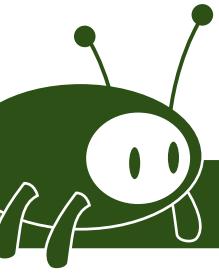


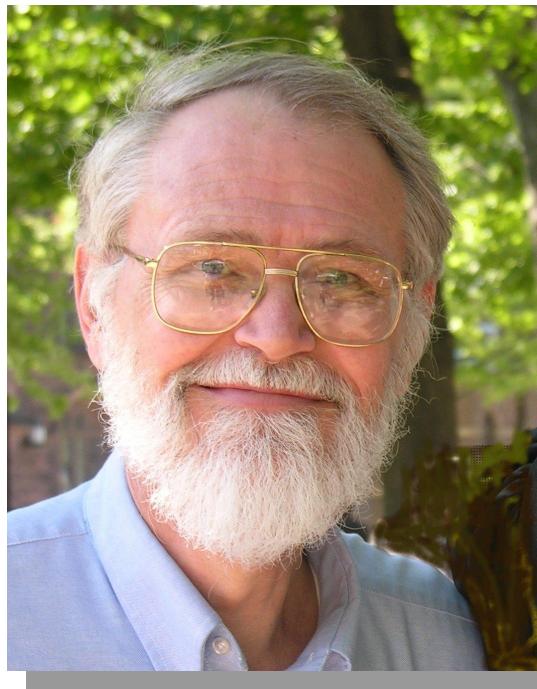


---

# Les tests logiciels

## Techniques de test

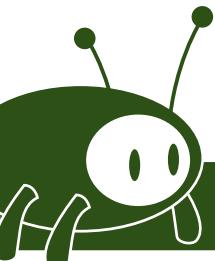




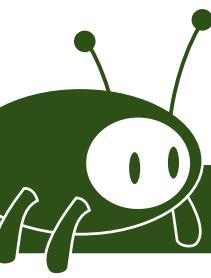
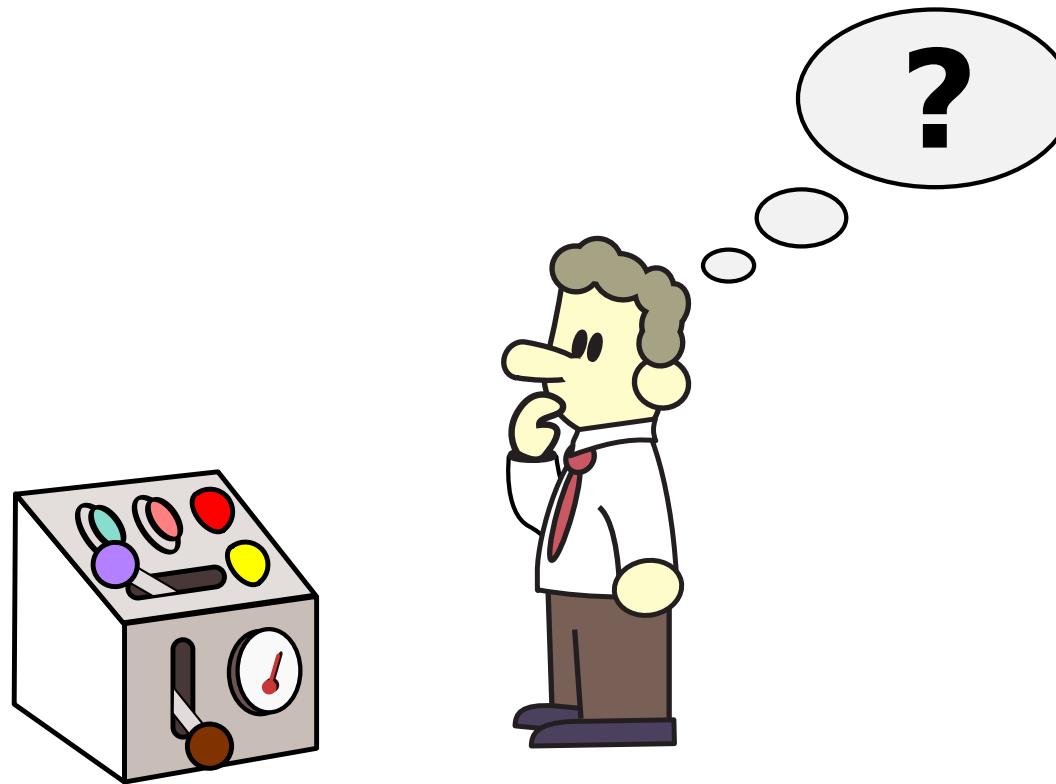
Déboguer est deux fois plus difficile que d'écrire le code en premier lieu. Dès lors, si vous écrivez le code aussi intelligemment que possible, vous êtes, par définition, pas assez futé pour le déboguer.



