Qualité et tests logiciels

L'ÉCOLE SUPÉRIEURE MULTINATIONALE DES TÉLÉCOMMUNICATIONS (ESMT)

LICENCE PROFESSIONNELLE EN TELECOMMUNICATIONS ET INFORMATIQUE

(LPTI L3)

Plan du Chapitre

- Introduction
- Le crash de la fusée Ariane
- Les erreurs, les fautes et les pannes de logiciel
- La qualité logiciel

Introduction

- Le logiciel est développé, maintenu et utilisé par des personnes dans une grande variété de situations.
- Les étudiants créent des logiciels dans leurs classes,
- les passionnés deviennent membres d'équipes de développement open-source
- les professionnels développent des logiciels pour divers domaines d'activité, de la finance à l'aérospatiale.

Tous ces groupes individuels devront s'attaquer aux problèmes de qualité qui se posent dans le logiciel avec lequel ils travaillent.

Ariane 5

- Une fusée européenne conçue pour lancer des charges utiles commerciales (par exemple des satellites de communication, etc.) en orbite terrestre
- Successeur des lanceurs Ariane 4 à succès
- Ariane 5 peut transporter une charge utile plus lourde qu'Ariane 4

Défaillance du lanceur

- Environ 37 secondes après un décollage réussi, le lanceur Ariane 5 a perdu le contrôle.
- Des signaux de contrôle incorrects ont été envoyés aux moteurs et ceux-ci pivotaient de sorte que des contraintes insoutenables ont été imposées à la fusée.
- Il a commencé à se briser et a été détruit par les contrôleurs au sol.
- La défaillance du système a été le résultat direct d'une défaillance du logiciel. Cependant, il était symptomatique d'une défaillance plus générale de validation des systèmes.

Le problème

- L'altitude et la trajectoire de la fusée sont mesurées par un système de référence inertielle informatisé. Cela transmet des commandes aux moteurs pour maintenir l'altitude et la direction.
- Le logiciel a eu une défaillance et ce système et le système de sauvegarde s'arrêtent.
- Les commandes diagnostiques ont été transmises aux moteurs qui les ont interprétées comme des données réelles et qui ont pivoté à une position extrême entraînant des contraintes imprévues sur la fusée.

Défaillance logicielle

- La défaillance logicielle s'est produite lorsqu'une tentative de convertir un nombre flottant 64 bits en un entier 16 bits signé a provoqué le débordement du nombre.
- Il n'y avait pas de gestionnaire d'exception associé à la conversion, de sorte que la gestion des exceptions système a été invoquée. Celle-ci a arrêté le logiciel.
- Le logiciel de sauvegarde était une copie et s'est comporté exactement de la même manière.

Échec évitable ?

- Le logiciel qui a échoué a été réutilisé à partir du lanceur Ariane 4.
- Des décisions ont été prises de ne pas le changer car cela pourrait introduire de nouveaux défauts;
- Ne pas tester les exceptions de débordement parce que le processeur était lourdement chargé. Pour des raisons de fiabilité, il a été jugé souhaitable d'avoir une certaine capacité de processeur de rechange.

Pourquoi pas Ariane 4 ?

- Les caractéristiques physiques d'Ariane 4 (Un véhicule plus petit) sont telles qu'il a une accélération initiale plus faible que celle d'Ariane 5.
- La valeur de la variable sur Ariane 4 ne pouvait atteindre le niveau qui a causé des débordements durant la phase de lancement.

Échec de validation

- Comme l'installation qui a échoué n'était pas requise pour Ariane 5, il n'y avait aucune exigence qui lui était associée.
- Comme il n'y avait aucune exigence connexe, il n'y avait aucun test de cette partie du logiciel et donc aucune possibilité de découvrir le problème.
- Pendant les essais système, des simulateurs des ordinateurs inertiels de système de référence ont été utilisés. Ceux-ci n'ont pas généré l'erreur car il n'y avait aucune exigence !

