



## Introduction au testing

Sandrine-Dominique Gouraud  
[sandrine.gouraud@mythalesgroup.io](mailto:sandrine.gouraud@mythalesgroup.io)



# Qui suis-je?

- Une étudiante en recherche de quête
- Une enseignante-chercheure en informatique, spécialité test de logiciels à l'université
- Une ingénieure qualité logiciel dans une PME
- Une testeuse dans une grande société
- Une *Software Test Leader* en prestation chez un grand compte

# Pourquoi il faut tester?

- Parce ce que tout être humain peut faire une erreur (méprise) qui produit un défaut (bogue) dans le code, dans un logiciel ou un système, ou dans un document
- Et si un défaut du code est exécuté, il peut générer une défaillance (ou pas)

# Erreur vs défaut vs défaillance

Erreur / méprise	Défaut / bogue	Défaillance
<p>Action humaine provoquant l'introduction d'un défaut dans le logiciel:</p> <ul style="list-style-type: none"><li>• Oublis dans des phases de conception</li><li>• Méprises sur la compréhension ou l'interprétation des spécifications</li><li>• Erreurs de réalisation lors du codage (ex: choix de structures de données ou d'algorithmes inadaptés)</li><li>• Non respect des standards de codage</li></ul>	<p>Imperfection présente dans le logiciel (incluant sa documentation):</p> <ul style="list-style-type: none"><li>• Fonctionnalités oubliées (présentes dans les spécifications)</li><li>• Défauts de programmation</li><li>• Défauts de portabilité</li></ul>	<p>Résultat de l'exécution d'un programme contenant un défaut: déviation observée du comportement d'un système par rapport à un comportement requis.</p> <ul style="list-style-type: none"><li>• Calculs erronés</li><li>• Fenêtre mal placée</li><li>• Service non rendu (ex: données non stockées dans une base, message non affiché, fenêtre non ouverte)</li><li>• ...</li></ul>

# Pourquoi il faut tester ?

1. Pour limiter le coût des bogues
  - Baisse de réputation
  - Économique (Perte de contrat, pénalités)
  - Humains, environnementaux, etc.
2. Pour assurer la qualité d'une application
  - Pour atteindre les objectifs de normes/lois
  - Pour atteindre un bon rapport qualité/prix
  - Pour maintenir la confiance du client

# « PETIT » FOCUS SUR LES BOGUES

17-  
9/9

0800 Action started  
1000 = stopped - action ✓ { 1.2700 9.000 000 025  
1200 020 MP-AC 2.13047045 (start) 9.415 925019(-1)  
020 P20 = 2.13047045  
conv 2.13067045

Relays 6-2 - 020 failed speed speed test  
in relay - 11.00 start -

Relays changed

1100 Started Cosine Tape (Sine check)  
1525 Started Multi's Aider Test

1545  Relay → Panel F  
(note) in relay.

1645 First actual case of bug being found.  
1700 changed start.  
1700 closed down.

# Bogue de l'an 2000: problème de conception

- Problème identifié dès les années 70: calcul des dates uniquement sur les deux derniers chiffres de l'année
- Coût estimé: >600 millions de dollars en travaux de contrôle et maintenance préventive
- Résultat: aucun problème critique à signaler

# Bogue de l'an 2000





# Et le bogue de l'an 2038?

- Problème identifié: le 19 janvier 2038 à 3h14m7s (temps universel), tout système/programme qui utilise des dates représentées en 32 bits passera à une date négative qui dans le meilleur des cas affichera 13 décembre 1901
- Qui est concerné? les ordinateurs, les formats de fichier, les systèmes de fichiers, les systèmes d'exploitation voire les horloges matérielles qui utilisent des dates sur 32 bits
- Coût estimé: ?
- Résultat: ?

# Bogue de l'an 2038

01111111 11111111 11111111 11

2147483632

2038-01-19 03:13:52 (UTC)

2038-01-19 03:13:52 (UTC)

# Bogue de l'an 2038

- Système en service depuis 1989 qui réalise des analyses médicales en utilisant des barrettes.
- Code des barrettes sur 3 caractères
  - [0-9A-Z]{3} qui représentent la date d'expiration
  - 0 pour 1989, A pour 1999 et Z pour 2024
- Correction retenue:
  - Q pour 2015, 0 pour 2024 et P pour 2050
- Gestion du bogue: fin de maintenance en 2035

110C 110

Un permanent  
du centre d'écoute  
Schindler vous parlera  
anglais. Soyez calme et  
patient en attendant  
l'intervention  
du technicien.

NO WEIGHT LIMIT 80001  
525 kg 7 PERS.



# Pourquoi il faut tester?

- 26 mars 2016: Software Update Destroys \$286 Million Japanese Satellite
  - Perte: 3 ans d'observation et certainement 10 ans de recherche
  - <http://hackaday.com/2016/05/02/software-update-destroys-286-million-japanese-satellite/>
- F-35: un avion à \$320 – \$400 billion
  - Premier vol: 2006, livraison pas avant 2021
  - <http://www.extremetech.com/extreme/222380-the-pentagons-official-f-35-bug-list-is-terrifying>

# Pourquoi il faut tester?

- The Tricentis Software Fail Report
- <https://www.tricentis.com/>

	2015	2016	2017
<b>Anomalies enregistrées</b>	483	548	606
<b>Personnes touchées</b>	4 milliards	4,4 milliards	3,7 milliards
<b>Actifs touchés en \$</b>	428 milliards	1,1 trillions	1,7 trillions
Sociétés touchées	239	363	314
Temps perdu cumulé	n.c.	365 années	268 années

# The Tricentis Software Fail Report 2017

- <file:///localhost/Users/sandrine/Desktop/sdg/ExtraitTricentis2017.pdf>



**° AU FAIT, C'EST QUOI LE TEST?**



# Définition(s) du test

- *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*
  - IEEE (Standard Glossary of Software Engineering Terminology)
- *Tester, c'est exécuter le programme dans l'intention d'y trouver des anomalies ou des défauts*
  - G. Myers (The Art of Software testing)

# Définition(s) du test

- *Testing can reveal the presence of errors*  
***but never their absence***
  - Edgar W. Dijkstra. *Notes on structured programming*. Academic Press, 1972.

# Objectifs du test

- Trouver des défauts
- Augmenter le niveau de confiance en la qualité d'un logiciel
- Fournir aux décideurs *go/no go* un état le plus précis possible de la qualité du logiciel
- Prévenir des défauts

# Tester ≠ Déboguer

- Tester et déboguer, ce n'est pas la même activité et les responsables sont différents:
  - Les testeurs testent
  - Les développeurs déboguent
- Déboguer: activité de développement permettant d'analyser, trouver et supprimer les causes de la défaillances

