Planification de test, documentation de test et sujets reliés

Signalisation du problème, suivi et gestion de test

En grande partie basé sur le matériel
de Ian Sommerville, Pankaj Jalote et João Pascoal Faria

Sommaire

- Gestion de projet de logiciel
- Planification de qualité de logiciel
- Planification de test

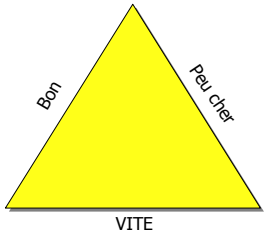
Gestion de projet du logiciel

Gestion de projet
Planification de projet

Gestion de projet

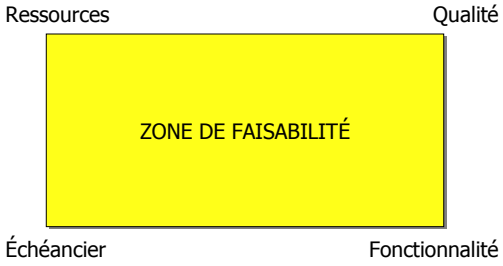
- Buts
  - Logiciel livré dans les limites du budget prévu.
  - Logiciel livré dans les limites de l'échéancier prévu.
  - Logiciel construit conformément aux besoins.
- Pourquoi ?
  - Des projets bien gérés échouent parfois.
  - Des projets mal gérés échouent inévitablement.
  - Le processus de développement du logiciel n'est pas standardisé.

Dilemme de la gestion du projet



VOUS POUVEZ EN AVOIR SEULEMENT DEUX !!!!

Rectangle du gestionnaire de projet



Responsabilités du gestionnaire de projet

- Écriture de la proposition
- Estimation du coût du projet
- Planification et ordonnancement du projet
- Contrôle et révisions du projet
- Sélection et évaluation du personnel
- Écriture et présentation du rapport

Sommerville, 1995

Alors comment faisons-nous cela ?

- Prendre le temps de comprendre le problème.
- Estimer le quantité d'effort requis (Ceci est dur !!)
  - Nombre de grandes fonctions.
  - Difficulté de chaque fonction.
- Développer l'échéancier avec un filet de sécurité incorporé.
  - Augmenter les estimations par quelques facteurs.
  - Avoir une copie de sauvegarde du plan pour les pires cas.
  - Être certain que l'échéancier est réaliste.
- Réviser l'échéancier à mesure que la compréhension du projet augmente.

Survol de l'estimation

- Difficile & sujet à l'erreur
- Raffinement graduel
  - Au début du projet, on a une idée « vague » du problème. Par conséquent, l'évaluation du temps et de l'effort sera « vague » aussi.
  - C'est seulement quand le projet se développera et que les problèmes et solutions deviendront plus clairs qu'on aura une meilleure précision pour les estimations.
- Processus d'estimation
  - Estimer la taille du produit.
    - ✓ Lignes de code (LOC)
    - ✓ Points de fonctions
    - ✓ Nombre de fonctions
  - Estimer l'effort
    - ✓ Personne-mois
    - ✓ 50-60 Heures/Crédit/Personne
  - Estimer l'échéancier
    - ✓ Calendrier du temps

De l'estimation à l'échéancier

- Raffinement
  - énoncé initial du problème (x4)
  - Besoins (x1.5)
  - Conception de haut niveau (x1.25)
  - Spécification de la Conception détaillée (x1.1)
  - Implémentation (x1)
- Descendant vs Ascendant
- Cas
  - Meilleur cas
  - Cas le plus probable
  - Cas courant
  - Pire cas

Planification

- Activités
  - Diviser le projet en tâches.
    - ✓ Estimer le temps et les ressources requis.
  - Organiser les tâches simultanément pour une utilisation optimale de la main-d'oeuvre.
  - Minimiser les dépendances des tâches pour éviter les retards.
  - Sortir les critères.
- Problèmes
  - L'estimation est difficile.
  - La productivité n'est pas proportionnelle au nombre de personnes.
  - Ajouter des personnes à un projet en retard le retarde plus.
  - L'imprévu survient toujours -- intégrer son « éventualité ».

Planification

- Points marquants (jalons): besoin d'environ 1 par semaine/mois.
- Ne pas utiliser le statut de soumission de rapport.
- Choisir de gouverner l'exécution du projet.

Quelques figures historiques (petits projets)

- Architecture/Conception 10%
- Conception détaillée 20%
- Coder/Déboguer 25%
- Test unitaire 20%
- Test d'intégration 15% → 45 %
- Test de système 10%

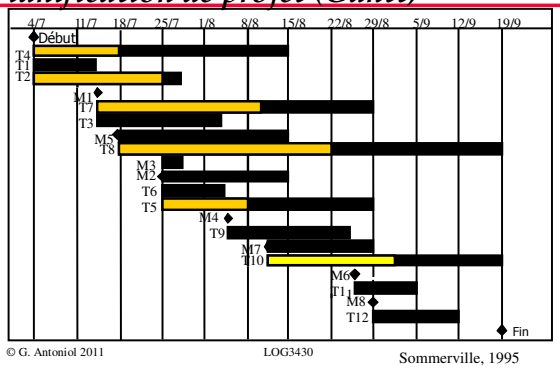
Planification

- Dépendances
- Qu'est-ce qui peut se faire avant de commencer cette activité ?
- Gouverner le séquencement.

Planification

- Dérivée de l'estimation du niveau d'effort requis.
- Ne pas oublier que le test et l'intégration prennent aussi du temps.
- Être réaliste :
  - Autres classes.
  - Travail / activités externes.
  - Manger & dormir.
- Filet de sécurité incorporé & planification de copie de sauvegarde.
- Trouver un chemin critique.

Planification de projet (Gantt)



Sommerville, 1995

Analyses de risque

- Risque
  - Quelque chose peut clocher.
  - Souvent le résultat d'une information inadéquate.
- Evaluation (Identifier, Analyser, Classer par priorité)
- Contrôler (Planification, Résolution, Surveillance)

Niveau de gestion de risque

- Gestion de crise (éteindre les feux)
- Réparer la défaillance
- Atténuer le risque
- Prévention du risque
- Élimination des causes racines

Alternatives de résolution du risque

- Acceptation
- Évitement (Éliminer)
- Protection (Redondance)
- Réduction (Mitigation, Prévention, Anticipation)
- Recherche (Besoin de plus d'infos)
- Réserves (fonds, banque, coussin)
- Transfert (relayer à quelqu'un d'autre)

En résumé …

- Une bonne gestion de projet est essentielle pour le succès d'un projet.
- Le gestionnaire a des rôles divers, mais se concentre sur :
  - la Planification
  - l'estimation
  - l'échéancier
- La planification et l'estimation sont des processus itératifs.
- Attention aux « Fuzzy Front End » (McConnell)

Sommerville, 1995

Gestion de la qualité de logiciel :

Assurance qualité :

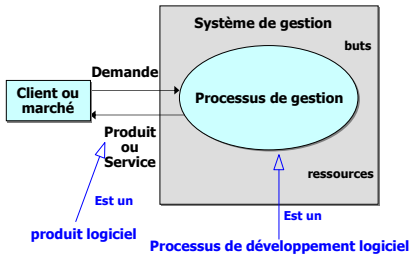
Un patron planifié et systématique de toutes les actions nécessaires pour fournir une confiance adéquate que l'item ou le produit est conforme aux besoins techniques établis.

Gestion de la qualité
du processus logiciel et de ses produits.

Index

- Introduction à la qualité du logiciel
- Assurance qualité et standards
- Planification et contrôle de la qualité
- Test de logiciel
- Conclusions

Produit et processus



Qu'est-ce qu'un produit logiciel ?

- Produit logiciel = programmes informatiques (sources et exécutables) + documentation associée.
- Les produits logiciels peuvent être
  - **Sur mesure** – développés pour un client particulier, en accord à ses spécifications.
  - **Génériques** (« ensemble ») – développés pour le **marché** en général, pour être vendus à une variété de consommateurs différents.
- Types de produits logiciels
  - logiciel d'aide au développement de logiciel
    - ✓ Inclut les outils du génie logiciel dans la gestion du génie logiciel.
  - logiciel du gestion de la productivité du personnel
    - ✓ Tableur, outils de traitement de texte, …
  - Logiciel embarqué
  - …

Qu'est-ce qu'est la qualité du produit ?

- La qualité, tout simplement, signifie qu'un produit devrait rencontrer ses spécifications.
  - Le produit logiciel devrait livrer la fonctionnalité requise (*fonctionnalités requises*) avec les **attributs de qualité** (*non-fonctionnalités requises*).
- Les attributs de qualité sont fréquemment en conflit et augmentent le coût de développement, c'est pourquoi il y a un besoin de compromis.
- Le génie logiciel est soucieux du développement d'un bon logiciel avec un coût optimisé.

Qualité

- La qualité est problématique pour les systèmes logiciels.
  - La tension entre les exigences de qualité du client (efficacité, fiabilité, …) et les exigences de la qualité du développement (maintenance, réutilisabilité, …).
  - Quelques exigences de qualité sont difficiles à spécifier d'une manière non équivoque.
  - Les spécifications logicielles sont normalement incomplètes et souvent incohérentes.
  - La qualité accepte un compromis : nous ne pouvons pas attendre que les spécifications s'améliorent pour nous intéresser à la qualité. Les procédures doivent être mises en place pour améliorer la qualité malgré la spécification imparfaite.

L'état courant de la qualité logicielle

- Microsoft Windows XP End-User License Agreement:

11. LIMITED WARRANTY FOR PRODUCT ACQUIRED IN THE US AND CANADA.

Microsoft **warrants that the Product will perform substantially in accordance with the accompanying materials for a period of ninety days from the date of receipt**.
(…)
If an implied warranty or condition is created by your state/jurisdiction and federal or state/provincial law prohibits disclaimer of it, you also have an implied warranty or condition, BUT ONLY AS TO DEFECTS DISCOVERED DURING THE PERIOD OF THIS LIMITED WARRANTY (NINETY DAYS).
(…)
Some states/jurisdictions do not allow limitations on how long an implied warranty or condition lasts, so the above limitation may not apply to you.
(…)
YOUR EXCLUSIVE REMEDY. Microsoft's and its suppliers' entire liability and your exclusive remedy shall be, at Microsoft's option from time to time exercised subject to applicable law, **(a) return of the price paid** (if any) for the Product, or **(b) repair or replacement of the Product**, that does not meet this Limited Warranty and that is returned to Microsoft with a copy of your receipt.
(..)
This Limited Warranty is void if failure of the Product has resulted from accident, abuse, misapplication, abnormal use or a virus.

