

# Tests unitaires - unit tests. Pourquoi tester?



**Debugging Sucks!**



**Testing Rocks!**

Ovidiu Ciule  
15 Septembre 2008

# Roadmap

1. Introduction
2. XP/TDD
3. Types
4. Pourquoi utiliser des unit tests ?
5. Conclusion, questions, discussion

# 1. Introduction - Définition

Unit test = test d'une unité

Unité = la partie la plus petite qu'on peut  
distinguer

OOP = méthode

# 1. Introduction - Exemple

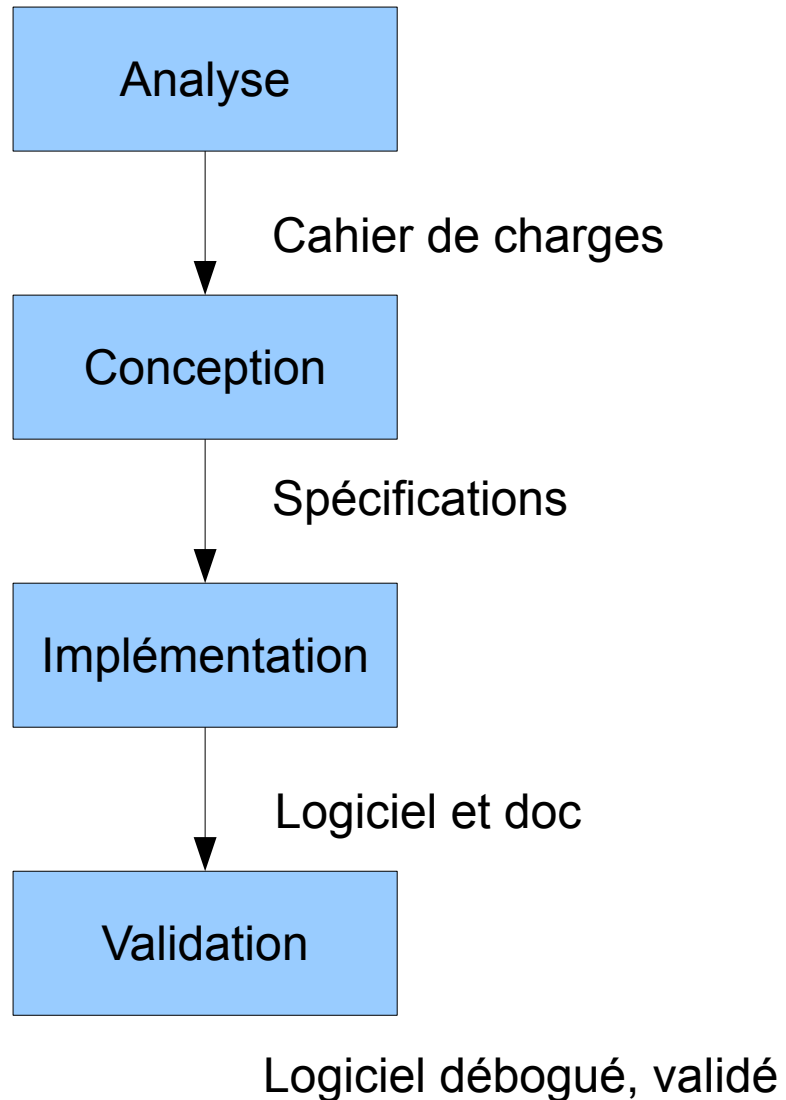
Avec SimpleTest:

```
function testDelete() {  
    p = new BlogPost();  
    p.save();  
    p.delete();  
    pp = BlogPost::getByKey(p.id);  
    this.assertNull(pp);  
}
```

