Chapitre 1 Qualité d'un logiciel

1. Introduction

Nous nous intéressons, dans ce chapitre à la qualité d'un logiciel et son suivi tout au long de son exploitation suivant des tests de produit avant et après l'exécution du code source.
La Génie Logicielle devient une discipline très importante pour plusieurs raisons par exemple la complexité des logiciels de plus en plus grande et nous avons besoin des techniques de GL pour l'améliorer. Cette discipline suppose des équipes de développeurs et non quelques spécialistes, une évolution des méthodes de travail et aussi un apprentissage à réutiliser l'existant (Ken Orr affirme que : 80% des développements existent déjà et que 20% seulement sont spécifiques)

2. Caractéristique de Logiciel

Produit unique : conçu et fabriqué une seule fois, reproduit
Inusable : Défauts pas dus à l'usure mais proviennent de sa conception
Complexe : Le logiciel est fabriqué pour soulager l'humain d'un problème complexe ; il est donc par nature complexe
Invisible : Fabrication du logiciel est une activité purement intellectuelle avec une difficulté de perception de la notion de qualité du logiciel
Techniques non matures : Encore artisanal (fait à la main) malgré les progrès.

3. Plan logiciel :

Ce plan est composé de
· La description du logiciel à réaliser en différents niveaux de produits (programmes et documents)
· Les moyens matériels et/ou logiciel à mettre à disposition ou à réaliser (Méthodes, Techniques, Outils)
· Le découpage du cycle de vie en phases, la définition des tâches à effectuer dans chaque phase et l'identification des responsables associés
· Les supports de suivi de l'avancement (Planning et calendriers)
· Les moyens utilisés pour gérer le projet
· Les points clés avec ou sans intervention du client

4. Constat sur le développement logiciel

· Le coût du développement du logiciel dépasse souvent celui du matériel,
· Les coûts dans le développement du logiciel :
◦ 20% pour le codage et la mise au point individuelle
◦ 30% pour la conception
◦ 50% pour les tests d'intégration
· Durée de vie d'un logiciel de 7 à 15 ans
· Le coût de la maintenance évolutive et corrective constitue la part prépondérante du coût total
· Plus de la moitié des erreurs découvertes en phases de tests proviennent d'erreurs introduites dans les premières étapes
· La récupération d'une erreur est d'autant plus coûteuse que sa découverte est tardive

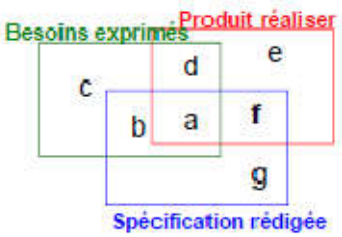
5. Exemples de désastres historiques

1988 abattages d'un Airbus 320 par l'USS Vincennes : affichage cryptique et confusion du logiciel de détection
1991 échecs de missile patriot : calcul imprécis de temps dû à des erreurs arithmétiques
London Ambulance Service Computer Aided Despatch System : plusieurs décès
Le 3 Juin 1980, North American Aerospace Defense Command (NORAD) rapporta que les U.S. étaient sous attaque de missiles
Echec du premier lancement opérationnel de la navette spatiale dont le logiciel d'exploitation temps réel consiste en environ 500,000 LOC : problème de synchronisation entre les ordinateurs de contrôle de vol
Réseau téléphonique longue distance d'AT&T : Panne de 9 heures provoqué par un patch de code non testé

5.1. Exemple type : Ariane 5

· C'est le lanceur successeur d' Ariane 4 pour le lancement de satellites en orbite terrestre.
· 37 seconde après décollage réussi, elle explose en vol
· Des informations incorrectes avaient été transmises mettant le système de contrôle en défaut
· L'expertise a montré que le défaut provenait de la tentative de conversion d'un nombre flottant 64-bit en entier signé 16-bit et qu'il n'y avait pas de gestion d'exceptions implantées
· Cette partie provenait d'Ariane 4 et du fait de l'accélération et de la vitesse moins importante, le problème n'était jamais apparu ni en vol réel ni en test
Ces erreurs prouvent que la complexité des systèmes informatisés peut conduire à des catastrophes et que les logiciels doivent être conçus avec méthode.

