Test de logiciel dans les méthodes agiles

Roozbeh SOLTANI

Roozbehsltn@gmail.com

On parle de ….

- Introduction
- Qu'est-ce que tester ?
- Que recherche-t-on avec le test ?
- Traits caractéristiques d'un test
- Le test dans le cycle de vie du logiciel
- Test Driven Development
- Comment écrire des tests automatiques
- Quels types de tests
- Le test dans le cycle de vie du logiciel
- Types de test – ce que l'on veut tester!
- Difficulté du test
- Des règles pratiques

/www.youtube.com/watch?v=59huPOAQEAk

Introduction

- Il est nécessaire d'assurer la fiabilité des logiciels
  - Domaines critiques : atteindre une très haute qualité imposée par les lois/normes/assurances/
  - Autres domaines : atteindre le rapport qualité/prix jugé optimal (attentes du client)

- Assurer la fiabilité du logiciel est une part cruciale du développement
  - Plus de 50% du développement d'un logiciel critique
  - Plus de 30% du développement d'un logiciel standard

- Plusieurs études ont été conduites et indiquent que
  - Entre 30 et 85 erreurs sont introduites par portion de 1000 lignes de code produites (logiciel standard).

- Ces chiffres peuvent fortement augmenter si les méthodes permettant d'assurer la fiabilité sont mal gérées au sein du projet
  - Voir la part importante de l'activité de tests dans les différentes méthodes de conception de logiciels

Introduction

- Dans le cycle de vie du logiciel, on assure la fiabilité / qualité du logiciel par l'activité de Vérification et de Validation

Vérification

- le développement est-il correct par rapport à la spécification initiale ?
- Est-ce que le logiciel fonctionne correctement?

Validation

- a-t-on décrit le « bon » système ?
- est-ce que le logiciel fait ce que le client veut ?

Règle agile : écrire les tests avant de coder

Introduction

- Les principales méthodes de validation et vérification
  - Test statique
    - examen ou analyse du texte
    - Revue de code, de spécifications, de documents de design
  - Test dynamique
    - Exécuter le code pour s'assurer d'un fonctionnement correct
    - Exécution sur un sous-ensemble fini de données possibles
  - Vérification symbolique
    - Run-time checking
    - Exécution symbolique, …
  - Vérification formelle :
    - Preuve ou model-checking d'un modèle formel
    - Raffinement et génération de code

Actuellement, le test dynamique est la méthode la plus répandue et utilisée.

dans ce cours, **test = test dynamique**

Qu’est-ce que tester ?

- Tester c’est réaliser l’exécution du programme pour y trouver des défauts et non pas pour démontrer que le programme ne contient plus d’ erreur
- Le but d’un testeur est de trouver des défauts à un logiciel
- Si ce n’est pas son but, il ne trouve pas

→→ aspect psychologique du test

- C’est pourquoi il est bon que

- Les tests soient écrits avant le développement (règle usuelle des méthodes agiles)
    - cf. le développement dirigé par les tests (test-driven development) proné par l’ eXtrem Programming

Qu'est-ce que tester ?

- En pratique : ces idées font leur chemin
  - Cycle en V : même entreprise, mais équipes différentes
  - XP : tests unitaires par le développeur, mais tests de recettes par le client
  - logiciels critiques : tests de recettes validés par une agence externe
  - Microsoft / Google / etc. : beta-testing par les futur usagers

Si vous devez ne retenir qu'une chose de ce cours, ce doit être que ***vous devez croire que votre code contient des erreurs*** avant de le tester

Que recherche-t-on avec le test ?

Défaut : mauvais point dans le logiciel (conception, codage, etc.)