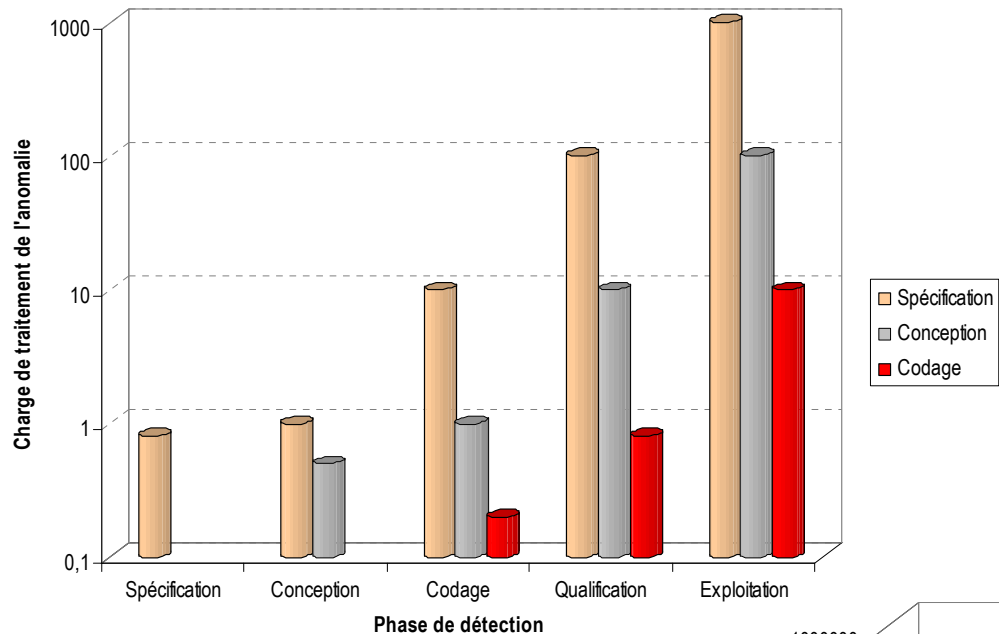


# **LES 7 PRINCIPES GÉNÉRAUX DU TEST**

# Les 7 principes généraux

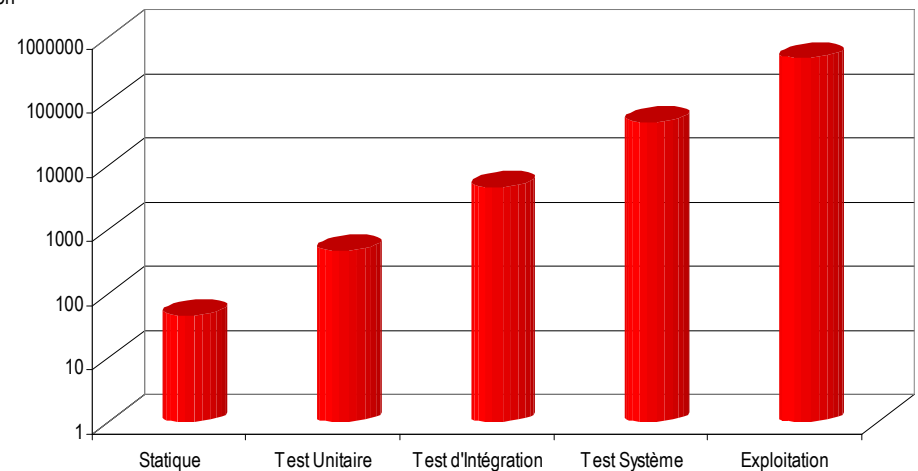
1. Les tests montrent la présence de défauts
  - **Jamais leur absence**
2. Les tests exhaustifs sont impossibles
  - Exemple Sopra: test d'une IHM contenant 20 écrans, 4 menus, 3 options, 10 champs, 2 types de données, 100 valeurs possibles
  - $20 * 4 * 3 * 10 * 2 * 100 = 480\ 000$  cas de tests
  - Un expert (10s par test!) mettrait 180 jours
3. Il faut tester le plus tôt possible

# Constat



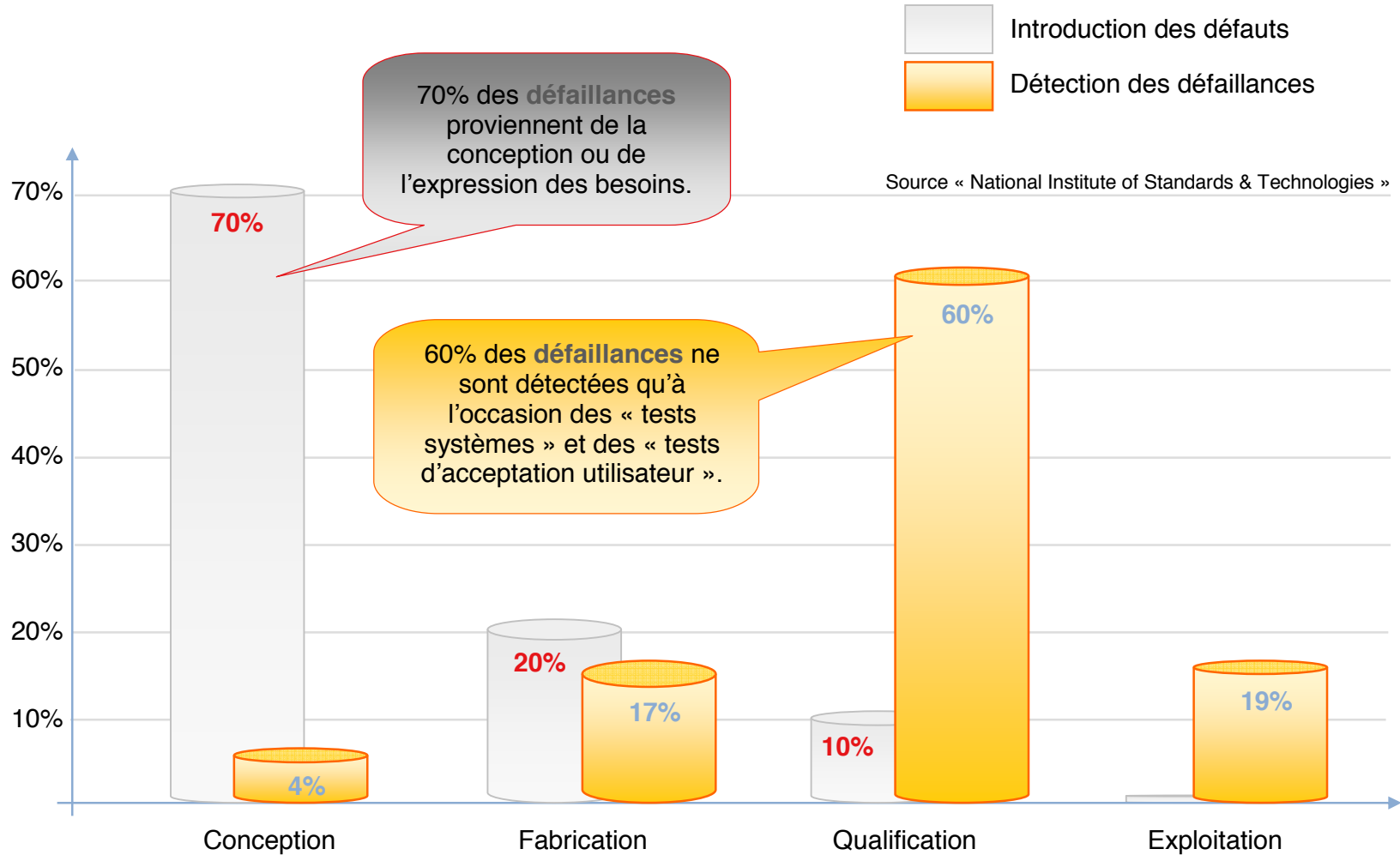
**Evolution de la charge du traitement (correction) d'une anomalie détectée pendant une activité**

**Evolution du coût (en €) du diagnostic et de la correction d'une anomalie par rapport aux niveaux de test (source : UK Orange services)**



# Constat

## Décalage entre injection des défauts et détection des défaillances





# Les 7 principes généraux

4. Les défauts sont généralement regroupés
  - Loi de Pareto: 80% des défauts sont concentrés dans 20% des modules
5. Paradoxe du pesticide
  - Les mêmes tests ne décèlent plus de défauts
6. Les tests dépendent du contexte
  - Les tests de sécurité dépendent de l'application testée

# Les 7 principes généraux

## 7. L'illusion de l'absence d'erreur

- Trouver et corriger des défauts ne sert à rien si le système conçu est inutilisable (opérativité, conformité, attractivité, performance)

