Les Tests de logiciel

Mélina MAIZEROI & Jerry FRANCHET

17 octobre 2012

Sommaire :

Les Tests : Où, Quand, Pourquoi, Pour qui ?
Qu'est-ce qu'un test ?

Il doit permettre d'apporter une réponse aux questions :

- "Est-ce que le système effectue ce pour quoi il est fait ?" (**Validation**)
- "Est ce que le système ne comporte aucune défaillance ?" (**Vérification**)

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Pourquoi tester ?

Tester est important pour :

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Pourquoi tester ?

Tester est important pour :

- S'assurer qu'il remplit bien les spécifications définies

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Pourquoi tester ?

Tester est important pour :

- S'assurer qu'il remplit bien les spécifications définies
- Faciliter et accélérer le développement du projet

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Pourquoi tester ?

Tester est important pour :

- S'assurer qu'il remplit bien les spécifications définies
- Faciliter et accélérer le développement du projet
- Faire en sorte que le logiciel ne comporte pas de danger

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Le Standard IEEE 829

Une classification des anomalies par leur gravité :

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Le Standard IEEE 829

Une classification des anomalies par leur gravité :

niveau 1 : *Négligeable*

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Le Standard IEEE 829

Une classification des anomalies par leur gravité :

niveau 1 : *Négligeable*

niveau 2 : *Marginal*

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Le Standard IEEE 829

Une classification des anomalies par leur gravité :

niveau 1 : *Négligeable*

niveau 2 : *Marginal*

niveau 3 : *Critique*

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Le Standard IEEE 829

Une classification des anomalies par leur gravité :

niveau 1 : *Négligeable*

niveau 2 : *Marginal*

niveau 3 : *Critique*

niveau 4 : *Catastrophique*

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Exemples de catastrophes

Voici quelques exemples de catastrophes :

- Décollage raté d'Ariane 5 : plus de 400M $ perdus.
- Therac-25 : mort de nombreuses personnes.
- Mauvais guidage d'un missile : mort de soldats Américains.
- Logiciel du FBI abandonné : plus de 200M $ perdus.

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Quand tester ?