Échec de l'examen

- La conception et le code de tous les logiciels doivent être examinés pour déceler les problèmes pendant le processus de développement
- Soit :
  - Le logiciel du système de référence inertiel n'a pas été examiné parce qu'il avait été utilisé dans une version précédente;
  - L'examen n'a pas révélé le problème ou que la couverture du test ne révélerait pas le problème;
  - L'examen n'a pas permis d'apprécier les conséquences de l'arrêt du système lors d'un lancement.

Leçons apprises

- N'exécutez pas de logiciel dans les systèmes critiques à moins qu'il ne soit réellement nécessaire.
- En plus de tester ce que le système devrait faire, vous devrez peut-être également tester ce que le système ne devrait pas faire.
- Pas de réponse de traitement d'exception par défaut qui est l'arrêt du système dans les systèmes qui n'ont pas de mode de fonctionnement dégradé.

Leçons apprises

- Dans les calculs critiques, toujours retourner les meilleures valeurs approchées, même si les valeurs absolument correctes ne peuvent pas être calculées.
- Dans la mesure du possible, utilisez de l'équipement réel et non des simulations.
- Améliorer le processus d'examen pour inclure les participants externes et examiner toutes les hypothèses formulées dans le code.

Défaillance évitable

- Le concepteur d'Ariane 5 a commis une erreur critique et élémentaire.
- Ils ont conçu un système où une seule défaillance de composant pourrait causer la défaillance de l'ensemble du système.
- En règle générale, les systèmes critiques doivent toujours être conçus pour éviter un seul point de défaillance.

Définir le logiciel

- Tout ou partie des programmes, procédures, règles et documents connexes d'un système de traitement de l'information.
- Programmes informatiques, procédures et éventuellement la documentation et les données connexes relatives au fonctionnement d'un système informatique.

ISO 24765

Définir la qualité logiciel

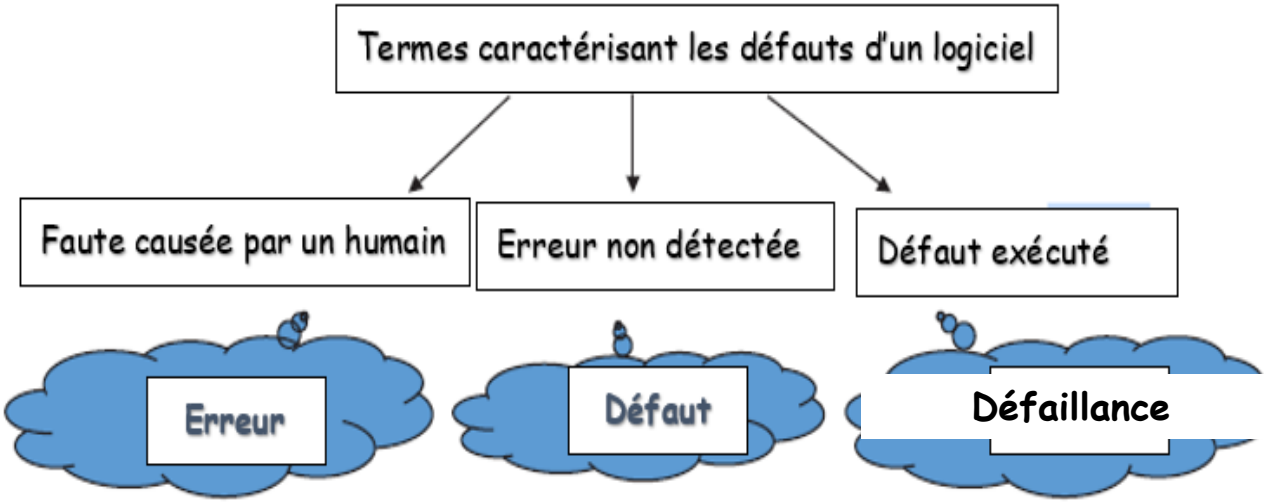
- Si l'on considère cette définition, il est clair que les programmes ne sont qu'une partie d'un ensemble d'autres produits (appelés livrables) et d'activités qui font partie du cycle de vie du logiciel.