## 2. XP/TDD

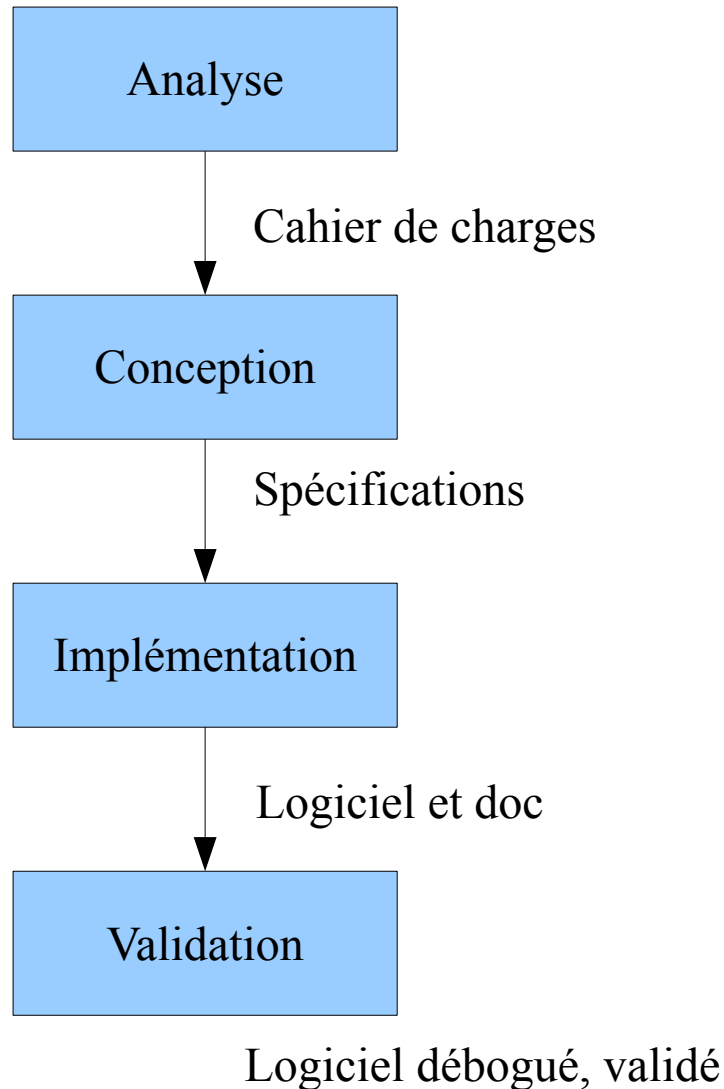
La méthodologie XP (Extreme Programming) utilise TDD (Test Driven Development) – Développement Dirigé par les Tests