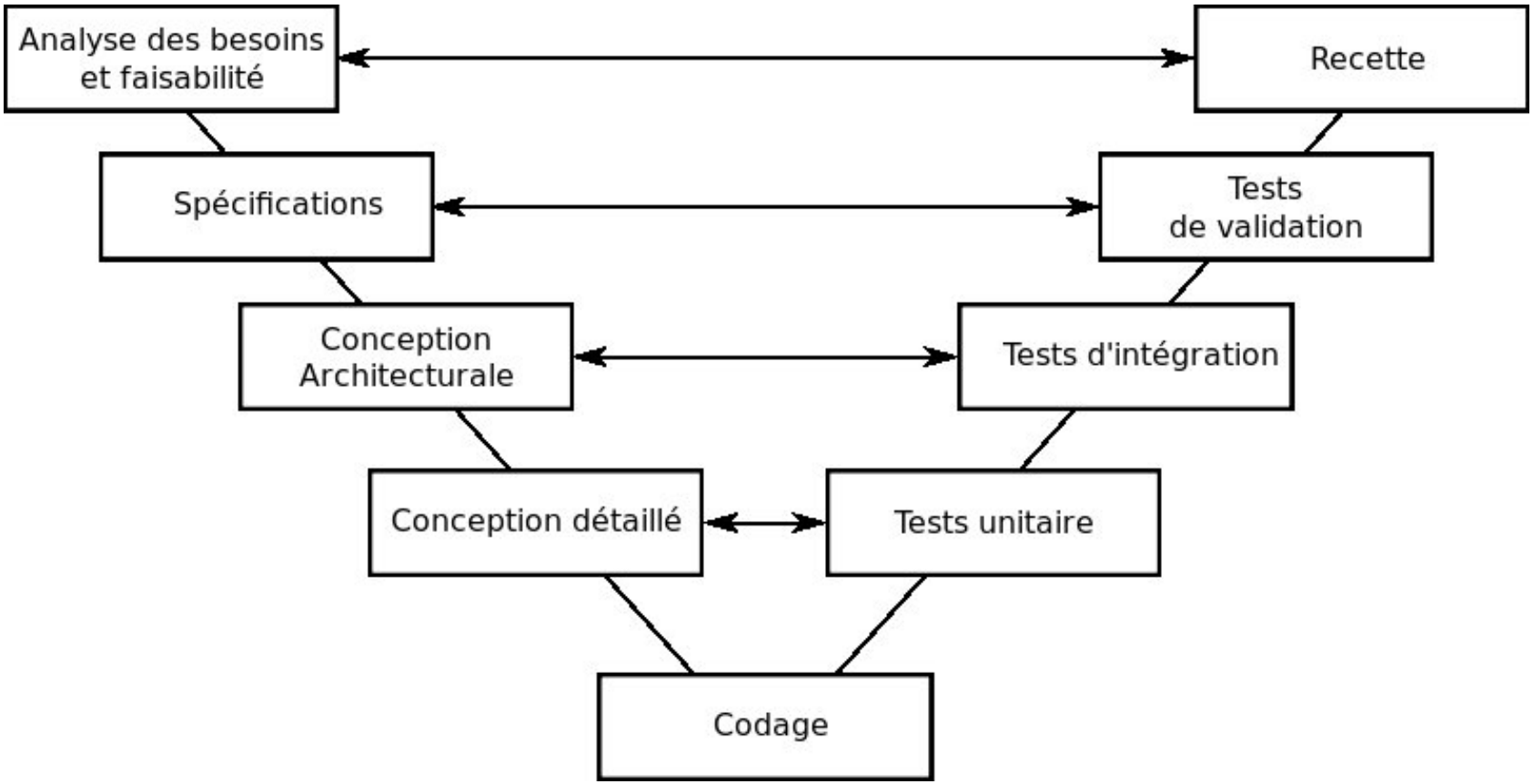


Figure: Cycle de développement

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Normes et organisations

Exemples d'organisations :

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Normes et organisations

Exemples d'organisations :

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Normes et organisations

Exemples d'organisations :

Les Tests : Où, Quand, Pourquoi, Pour qui ?

Normes et organisations

Exemples d'organisations :

IEEE

ISTQB International Software Testing Qualifications Board

Comité Français des Tests Logiciels CFTL Certification CERTIFIED

Les Différents Tests

Des Tests à différents niveaux

Exemples de tests :

Les Différents Tests

Des Tests à différents niveaux

Exemples de tests :

test unitaire Considération d'un seul élément minimal du système pour le test. Cet élément peut par exemple être une fonction ou une classe.

Les Différents Tests

Des Tests à différents niveaux

Exemples de tests :

test unitaire Considération d'un seul élément minimal du système pour le test. Cet élément peut par exemple être une fonction ou une classe.

test d'intégration Le test porte sur le fonctionnement coordonné des éléments du système, préalablement testés.

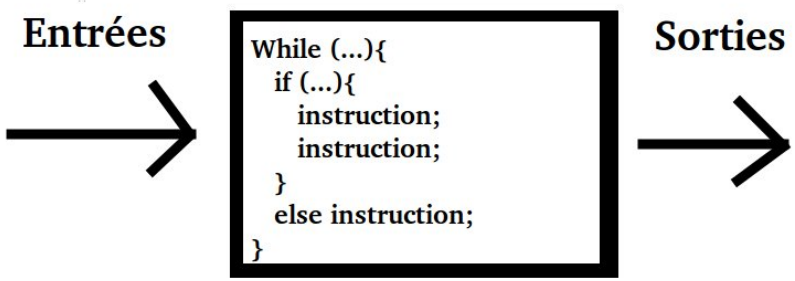
Les Différents Tests

Différentes visibilités de test

Les Différents Tests

Différentes visibilités de test

Boite Noire



Boite Blanche

Les Différents Tests

Plusieurs caractéristiques à tester

Des tests selon les caractéristiques :

Les Différents Tests

Plusieurs caractéristiques à tester

Des tests selon les caractéristiques :

test de vérification fonctionnelle Validation de l'élément.

Les Différents Tests

Plusieurs caractéristiques à tester

Des tests selon les caractéristiques :

test de vérification fonctionnelle Validation de l'élément.

test de non-régression Vérification des changements apportés.

Les Différents Tests

Plusieurs caractéristiques à tester

Des tests selon les caractéristiques :

test de vérification fonctionnelle Validation de l'élément.