Erreur-Défaut-Défaillance

- Lors de diverses réunions, on remarque qu'il existe de nombreux termes qui sont utilisés pour décrire les problèmes avec un logiciel. Par exemple :
- Le système a crashé en cours de production.
- Le concepteur a fait une erreur.
- Après un examen, nous avons constaté un défaut dans le plan de test.
- J'ai trouvé un bug dans un programme aujourd'hui.
- Le système est tombé en panne.
- Le client s'est plaint d'un problème avec un calcul dans le bulletin de salaire.
- Une défaillance a été signalée dans le sous-système de surveillance.

Erreur-Défaut-Défaillance

- C'est important d'utiliser une terminologie claire et précise pour donner un sens spécifique à chacun de ces termes. La figure suivante décrit comment utiliser ces termes correctement.



Erreur-Défaut-Défaillance

- Les 3 termes décrivent les problèmes de qualité du logiciel.
- Ils identifient également les questions qui sont étudiées par les chercheurs dans le domaine de la qualité des logiciels afin de découvrir des moyens pour aider à les éliminer.
- Des erreurs peuvent se produire dans n'importe quelle phase de développement de logiciels tout au long du cycle de vie.
- Les défauts doivent être identifiés et corrigés avant qu'ils ne conduisent à des défaillance(pannes).
- La cause des pannes, défauts et erreurs doit être identifiée.

Erreur-Défaut-Défaillance

- Ex 1: Une pharmacie a ajouté une exigence à sa caisse enregistreuse pour empêcher les ventes de plus de 50kf aux clients qui doivent plus de 250kf sur leur carte de crédit de pharmacie. Le développeur n'a pas bien compris la spécification et a créé une limite de vente de 600kf dans le programme. Ce défaut n'a jamais causé de défaillance puisqu'aucun client ne pouvait acheter pour une valeur de plus de 600kf étant donné que la carte de crédit avait une limite de 500kf ...

- Ex 2: En 2009, un programme de fidélisation a été lancé aux clients d'AS, un important fournisseur de meubles. Les spécifications décrivaient les règles d'affaires suivantes : "***un client qui effectue un achat mensuel supérieur au montant moyen des achats mensuels pour tous les clients sera considéré comme un client privilégié(CP). Le CP sera identifié lors d'un achat, et recevra immédiatement un cadeau ou un rabais important une fois par mois***". Le défaut introduit dans le système (en raison d'une mauvaise compréhension de l'algorithme de mise en place de l'exigence) prenait seulement en compte le montant moyen des achats en cours et non l'historique mensuel du client. La défaillance du logiciel(la caisse enregistreuse identifiait beaucoup trop de CP), a entraîné une perte pour AS.

Erreur-Défaut-Défaillance

Ex 3: : La cellule AQ teste le programme de Baba. Il constate un défaut dans le calcul d'un régime d'épargne-retraite conçu pour appliquer la nouvelle loi d'exonération fiscale pour ce type de placement. Il retrace l'erreur jusqu'à la spécification du projet et en informe l'analyste. Dans ce cas, l'activité de test a correctement identifié le défaut et la source de l'erreur.

Erreur-Défaut-Défaillance

Cycle de vie

- Évolution d'un système, d'un produit, d'un service, d'un projet ou de tout autre entité fabriquée, de la conception au retrait du produit.
- **Cycle de vie de développement**

