Test Logiciel - Master 1

Christopher Rousseau

Quizz

Qu’avez-vous retenu ?

Quizz

La politique de test est :

1. Définie par la direction
2. Validée par la direction
3. Définie au niveau projet
4. Définie au niveau entreprise

(2 réponses)

Quizz

La politique de test est définie au niveau entreprise et validée par la direction, c'est au test manager de la définir, parfois en collaboration.

Quizz

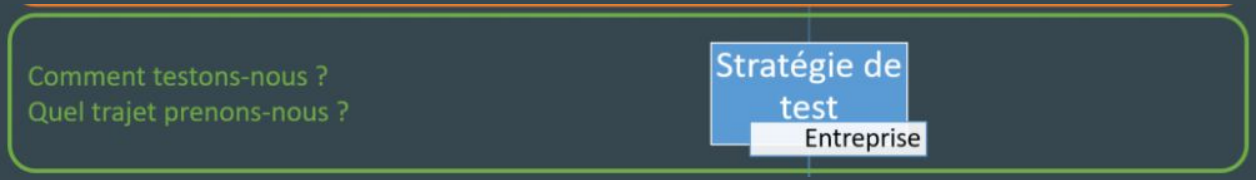
La stratégie de test est :

1. Définie par l'analyste de test
2. Définie par le test manager
3. Définie au niveau projet
4. Définie au niveau entreprise

(2 réponses)

Quizz

La stratégie de test est définie au niveau entreprise, c'est au test manager de la définir, parfois en collaboration.