Les attributs de la qualité du produit (1)

- Les attributs des bons logiciels (au-delà de livrer la fonctionnalité requise):
- Efficacité
  - Le logiciel ne devrait pas gaspiller l'utilisation des ressources du système (disque et espace de mémoire, temps du CPU, etc.) et devrait présenter un temps réponse approprié.
- Utilisabilité (facilité d'utilisation)
  - Le logiciel doit être utilisable par les usagers pour qui il a été créé.
- Dependabilité (fiabilité, accessibilité, sécurité, sûreté, …).
  - Le logiciel doit être digne de confiance.
- Maintenabilité (facilité de maintenance)
  - Le logiciel doit être développé pour rencontrer les besoins de changement.
  - Le logiciel coûte plus cher à entretenir qu'à développer. Pour les systèmes à longue vie, les coûts de la maintenance peuvent être plusieurs fois ceux du développement.
- …

Les attributs de la qualité du produit(2)

- Autres attributs de qualité :
  - Résilience (résistance)
  - Robustesse
  - Compréhensibilité
  - Testabilité
  - Adaptabilité
  - Modularité
  - Simplicité
  - Portabilité
  - Réutilisabilité
  - Facilité d'apprentissage

Dimensions principale de confiance

- ***Fiabilité*** – La probabilité d'un système soit sans panne en un temps spécifique dans un environnement donné pour un but donné.
- ***Accessibilité*** – La probabilité qu'un système, à un point dans le temps, sera opérationnel et capable de livrer les services requis.
  - Il est possible d'avoir une haute accessibilité avec une fiabilité faible si les pannes sont rapidement réparées.
- ***Sûreté*** – La capacité du système à opérer, normalement ou anormalement, sans danger de causer des blessures humaines ou la mort et sans endommager l'environnement du système.
- ***Sécurité*** – La capacité du système à se protéger d'attaques externes accidentelles ou délibérées.

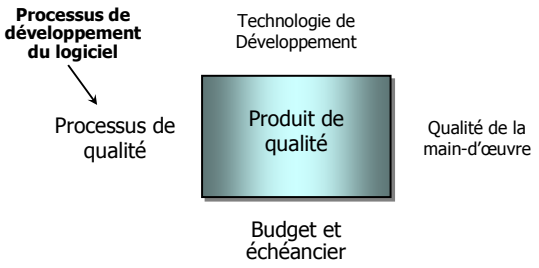
Dependabilité des Systèmes critiques

- Pour des systèmes critiques, il est normal que la propriété la plus importante du système est qu'il soit digne de confiance.
- Types de systèmes critiques :
  - ***Système Sûreté-critique*** – un système dont les pannes peuvent résulter en blessure, perte de vie ou dommage majeur de l'environnement.
    - ✓ e.g. un système de distribution d'insuline.
  - ***Système Mission-critique*** – un système dont les pannes peuvent résulter en une panne de quelques activités dirigées cibles.
    - ✓ e.g. un système de navigation pour un véhicule spatial.
  - ***Système Affaire-critique*** – un système dont les pannes peuvent résulter dans une panne des opérations utilisant le système.
    - ✓ e.g. un système de comptes-clients dans une banque.

Est-ce que l'usage et le temps peuvent causer une dégradation de la qualité d'un produit logiciel ?

- Par définition, ne devraient pas, mais …
- Un programme fonctionnant continuellement pour une longue période de temps (sans s'arrêter) peut fonctionner de plus en plus lentement ou même planter.
  - e.g. À cause d'une fuite ou une fragmentation de mémoire.
  - Heureusement, la qualité originale est restaurée en fermant et repartant le système.
  - Connaissez-vous des produits comme cela ?
- La performance décroît avec le nombre d'utilisateurs courants et la taille des données.
  - Peut nécessiter une mise à jour du matériel et, conséquemment, du logiciel .
- La maintenabilité décroît avec le temps.
  - Peut nécessiter une maintenance préventive (migration vers de nouvelles technologies, etc).
- Un logiciel devient désuet très rapidement.
  - À cause de la rapidité de l'évolution de la technologie, des exigences ou de la connaissance.
  - Parfois, le logiciel est utilisé pour un temps plus long que prévu (souvenez-vous du bogue Y2K).
  - Nécessite une innovation et une évolution continuelles.

Les facteurs principaux de la qualité du produit (1)



Les facteurs principaux de la qualité du produit (2)

- Processus de la qualité
  - Un bon processus est normalement requis pour produire un bon produit.
  - Pour des objets manufacturés, le processus est la principale qualité déterminante.
  - Pour une activité basée conception (comme le développement de logiciel), d'autres facteurs sont aussi impliqués spécialement les capacités des concepteurs.
  - Pour des **projets à grande étendue** avec des capacités « moyennes », le processus de développement détermine la qualité du produit.
- Qualité de la main-d'œuvre
  - Pour de **petits projets**, les capacités des développeurs sont le déterminant principal.
  - Conséquence : vous avez besoin d'une plus faible qualité de main-d'œuvre (et d'un meilleure processus de qualité) pour les projets à grande étendue ?
  - Taille du projet x Qualité de la main-d'œuvre = Constant ?
- Technologie de développement
  - Elle est particulièrement significative pour les **petits projets**.
- Budget et échéancier
  - Dans **tous les projets**, si un échéancier irréaliste est imposé, alors la qualité du produit en souffrira.

Les attributs de la qualité du processus

Processus caractéristique	***Description***
Facilité de compréhension	À quel degré, le processus est-il défini explicitement et avec quelle facilité comprend-on la définition du processus ?
Visibilité	Est-ce que les activités du processus culminent en résultats clairs de telle manière que le progrès du processus est extérieurement visible ?
Degré du support	À quel degré les activités du processus sont supportées par les outils CASE ?
Acceptabilité	Est-ce que le processus défini est acceptable et utilisable par des ingénieurs responsables de la production de produit logiciel ?
Fiabilité	Est-ce que le processus est créé de manière à ce que les erreurs de processus soient évitées ou captées avant qu'elles ne résultent en erreurs de produit ?
Robustesse	Est-ce que le processus peut continuer malgré des problèmes inattendus ?
Facilité de maintenance	Est-ce que le processus peut refléter le changement organisationnel pour les exigences ou identifie-t-il le développement du processus ?
Rapidité	À quelle vitesse à partir des spécifications, le processus de livraison du système peut-il être complété?

Qualité de service

- Quelques services reliés au produit et à leurs attributs de qualité.
  - Formation de l'utilisateur.
  - Aide pour l'utilisateur.
    - ✓ Réponse rapide et utile (évitez « L'aide n'aide pas »).
  - Réparation et exploitation de nouvelles versions du produit.
    - ✓ Réparation rapide et efficace.
    - ✓ Qualités de conservation :
      - Les choses qui fonctionnaient bien dans l'ancienne version doivent continuer à bien fonctionner dans la nouvelle version (les tests de régression sont très importants ici) et ne requièrent pas une nouvelle formation de l'utilisateur.
      - L'installation de la nouvelle version ne cause pas la perte de données de l'utilisateur (compatibilité descendante).
      - L'installation de la nouvelle version ne requiert pas l'arrêt du système pour une trop longue durée.
    - ✓ Progrès des qualités :
      - Les choses qui fonctionnaient mal ou qui ne fonctionnaient pas du tout dans l'ancienne version maintenant fonctionnent bien ou de nouvelles fonctionnalités utiles ont été ajoutées.
- Pas visé dans cette présentation (se concentre plus sur le produit que sur le service).

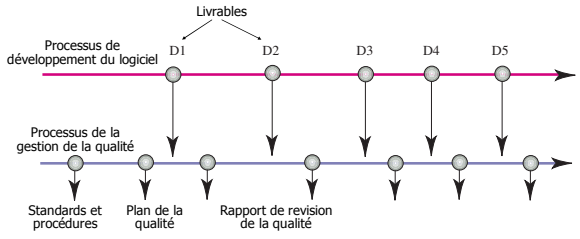
Les activités reliées à la qualité (1)

- Vérification et Validation du logiciel (V & V).
  - Buts :
    - ✓ Établir l'existence d'imperfections dans le produit.
    - ✓ Évaluer si le produit est utilisable ou pas dans une situation opérationnelle.
  - **Vérification**
    - ✓ S'assurer que nous **construisons le produit correctement** en accord avec les spécifications.
  - **Validation**
    - ✓ S'assurer que nous **construisons le produit correctement** en accord avec les besoins de l'utilisateur.
  - V & V sont une partie intégrale du processus de développement.
  - Elles sont en rapport direct avec la qualité du produit.

Les activités reliées à la qualité (2)

- **Gestion de la qualité logicielle (Software Quality Management (SQM))** : s'assurer que le niveau requis de qualité est atteint dans les produits logiciels :
  - S'assurer que les standards et procédures définis sont suivis.
  - La SQM devrait viser à développer une « culture de qualité » où la qualité est vue comme la responsabilité de tout le monde.
  - Sous-activités :
    - ✓ (Niveau organisation) **Assurance qualité (Quality assurance (QA))**
      - Établir des procédures et standards organisationnels pour la qualité dans un manuel de la qualité.
    - ✓ (Niveau projet) **Planification de la qualité**
      - Sélectionner des procédures et standards applicables pour un projet particulier et les modifier selon le besoin. Produire un plan de qualité.
    - ✓ (Niveau projet) **Contrôle de la qualité (Quality control (QC))**
      - S'assurer que les procédures et standards sont suivis par l'équipe de développement du logiciel. Produire des rapports d'analyse de la qualité.
  - La gestion de la qualité devrait être séparée de la gestion de projet pour assurer l'indépendance du budget et des pressions de l'échéancier.
  - Directement liée à la qualité du processus et indirectement à la qualité du produit.

Gestion de la qualité et développement du logiciel



Approches principales pour la V&V et le QC

- Tests
  - La technique dynamique, consiste à exercer et observer le comportement du produit pour en découvrir les **défauts**.
  - Le système est exécuté avec des données de test (définis par les cas de test) et son comportement opérationnel est observé pour trouver les défauts (différences entre observées et prévues).
  - Le test des GUI est difficile à automatiser; le test des API est plus facile à automatiser.
- Inspections et révisions
  - La technique statique – est relative à l'analyse de la représentation statique du système (code source, documentation, …) pour découvrir les **problèmes**.
  - Peut être un supplément par un document basé sur outil et l'analyse de code.
- Mesure
  - La valeur des métriques définies est mesurée automatiquement sur les composants du produit sélectionnés pour les objectifs de prédiction ou de contrôle.
  - Utilisées principalement pour le QC.
- Tous incluent la planification, l'exécution, l'analyse des résultats et rapports.