# Le test le plus difficile à passer

lundi 13 janvier 2014 by CommitStrip



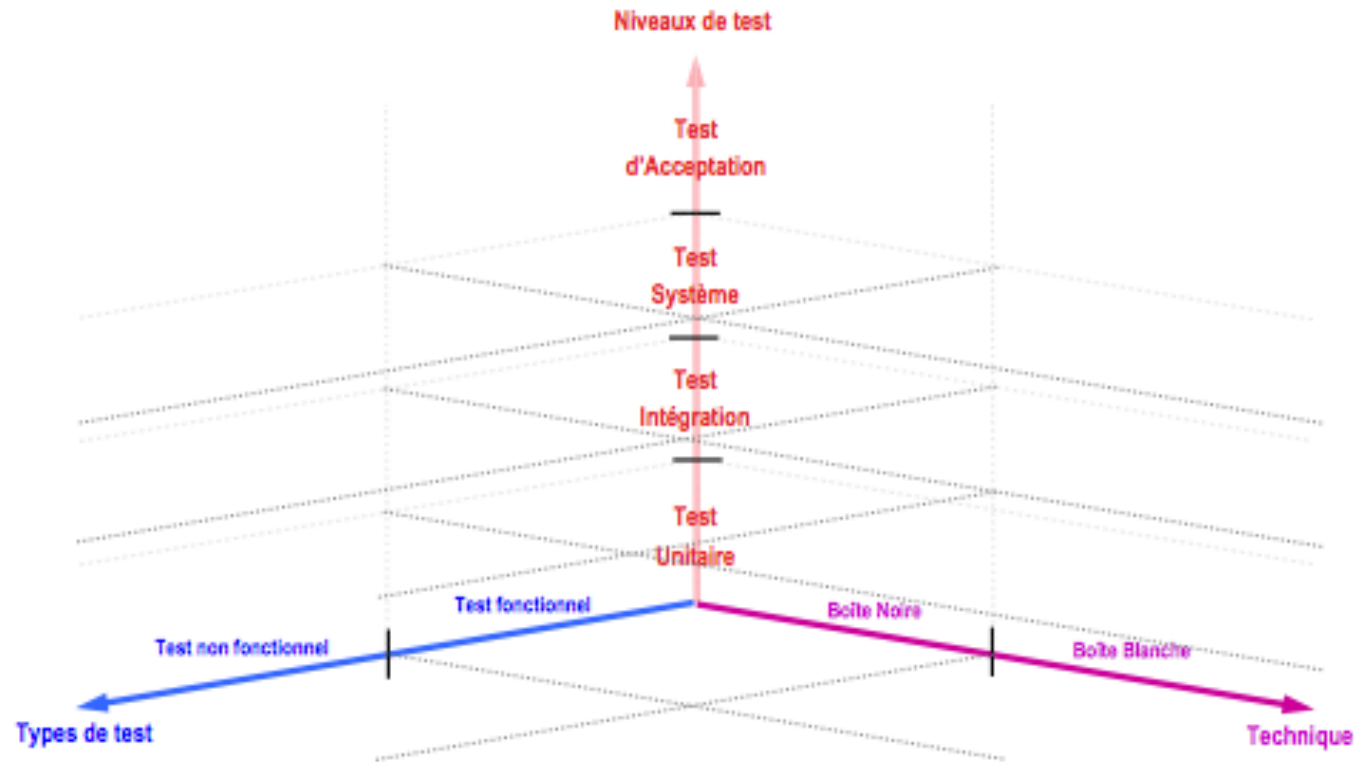


**TESTER: QUOI, QUAND,  
COMMENT**

# Typologie des tests

## Typologie

Vue d'ensemble



° **TESTER: *QUOI*, QUAND,  
COMMENT**

# Les différents types de test

- **Test des caractéristiques fonctionnelles:**
  - *Fonction spécifiée que doit remplir un système*
  - pertinence, exactitude, interopérabilité, sécurité, conformité
- **Test des caractéristiques non-fonctionnelles:**
  - *Attribut du comportement dont le système doit faire preuve pour remplir sa fonction*
  - fiabilité, facilité d'utilisation, rendement/efficacité, maintenabilité, portabilité

# Les différents types de test

- **Test de la structure:** architecture logicielle
- **Test lié au changement:** test de confirmation, test de régression
- **Test de maintenance:** modification, migration, suppression

Regression:  
"when you fix one bug, you  
introduce several newer bugs."





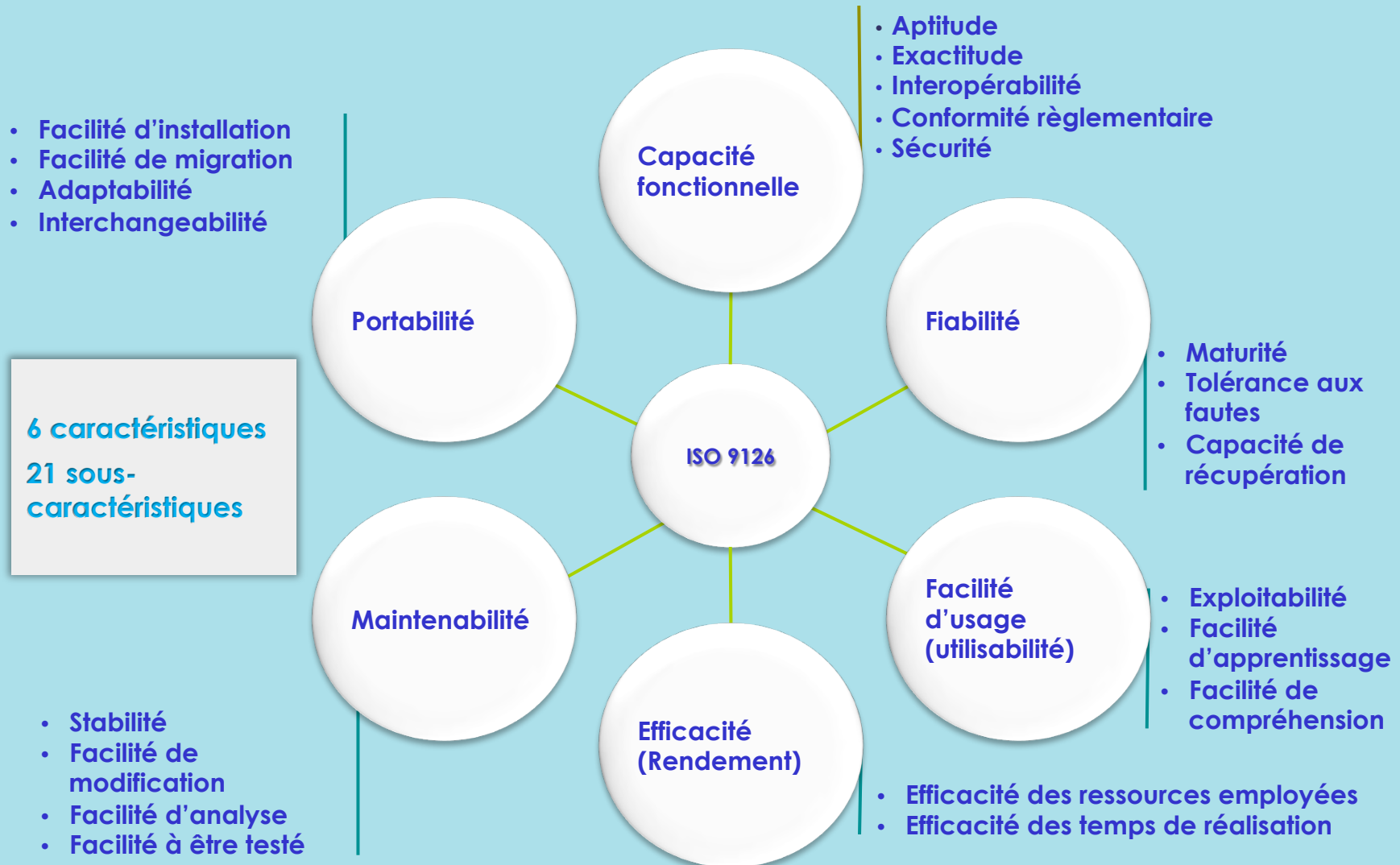
# ISO 9126: qualité d'un logiciel

- Capacité fonctionnelle:
  - Le logiciel répond-il aux besoins fonctionnels exprimés?
- Fiabilité:
  - Le logiciel maintient-il son niveau de service dans des conditions précises et pendant une période déterminée?
- Facilité d'utilisation:
  - Le logiciel requiert-il peu d'effort à l'utilisation?

# ISO 9126: qualité d'un logiciel

- Rendement/efficacité:
  - Le logiciel requiert-il un dimensionnement rentable et proportionné de la plate-forme d'hébergement en regard des autres exigences?
- Maintenabilité:
  - Le logiciel requiert-il peu d'effort à son évolution par rapport aux nouveaux besoins?
- Portabilité:
  - Le logiciel peut-il être transférer d'une plate-forme d'environnement à une autre?

