Tests

Table of Contents

1. Les Tests

– We don't write tests. – Why? – Because we don't have time for it. – Why? – Because there are too much work and pressure. – Why? – Because we don't move fast enough. – Why? – Because changing software has become difficult and risky. – Why? – Because we don't write tests.

Quelle que soit la méthode de développement choisie, les Tests sont le **seul moyen de garantir** que le produit livré est **conforme aux exigences du client**.

1.1. Tests unitaires

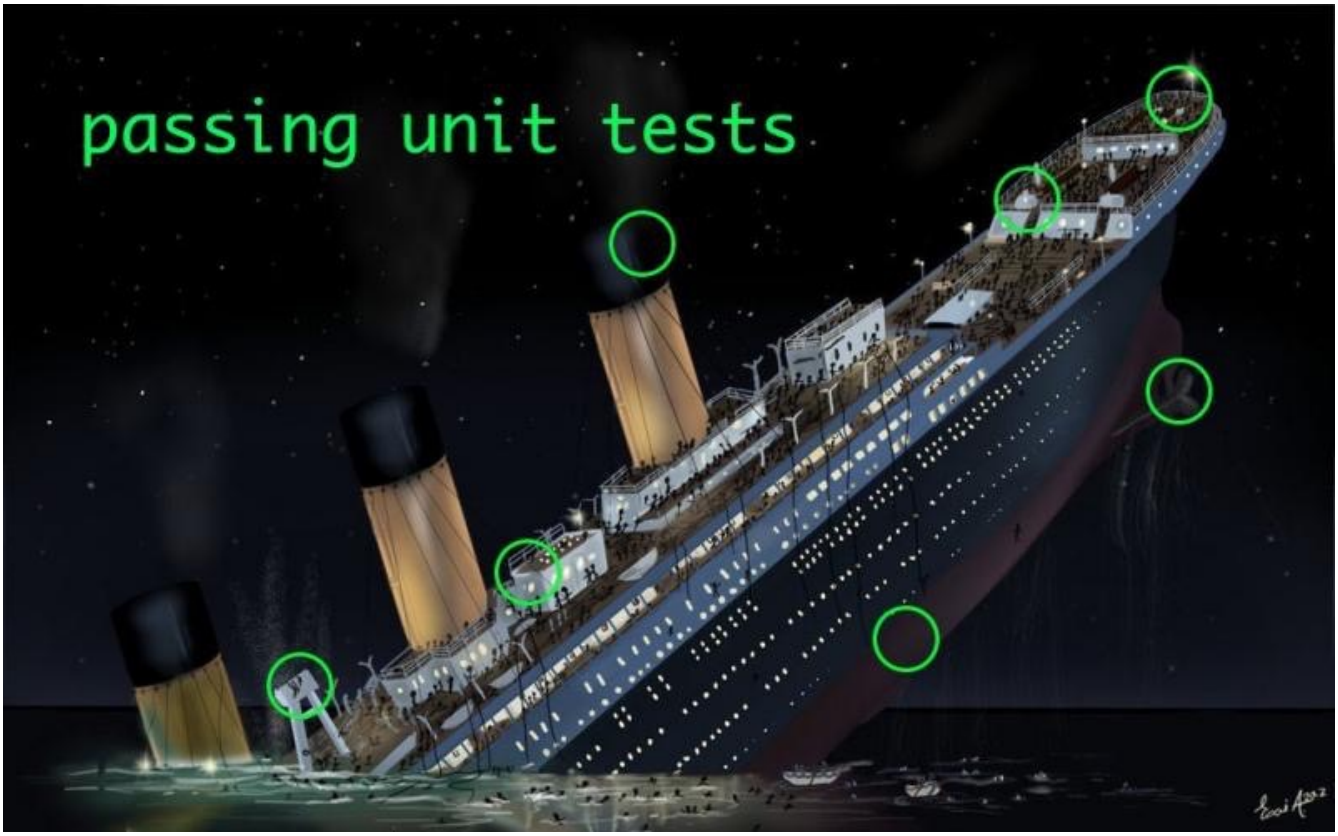
Ce sont les plus simples. Et pourtant il s'agit …

- d'oublier les approches 'manuelles'
- d'expliciter les 'limites'
- de traiter le 'qualitatif' et le 'quantitatif'

Le programme de test fourni est :

- un outil de **détection des régressions**
  - qui pourraient intervenir à la suite d'une modification du code
- une *documentation* de **spécification**
  - précise et concise
- une *documentation* pour les **programmeurs**
  - opérationnelle