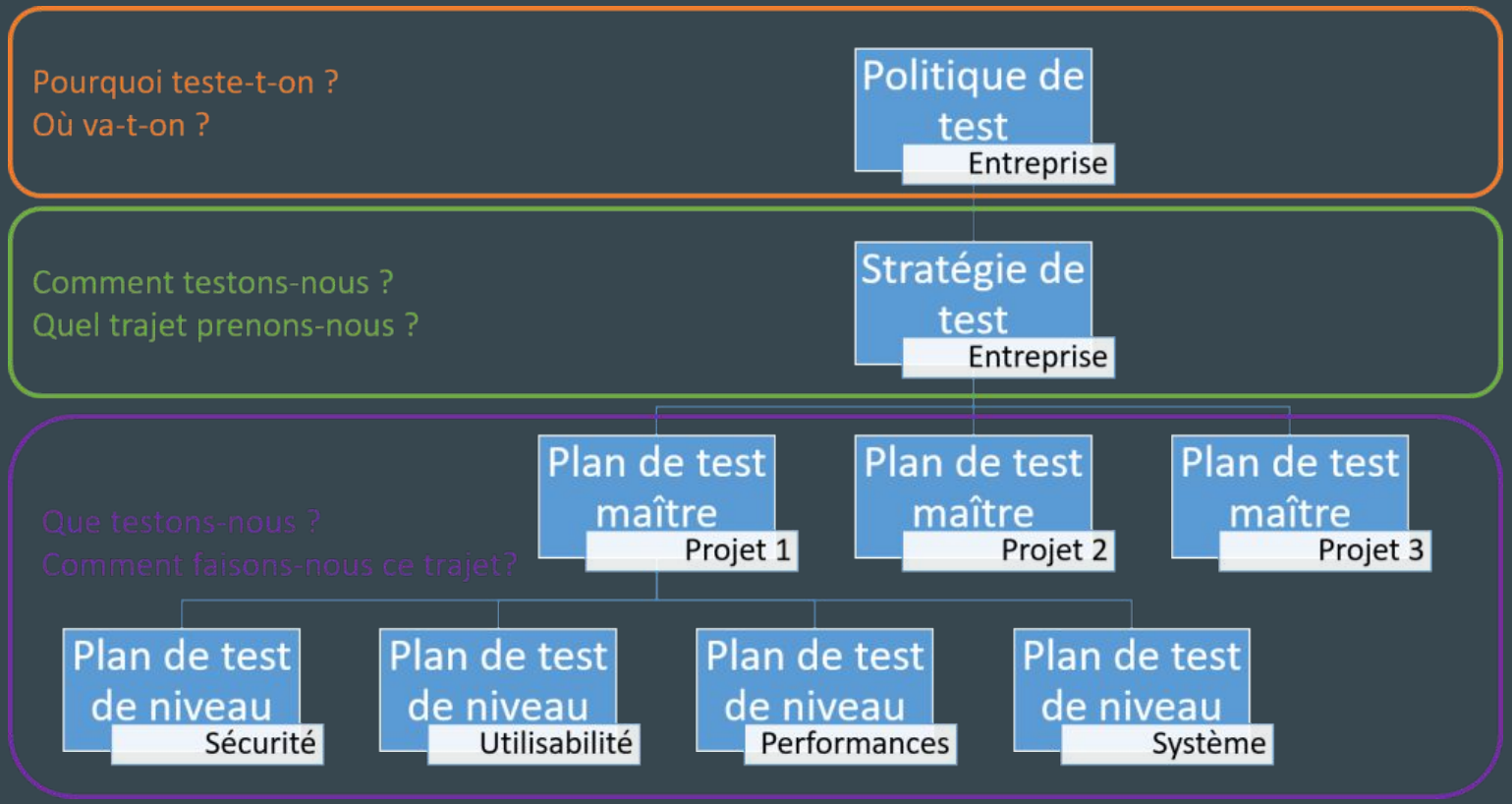
Quizz

Un plan de test :

1. Peut être défini à plusieurs niveaux
2. Peut regrouper plusieurs plans de niveaux au sein d'un plan de test maître
3. Est une preuve d'exécution des tests
4. Est appliqué au niveau entreprise

(2 réponses)

Les documents liés au test



https://latavernedutesteur.fr/2019/08/19/politique-strategie-et-plans-de-test/

Quizz

Une campagne de test est :

1. Le résultat du paradoxe du pesticide
2. L'exécution d'un test
3. Une planification de l'exécution d'une suite de test
4. Un plan de test maître

Les documents liés au test

Campagne de test

Une campagne de test est un ensemble de cas de tests à exécuter sur une période donnée. Le résultat de ces exécutions est alors synthétisé dans un bilan.

Quizz

Une suite de tests est constituée de :

1. Plusieurs pas de test
2. Une date et une matrice d'exécution de tests
3. Plusieurs cas de test
4. Une description d'environnement accompagné des risques métiers

Les documents liés au test

Suite de tests

Suite de tests : Ensemble de cas de test ou de procédures de test à exécuter dans un cycle de test spécifique

Cas 1	Etapes
Cas 2	Etapes
Cas 3	Etapes

Quizz

La différence entre une suite de test et une matrice d'exécution des tests ?

1. Le nombre de cas de tests
2. La date
3. La description de l'environnement
4. Le statut d'exécution

Les documents liés au test

Matrice d'exécution des tests

Cas 1	Etapes	**OK**
Cas 2	Etapes	**OK**
Cas 3	Etapes	**KO**

Quizz

Quelle réponse n’est pas une couverture de tests :

1. La couverture de méthodes
2. La couverture de racines
3. La couverture de branches
4. La couverture d’instructions

Les couvertures de tests

Couverture : le degré, exprimé en pourcentage, selon lequel un élément de couverture spécifié a été exécuté lors d'une suite de test.

- La couverture des méthodes (des fonctionnalités)
- La couverture des instructions (du code)
- La couverture des chemins ou de décisions

Quizz

Comment calcule t-on la criticité ?

1. A partir des priorités métiers
2. A partir du temps de réponse d'une API
3. A partir de la probabilité et de l'impact
4. En comptant le nombre d'appels

Les documents liés au test

L'analyse de risques

		Niveau de gravité			
		Insignifiant	Marginal	Critique	Catastrophique
Probabilité	Très probable	A gérer	Inacceptable	Inacceptable	Inacceptable
	Probable	A gérer	A gérer	Inacceptable	Inacceptable
	Possible	Néglibeable	A gérer	A gérer	Inacceptable
	Peu probable	Néglibeable	Néglibeable	A gérer	Inacceptable
	Très improbable	Néglibeable	Néglibeable	A gérer	A gérer

MATRICE DE CRITICITE DES RISQUES – Pour un projet exigeant

https://savoir.plus/gestion-de-projet/analyse-risques/

Les couvertures de tests

La couverture de code

Couverture de code : une méthode d'analyse qui détermine quelles parties du logiciel ont été exécutées (couvertes) par une suite de tests et quelles parties ne l'ont pas été, p.ex. couverture des instructions, des décisions ou des conditions.

