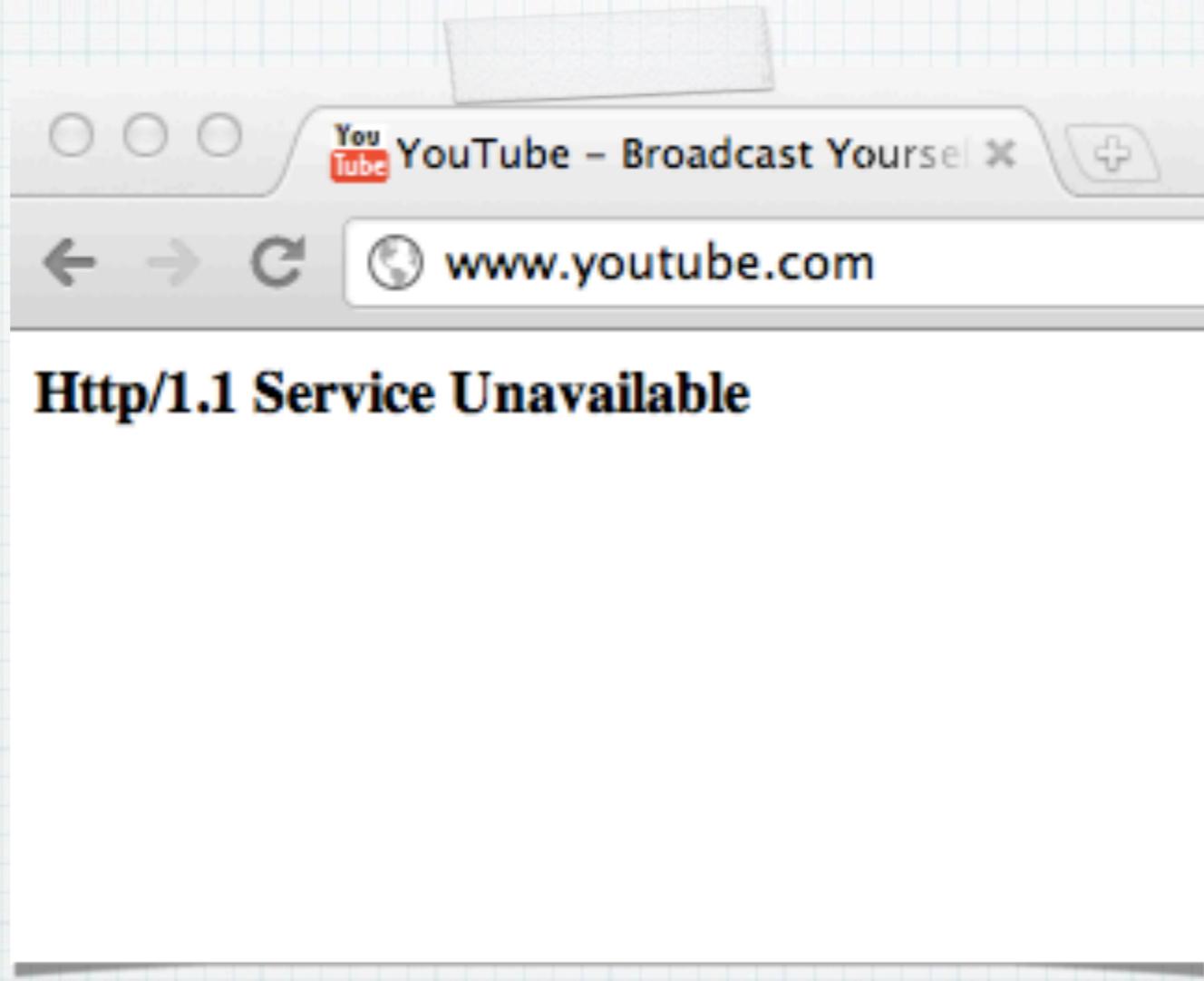


Faire des tests



Les tests ça sert à
rien !

C'est quoi un test ?

- * La vérification d'un objectif.
- * Simplement: On effectue une action, on vérifie que le résultat est le résultat attendu.

Des tests oui... Mais automatisés

- * Le problème avec les tests «à la main»:
 - * Il faut trouver une main disponible
 - * Dans ce cas il faut automatiser les tests

