420-KBE-LG, Projet dirigé

Saliha Yacoub

Les tests

Plan de la séance

- Les tests unitaires
- Les tests de composants
- Les tests d’intégration
- Les tests d’acceptation
- Les test de non-régression
- Scrum et les tests
- Conclusion

Définition

- **Les Bogues :** Viennent du mot anglais Bug, qui signifie insecte parasite. L'utilisation de ce mot pour désigner une anomalie vient de l'époque des premiers ordinateurs (à tubes à vides). Des cafards attirés par la chaleur dégagée par ces tubes venaient se réfugier dans ces machines et y mourraient. Leurs cadavres provoquaient des courts-circuits d'où des pannes qui ont été appelaient des Bug.
- **Une erreur :** peut être commise par un programmeur.
- **Un défaut ou bogue :** est le résultat de l'erreur
- **Une panne :** est la conséquence du défaut
- **Une panne,** une **défaillance** ou une **anomalie** est un comportement anormal d'un programme.

Un logiciel est dit **correct** s'il ne contient aucun défaut

Un logiciel est dit **fiable** s'il s'exécute sans défaillance

Définition

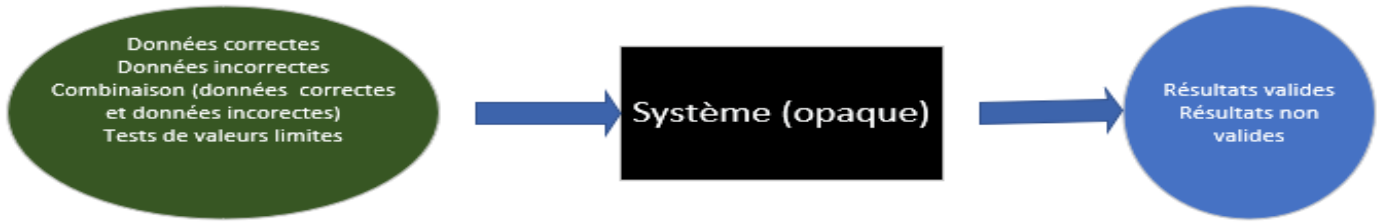
- **Définition**: Tester, c'est exécuter le programme dans l'intention d'y trouver des anomalies ou des défauts .G. Myers(the Art of Software Testing).
- Le but: le but d'un test est de trouver des erreurs. Si celui qui teste n'a pas comme but de trouver des erreurs, mais plutôt de démontrer que son programme fonctionne, il ne trouvera pas ses erreurs.

- Hiérarchie des tests:
  - Tests unitaires ou programmes: (morceaux de codes, procédures) tests effectués par ceux qui développent. (Exemple : cas de valeurs limites, cas d'invalidité, cas de nullité)
  - Test des composants (modules), ici rentrent en jeu les testeurs
  - Tests d'intégration, visent à vérifier l'intégrité de données sur une chaîne complète de programmes. Tests réalisés par le chef de projet ou l'analyste (celui qui a plus connaissance du système)
  - Tests d'acceptation ou de validation : tests réalisés chez le client.

Les tests unitaires

Stratégie de test:

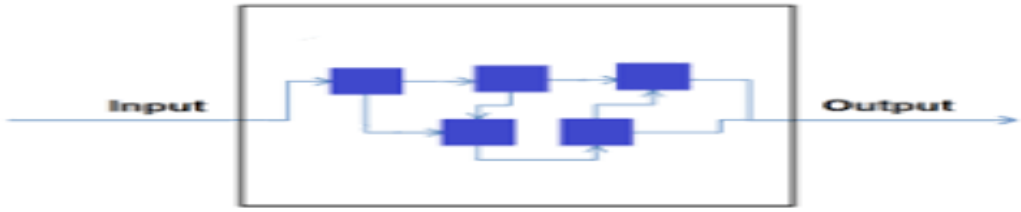
- **Les tests boite noire**: (tests fonctionnels): Le testeur ne s'intéresse pas aux mécanismes internes du logiciel ou de la fonction à tester, mais uniquement à ses entrées/sorties. Il peut être appliqué à tous les niveaux de test logiciel: test unitaire, test d'intégration, etc. Ici on teste que le programme fonctionne.
  - test choisi à partir des spécifications (use-case, user story, etc.)
  - évaluation de l'extérieur (sans regarder le code), uniquement en fonction des entrées et des sorties