**Brian Kernighan** (1942)  
informaticien canadien



# Un système bizarre

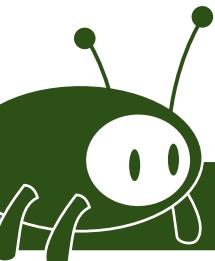




## Test logiciel (*software testing*)

Processus consistant à analyser un composant logiciel pour y détecter des différences entre les conditions existantes et les conditions requises, et pour en évaluer les fonctionnalités

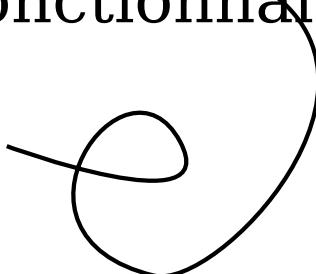
(Source: IEEE)



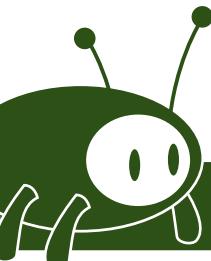
## Test logiciel (*software testing*)

Processus consistant à analyser un composant logiciel pour y détecter des différences entre les conditions existantes et les conditions requises, et pour en évaluer les fonctionnalités

Bug



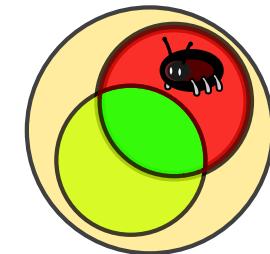
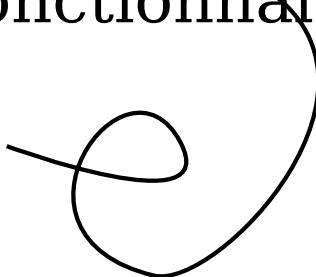
(Source: IEEE)



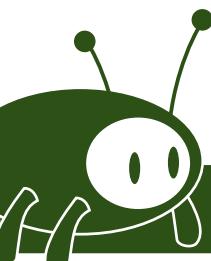
## Test logiciel (*software testing*)

Processus consistant à analyser un composant logiciel pour y détecter des différences entre les conditions existantes et les conditions requises, et pour en évaluer les fonctionnalités

Bug



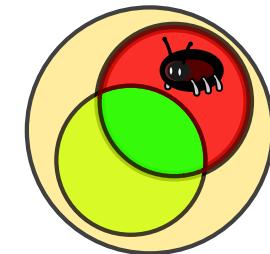
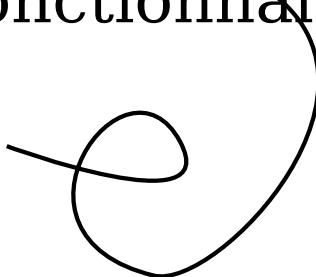
(Source: IEEE)



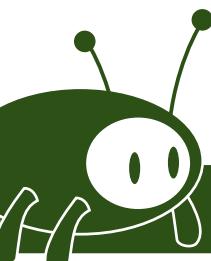
## Test logiciel (*software testing*)