Assurance qualité et les standards

- Les standards sont la clé d'une gestion efficace de la qualité.
- Ce peut être des standards internationaux, nationaux, organisationnels ou pour un projet.
- Les standards de produit définissent les caractéristiques que tous les composants devraient montrer e.g. un style de programmation répandu.
- Les standards de processus définissent comment le processus logiciel devrait être décrété.

Importance des standards

- Encapsulation de la meilleure pratique – évite la répétition des erreurs passées.
- Structure (framework) pour le processus de l'assurance qualité – elle implique la vérification de la conformité des standards.
- Fournit la continuité – le nouveau personnel peut comprendre l'organisation en comprenant les standards appliqués.

Problèmes avec les standards

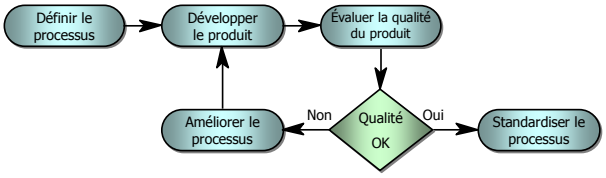
- Pas vus comme pertinents et à jour par les ingénieurs logiciels.
  - Les consultants devraient être impliqués dans le développement. Les ingénieurs devraient comprendre la théorie sous-jacente d'un standard.
  - Les standards et leur usages devraient être **révisés régulièrement**. Ils peuvent être rapidement dépassés et cela réduit leur crédibilité chez les consultants.
- Impliquent trop de formulaires bureaucratiques à remplir.
- Non supportés par les manuels de travail ennuyeux des outils logiciels qui sont impliqués pour maintenir les standards.
  - Les standards détaillés devraient être associés à un support d'outil. Un travail fastidieux et excessif est la plainte la plus significative contre les standards.

Documentation des standards

- Particulièrement importante – les documents sont la manifestation tangible du logiciel.
- Standards du processus de documentation
  - Comment les documents devraient être développés, validés et maintenus.
- Les standards des documents
  - Concernés par l'identification, la structure, la présentation, les points marquants des changements, etc.
- Les standards d'échange de documents
  - Comment les documents sont stockés et échangés entre les différents systèmes de documentation.
  - XML est un standard émergeant pour l'échange de documentation qui sera grandement supporté dans le futur.

Développement des standards du processus

- Une attention doit être prise afin de ne pas imposer des standards inappropriés au processus.



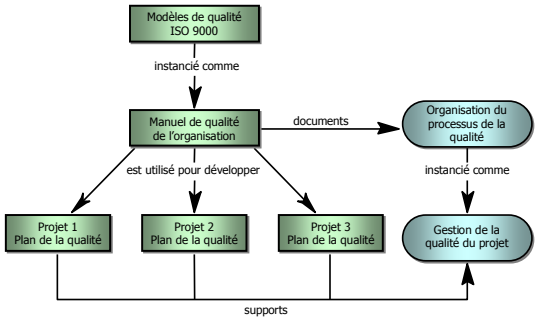
ISO 9000

- Ensemble de standards internationaux pour la gestion de la qualité (ISO 9000:2000, ISO 9001:2000, ISO 9004:2000, etc.)
- Applicable à une variété d'organisations des industries manufacturières et de service.
- ISO 9001:2000 spécifie les besoins pour un système de gestion de qualité pour n'importe quelle organisation qui a besoin de démontrer ses habilités de fournir un **produit** qui satisfait le consommateur et les exigences rigides applicables, et vise à rehausser la satisfaction du consommateur, dans les secteurs du commerce.
  - Intègre les standards précédents ISO 9001, ISO 9002 et ISO 9003.
  - ISO 9001 est un modèle générique qui doit être instancié pour chaque organisation.
- ISO 9004:2000 fournit le conseil pour l'amélioration continuelle du système de gestion de qualité pour bénéficier à tous les partis (employés, propriétaires, fournisseurs, société en général, …) par la satisfaction continue du consommateur. Il devrait être utilisé pour étendre les bénéfices obtenus par ISO 9001:2000 pour tous les partis qui sont intéressés ou affectés par les opérations de gestion.

Certification ISO 9000

- Les standards et procédures de qualité devraient être documentés dans un manuel de qualité de l'organisation
- Un corps externe peut certifier que le manuel de qualité de l'organisation se conforme aux standards ISO 9000 (c'est-à-dire ISO 9001).
- Les clients exigent de plus en plus que les fournisseurs soient certifiés ISO 9000.

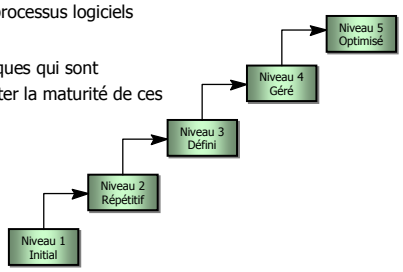
ISO 9000 et la gestion de la qualité



L'institut de génie logiciel (Software Engineering Institute (SEI)) Modèle de la maturité de capacité pour les logiciels (Capability Maturity Model for Software (CMM))

Est un modèle pour

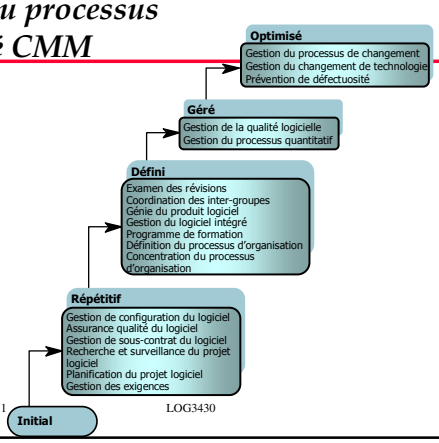
- juger la maturité des processus logiciels d'une organisation,
- identifier les clés pratiques qui sont requises pour augmenter la maturité de ces processus.



Niveaux de maturité du CMM

- **1) Initial.** Le processus logiciel est caractérisé comme étant ad hoc et occasionnellement même chaotique. Peu de processus sont définis et le succès dépend des efforts individuels et "héroïques".
- **2) Répétitif.** Les processus de base de gestion de projet sont établis pour définir le coût, l'échéancier et la fonctionnalité. La discipline nécessaire du processus est en place pour répéter les succès précédents des projets avec des applications similaires.
- **3) Défini.** Le processus logiciel pour les deux activités de gestion et de génie est documenté, standardisé et intégré dans un processus logiciel standard pour l'organisation. Tous les projets utilisent une version approuvée et façonnée du processus logiciel standard de l'organisation pour développer et maintenir le logiciel.
- **4) Géré.** Des mesures détaillées du processus logiciel et de la qualité du produit sont assemblées. Ensemble, le processus logiciel et les produits sont compris et contrôlés quantitativement.
- **5) Optimisé.** L'amélioration continue du processus est possible grâce au feedback quantitatif du processus et de l'innovation conduite par les idées et technologies.

Zones du processus de la clé CMM



Le CMM et ISO 9000

- Il y a un lien clair entre les clés des processus dans le CMM et les processus de gestion de la qualité en ISO 9000.
- Le CMM est plus détaillé et normatif, et il inclut une structure plus détaillée pour l'amélioration.
- Les organisations évaluées comme niveau 2 dans le CMM sont susceptibles d'être conformes à ISO 9000.

Planification de la qualité

- Un plan de qualité expose (dans un projet particulier) les qualités désirées du produit et comment celles-ci sont évaluées. Il définit les attributs de qualité les plus significatifs.
- Il devrait définir le processus d'évaluation de la qualité.
- Il devrait exposer quels standards organisationnels devraient être appliqués et, si nécessaire, définir les nouveaux standards.
- Les plans de qualité devraient être courts; des documents succincts.
  - S'ils sont trop longs, personne ne va les lire.

Contrôle de qualité

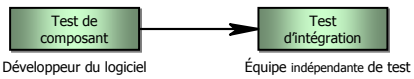
- Vérifier le processus de développement de logiciel (dans un projet particulier) pour assurer que les procédures et standards, comme définis dans le plan de qualité, sont suivis.
- Deux approches pour le contrôle de qualité.
  - (Manuel) Révisions de la qualité – approche principale.
  - (Automatisé) Mesure de la qualité.

Test de boîte noire et de boîte blanche

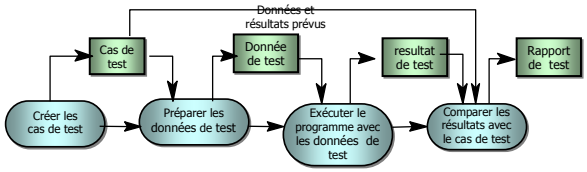
- Test de boîte noire
  - Une approche pour tester, où le programme est considéré comme une « boîte noire ».
  - Les cas de test du programme sont basés sur les spécifications du système.
  - La planification des tests peut commencer tôt dans le processus logiciel.
- Test de boîte blanche
  - Parfois appelé test structurel.
  - La dérivation des cas de test est guidée par la structure du programme. La connaissance du programme est utilisée pour identifier les cas de test additionnels.
  - L'objectif minimal est d'exécuter toutes les déclarations du programme (pas toutes les combinaisons de chemins).

Test des composants et Intégration

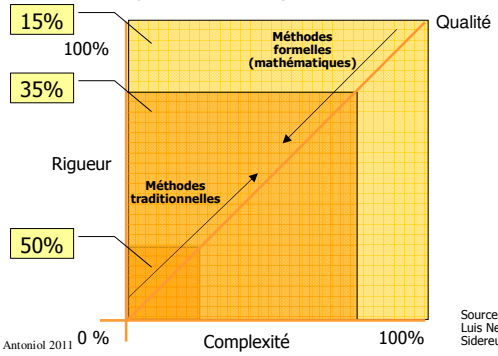
- Test des composants
  - Test des composants du programme individuellement.
  - Normalement la responsabilité du développeur du composant (excepté quelques fois pour des systèmes critiques).
  - Les tests sont générés selon l'expérience des développeurs.
- Test d'intégration
  - Test de groupes de composants intégrés pour créer un système ou sous-système.
  - La responsabilité d'une équipe indépendante de test.
  - Les tests sont en général basés sur les spécifications du système (boîte noire).



Le processus de test d'erreur



Rigueur, complexité et qualité



Points Clefs

- La gestion de qualité logicielle à pour but de garantir que le logiciel rencontre les standards requis.
- Les procédures d'assurance qualité devraient être documentées dans un manuel de la qualité de l'entreprise.
- Les logiciels standards renferment les meilleures pratiques.
- Les test et les révisions sont les approches les plus utilisées pour évaluer la qualité logicielle.

Points Clefs

- Le mesurage logiciel collecte les informations sur le processus logiciel et sur le produit logiciel.
- Les métriques de la qualité du produit devraient être utilisées pour identifier les composants problématiques.
- Il n'y a pas de métriques logicielles applicables qui soient normalisées et universelles.