Les tests unitaires

Stratégie de test:

- **Les tests boite blanche** (tests structuraux) :ont pour but de vérifier le fonctionnement interne du système ou d'une méthode particulière. Dans cette stratégie, les tests sont basés sur la couverture d'instructions de code, de branches, de chemins ou de conditions. Ici, c'est la qualité du code qui est examinée. Ce n'est pas parce que ça marche que votre code est bon
  - Y a-t-il un if dans lequel le programme ne rentre jamais ?
  - Y' a-t-il une itération de trop ?
  - Les variables sont-elles initialisées correctement ?



Les tests unitaires

- Effectuer les tests unitaires, revient à effectuer les tests boite noire et les tests boites blanche
- Exemple, pour tester le retrait :Test boite noire
  - l'état d'un compte bancaire peut être positif, négatif ou zéro.
  - Le retrait peut être situé à l'intérieur de la fourche autorisée
  - Le retrait va être à l'extérieur de la fourche autorisée
  - Le retrait va être au frontière de la fourche autorisé
  - Le retrait va être une valeur interdite.
  - D'un compte ayant un solde négatif
  - D'un compte ayant un solde nul
  - D'un compte ayant un montant inférieur à la valeur du retrait.

Stratégie de test

Données du test

On test quoi ?	État du compte	Valeur du retrait	Résultat attendu
A l'intérieur de la fourche autorisée	500	400	Retrait effectué
Valeur limite	500	500	Retrait effectué
Valeur limite	500	20	Retrait effectué
Valeur limite	500	0	Pas de retrait
Valeur incorrecte	500	-20	Pas de retrait
Valeur incorrecte	500	15	Refusé (pas un multiple de 20)
Valeur à l'extérieur de la fourche autorisé	40	100	Retrait refusé
Retrait d'un compte négatif	-100	20	Retrait refusé
Retrait d'un compte vide	0	20	Retrait refusé
Retrait avec solde insuffisant	100	400	Retrait refusé

Les tests unitaires

Exemple, pour tester le retrait :Test boite blanche

- Pour l'exemple précédent il faudra s'assurer que votre code exécute toutes les instructions IF.
- On vérifie en premier que le solde est plus élevé ou égal au montant du retrait, (le 1er IF)
- Ensuite on vérifie que le montant du retrait est positif ou nul (2em IF)
- Etc…

Les tests de composants

- Les tests de composants: sont les tests effectués à l'intérieur d'un module (composant) considéré comme un tout.

Exemple : Le module : Évaluer les restaurants

- Évaluer des restaurants sur son application mobile (sur son cellulaire)
- Envoyer l'évaluation sur un serveur MySQL
- Vider la base de données de son cellulaire .

Les tests d’intégration

- Les tests d’intégration: Sont les tests qui consistent à mettre deux (ou plus) composants logiciels ensemble

Exemple:

  - Module 1: Le module évaluer les restaurants par le biais d’une application mobile.
  - Module 2: Le module qui compile les données provenant du serveur MySQL pour connaître la moyenne des évaluations, le nombre de personnes ayant données 5 étoiles pour un restaurant …etc. et affiche le résultat sur un site Web

La question est: le module 1 et le module 2 vont-ils fonctionner ensemble (est-ce que les données compilées et affichées sur le site web sont celles envoyées par l’application mobile ?)

Ici on aurait pu ajouter un module 3: Envoyer la compilation des résultats sur l’application mobile

- Les tests d’acceptation sont les tests qui garantissent que le produit final est conforme aux exigences du client.