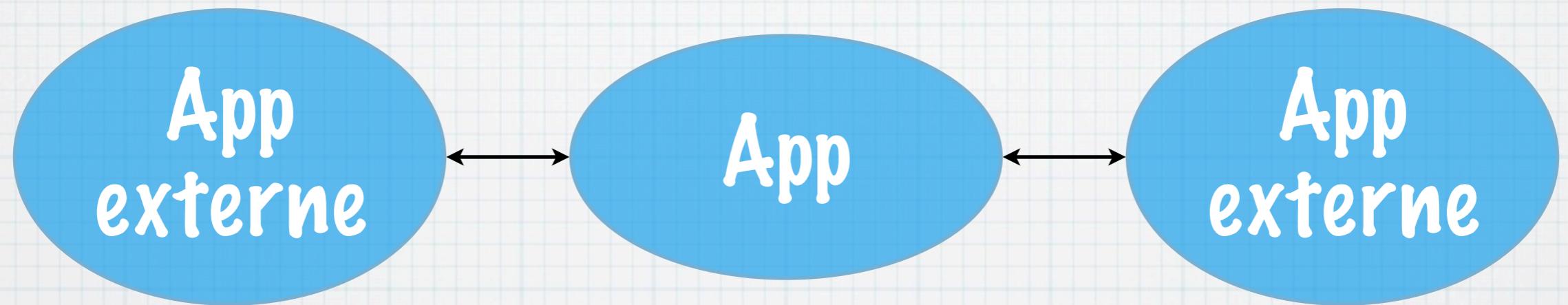
Pourquoi faire des tests ?

- * Assurer la qualité du produit finit.
- * Vérifier qu'il n'y a pas de régression.
- * Déetecter les bugs.

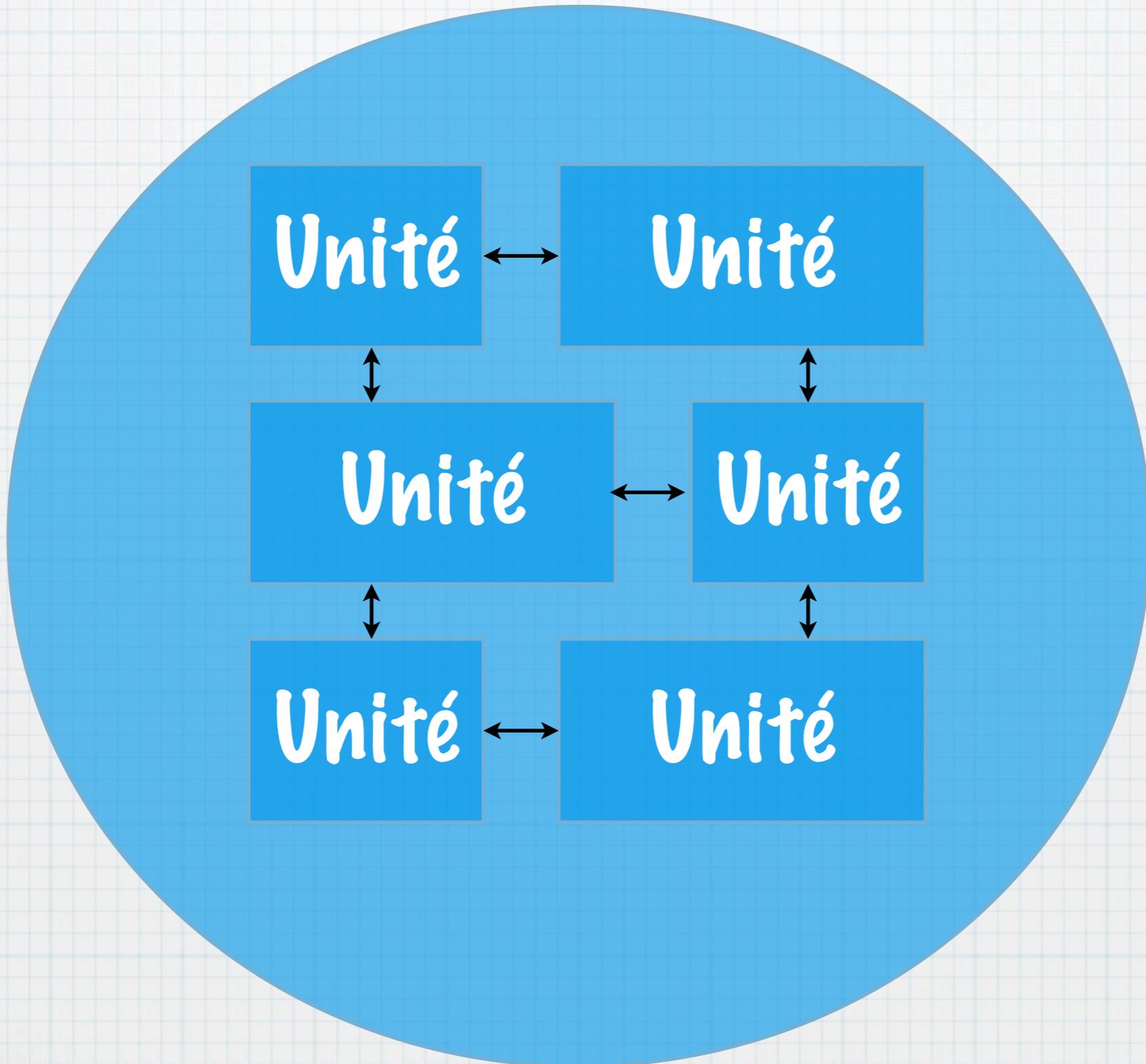
Oui mais surtout...