Faute (= bug) : défaut amenant un comportement fautif *observable*

- La différence est qu'il existe toujours des défauts mais qu'ils ne sont pas forcément observables
  - ex : fonction boguée mais jamais appelée

- Veut-on éviter les fautes ou les défauts ?
  - on veut éviter les fautes, mais souvent on cherche les défauts

https://www.youtube.com/watch?v=29S9vyfgDEk&list=PLmhQGrnqvRQxHbtA_AG0S_VQWJZS1zblN
https://www.youtube.com/watch?v=CCXUxiZPh80&list=PLmhQGrnqvRQxHbtA_AG0S_VQWJZS1zblN&index=3

Traits caractéristiques d'un test

- Définition IEEE

Le test est l'exécution ou l'évaluation d'un système ou d'un composant par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus

(Standard Glossary of Software Engineering Terminology)

- Le test est une méthode **dynamique** visant à trouver des bugs

Tester, c'est exécuter le programme dans l'intention d'y trouver des anomalies ou des défauts

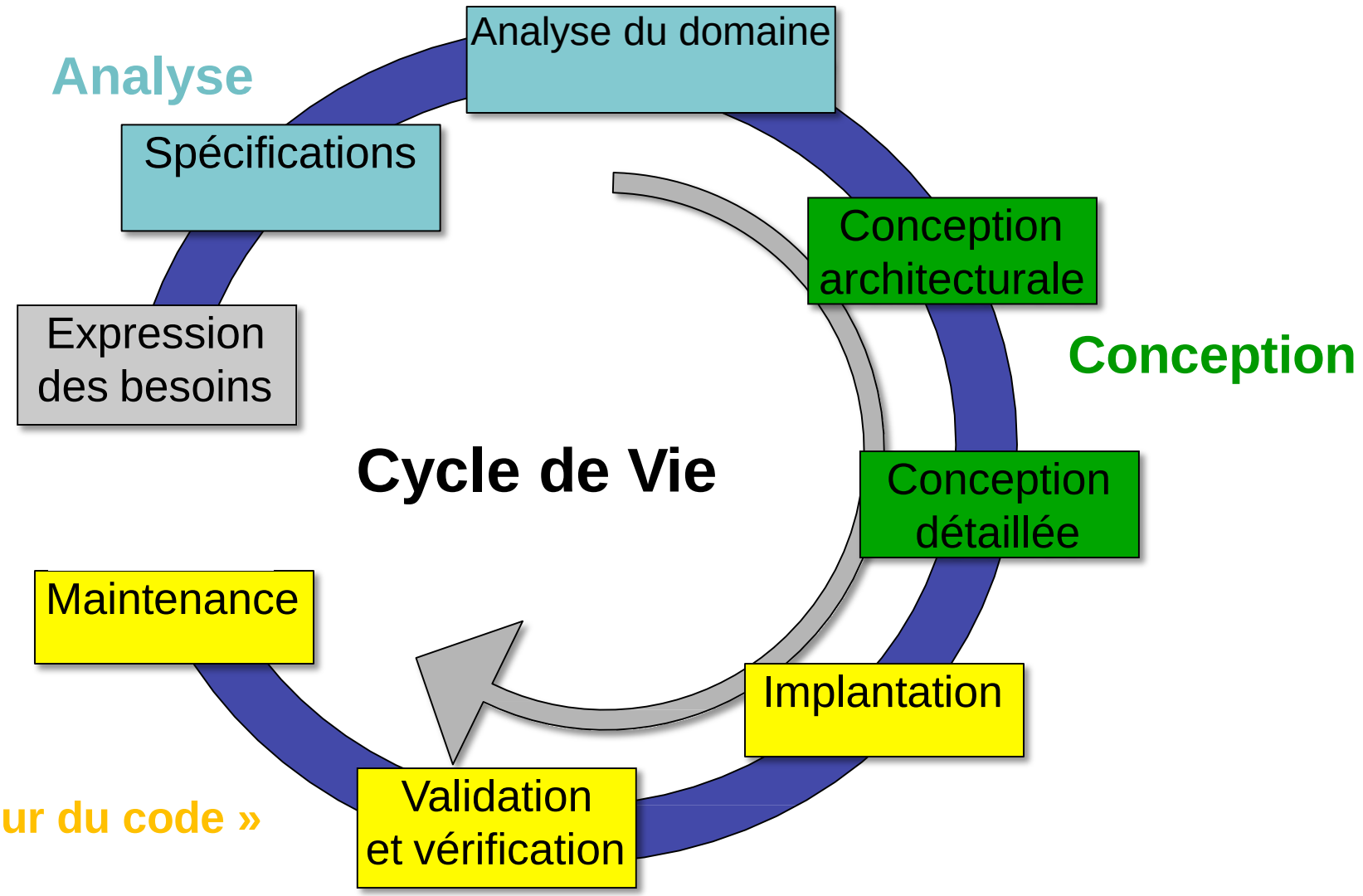
G. J. Myers (The Art of Software Testing, 1979)

- Le test est une méthode de validation **partielle** des logiciels

Tester permet seulement de révéler la présence d'erreurs mais jamais leur absence.