# Norme ISO 9126: définition de la qualité d'un logiciel





◦ **TESTER: QUOI, QUAND,  
COMMENT**

# Les différents niveaux de test

- **Tests unitaires ou de composant** : tester chaque élément de manière isolé
  - Élément: méthode, classe, composant, etc.
  - Acteurs: les développeurs
  - Le code est accessible
  - Aucun rapport d'incident
  - Vocabulaire: bouchon/simulateur/pilote, test statique/dynamique

# Les différents niveaux de test

- **Test d'intégration:** tester le bon comportement/l'interaction des éléments suite à une composition d'éléments
  - Acteurs: les développeurs ou équipe dédiée
  - Le code est accessible ou pas
  - Vocabulaire: big-bang, bottom-up, top-down, ihm, simulateur

# Les différents niveaux de test

- **Test système ou de conformité:** assurer que l'application présente les fonctionnalités attendues
  - Acteurs: l'équipe dédiée
  - Le code n'est pas accessible
  - Vocabulaire: qualification, performance

# Les différents niveaux de test

- **Test de validation ou d'usine ou d'opération:** valider l'adéquation avec les besoins du client
  - Acteurs: l'équipe dédiée, le client
  - Le code n'est pas accessible
  - Vocabulaire: Alpha tests, Beta tests





**TESTER: QUOI, QUAND,  
*COMMENT***

# Les différentes techniques de test

*Comment sélectionner les tests?*

- **Test fonctionnel ou boîte noire:** la sélection se fait en se basant sur la spécification
  - On teste ce que doit faire l'application
- **Test structurel ou boîte blanche/transparente:** la sélection se fait en se basant sur le code
  - On teste la manière dont l'application le fait
- **Test probabiliste:** la sélection se base sur le domaine d'entrée en fonction d'arguments probabilistes

# Les différentes techniques de test

*Comment sélectionner les tests?*

- Ces différentes techniques sont utilisées de manières complémentaires car elles ne remontent pas les mêmes bogues.
- Ils existent même des approches qui les combinent pour compenser les défauts des unes par les qualités de l'autre.



# **PLACE DU TEST DANS LE CYCLE DE LOGICIEL**

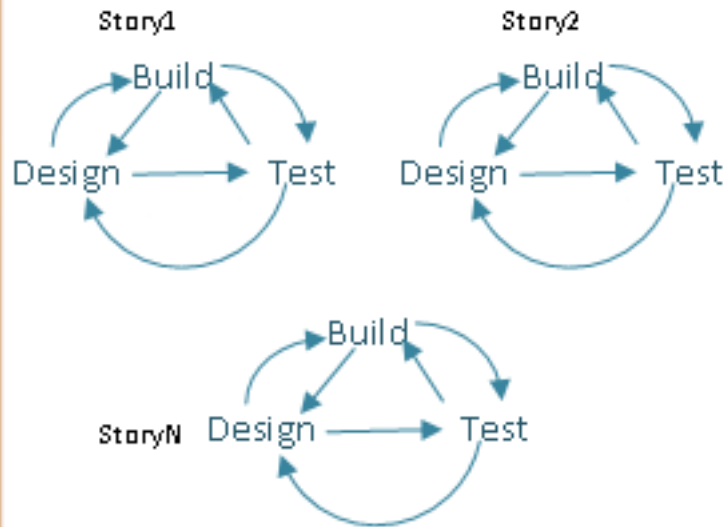
# Positionnement du test dans le cycle de vie du logiciel

- Source:

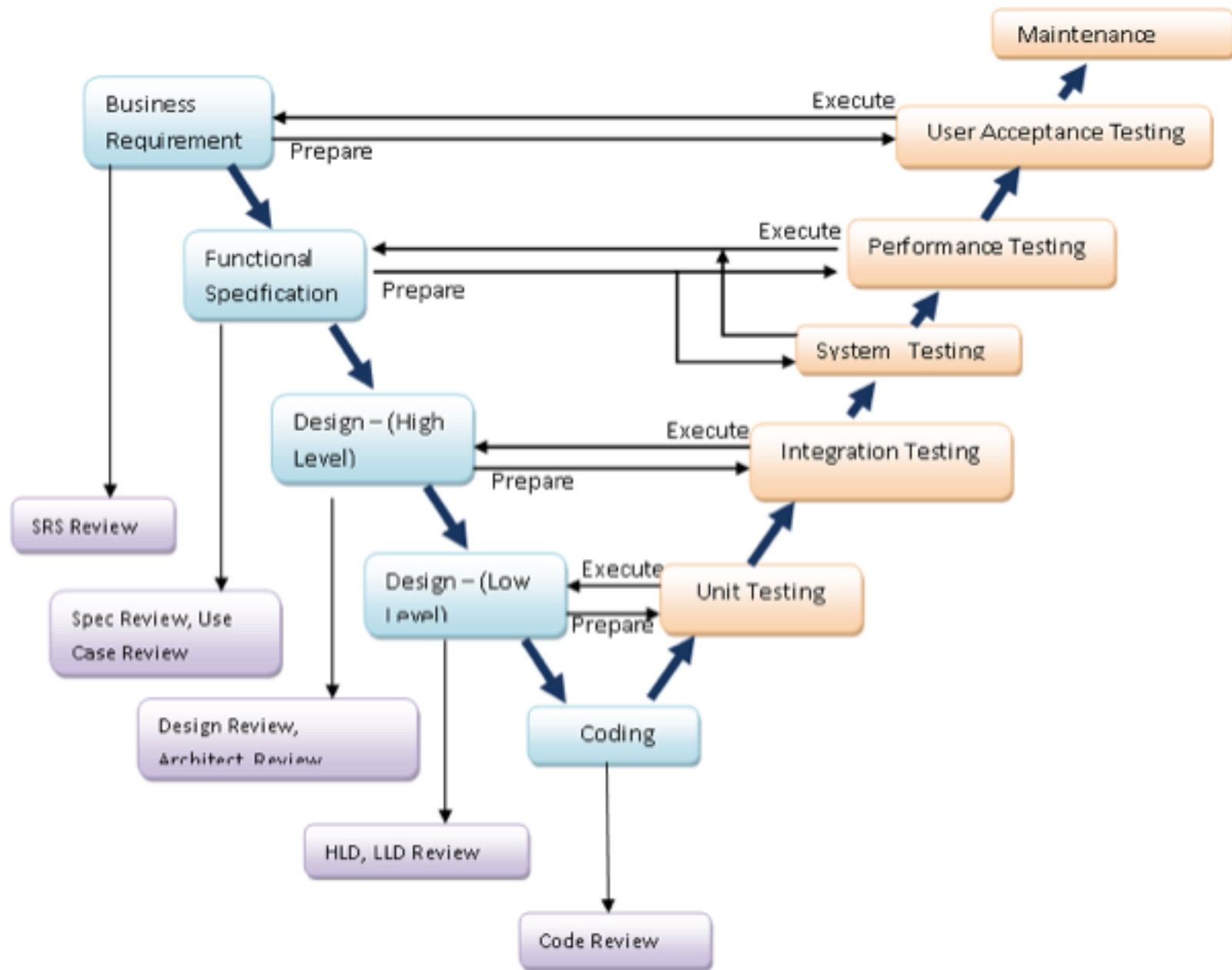
<http://onlinewebapplication.com/agile-test-methodology-model/>