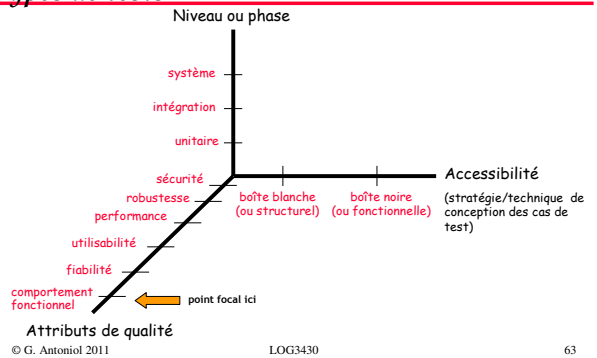
Test du logiciel

- Le test du logiciel consiste en la vérification **dynamique**(1) du comportement du programme, dans un ensemble **fini**(2) de **cas de test** convenablement **sélectionné**(3) à partir du domaine des exécutions infini, vis-à-vis d'un comportement spécifié **prévu**(4) [source : SWEBOK].
  - (1) Le test implique toujours l'exécution du programme sur quelques données d'entrée.
  - (2) Même pour des programmes simples, il y a trop de cas de test théoriquement possibles pour qu'un test exhaustif soit possible.
    - ✓Un compromis entre les ressources limitées, les échéanciers et les exigences d'un test illimité.
  - (3) Nous avons vu les techniques de conception de cas de test et comment les sélectionner.
  - (4) Il doit être possible de décider si les résultats (les valeurs de sorties) observés du programme sont acceptables ou non.
    - ✓La décision passage/échec est communément traitée comme le **problème oracle**.

Cas de test

- Le cas de test – des données d'entrée pour tester le système et les sorties attendues (ou résultats prédits) à partir de ces données d'entrée si le système fonctionne correctement sous des conditions d'exécution spécifiées.
  - Les données d'entrée peuvent inclure un état initial du système.
  - Les sorties peuvent inclure un état final du système.
  - Quand les cas de test sont exécutés, le système reçoit les données spécifiées en entrée; les sorties sont en suite comparées avec les sorties attendues.
- Exemple pour une calculatrice :
  - 3 + 5 (entrée) *devrait donner* 8 (sortie attendue ou prévue)

Types de tests



Qualité

- La qualité signifie « conformité aux besoins ».
  - Même les meilleurs testeurs ne peuvent détecter que les erreurs qui contredisent la spécification.
  - Le test ne rend pas le logiciel parfait.
  - Pour une compagnie qui n'a pas de bonnes pratiques d'ingénierie de besoins, il est très difficile de livrer un logiciel qui remplit les besoins de l'utilisateur, parce que l'équipe de production ne connaîtra pas vraiment ce que sont ces besoins.

Plans de test

- Le but de la planification de test est d'établir la liste de tâches qui, si exécutées, identifiera toutes les exigences qui n'ont pas été satisfaites dans le logiciel. Le produit est le *plan de test.*
  - Le plan de test documente l'approche de test dans l'ensemble. Dans plusieurs cas, le plan de test sert de sommaire des activités de test qui seront exécutées.
  - Il montre comment les tests seront organisés et donne un aperçu des besoins que doivent satisfaire les testeurs afin que le test soit proprement exécuté.
  - Le plan de test devrait être inspecté par des membres de l'équipe de génie et des gestionnaires principaux.

Cas de Test

- Un *cas de test* est la description d'une interaction spécifique qu'un testeur aura afin de tester un comportement simple du logiciel. Les cas de test peuvent parfois être très similaires aux cas d'utilisation (use case), dans la mesure où ils sont le récit étape par étape d'une interaction spécifique entre l'usager et le logiciel.
  - Un cas de test typique est décrit dans une table et inclut :
    - ✓ Un ***nom*** et un ***nombre*** unique.
    - ✓ Un ***besoin*** pour lequel ce cas de test est exercé.
    - ✓ Des ***pré-conditions*** qui décrivent l'état du logiciel avant le cas de test (qui est souvent un cas de test précédent qui doit toujours être exécuté avant le cas de test courant).
    - ✓ Des ***étapes*** qui décrivent les étapes spécifiques constituant l'interaction.
    - ✓ Des ***résultats prévus*** qui décrivent l'état **attendu** du logiciel après que le cas de test ait été exécuté.
- Les cas de test doivent être répétables.
  - Les bons cas de tests sont des données spécifiques et décrivent chaque interaction nécessaire pour répéter exactement le test.

Exécution de test

- Les testeurs de logiciel commencent à exécuter le plan de test après que les programmeurs aient livré la ***version alpha***, ou une version qui leur semble être complète.
  - La version alpha devrait être de haute qualité – les programmeurs devraient approuver que c'est une version prête à être distribuée, et qu'elle soit le mieux qu'ils peuvent obtenir.
- Il y a typiquement plusieurs itérations de l'exécution des tests.
  - La première itération se concentre sur les nouvelles fonctionnalités qui ont été ajoutées depuis la dernière série de tests.
  - Un ***test de régression*** est un test créé pour être certain qu'un changement dans une zone du logiciel n'engendre pas un mauvais fonctionnement d'une ou d'autres parties du logiciel qui ont été testée antérieurement.
  - Le test de régression implique normalement l'exécution de tous les cas de test exécutés antérieurement.
  - Il y a typiquement au moins deux tests de régression pour tout projet logiciel.

Le traçage des erreurs

- Le ***système de traçage d'erreur (bug tracking system)*** est un programme que les testeurs utilisent pour enregistrer et détecter les erreurs. Il suit le chemin de chaque erreur entre les testeurs, les développeurs, le gestionnaire de projet et les autres, suivant diagramme de flux d'information conçu pour s'assurer que l'erreur est vérifiée et réparée.
  - Chaque erreur rencontrée dans l'exécution du test est enregistrée et entrée dans le système pour détecter les erreurs afin qu'elles puissent être classées par priorité.
  - Le diagramme de flux d'informations sur les défauts devrait montrer l'interaction entre les testeurs qui ont trouvé l'erreur et les programmeurs qui l'ont fixée.
  - Chaque erreur doit être proprement classée par priorité et révisée par tous les « stakeholders » pour déterminer si elle doit être corrigée ou non. Ce processus de révision et de classement par priorité est appelé ***triage.***

Test de fumée (Smoke Tests)

- Un « ***test de fumée » (smoke test***) est un sous-ensemble de cas de test qui est typiquement représentatif de l'ensemble du plan de test.
  - Les tests de fumée sont bons pour vérifier les déploiements propres ou les autres changements non invasifs.
  - Ils sont aussi utiles pour vérifier si une version est prête à être testée.
  - Les tests de fumée ne sont pas un remplacement du test fonctionnel actuel.
  - Ce sont des tests primitifs pour vérifier la bonne marche d'un système (le nom vient des constructeurs de hardware)

Automatisation du Test

- ***L'automatisation du test*** est une pratique par laquelle les testeurs emploient un outil logiciel pour réduire ou éliminer les tâches répétitives.
  - Soit que les testeurs écrivent des scripts ou soit qu'ils utilisent un enregistrement et une lecture pour capter les interactions de l'usager avec le logiciel qui est testé.
  - Ceci peut sauver beaucoup de temps aux testeurs si plusieurs itérations de test seront requises.
  - Développer et maintenir des suites de tests automatisés coûte cher, c'est pourquoi cela ne vaut généralement pas la peine de les développer pour des tests qui ne s'exécuteront que seulement quelquefois.

Rapports d'autopsie

- Le *rapport d'autopsie* (*postmortem report)* est un compte rendu de l'expérience de l'équipe qui a construit le logiciel, et l'expérience des usagers et des « stakeholders » travaillant avec l'équipe.
  - Le rapport devrait contenir une déclaration honnête de comment les membres de l'équipe, les usagers et les « stakeholders » perçoivent le produit final et évaluer les décisions faites durant le projet.
  - Le but du rapport d'autopsie est de mettre l'emphase sur les succès de l'équipe et d'identifier n'importe quel problème qui devrait être corrigé dans les versions futures.

Niveaux ou phases de test (1)

- Test unitaire
  - Tester les unités ou composants d'un programme individuels.
  - Normalement la responsabilité du développeur du composant (à l'exception parfois des systèmes critiques).
  - Les tests sont basés sur l'expérience, les spécifications et le code.
  - Un but principal est de détecter les erreurs fonctionnelles et structurelles dans l'unité.

Niveaux ou phases de test (2)

- Test d'intégration
  - Tester des groupes de composants intégrés pour créer un sous-système.
  - Normalement, la responsabilité d'une équipe de test indépendante (à l'exception parfois dans de petits projets).
  - Les tests sont basés sur les spécifications d'un système (spécifications techniques, conceptions).
  - Le but principal est de détecter les erreurs qui se produisent dans les interfaces d'unités et leur comportement commun.

Niveaux ou phases de test (3)

- Test système
  - Tester le système en entier.
  - Normalement, la responsabilité d'une équipe de test indépendante.
  - Les tests sont normalement basés sur un document (des requis fonctionnels/spécifications et requis de qualité).
  - Un but principal est d'évaluer les attributs tels que l'utilisabilité, la fiabilité et la performance (assumons que le test unitaire et d'intégration aient été exécutés).

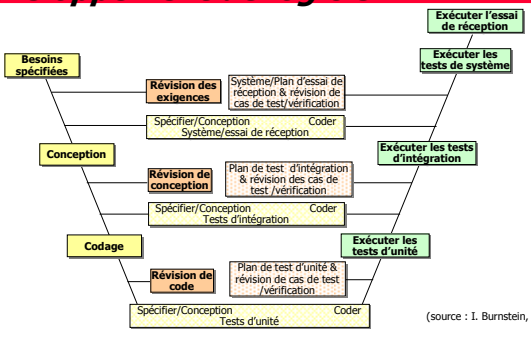
(source: I. Sommerville)

Niveaux ou phases de test (4)

- Test de réception (*Acceptance testing*)
  - Tester le système en entier.
  - Normalement la responsabilité du consommateur.
  - Les tests sont basés une spécification requise ou un manuel d'utilisateur.
  - Un but principal est de vérifier si le produit rencontre les exigences et les attentes du client.
- Contrôle par régression (*Regression testing*)
  - Répétition de tests à n'importer quel niveau après un changement logiciel.