Processus consistant à analyser un composant logiciel pour y détecter des différences entre les conditions existantes et les conditions requises, et pour en évaluer les fonctionnalités

Bug



(Source: IEEE)

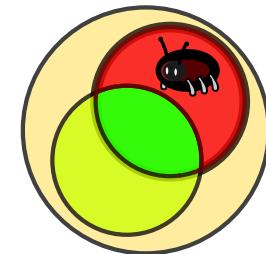
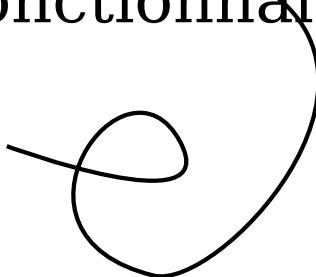




## Test logiciel (*software testing*)

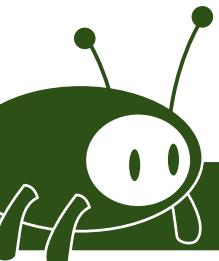
Processus consistant à analyser un composant logiciel pour y détecter des différences entre les conditions existantes et les conditions requises, et pour en évaluer les fonctionnalités

Bug



## Fiabilité

La capacité d'un système à fournir la fonctionnalité attendue, selon les conditions établies, pour une période de temps spécifique

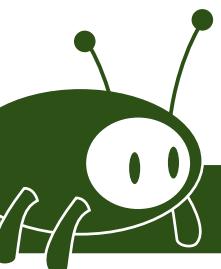
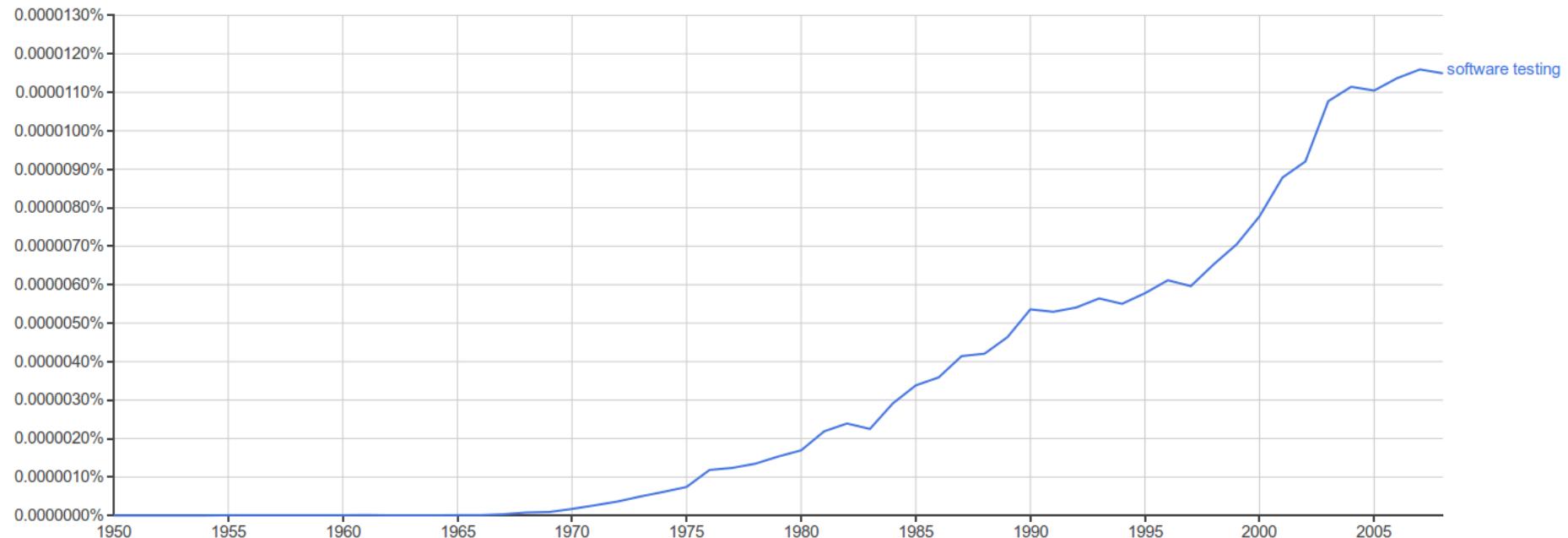