E. W. Dijkstra (Notes on Structured Programming, 1972)

Le test dans le cycle de vie du logiciel



Test Driven Development

- Le développement dirigé par les tests est une pratique basée sur
  - Les tests unitaires
  - L'écriture des tests par les développeurs
  - L'écriture des tests avant celle du code

- L'écriture du code se fait suivant 6 itérations rapides

1. Conception rapide (optionnelle)
2. Ecriture d'un petit nombre de test unitaires **automatisés**
3. Lancement des tests pour vérifier qu'ils échouent (car pas encore de code correspondant)
4. Implantation du code à tester
5. Lancement des tests jusqu'à ce qu'ils passent tous
6. Restructuration du code et des tests si nécessaire pour améliorer les performances ou le design

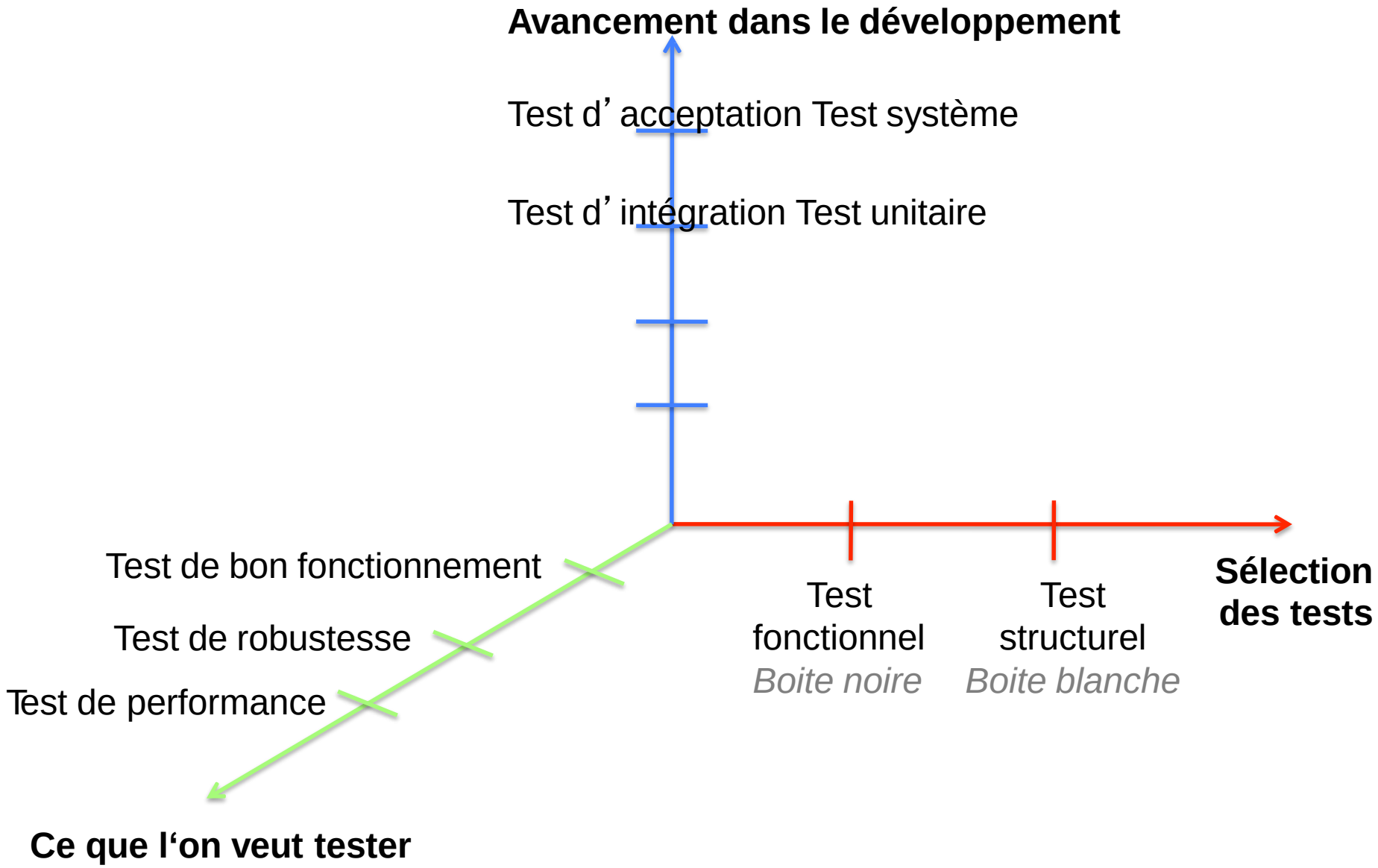
Comment écrire des tests automatiques

- Suivre le « Test Automation Manifesto »

Meszaros, Smith, et al, The Test Automation Manifesto, Extreme Programming and Agile Methods – XP/Agile Universe 2003, LNCS 2753, Springer.

- Les tests automatisés doivent être
  - Concis – écrit aussi simplement que possible
  - Auto-vérifiable – aucun intervention humaine ne doit être nécessaire
  - Répétable – toujours sans intervention humaine
  - Robuste – un test doit toujours produire le même résultat
  - Suffisants – ils doivent couvrir tous les besoins du système
  - Nécessaires – pas de redondances dans les tests
  - Clairs – faciles à comprendre
  - Spécifiques – en particulier chaque test d'échec pointe un point précis
  - Indépendants – ne dépendent pas de l'ordre d'exécution des tests
  - Maintenables – ils doivent pouvoir être aisément modifiés
  - Traçables

Quels types de tests ?



Le test dans le cycle de vie du logiciel

Selon l'avancement du développement, on distingue plusieurs types de test

- Le ***test unitaire*** consiste à tester des composants isolément
  - Test de fonctions, du comportement d'une classe
  - Doivent répondre à des scénarios de test préalablement établis
  - Généralement effectué dans les environnements de développement (éclipse+Junit, Visual Studio)
