



The Pragmatic Programmers

Investigator: _____
Date: _____
Case #: _____
Location: _____

Your Code as a Crime Scene

Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs

Pragmatic Design Quality Assessment

Tudor Gîrba
University of Bern, Switzerland

Michele Lanza
University of Lugano, Switzerland

Radu Marinescu
Politehnica University of Timisoara, Romania

[PDQA]

<https://fr.slideshare.net/girba/pragmatic-quality-assessment-tutorial-icse-2008>

Les **problèmes de conception** sont ...

Fréquents

Inévitables

Dispendieux

[PDQA]

Comment les éviter ?

On peut pas. Mais on peut
vivre avec et s'améliorer !

[PDQA]

1,000,000,000 de lignes de code

$\times 2s = 2,000,000,000$ secondes

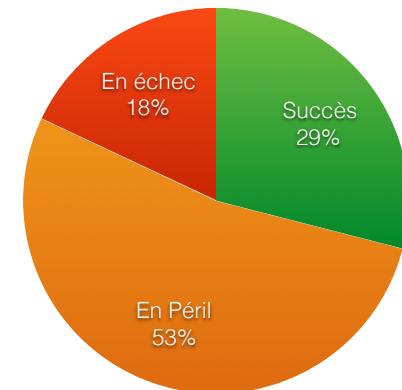
$/ 3600 = 560$ heures

$/ 8 = 70$ jours

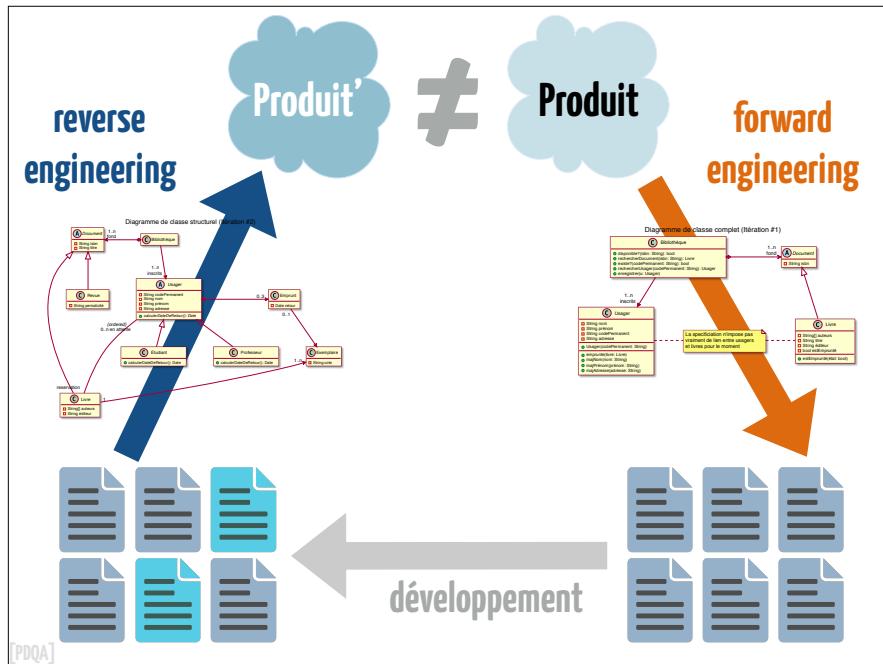
$/ 20 = 3$ mois !!

[PDQA]

Le logiciel, cet objet complexe



[The Stanford Group, 2004]



How the UML diagram
describes the software



<https://twitter.com/JobOpportunitiesIT/status/1106631344233295872?s=19>

How the code is
actually written



Un vrai système contient
beaucoup de détails !

Comment juger
de sa qualité ?

[PDQA]

Quantifier 1

You cannot **control**
what you cannot **measure**

[PDQA]

Tom de Marco

Métrique (définition)

Une **métrique** est une **fonction** qui **assigne un nombre** à un **produit**, un **processus** ou une **ressource**.