## 2. XP/TDD - TDD

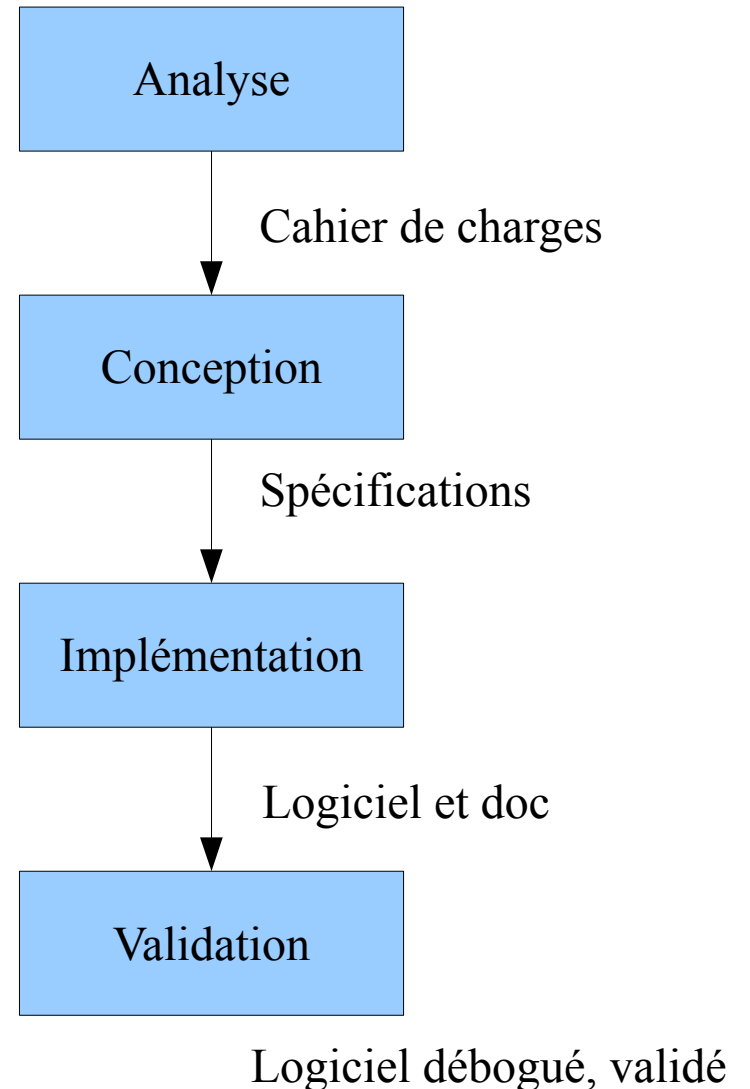


Méthodologie classique - Waterfall

## 2. XP/TDD - TDD

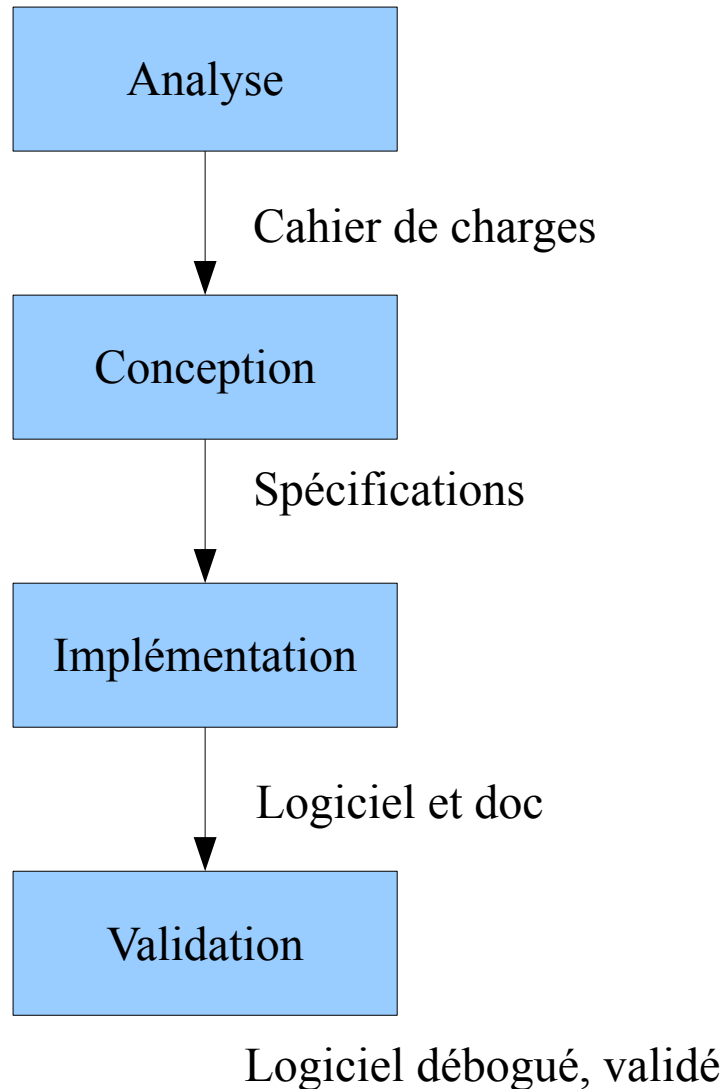


Méthodologie classique - Waterfall

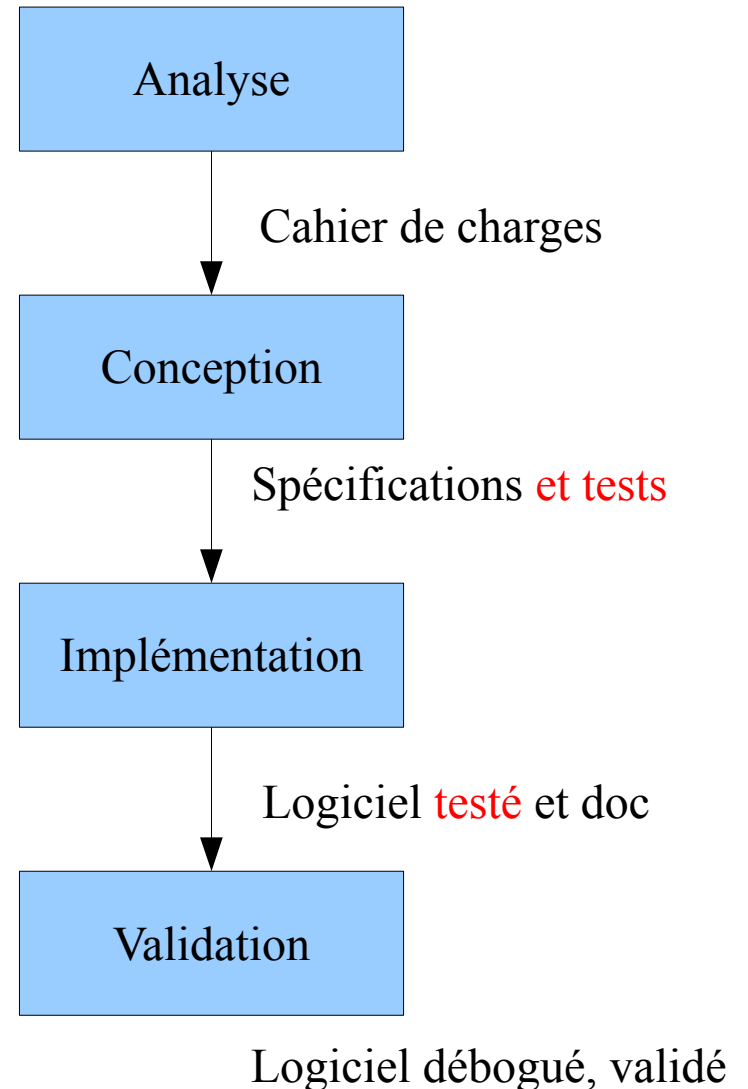


TDD

## 2. XP/TDD - TDD



Méthodologie classique - Waterfall



TDD

## 2. XP/TDD - TDD

On crée les tests avant d'avoir quoi tester ?!?

## 2. XP/TDD - TDD

On crée les tests avant d'avoir quoi tester ?!?

Oui. Tests = formalisation en code des specs.  
(donc besoin des specs détaillées)

## 2. XP/TDD - TDD

Les tests testent pas seulement l'implémentation  
(le code) mais aussi les specs

On s'assure que les specs sont suffisamment  
détaillées et qu'elles ne contiennent pas d'erreurs  
(contradictions, omissions)