(Source: IEEE)



## Google books Ngram Viewer

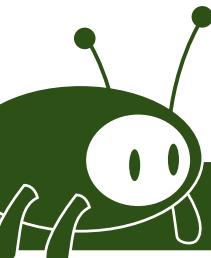
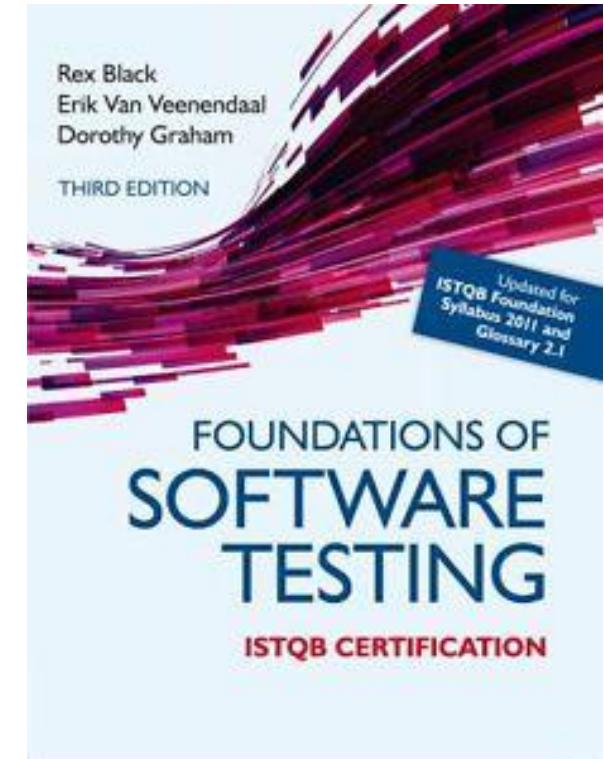
Graph these comma-separated phrases:   case-insensitive  
between  and  from the corpus  with smoothing of





## Quelques organismes en lien avec la certification ou la normalisation de processus informatiques

- International Software Testing Qualifications Board:  
certification ISTQB 2011 2.1
- Control Objectives for Information and related Technology (COBIT)
- Information Technology Infrastructure Library (ITIL)





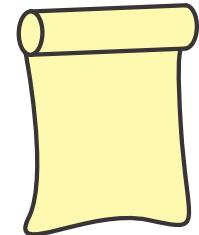
## Oracle (de test)

Processus chargé d'observer le résultat produit par le système (données, comportement) et de déclarer si le test est un succès ou un échec



## Verdict

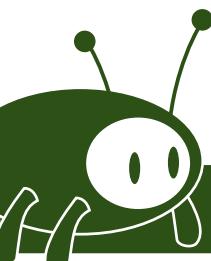
État de succès ou d'échec du test



Système faisant l'objet du test  
(*System Under Test* ou SUT)

## Testeur

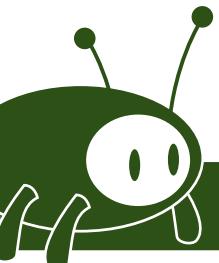
Personne ou processus chargé de l'exécution d'un scénario de test



## Tester ≠ Déboguer

Les tests servent à trouver la présence de défauts

Le débogage est une activité de **développement** qui vise  
à **corriger** les défauts trouvés