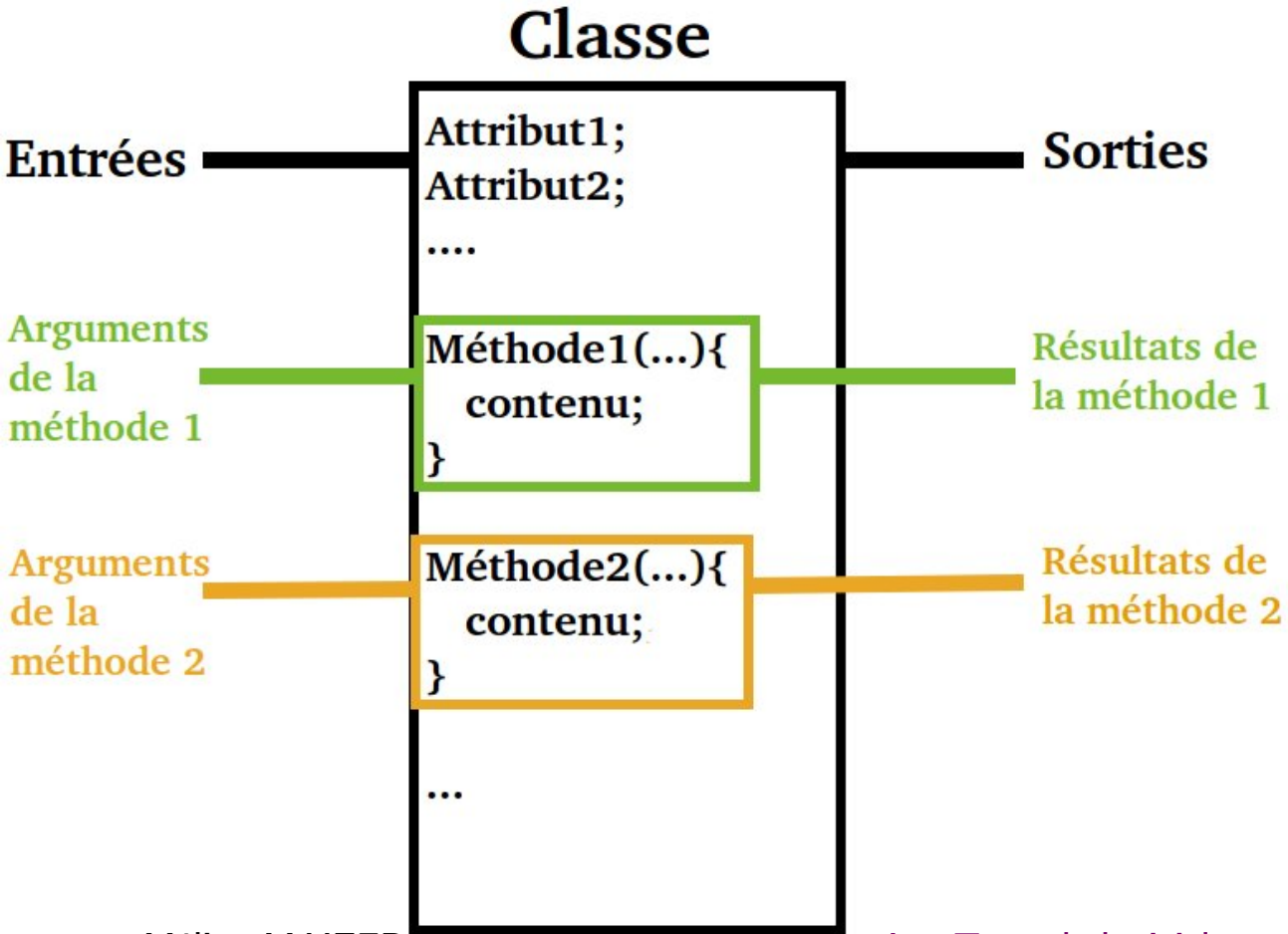
test de non-régression Vérification des changements apportés.

test de performances Test des différentes caractéristiques du système.