- * C'est un moyen de gérer la peur pendant le développement - Kent Benck
- * Au moindre doute, le développeur peut lancer les tests.

Une application...



Des unités



Les tests

- * Terminologie:
 - * La partie de l'application que l'on teste est souvent appelé «SUT» (System Under Test).
- * Que tester ?
 - * Tout...
 - * Sauf les librairies/services externes.

Les tests en 5 étapes

- 1) Préparer l'environnement
- 2) Initialiser les variables
- 3) Faire l'appel
- 4) Vérifier l'environnement et la réponse
- 5) Nettoyer l'environnement

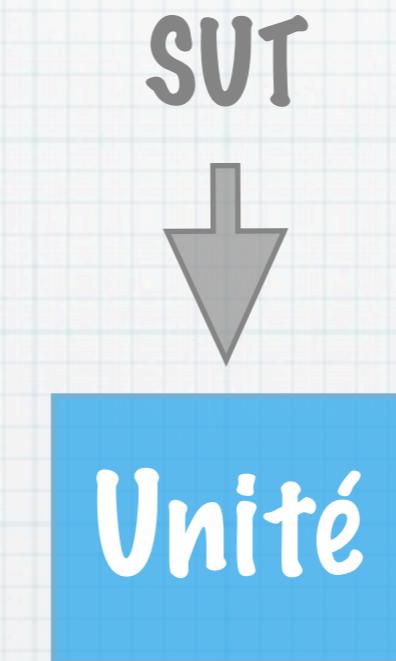
Organisation des tests

- * Les tests sont organisés en TestCase. Ils constituent un ensemble logique de tests (le même SUT, ...).
- * Les TestCase sont organisés en TestSuite. Ils constituent là aussi un ensemble logique de TestCase (tous les tests, tous les tests unitaires, ...).