(source: I. Sommerville)

Niveaux de test le modèle V du développement de logiciel

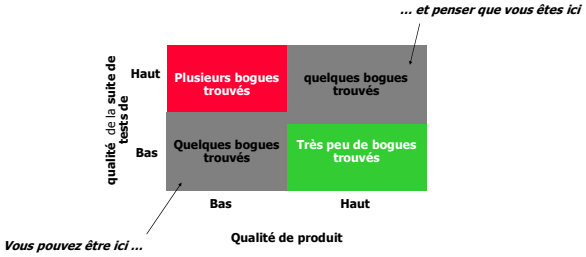


(source : I. Burnstein, pp.15)

Qu'est-ce qu'un bon jeu de test ?

- Capacité de trouver des erreurs.
  - Particulièrement les erreurs à haut risque.
  - Risque = fréquence de défaillance (manifestation pour l'usagers) * impact d'échec.
  - Coût (de défaillance post-distribution) ≈ risque
- Capacité d'exécuter des aspects multiples du système sous test.
  - Réduit le nombre de cas de test requis et le coût.
- Coût bas.
  - Développement : spécification, conception, code.
  - Exécution.
  - Analyse résultante : analyses passage/échec, erreur de localisation.
- Facile d'entretien.
  - Réduit le coût de tout le cycle de vie.
  - Coût de la maintenance ≈ taille des artéfacts de tests.

L'importance d'un bon cas de test



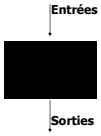
Critères de sélection de test/satisfaction/couverture/arrêt

- Critères d'adéquation – Des critères pour décider si un jeu de tests donné est adéquat, i.e., pour nous donner « assez » de confiance que « la plupart » des erreurs sont révélées.
  - En pratique, revient a réaliser le maximum de couverture des critères.
- Couverture des critères : Requis/spécification de la couverture.
  - ✓ Au moins un cas de test pour chaque requis/besoin.
  - ✓ Couvrir toutes les déclarations dans une spécification formelle.
  - Couverture du modèle.
    - ✓ Couverture état-transition.
    - ✓ Use-case et couverture de scénario.
  - Couverture de code.
    - ✓ Couverture d'instructions,
    - ✓ Couverture du flux de données, …
  - Couverture des erreurs.

« Même s'il est commun dans la pratique de test de logiciels courants que les processus de test aux deux niveaux haut et bas s'arrêtent quand l'argent ou le temps sont manquants, il y a une tendance d'utiliser des méthodes de tests systématiques avec l'application de critères d'adéquation de test. »

Software Unit Test Coverage and Adequacy, Hong Zhu et al, ACM Computing Surveys, December 1997

Stratégies et techniques de conception de cas test

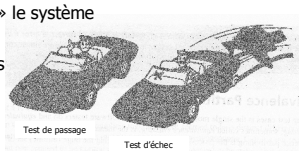
Stratégies	Vision du testeur	Sources de connaissance	Techniques / Méthodes
Test de boîte noire (non basé sur le code) (parfois appelé test fonctionnel)		Document des besoins Les spécifications Manuel d'utilisateur Modèles Connaissance du domaine Données d'analyses des erreurs Intuition Expérience	Classes d'équivalence Analyses des valeurs limites Graphe cause à effet Deviner les erreurs Test aléatoire Test état-transition Test basé sur le scénario
Test de boîte blanche (aussi dit "code based" ou test structurel)		Code source du programme Graphes du flux de contrôle Graphes du flux de données Complexité cyclomatique Conception haut niveau Conception détaillée	Test/couverture du flux de contrôle : - Couverture de déclaration - Couverture des branches (ou décisions) - Couverture des conditions - Couverture des branches et des conditions - Couverture des conditions/décisions modifiée - Couverture conditions multiples - Couverture chemin indépendant - Couverture des chemins Test/couverture du flux de données Couverture/test des classes Test par mutation

(adapted from: I. Burnstein, pg.65)

Test d'itérations : test de passage et test d'échec

- Premières itérations du test : Test de passage (Test-to-pass)
  - Vérifier si le logiciel fonctionne fondamentalement
  - avec des données valides
  - sans stresser le système
- Test d'itérations subséquent : Test d'échec (Test-to-fail)
  - Essayer de « mettre en panne » le système
  - avec des données valides mais aux limites opérationnelles
  - avec des données invalides

(source : Ron Patton)



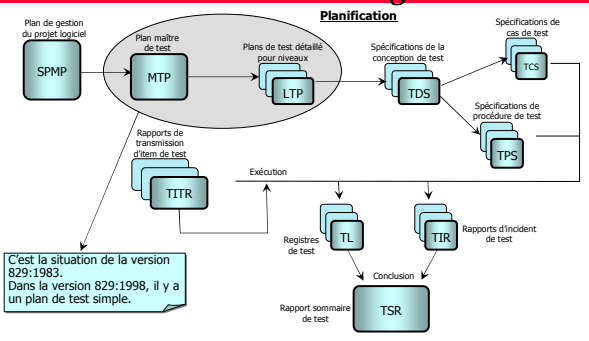
Automatisation du Test

- Exécution automatique des cas de test
  - Exige que les cas de test soient écrits dans un langage exécutable.
  - Augmente les coûts des tests de développement (codage) mais élimine pratiquement les coûts de (ré)exécution des tests qui sont particulièrement importants dans le test de régression.
  - Structures et outils d'unité de test pour tester l'API.
  - Outils de capture/relecture pour tester IGU (GUI).
- Génération automatique des cas de test
  - La génération automatique de données de test (entrées) est plus facile que la génération automatique de des sorties attendues (normalement exige une spécification formelle).
  - Réduit les coûts de test de développement.
  - Normalement a une capacité inférieure de détecter les erreurs par cas de test, mais l'ensemble des données de test peut être généré automatiquement (plutôt que manuellement).

Documents de qualité et de test

IEEE std 829-1998 - IEEE Std 730-1998

Le standard 829-1998 IEEE pour la documentation de test de logiciel



Standard IEEE (829-1998) - Les documents types 1/3 – plan

- Planification de test :
  - **Plan de test**
    - ✓Prescrit l'envergure, l'approche, les ressources et l'échéancier des activités de test.
    - ✓Identifie les items à tester, les fonctionnalités à tester, les tâches de test à compléter, le personnel responsable de chaque tâche et les risques associés avec le plan.

Les documents types 829-1998 standard IEEE (2/3) - Spécification

- **Spécification de la conception du test**
  - ✓Raffine l'approche de test et identifie les caractéristiques devant être couvertes par la conception et ses tests associés.
  - ✓*Identifie* les cas de test et les procédures de test, s'il y en a, requis pour accomplir le test et spécifier les critères de passage/échec.
- **Spécification du cas de test**
  - ✓Documente les valeurs réelles utilisées comme entrées avec les sorties anticipées.
  - ✓Identifie aussi les contraintes sur les procédures de test résultant de l'utilisation du cas de test spécifique.
- **Spécification de la procédure de test**
  - ✓Identifie toutes les étapes requises pour utiliser le système et exercer les cas de test spécifiés dans le but d'implémenter la conception de test associée.

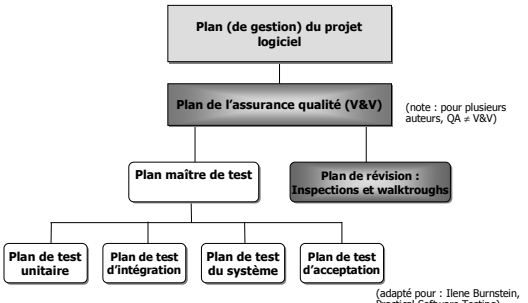
Les documents types 829-1998 standards IEEE (3/3) - Rapport

- **Rapport de transmission de l'item test**
  - ✓Identifie les items de test qui ont été transmis pour le test, dans l'éventualité où le développement séparé et le test de groupes sont impliqués, ou si un début formel de l'exécution de test est désiré.
- **« Log de Test »**
  - ✓Utilisé par l'équipe de test pour enregistrer ce qui s'est produit durant l'exécution du test.
- **Rapport d'incident de test**
  - ✓Décrit tous les événements qui se sont produits durant l'exécution qui requiert une investigation supplémentaire.
- **Rapport sommaire de test**
  - ✓résume les activités de test associées avec une ou plusieurs spécifications de conception de test.

Note ...

- Les cas de test sont séparés des conceptions de test afin de permettre l'utilisation dans plus d'une conception et pour permettre de les réutiliser dans d'autres situations.
- Les procédures de test sont séparées des spécifications de conception de test puisqu'elles sont destinées à être suivies étape par étape et ne devraient pas avoir de détails supplémentaires.

Plan du projet, plan de l'assurance qualité et plan de test



(adapté pour : Ilene Burnstein, Practical Software Testing)

Standard IEEE pour les plans d'assurance qualité de logiciel (Std IEEE 730-1998)

- **Contenu d'un SQAP recommandé** :
  1. But;
  2. Documents de référence;
  3. Gestion;
     - ✓ Décrit l'organisation, les tâches et les responsabilités.
  4. Documentation :
     a) Identifier la documentation gouvernant le développement, la vérification et la validation, l'utilisation et la maintenance du logiciel ;
     b) Détermine comment les documents doivent être vérifiés vis-à-vis leur adéquation. Cela devra inclure les critères et l'identification de la révision ou de la vérification par lesquelles l'adéquation de chaque document devra être confirmée, en faisant référence à la section 6 du SQAP.
  5. Standards, pratiques, conventions, et métriques :
     a) Identifier les standards, les pratiques, les conventions et les métriques qui doivent être appliquées ;
     b) Déterminer comment la conformité de ces items doit être contrôlée et assurée.

Standard IEEE pour les plans d'assurance qualité de logiciel (Std IEEE 730-1998)

6. Revues et vérifications (voir Std IEEE 1028-1997) :
   a) Définir les revues et les vérifications techniques et de gestion qui doivent être menées ;
   b) Déterminer comment les revues et les vérifications doivent être accomplies ;
   c) Déterminer quelles actions supplémentaires sont requises et comment elles doivent être implémentées et vérifiées.
7. Test (voir Std IEEE 829-1998 et Std IEEE 1008-1987) :
   - ✓ Identifier tous les tests non inclus dans le SVVP (Soft. Verification & Validation Plan) pour le logiciel couvert par le SQAP et déterminer les méthodes à utiliser.
8. Problème rapporté et action corrective :
   a) Déterminer les pratiques et procédures à suivre pour rapporter, tracer et identifier à la fois les items logiciels et le processus de développement et de maintenance du logiciel;
   b) Déterminer les responsabilités organisationnelles spécifiques concernées par leur implémentation.