1.2. Tests d’intégration



Plus délicat, il s’agit :

- de tester les exigences du client
- de tester les intéractions système
- de ne pas refaire les tests unitaires

Programme de Test du programme `JourSuivantAvecLibDate.class`

```
//import junit.textui.TestRunner;
import junit.framework.TestSuite;
import junit.framework.TestCase;
import java.io.*;

public class JourSuivantAvecLibDateTest extends TestCase {
  static String programmeATester = "JourSuivantAvecLibDate" ;  ①
  Process executionProgrammeATester ;                          ②
  BufferedReader ecranProgrammeATester ;                       ③
  BufferedWriter clavierProgrammeATester ;                     ④

  String finDeLigne = System.getProperty("line.separator") ;   ⑤

  public static void main(String[] args) {
    if ( args.length > 0 ) { programmeATester = args[0] ; }
    System.out.println("Tests du programme : " + programmeATester);
    junit.textui.TestRunner.run(new TestSuite(JourSuivantAvecLibDateTest.class)); ⑥
  }
```

```
  protected void setUp() throws IOException {  ⑦
    executionProgrammeATester = Runtime.getRuntime().exec("java -cp .
"+programmeATester); ⑧
    ecranProgrammeATester = new BufferedReader(new  InputStreamReader(
executionProgrammeATester.getInputStream() )); ⑨
    clavierProgrammeATester  = new BufferedWriter(new OutputStreamWriter(
executionProgrammeATester.getOutputStream() )); ⑩
  }

  // Saisies valides
  public void test_31_1_2013() throws IOException {
    assertEquals("Affiche : 'Saisir une date : jour mois annee ? '","Saisir une date :
jour mois annee ? ",ecranProgrammeATester.readLine()); ⑪
    clavierProgrammeATester.write("31 1 2013"+finDeLigne); ⑫
    clavierProgrammeATester.flush();                        ⑬
    assertEquals("Affiche : 'Le lendemain du 31/1/2013'","Le lendemain du
31/1/2013",ecranProgrammeATester.readLine());
    assertEquals("Affiche : 'sera le 1/2/2013.'","sera le
1/2/2013.",ecranProgrammeATester.readLine()); ⑭
  }

  public void test_28_2_2013() throws IOException {
    String messageSaisie = "Saisir une date : jour mois annee ? " ;
    String[] ligneJeuDEssai = {"28 2 2013","Le lendemain du 28/2/2013","sera le
1/3/2013."} ;

    assertEquals("Affiche :
"+messageSaisie,messageSaisie,ecranProgrammeATester.readLine());
    clavierProgrammeATester.write(ligneJeuDEssai[0]+finDeLigne);
clavierProgrammeATester.flush();
    assertEquals("Affiche :
"+ligneJeuDEssai[1],ligneJeuDEssai[1],ecranProgrammeATester.readLine());
    assertEquals("Affiche :
"+ligneJeuDEssai[2],ligneJeuDEssai[2],ecranProgrammeATester.readLine());
  }

  protected void assertsPourSaisieValide(String messageSaisie,String saisie,String
affichage1,String affichage2) throws IOException {
     assertEquals("Affiche :
"+messageSaisie,messageSaisie,ecranProgrammeATester.readLine());
     clavierProgrammeATester.write(saisie+finDeLigne);
clavierProgrammeATester.flush();
     assertEquals("Affiche :
"+affichage1,affichage1,ecranProgrammeATester.readLine());
     assertEquals("Affiche :
"+affichage2,affichage2,ecranProgrammeATester.readLine());
  }
  public void test_31_3_2013() throws IOException {
    String messageSaisie = "Saisir une date : jour mois annee ? " ;
    String[] ligneJeuDEssai = {"31 3 2013","Le lendemain du 31/3/2013","sera le
```

```
1/4/2013."} ;

assertsPourSaisieValide(messageSaisie,ligneJeuDEssai[0],ligneJeuDEssai[1],ligneJeuDEss
ai[2]);
  }

  // Saisies invalides
  protected void assertsPourSaisieInvalide(String messageSaisie,String saisie,String
affichage) throws IOException {
     assertEquals("Affiche :
"+messageSaisie,messageSaisie,ecranProgrammeATester.readLine());
     clavierProgrammeATester.write(saisie+finDeLigne);
clavierProgrammeATester.flush();
     assertEquals("Affiche : "+affichage,affichage,ecranProgrammeATester.readLine());
  }

  public void test_1_1_1581() throws IOException {
    String messageSaisie = "Saisir une date : jour mois annee ? " ;
    String[] ligneJeuDEssai = {"1 1 1581","La date du 1/1/1581 n'est pas une date
valide."} ;
    assertsPourSaisieInvalide(messageSaisie,ligneJeuDEssai[0],ligneJeuDEssai[1]);
  }

  public void test_32_1_2013() throws IOException {
    String messageSaisie = "Saisir une date : jour mois annee ? " ;
    String[] ligneJeuDEssai = {"32 1 2013","La date du 32/1/2013 n'est pas une date
valide."} ;
    assertsPourSaisieInvalide(messageSaisie,ligneJeuDEssai[0],ligneJeuDEssai[1]);
  }

} // fin class
```