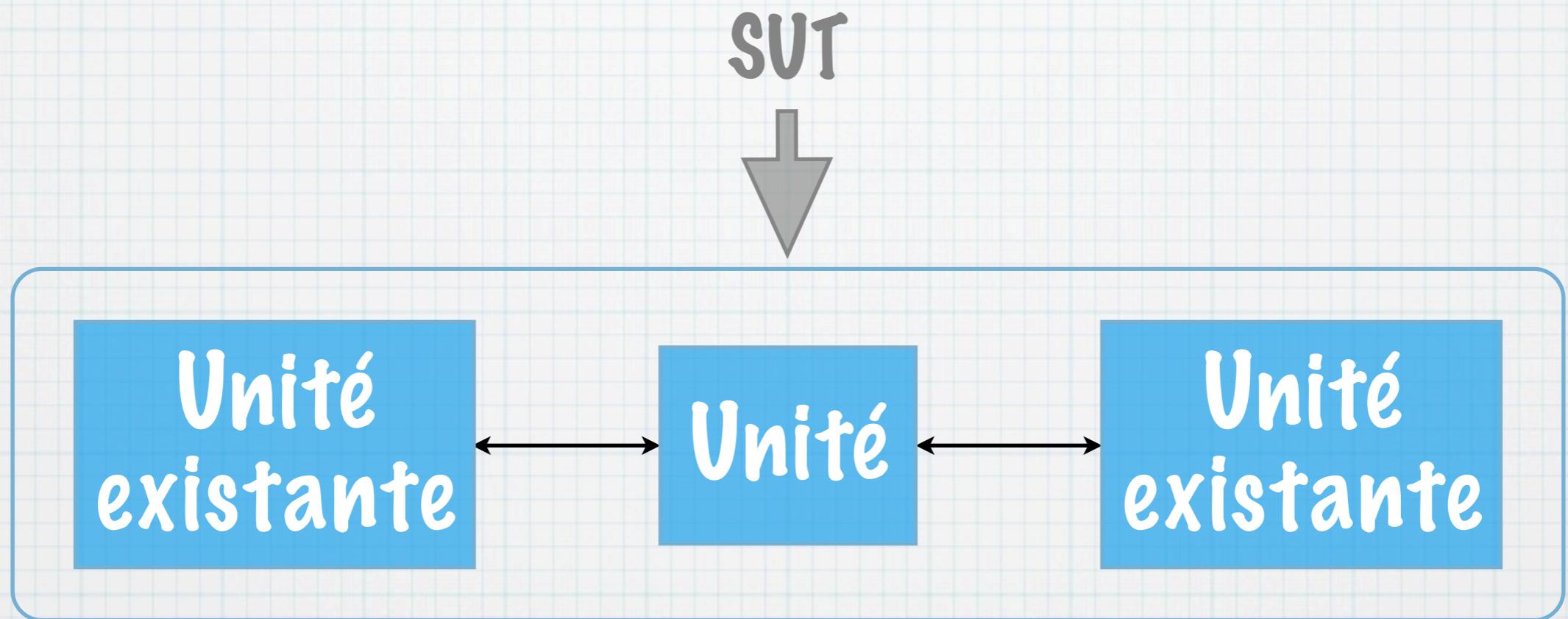


Différents niveaux de tests

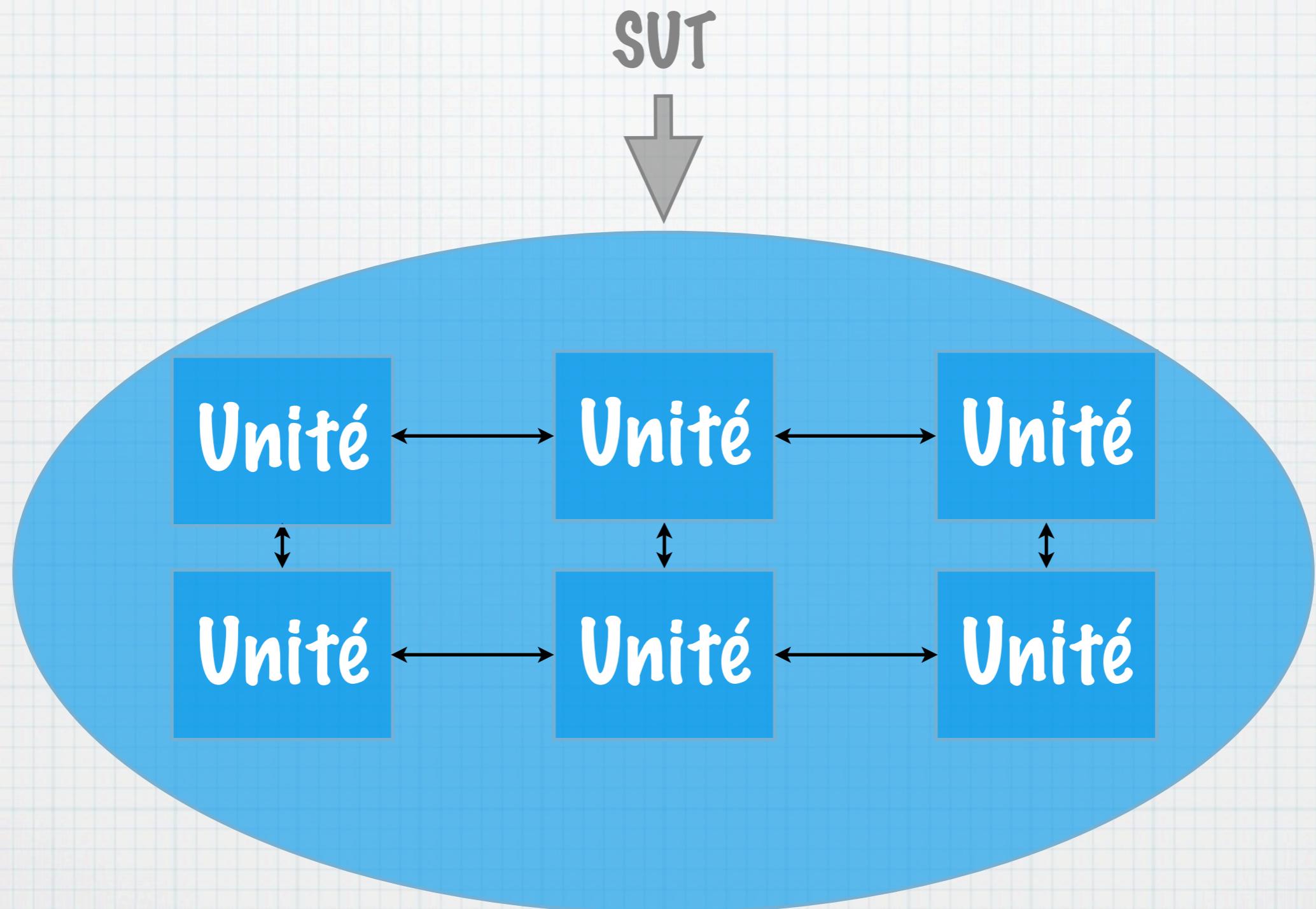
Test unitaire



Test d'intégration



Tests fonctionnels (tests d'IHM)



