Cours 8 - Tests

MODULE INTRODUCTION AU GÉNIE LOGICIEL

MOSTEFAI MOHAMMED AMINE – BATATA SOFIANE

Objectifs du Cours

Présenter l'activité de tests

Donner les différents types de tests

Introduire les tests en boîte noire

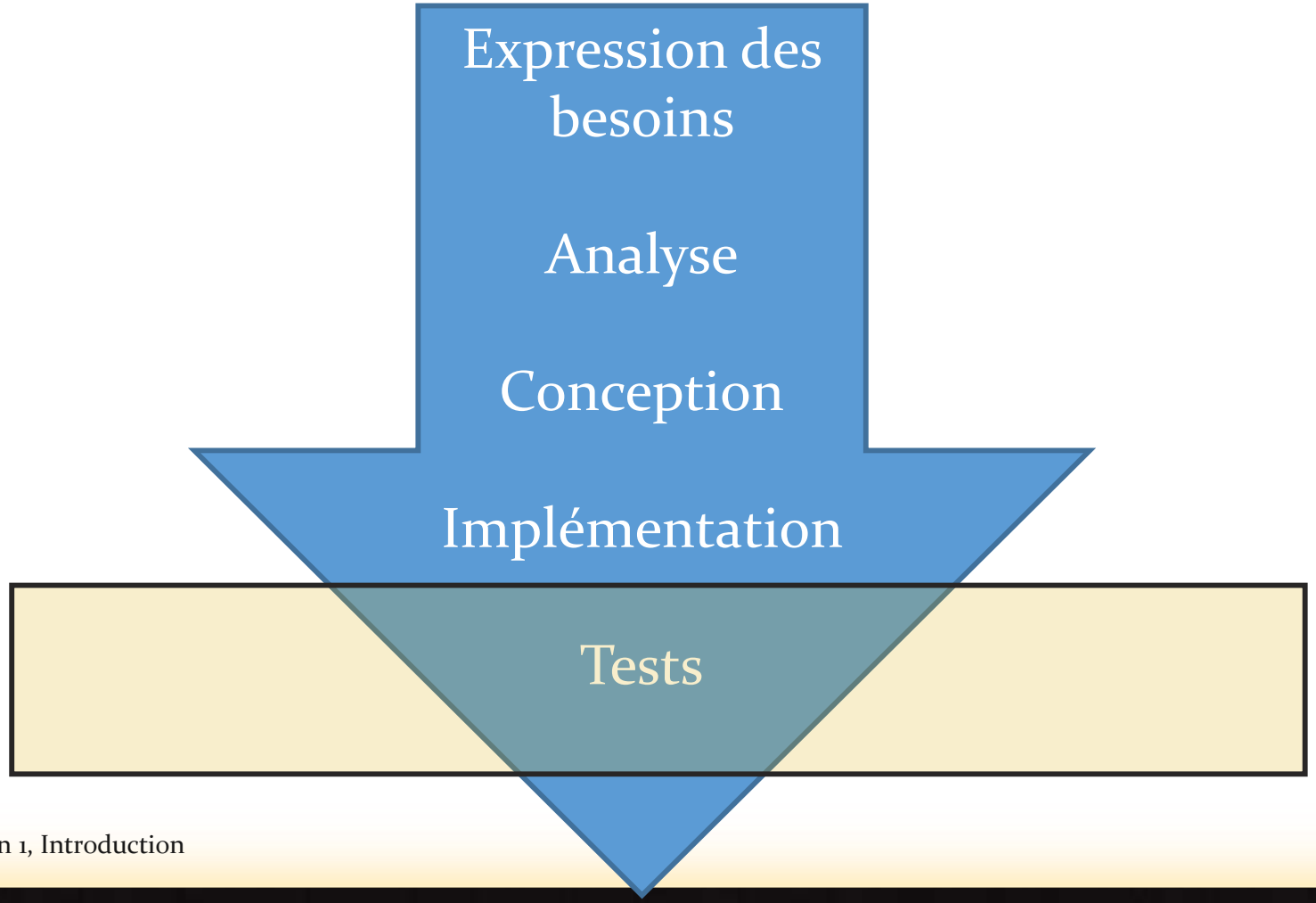
Introduire les tests unitaires

Plan du Cours

Introduction

SECTION 1

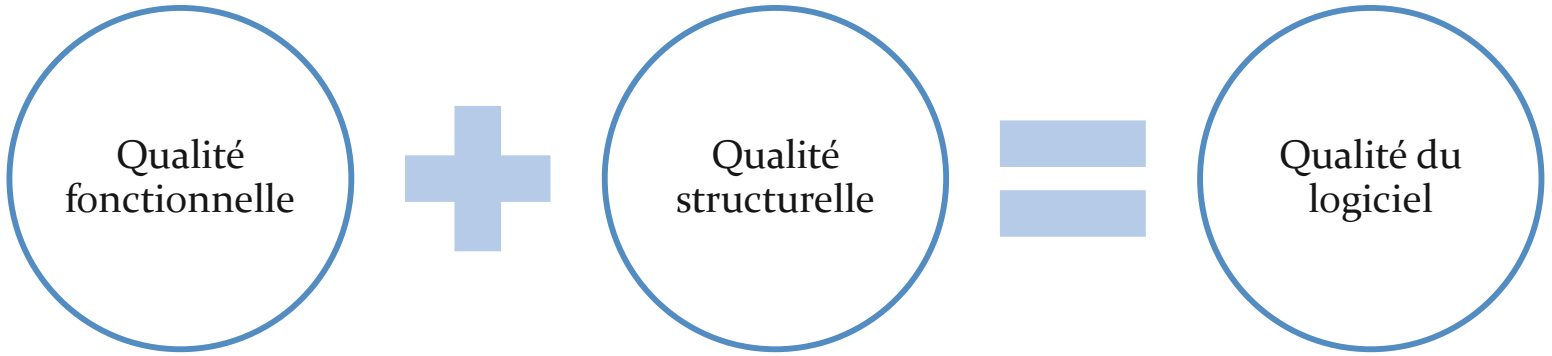
Cycle de Vie



Qualité d'un logiciel

- Les tests sont le procédé de ***validation*** et de ***vérification*** d'un logiciel
- Les tests assurent que le produit est ***conforme*** à ses spécifications
- Les tests assurent que le produit fait ce qu'on attend de lui
- Les tests identifient les ***bugs***, les ***erreurs*** ou les ***défaillances*** les plus ***importantes*** dans le but de les réparer
- Une défaillance importante est celle qui affecte ***l'utilisabilité*** du produit

Qualité de logiciels



Qualité d'un logiciel

- La qualité fonctionnelle assure que le produit est ***conforme*** à ses spécifications fonctionnelle.
- La qualité structurelle concerne la conformité aux spécifications ***non fonctionnelles*** (par exemple robustesse, rapidité, performance) et aussi à l'analyse de la ***structure interne*** du produit (code source, architecture,…)
- L'objectif des tests n'est pas d'améliorer la qualité mais plutôt de la mesurer

Objectifs des tests

Vérification

Validation

Recensement des défaillances

Objectifs des tests

- La vérification consiste à vérifier une spécification par rapport aux ***attentes techniques***. Par exemple, après la validation d'une facture, retrouver le champ « valide » à « true » dans la base de données
- La validation consiste à vérifier une spécification par rapport aux ***attentes métier***. Par exemple, après la validation d'une facture, la quantité du produit vendu doit décroître
- Une défaillance est une variation entre les résultats ***actuels*** et les résultats ***attendus***. L'origine de la défaillance peut être dans l'expression de besoins, la conception, l'analyse ou l'implémentation.

Objectifs des tests - réponses

Est-ce que ça marche comme prévu ?	Est-ce que c'est conforme aux spécifications ?	Est-ce que ça correspond aux attentes du client ?	Est-ce que le client apprécie ?
Est-ce que c'est compatible avec les autres systèmes ?	Est-ce que ça s'adapte à un nombre important d'utilisateurs ?	Quels sont les points à améliorer ?	Est-ce que c'est prêt pour le déploiement ?