5.2. Problème de conformité aux besoins



a) Satisfaction
b) Non-conformité
c) Insatisfaction client
d) Satisfaction hasardeuse
e) Prestations inutiles hors spécification
f) Spécifications et prestations inutiles
g) Spécification inutiles

6. Indicateurs de qualité logicielle

La norme ISO 9126 définit six groupes d'indicateurs de qualité des logiciels:
1. la capacité fonctionnelle. C'est-à-dire la capacité qu'ont les fonctionnalités d'un logiciel à répondre aux exigences et besoins explicites ou implicites des usagers. En font partie la précision, l'interopérabilité, la conformité aux normes et la sécurité

2. La facilité d'utilisation, qui porte sur l'effort (le peu d') nécessaire pour apprendre à manipuler le logiciel. En font partie la facilité de compréhension, d'apprentissage et d'exploitation et la robustesse - une utilisation incorrecte n'entraîne pas de dysfonctionnement
3. La fiabilité, c'est-à-dire la capacité d'un logiciel de rendre des résultats corrects quelles que soient les conditions d'exploitation. En font partie la tolérance de pannes - la capacité d'un logiciel de fonctionner même en étant handicapé par la panne d'un composant (logiciel ou matériel) ;
4. La performance, c'est-à-dire le rapport entre la quantité de ressources utilisées (moyens matériels, temps, personnel), et la quantité de résultats délivrés. En font partie le temps de réponse, le débit et l'extensibilité - capacité à maintenir la performance même en cas d'utilisation intensive ;
5. La maintenabilité, qui porte sur l'effort nécessaire en vue de corriger ou de transformer le logiciel. En font partie l'extensibilité, c'est-à-dire le peu d'effort nécessaire pour y ajouter de nouvelles fonctions ;
6. La portabilité, c'est-à-dire l'aptitude d'un logiciel de fonctionner dans un environnement matériel ou logiciel différent de son environnement initial. En font partie la facilité d'installation et de configuration pour le nouvel environnement.
Chaque caractéristique contient des sous-caractéristiques. Il y a 27 souscaractéristiques.

6. Cycle de vie d'un logiciel

6.1. Définition

· Le « cycle de vie d'un logiciel » (software lifecycle), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition.
· Ce découpage permet de définir des jalons intermédiaires permettant la validation du développement logiciel (conformité du logiciel avec les besoins exprimés), et la vérification du processus de développement (adéquation des méthodes mises en œuvre).
· L'origine de ce découpage provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation.
· Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés.

6.2 Différentes phases du cycle de vie logiciel

a. Phase : définition des besoins (Pourquoi?)
Activités (du client, externe ou interne) : il faut effectuer des consultations d'experts et d'utilisateurs potentiels et réaliser des questionnaires d'observation de l'usager dans ses tâches :
· Pourquoi faut-il réaliser un logiciel?
· Y a-t-il de meilleures alternatives?
· Le logiciel sera-t-il satisfaisant pour les utilisateurs?
· Y a-t-il un marché pour le logiciel?
· A-t-on le budget, le personnel, le matériel nécessaire?
Productions :
· Cahier des charges (les « exigences »)
· Fonctionnalités attendues
· Contraintes non fonctionnelles
· Qualité, précision, optimalité, ...
· Temps de réponse, contraintes de taille, de volume de traitement, ...

b. Phase : Analyse des besoins / spécification (Quoi?)
Activités : Définir précisément les fonctions que le logiciel doit réaliser ou fournir en définissant les spécifications générales : Objectifs, contraintes (utilisation de matériels et outils existants) et généralités à respecter ; les spécifications fonctionnelles : description détaillé des fonctionnalités du logiciel et les spécifications d'interface : description des interfaces du logiciel avec le monde extérieure (hommes, autres logiciels, matériels...)
Productions :
· Dossier d'analyse
· Ébauche du manuel utilisateur
· Première version du glossaire

c. Phase : Codage et tests unitaires (Comment ?)
Activités : Définir la structure ou l'architecture du logiciel (décomposition en modules) et la spécification des interfaces des modules, ensuite réaliser la conception détaillée du logiciel en implémentant les algorithmes de chacune des procédures des modules et les tester indépendamment les uns des autres
Productions :
· Modules codés et testés
· Documentation de chaque module
· Résultats des tests unitaires
· Planning mis à jour

d. Phase : Intégration
Activités : Les modules doivent être testés et intégrés suivant un plan d'intégration, de plus le test de l'ensemble (modules intégrés) doit être réalisé conformément au plan de tests. Il faut intégrer les différents modules avec validation / vérification de l'adéquation de l'implantation, de la conception et de l'architecture avec le cahier de charge (acceptation)
Productions :
· Logiciel testé
· Jeu de tests de non-régression
· Manuel d'installation : guide dans les choix d'installation
· Version finale du manuel utilisateur
· Manuel de référence : liste exhaustive de chaque fonctionnalité