Les Différents Tests

Exemple 1

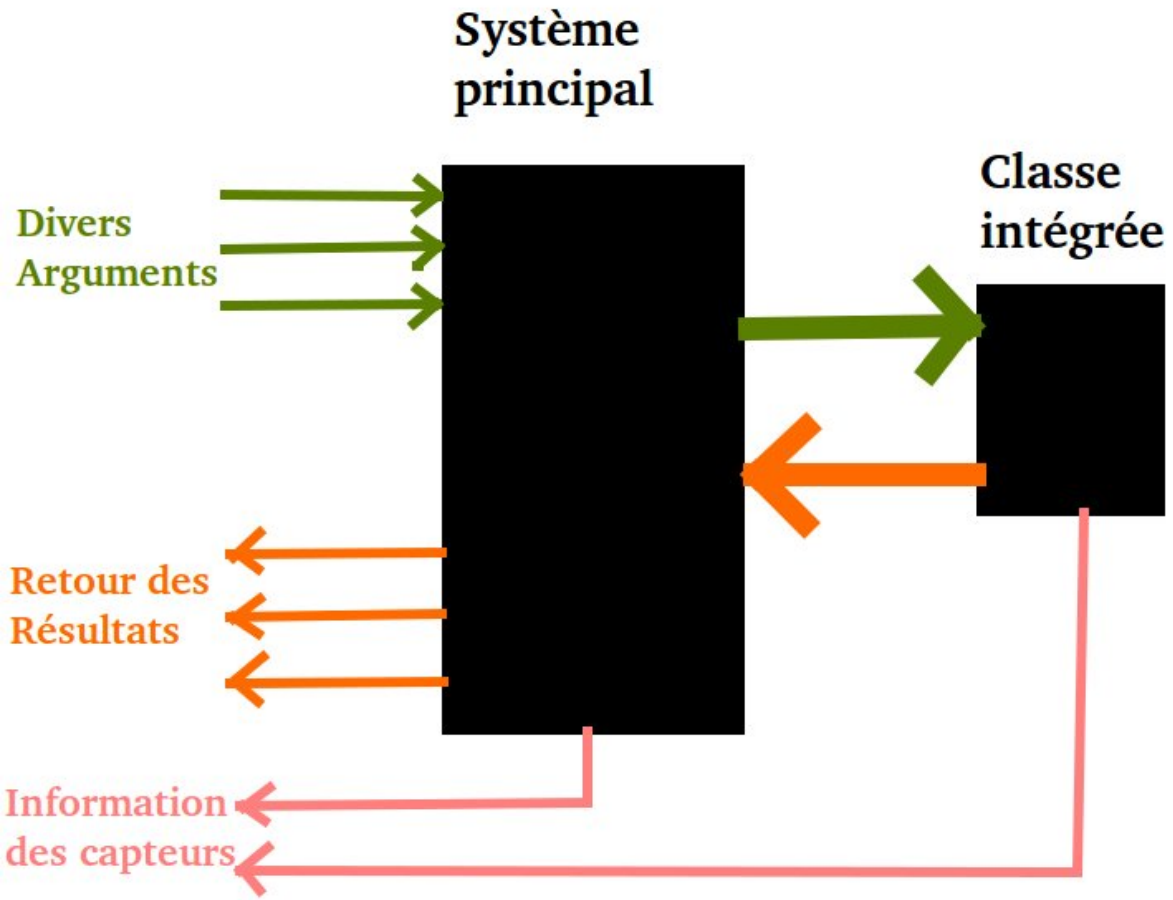
Un test Boite Blanche unitaire et fonctionnel



Les Différents Tests

Exemple 2

Un test Boite Noire d'intégration et de performances



Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.

Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.
Cinq étapes :

Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.
Cinq étapes :

1. écrire les tests unitaires

Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.
Cinq étapes :

1. écrire les tests unitaires
2. vérifier que les tests échouent

Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.
Cinq étapes :

1. écrire les tests unitaires
2. vérifier que les tests échouent
3. écrire le code de l'élément

Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.
Cinq étapes :

1. écrire les tests unitaires
2. vérifier que les tests échouent
3. écrire le code de l'élément
4. vérifier que les tests passent

Les Différents Tests

Test Driven Development

Développement d'un projet dirigé par les tests.
Cinq étapes :

1. écrire les tests unitaires
2. vérifier que les tests échouent
3. écrire le code de l'élément
4. vérifier que les tests passent
5. améliorer l'élément

Des Méthodes pour Tester

Les mauvaises méthodes

Ce qu'il ne faut pas faire :

Des Méthodes pour Tester

Les mauvaises méthodes

Ce qu'il ne faut pas faire :

- La méthode de test naturelle :

Des Méthodes pour Tester

Les mauvaises méthodes

Ce qu'il ne faut pas faire :

- La méthode de test naturelle :
  - exécuter pour repérer les erreurs.