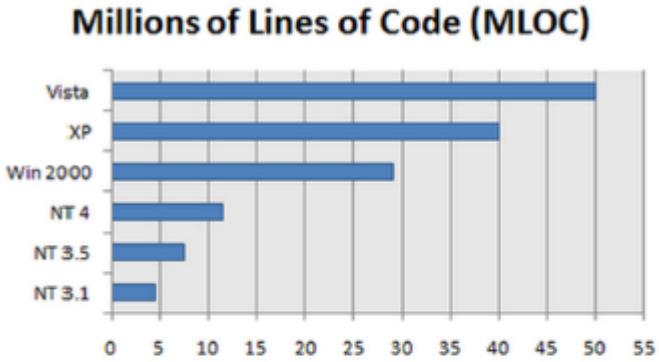
  Processus du cycle de vie logiciel qui contient les activités recueil des exigences, conception, codage, intégration, test, installation et support pour l'acceptation des logiciels.
  ISO 12207

Erreur-Défaut-Défaillance

- Pendant le cycle de vie de logiciels, les défauts sont constamment introduits involontairement et doivent être localisés et corrigés dès que possible.
- Par conséquent, il est utile de recueillir et d'analyser des données sur les défauts trouvés ainsi que le nombre estimé de défauts laissés dans le logiciel.
- Ce faisant, nous pouvons améliorer les processus d'ingénierie logicielle et, en retour, minimiser le nombre de défauts introduits dans les nouvelles versions de produits logiciels à l'avenir.

Complexité Logiciels

- Measured in MLOC
  - Million lines of code
  - Beware, LOC is a dubious measure for software complexity
- # faults is linearly related to LOC
  - ~ 3K faults / MLOC.
- Some system sizes
  - WoW: > 5 MLOC
  - ASML wafer scanner: > 35 MLOC (2011)
  - LHC: > 50 MLOC, ergo more faults than detected hadrons!



- LHC Large Hadron Collider
- Linux: 15 MLOC (2011)

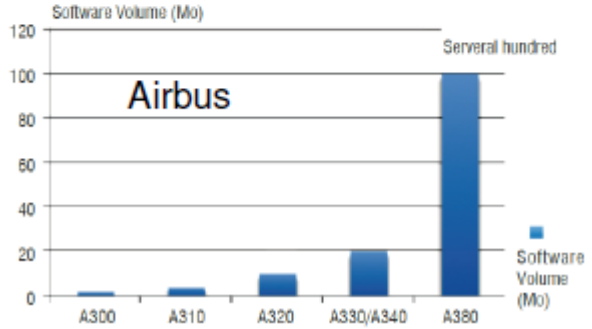


Figure 2 - Evolution of the volume of embedded software

Véhicules modernes

- What's Got 10M Lines of Code & An IP Address?
  - The Volt (http://gigaom.com/cleantech/whats-got-10m-lines-of-code-an-ip-address-the-volt)

- IEEE Magazine, January 2009:
  - 30-100 ECUs (microprocessors) in modern cars
  - 10M-100M lines of code!

Air-bag system	Antilock brakes	Automatic transmission
Alarm system	Climate control	Collision-avoidance system
Cruise control	Communication system	Dashboard instrumentation
Electronic stability control	Engine ignition	Engine control
Electronic-seat control	Entertainment system	Navigation system
Power steering	Tire-pressure monitoring	Windshield-wiper control

- Hence
  - .... 30K – 300K errors
  - integration problem: need clear structures, interfaces, standards

Erreur-Défaut-Défaillance

- De nombreux chercheurs ont étudié la source des erreurs logicielles et ont publié des études classifiant les erreurs logicielles par type afin d'évaluer la fréquence de chaque type d'erreur.
- Beizer (1990) fournit une étude qui a combiné le résultat de plusieurs autres études pour donner une indication de l'origine des erreurs.
- Voici une liste résumée des résultats de cette étude.

Erreur-Défaut-Défaillance

- 25 % structurelles;
- 22 % de données;
- 16 % de fonctionnalités mises en œuvre;
- 10 % de construction/codage;
- 9 % d'intégration;
- 8 % d'exigences/spécifications fonctionnelles;
- 3 % de définition/tests d'exécution;
- 2 % d'architecture/conception;
- 5 % non spécifiés.