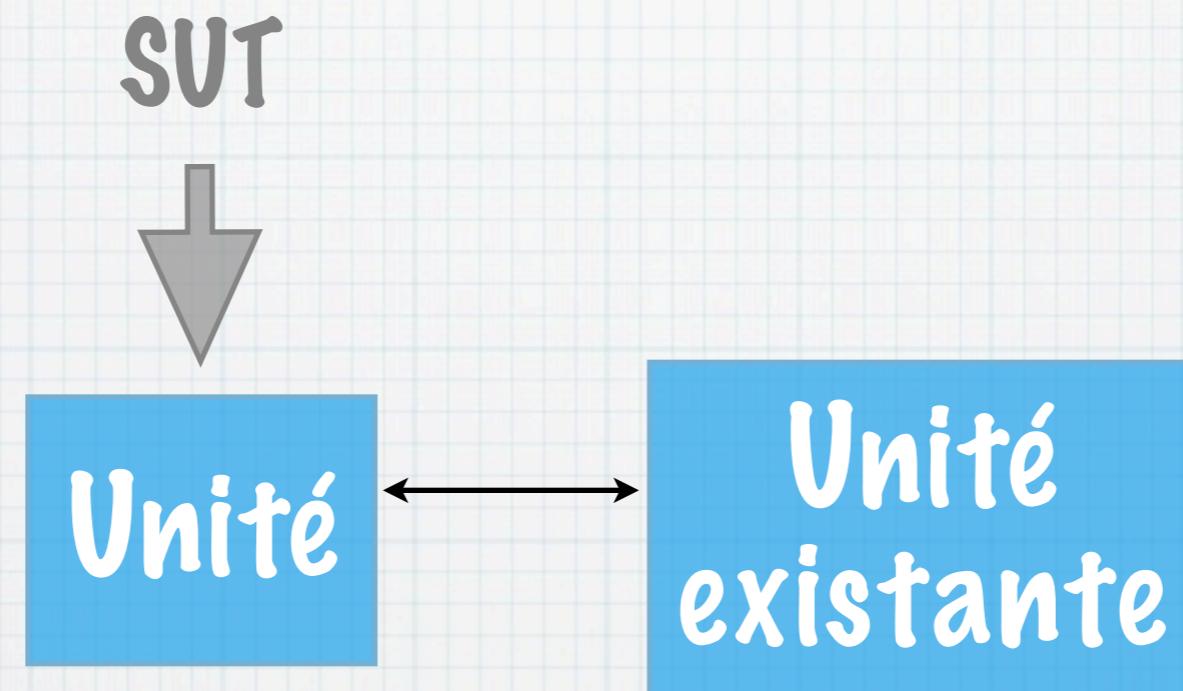
Les niveaux sont complémentaires

- * Chacun de ces niveaux permet de tester le produit final...
- * Mais il faut les voir comme des outils complémentaires.
- * Si un test unitaire ne passe pas, il ne sert à rien de lancer les tests d'intégration ni fonctionnels, etc...

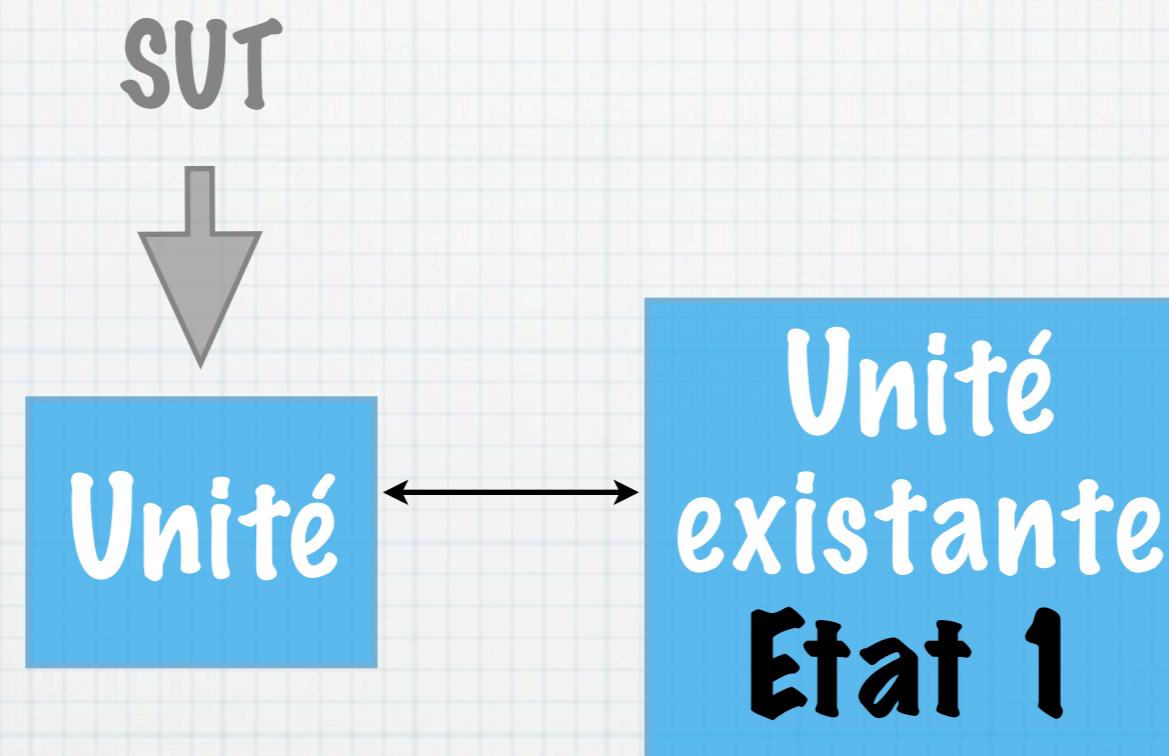
Caractéristique des tests

- * Résultat déterministe, non aléatoire.
- * Au moins une suite de tests «rapide» à exécuter.