  - Facile à mette en œuvre
  - Cible : les méthodes et les classes

- Le ***test d'intégration*** consiste à tester un ensemble de composants qui viennent d'être assemblés. Il comporte 2 types de tests
  - Les *test de validité* – le nouveau module répond à ce qui a été demandé lorsqu'on l'intègre dans le système global
  - Les *tests de non régression* – L'intégration du nouveau module n'induit pas de régressions fonctionnelles (modification du comportement des modules existants) ou non fonctionnelles (instabilité) du système dans sa globalité
  - Cible : les composants et le système global

Le test dans le cycle de vie du logiciel

- Le ***test système*** consiste à tester le système sur son futur site d'exploitation dans des conditions opérationnelles et au-delà (surcharge, défaillance matérielle,…)
  - Cible : le système global sur l'architecture informatique du client

- Le ***test d'acceptation*** est la dernière étape avant l'acceptation du système et sa mise en œuvre opérationnelle
  - On teste le système dans les conditions définies par le futur utilisateur, plutôt que par le développeur.
  - Les tests d'acceptation révèlent souvent des omissions ou des erreurs dans la définition de besoins
- Donc des erreurs de validation et non de vérification

Dans tous les cas, pour assurer la traçabilité et la reproductibilité des tests (lors des étapes de maintenance), on doit fournir des documents indiquant les modules testés, les scénarios de test (entrées, environnements, etc.) et les résultats des tests

Le test dans le cycle de vie du logiciel

Test unitaire	Tester les modules, classes, opérations, au fur et à mesure de leur développement	Fait par le développeur du module (avant ou après écriture)
Test d'intégration	Tester l'assemblage des modules, des composants et leurs interfaces	Fait par le développeur du module ou un autre développeur
Test de système	Tester les fonctionnalités du système dans son ensemble	Développeurs
Test d'acceptation	Confirmer que le système est prêt à être livré et installé	Client

Types de test – ce que l'on veut tester!