Erreur-Défaut-Défaillance

- Les chercheurs tentent également de déterminer combien d'erreurs peuvent être attendues dans un logiciel typique.
- McConnell a suggéré que ce nombre variait en fonction de la qualité et de la maturité des processus d'ingénierie logicielle ainsi que de la formation et de la compétence des développeurs.
- Plus les processus sont matures, moins les erreurs sont introduites dans le cycle de vie du développement du logiciel.

Erreur-Défaut-Défaillance

- Humphrey a également recueilli des données auprès de nombreux développeurs.
- Il a constaté qu'un développeur crée involontairement environ 100 défauts pour chaque 1000 lignes de code source écrites.
- En outre, il a noté de grandes variations pour un groupe de 800 développeurs expérimentés,
- c'est-à-dire de moins de 50 défauts à plus de 250 défauts injectés par 1000 lignes de code.

Erreur-Défaut-Défaillance

- L'utilisation de processus éprouvés, de développeurs compétents et bien formés, et la réutilisation de composants logiciels déjà testés peuvent réduire considérablement le nombre d'erreurs d'un logiciel.
- AQL devra développer une classification des causes de l'erreur logicielle par catégorie qui peut être utilisée par toutes les personnes impliquées dans le processus d'ingénierie logicielle.

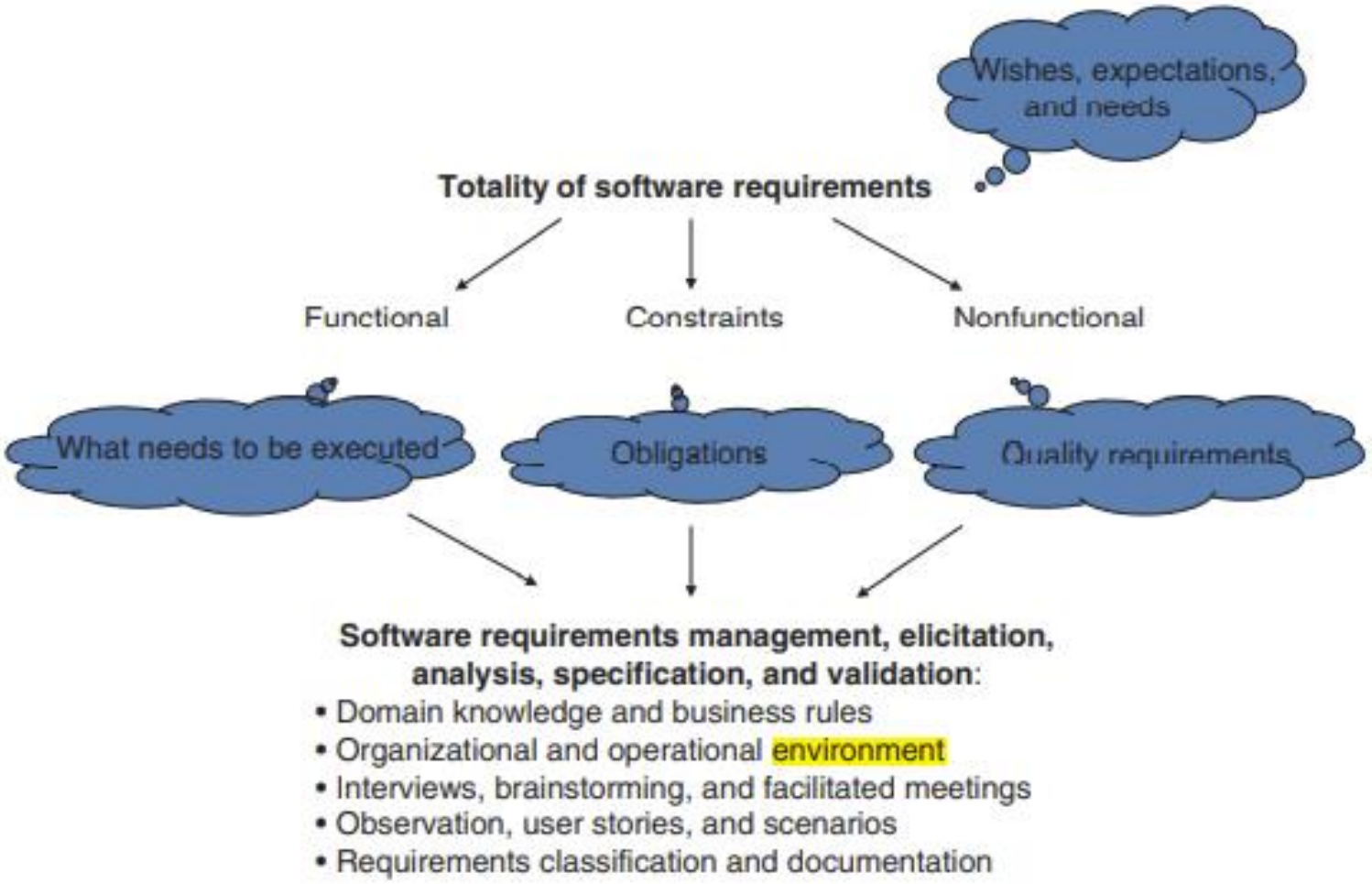
Erreur-Défaut-Défaillance

Huit catégories populaires de cause d'erreur :

- problèmes liés à la définition des exigences;
- communication entre le client et le développeur;
- déviations par rapport aux spécifications;
- erreurs d'architecture et de conception;
- erreurs de codage (y compris le code de test);
- non-conformité aux processus/procédures en cours;
- des revues et des tests inadéquats;
- erreurs de documentation.

Ces catégories sont revues en détail dans la suite, sous forme d'exercice.

Erreur-Défaut-Défaillance



La qualité du logiciel

- Nous voyons qu'il existe de nombreuses sources d'erreurs potentielles, et que sans AQL, ces défauts peuvent entraîner des défaillances si elles ne sont pas découvertes.