F  
I  
X  
E  
D  
T  
I  
M  
E

## Sprint Iteration Planning



## Sprint Review



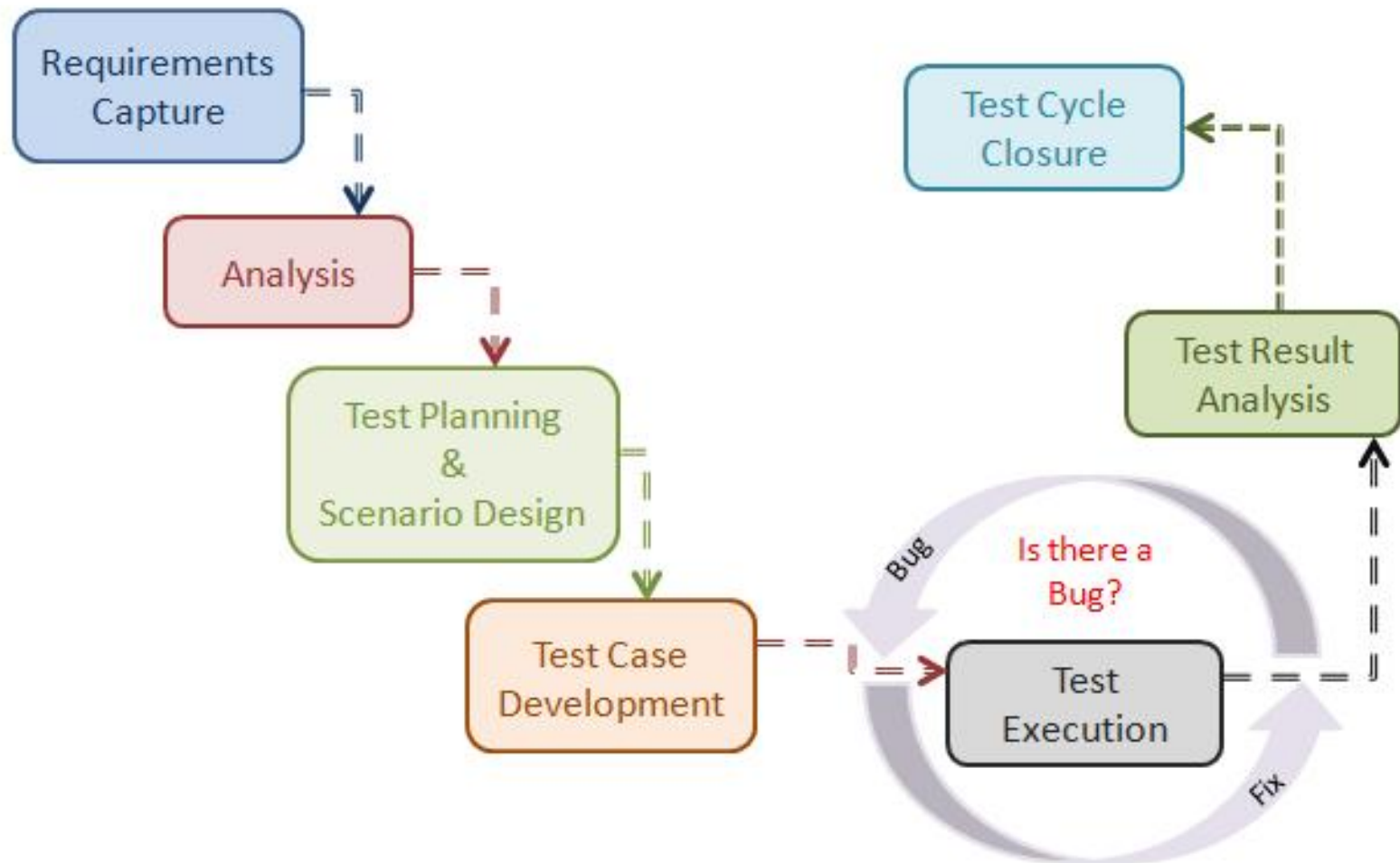
Agile Test Methodology Vs V Model Figure 1



# **PROCESSUS DE L'ACTIVITÉ TEST**



# Le cercle de vie de l'activité Testing



# Planification et contrôle

- Mesure et analyse de résultats
- Suivi et publication de rapport d'avancement
  - Couverture des tests
  - Critères de sortie
- Prise de décision
- Initiation des actions de correction

# Analyse et conception des tests

- Revue de la base des tests (ex: analyse de risques, interfaces spécifiées, documents de conception, etc.)
- Evaluation de la testabilité
- Identification et priorisation en fonction du plan de test/stratégie de test
- Conception des cas de test
- Identification des données de test
- Définition de l'environnement de test

# Implémentation et exécution des tests

- Écriture des scénarios de test
- Génération des données de test
- Validation des critères d'entrée
- Mise en place des environnements de test
- Exécution des cas de test
- Comparaison des résultats obtenus par rapport aux résultats attendus
- Analyse des déviations

# Evaluation des critères de sortie

- Synthèse des données en fonction des différentes activités
- Evaluation de la nécessité d'effectuer d'autres tests ou de modifier les critères
- Rédaction du rapport de synthèse des tests

# Clôture des activités de test

- Vérifier l'ensemble des documents dont ceux liés aux déviations et aux anomalies
- Utiliser les informations recueillies pour améliorer la maturité
- Versionner et archiver tous les éléments des tests

# Quelque soit le contexte système ou le projet

1. Choisir un scénario à exécuter (cas de test)
2. Estimer le résultat attendu (oracle)
3. Déterminer les données du test et le résultat concret attendu
  - État initial
  - Données du programme
  - Suite d'actions à exécuter
4. Exécuter le test
5. Comparer le résultat attendu et le résultat obtenu



# **PETIT FOCUS SUR LA CONCEPTION DES TESTS**



# Comment choisir les bons scénarios? (critères objectifs)

- Idéalement, il faudrait qu'ils soient exhaustifs mais cela est généralement impossible.
- Ils doivent donc permettre d'exécuter un maximum de comportements différents de l'application
  - Couverture des cas dits nominaux : les cas de fonctionnements les plus fréquents
  - Couverture des cas limites ou délicats
  - Entrées invalides
  - Montée en charge
  - Sécurité

# Comment choisir les bons scénarios? (critères qualitatifs)

- Ils doivent aussi permettre de contribuer à assurer la qualité de l'application: en trouvant rapidement le plus de bogues possibles
- Ils doivent aussi permettre de démontrer la qualité de l'application à un tiers: en étant le plus représentatif possible

# Comment choisir les bons scénarios? (critères plus subjectifs)

- L'expérience des anomalies: certains tests sont effectués en fonction d'anomalies récurrentes, de cas récurrents
- La connaissance du développeur:
  - ses défauts/qualités,
  - ses affinités par rapport aux fonctionnalités,
  - ses classiques



**CONCEPTION DES  
TESTS:  
LE TEST FONCTIONNEL**

# Le test fonctionnel:

*Tester ce que doit faire l'application*

- **Avantages:**
  - Ne dépend pas du code: langage, technique, style
  - Peut donc être défini avant le développement
  - Basé sur l'interface et les fonctionnalités, le nombre de scénarios est de taille raisonnable
  - Assure l'adéquation du code avec la spécification
- **Défauts:**
  - Ignore les défauts de programmation
  - La concrétisation des scénarios n'est pas toujours évidente
- **Utilisé pour tous les niveaux de test**

# Le test fonctionnel

## Exemple 1

- **Pré-conditions** : L'utilisateur doit être authentifié en tant que client ou commercial (Cas d'utilisation « S'authentifier » – package « Authentification »)
- **Démarrage** : L'utilisateur a demandé la page « Consultation catalogue »
- **Post-conditions** : Aucun
- **Ergonomie:**
  - L'affichage des produits d'une catégorie devra se faire par groupe de 15 produits. Toutefois, afin d'éviter à l'utilisateur d'avoir à demander trop de pages, il devra être possible de choisir des pages avec 30, 45 ou 60 produits.
- **Performance attendue** :
  - La recherche des produits, après sélection de la catégorie, doit se faire de façon à afficher la page des produits en moins de 10 secondes.
- **Problèmes non résolus** :
  - Nous avons fait la description basée sur l'information que les produits appartiennent à une catégorie. Est-ce qu'il existe des sous-catégories ? Si tel est le cas, la description devra être revue.
  - Etc.

# Le test fonctionnel

## Exemple 2

- **Le scénario nominal**
  1. **Le système** affiche une page contenant la liste les catégories de produits.
  2. *L'utilisateur* sélectionne une des catégories.
  3. **Le système** recherche les produits qui appartiennent à cette catégorie.
  4. **Le système** affiche une description et une photo pour chaque produit trouvé.
  5. *L'utilisateur* peut sélectionner un produit parmi ceux affichés.
  6. **Le système** affiche les informations détaillées du produit choisi.
  7. *L'utilisateur* peut ensuite quitter cette description détaillée.
  8. **Le système** retourne à l'affichage des produits de la catégorie (retour à l'étape 4)

# Le test fonctionnel

## Exemple 2

- **Les scénarios alternatifs**
- 2.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.
- 2.b *L'utilisateur* décide de quitter la consultation du catalogue.
- 5.a *L'utilisateur* décider de quitter la consultation de la catégorie de produits choisie.
- 5.b *L'utilisateur* décide de quitter la consultation du catalogue.
- 7.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.
- 7.b *L'utilisateur* décide de quitter la consultation du catalogue.