Objectifs des tests - Suite

Economiser le temps et l'argent en identifiant rapidement les défaillances

Rendre le développement plus efficace

Augmenter la satisfaction du client

Correspondre le résultats aux attentes

Identifier les modifications à inclure dans les prochaines versions

Identifier les composants et les modules réutilisables

Identifier les lacunes des développeurs

Principes de tests

Commencer par les cas généraux avant les cas particuliers

Commencer par les cas prioritaires

Quand une fonction est signalée défaillante, la cause n'est pas forcément le code

Portée des tests

- Conformité aux spécifications fonctionnelles
- Conformité aux spécifications techniques
- Exigences juridiques
- Le code source
- Les restriction et recommandations des administrateurs du système
- Culture et standards de l'entreprise
- La configuration matérielle
- Propriétés culturelles et linguistiques

Introduction

SECTION 1 – DÉBAT (10 MNS)

Processus de Tests

SECTION 2

L'équipe de testeurs

- Les tests ne sont pas le travail d'un seul homme mais celui d'une ***équipe***
- La taille de cette équipe dépend de la ***taille*** et de la ***complexité*** du projet
- Les développeurs doivent avoir un rôle dans les tests mais d'une façon ***réduite***
- Le testeur doit être minutieux, critique (pas au sens jugement), curieux doté d'une bonne communication

L'équipe de testeurs – Questions à Poser

Comment pouvez-vous dire que ça marche ?

Que veut dire pour vous « ça marche » ?

Pourquoi ça marchait et que ça ne marche plus ?

Qu'est-ce qui a causé le mauvais fonctionnement ?

L'équipe de testeurs

- Le ***coordinateur de tests*** crée les plans de tests et les spécifications de tests sur la base des spécifications techniques et fonctionnelles
- Les ***testeurs*** exécutent les tests et documentent les résultats

Catégories de tests

Tests en boîte noire

- s'exécutent en ***ignorant*** les mécanismes internes du produit
- le testeur ***n'accède pas*** au code source

Tests en boîte blanche

- Prennent les mécanismes internes ***en considération***
- le testeur ***accède*** au code source

Types de tests

Tests unitaires

- Des tests de ***blocs individuels*** (par exemple des blocs de code)
- Généralement écrits par les ***développeurs*** eux-mêmes pour la validation de leurs classes et leurs méthodes
- Exécutés par les machines

Tests d'intégration

- Exécuté en boîte noire ou boîte blanche
- Vérifient que les composants ***s'intègrent bien*** avec d'autres composants ou systèmes
- Vérifient aussi que le produit est ***compatible*** avec l'environnement logiciel et matériel du client

Tests fonctionnels

- S'exécutent en boîte noire
- Vérifie que le produit ***assure les fonctionnalités*** spécifiées dans les spécifications fonctionnelles

Tests d'acceptation

- Tests fonctionnels rédigés et exécutés par le client

Types de tests - Suite

Tests système

- S'exécute en boîte noire
- S'oriente vers les ***spécifications non fonctionnelles***
- Composé de plusieurs tests :
  - ***Tests de stress*** : tester le produit au-delà des attentes non fonctionnelles du client. Par exemple, un système de sauvegarde qui doit tout sauvegarder deux fois par jour, le tester en mode trois fois par jour.
  - ***Tests de performance*** : évaluation par rapport à des exigences de performances données. Par exemple, un moteur de recherche doit renvoyer des résultats en moins de 30 millisecondes.
  - ***Tests de chargement*** : vérifie que l'application fonctionne dans des conditions normales (contraire aux tests de stress)

Tests de régression

- Sont un ***sous-ensemble*** des tests originaux
- Vérifient qu'une modification n'a pas eu ***d'effets de bord*** sur le produit
- Utiles parce que la réexécution de tous les tests est *très coûteuse*

Bêta-Tests

- Quand un produit est ***quasiment terminé***, il est distribué gratuitement à certains de ses utilisateurs
- Ces utilisateurs sont appelés ***testeurs béta***
- Ces utilisateurs vont utiliser le produit et reporter tout éventuel disfonctionnement de celui-ci