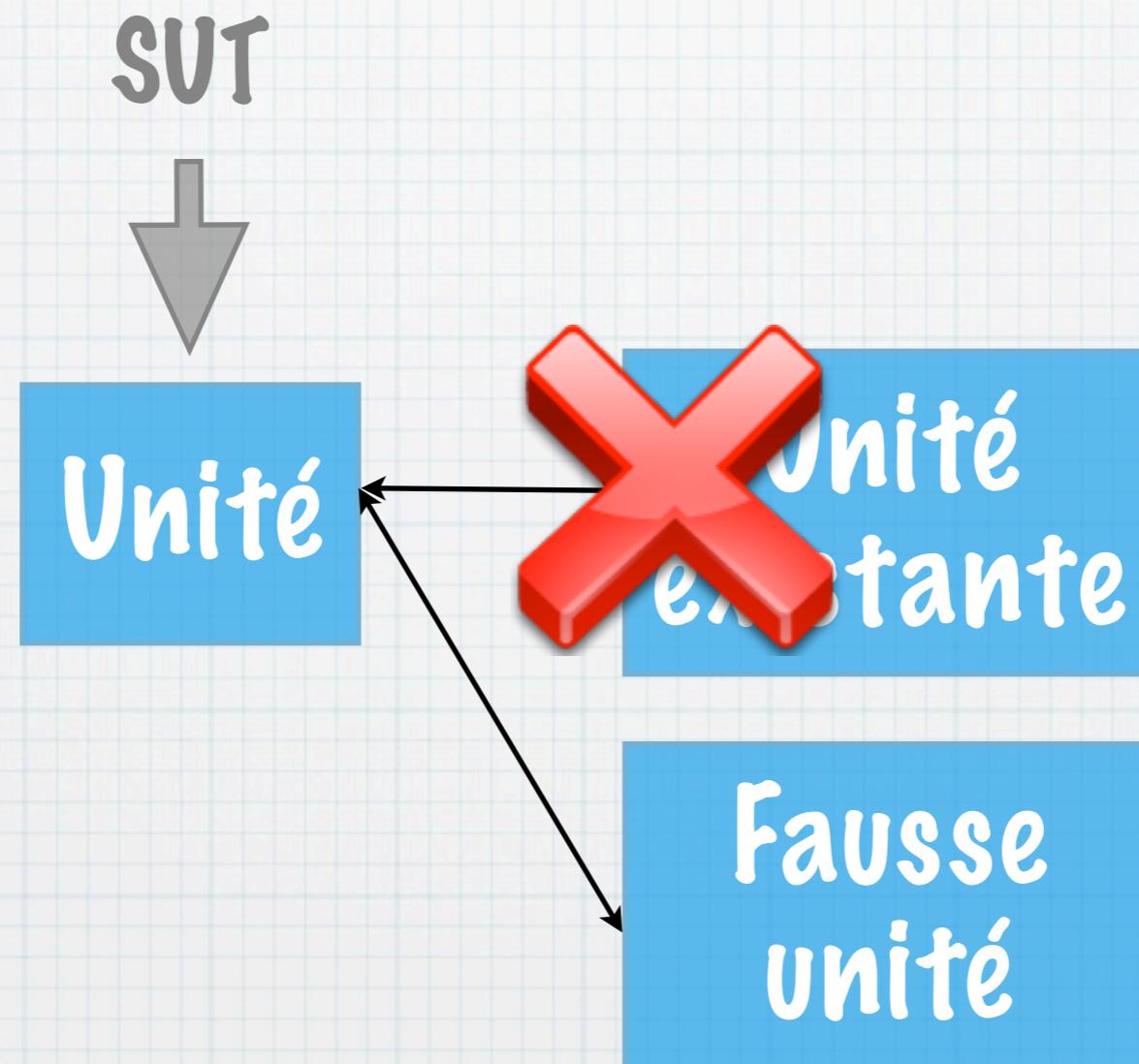
Tester les interactions



La manière «classique»



On remplace !



Pourquoi remplacer une unité ?

- * Remplacer un comportement non déterministe (l'heure ou la température ambiante).
- * Si l'unité a des états difficiles à reproduire (une erreur de réseau par exemple).
- * Si l'initialisation de l'unité est longue (ex : création d'une base de données).
- * Si l'unité n'existe pas ou si son comportement peut encore changer.
- * S'il est nécessaire d'inclure des attributs et des méthodes uniquement à des fins de test.
- * Pour isoler les tests. Pour éviter qu'un bug introduit dans une unité fasse échouer toute la suite de tests.