Les



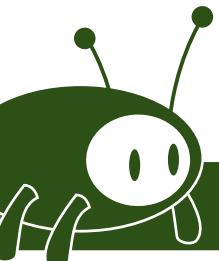
principes de  
test





## Les tests servent à exhiber la présence de défauts.

- Chaque test positif réduit la **probabilité** que des défauts subsistent (jamais jusqu'à zéro)
- Si aucun défaut n'est trouvé, pas une preuve que le système est absolument correct





## Les tests exhaustifs sont impossibles.

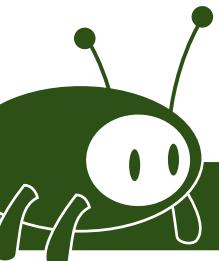
- Ceci inclut toutes les combinaisons possibles de valeurs d'entrée du système et de conditions préalables
- Impossible de couvrir tous ces cas, sauf pour des systèmes triviaux
- On doit donc **prioriser** ce que l'on souhaite tester
- Cf. analogie des histoires possibles





## Il faut tester tôt.

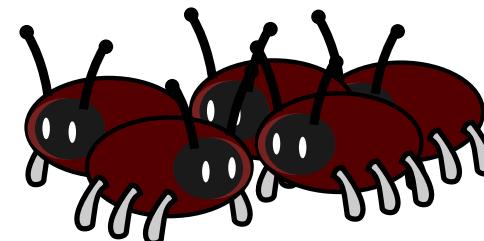
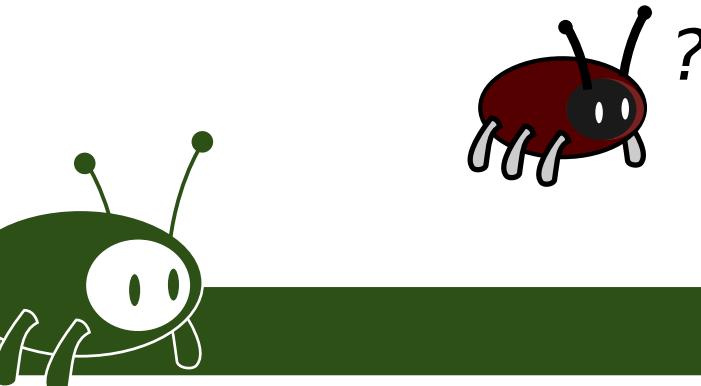
- Afin de trouver des défauts rapidement, les activités de test devraient être commencées aussitôt que possible dans le cycle de développement.
- Mieux valent de petits efforts réguliers qu'une grande phase de test gardée pour la fin!
- Cf. graphique du coût de correction d'un défaut



## 4

Concentrer ses énergies là où il y a des problèmes (*defect clustering*)

- L'effort de test devrait être proportionnel à la densité de défauts anticipés/trouvés
- Loi de **Pareto** (80/20): un petit nombre de modules contient souvent la majorité des défauts...
- ...i.e. les bugs se tiennent entre eux!





## Le paradoxe des pesticides

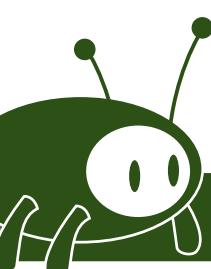
- Si on répète toujours le même ensemble de tests, éventuellement cet ensemble ne trouvera plus aucun nouveau défaut



BJ

- BJ Rollison: "le système apprend à passer les tests"

- On doit renouveler l'ensemble de tests pour faire appel à des parties différentes du système





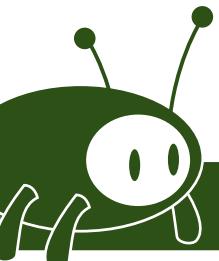
## Tester dépend du contexte

- On ne teste pas tous les systèmes de la même manière (avionique vs. site web)



- BJ Rollison: "je n'ai rien à dire à ce propos"

BJ





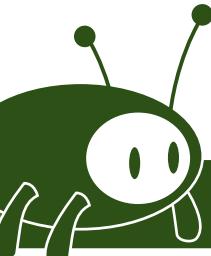
## Gare à l'absence d'erreurs!