[PDQA]

Exemples de métriques Orientée-Objet

Métrique	Définition
NOM	Nombre de Méthodes
NOA	Nombres d'Attributs
LOC	Nombre de Lignes de Code
NOS	Nombre d'instructions
NOC	Nombre de classes enfant

[PDQA]

Métrique Logicielle (définition)

Les **métriques logicielles** sont des **mesures** associées à des **systèmes logiciels**, leurs **processus de développement** et les **documents associés**.

[PDQA]

Complexité Cyclomatique (CYCLO)

La **complexité cyclomatique** (ou **nombre de McCabe**) compte le **nombre de chemins indépendants à travers le code** d'une fonction

Indication sur le
nombre de tests à écrire

[PDQA]

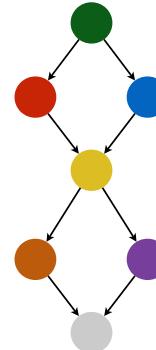
A part ça ...
pas grand chose

CYCLO(P) ?

P =

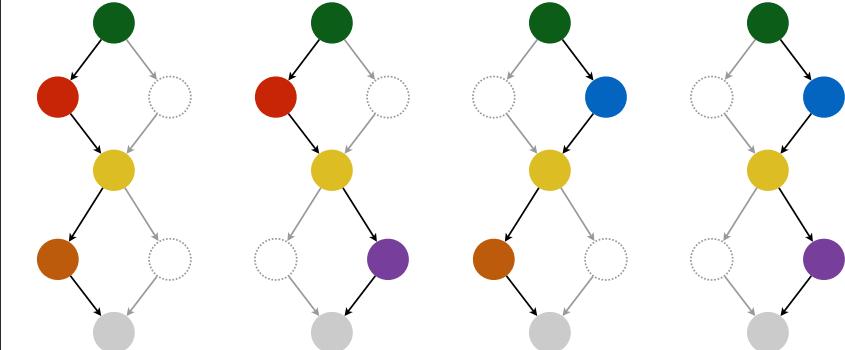
```
if( c1() )  
    f1();  
else  
    f2();  
  
if( c2() )  
    f3();  
else  
    f4();
```

≡



https://en.wikipedia.org/wiki/Cyclomatic_complexity

CYCLO(P) = 4



https://en.wikipedia.org/wiki/Cyclomatic_complexity

Méthodes pondérées (WMC)

“Weighted Method Count” (WMC) fait la somme pondérée des méthodes d’une classe (classiquement avec CYCLO pour pondérer)

Configurable, on peut changer la pondération au besoin

A part ça ...
pas grand chose

Profondeur d'héritage (DIT)

“Depth of Inheritance Tree” (DIT) est la profondeur maximale de l’arbre d’héritage pour une classe donnée.

Donne une quantification à la notion d’héritage

Comment l’interpréter ?

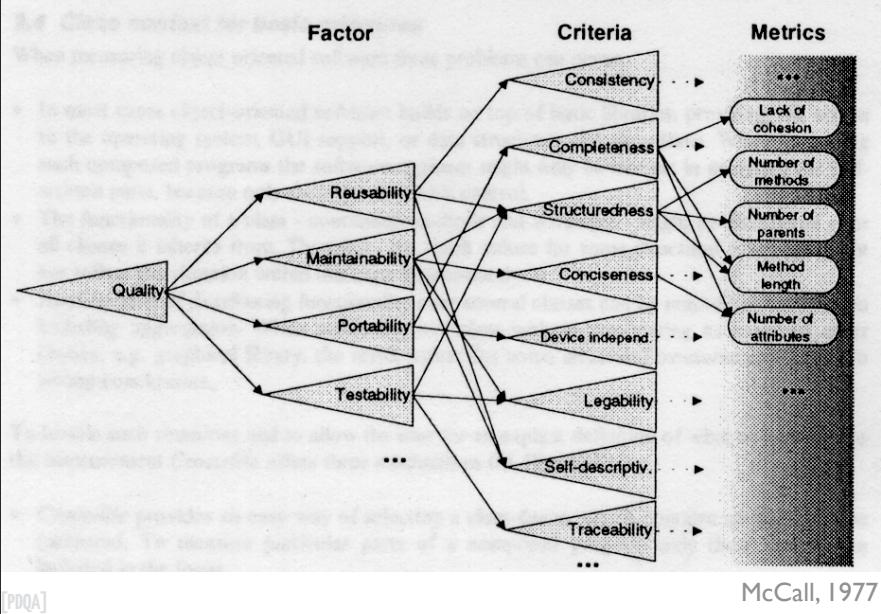
Couplage entre objets (CBO)

Mesure du couplage en identifiant les **méthodes et attributs utilisés depuis l'extérieur** de la classe.