Des Méthodes pour Tester

Les mauvaises méthodes

Ce qu'il ne faut pas faire :

- La méthode de test naturelle :
  - exécuter pour repérer les erreurs.
  - lire le code pour chercher la provenance du problème.

Des Méthodes pour Tester

Les mauvaises méthodes

Ce qu'il ne faut pas faire :

- La méthode de test naturelle :
  - exécuter pour repérer les erreurs.
  - lire le code pour chercher la provenance du problème.
  - placer des affichages de texte et des conditions en plus et constater les résultats en exécutant.

Des Méthodes pour Tester

Les mauvaises méthodes

Ce qu'il ne faut pas faire :

- La méthode de test naturelle :
  - exécuter pour repérer les erreurs.
  - lire le code pour chercher la provenance du problème.
  - placer des affichages de texte et des conditions en plus et constater les résultats en exécutant.
- Implémenter les tests à la fin du projet.

Des Méthodes pour Tester

Les méthodes de test Boite Noire

Trois exemples de méthodes :

Des Méthodes pour Tester

Les méthodes de test Boite Noire

Trois exemples de méthodes :

- Test aux limites

Des Méthodes pour Tester

Les méthodes de test Boite Noire

Trois exemples de méthodes :

- Test aux limites
- Graphe causes-effet

Des Méthodes pour Tester

Les méthodes de test Boite Noire

Trois exemples de méthodes :

- Test aux limites
- Graphe causes-effet
- Test statistique

Des Méthodes pour Tester

Exemples :

Soit une fonction créant un Rectangle d'après deux paramètres entrés par l'utilisateur : longueur et largeur.

Des Méthodes pour Tester

Exemples :

Soit une fonction créant un Rectangle d'après deux paramètres entrés par l'utilisateur : longueur et largeur.

Test aux limites

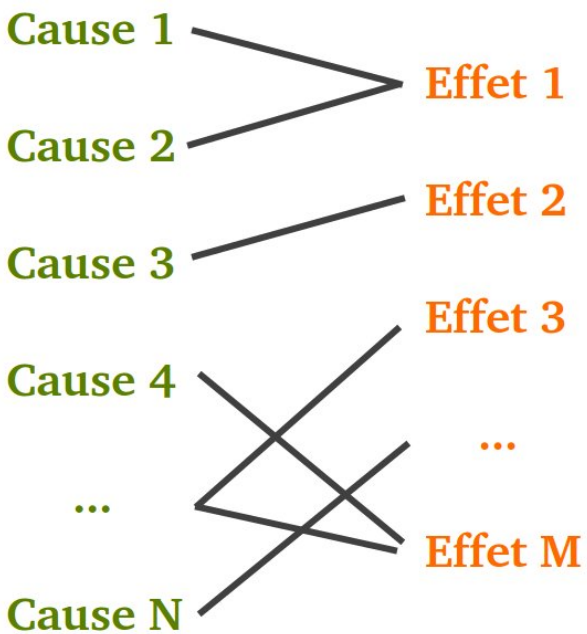
Paramètres testés :

(2,3)	longueur et largeur quelconque
(2,2)	longueur égal largeur
(0,0)	valeurs nulles
(2,3,4)	trop de paramètres
(1000000,1000000)	très grandes valeurs
(0.00001,0.00003)	très petites valeurs
(-5,-4)	valeurs négatives
(0.00001,1000000)	valeurs extrêmes

Des Méthodes pour Tester

Exemples :

Graphe causes-effet



ENTREES	largeur != longueur	largeur == longueur	longueur < 0
SORTIES	Rectangle quelconque	Carré	Erreur !

Des Méthodes pour Tester

Exemples :

Test statistique

SORTIES	Rectangle quelconque	Carré	Erreur !
POURCENTAGES	49%	1%	50%

Des Méthodes pour Tester

Les méthodes de test Boite Blanche

On distingue deux types de test Boite Blanche :

Des Méthodes pour Tester

Les méthodes de test Boite Blanche

On distingue deux types de test Boite Blanche :

1. Test structurel *Statique* : sans exécution du code.

Des Méthodes pour Tester

Les méthodes de test Boite Blanche

On distingue deux types de test Boite Blanche :

1. Test structurel *Statique* : sans exécution du code.
2. Test structurel *Dynamique* : avec exécution du code.

Des Méthodes pour Tester

Les méthodes de test Boite Blanche

Test Structurel Statique

- Relecture du code.
  - preuve de fonctionnement.
  - preuve de validité.
- Estimation de la complexité du code.