Standard IEEE pour les plans d'assurance qualité de logiciel (Std 730-1998 IEEE)

9. Outils, techniques et méthodologies :
   - ➢ Identifier les outils logiciels spéciaux, les techniques et les méthodologies qui supportent le SQA, déterminer leurs buts et décrire leur usage.
10. Contrôle du code :
    - ➢ Définir les méthodes et les moyens utilisés pour maintenir, stocker, sécuriser et documenter les versions contrôlées du logiciel identifié, durant toutes les phases du cycle de vie du logiciel ;
    - ➢ Ceci peut être implémenté conjointement avec une librairie de programmes
    - ➢ Ceci peut être fourni comme une partie du SCMP. Si c'est le cas, une référence appropriée devra être faite là aussi.
11. Contrôle du media (support de stockage) :
    - ➢ Identifier le media pour chaque produit informatique et la documentation requise pour stocker le media, incluant la copie et le processus de restauration ;
    - ➢ Protéger le media physique des programmes informatiques de l'accès non autorisé ou des dommage et des dégradation involontaires pendant toutes les phases du cycle de vie du logiciel.

Standard IEEE pour les plans d'assurance qualité de logiciel (Std IEEE 730-1998)

12. Contrôle de fournisseur :
    - Déterminer les provisions pour assurer que les logiciels fournis par les fournisseurs rencontrent les requis établis ;
    - Déterminer les méthodes qui seront utilisées pour assurer que le fournisseur du logiciel reçoive les requis adéquats et complets. Pour un logiciel développé préalablement, cette section devra déterminer les méthodes à utiliser pour assurer la convenance du produit dans son utilisation avec les items logiciels couverts par le SQAP ;
    - Pour un logiciel qui sera développé ultérieurement, le fournisseur devra nécessairement préparer et implémenter un SQAP en accord avec ce standard ;
    - Déterminer les méthodes qui devront être employées qui assureront que les développeurs se conforment avec les exigences de ce standard.
13. Enregistrement, collecte, maintenance, et préservation:
    - Identifier la documentation du SQA qui doit être retenue ;
    - Déterminer les méthodes et les facilités qui seront utilisées pour assembler, sauvegarder et maintenir cette documentation ;
    - Désigner la période de conservation.
14. Formation :
    - Identifier les activités de formation nécessaires pour rencontrer les besoins du SQAP.
15. Gestion du risque:
    - Spécifier les méthodes et procédures employées pour identifier, estimer, surveiller et contrôler les périodes du risque qui peut survenir durant la portion du cycle de vie couverte par le SQAP.

IEEE 829-1998

Contenus du plan de test

Standard IEEE 829-1998 – Contenus du plan de test

Applicable au plan maître de test et à chacun des niveaux basés sur les plans de test (unitaire, intégration, etc.) :

1. Identificateur du plan de test :
    - Peut servir pour l'identifier comme un item de configuration.
2. Introduction *(pourquoi)* :
    - Description globale du projet, le système logiciel en cours de développement ou de maintenance et les items logiciels et/ou caractéristiques à être testés
    - Description globale des buts de test (objectifs) et des approches de test à utiliser
    - Références aux documents reliés ou supportés.
3. Items de test *(quoi)* :
    - Lister les items à tester : procédures, classes, modules, bibliothèques, composants, sous-systèmes, systèmes, etc. ;
    - Inclure les références dans les documents où ces items et leurs comportements sont décrits (requis et documents de conception, manuels d'utilisateur, etc.) ;
    - Lister aussi les items qui ne seront pas testés.

Standard 829-1998 IEEE – Contenus du plan de test

4. Fonctionnalités à tester *(quoi)* :
    - Les fonctionnalités sont des caractéristiques distinctes (fonctionnalités, attributs de qualité). Elles sont fortement reliées à la manière avec laquelle nous décrivons le logiciel en termes de ses requis fonctionnels et de qualité
    - Identifier toutes les fonctionnalités logicielles et les combinaisons de fonctionnalités à tester. Identifier les spécifications du test de conception associées à chaque fonctionnalité et chaque combinaison de fonctionnalités.
5. Fonctionnalités à ne pas tester (*quoi*) :
    - Identifier toutes les fonctionnalités et combinaisons de fonctionnalités significatives qui ne seront pas testées et les raisons de ceci.

Standard IEEE 829-1998 –Contenus du plan de test

6. Approche *(Comment)* :
   - Description des activités de test, de telle manière que les tâches majeures de test et les durées de test peuvent être identifiées ;
   - Pour chaque fonctionnalité ou combinaison de fonctionnalités, l'approche qui sera utilisée pour assurer que chacune est testée adéquatement ;
   - Outils et techniques.
   - Les prévisions pour la complétude du test (tel que le degré de la couverture de code pour les tests boîte blanche) ;
   - Test des contraintes, telles que les limitations de temps et budget ;
   - Les critères d'arrêt de test.
7. Critères de suucès-échec d'item :
   - Étant donné un item de test et un cas de test, le testeur doit avoir un ensemble de critères pour décider si le test a passé ou échoué à l'exécution ;
   - Le plan de test devrait fournir une description générale de ces critères ;
   - Les échecs à un certain niveau de sévérité peuvent être acceptés.

Standard IEEE 829-1998 –Contenus du plan de test

8. Critères de suspension et requis de reprise :
   - Spécifier les critères utilisés pour suspendre toute ou une partie de l'activité des test des items de test associés à ce plan ;
   - Spécifier les activités de test qui doivent être répétées quand un test est repris ;
   - Le test est fait sous forme de cycles : tester – réparer – (reprise) (suspension) le test– réparer - ...
   - Les tests peuvent être suspendus quand un certain nombre de défauts critiques ont été observés.
9. Livrables de test :
   - Test de documents (possiblement un sous-ensemble de ceux décrits dans le standard IEEE) ;
   - Harnais de test (drivers, stubs, outils développés spécialement pour ce projet, etc.).
10. Tâches de test
    - Identifier toutes les tâches reliées au test, les dépendances inter-tâche et les compétences spéciales requises ;
    - *Work Breakdown Structure* (WBS): (Structure éclatée - Structure de découpage du projet)

Standard 829-1998 IEEE – Contenus de plan de test

11. Besoins environnementaux :
    - Les logiciels et le matériel nécessaires pour l'effort de test.
12. Responsabilités :
    - Rôles et responsabilités;
    - Personnel réellement impliqué (?).
13. Besoins de recrutement et de formation:
    - Description des besoins en personnel et les compétences nécessaires pour accomplir les responsabilités reliées au test.
14. Échéancier :
    - Durées et échéancier de la tâche ;
    - Événements marquants (Jalons);
    - Planification pour l'usage du personnel et autres ressources (outils, laboratoires, etc.).

Standard IEEE 829-1998 – Contenus du plan de test

15. Risques et contingences
    - Les risques devraient être (i) identifiés, (ii) évalués en termes de leur probabilité de se produire, (iii) assignés des niveaux de priorité et (iv) les plans de contingences doivent être développés pour être activés si le risque se produit.
    - Exemple d'un risque : quelques items de test ne sont pas livrés à temps aux testeurs.
    - Exemple d'un plan de contingence : la flexibilité dans l'allocation des ressources de telle manière que les testeurs et l'équipement peuvent opérer au-delà des heures normales de travail (pour rattraper les retards de livraison).

Standard IEEE 829-1998–Contenus du plan de test

16. Coûts de test (ne sont pas inclus dans le standard IEEE) :
    - Les types de coûts :
      - ✓ les coûts de planification et conception des tests ;
      - ✓ les coûts de l'acquisition du matériel et du logiciel nécessaire ;
      - ✓ les coûts de l'exécution des tests ;
      - ✓ les coûts de l'enregistrement et de l'analyse des résultats des tests ;
      - ✓ les coûts (tear-down) pour restaurer l'environnement.
    - L'estimation du coût peut être basée sur :
      - ✓ modèles (par exemple COCOMO pour les coûts de projet) et heuristiques (tel que 50% des coûts du projet) ;
      - ✓ tâches du test et WBS ;
      - ✓ le ratio développeur/testeur (par exemple 1 testeur pour 2 développeurs) ;
      - ✓ Les items qui impactent le test (comme le nombre de procédures) et le facteurs du coût du test (comme KLOC) ;
      - ✓ jugement d'expert.
17. Approbations
    - Les dates et signatures de ceux qui doivent approuver le plan de test.

Standard IEEE 829-1998 – Spécifications de conception de test

- Un ou plusieurs documents.
- Une spécification de conception de test décrit comment un groupe de fonctionnalités et/ou d'items de test, est testé par un ensemble de cas de test et des procédures de test.
- Peut inclure une matrice de traçabilité de fonctionnalités/requis.

Standard IEEE 829-1998 –Spécifications de conception de test - contenu

- Contient :
  - identificateur ;
  - fonctionnalités à tester :
    - ✓Les items de test et les fonctionnalités couvertes par ce document.
  - Raffinements de l'approche :
    - ✓Les techniques de test.
  - L'identification des cas de test.
  - Les critères succès/échec de la fonctionnalité.

Standard IEEE 829-1998 – Spécifications du cas de test

- Contient :
  - L'identificateur de la spécification du cas de test.
  - Les items de test :
    - ✓La liste des items et fonctionnalités devant être testés par ce cas de test.
  - Les spécifications des entrées.
  - Les spécifications de sorties.
  - Les besoins environnementaux.
  - Les requis procéduraux spéciaux.
  - Les dépendances inter cas.

Standard IEEE 829-1998 - Spécifications de procédure de test

- Décrit les étapes requises pour exécuter un ensemble de cas de tests ou, plus généralement, les étapes utilisées pour analyser un item logiciel en vue d'évaluer un ensemble de fonctionnalités.
- Contient :
  - l'identificateur de la spécification de procédure de test
  - le but
  - les requis spécifiques
  - les étapes de la procédure
    - ✓ Enregistrer un journal, configurer, procéder, mesurer, fermer, redémarrer, arrêter, envelopper, les contingences.

Standard 829-1998 IEEE – Rapport de transmission d'item de test

- Accompagne un ensemble d'items de test qui sont livrés pour être testés.
- Contient :
  - Identificateur du rapport de transmission ;
  - Items transmis :
    - ✓ le niveau de la version/révision ;
    - ✓ les références de la documentation des items et du plan de test relié aux items transmis ;
    - ✓ les personnes responsables pour les items.
  - Location.
  - Statut :
    - ✓ les déviations de la documentation, des transmissions antérieures ou du plan de test ;
    - ✓ les rapports d'incidents qu'on a prévu de résoudre ;
    - ✓ synchroniser les modifications et la documentation.
  - Approbations.