- Un système exempt d'erreurs n'aura pas nécessairement du succès
- Celui-ci doit tout de même **répondre à un besoin** et être utilisable

```
Processes: 71 total, 2 running, 69 sleeping... 227 threads
Load Avg: 0.31, 0.34, 0.27 CPU usage: 3.6% user, 14.8% sys, 81.6% idle
Shared Avg: 0.31, 0.34, 0.27 CPU usage: 3.6% user, 14.8% sys, 81.6% idle
SharedAvg: num = 197, resident = 30.0M code, 4.48M data, 4.78M Link-Edit
MemRegions: num = 10489, resident = 892M + 12.8M private, 108M shared
PageMgmt: 231M wired, 812M active, 244M inactive, 988M used, 34.8M free
VM 12.1G + 128M 327980() pageins, 373133() pageouts

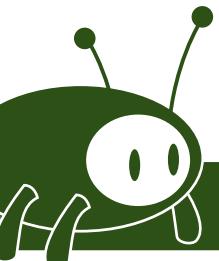
PID COMMAND %CPU TIME #THR #PRTS #REGS RPRVT RSSRD RSIZE VSIZE
22839 DashboardC 0.0% 0:31.78 4 103 147 6.71M 4.86M 18.4M 348M
22838 DashboardC 0.0% 0:37.00 4 104 157 7.38M 5.18M 19.1M 347M
22837 DashboardC 0.0% 0:01.38 3 98 133 1.82M 4.88M 12.8M 342M
22836 DashboardC 0.0% 0:24.32 4 109 176 4.82M 6.49M 17.7M 353M
22834 DashboardC 0.0% 0:39.80 4 104 162 7.64M 8.68M 19.8M 348M
22833 DashboardC 0.0% 0:08.09 3 99 164 3.84M 6.83M 17.7M 362M
22826 iChat 0.0% 0:14.33 7 269 364 11.7M 11.1M 14.8M 390M
22840 Transmit 0.0% 0:05.11 8 129 334 38.6M 11.0M 40.7M 416M
22700 Safari 4.8% 48:47.94 12 553 3303 399M 32.2M 374M 1.12G
23147 cut 0.0% 0:00.00 1 13 17 118K 638K 349K 28.6M
23148 grep 0.0% 0:00.00 1 13 18 140K 708K 432K 28.7M
23145 top 0.0% 0:00.08 1 16 20 308K 700K 772K 27.0M
23144 sh 0.0% 0:00.00 1 13 16 118K 948K 578K 27.1M
23142 top 0.0% 0:00.00 1 13 16 118K 948K 578K 27.1M
22875 bash 10.2% 0:00.31 1 22 20 824K 700K 928K 26.9M
22874 login 0.0% 0:00.01 1 14 18 232K 1.08M 816K 27.1M
22872 Terminal 0.0% 0:00.28 4 90 126 1.64M 7.74M 6.82M 343M
```

VS.



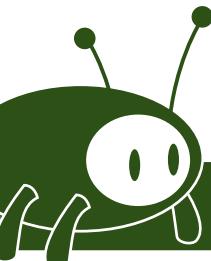
L'ISTQB définit le processus fondamental de test en 5 activités:

1. Planification et contrôle
2. Analyse et design
3. Implémentation et exécution
4. Évaluation des critères de sortie et rapport
5. Fermeture des tests



## 1. Planification et contrôle

- Définir les objectifs du test
- Découvrir et spécifier les activités requises pour satisfaire ces objectifs
- Comparer l'évolution du processus de test aux objectifs
- (Possiblement) Modifier la planification



## 2. Analyse et design

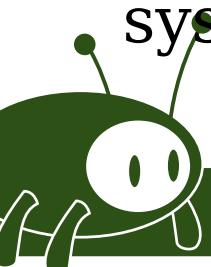
- Transformer les objectifs de l'étape 1 en scénarios de tests et en **conditions** de test
- Réviser la base de tst
- Évaluer la "testabilité" de la base de test pour faire ces objectifs

### Condition de test

Item ou événement d'un système qui pourrait être vérifié par un ou plusieurs scénarios de test (ex.: fonction, transaction, etc.)