# Le test fonctionnel

## Exemple 3

Spécification: *retourne la somme de 2 entiers modulo 15000*

- *S1: 2 entiers dont la somme dépasse 15000*
- *S2: 2 entiers dont la somme ne dépasse pas 15000*

```
Somme(x,y)=  
  If (x=123 & y=421)  
    then resultat:= x-y  
    else resultat:=x+y  
  Return resultat
```

# Critère d'arrêt: test fonctionnel

*Quand arrêter de tester ou comment déterminer un bon jeu de test?*

- Un bon critère doit être: efficace, objectif, simple, automatisable
- Test fonctionnel:
  - couverture de scénarios d'utilisation
  - couverture des partitions des domaines d'entrée
  - couverture des cas limites

# Couverture des scénarios d'utilisation

Spécification: *trier un tableau  $T$  par ordre croissant sans doublon*

1. Choisir un scénario à exécuter (cas de test)
  1. S1:  $T$  est vide
  2. S2:  $T$  contient des doublons
  3. S3:  $T$  ne contient pas de doublons
2. Estimer le résultat attendu (oracle)
  1. O1: le résultat est un tableau vide
  2. O2, O3: le résultat est un tableau trié par ordre croissant sans doublon

# Couverture des scénarios d'utilisation

Spécification: *trier un tableau  $T$  par ordre croissant sans doublon*

3. Déterminer les données du test et le résultat concret attendu

S1	T= []	R=[]
S2	T=[3;3;4;5;0;4;-4]	R=[-4;0;3;4;5]
S3	T=[-3;-5;7;3;100]	R=[-5;-3;3;7;100]

# Couverture des partitions des domaines d'entrée

- À partir de la spécification
  - déterminer le domaine des entrées
- Partitionner le domaine des entrées en classes d'équivalences
  - identifier des classes d'équivalence pour chaque domaine d'entrée
  - les classes d'équivalence forment une partition du domaine de chaque donnée en entrée
  - choisir une donnée dans chacune

# Couverture des partitions des domaines d'entrée

- Le programme lit trois nombres réels qui correspondent à la longueur des côtés d'un triangle.
  - Si ces trois nombres ne correspondent pas à un triangle, imprimer un message d'erreur.
  - S'il s'agit d'un triangle, le programme détermine
    - s'il est isocèle, équilatéral ou scalène
    - et si son plus grand angle est aigu, droit ou obtu.

# Couverture des partitions des domaines d'entrée

	aigu	obtu	droit
scalène	6,5,3	5,6,10	3,4,5
isocèle	6,1,6	7,4,4	$\sqrt{2},2,\sqrt{2}$
équilatéral	4,4,4	impossible	impossible

# Couverture des cas limites

- Intuition:
  - de nombreuses erreurs se produisent dans les cas limites
- Pour chaque donnée en entrée
  - déterminer les bornes du domaine
  - prendre des valeurs sur les bornes et juste un peu autour
- Souvent utilisée avec la technique de partitionnement : les valeurs aux limites peuvent être des valeurs aux frontières des partitions



# Couverture des cas limites

- Le programme lit trois nombres réels qui correspondent à la longueur des côtés d'un triangle.
  - Si ces trois nombres ne correspondent pas à un triangle, imprimer un message d'erreur.
  - S'il s'agit d'un triangle, le programme détermine
    - s'il est isocèle, équilatéral ou scalène
    - et si son plus grand angle est aigu, droit ou obtu.

# Couverture des cas limites

1, 1, 2	non triangle
0, 0, 0	un seul point
4, 0, 3	une des longueurs est nulle
1, 2, 3.00001	presque triangle
0.001, 0.001, 0.001	très petit triangle
88888, 88888, 88888	très grand
3.00001, 3, 3	presque équilatéral
2.99999, 3, 4	presque isocèle
3, 4, 5.00001	presque droit
3, 4, 5, 6	quatre données
3	une seule donnée
5, 5, A	une lettre
	pas de donnée
-3, -3, 5	données négatives
...	



**CONCEPTION DES  
TESTS:  
LE TEST STRUCTUREL**

# Le test structurel

- **Avantages:**
  - Basé sur le code: le nombre de scénarios est plus important mais plus précis
  - Sensible aux défauts de programmation
- **Défauts:**
  - Le code doit être accessible
  - Problème d'Oracle: le résultat est-il celui réellement attendu?
  - Ignore les fonctionnalités absentes
- **Utilisé pour le test unitaire**

# Le test structurel

## Exemple

Spécification: *retourne la somme de 2 entiers modulo 15000*

Somme(x,y)=

    If (x=123 & y=421)

        then **resultat:= x-y**

        else **resultat:=x+y**

    Return resultat

- S1: 123 et 421
- S2: 124 et 421

# Critère d'arrêt: test structurel

*Quand arrêter de tester ou comment déterminer un bon jeu de test?*

- Un bon critère doit être: efficace, objectif, simple, automatisable
- Test structurel:
  - couverture des instructions, des enchaînements
  - couverture des conditions
  - couverture des dépendances de données

# Critère d'arrêt: test structurel

## PGCD de 2 nombres

**Précondition:** p et q entiers naturels positifs

```
pgcd: integer is  
local p,q : integer;  
do
```

```
  read(p, q) B1
```

```
  while p<>q do P1
```

```
    if p > q P2
```

```
      then
```

```
        p := p-q B2
```

```
      else
```

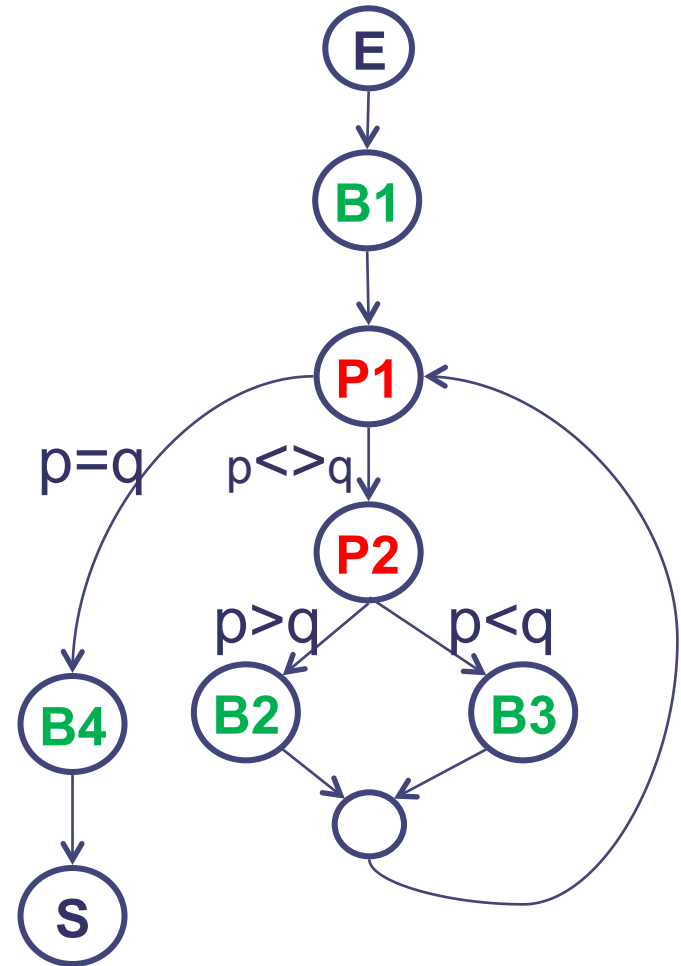
```
        q:= q-p B3
```

```
      end -- if
```

