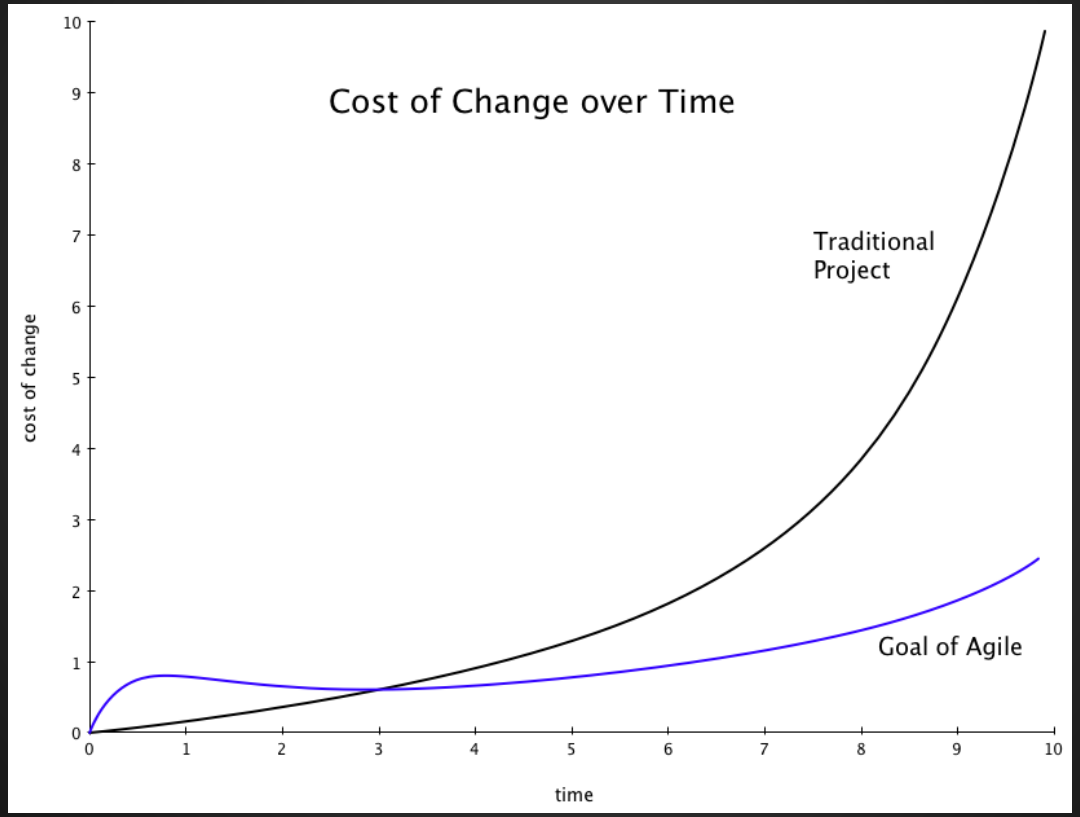
Qualité du code

La dure réalité

Qualité logicielle



Qualité Logicielle

- Le code propre c’est quoi ?
  - Passe les tests
  - N’est pas redondant
  - Exprime les idées de conception
  - Limite le nombre d’entités
- Première loi de la programmation
  - “Baisser la qualité augmente le temps de développement”
  - (Contrairement au dogme managérial bien connu)