# 3. Types de tests

Black box / white box  
Automatisées / manuelles  
Déterministe / non-déterministe

# 4. Pourquoi utiliser des unit tests ?

Mais pourquoi ne pas en utiliser?

Raisons de cout:

Cout de temps d'implémentation

Pas d'utilité directe – donc pas de valeur évidente  
pour le client (sans expérience)

Pas d'avance visible – décourageant

# 4. Pourquoi utiliser des unit tests ?

Mais pourquoi ne pas en utiliser?

Raisons d'efficacité:

Les tests ne pourront jamais valider toutes les chemins d'exécution, sauf pour le code trivial, et même pas:

```
function add(a, b) :  
    return a + b
```

Si a et b sont des int 32 bits,  $2^{64}$  chemins

# 4. Pourquoi utiliser des unit tests ?

Mais pourquoi ne pas en utiliser?

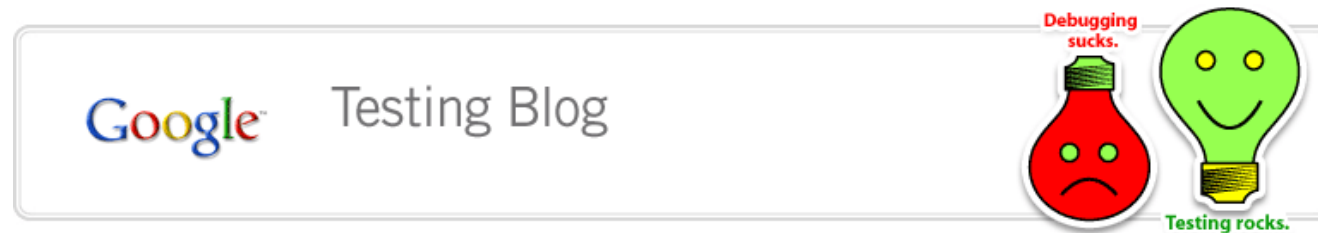
Raisons d'efficacité:

Ne valident pas des aspects fonctionnels –  
ergonomie, performance

Les tests ne pourront pas valider le système –  
faux argument

# 4. Enfin, pourquoi utiliser des unit tests ?

Parce que des gens intelligents ont décidé que c'est une Bonne Chose



# 4. Enfin, pourquoi utiliser des unit tests ?

Les unit tests = système de sécurité



# 4. Enfin, pourquoi utiliser des unit tests ?

« T'as qu'a esquiver les objets qui tombent. »



# 4. Enfin, pourquoi utiliser des unit tests ?

« Tu tomberas pas si tu fais un peu gaffe. »



# 4. Enfin, pourquoi utiliser des unit tests ?

« T'as qu'a éviter les autres voitures. »



# 4. Enfin, pourquoi utiliser des unit tests ?

« T'as qu'a plus faire des bugs, et on va laisser tomber les tests. Le projet sera sauvé. »



# 4. Enfin, pourquoi utiliser des unit tests ?

Pas convaincu de l'inévitabilité des erreurs ?

# 4. Enfin, pourquoi utiliser des unit tests ?

Pas convaincu de l'inévitabilité des erreurs ?

Moyenne de l'industrie : un bug toutes les dix lignes de code

# 4. Enfin, pourquoi utiliser des unit tests ?

Pas convaincu de l'inévitabilité des erreurs ?

Moyenne de l'industrie : un bug créé toutes les dix lignes de code

Pas de variabilité en fonction du langage, expérience etc.

# 4. Enfin, pourquoi utiliser des unit tests ?

Ne pas en faire = fausse économie



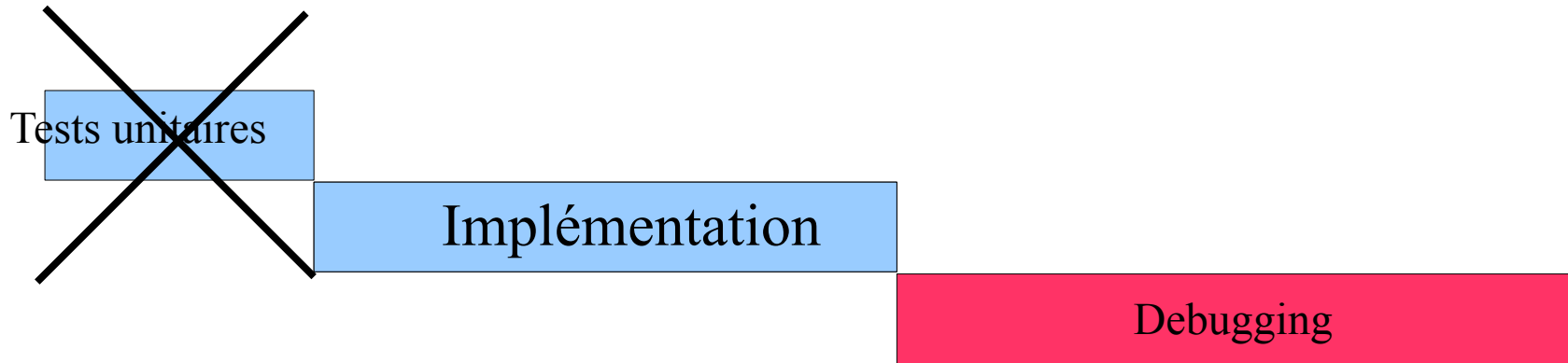
# 4. Enfin, pourquoi utiliser des unit tests ?