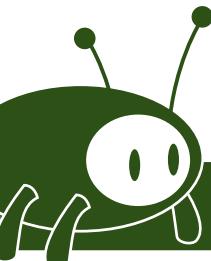
### Base de test

Documents à partir desquels les exigences sur un système peuvent être déduites



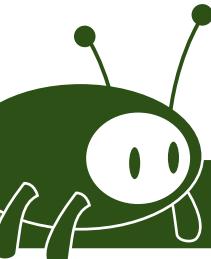
## 3. Implémentation...

- Spécifier des procédures de test (i.e. transformer les scénarios de haut niveau en scénarios concrets)
- **Formellement** (ex.: écrire un script) ou **informellement** (produire un test à la volée en lisant la description)
- Prioriser les procédures de test pour obtenir l'ordre d'exécution optimal
- Si on automatise, préparer le harnais de test (*test harness*) et écrire les scripts
- Vérifier que l'environnement de test a été correctement mis en place



## 3. ...et exécution

- Exécuter les procédures de test manuellement ou au moyen d'outils automatisés, selon la séquence prévue
- Comparer les résultats obtenus avec les résultats attendus. Les différences sont rapportées comme des **incidents**.
- Rapporter le résultat (verdict) de l'exécution des tests (incluant infos sur l'environnement, versions, etc.)
- Analyser les incidents pour en déterminer la cause.
- Peuvent être issus d'un défaut du système testé, ou d'erreurs dans les données utilisées pour le test ou son exécution.
- Répéter ces activités au besoin



## 4. Évaluation des critères de sortie

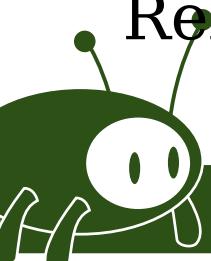
- Confronter l'exécution des tests aux objectifs établis
- Vérifier les logs de tests
- Déterminer si d'autres tests sont nécessaires ou si les critères de sortie doivent être changés
- Écrire un rapport de test pour le client

### Critères de sortie

Ensemble des conditions génériques et spécifiques, établies avec les clients, permettant à un processus d'être officiellement complété. Dans le cas des tests, permet de décider quand on peut arrêter de tester

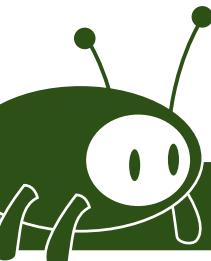
### Log de tests

Relevé chronologique d'événements pertinents à l'exécution



## 5. Fermeture des tests

- Déterminer si les livrables ont été livrés
- Vérifier que tous les incidents rapportés ont été résolus
- Documenter l'acceptabilité du système
- Finaliser et archiver l'environnement de test et le passer à l'organisation responsable de la maintenance
- Tirer des leçons des processus de tests



On peut classer les tests selon plusieurs critères:

- La **technique** employée pour générer des scénarios  
Ex.: tests concoliques, combinatoires, de valeurs limites, etc.
- L'**objectif** visé par le test  
Ex.: tests de régression, de confirmation, de stress, exploratoires, unitaires, etc.
- La **composante** sujette au test  
Ex.: test de réseau, d'interface graphique
- Le degré d'accès à l'**implémentation** interne  
• Ex.: *black/white box testing*



## Test de confirmation

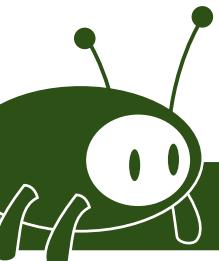
Consiste à exécuter des scénarios de test qui ont échoué lors de leur dernière tentative, de manière à s'assurer du succès des correctifs mis en place