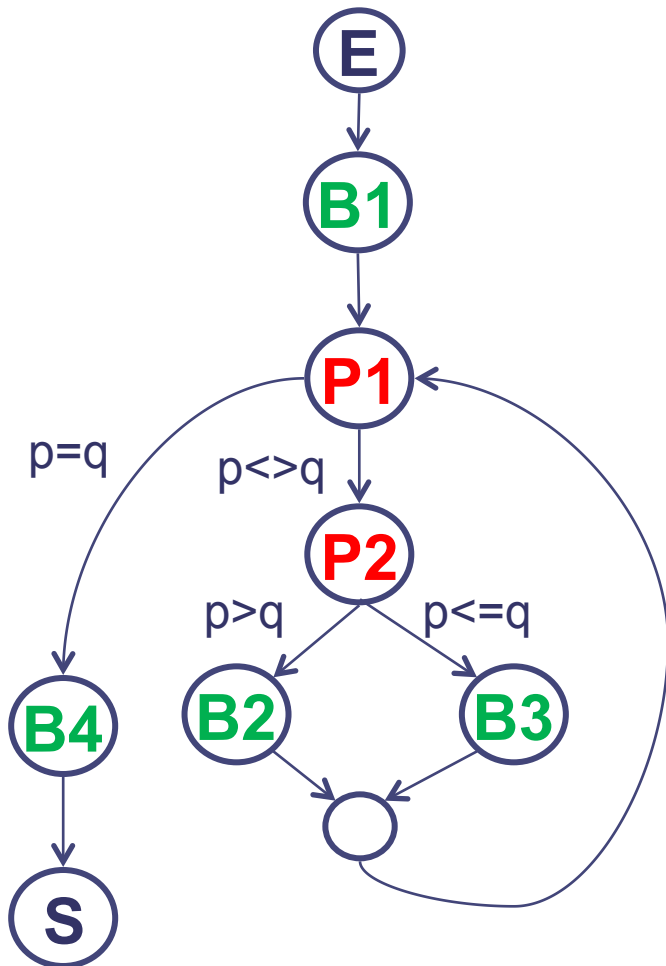
```
    end -- while
```

```
    result:=p B4
```

```
  end-- pgcd
```



# Critères d'arrêt: test structurel



*Tous les noeuds:*

(E, B1, **P1**, **P2**, B2, **P1**, B4, S)

(E, B1, **P1**, **P2**, B3, **P1**, B4, S)

*Tous les arcs : idem*

*Tous les chemins élémentaires (1-chemin) :*

idem + (E, B1, **P1**, B4, S)

*Tous les 2-chemins :*

idem +

(E, B1, **P1**, **P2**, B2, **P1**, **P2**, B2, **P1**, B4, S)

(E, B1, **P1**, **P2**, B2, **P1**, **P2**, B3, **P1**, B4, S)

(E, B1, **P1**, **P2**, B3, **P1**, **P2**, B2, **P1**, B4, S)

(E, B1, **P1**, **P2**, B3, **P1**, **P2**, B3, **P1**, B4, S)





# **EXÉCUTION DES TESTS**

Désolé, mais j'ai...



AQUAPONNEY !

# Les deux profils de testeurs



Developer

Understands the system  
but, will test "gently" and, is  
driven by "delivery"



Independent tester

Must learn about the system,  
but, will attempt to break it  
and, is driven by quality

# Automatisation des tests

- C'est très important car l'activité est **difficile** et **très chère**.
  - Difficultés:
    - Trouver des défauts n'est pas naturel (surtout chez les développeurs)
    - La qualité dépend de la pertinence des jeux de test
  - Coûts:
    - Logiciels critiques: >50% du coût total du développement.
    - Logiciel standard: environ 30% du coût total
- Exemple: environ 10 milliards de \$/an sont dépensés en tests rien qu'aux U.S.A

# Automatisation des tests

- Une bonne automatisation est synonyme:
  - D'amélioration de la qualité
  - De maintenance simplifiée
- Il faut donc:
  - beaucoup de scénarios efficaces
  - une plateforme dédiée (serveur, base de données)
- Quelques outils: Sélénium, TestComplete, Quality Test Plan, etc.
- Bien choisir son outil: exécution, dépouillement, mise à jour

# Pourquoi encore des tests manuels?

- Application très paramétrable
- Application qui doit tourner:
  - Sur toutes les plateformes (Windows, Mac, Unix, Linux)
  - Sur tous les navigateurs (de IE6 à IE11, Firefox, Chrome, etc.)
  - Avec une base de données (sous divers Oracle, divers SQLServer, DB2)
  - Dans plusieurs langues
- Un robot n'est pas un humain: ergonomie, design, temps de réponse, indiscipline, etc.



# **QUID DES *TECHNICAL* *FACTS***

# Exemple de bogues

- Problème de p
- DF2483: calcul
- Civilité: il man
- Le calcul des c
- Il requête avec
- Une personne
- J'arrête la sect
- La restauratio
- système dans
- L'heure du sys
- Une analyse «
- L'impression F
- Le nombre de



ème lot

ée met le

indows

ment



# <http://cartoontester.blogspot.fr/>

## BUGS HAVE FEELINGS TOO

IF YOU FIND A BUG:  
REPORT IT

BUGS DON'T LIKE  
TO BE FORGOTTEN



IF YOU FIND A BUG:  
GET TO KNOW THEM

BUGS LIKE TO BE  
UNDERSTOOD



This ladybird  
has 3 spots

IF YOU FIND A BUG:  
TAKE A PHOTO

BUGS LIKE TO KEEP MEMORIES  
OF THE OCCASION



IF YOU FIND A BUG:  
GET TO KNOW THEIR MATES

BUGS ARE SOCIALITES



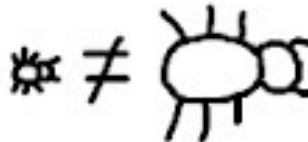
IF YOU FIND A BUG:  
REPORT IT QUICK

OTHERWISE BUGS SETTLE IN AND  
MAKE A HOME FOR THEM SELVES



IF YOU FIND A BUG:  
BE HONEST

BUGS DON'T LIKE  
GOSSIPS



IF YOU FIND A BUG:  
NOTE HOW YOU  
MEET THEM

BUGS ARE ROMANTICS



IF YOU FIND A BUG:  
DON'T IGNORE IT

BUGS CAN BITE IF  
NOT APPRECIATED



# Priorité vs sévérité

		SÉVÉRITÉ	
		CRITIQUE	NON CRITIQUE
PRIORITÉ	URGENTE	Une fonctionnalité clef ne fonctionne pas	Le logo de la société n'est pas de la bonne couleur
	PEU ATTENDRE	Une fonctionnalité rarement utilisé ne fonctionne pas	Le titre d'une image n'a pas la bonne fonte

# Exemple de bogues

- Problème de performance lors de l'affichage du menu
- DF2483: calcul incorrect
- Civilité: il manque « Monsieur le docteur »
- Le calcul des coûts est faux
- Il requête avec plus de 20 colonnes génère un log
- Une personne ne peut pas être deux fois propriétaire d'un même lot
- J'arrête la section D, mais c'est la C qui s'initialise
- La restauration d'une base pendant qu'une analyse est effectuée met le système dans un état instable
- L'heure du système n'est pas au même format que l'heure windows
- Une analyse « automatic only » peut être effectuée manuellement
- L'impression PDF met 2,58 secondes au lieu de 2 secondes
- Le nombre de page indiqué est 99/13
- Le changement d'heure de Java 1.6 n'est pas toujours le même que celui de Windows 2008 (bogue Windows) ou de Windows 2016 (bogue Java)

Toutes les anomalies doivent être corrigées, bien sûr!



# Pour finir ou presque

- Le test est un métier à part
- Il doit être effectué à tous les niveaux
- Il doit être effectué avec le plus d'indépendance possible
- Il faut accepter qu'il reste des bogues...