Standard 829-1998 IEEE – Journal de test

- Enregistre les détails des résultats de l'exécution du test.
- Contient :
  - L'identificateur du journal de test ;
  - La description :
    - ✓ identifie les items ayant été testés ,incluant leurs niveaux de version/révision ;
    - ✓ identifie les attributs des environnements dans lesquels le test a été effectué.
  - Les entrées de l'activité et de l'événement :
    - ✓ la description de l'exécution ;
    - ✓ les résultats de la procédure ;
    - ✓ l'information environnementale ;
    - ✓ les événements avec anomalies ;
    - ✓ les identificateurs du rapport de l'incident ;
    - ✓

Standard 829-1998 IEEE – Rapport d'incident de test

- Aussi appelé rapport de problème :
  - Identificateur du rapport de test d'incident.
  - Sommaire :
    - ✓ Résumé de l'incident ;
    - ✓ Identifie les items de test impliqués en indiquant leur niveau de version/révision ;
    - ✓ Les références de la spécification de procédure du test approprié, la spécification du cas de test et le journal de test.
  - Description de l'incident :
    - ✓ Les entrées, les résultats attendus, les résultats réels, les anomalies, la date et l'heure, l'étape de la procédure, l'environnement, les tentatives de répétition, les testeurs, les observateurs ;
    - ✓ Toutes informations utiles pour reproduire et réparer.
  - Impact :
    - ✓ Si connu, indique quel impact cet incident aura sur les plans de test, les spécifications de la conception du test, les spécifications de procédure du test ou les spécifications du cas de test ;
    - ✓ Le ratio de sévérité (?)

Standard IEEE 829-1998 – Rapport sommaire du test

- Contient :
  - L'identificateur du rapport sommaire de test.
  - Le sommaire :
    - ✓ résumer l'évaluation des items du test;
    - ✓ identifier les items testés en indiquant l'environnement dans lequel les activités du test ont eu lieu.
  - Les variances :
    - ✓ Des items du test par rapport des spécifications de conception originales.
  - L'évaluation de la généralité :
    - ✓ évaluer la généralité du processus de test contre la généralité des critères spécifiés dans le plan de test si le plan de test existe ;
    - ✓ identifier les fonctionnalités ou les combinaisons de fonctionnalités qui n'étaient pas suffisamment testées et expliquer les raisons.

Standard IEEE 829-1998 –Rapport sommaire du test – cont.

- Sommaire des résultats :
  - Résumer les résultats de test.
  - Identifier tous les incidents résolus et résumer leurs résolutions
  - Identifier tous les incidents non résolus.
- Évaluation :
  - Fournir une évaluation en général de chaque item de test ainsi que ses limitations ;
  - Cette évaluation devra être basée sur les résultats du test et sur le niveau des critères passe/échec de l'item ;
  - Une estimation de risque d'échec peut être inclus.
- Sommaire des activités :
  - Résumer les activités du test majeur et des événements ;
  - Résumer les données de consommation des ressources, e.g., niveau total du personnel, temps total machine et temps total écoulé pour les activités majeures de test.
- Approbations.

Leçons par Cem Kaner et al Test de documentation

- Leçon 142 Pour appliquer une solution efficacement, vous avez besoin de comprendre clairement le problème.
- Leçon 143 N'utilisez pas les modèles de documentation de test : Un modèle n'aidera pas sauf si vous n'en avez pas besoin.
- Leçon 144 Utilisez les modèles de documentation de test : ils favorisent une communication cohérente.
- **Leçon 145 Utilisez le standard IEEE 829 pour la documentation de test.**
- **Leçon 146 N'utilisez par le standard IEEE 829 :**
  - le Std IEEE fournit des modèles, voir leçon 143 ;
  - assumez une approche en cascade ;
  - pas de directives pour ajuster selon les besoins du projet ;
  - trop de documentation ;
  - absence de conscience concernant le coût
  - difficile de concilier avec le test automatisé
  - ...
- Leçon 147 Analysez vos besoins avant de décider quels produits construire ; ceci s'applique autant pour votre documentation que pour votre logiciel.

Leçons par Cem Kaner et al documentation de Test

- Leçon 148 Pour analyser les requis de la documentation de test, posez des questions comme celles dans cette liste :
  - Quelle est la mission de votre groupe et quels sont vos objectifs en testant votre produit ?
  - Est-ce que votre documentation de test est un *produit* (à donner à quelqu'un d'autre pour utiliser) ou un *outil* (pour usage interne) ?
  - Est-ce que la qualité de votre logiciel est contrôlée dans un but légal ou pour le marché ?
  - Quelle est la vitesse du changement de la conception ?
  - Quelle est la vitesse du changement de la spécification pour refléter le changement de conception ?
  - Quand vous testez, désirez-vous prouver la conformité aux spécifications ou la conformité avec les attentes du client ?
  - Est-ce que votre style de tests se base plus sur des tests déjà définis ou sur l'exploration ?
  - Est-ce que la documentation du test doit viser sur *quoi* tester (les objectifs) ou sur *comment* tester (procédures) ?
  - (...)

Leçons par Cem Kaner et al documentation de Test

- Leçon 149 Résumez les requis de base de votre documentation en une phrase avec pas plus de trois composants.
  - Exemple : « L'ensemble de documentation du test supportera principalement nos efforts de trouver les bogues dans cette version, de déléguer le travail et de tracer les états ».

Leçons par Cem Kaner et al planification de Test

- Leçon 274 Trois questions de base à se poser au sujet de la stratégie de test sont « p*ourquoi s'en faire* ? », « qui s'inquiète ? » et « *combien* ? ».
- Leçon 275 Il y a plusieurs stratégies possibles de test.
- Leçon 276 Le vrai plan de test est l'ensemble d'idées qui guide votre processus de test.
- Leçon 277 Créez votre plan de test en accord avec votre contexte.
- Leçon 278 Utilisez le plan de test pour exprimer les choix au sujet de la stratégie, des logistiques et des produits de travail.
- Leçon 279 Ne vous laissez pas aveugler par les logistiques et les produits de travail.
- Leçon 280 Comment se trouver avec les cas de test.
- Leçon 281 Votre stratégie est plus que vos tests.
- Leçon 282 Votre stratégie de test explique votre test.
- Leçon 283 Appliquez des demi-mesures diverses**.**

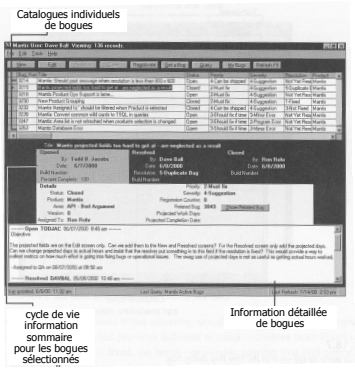
Leçons par Cem Kaner et al planification de *Test*

- Leçon 284 Cultivez la matière brute des stratégies de test puissantes.
- Leçon 285 Votre première stratégie sur un projet est toujours mauvaise.
- Leçon 286 À toutes les phases du projet, demandez-vous « qu'est-ce que je peux tester et comment puis-je le tester ? »
- Leçon 287 **Tester jusqu'à la maturité du produit.**
- Leçon 288 Utilisez les niveaux de test pour simplifier les discussions sur la complexité de test.
- Leçon 289 **Tester la boîte grise.**
- Leçon 290 Attention aux anciens cultes quand on réutilise des matériaux de test.
- Leçon 291 Deux testeurs testant la même chose ne sont probablement pas en train de dupliquer les efforts.
- Leçon 292 Créez votre stratégie de test en réponse aux facteurs de projet autant qu'aux risques de produit.
- Leçon 293 Traitez les **cycles de test** comme le battement de cœur du processus de test.

Rapport et suivi des bogues

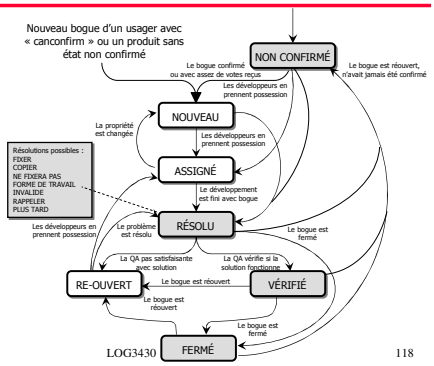
Compte-rendu et détection de bogue automatisés

(Mantis)



(source: Ron Paton)

Compte-rendu et détection de bogue automatisés



Copie de la figure 6-1 - Cycle de vie de Bugzilla Bug

(Bugzilla – open source)

(source: http://www.bugzilla.org/docs/*/pdf/Bugzilla-Guide.pdf)

Rapport écrit d'un bogue

Nom de votre compagnie Confidentiel Rapport de problème # _____

Programme _____ Distribution _____ Version _____

Type de rapport (*1-6*) ___ Sévérité (*1-3*) ___ Pièce jointe (*O/N*) ___
1 – erreur de codage 4 – documentation
2 – problème de conception 5 – matériel
3 – suggestion 6 - question
1 – fatal
2 – sérieux
3 – mineur
Si oui, décrire : _____

Résumé du problème : _____

Pouvez-vous reproduire le problème ? (*O/N*) ____

Problème et comment le reproduire _____

Solution suggérée (optionnel) _____

Rapporté par _____ Date ___ / ___ / ___

Les items suivants sont seulement pour l'usage de l'équipe de développement

Zone fonctionnelle _____ Assigné à _____

Commentaires _____

Statut (*1-2*) ____ Priorité (*1-5*) ____
1 – ouvert 2 - fermé

Résolu par _____ Date ___ / ___ /__

Solution testée par _____ Date ___ / ___ /__

Traitement reporté (*O/N*) ___

Copie de la figure 1.3 – formulaire du rapport d'un problème

(source: Cem Kaner, Testing Computer Software)

Suivi des jeux detest

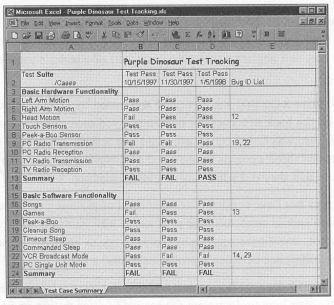


Figure 17.4 – Une feuille de travail peut être utilisée pour traquer et gérer efficacement les suites de tests et les cas de tests.