Prend en compte
les vrais dépendances

Pas de mesure de
l'intensité du couplage

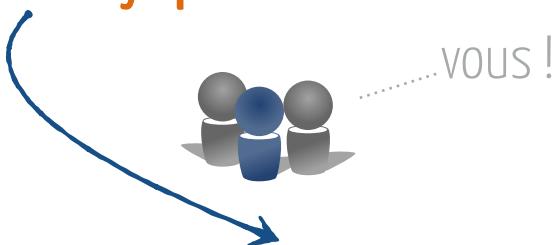
[PDQA]



McCall, 1977

Le problème #1 avec les métriques ...

Capture un symptôme



Comment en tirer le traitement ?

[PDQA]

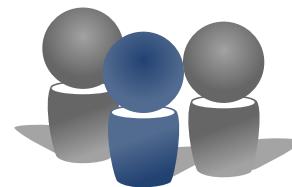
Le problème #2 avec les métriques ...

SOLID

GRASP

DRY

KISS



On raisonne sur des **principes**, pas des **métriques**

[PDQA]

Exploiter

2

Metric	Value
LOC	35175
NOM	3618
NOC	384
CYCLO	5579
NOP	19
CALLS	15128
FANOUT	8590
AHH	0.12
ANDC	0.31

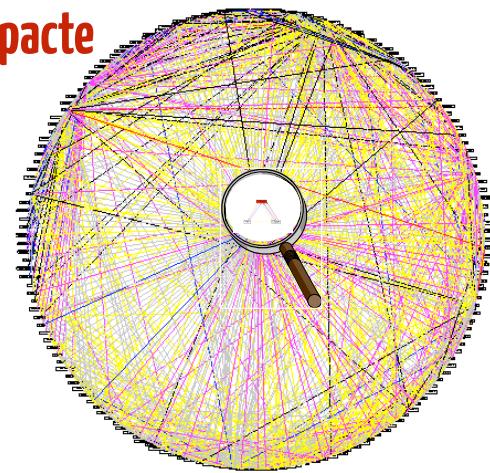
En quoi ça nous aide ?

En quoi ça nous aide ?

Si je change
juste cette méthode ...



Mais qu'en fait ça impacte
33% du code



C'est l'analyse conjointe des métriques
qui aide les développeurs.

Votre Premier Anti-Patron

Un **BLOB** (ou **classe Dieu**) tend à **centraliser toute l'intelligence** du système, à **tout faire** et à **utiliser** des données en provenance de **classes purement structurelle**



[PDQA]

Un **BLOB** :



- **centralise toute l'intelligence** du système;
- **fait tout**;
- **utilise** des données de **classes structurelle**.

[PDQA]

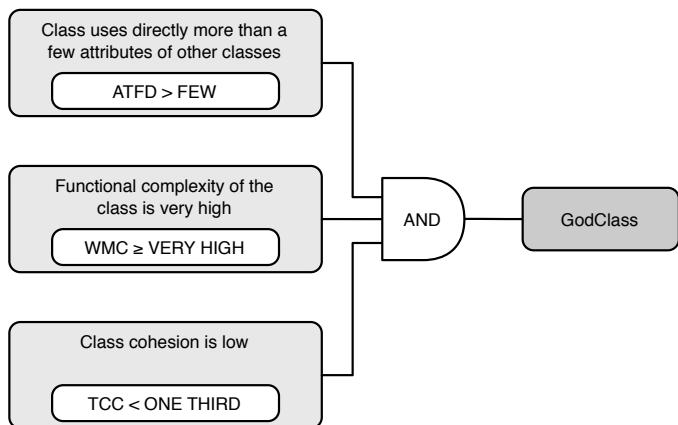
Un **BLOB** :



- **Est complexe**
- **N'est pas cohésif**
- **Utilise des données externes**

A **God Class** centralizes too much intelligence in the system.