e. Phase : Qualification
Activités : Déploiement du logiciel chez le client et tests avec des sous ensemble d'usager choisi.
Dans cette étape il faut réaliser les tests en vraie grandeur dans les conditions normales d'utilisation et intégrer le logiciel dans l'environnement extérieur (autres logiciels, utilisateurs), ensuite vérifier que le logiciel développé répond aux besoins exprimés dans la phase de spécifications des besoins.
Productions :
· diagnostic OK / KO

f. Phase : Maintenance
Activités : Utilisation en situation réelle, retour d'information des usagers, des administrateurs, des gestionnaires…Il faut organiser une formation de l'utilisateur avec une

assistance technique et effectuer si nécessaire des maintenance correctives en cas de non conformité aux spécifications, avec détection et correction des erreurs résiduelles ; maintenance adaptative concernant la modification de l'environnement (matériel, fournitures logicielles, outil, …) ou maintenance évolutive à cause de changement des spécifications fonctionnelles du logiciel.

Productions :

· Logiciel corrigé
· Mises à jour, patch
· Documents corrigés.

7. Qualité du logiciel

7.1. Définition Intuitive :

La qualité est la conformité avec les besoins, l'adéquation avec l'usage attendu, le degré d'excellence ou, tout simplement, la valeur de quelque chose pour quelqu'un

7.2. Définition ISO :

Ensemble des traits et des caractéristiques d'un produit logiciel portant sur son aptitude à satisfaire des besoins exprimés ou implicites

7.3. Définition IEEE :

La qualité du logiciel correspond au degré selon lequel un logiciel possède une combinaison d'attributs désirés.

7.4. Importance de qualité de logiciel

Le logiciel est une composante majeure des systèmes informatiques (environ 80% du coût) utilisés pour : communication (ex. syst. téléphone, syst. Email), santé (monitoring), transport (ex. automobile, aéronautique), échanges économiques (ex. commerce), Entertainment...
Les défauts du logiciel sont extrêmement coûteux en termes d'argent, de réputation et de perte de vie

7.5. Facteurs de non qualité de logiciel

Mauvaise spécification : Les spécifications des clients qui ne sont pas des informaticiens Vague, incomplète, instables…
Mauvaise estimation : Fausse, oublis, précisions insuffisantes…
Mauvaise répartition des tâches : Organisation inadaptée, contraintes omises ou écart non détecté à temps
Mauvaise réalisation technique : Codage, tests, documentation
Problèmes humains : Mauvaise distribution des travaux, conflits ou rétention d''information
Manque d'expérience du métier de chef de projet : compétence inadaptés

8. Critères de qualité logicielle

ISO/IEC 9126 propose 6 caractéristiques de qualité du produit logiciel. Pour chaque facteur ou critère de qualité, il faut connaître sa définition son importance et comment l'évaluer du point de vue du fournisseur (évaluation en amont) et du point de vue du client (évaluation en aval) et savoir comment l'améliorer ou l'optimiser

8.1. Capacité fonctionnelle (Functionality)

Définition : Ensemble d'attributs portant sur l'existence d'un ensemble de fonctions et leurs propriétés. Les fonctions sont celles qui satisfont aux besoins exprimés ou implicites.

Ce critère possède les sous caractéristiques :

· Aptitude : présence et adéquation d'une série de fonctions pour des tâches données

· Exactitude : fourniture de résultats ou d'effets justes ou convenus

· Interopérabilité : capacité à interagir avec des systèmes donnés

· Sécurité : aptitude à empêcher tout accès non autorisé (accidentel ou délibéré) aux programmes et données

Ce critère est plus important des critères de qualité on souhaite développer des logiciels qui répondent aux besoins de l'utilisateur tout en faisant un bon usage de ses ressources. Il est capitale pour l'utilisateur final et le client.

8.2. Fiabilité (Reliability)

Définition : Ensemble d'attributs portant sur l'aptitude du logiciel à maintenir son niveau de service dans des conditions précises et pendant une période déterminée.

Ce critère possède les sous caractéristiques :

· Maturité : fréquence des défaillances dues à des défauts du logiciel

· Tolérance aux fautes : aptitude à maintenir un niveau de service donné en cas de défaut du logiciel ou de violation de son interface

· Possibilité de récupération : capacité à rétablir son niveau de service et de restaurer les informations directement affectées en cas de défaillance ; temps et effort nécessaire pour le faire Ce critère correspond à l'aptitude d'un logiciel à fournir des résultats voulus dans les conditions normales d'utilisation selon deux point de vue : qualité de la conception et qualité de la réalisation.

Un logiciel est fiable doit répondre aux spécifications, ne jamais produire de résultats faux ou être dans un état incohérent, il doit aussi réagir utilement dans une situation inattendue et ne jamais être en défaut qu'en cas extrême

8.3. Facilité d'utilisation (Usability)

Définition : Ensemble d'attributs portant sur l'effort nécessaire pour l'utilisation et l'évaluation individuelle de cette utilisation par un ensemble défini ou implicite d'utilisateurs.