Les tests d’acceptation

- Les tests d’acceptation sont les tests qui garantissent que le produit final est conforme aux exigences du client. (voir le cours sur les user-stories)
- En général, ils sont rédigés sous forme de scénario ou sous forme de liste.

Exemple 1: (sous forme de scénario)

US: En tant qu’intéressé, je veux m’inscrire à la conférence sur le climat «SOS Terre» qui déroule à la salle «Talbot»

Critères d’acceptation :

Inscription acceptée:

Étant donné l’utilisateur Primogene connecté à la conférence «SOS Terre» qui se déroule à la salle Talbot d’une capacité de 200 places et dont le nombre d’inscrit actuels est 152. Quand Primogene s’inscrit à «SOS Terre» alors, l’inscription est acceptée, un message de confirmation est envoyé à Primogene et le nombre d’inscrits sera augmenté de 1.

Inscription refusée:

Étant donné l’utilisateur Patoche connecté à la conférence «SOS Terre» qui se déroule à la salle Talbot d’une capacité de 200 places et dont le nombre d’inscrit actuels est 200. Quand Patoche s’inscrit à «SOS Terre» alors, l’inscription est refusée, un message de refus est envoyé à Patoche

Les tests d’acceptation

Exemple 2(sous forme de scénario)

US: En tant que joueur connecté, je souhaite ajouter un items à mon panier

Critères d’acceptation:

Étant donné le joueur Barackuda connecté à Chevalersk , lorsqu’il choisit d’ajouter l’item « Hache de fer » à son panier avec une quantité x , le montant du panier est mis à jour.

Exemple 3 (sous forme de liste)

US: En tant que joueur connecté, je souhaite effectuer une recherche d’items afin de pouvoir l’acheter ou non.

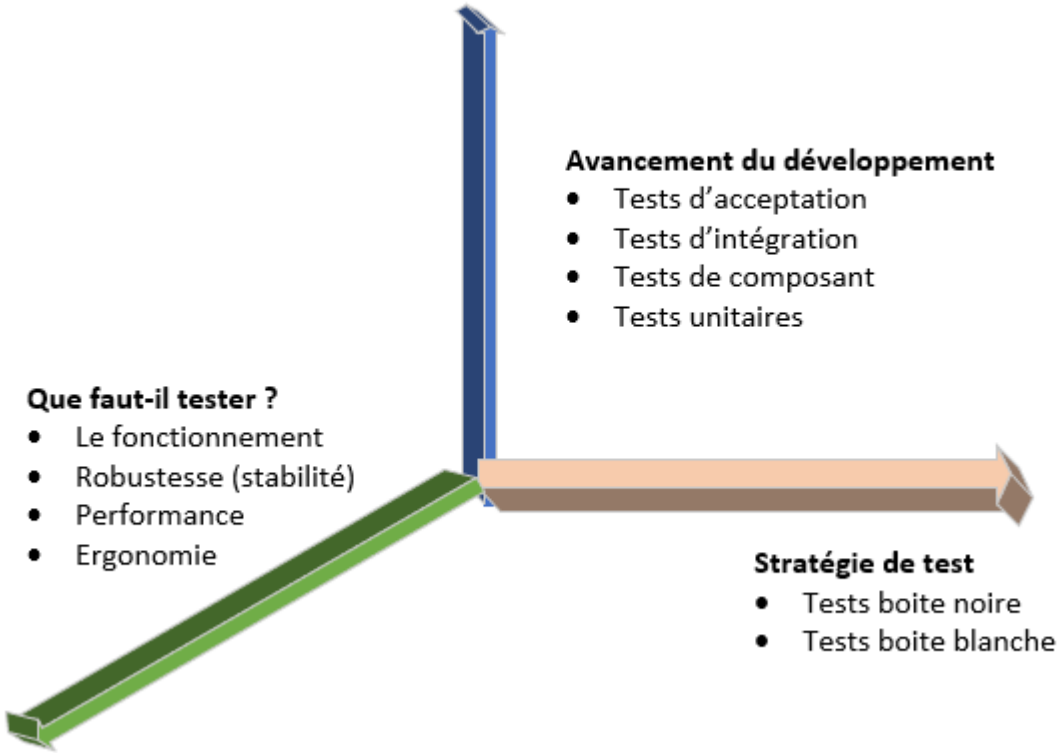
Critères d’acceptation:

- Résultat unique: afficher l’item avec ses détails: nom, prix, efficacité …
- Pas de résultat: inviter le joueur à faire une autre recherche
- Plusieurs résultats : afficher la liste d’items ordonnée par prix

Que faut-il tester ?

- Que faut-il tester
  - Le bon fonctionnement
    - L'exactitude des outputs;
    - La sécurité des données;
  - La robustesse:
    - La capacité de traiter le volume requis de données;
    - La capacité de recouvrement après un arrêt de fonctionnement;
  - La performance
    - Le temps de réponse;
  - Ergonomie :
    - Facile à utiliser
    - Qualité de la documentation (Guides utilisateurs, guide d'installation , etc …)

Que faut-il tester ?



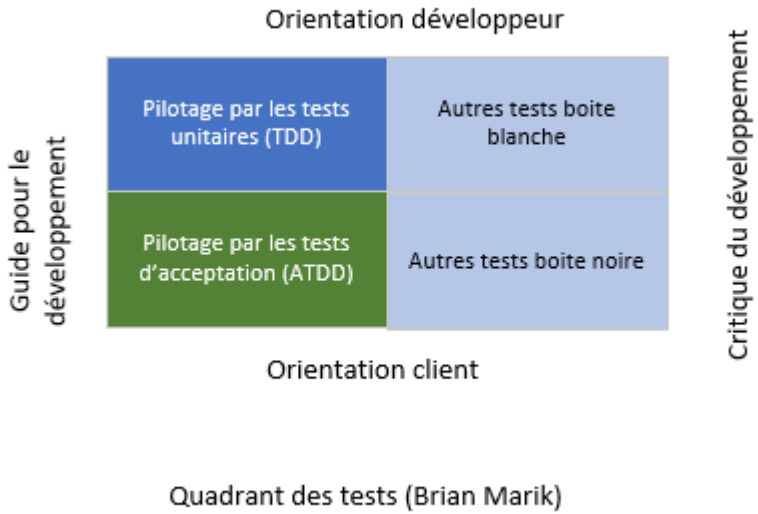
Les tests et SCRUM

SCRUM et les tests

- N’oublions pas que SCRUM n’est pas une méthode de développement. C’est une approche pour la gestion de projet.
- Contrairement au modèle en cascade, pour lequel « effectuer les tests » est une phase de développement, pour SCRUM le test n’est pas une phase.
- Avec l’approche itérative, les tests sont intégrés dans chaque sprint. Dès le premier jour du premier sprint, les testeurs commencent à tester les stories.
- Pour les méthodes Agiles, les tests ont un objectif qui est de guider dans le développement et non nécessairement la détection d’érreurs (qui vise à critiquer)

Les tests et SCRUM

Dans le TDD(Driven Test Developement) le développement est centré sur les tests unitaires. C’est aussi le but pour les tests d’acceptation qui sont des tests orientés clients (Acceptance Test Driven Developement



Quadrant des tests (Brian Marik)

Les tests et SCRUM

Pratiques autour du code:

- **L'intégration continue**

L'intégration continue est une pratique logiciel qui conduit une équipe à intégrer leur travail fréquemment. Habituellement au moins un fois par jour. À chaque COMMIT (Mettre le travail dans un serveur commun pour l'équipe)

- Compiler et vérifier le build
- Lancer les tests unitaires
- Lancer les tests d'intégration et les tests d'acceptation
- Produire un rapport d'erreurs.

Quand ?