Des Méthodes pour Tester

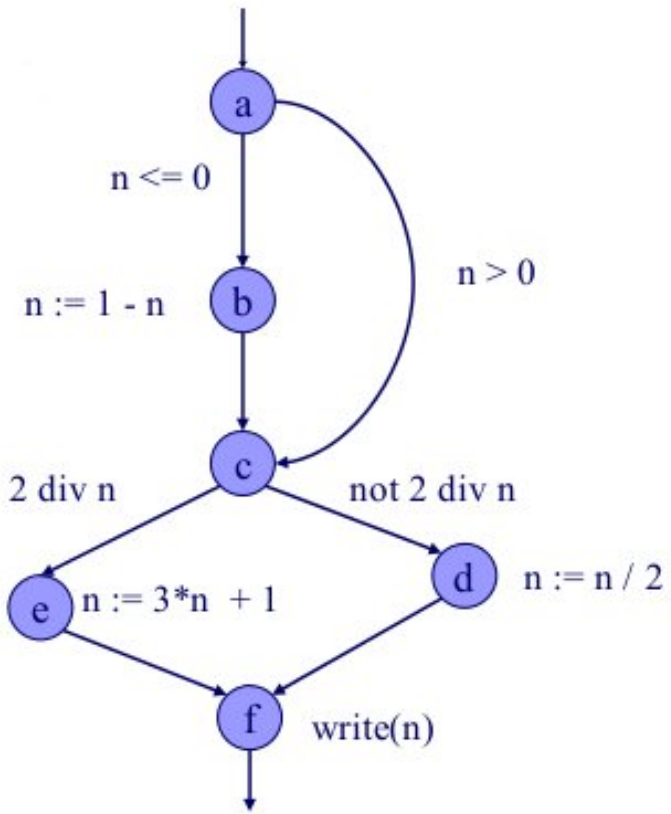
Les méthodes de test Boite Blanche

Test Dynamique Flot de contrôle

```
if (n≤0){
   n = 1 - n;
}

if (n%2==0){
   n = 3*n + 1;
}
else{
   n = n / 2;
}

printf("%d",n);
```



Des Méthodes pour Tester

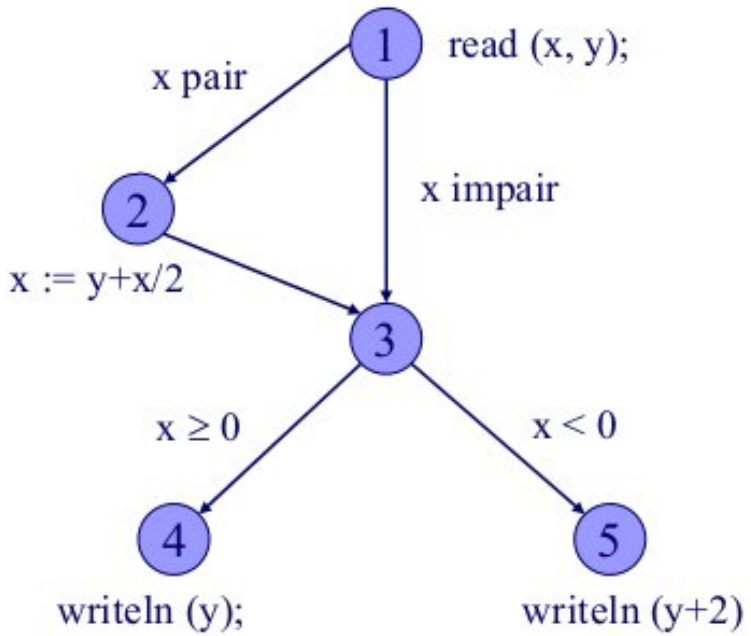
Les méthodes de test Boite Blanche

Test Dynamique Flot de données

```
int x, y;
scanf ("%d %d", &x, &y);

if (x%2==0){
  x = y + x/2;
}

if(x≥0){
  printf ("%d", y);
}
else{
  printf("%d", y + 2);
}
```