# RÉFÉRENCES

# Bibliographie (1/2)

- Syllabus de l'ISTQB

<http://www.istqb.org/>

- G. Myers *The Art of Software Testing*
- Les cours de Y. Le Traon et B. Baudry

[http://www.irisa.fr/triskell/perso\\_pro/yletraon/cours/](http://www.irisa.fr/triskell/perso_pro/yletraon/cours/)

- Les cours de B. Legeard et F. Bouquet

[http://lifc.univ-fcomte.fr/~bouquet/Test/cours\\_Test.pdf](http://lifc.univ-fcomte.fr/~bouquet/Test/cours_Test.pdf)

# Bibliographie (2/2)

- Le cours de I. Parissis

*<http://membres-liglab.imag.fr/donsez/ujf/mlinfo/tagl/Test-Logiciel-TAGL.pdf>*

- Le cours de D. Longuet

*[https://www.lri.fr/~longuet/ante\\_enseignements.html](https://www.lri.fr/~longuet/ante_enseignements.html)*

- Le cours de S. Bardin

*<http://sebastien.bardin.free.fr/>*



<https://www.ministryoftesting.com>



# Standards/Normes (1/2)


- ISTQB: International Software Testing Qualification Board
  - <http://www.istqb.org/>
  - Organisme de certification, leader mondial sur la certification du test logiciel
- CFTL: Comité Français des Tests Logiciels

# Standards/Normes (2/2)

- IEEE (Institute of Electrical and Electronics Engineers)
  - IEEE 610: standard définissant les termes de l'ingénierie logicielle
  - IEEE 829: standard définissant la documentation pour le test de logiciel
- ISO (International Organization for Standardization)
  - ISO 9126: norme définissant un langage commun pour modéliser les qualités d'un logiciel

# Liens sympas

- <https://richrtesting.com/2015/12/15/star-flaws-the-empire-strikes-out/>
- <https://www.commitstrip.com/fr/>
- <http://www.ministryoftesting.com/>



*Mon ordinateur, j'essaie de faire tout ce qu'il me dit  
mais lui il fait rien de ce que je veux.*

**Anne Roumanof, Internet**

# Les différentes méthodes V&V

- **Test dynamique**
- Test statique: **revue de code** ou de spécifications, etc.
- Vérification symbolique: exécution symbolique, etc.
- Vérification formelle: preuve, model-checking, etc.

# Le test(ing) en entreprise

- Le test est effectué par deux types de profils: les développeurs et les testeurs
- Le test effectué généralement en entreprise est un test dynamique: il s'agit d'exécuter du code pour s'assurer d'un fonctionnement correct
- Il appartient aux méthodes dites V & V:
  - Validation: l'application répond-t-elle aux besoins du client?
  - Vérification: l'application fonctionne-t-elle correctement?

# Le développeur, un bâtisseur

- C'est un expert qui trouve les bons algorithmes et la meilleure solution aux problèmes
- Il interprète les éventuelles ambiguïtés de la spécification
- Il peut oublier les détails visuels
- Il n'a pas une vue d'ensemble
- Il travaille *sans limite de temps*
- Il n'aime pas les erreurs qui montrent qu'il est faillible
- Il voit le testeur comme un messenger porteur de mauvaise nouvelle et voit donc son travail comme une activité destructive



# Le testeur, un destructeur ?

- Il trouve les cas où le logiciel peut être défaillant
- Il a une vue d'ensemble
- Il vérifie que le logiciel répond bien aux exigences
- Il anticipe les problèmes possibles
- Il donne de l'importance au détail surtout s'il s'agit d'une exigence
- Il est limité par le temps et n'a pas le droit aux débordements
- Plus il trouve de bogues, plus il est bon (*vraiment?*)
- Il voit son travail comme une activité constructive

# Et pourtant: L'ennemi de mon ennemi (le bogue) est mon ami...

- Testeurs et développeurs travaillent pour un même objectif: améliorer la qualité du logiciel
- Les testeurs ne sont pas là pour juger le travail des développeurs, d'ailleurs, que dire:
  - des tests qui remontent peu de bogues:
    - Les tests sont mauvais ou en nombre insuffisants
    - Le développement est de bonne qualité
  - des tests qui remontent beaucoup de bogues:
    - Les tests sont complets: la majeure partie des bogues a été remontée
    - Le développement est de très mauvaise qualité: quid de son avenir.

# Focus sur la PME Foederis (2008-2013)

- Éditeur de logiciel de ressources humaines
- Plusieurs versions à maintenir, patchs hebdomadaires, mode licence et SAAS
- Gestion de projet: méthode V réduite
- Outil pour les tests: Excel
- Automatisation des tests fonctionnels avec TestComplete (> 200)
- Gestion des anomalies:
  - Outil maison
  - Sévérité: Bloquant, Majeur, mineur
  - Priorité: en fonction du contexte client

# Focus sur le groupe Sopra (2013-2016)

- Editeur de logiciel de gestion d'immobilier
- Plusieurs outils, plusieurs versions, fréquence des patches en fonction des outils
- Gestion de projet: eMedia (V+Agile)
- Outils pour les tests: HPQC
- Automatisation des tests fonctionnels avec QTP
- Gestion des anomalies:
  - Outil: PeopleSoft, JIRA, HPQC
  - Sévérité/Priorité: Bloquant/P1, Majeur/P2, mineur/P3

# Focus sur le groupe Thales Services (2016-) et le client bioMérieux

- Producteur de produits d'analyses médicales (instrument, logiciel, PC)
- Plusieurs produits, plusieurs versions, fréquence des patchs en fonction des outils
- Gestion de projet: V pure et sécurisé car auditable
- Outils pour les tests: TestTrack Suite
- Automatisation des tests fonctionnels avec JUnit

# Focus sur le groupe Thales Services (2016-) et le client bioMérieux

- Gestion des anomalies:
  - Outil: celui des clients, TestTrackSuite
  - 5 niveaux de sévérité:
    - 1 - Compromises product safety and efficacy
    - 2 - Function critical to intended use not running; no work around
    - 3 - Major inconvenience to user possibly needing a work around
    - 4 - Minor inconvenience to user
    - 5 - Transparent to user with no performance effects
  - Anomaly Review Board
  - Priorité: prise en compte de plusieurs paramètres dont l'occurrence

# LE Manifeste DU TEST



LE



DU

# TEST



Nous valorisons :

Tester  
au fil de l'eau  
**PLUS QUE**  
Tester  
à la fin

Eviter  
les bugs  
**PLUS QUE**  
Trouver  
les bugs

Tester la bonne  
compréhension  
**PLUS QUE**  
Vérifier  
la fonctionnalité

Construire le  
meilleur système  
**PLUS QUE**  
Casser  
le système

La responsabilité  
de l'équipe sur  
la qualité  
**PLUS QUE**  
La responsabilité  
du testeur

[www.growingAgile.co.za](http://www.growingAgile.co.za)

@growingAgile

# Ai-je le profil d'un testeur?

- Il faut:
  - De la curiosité
  - Du pessimisme professionnel
  - Un œil critique
  - L'attention du détail
  - Savoir être neutre et factuel
  - Savoir être diplomate
  - Savoir communiquer
  - Ne pas donner de solution
  - Ne pas être critique ou juge



# C'est quoi un bogue?

- **Anomalie** (fonctionnement) : différence entre comportement attendu et comportement observé
- **Défaut** (interne) : élément ou absence d'élément dans le logiciel entraînant une anomalie
- **Erreur** (programmation, conception) : comportement du programmeur ou du concepteur conduisant à un défaut

# Code Review

- The code review objectives are :
  - to check that code does what it is supposed to do
  - to ensure maintainability of code
  - to detect bugs and potential performance issues
  - to train the software development team members with good coding practices
- The assembly must be focused on two different points:
  - Bugs tracking  
The objectives are to identify the obvious bugs, to verify that a feature is correctly implemented, or at least looks like to be correctly implemented.
  - Understanding the code  
People of the assembly should be able to maintain the code presented after the review. To do so, it is highly recommended to ask questions on all parts of the code. Or even to ask to add some more comments in the code itself.

# Définitions

- **Jeu de test:** ensemble de cas de test
- **Script de test:** code/script qui exécute automatiquement les étapes 4 et 5 en fonction de plusieurs données issues de l'étape 3

# Mais tout ne s'explique pas que par le manque de tests.....

- La réalité des projets (chiffres GARNET)
  - Dérive dans le temps
    - 51% des projets ne respectent pas leurs dates de livraison
    - 16 % des projets livrent dans les temps et avec leur budget prévu
  - Surcoût du projet
    - 43% des projets dépassent leur budget initial
    - dont 53% des projets coûtent 189% du budget initial estimé