Lanza, Marinescu 2006



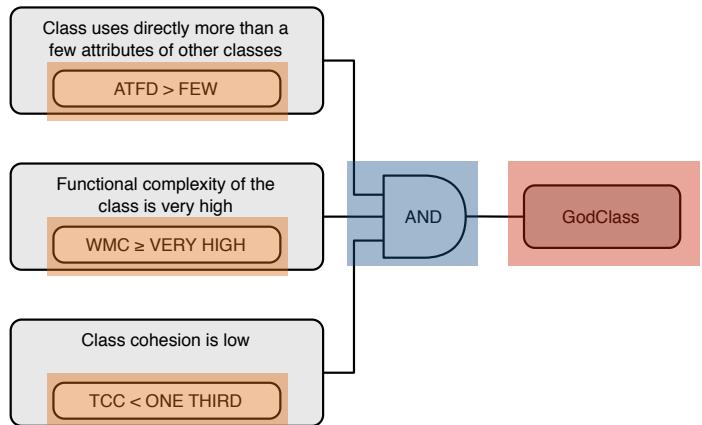
Access To Foreign Data (ATFD)

Weighted Method Count(WMC)

Tight Class Cohesion (TCC)

A **God Class** centralizes too much intelligence in the system.

Lanza, Marinescu 2006



Composition de métrique

Problème de conception

Class uses directly more than a few attributes of other classes

ATFD > FEW

Functional complexity of the class is very high

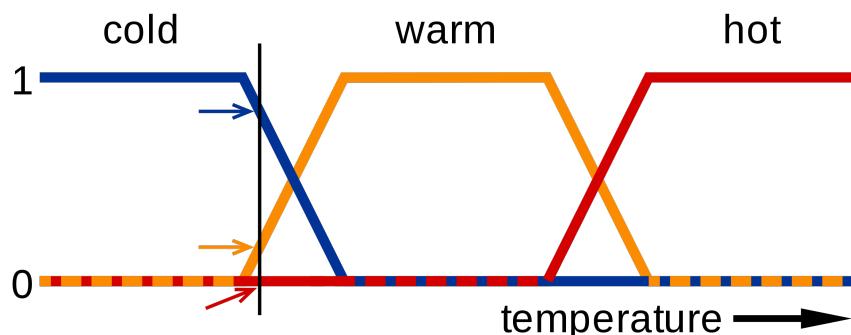
WMC ≥ VERY HIGH

Class cohesion is low

TCC < ONE THIRD

Comment Définir Les Seuils ?

Définition des seuils



Principes de logique dite "floue"

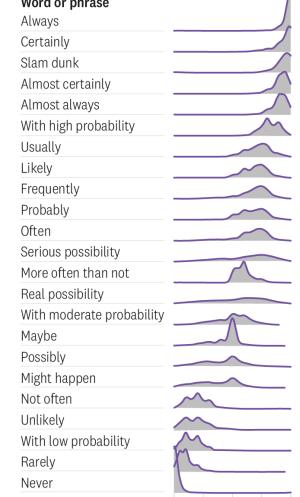
https://en.wikipedia.org/wiki/Fuzzy_logic

Il est toujours difficile de quantifier les seuils de detection

Qui pour les définir ?

How People Interpret Probabilistic Words
"Always" doesn't always mean always.