Les couvertures de tests

La couverture de méthodes (fonctions)

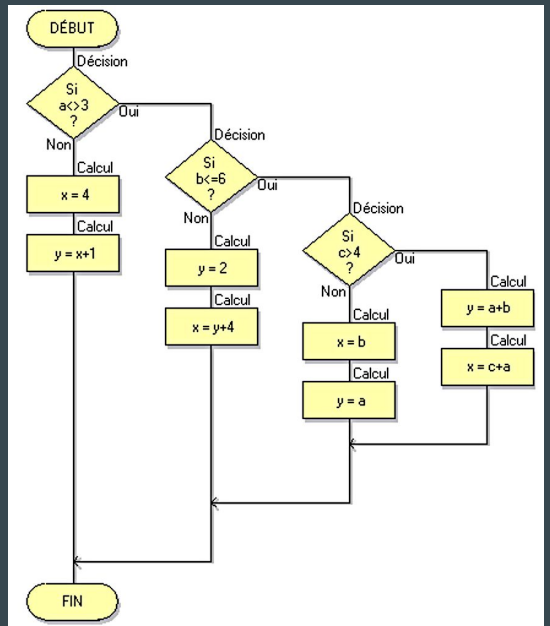
Couverture de code : une méthode d'analyse qui détermine quelles parties du logiciel ont été exécutées (couvertes) par une suite de tests et quelles parties ne l'ont pas été, p.ex. couverture des instructions, des décisions ou des conditions.

```
public interface Compte {

  void deposer(int montant) throws OperationInterrompueException,
                                   CompteBloqueException;

  int retirer(int montant) throws OperationInterrompueException,
                                  CompteBloqueException;

  int getBalance() throws OperationInterrompueException;

  }
```

Les couvertures de tests

La couverture d'instructions

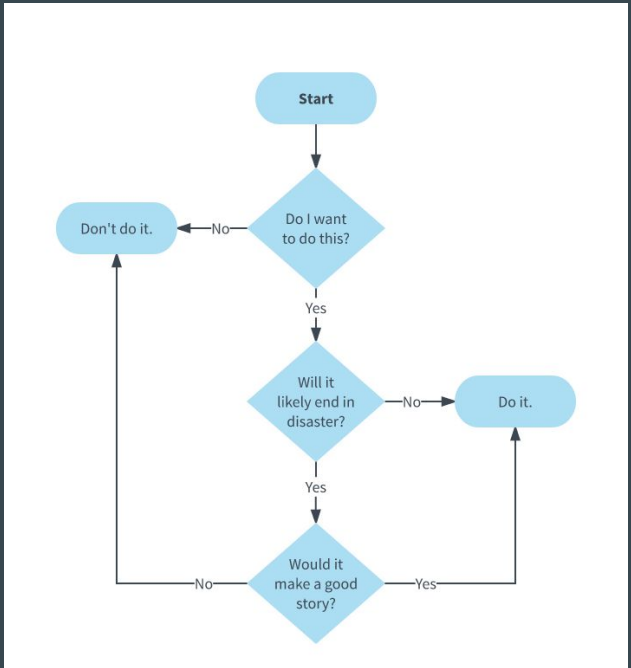
Couverture des instructions : le pourcentage des instructions exécutables qui ont été exécutées par une suite de tests.



Les couvertures de tests

La couverture des chemins ou de décisions

Couverture des décisions : La couverture des résultats des décisions.



Un exemple concret

Ecrire une classe et la tester

- Classe
- Méthode
  - Avec condition
  - Sans condition

Un exemple concret

```java
package calculator;

public class Calculation {
    int result;
    public int add(int num1, int num2){
        result = num1+num2;
        return result;
    }
    public int substract(int num1, int num2){
        result = num1-num2;
        return result;
    }
    public int multiply(int num1, int num2){
       if (num1 == 0 || num2 == 0) {
           result =  0;
       } else {
        result = num1*num2;
       }
        return result;
    }
    public float divide(int num1, int num2){
       if (num2 == 0) {
           throw new IllegalArgumentException ();
       }
        result = num1/num2;
        return result;
    }
}
```

```java
public class Calculation {
    int result;
    public int add(int num1, int num2){
        result = num1+num2;
        return result;
    }
    public int substract(int num1, int num2){
        result = num1-num2;
        return result;
    }
    public int multiply(int num1, int num2){
       if (num1 == 0 || num2 ==0) {
           result = 0;
       } else {
        result = num1*num2;
       }
        return result;
    }
    public float divide(int num1, int num2){
       if (num2 == 0) {
           throw new IllegalArgumentException ();
       }
        result = num1/num2;
        return result;
    }
}
```

Analyse des risques

Plusieurs facteurs

- Facteur business
- Facteur projet
- Facteur qualité
- Facteur environnemental

Analyse des risques

Des exemples de risques

- Les risques liés à la concurrence
- Les risques liés à la réglementation
- Les risques technologiques
- Les risques humains
- Les risques liés à la taille du projet
- Les risques liés au secteur d’activité