- Comment définir la qualité du logiciel? Les groupes de normalisation suggèrent la définition suivante.

La qualité du logiciel

qualité logiciel

Conformité aux exigences logicielles établies; la capacité d'un produit logiciel de satisfaire les besoins déclarés et implicites lorsqu'il est utilisé dans des conditions précises (ISO 25010)

La mesure dans laquelle un produit logiciel répond aux exigences établies; Cependant la qualité dépend de la mesure dans laquelle ces exigences établies représentent avec précision les besoins, les désirs et les attentes des parties prenantes. (IEEE 730).

La qualité du logiciel

Par conséquent, les logiciels de qualité sont des logiciels qui répondent aux besoins réels des parties prenantes tout en respectant les contraintes de coûts et de temps prédéfinies.

Les besoins du logiciel peuvent être définis à 4 niveaux :

- Besoins réels
- Besoins exprimés
- Besoins spécifiés
- Besoins atteints

La qualité du logiciel

Facteurs qui peuvent influer sur le respect des véritables exigences du client

Type of requirement	Origin of the expression	Main causes of difference
True	Mind of the stakeholders	– Unfamiliarity with true requirements – Instability of requirements – Different viewpoints of ordering party and users
Expressed	User requirements	– Incomplete specification – Lack of standards – Inadequate or difficult communication with the ordering party – Insufficient quality control
Specified	Software Specification Document	– Inappropriate use of management and production methods, techniques, and tools – Insufficient tests
Achieved	Documents and Product Code	– Insufficient quality control techniques

La qualité du logiciel

- La capacité du logiciel à répondre (ou non) aux besoins du client peut être décrite dans les différences entre ces quatre niveaux.
- Tout au long de l'élaboration d'un projet, il y aura des facteurs qui auront une incidence sur la qualité finale.

Bibliographie

- Claude Y Laporte et Alain April, Software Quality Assurance, Ch 1 Wiley-IEEE, jan 2018, 624 pages.