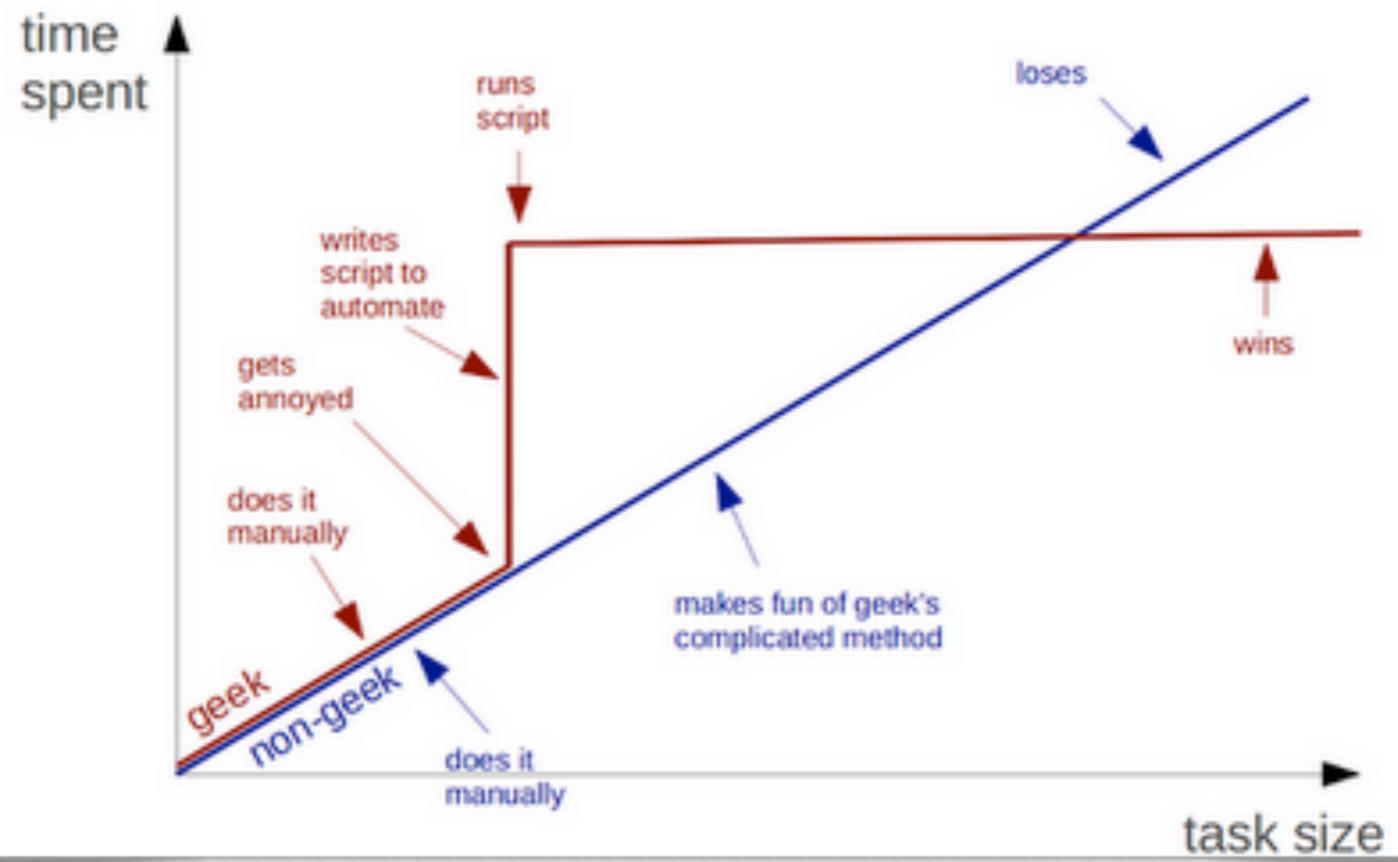
On remplace !

- * Cela consiste à utiliser des **mocks**, des objets avec des réponses prédéfinies.
- * Ainsi il est plus simple de «simuler» des erreurs, des cas difficiles à configurer, ...

Quand faire des tests ?

- * Idéalement, le plus tôt possible (voir avant le code cf. TDD).
- * Plus un problème est détecté tard, plus il sera coûteux à corriger.
- * L'un des problème des tests est qu'il demande un certain investissement.

Geeks and repetitive tasks



L'investissement à long-terme...

ET à court-terme

- * Plusieurs moyens existent pour «profiter» immédiatement des tests:
 - * Lors de la correction des bugs. 1 Bug = 1 Test
 - * Lors qu'un développeur travaille sur une partie inconnue du code. D'abord il devrait écrire des tests.

Les indicateurs donnés par les tests

- * Le nombre de tests qui passent/ne passent pas.
- * Le taux de couverture de code (= quelle proportion du code a été exécuté par les tests).