Comparaison entre les tests

Test	Portée	Catégorie	Exécutant
Unitaires	Petites portions du code source	Boîte blanche	Développeur Machine
Intégration	Classes / Composants	Blanche / noire	Développeur
Fonctionnel	Produit	Boîte noire	Testeur
Système	Produit / Environnement simulé	Boîte noire	Testeur
Acceptation	Produit / Environnement réel	Boîte noire	Client
Beta	Produit / Environnement réel	Boîte noire	Client
Régression	N'importe lequel	Blanche / noire	N'importe

Processus de Tests

SECTION 2, DÉBAT (05 MNS)

Tests en Boîte Noire

SECTION 3

Introduction

- Les tests en boîte noire détermine si un produit fait ce qu'on attend de lui ***indépendamment*** de sa structure interne
- Plusieurs tests sont effectués en boîte noire tels que les tests fonctionnels et les tests d'acceptation
- Les tests en boîte noire déterminent les fonctions manquantes ou mal implémentées, les erreurs d'interface, des anomalies de performance ou de comportement, les bugs et les crashs
- Quels que soit la consistance ou le volume des tests, ***aucune garantie*** n'est donnée sur l'absence d'erreurs

Introduction

- Il vaut mieux que les testeurs soient des personnes différentes des programmeurs
- Les testeurs s'intéressent à une fonction en terme d'entrée et de sorties

Le plan de tests

- La planification des tests doit se faire dans les ***premières phases*** du projet
- Le plan de tests est un document ***obligatoire*** déterminant le ***déroulement*** des tests durant le projet
- Le tableau suivant indique le contenu d'un plan de test selon le standard « American National Standards Institute and Institute for Electrical and Electronic Engineers Standard 829/1983 for Software Test Documentation"

Le plan de tests - Composition

Élément	Description	Objectif
Responsabilités	Acteurs et affectation	Décrit qui fait quoi dans les tests. Assure le suivi et les affectations
Tests	Portée de tests, plannings, durées et priorités	Définit le processus et détaille les actions à entreprendre
Communication	Plan de communication	Tout le monde doit savoir ce qu'il doit savoir avant les tests et ce qu'il doit faire savoir après les tests
Analyse de risques	Éléments critiques à tester	Identification des domaines qui sont critiques dans le projet
Traçage des défaillance	Documentation des défaillances	Documenter les défaillances et les détails les concernant

Priorisation des défaillances

- Le plan de tests détermine la ***priorisation*** des défaillances
- La priorisation facilite la communication entre développeurs et testeurs ainsi que l'affectation et la planification des tâches

Priorisation des défaillances - Exemple

Priorité	Description
1 – Fatal	Impossible de continuer les tests à cause de la sévérité des défaillances
2- Critique	Les tests peuvent continuer mais l'application ne peut passer en mode production
3- Majeur	L'application peut passer en mode production mais des exigences fonctionnelles importantes ne sont pas satisfaites
4- Medium	L'application peut passer en mode production mais des exigences fonctionnelles sans très grand impact ne sont pas satisfaites
5- Mineur	L'application peut passer en mode production, la défaillance doit être corrigée mais elle est sans impact sur les exigences métier
6- Cosmétique	Défaillances mineures relatives à l'interface (couleurs, police, …) mais n'ayant pas une relation avec les exigences du client

Cas de test (Test Case)

- Un cas de test est un ensemble ***d'entrées de tests***, de ***conditions d'exécution*** et de ***résultats attendus*** pour un objectif particulier tel que la conformité du programme avec une spécification données

Anatomie d'un cas de test

ID
Description
Priorité
Préconditions
Scénario
Résultats attendus
Résultats actuels
Remarques

Anatomie d'un cas de test

- L'ID est un ***numéro unique*** qui permet la traçabilité des cas de test, les lier à des spécifications ou à d'autres cas de test
- Les préconditions déterminent les ***conditions nécessaires*** à un cas de test. Elles indiquent aussi les ***cas de test qui doivent être exécutés précédemment***
- La description est un texte ***décrivant*** le tests et ses attentes
- Le scénario détermine les ***étapes détaillées*** à suivre par le testeur
- Les résultats attendus sont ***attendus*** de l'exécution du scénario
- Les résultats actuels sont les ***vrais résultats*** obtenus, s'ils sont différents des résultats attendus, une défaillance est signalée
- Les remarques sont signalées par le testeur pour ajouter des information concernant une exécution donnée