L'intégration continue doit-être mise en place avant le premier sprint. Si l'équipe ne dispose pas de serveur et du logiciel requis, alors l'équipe doit créer une « story technique » qui va dans le backlog. L'équipe doit alors déterminer la priorité de la story en question.

Avantage:

L'intégration continue permet d'avoir à tout moment un logiciel qui marche. Ce qui motive l'équipe et les utilisateurs.

Les tests et SCRUM

Pratiques autour du code:

- **Pilotage par les test:**

Un développeur qui écrit du code a la responsabilité de le tester morceau par morceau: tests unitaires.

Le développeur écrit les tests en premier, ce qui lui permet de prévoir le comportement du composant, ensuite il écrit le code pour que les tests passent avec succès.(TBN, tests boite noir)

Le remaniement du code consiste à améliorer le code sans changer son comportement (Tests boite blanche). Le but est d'améliorer la qualité du code et optimiser la solution actuelle.

À chaque remaniement, il convient de lancer les tests à nouveau.

Quand ?

Avant le premier sprint si possible pour déterminer les TBB. Pour le remaniement du code (TBN), il pourrait être au sprint suivant ou plus loin selon la priorité attribuée.

Avantage:

Code testé et optimisé.

Les tests et SCRUM

Pratiques autour du code:

- **Programmation en binôme:**

Pratique qui consiste à mettre deux développeurs D1 et D2 sur un même poste de travail, de façon à ce que la collaboration améliore la qualité du code.

  - D1 qui code, D2 qui regarde.
  - D1 orienté vers le détails du code, D2 orienté sur la structure de l'ensemble du programme.

Quand ?

C'est à l'équipe de décider

Avantages:

  - Améliore la qualité du produit
  - La dette technique sera moindre.

Les tests et SCRUM

Gestion des BUGS

- Défaut dans une story: un défaut trouvé dans une story d'un sprint en cours est une condition de satisfaction en « échec »: La story n'est pas finie. L'équipe doit ajouter les tâches qu'il faut pour corriger le défaut.
- Défaut sur une story considérée comme finie:
  - Défaut critique: ce qui empêche le fonctionnement d'une ou plusieurs stories. Ce qui rend l'utilité de la story nulle
  - Défaut majeur: ne permet pas un fonctionnement normal et fait perdre une grande utilité à la story
  - Défaut mineur: fait perdre un peu de valeur à la story et rendra son utilisation difficile.

Les tests et SCRUM

Mettre les BUGS dans le backlog

- Traitement des défauts critiques: traité d'une façon prioritaire. Tout de suite.
  - On crée une tâche à l'intérieur du sprint.
  - Pour la tâche on met 1 heure de travail
  - Dès qu'une personne se libère, elle traite le défaut.
  - Si plus qu'une heure est nécessaire, alors on identifie toutes les tâches à faire et on crée une entrée dans le **backlog de sprint**.
  - Il est possible que la « vélocité du sprint » soit augmentée.

Les tests et SCRUM

Mettre les BUGS dans le backlog

- Traitement des défauts majeurs
  - On suit les même étapes que pour un défaut critique sauf que l'entrée est créée dans le **backlog du produit**
- Traitement des défauts mineurs
  - Un défaut mineur va directement dans le backlog du produit.
  - Les tests de non-regression ont pour but de s'assurer que les modifications et évolutions effectuées par les développeurs lors du dernier sprint n'ont pas entrainé d'effet de bord, en altérant les parties du code non modifiées. Ils doivent être lancés à chaque livraison

Les tests et SCRUM

- Sources:
  - SCRUM: le guide pratique de la méthode agile la plus populaire., Claude Aubrey, DUNOD 2011
  - Le processus unifié (Ivar Jacobson, Grady Booch, James Rumbaugt)

  - https://inf1410.teluq.ca/teluqDownload.php?file=2014/01/INF1410-Module4-Texte4_1.pdf
  - https://www.ibisc.univ-evry.fr/~belardinelli/Documents/testlogiciel.pdf

CONCLUSION

QUESTIONS ??