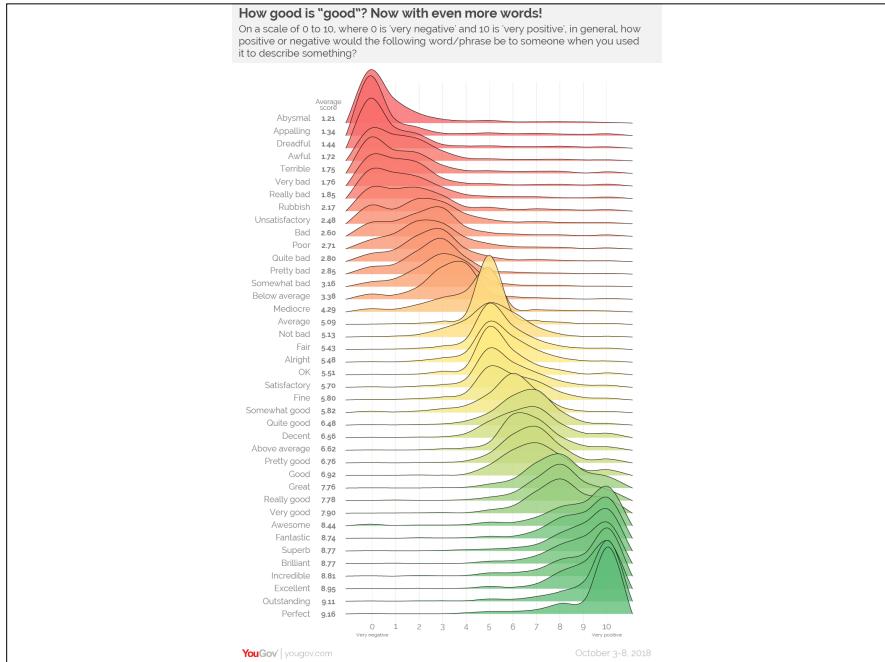
Distribution of responses according to respondents' estimate of likelihood



<https://hbr.org/2018/07/if-you-say-something-is-likely-how-likely-do-people-think-it-is>

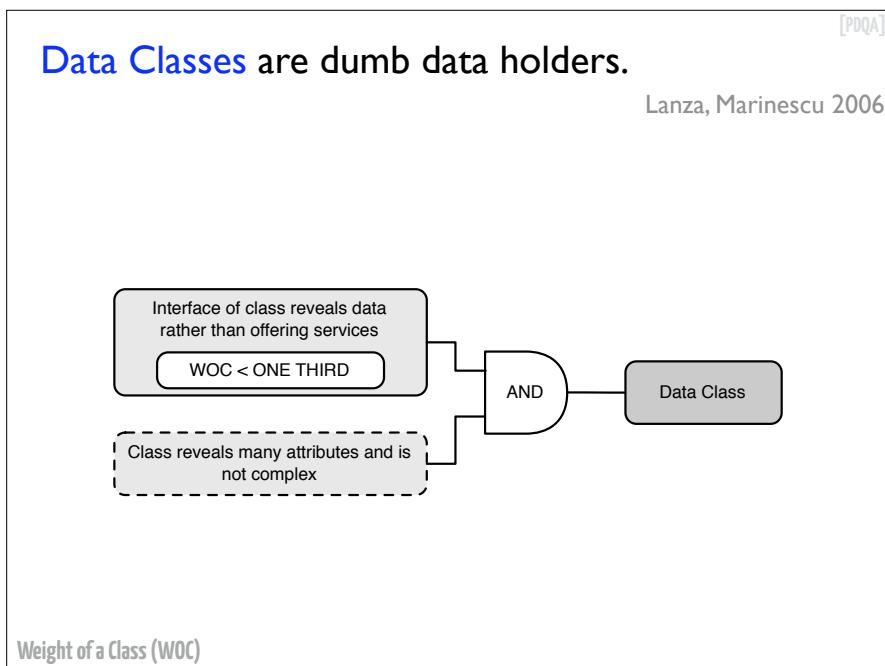
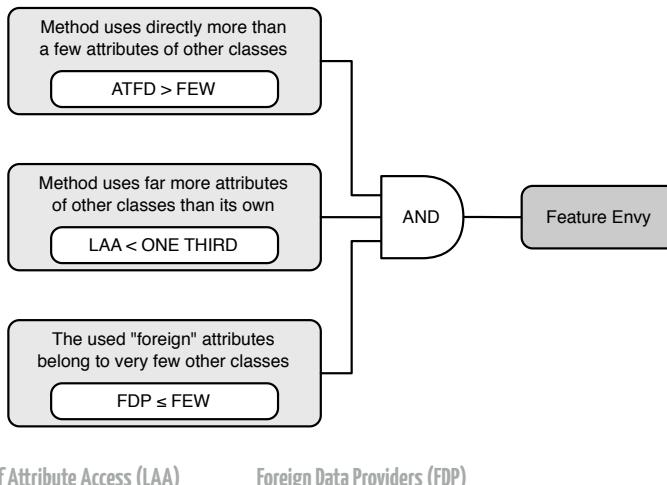
Source: Andrew Mauboussin and Michael J. Mauboussin