Analyse des risques

Méthode d'amélioration continue

Problème posé	•Description de la situation, recensement des dangers et identification des risques
Evaluation des risques	•Tableau d'analyse reprenant : •Données géométriques •Données objectives •Données subjectives
Relevé des mesures de prévention	•Evaluation des mesures adéquates par rapport au risque (supprimer, réduire, informer, surveiller)
Relevé des objectifs et contraintes	•Objectifs recherchés par le(s) demandeur(s) •Contraintes (budget, avis contraires, contraintes géométriques...)
Relevés des aménagements possibles	•Examen des différentes propositions en fonction de l'analyse de risque et de la définition des mesures adéquates mais aussi du coût et de la cohérence
Définition du mode de traitement	•Est-ce que la solution va être mise en œuvre directement car il est aisé de lancer une action ? Est-ce qu'elle va être réalisée dans le cadre d'un marché en cours ou faudra-t-il recourir à un nouveau marché ?
Synthèse graphique	•Tableau récapitulatif qui permet de reprendre les différents aménagements, les avantages et inconvénients et le mode de traitement
Décision	•Choix en fonction de l'angle d'approche privilégié

Analyse des risques

Méthode de test basée sur les risques

Le « risk based testing » utilise les risques, soit la probabilité d'apparition d'un bug, pour prioriser et adapter les tests à exécuter lors d'une campagne. Chaque bug est ainsi associé à une criticité. Plus la criticité est forte, plus l'impact sera grand voire bloquant, pour l'utilisation de l'application ou du site à tester.

En pratique, il est compliqué de supprimer tous les bugs en adoptant une méthode de test basée sur les risques. L'objectif principal est d'établir un équilibre entre risque, qualité, fonctionnalités, budget et timing.

https://blog.testingdigital.com/zoom-methode-de-tests-basee-risques-232#:~:text=Le%20%C2%AB%20risk%20based%20testing%20%C2%BB%20utilise,ainsi%20associ%C3%A9%20%C3%A0%20une%20criticit%C3%A9.

Analyse des risques

Méthode de test basée sur les risques

- Lister les bugs probables en les classant par ordre de priorité. Du risque le plus bloquant au moins critique
- Programmer des tests pour chaque risque (automatisation)
- Mettre à jour les risques, et continuer d'ordonner l'exécution en fonction de la criticité.

En cas de réduction du timing impactant les phases de tests, ou en phase de tests de non régression le Risk Based Testing permettra de tester en priorité les fonctionnalité les plus importantes.

https://blog.testingdigital.com/zoom-methode-de-tests-basee-risques-232#:~:text=Le%20%C2%AB%20risk%20based%20testing%20%C2%BB%20utilise,ainsi%20associ%C3%A9%20%C3%A0%20une%20criticit%C3%A9.

Les plans de test dans la pratique

Les tables de décisions

La modified condition / decision coverage (MC/DC - couverture des conditions modifiées / décisions)

- Condition : Une condition est une expression booléenne atomique (ne contenant pas d'opérateur booléen).
- Décision : Une décision est une expression booléenne composée de Conditions et éventuellement d'opérateurs booléens. Une décision sans opérateur booléen est aussi une condition.

Par exemple la décision (A or B) and (A or C) comporte 4 conditions.

Les plans de test dans la pratique

Les tables de décisions

- Chaque décision teste toutes les sorties possibles
- Chaque condition dans une décision prend toutes les sorties possibles
- Chaque point d'entrée et de sortie est passé
- Chaque décision teste toutes les sorties possibles
- Il est démontré que chaque condition dans une décision affecte indépendamment la sortie de la décision.

Les plans de test dans la pratique

Les tables de décisions

- Exemple Un système de paiement
  - Echoue
    - Si la carte n'est pas valide
    - Si la carte n'est pas créditée
    - S'il y a une erreur lors de la transaction
  - Réussi
    - Si la carte est provisionnée, valide et qu'il n'y a pas d'erreurs
- a.b.ō

Les plans de test dans la pratique

Les tables de décisions

- a.b.ō

Résultat	Carte valide	Compte provisionné	Erreur à l'execution

Les plans de test dans la pratique

Les tables de décisions

- a.b.c̄

Résultat	Carte valide	Compte provisionné	Erreur à l'execution
Succès	True	True	False
Echec	True	True	True
Echec	False	True	True
Echec	False	False	False
Echec	False	False	True
Echec	True	False	True
Echec	False	True	False
Echec	True	False	False