(source: Ron Paton)

Leçons par Cem Kaner et al
Plaidoyer du bogue

- Leçon 55 Vous êtes ce que vous écrivez
- **Leçon 56 Votre plaidoyer conduit à la réparation des bogues que vous avez rapportés**
- Leçon 57 Faites de votre rapport de bogue un outil de ventes efficace
- Leçon 58 Votre rapport de bogues est votre représentant
- Leçon 59 Prenez le temps de rendre précieux votre rapport de bogues
- **Leçon 60 Chaque intervenant devrait être capable de rapporter un bogue**
- Leçon 61 Soyez prudent avant de gratifier les rapports de bogues d'autres personnes
- Leçon 62 Le rapport perçoit des lacunes de qualité comme des bogues
- Leçon 63 Quelques intervenants ne peuvent pas rapporter les bogues – vous êtes leur proxy
- Leçon 64 Attirez l'attention affectée d'intervenants sur les bogues controversés
- Leçon 65 N'utilisez jamais le système de bug-tracking pour contrôler la performance de programmes
- Leçon 66 N'utilisez jamais le système de de bug-tracking pour contrôler la performance des testeurs
- **Leçon 67 Rapportez promptement les anomalies**
- Leçon 68 N'assumez jamais qu'un bogue évident a déjà été classé

Leçons par Cem Kaner et al
Plaidoyer du bogue

- Leçon 69 Rapportez les erreurs de conception
- Leçon 70 Les bogues extrêmes sont des imperfections potentielles de sécurité
- Leçon 71 Décoincez les cas critiques
- Leçon 72 Les bogues mineurs sont rapportés et réparés
- Leçon 73 Gardez claire la différence entre la sévérité et la priorité
- Leçon 74 Une défaillance est le symptôme d'une erreur, pas l'erreur elle-même
- Leçon 75 Faites le suivi des tests sur les erreurs de codage apparemment mineures
- **Leçon 76 Rapportez toujours les erreurs non reproductibles ; elles peuvent être des bombes à retardement**
- **Leçon 77 Les bogues non reproductibles sont reproductibles**
- Leçon 78 Soyez conscient du coût du processus de vos rapports de bogues
- Leçon 79 Donnez une manipulation spéciale aux bogues reliés aux outils ou à l'environnement
- **Leçon 80 Demandez avant de rapporter des bogues s'il s'agit des prototypes ou des versions privées anticipées**
- Leçon 81 Dupliquer les rapports de bogues est un problème d'auto correction
- Leçon 82 Chaque bogue mérite son propre rapport

Leçons par Cem Kaner et al
Plaidoyer du bogue

- Leçon 83 La ligne de synthèse est la ligne la plus importante dans le rapport de bogue
- Leçon 84 N'exagérez jamais vos bogues
- **Leçon 85 Rapportez clairement le problème, mais n'essayez pas de le régler**
- Leçon 86 Soyez attentifs dans la formulation de votre rapport. Toutes les personnes que vous critiquerez verront ce rapport
- Leçon 87 Rendez vos rapports lisibles, même par des personnes qui sont épuisées et grognons.
- Leçon 88 Améliorez votre habilité de rapporter (de reportage)
- Leçon 89 Utilisez les données de support ou du marché quand c'est approprié
- Leçon 90 Révisez chacun des rapports de bogues
- **Leçon 91 Rencontrez les programmeurs qui liront vos rapports**
- **Leçon 92 La meilleure approche peut être de démontrer les bogues aux programmeurs**
- **Leçon 93 Quand le programmeur dit que c'est réparé, soyez certain que ce n'est pas encore brisé**
- **Leçon 94 Vérifiez rapidement si le bogue est réparé**
- **Leçon 95 Quand les réparations échouent, parlez avec le programmeur**

Leçons par Cem Kaner et al
Plaidoyer du bogue

- Leçon 96 Les rapports de bogues devraient être fermés par les testeurs
- Leçon 97 N'insistez pas pour que tous les bogues soient réparés. Choisissez votre bataille
- Leçon 98 Ne laissez pas reporter les bogues disparus
- Leçon 99 L'impossibilité de tester ne devrait pas être la cause du report d'un bogue
- Leçon 100 Faites appel immédiatement aux bogues reportés
- Leçon 101 Quand vous décidez de vous battre, décidez de gagner !

Mesurages du test et gestion du test

Mesures du test

- Mesures pour contrôler les coûts de test :
  - Le nombre de cas de test développés
  - La taille de l'infrastructure de test (dans KLOC)
  - durée de test
  - Le coût de test
- Mesures pour contrôler l'adéquation du test :
  - Le pourcentage de la couverture du code source
  - Le pourcentage de la couverture des requis
- Mesures pour contrôler la qualité du produit :
  - Le nombre d'anomalies trouvées dans le produit avant la distribution (par le test)
  - Le nombre d'anomalies trouvées dans le produit après la distribution (par les usagers)
  - Le nombre total d'anomalies trouvées
- Mesures pour contrôler les test d'efficacité :
  - Le nombre d'anomalies trouvées par le test / KLOC (du logiciel sous test)
  - Le nombre d'anomalies trouvées par le test / le nombre de cas de tests
  - Le nombre d'anomalies trouvées par le test / Le nombre total d'anomalies trouvées (en testant et par les usagers)
- Évolution des mesures précédentes en fonction du temps

Critères d'arrêt de test (1/2)

- Le temps et les ressources du projet expirent :
  - Pas un bon critère.
- La détection d'un nombre spécifique d'anomalies a été accomplie.
  - Pas un bon critère (pourquoi ?).
- Tous les tests planifiés qui ont été développés ont été exécutés avec succès
- Tous les buts de couverture spécifiés ont été atteints :
  - Exemple : une couverture de branche à 100% dans un test unitaire et tous les requis ont été couverts par les tests de système.
- Les taux de détection d'anomalies pour une certaine période de temps ont chuté sous un niveau spécifié.
  - Exemple : "nous avons cessé de tester quand nous avons trouvé 5 anomalies ou moins, avec un impact inférieur ou égal au niveau de sévérité 3, par semaine".

Critères d'arrêt de test (2)

- Les ratios d'injection d'anomalies sont favorables :
  - injection d'anomalies basée sur les données historiques d'anomalies.
  - Exemple : "au moins 90% d'anomalies injectées ont été détectées par le jeu de test".
- Un niveau de fiabilité spécifié a été atteint :
  - Exemple : l'intervalle moyen avant la défaillance (*mean time to failure* (MTTF)) est plus long que 24 heures.
  - Estimé avec un test statistique basé sur un profil usage/opérationnel.
  - A besoin d'une organisation de test de haute maturité.
- Un niveau de confiance d'absence d'anomalies a été atteint.
  - Le logiciel ne contient pas d'anomalies, ou en a moins qu'un certain nombre d'anomalies par KLOC, avec un certain niveau de confiance (exemple : 95%)
  - Estimé avec des techniques d'injection d'anomalies
  - A besoin d'une organisation de test de haute maturité.
- Utilisation d'une combinaison de critères, dépendant d'un projet à un autre!

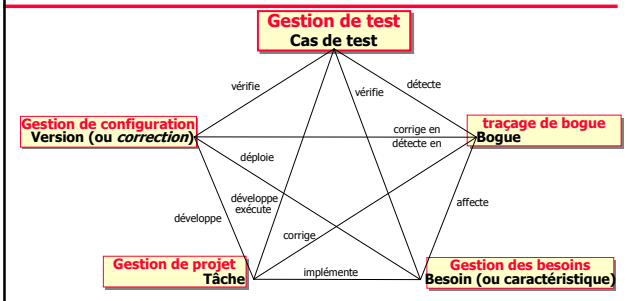
Balance pour la décision optimale d'arrêt de test

- Arrêter de tester trop tard :
  - Un gaspillage des ressources
  - Un temps de retard pour la mise en marché
  - Une augmentation des coûts
  - Les échéanciers retardés
- Arrêter de tester trop tôt :
  - Les défauts restent et causent des pertes ou dommages à la vie ou à la propriété
  - Les clients sont insatisfaits
  - Les coûts élevés pour réparer
  - Les coûts d'appels d'assistance téléphonique

Décision relâchée

- Normalement, ce n'est pas bénéfique (cost-effective) de réparer tous les bogues trouvés avant de distribuer le produit.
- Quelques bogues sont très difficiles à réparer :
  - Difficile à trouver (exemple : bogues aléatoires)
  - Difficile à corriger (exemple : composant tierce partie).
- Réduit le risque dans l'ensemble :
  - Réparer tous les bogues de haute sévérité
  - Réparer (les plus faciles) 95% de bogues de sévérité médium
  - Réparer (les plus faciles) 70% de bogues de basse sévérité.

Outil de support



Ce qui est nécessaire est un environnement de génie logiciel intégré (*Integrated Software Engineering Environment* (ISEE)

Plus d'information

- Ian Sommerville, "Software Engineering", 7th edition, Addison-Wesley
- ISO 9000
  http://www.iso.ch/iso/en/iso9000-14000/iso9000/iso9000index.html
- SEI Capability Maturity Model / ICMM
  http://www.sei.cmu.edu/cmm/cmm.html
- Certified Software Quality Engineer (CSQE) Body of Knowledge
  http://www.asq.org/cert/types/csqe/bok.html

Plus d'information

- Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE Computer Society
- IEEE Standard for User Documentation (IEEE Std 1063-2001)
- IEEE Recommended Practices for Software Requirements Specification (IEEE Std 830-1993)
- IEEE Recommended Practices for Software Design Descriptions (ANSI/IEEE Std 1016-1987)
- IEEE Standard for Software Reviews and Audits (IEEE Std 1028-1988)
  - Available from ieeeexplore from FEUP
- IEEE Standard for Software Quality Assurance Plans (IEEE Std 730-1998)
  - Available on the web
- Producing Quality Technical Information (PQTI), IBM Corporation, 1983
  - considered by many to contain one of the earliest comprehensive discussions about the multidimensional nature of quality documentation
- Developing Quality Technical Information (DQTI), G. Hargis, Prentice-Hall, 1997 (first edition), 2004 (second edition)
  - a revised edition of PQTI

Plus d'information

- Practical Software Testing, Ilene Burnstein, Springer-Verlag, 2003
  - From academics
  - Chapter 7 (Test Goals, Policies, Plans, and Documentation) describes the IEEE standard 829-1983 for Software Test Documentation
  - Appendix II (Sample Test Plan) provides a complete example of a test plan documented according to the standard
- IEEE Standard For Software Test Documentation - IEEE Std 829-1998
  - Includes examples
- Lessons Learned in Software Testing: a Context-driven Approach, Cem Kaner, James Bach, John Wiley & Sons Inc, 2002
  - Practical advice
  - Chapters 4 (Bug Advocacy), 6 (Documenting Testing) and 11 (Planning the Testing Strategy)

Plus d'information

- Software Testing, Ron Patton, SAMS, 2001
  - From practitioners, with lots of practical information, ready to use
  - Part V (Working with Test Documentation): Chapters 16 (Planning Your Test Effort), 17 (Writing and Tracking Test Cases), 18 (Reporting What You Find) and 19 (Measuring Your Success)
- Testing Computer Software, 2nd Edition, Cem Kaner, Jack Falk, Hung Nguyen, John Wiley & Sons, 1999
  - Chapters 5 (Reporting and analyzing bugs) and 12 (Test planning and test documentation)