© HBR



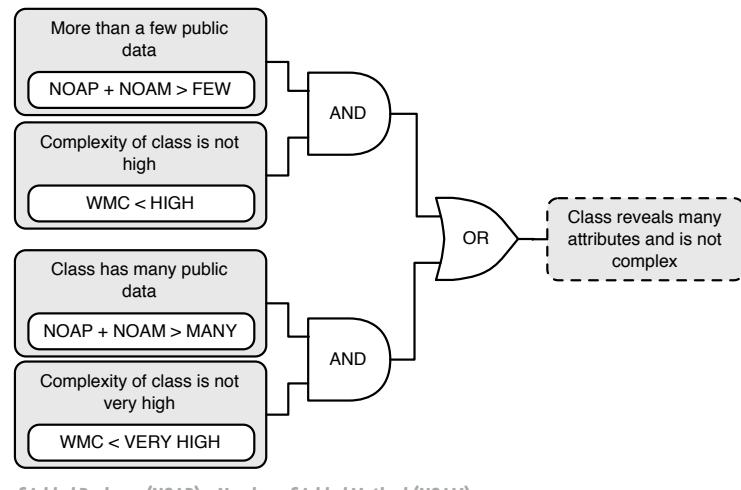
An Envious Method is more interested in data from a handful of classes.

Lanza, Marinescu 2006

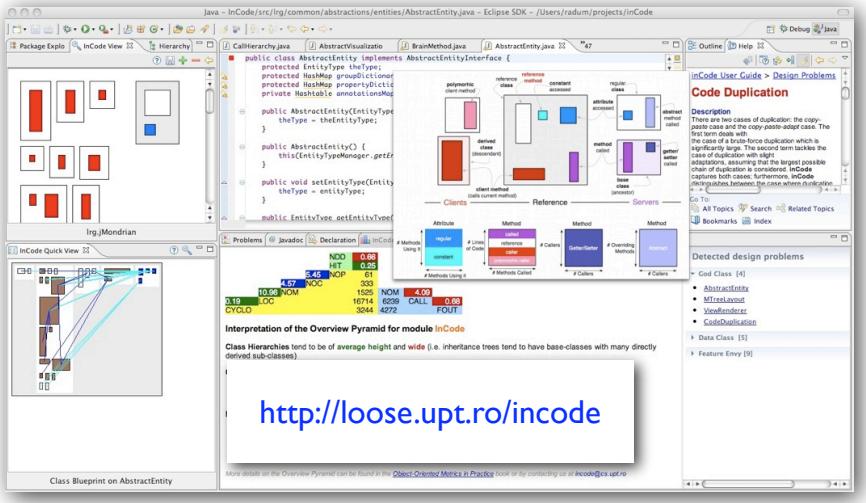


Data Classes are dumb data holders.

Lanza, Marinescu 2006



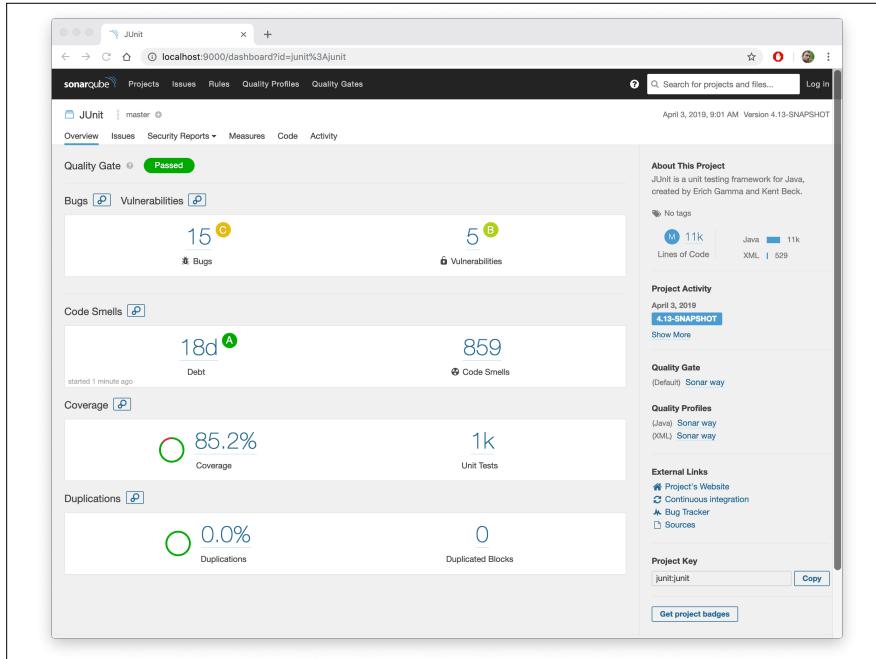
L'analyse de qualité fait partie du cycle de vie



Exemple Industriel : SonarQube

```
lucifer:tmp$ git clone https://github.com/junit-team/junit4.git
lucifer:tmp$ cd junit4
lucifer:junit4$ mvn org.jacoco:jacoco-maven-plugin:prepare-agent \
               clean verify
lucifer:junit4$ mvn sonar:sonar
```

Fonctionne avec n'importe quel projet Java géré par Maven

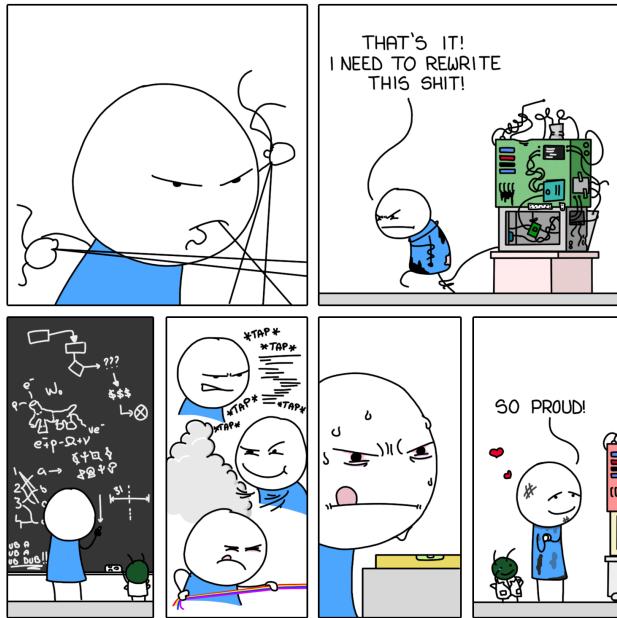
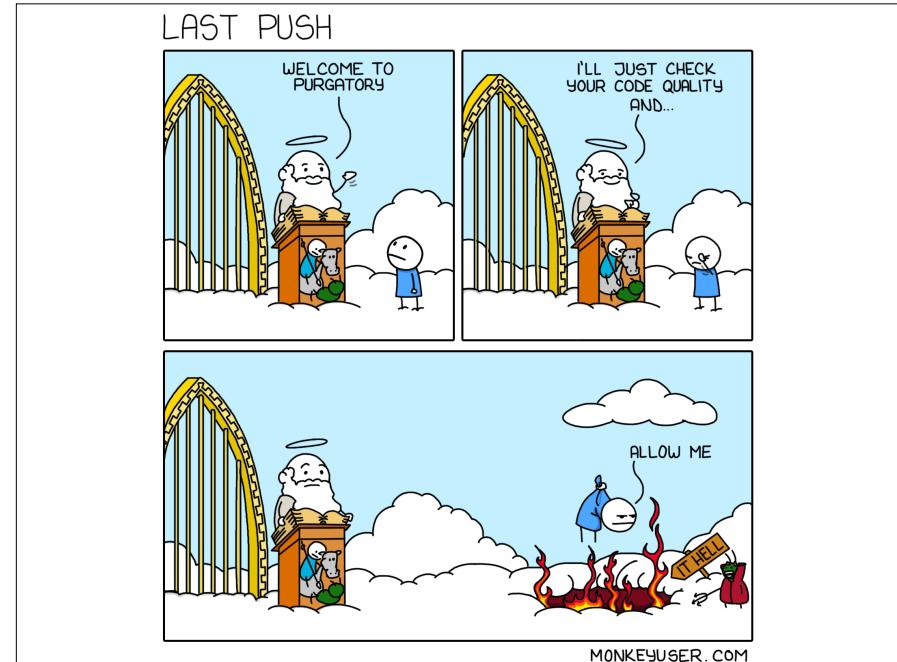