Des Méthodes pour Tester

Les méthodes de test Boite Blanche

Test Mutationnel

```
int main(){
  int a,b;
  if(a<3){
   b=a*2;
  }
}
```

Des Méthodes pour Tester

Les méthodes de test Boite Blanche

Test Mutationnel

```
int main(){
  int a,b;
  if(a<3){
   b=a*2;
  }
}
```

```
int main(){
  int a,b;
  if(a>3){
   b=a*2;
  }
}
```

Des Méthodes pour Tester

Les méthodes de test Boite Blanche

Test Mutationnel

```
int main(){
  int a,b;
  if(a<3){
   b=a*2;
  }
}
```

```
int main(){
  int a,b;
  if(a>3){
   b=a*2;
  }
}
```

```
int main(){
  int a,b;
  if(a<3){
   b=a;
  }
}
```

Les Tests en pratique

Exemple d'implémentation

```
int multiplication(int a, int b){
  return a*b;
}
int testMultiplication(){
  int retour = 0;

  int a = 1, b = 1; // Arranger
  int r = multiplication(a,b); // Agir
  if(r!=1){printf("Echec test 1"); // Auditer
    retour=-1;}

  a = 0; b = 1; // Arranger
  r = multiplication(a,b); // Agir
  if(r!=0){printf("Echec test 2"); // Auditer
    retour=-1;}

  a = -3; b = 5; // Arranger
  r = multiplication(a,b); // Agir
  if(r!=-15){printf("Echec test 3"); // Auditer
    retour=-1;}

  return retour;
}
int main(){
  int etat = testMultiplication();
}
```

Les Tests en pratique
Les tests facilités par les IDE

- Détection des erreurs de syntaxe par le Débogueur lors de la compilation

Les Tests en pratique
Les tests facilités par les IDE

- Détection des erreurs de syntaxe par le Débogueur lors de la compilation
- Parcours étape par étape de l'exécution de l'application

Les Tests en pratique
Les tests facilités par les IDE

- Détection des erreurs de syntaxe par le Débogueur lors de la compilation
- Parcours étape par étape de l'exécution de l'application
- Vérification syntaxique et référentielle lors de l'écriture du code.

Les Tests en pratique
Les tests facilités par les IDE

- Détection des erreurs de syntaxe par le Débogueur lors de la compilation
- Parcours étape par étape de l'exécution de l'application
- Vérification syntaxique et référentielle lors de l'écriture du code.
- Exemples d'avertissements :

Les Tests en pratique
Les tests facilités par les IDE

- Détection des erreurs de syntaxe par le Débogueur lors de la compilation
- Parcours étape par étape de l'exécution de l'application
- Vérification syntaxique et référentielle lors de l'écriture du code.
- Exemples d'avertissements :
  - attributs non utilisés ou portion de code non-atteinte

Les Tests en pratique
Les tests facilités par les IDE

- Détection des erreurs de syntaxe par le Débogueur lors de la compilation
- Parcours étape par étape de l'exécution de l'application
- Vérification syntaxique et référentielle lors de l'écriture du code.
- Exemples d'avertissements :
  - attributs non utilisés ou portion de code non-atteinte
  - utilisation d'une classe non-correspondante a la classe créée ou inexistante

Les Tests en pratique
Les tests dans les langages à objet

Le langage objet intègre des fonctionnalités facilitant les tests :

Les Tests en pratique
Les tests dans les langages à objet

Le langage objet intègre des fonctionnalités facilitant les tests :

code partitionné en classes : facilite les tests unitaires.

Les Tests en pratique
Les tests dans les langages à objet

Le langage objet intègre des fonctionnalités facilitant les tests :

code partitionné en classes : facilite les tests unitaires.

notion d'héritage : facilite les tests d'intégration.

Les Tests en pratique

Les tests dans les langages à objet

Le langage objet intègre des fonctionnalités facilitant les tests :

code partitionné en classes : facilite les tests unitaires.

notion d'héritage : facilite les tests d'intégration.

assertions : déjà des vérifications en elles-même.

Les Tests en pratique
Les tests dans les langages à objet

Le langage objet intègre des fonctionnalités facilitant les tests :

code partitionné en classes : facilite les tests unitaires.

notion d'héritage : facilite les tests d'intégration.

assertions : déjà des vérifications en elles-même.

exceptions : permettent de récupérer des informations sur des tests effectués.