Ne pas en faire = fausse économie



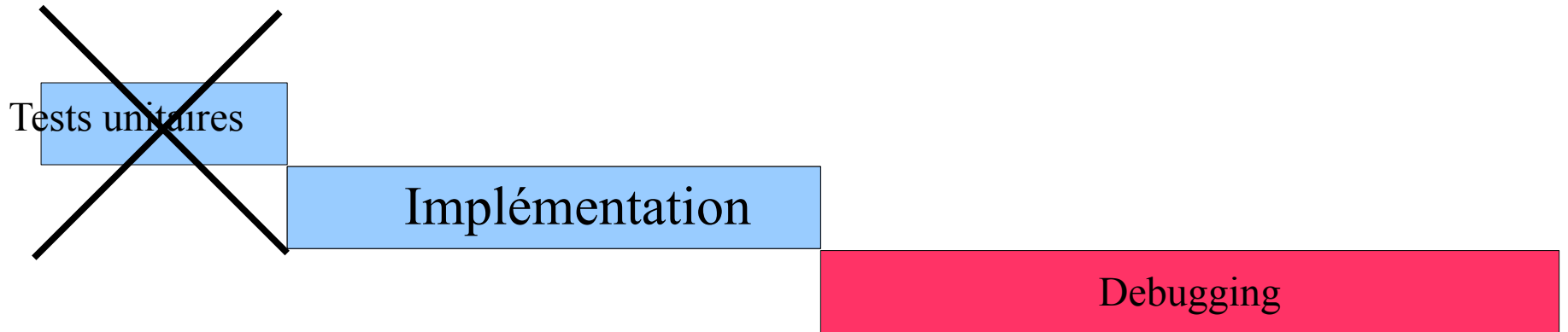
# 4. Enfin, pourquoi utiliser des unit tests ?

Ne pas en faire = fausse économie



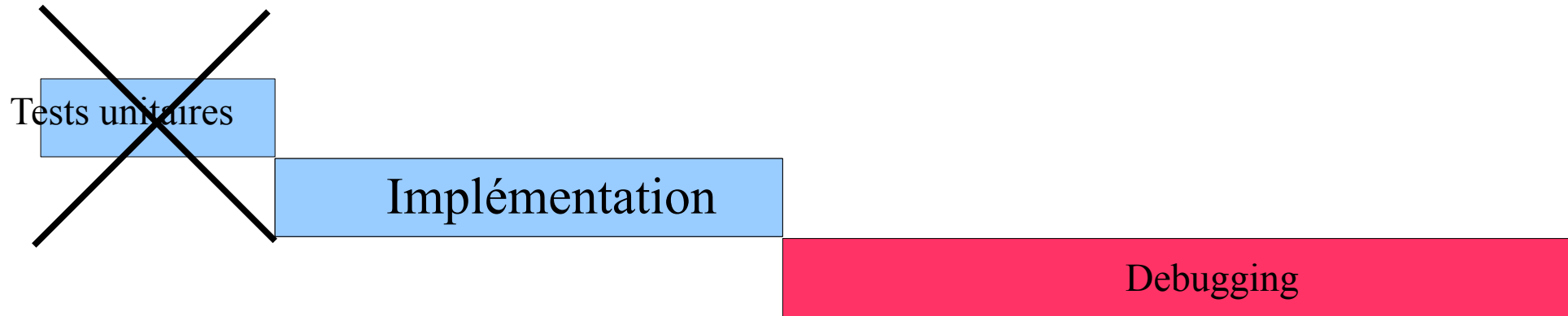
# 4. Enfin, pourquoi utiliser des unit tests ?

Ne pas en faire = fausse économie



# 4. Enfin, pourquoi utiliser des unit tests ?

Ne pas en faire = fausse économie



# 4. Enfin, pourquoi utiliser des unit tests ?

« Mais cette fonctionnalité je connais, j'ai déjà fait ... »

# 4. Enfin, pourquoi utiliser des unit tests ?

« Mais cette fonctionnalité je connais, j'ai déjà fait ... »

Et la première fois, pourquoi tu n'as pas testé ?  
Les tests sont facilement distribuables avec le code – et réutilisables

## 4. Enfin, pourquoi utiliser des unit tests ?

Si on réutilise beaucoup des fois le code, le cout d'implémentation des tests est divisé par le nombre de réutilisations

## 4. Enfin, pourquoi utiliser des unit tests ?

Si on réutilise beaucoup des fois le code, le cout d'implémentation des tests est divisé par le nombre de réutilisations

Pour un framework comme Turbulences, l'avantage est évident

# 6. Conclusion

Combien de lignes de code vous allez écrire  
aujourd'hui?

Merci pour les bugs!