Qui dit **métrique** dit aussi **bon sens** ...



Analyser

3



Maintenir la qualité, c'est compliqué

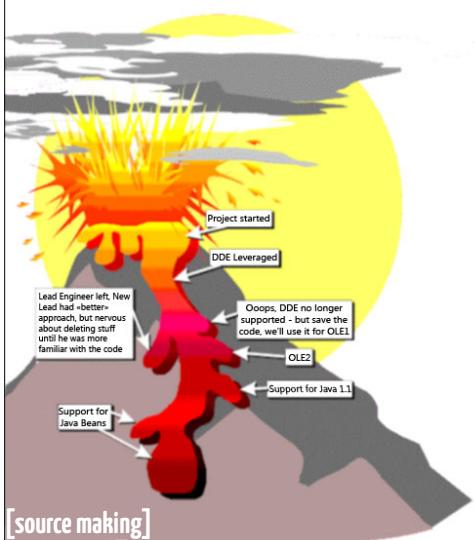
Qu'est-ce qu'un anti-patron ?

Votre Premier Anti-Patron

Un **BLOB** (ou **classe Dieu**) tend à **centraliser toute l'intelligence** du système, à **tout faire** et à **utiliser** des données en provenance de **classes purement structurelle**



Le fleuve de lave



Mauvaises
décisions
prises

Code “mort”
dans la base
de code

Le poltergeist

Classes inutiles
(sans responsabilité)

Cycle de vie
(ultra court)

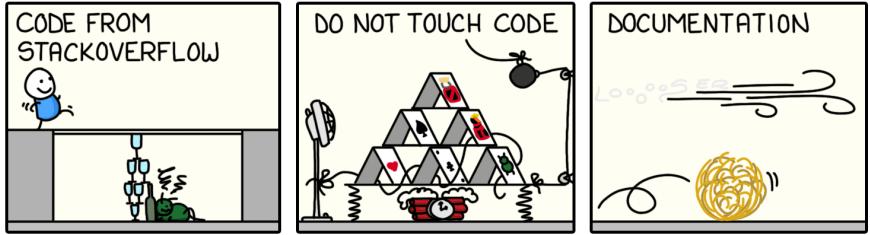


[source making]

Le champ de mine

Bug mineur mais **conséquences majeures**

Absence de maîtrise des objets conçus



Et tous les autres

- Le **code spaghetti** / la programmation copier-coller
 - Fort couplage, tout est entrelacé, cauchemar de maintenance
- Le **Marteau doré** (Golden Hammer)
 - Un outil / une technologie / une méthodologie pour les gouverner tous.
 - Aussi appelé : "MVC everywhere", "RoR everywhere", ...
- La **balle d'argent** (Silver Bullet)
 - Recherche de la solution ultime qui résoudra TOUS les problèmes
 - Même ceux qu'on connaît pas encore

La classe Couteau-Suisse

Ajout de **fonctionnalités non cohésives**



Manque de responsabilisation des entités