## Test d'acceptabilité

Ensemble de scénarios visant à s'assurer que le produit réalisé respecte les exigences du client (IEEE 610)

## Test *ad hoc*

Test réalisé informellement, sans préparation et sans design de test



## Test de régression

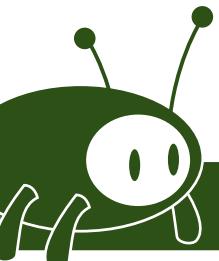
Test d'un programme déjà testé, après des modifications, pour s'assurer que des défauts n'ont pas été introduits ou révélés dans des portions inchangées du programme.

## Test aléatoire (*monkey testing*)

Stratégie de test où des scénarios sont choisis en utilisant un générateur pseudo-aléatoire.

## Test de performance

Processus de test visant à déterminer la performance d'un produit logiciel.



## Test aléatoire (*monkey testing*)

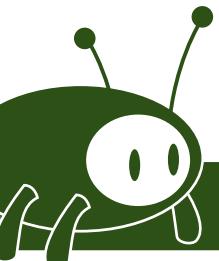
Technique de test où des scénarios sont fabriqués en utilisant un générateur pseudo-aléatoire.

## Test combinatoire

Technique de test systématique où un ensemble de scénarios est généré, visant à couvrir certaines combinaisons de valeurs de paramètres d'entrée

## Test de valeurs limites (*Boundary Value Testing*)

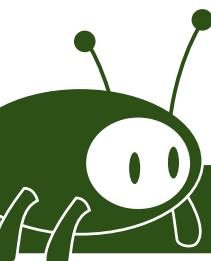
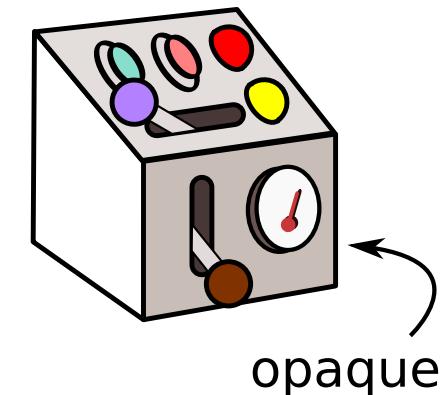
Technique de test où les valeurs des paramètres d'entrée sont choisies en fonction des frontières de leur domaine





## *Black box testing*

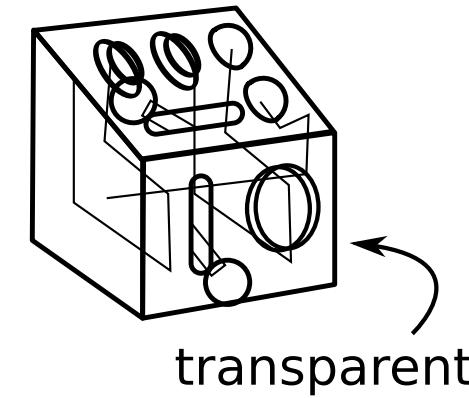
- On connaît la fonctionnalité attendue du système, mais pas son implémentation précise
- Les fonctionnalités sont testées en observant le comportement **externe**
- La fonctionnalité est testée "de l'extérieur"; dans le cas d'un logiciel, c'est donc son **interface** ( $\neq$  GUI):
  - Quelles sont les entrées du système?
  - Que peut-on faire de l'extérieur pour affecter le système?
  - Quelle est la sortie du système?





## *White box testing*

- On connaît le fonctionnement interne du système (i.e. son code)
- On peut tester ce qui se passe à un grand niveau de détail: conditions évaluées, chemins empruntés, boucles itérées, etc.
- La connaissance de l'implémentation peut servir à choisir les scénarios de test appropriés
- En général, un test exhaustif est impossible à ce niveau de détail



transparent