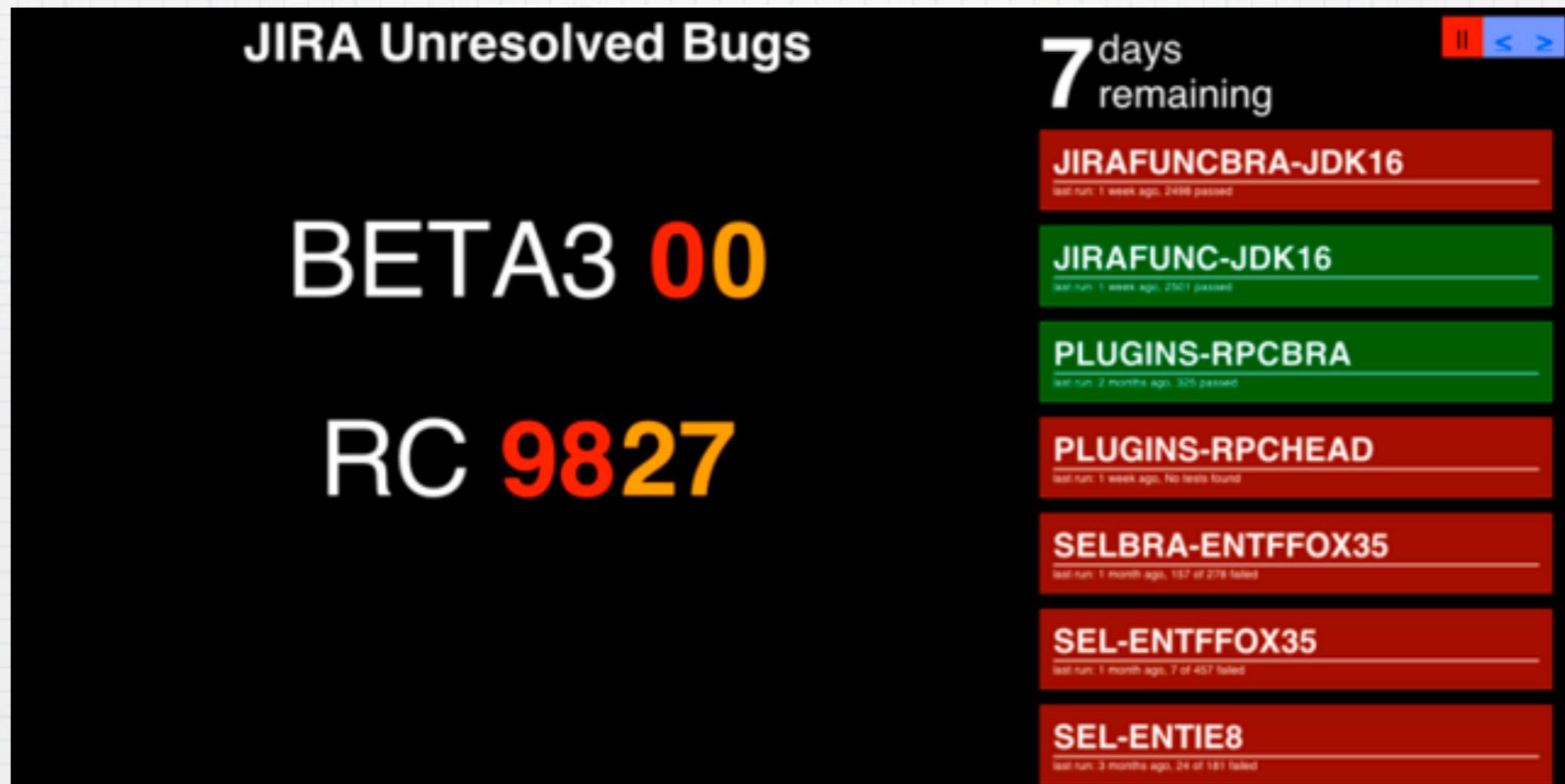
Avoir les bons outils !

- * Librairies de tests:
 - * PHP: PHPUnit, atoum...
 - * Python: Unittest, pytest...
- * Pour les tests d'interface web:
 - * Selenium est la référence

Bonnes pratiques

- * Utiliser un outil d'intégration continue.
- * Utiliser les indicateurs de tests dans le processus de déploiement.
- * Partager les résultats des builds...

Et il y a le choix



Pour aller plus loin

Test Driven Development

Et non l'inverse !



Conclusion

* Faîtes des tests ! Même si il est impossible de tout tester, quelques tests vous simplifieront la vie !

Et n'oubliez pas !

«Aujourd'hui, la programmation est devenue une course entre le développeur, qui s'efforce de produire de meilleures applications à l'épreuve des imbéciles et l'univers, qui s'efforce de produire de meilleurs imbéciles. Pour l'instant, l'univers a une bonne longueur d'avance.»

- Rich Cook