① nom de l'application (du programme) à tester

② Process (Processus) = programme en cours d'exécution

③ lien vers l'écran du programme en cours d'exécution

④ lien vers le clavier du programme en cours d'exécution

⑤ récupération **portable** du retour à la ligne

⑥ lancement de toutes les fonctions débutant par 'test'

⑦ fonction (ré-)exécutée avant chaque fonction de test et qui exécute le programme à tester

⑧ lance le programme

⑨ se connecte à l'écran (sortie standard) du programme lancé

⑩ se connecte au clavier (entrée standard) du programme lancé

⑪ lit une ligne sur l'écran du programme lancé

⑫ écrit une ligne au clavier du programme lancé

⑬ force l'envoi de la ligne au clavier (vide le tampon de sortie)

⑭ lit une autre ligne sur l'écran du programme lancé

1.3. Les tests implémentent des algorithmes simples

given/when/then

```
public void test_dates_invalides() {
  int[][] tabJeuDEssaiDatesInvalides = {  ①
      {1,1,1581},{0,1,2013},{99,99,2099},
      {32,1,2013},{29,2,2013},{32,3,2013},
      {31,4,2013},{32,5,2013},{31,6,2013},
      {32,7,2013},{32,8,2013},{31,9,2013},
      {32,10,2013},{31,11,2013},{32,12,2013},
      {29,2,1900},{30,2,2000}
  } ;
  for ( int indice = 0, taille = tabJeuDEssaiDatesInvalides.length;
        indice < taille ;
        indice = indice + 1){
    int[] date = tabJeuDEssaiDatesInvalides[indice] ;
    assertFalse(date[0]+"/"+date[1]+"/"+date[2]+" est invalide"
    , LibDate.dateValide(date[0],date[1],date[2])); ② ③
  }
  bilanAssertions = bilanAssertions + tabJeuDEssaiDatesInvalides.length ;
}
```

① **given** : dans les situations suivantes

② **when** : quand on vérifie la validité de la date

③ **then** : on doit obtenir `false`

1.4. Tout est-il testable ?

- les librairies
- les intéractions systèmes (concurrence, etc.)
- les services réseau
- les interfaces graphiques (html, java, flash, etc.)
- ...
- PEUT-ETRE PAS, mais seulement après avoir essayé!

1.5. Toutes les manières de faire sont exploitables

- Ecrire le programme qui fasse passer les tests fournis
- Ecrire les tests d'un programme fourni
- Coder une fonctionnalité et ajouter le test correspondant
- Ajouter un test et intégrer la fonctionnalité correspondante (TDD)

- Retrouver l'équilibre du couple Programme / Programme de Test

2. Behavioural-Driven Development

2.1. Principes

- Étant donné : *Given*
- Quand : *When*
- Alors : *Then*
- Et : *And*

2.2. Exemple

```
Feature: Reviewing a Ph.D. Thesis
    Every PhD thesis review has some recurrent steps

Scenario: A reviewer, being an expert on the field, should be cited somewhere
    Given A PhD thesis to review
    And a reviewer John Smith
    Then The thesis should cite John Smith's work

Scenario: A Ph.D. thesis should be an original work
    Given A PhD thesis to review
    And a reviewer John Smith
    Then The PhD.pdf should be checked agains plagiarism
```

2.3. Conceils

Suivre le tutoriel:

```
https://cucumber.io/docs/guides/10-minute-tutorial/
```

2.4. Les erreurs les plus courantes

http://blog.takipi.com/the-top-10-exceptions-types-in-production-java-applications-based-on-1b-events/?utm_content=buffer0c58b&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer