

Une architecture agile et testable

3 clés pour réussir l'automatisation des tests à l'échelle

\$ whoami

- Java, javascript, Clojure, C#, PHP
- Independant, joueur d'équipe
- Craftsmanship
- Passionné du code et design du soft
- Ouvert

`martinsson.johan@gmail.com`
`@johan_alps`

A retenir

- Ecouter les tests - Modeler pour testabilité
- Trois besoins distincts
 - Unit
 - Adapter
 - Application
- Prenez le contrôle

Ecouter les tests

- “Sometimes we find it difficult to write a test ... this usually means that our design can be improved” - S. Freeman, N. Pryce
- “The tests are a canary in coal mining revealing by their distress the presence of evil design vapors” - Kent Beck

Ecouter les tests

- Les tests nous disent “fais qq chose!”
- A nous de trouver
- Il y en a toujours
- Parfois nous ne connaissons pas encore





Carte

- **Problème**
- Pyramide des tests
- Adaptateurs + tests
- Tests d'application + Composition root
- Résultat flexible



TDD était top!

- Nombre romains, FizzBuzz!
- jusqu'à ce qu'on le fasse avec du vrai code.

On visait le ciel

- Tester l'ensemble
- Avec vraies dépendances
- De façon rapide et fiable
- Couvrir tout

... on atterit en enfer!

- Dommages collatéraux
 - Evolutions => faux négatifs
- Dépendances volatiles
 - Instabilité => faux négatifs
 - Lenteurs => huile sur le feu

Dommmages collatéraux

- Evolution données de test

- Nouvelle fonctionnalité



- Assertions trop strictes
`expect(resultFile).to.equal(referenceFile)`

Dépendances Volatiles

- Lents
- Difficile à initialiser
- Etat entre tests
- Erratic
- Debug-a-lot

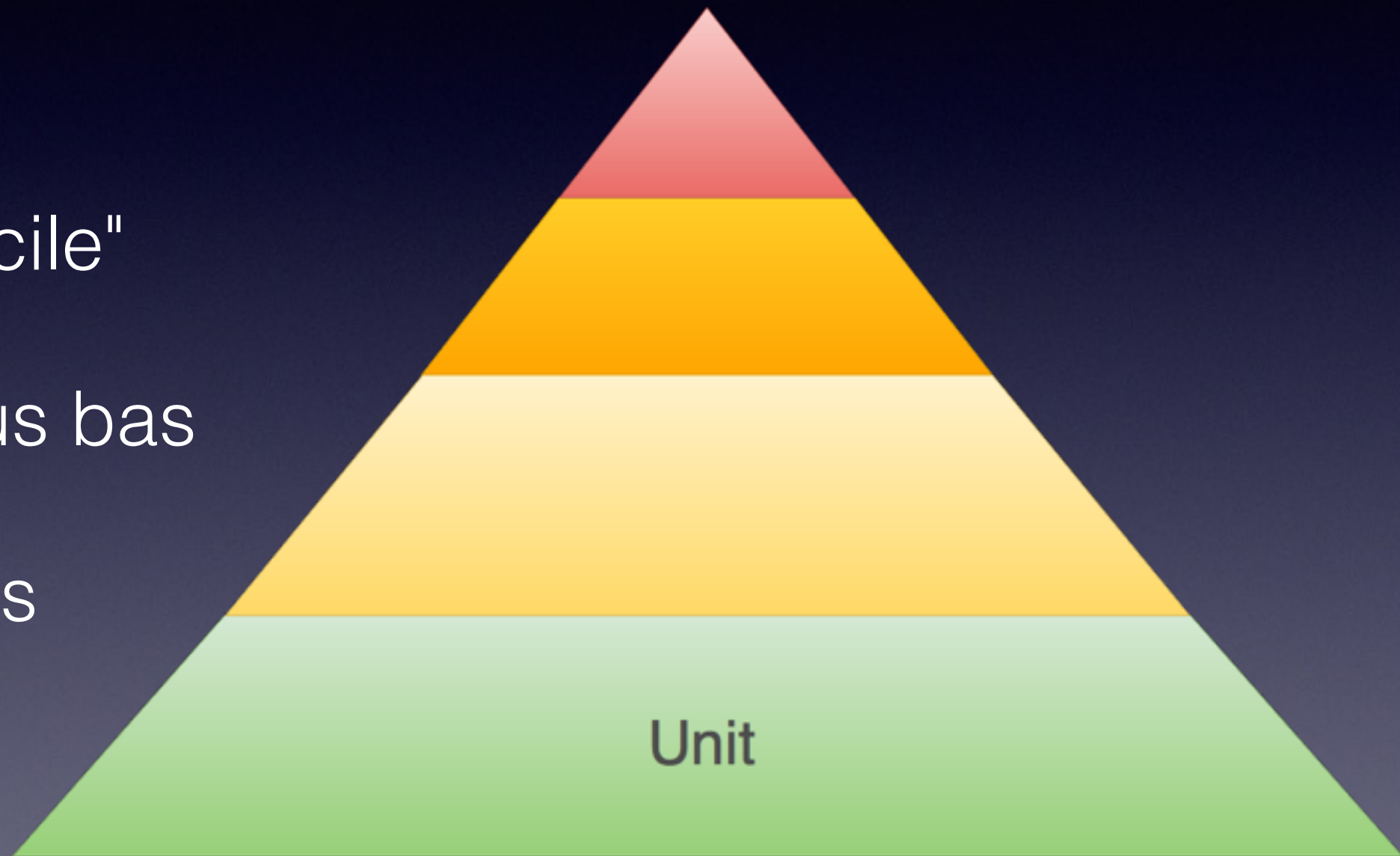
Carte



- Problème
- **Pyramide des tests**
- Adaptateurs + tests
- Tests d'application + Composition root
- Résultat flexible

Dommmages collatéraux

- Solution "facile"
- Tester plus bas
- Assertions précises



Tests unitaires - cout de maintenance constante!

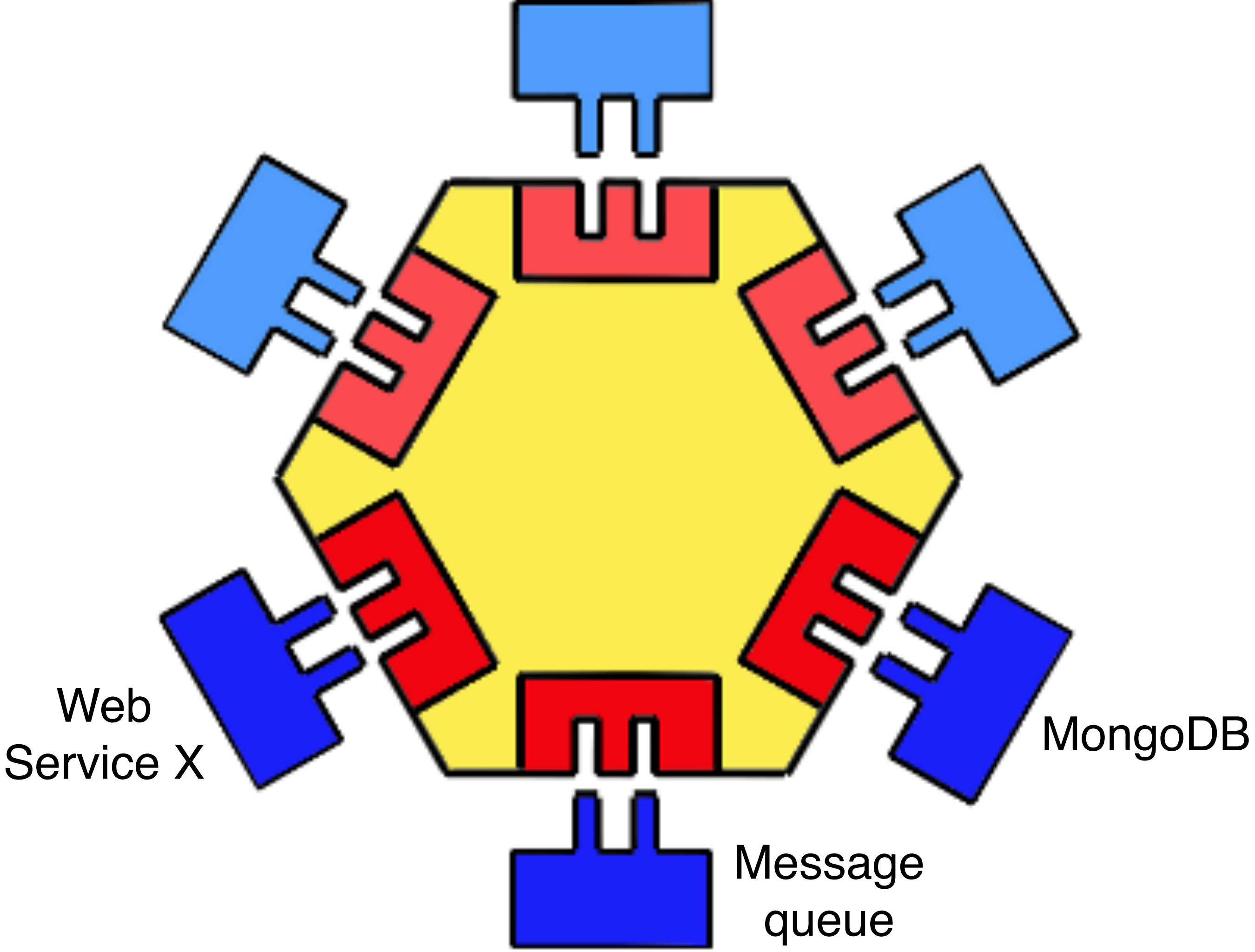
Carte

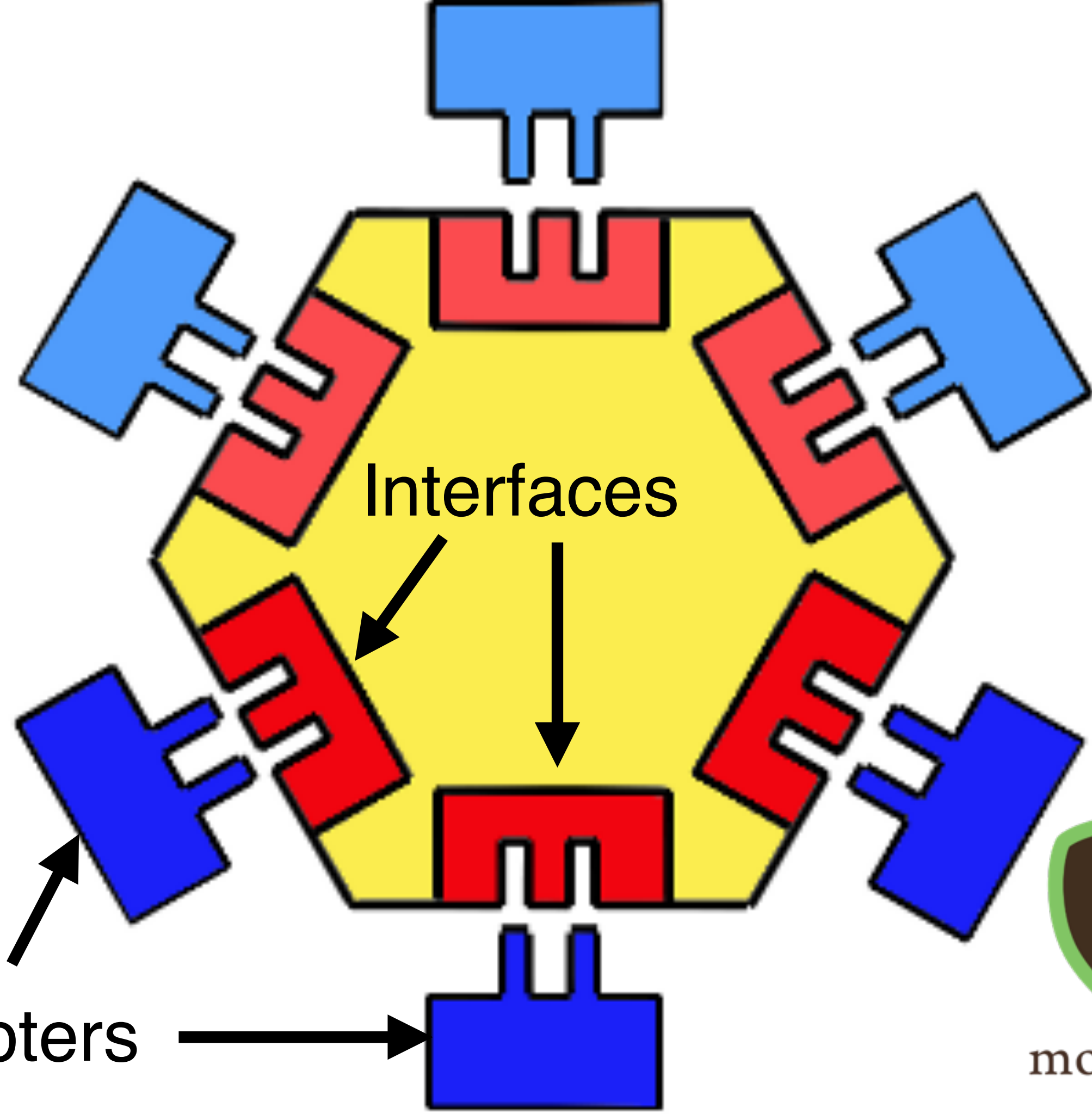


- Problème
- Pyramide des tests
- **Adaptateurs + tests**
- Tests d'application + Composition root
- Résultat flexible

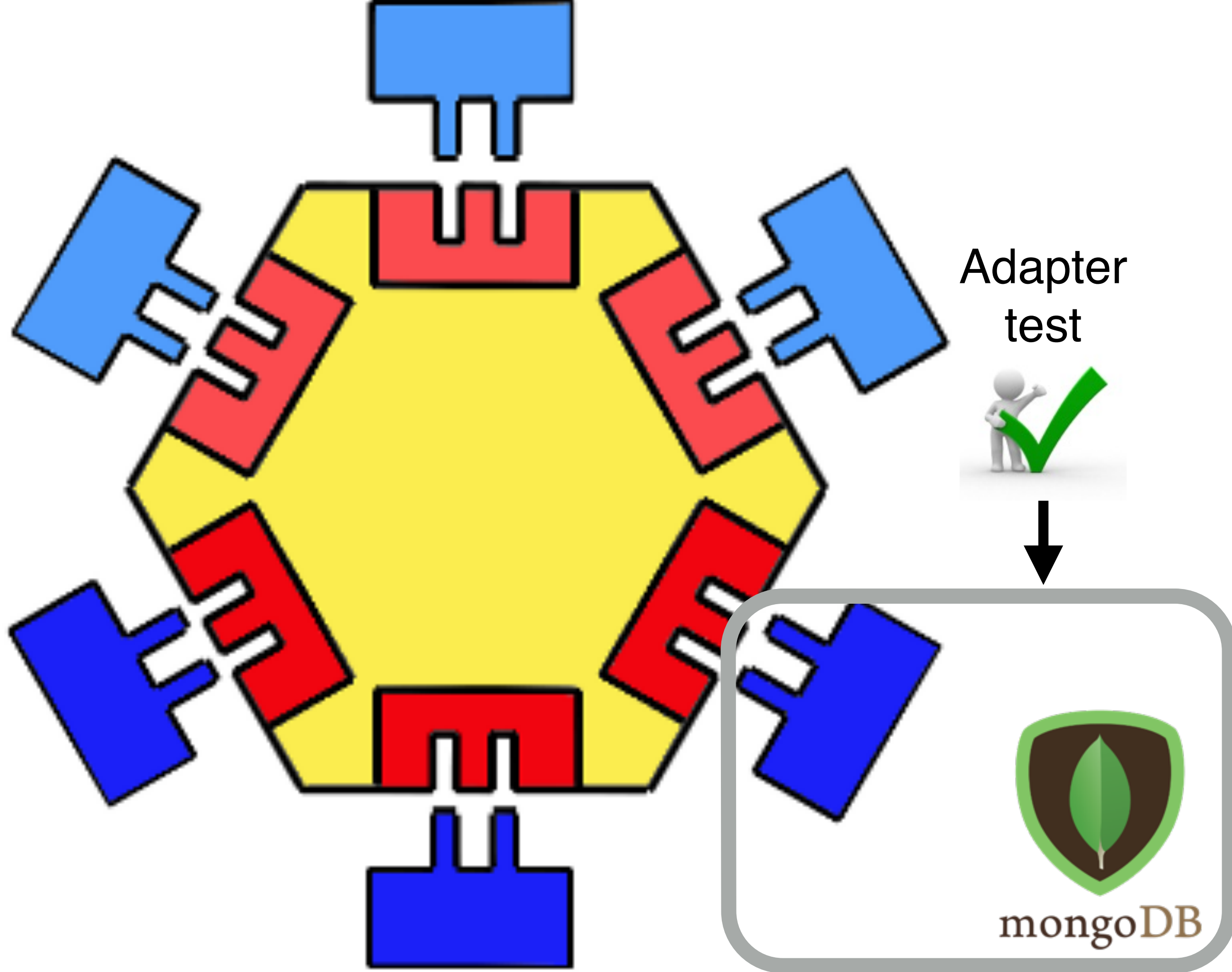
Même chose pour le dépendances volatiles

- Encapsuler avec une interface minimaliste - ADAPTER
- Tester à fond avec “la vraie”
- Stub/mock pour tous les autres tests





mongoDB



Avantages

- Adapteurs sont (plus) faciles à tester en isolation
- le reste est facile à tester.
- Le code est simplifié aussi

Inconvénients

- A court d'excuses! :)
- Pas de tests qui traversent la frontière

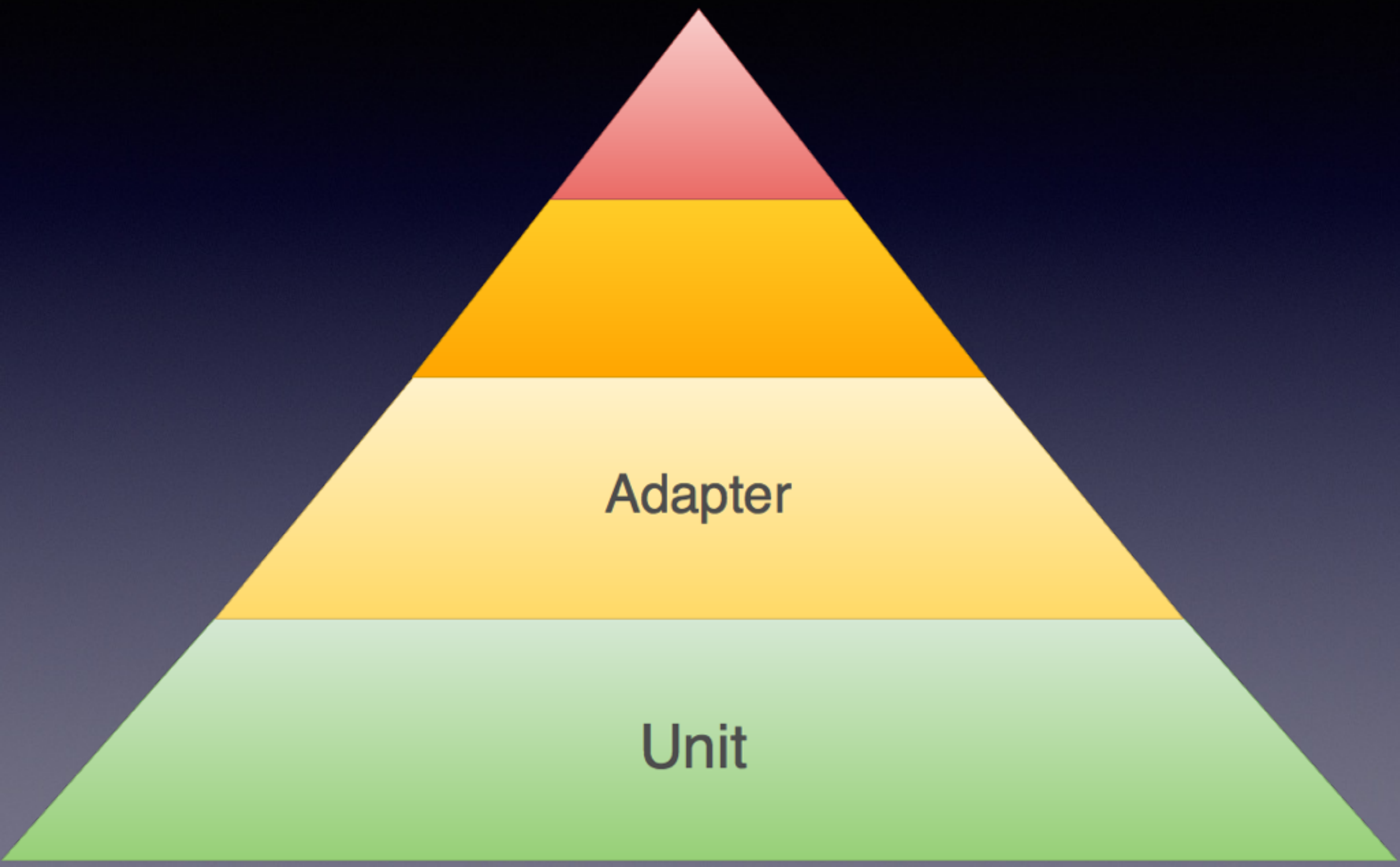
Mais encore

Changez les adaptateurs!

- Faites passer le test.
- Poussez en prod!

Une suite indépendante

- Vous saurez pourquoi vos tests systèmes plantent ... 10 min avant qu'ils tournent!
- Moins de debug
- Moins dépendant de bugs d'autres équipes
- Donnez-vos tests



Adapter

Unit

Résumons

- On a isolé la partie difficile
- Choisissez une stratégie pour le reste
 - Tests unitaires
(métier et tech)
 - Tests d'application / tests fonctionnels
(branchement OK? protection de gros refacto)

Carte



- Problème
- Pyramide des tests
- Adaptateurs + tests
- **Tests d'application + Composition root**
- Résultat flexible

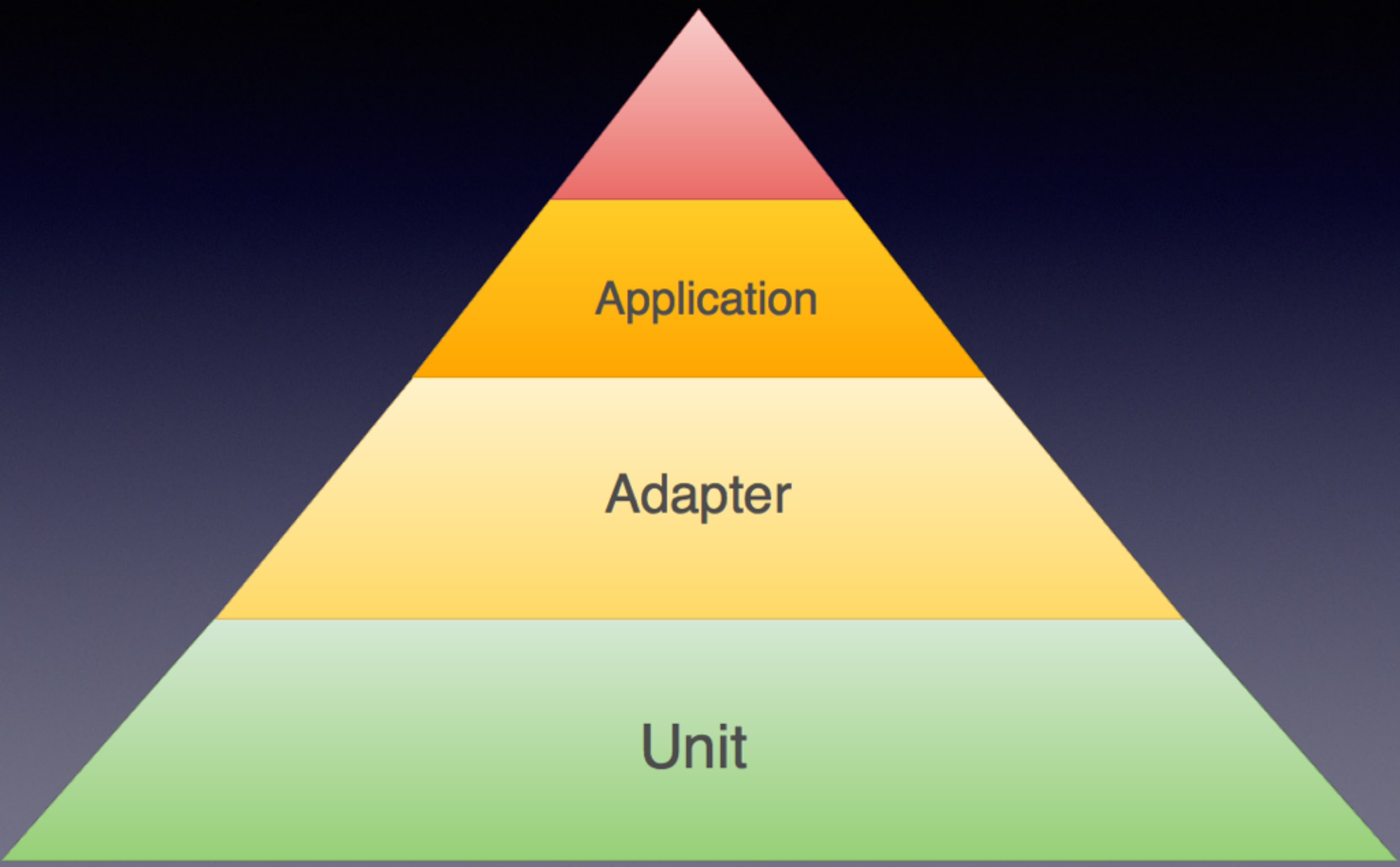
Tests d'application

- Isoler de toute Dépendance Volatile
- Facile à tester

Tests d'application

- Composition Root / Main module
- 2 phases
 1. Build hard-to-test objects
 2. Use those to build the rest:
`composeApplication(fakeDB, fakeWSCClient, tmpDir)`

Composition Root - Dependency Injection in .NET
Main module - Bob Martin



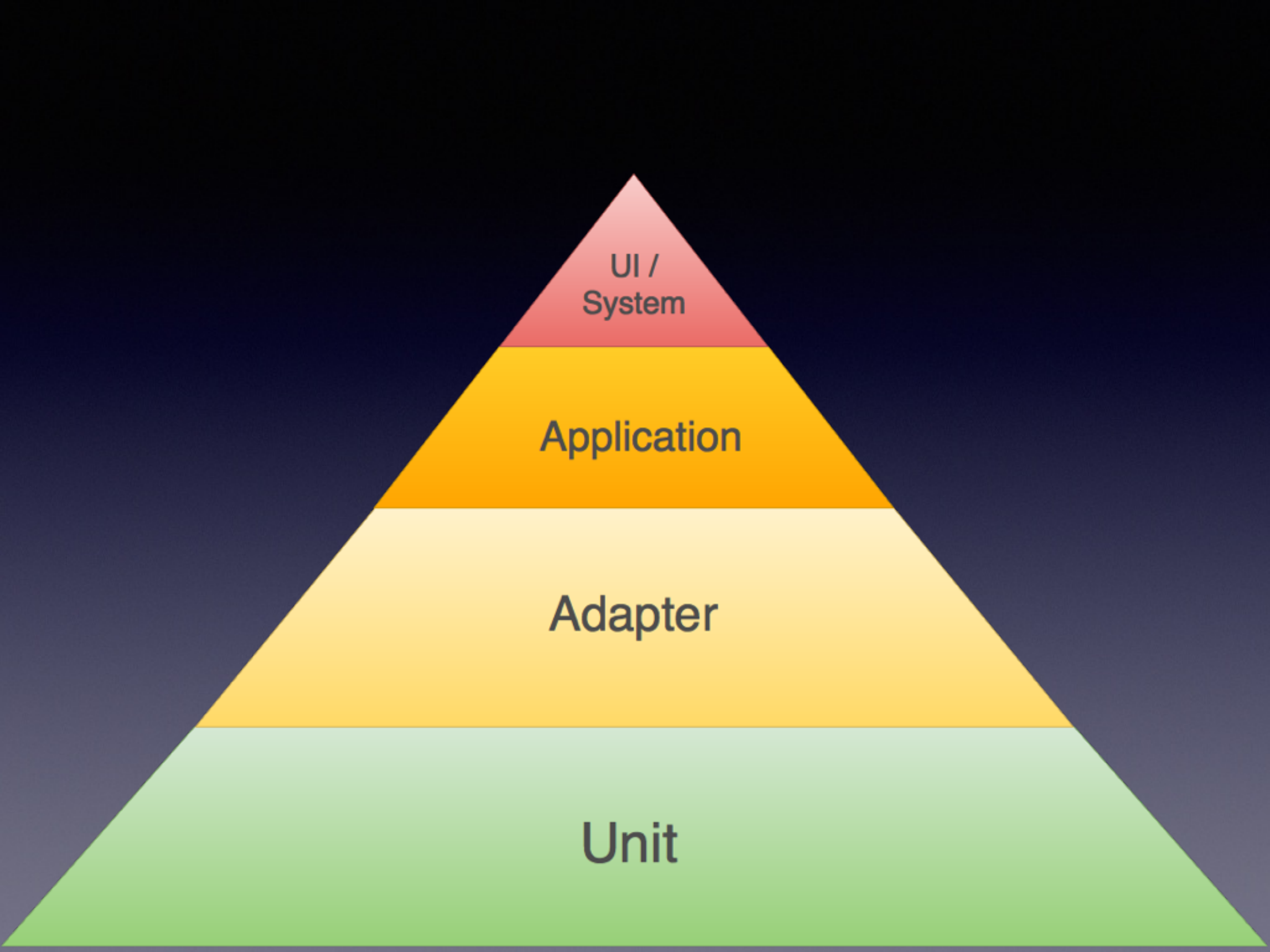
Carte



- Problème
- Pyramide des tests
- Adaptateurs + tests
- Tests d'application + Composition root
- **Résultat flexible**

Autre tests

- Tests End-to-end (frontière manquante)
- IHM
- Micro-services?
- Stress-, performance-tests => paramétrer tests existants



UI /
System

Application

Adapter

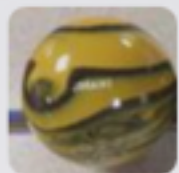
Unit

De quoi devons nous nous isoler

- Services web
- Bases de données (habituellement)
- Processes tierces
- Serveurs d'application
- IHMs
- Lenteurs (frameworks injection, attentes, ...)



Hans Sowa and 2 others Retweeted



J. B. Rainsberger @jbrains · Oct 22

Once you've written a test that depends on **Spring**, please put "Stop test from depending on **Spring**" in your backlog. [#geecon](#)



16



10



Conclusions

- Tester bas niveau
 - Unit & Adapter => stabilité, rapidité, isolation
- Tester => pression code/architecture
 - Application malléable
- Le reste est “facile” :)