Les Tests en pratique
Les tests dans les langages à objet

Le langage objet intègre des fonctionnalités facilitant les tests :

code partitionné en classes : facilite les tests unitaires.

notion d'héritage : facilite les tests d'intégration.

assertions : déjà des vérifications en elles-même.

exceptions : permettent de récupérer des informations sur des tests effectués.

introspections : permettent d'automatiser les tests sur des classes génériques.

Les Tests en pratique
L'automatisation des tests

Elle permet de :

- Réduire le temps requis pour tester
- Améliorer la productivité des testeurs
- Améliorer la couverture des tests de régression
- Améliorer la qualité des tests
- Faciliter la maintenance et l'exécution d'un ensemble de test
- Améliorer la répétabilité ou la réutilisabilité des tests
- Améliorer la qualité de l'application
- Obtenir un enregistrement organisé et détaillé des tests exécutés
- Simplifier le déverminage.

Les Tests en pratique
Les XUnits

Outils permettant de réaliser des test unitaires dans un language donné.
Quelques exemples :

- SUnit pour Smalltalk
- CUnit pour C
- cppunit pour C++
- JUnit pour Java

Les Tests en pratique
Implémentation de CUnit

```
int max (int n1, int n2 ){
   if ( n2 > n1 )  return n2;
   return n1;
}
void test_case_sample(void){
   CU_ASSERT(CU_TRUE);
   CU_ASSERT_NOT_EQUAL(2, -1);
   CU_ASSERT_STRING_EQUAL("string #1", "string #1");
   CU_ASSERT_STRING_NOT_EQUAL("string #1", "string #2");

   CU_ASSERT(CU_FALSE);
   CU_ASSERT_EQUAL(2, 3);
   CU_ASSERT_STRING_NOT_EQUAL("string #1", "string #1");
   CU_ASSERT_STRING_EQUAL("string #1", "string #2");
}
void max_test_1(void) {
  CU_ASSERT_EQUAL( max(1,2), 2);
  CU_ASSERT_EQUAL( max(2,1), 2);
}
void max_test_2(void) {
  CU_ASSERT_EQUAL( max(2,2), 2);
  CU_ASSERT_EQUAL( max(0,0), 0);
  CU_ASSERT_EQUAL( max(-1,-1), -1);
}
void max_test_3(void) {
  CU_ASSERT_EQUAL( max(-1,-2), -1);}
```

Les Tests en pratique
Implémentation de CUnit

```
int main (void){
   CU_pSuite pSuite = NULL; //Définition de la suite de test.

   /* initialisation du registre */
   pSuite = CU_add_suite( "max_test_suite", init_suite, clean_suite); //Suite initialisée.
   CU_add_test(pSuite, "max_test_1", max_test_1); //
   CU_add_test(pSuite, "max_test_2", max_test_2); // Ajout des tests dans la suite.
   CU_add_test(pSuite, "max_test_3", max_test_3); //

   CU_basic_set_mode(CU_BRM_VERBOSE);              // Affichage pendant l'exécution des tests.
   CU_basic_run_tests(); printf("\n");             // Lancement de l'exécution des tests.
   CU_basic_show_failures(CU_get_failure_list()); // Affichage des erreurs détectées.
   printf("\n\n");

   /* Nettoyage du registre */
   return CU_get_error();
}

Suite: max_test_suite
  Test: max_test_1 ...passed
  Test: max_test_2 ...passed
  Test: max_test_3 ...passed
Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      3      3      3      0        0
             asserts      6      6      6      0      n/a
```

Conclusion et Bibliographie

Conclusion

- Des tests couteux mais nécessaires

Conclusion et Bibliographie

Conclusion

- Des tests couteux mais nécessaires
- Des métiers pour la confiance

Conclusion et Bibliographie

Conclusion

- Des tests couteux mais nécessaires
- Des métiers pour la confiance
- Les tests de plus en plus informatisés

Conclusion et Bibliographie

Bibliographie

- Des livres intéressants :
  * Karapoulios, Le test des logiciels, Hermès - Lavoisier, 1999
  * John Watkins,Test logiciel en pratique,Vuibert,2002
  * Bernard Homès,Les tests logiciels,Hermès - Lavoisier,2011
- Des sites à consulter :
  - `http://fr.wikibooks.org/`
  - `http://istqb.org/`
  - `http://www.cftl.fr/`
  - `http://wpollock.com/`
  - `http://sebastien.bardin.free.fr/`