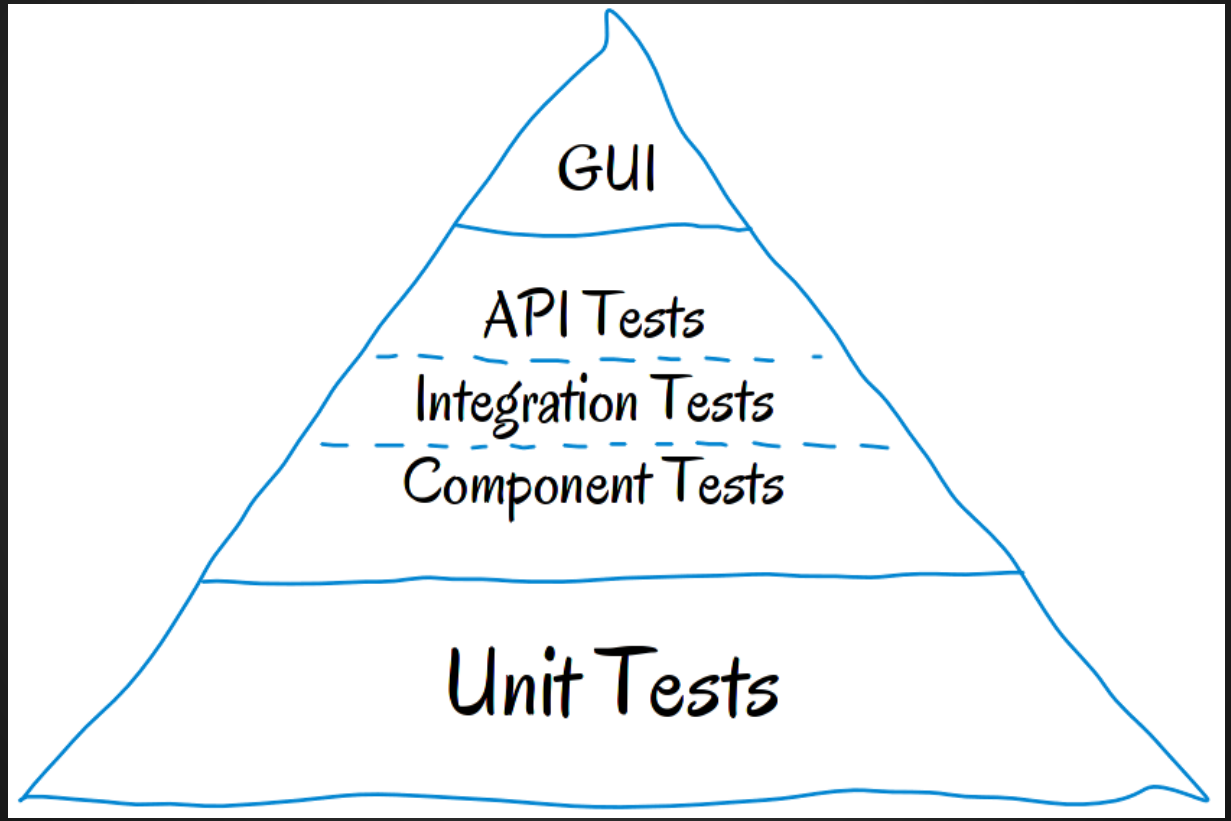
Passe les tests

- Tests = feedback
  - Savoir si ça marche ou pas
  - Si ça marche pas : comment réparer ?
- Tests = design
  - Dur à tester = mal fait
  - Dur à tester ~ pas testé ~ marche pas

