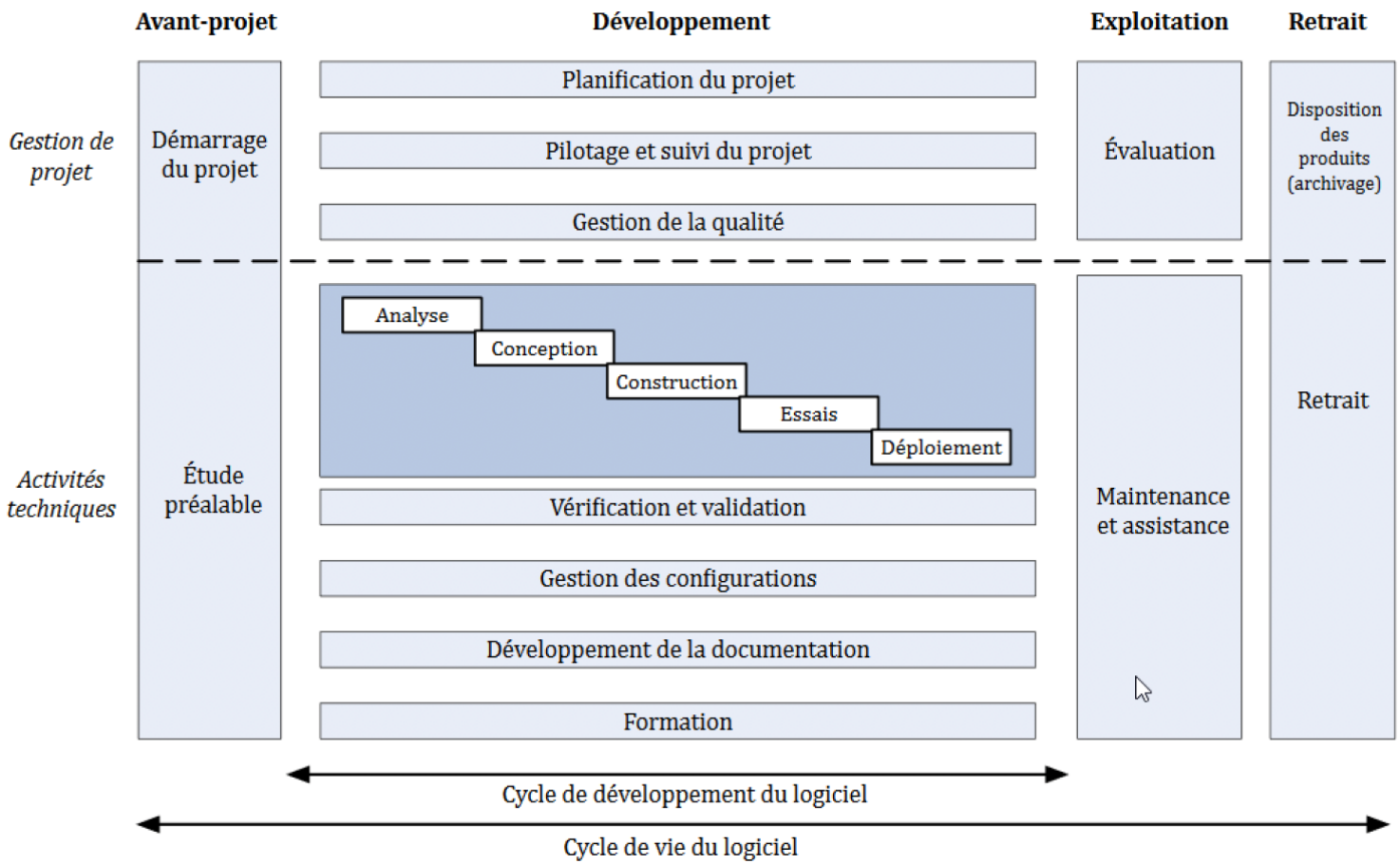
Parmi les métriques courantes, on peut citer la taille du code, le nombre de défauts, la complexité cyclomatique et le taux de couverture des tests.

Génie logiciel - Cycle de vie

Le cycle de vie d'un logiciel est l'ensemble des étapes qui décrivent son évolution depuis sa conception jusqu'à sa fin de vie. Il inclut généralement des phases telles que l'analyse des besoins, la conception, le développement, les tests, la mise en production et la maintenance.

Un modèle de cycle de vie décrit la séquence et les interactions entre ces phases, ainsi que les rôles et responsabilités des acteurs impliqués.

Génie logiciel - Cycle de vie



Génie logiciel - Analyse des besoins

Cette phase consiste à recueillir et analyser les exigences et les besoins des parties prenantes, telles que les utilisateurs, les clients et les développeurs.

L'objectif est de comprendre les objectifs et les contraintes du projet, ainsi que les fonctionnalités attendues du logiciel.

Les résultats de cette phase sont souvent documentés dans un cahier des charges ou un document de spécifications des besoins.

Génie logiciel - Conception

La phase de conception vise à définir l'architecture et la structure du logiciel, en tenant compte des exigences et des contraintes identifiées lors de l'analyse des besoins.

Les concepteurs déterminent les principaux composants du logiciel, leurs interactions et leurs responsabilités, ainsi que les algorithmes et les structures de données nécessaires.

Cette phase produit des documents de conception et des maquettes, qui serviront de base pour le développement.

Génie logiciel - Développement

Le développement consiste à écrire et à assembler le code source du logiciel, en suivant les spécifications de la conception.

Les développeurs utilisent des langages de programmation, des environnements de développement et des outils pour créer et organiser les différents composants du logiciel.

Ils travaillent également sur l'intégration des bibliothèques, des frameworks et des services externes nécessaires pour répondre aux exigences du projet.

Génie logiciel - Tests

La phase de tests vise à vérifier et à valider que le logiciel fonctionne correctement et répond aux exigences et aux attentes des parties prenantes.

Les tests peuvent être réalisés à différents niveaux, tels que les tests unitaires (pour vérifier le fonctionnement des composants individuels), les tests d'intégration (pour vérifier les interactions entre les composants) et les tests système (pour vérifier le fonctionnement global du logiciel).

Cette phase permet d'identifier et de corriger les défauts, les erreurs et les problèmes de performance.

Génie logiciel - Mise en production

La mise en production est le processus de déploiement du logiciel dans l'environnement de l'utilisateur final, où il sera utilisé pour effectuer les tâches pour lesquelles il a été conçu.

Cette phase implique souvent des activités de configuration, d'installation et de formation des utilisateurs, ainsi que la préparation de la documentation utilisateur et technique.

Génie logiciel - Maintenance

La maintenance du logiciel englobe toutes les activités liées à la correction des défauts, à l'adaptation aux nouvelles exigences ou aux évolutions technologiques, et à l'amélioration des performances et de la qualité du logiciel après sa mise en production.

La maintenance peut inclure la résolution de problèmes, la mise à jour des bibliothèques et des services externes, l'ajout de nouvelles fonctionnalités et la migration vers de nouvelles plateformes.

Chaque projet logiciel peut suivre un modèle de cycle de vie différent, en fonction de ses objectifs, de ses contraintes et de sa complexité.

Bibliographie

- Prof. Antoine Moevus, Collège Ahuntsic
- https://www5.in.tum.de/~huckle/DIABaggage.pdf