Anatomie d'un cas de test

- Le scénario est composé de plusieurs ***actions numérotées***
- Chaque action ***peut*** avoir un ***résultat attendu***
- L'exécution de test affecte un ***résultat actuel*** à chaque étape attendant un résultat
- Si chaque résultat actuel est ***conforme*** au résultat attendu alors le test ***réussit*** sinon le test ***échoue***

Exemple 1

ID	1
Description	Ouverture d'un document
Préconditions	Existence d'un document
Scénario	1- Cliquez sur le bouton « Ouvrir » 2- Sélectionnez le fichier désiré 3- Appuyez sur OK
Résultats attendus	1- Une boîte d'ouverture de fichiers s'affiche 2- 3- Le Document est ouvert et prêt à être manipulé
Résultats Actuels	1- 2- 3-

Exemple 2

ID	2
Description	Impression d'un document
Préconditions	TC 1
Scénario	1- Cliquez sur le bouton imprimer 2- Sélection la config puis cliquer sur « Imprimer »
Résultats attendus	1- La boîte de configuration d'imprimante s'affiche 2- Le document sort sous format papier
Résultats actuels	1- 2-

Campagne de tests

- Une campagne de tests sélectionne un certain nombre de cas de tests sémantiquement relatifs et les exécute
- Chaque campagne de tests a un objectif tracé : validation du produit, performance, … etc.

Exercice

- Ecrire un cas de test d'inscription à un mail en tenant en compte des cas particuliers suivants : adresse mail non valide, mail existant, mot de passe non sécurisé

Tests en Boîte Noire

SECTION 3, DÉBAT 05 MNS

Tests Unitaires

SECTION 4

Introduction

- Les tests unitaires sont des tests en boîte blanche
- Les tests unitaires sont composé d'un ensemble de classes appelées « ***classes de test*** »
- Ces classes valident que des portions de code répondent à un certain besoin
- Les tests unitaires sont importants car ils permettent de détecter ***le maximum de défaillance*** avant les tests en boîte noire et qu'ils peuvent s'exécuter d'une manière automatique

Objectifs

- Les tests unitaires ont deux objectifs : « ***couverture de code*** » et « ***couverture de données*** »
- La couverture du code stipule de tester chaque ligne de code écrite (appel de fonction, boucles, décisions, décisions multiples,…)
- La couverture des données oriente les tests vers les données (données valides, données invalides, accès aux éléments en dehors de la capacité d'un tableau, peu de données, trop de données,…etc)

Caractéristiques

- Les tests unitaires doivent être conforme à l'acronyme ***FIRST***:
- ***FAST*** : un test unitaire doit s'exécuter rapidement
- ***INDEPENDANT***: chaque test doit être indépendant des autres
- ***REPEATABLE***: chaque test peut être répété autant de fois que voulu
- ***SELF-VALIDATING***: Un test se valide lui-même par un succès ou par un échec
- ***TIMELY***: On écrit les tests quand on en a besoin

Exemple

```
public class Calculator
{
public int add(int x, int y)
{
return x + y;
}
public int multi(int x, int y)
{
        return x*y;
}
}
```

Exemple

```
// tester la classe
public class CalculatorTest
{
// tester l'addition
public int addTest()
{
        // valeurs en entrée
        int a = 15;
        int b = 18;
        Calculator calc = new Calculator();
        int c = calc.Add(a,b);
        // résultat attendu et résultat actuel
        assertEquals(33, c);
}

}
```

Tests Unitaires

SECTION 4, DÉBAT 05 MNS

Bibliographie

- UML Component Diagrams, Veronica Carrega, 2004
- Introduction to Software Architecture”David Garlanand Mary Shaw, January 1994
- Analyse, Conception Objet
- Diagrammes de déploiement, SIMMO/ENSM.SE, 2002