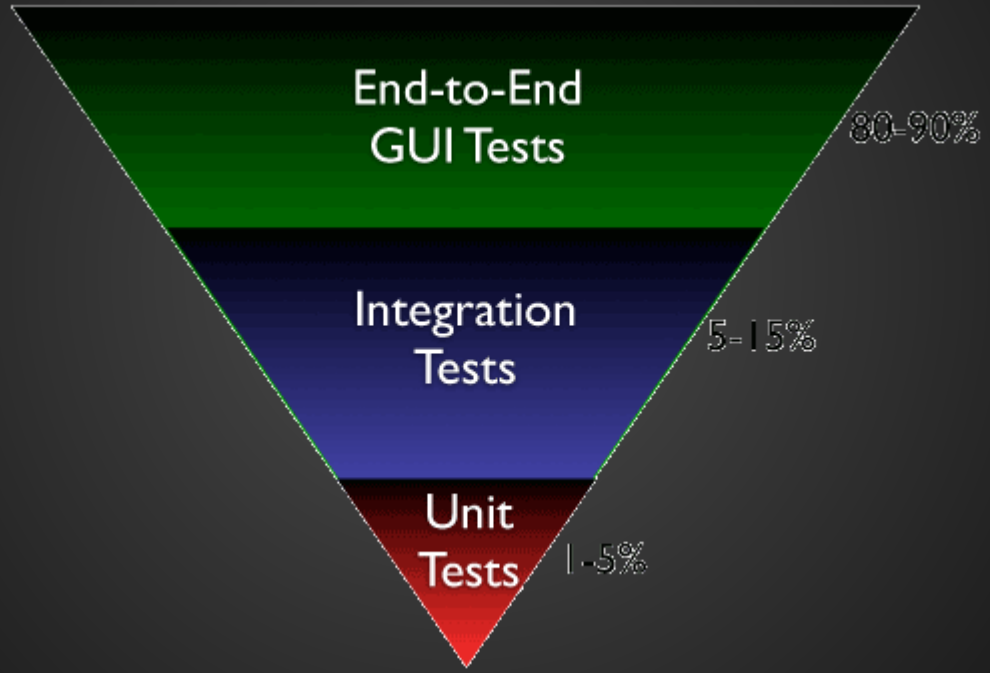
Passe les tests

- Combien de tests ?
  - Assez pour avoir confiance dans l’application
  - Soyons paranoïaques
- Quels tests ?
  - techniques, fonctionnels, de performance,...?
  - ceux qu’il faut pour avoir confiance ?
- Quelle fréquence
  - le plus fréquemment possible

Passe les tests



Passe les tests



Passe les tests

- Dur à tester == défaut de compétence
  - Soit on sait pas écrire des tests
  - Soit on sait pas écrire de code de prod
- On sait pas écrire les tests :
  - Framework de tests unitaires
  - Mock (framework ou pas)
  - Injection de dépendances

Passe les tests

- On sait pas écrire de code de prod
  - Apprendre à identifier les erreurs de conception
  - Apprendre les patterns pour les éviter

Pas redondant

- Pas. De. Copier. Coller.
  - Mutualisation de code
  - Copier du code = copier des bugs.
- Le meilleur code est celui qu’on écrit pas
  - Moins de code, moins de bugs

Pas redondant

- Commentaires

```
int i = 10; // affecte 4 à i
// Effectue un Get sur une url et renvoie le status
func Get(url string) int {
    // FIXME : ça marche pas (t'as qu'a corriger)
    // TODO : gérer les erreurs (t'as qu'a le faire)
    ...
    return status
}
```

Exprime la conception

- Nommage des entités

```
body := “http://perdu.com”
--
func Add3To(int value) int {
	return value * 3
}
```

- Plus un objet est utilisé loin de sa déclaration, plus son nom doit être précis.
- “Ce qui se conçoit bien s’exprime clairement et les mots pour le dire arrivent aisément.” J.Boileau (1674)
- Si c’est trop dur : problème de design.

Limite les entités

- “Entia non multiplicanda praeter necessitatem.” -- G. d’Occam (~1230)
- Orienter les choix de design vers la limitation des entités :
  - Réutilisation
  - Généralisation
  - Mutualisation

- Faire du code correct
- Qui exprime clairement la conception
- Sans se répéter
- Avec un design efficace

On fait comment ?

Amélioration de la qualité

- Identification de la non-qualité
- Pratiques de résorbtion de dette
- Revue de code
- Propriété collective du code
- Boy scout
- Bienveillance

Identification de la non-qualité

- Facile : code smells
  http://blog.codinghorror.com/code-smells/
- Indispensable
- À adapter en fonction du langage
- Vocabulaire de la non qualité
- Permet de communiquer entre nous

Résorbtion de dette

- Facile :
  - http://en.wikipedia.org/wiki/Code_refactoring
- Indispensable
- À adapter en fonction du langage
- Vocabulaire de l'amélioration de qualité
- Permet de communiquer entre nous

Revues de code

- Faciles si on connaît 1) et 2)
  - Parce qu'on a le vocabulaire
  - Parce qu'on a les techniques de correction
- Au moins une fois par itération
- A plusieurs
- Backlog de refactorings
- Etablissement de bonnes pratiques
- Item de la définition de terminé ?

Propriété collective

- “La propriété c’est le vol.” K. Marx
- Mon code est ton code, ton bug est mon bug.
- Tout le monde est responsable.

Boy Scout

“Toujours laisser le camp un peu plus propre qu’on l’a trouvé.”

Bienveillance, exemplarité

- “La bienveillance est la disposition affective d'une volonté qui vise le bien et le bonheur de chacun.”
- “Agis seulement d'après la maxime grâce à laquelle tu peux vouloir en même temps qu'elle devienne une loi universelle.” E.Kant
- “Ça n’est pas le chemin qui est difficile, c’est le difficile qui est le chemin” Id.

Bouquins

- Écrits par des gens qui ont plus d’expérience que nous
- “Coder proprement” -- Bob Martin
- “Code Complete” -- Steve McConnell
- “Design Patterns” -- GOF
- “Extreme Programming” -- Kent Beck
- ...