Des outils bien pratiques

Git

https://git-scm.com/

git --distributed-is-the-new-centralized

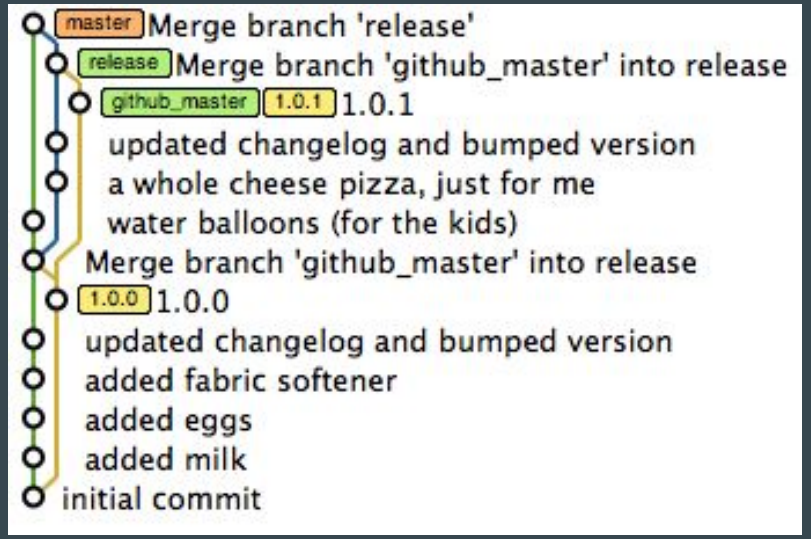
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Des outils bien pratiques

Git

https://git-scm.com/



Des outils bien pratiques

Maven

Gestionnaire de paquets & de cycle de vie

https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Des outils bien pratiques

Maven

Un gestionnaire de cycle de vie : lifecycle

- compile: compile the source code of the project
- test: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- verify: run any checks to verify the package is valid and meets quality criteria
- install: install the package into the local repository, for use as a dependency in other projects locally
- clean: cleans up artifacts created by prior builds
- site: generates site documentation for this project

https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Des outils bien pratiques

Maven

Un gestionnaire de cycle de vie : lifecycle

- validate: validate the project is correct and all necessary information is available
- package: take the compiled code and package it in its distributable format, such as a JAR.
- integration-test: process and deploy the package if necessary into an environment where integration tests can be run
- deploy: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

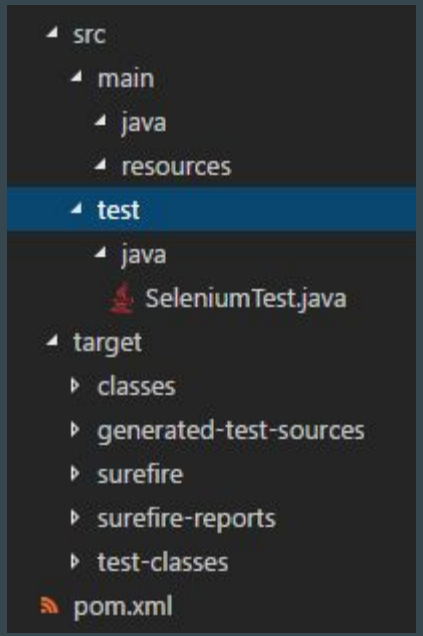
Des outils bien pratiques

Maven

mvn --version

mvn archetype:generate -DgroupId=com.mycompany.app
-DartifactId=my-app
-DarchetypeArtifactId=maven-archetype-quickstart
-DarchetypeVersion=1.4 -DinteractiveMode=false

Un fichier de configuration : pom.xml



https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Des outils bien pratiques

Maven

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.mycompany.app</groupId>
 <artifactId>my-app</artifactId>
 <version>1.0-SNAPSHOT</version>

 <properties>
   <maven.compiler.source>1.7</maven.compiler.source>
   <maven.compiler.target>1.7</maven.compiler.target>
 </properties>

 <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>4.12</version>
     <scope>test</scope>
   </dependency>
 </dependencies>
</project>
```

Des outils bien pratiques

Maven

Générons un projet

Des outils bien pratiques

JUnit

Testons un projet

Des outils bien pratiques

JUnit

Testons un projet

Des outils bien pratiques

Assertj

Testons un projet

https://assertj.github.io/doc/

Des outils bien pratiques

JMeter / Gatling

Description et comparaison

https://octoperf.com/blog/2015/06/08/jmeter-vs-gatling/#jmeter-test-execution

PLAN

Stratégie de tests

- Test unitaire dans la pratique
- Mock dans la pratique
- Le test d'integration dans la pratique
- Les tests API
- L'automatisation des tests API
- Le test IHM
- L'automatisation des tests IHM
- Le troubleshooting