Ce critère possède les sous caractéristiques :

· Facilité de compréhension : effort que doit faire l'utilisateur pour reconnaître la logique et sa mise en œuvre

· Facilité d'apprentissage : effort que doit faire l'utilisateur pour apprendre son application, il doit comprendre ce que l'on peut faire avec le logiciel, et savoir comment le faire.

· Facilité d'exploitation : effort que doit faire l'utilisateur pour exploiter et contrôler l'exploitation de son application

Ce critère concerne l'effectivité, l'efficacité et la satisfaction avec lesquelles des utilisateurs spécifiés accomplissent des objectifs bien définis dans un environnement particulier. Avant de réaliser un logiciel il faut analyser le mode opératoire des utilisateurs, ensuite adapter l'ergonomie des logiciels aux utilisateurs. Ce critère comprend la façon dont l'information et les suggestions sont présentées à l'utilisateur final, et aussi la documentation disponible, et l'effort nécessaire pour apprendre à se servir du système.

8.4. Rendement (Efficiency)

Définition : Ensemble d'attributs portant sur le rapport existant entre le niveau de service d'un logiciel et la quantité de ressources utilisées, dans des conditions déterminées.

Ce critère possède les sous caractéristiques :

· Comportement vis-à-vis du temps : temps de réponses et de traitement ; débits lors de l'exécution de sa fonction

· Comportement vis-à-vis des ressources : quantité de ressources utilisées ; durée de leur utilisation lorsqu'il exécute sa fonction

Les logiciels doivent satisfaire aux contraintes de temps d'exécution en développant des produits plus simples tout en veillant à la complexité des algorithmes et en utilisant des machines plus performantes.

8.5. Maintenabilité (Maintainability)

Définition : Ensemble d'attributs portant sur l'effort nécessaire pour faire des modifications données.

Ce critère possède les sous caractéristiques :

· Facilité d'analyse : effort nécessaire pour diagnostiquer les déficiences et causes de défaillance ou pour identifier les parties à modifier

· Facilité de modification : effort nécessaire pour modifier, remédier aux défauts ou changer d'environnement

· Stabilité : risque des effets inattendus des modifications

· Facilité de test : effort nécessaire pour valider le logiciel modifié

Ce critère concerne la gestion du processus de modification pour éviter que des corrections partielles soient faites en dehors du processus d'itérations.

Il existe trois types de maintenance

· Maintenance perfective (évolutive) qui consiste à maintenir les fonctionnalités antérieures tout en ajoutant des nouvelles fonctionnalités qui modifient profondément l'architecture comme le changement de SGBD…

· Maintenance adaptative qui correspond à l'ajout de petites fonctionnalités qui ne modifie pas l'architecture comme passage de données par fichiers…

· Maintenance corrective qui permet la correction des anomalies ou des erreurs mises à jour par le client et non pas lors des tests de vérification et de validation.

Le coût de la maintenance est influencé par l'âge du logiciel, l'importance des modifications, la stabilité de l'équipe, la qualité de la documentation technique ou de document de maintenance.

8.6. Portabilité (Portability)

Définition : Ensemble d'attributs portant sur l'aptitude du logiciel à être transféré d'un environnement à l'autre. Ce critère possède les sous caractéristiques :

· Facilité d'adaptation : possibilité d'adaptation à différents environnements donnés sans que l'on ait recours à d'autres actions ou moyens que ceux prévus à cet effet pour le logiciel considéré.

· Facilité d'installation : effort nécessaire pour installer le logiciel dans un environnement donné

· Conformité aux règles de portabilité : conformité aux normes et aux conventions ayant trait à la portabilité
· Interchangeabilité : possibilité et effort d'utilisation du logiciel à la place d'un autre logiciel donné dans le même environnement
Ce critère est aussi très important car il faut toujours essayer de développer un logiciel portable d'une plate-forme à une autre, d'un site à un autre ; sachant que les architectures machines diffèrent, les systèmes d'exploitation, les performances aussi… de plus il faut prendre en considération qu'un logiciel vit et dure dans le temps et que l'environnement humain et matériel évolue.

8.7. Compromis de qualité
Les critères de qualité peuvent être contradictoires, pour chaque, le choix des compromis doit s'effectuer en fonction du contexte.
· Facilité d'emploi vs maintenabilité
· Fiabilité vs portabilité
· Capacité vs rendement
La qualité s'obtient par la mise en place de procédure et des méthodes de phase de spécification tout en prenant en compte que la démarche qualité ne consiste pas à corriger les défauts mais à les prévenir