- **Tests nominal ou test de bon fonctionnement**
  - Les cas de test correspondent à des données d'entrée valide.

Test-to-pass

- **Tests de robustesse**
  - Les cas de test correspondent à des données d'entrée invalides

Test-to-fail

- Règle usuelle : Les tests nominaux sont passés avant les tests de robustesse

- **Tests de performance**
  - *Load testing* (test avec montée en charge)
  - *Stress testing* (soumis à des demandes de ressources anormales)

Difficulté du test

- Limite théorique = Notion d'indécidabilité
  - propriété indécidable = qu'on ne pourra **jamais prouver** dans le cas général (pas de procédé systématique)
- Exemples de propriétés indécidables
  - l'exécution d'un programme termine
  - deux programmes calculent la même chose
  - un programme n'a pas d'erreur
  - un paramètre du programme fait passer l'exécution
    - sur une instruction, une branche, un chemin donné
    - sur toutes les instructions, branches ou chemins
- Une bataille perdue d'avance
  - un programme a un nombre infini (ou immense) d'exécutions possibles
  - un jeu de tests n'examine qu'un nombre fini d'exécutions possibles
- Trouver des heuristiques :
  - approcher l'infini (ou le gigantesque) avec le fini (petit)
  - tester les exécutions les plus « représentatives »

Difficulté du test

- Le test exhaustif est en général impossible à réaliser
  - En test fonctionnel, l'ensemble des données d'entrée est en général infini ou très grande taille

Exemple : la somme de 2 entiers 32 bits à raison de 1000 tests par seconde prendrait plus de 580 millions d'années

  - En test structurel, le parcours du graphe de flot de contrôle conduit à une forte explosion combinatoire
  - Conséquence

- le test est une méthode de vérification partielle de logiciels
- la qualité du test dépend de la pertinence du choix des données de test

Exemple célèbre (G.J. Myers, *The Art of Software Testing*, 1979)

> Un programme prend 3 entiers en entrée, qui sont interprétés comme représentant les longueurs des côtés d'un triangle. Le programme doit retourner un résultat précisant s'il s'agit d'un triangle scalène, isocèle ou équilatéral.

Quels cas test pour ce programme ?

Difficulté du test

- Difficultés d'ordre psychologique ou «culturel»
  - Le test est un processus destructif : un bon test est un test qui trouve une erreur alors que l'activité de programmation est un processus constructif - on cherche à établir des résultats corrects
  - Les erreurs peuvent être dues à des incompréhensions de spécifications ou de mauvais choix d'implantation
- L'activité de test s'inscrit dans le contrôle qualité, indépendant du développement

Des règles pratiques (1/2)

- Vos efforts pour les tests devraient toujours être axés pour essayer de faire *planter* votre code
  - Essayez des valeurs hors de la plage que vous avez spécifiée
  - Essayez des quantités d'information qui sont nulles, en trop, et des quantités moyennes.
  - Essayez des valeurs erronées (entrez une chaîne de caractères au lieu d'un nombre)…
- Les tests doivent donc tester les bornes pour lesquelles le programme a été écrit.

Des règles pratiques (2/2)

- Votre programme doit être écrit pour être testable
  - Les modules et les fonctions doivent être petites et cohésives
  - Si vous écrivez des fonctions qui sont complexes, vos tests vont être complexes
  - Vous devez toujours penser à vos tests quand vous faites le design de vos programmes, car tôt ou tard vous allez avoir à tester ce que vous avez produit

- Il y a des parties de votre code qui ne sont pas testables facilement
  - Les tests de défaillance, le traitement des erreurs et certaines conditions anormales
  - Les fuites mémoire ne peuvent pas être testées, vous devez inspecter votre code et vérifier que les blocs de mémoires sont libérés quand vous n'en avez plus de besoin

https://www.youtube.com/watch?v=AeIPqrt2s7E&list=PLmhQGrnqvRQxHbtA_AG0S_VQWJZS1zblN&index=6