Génie Logiciel : Conception

Métriques & Visualisation de code

Sébastien Mosser

INF-5153, Hiver 2019, Cours #10.1

UQÀM

Images: PixaBay

1

Pragmatic Design Quality Assessment

Tudor Gîrba
University of Bern, Switzerland

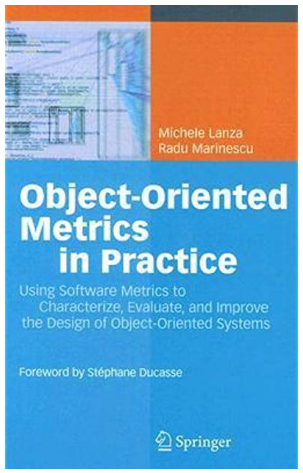
Michele Lanza
University of Lugano, Switzerland

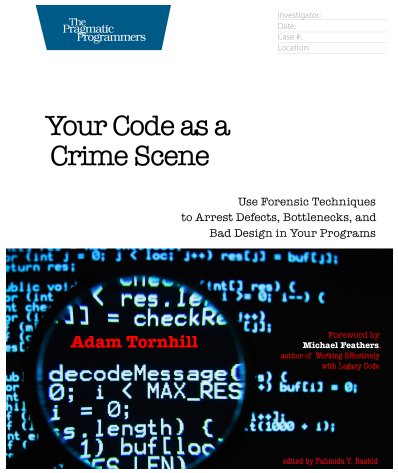
Radu Marinescu
Politehnica University of Timisoara, Romania

[PDQA]

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

2





3

1 Quantifier le logiciel

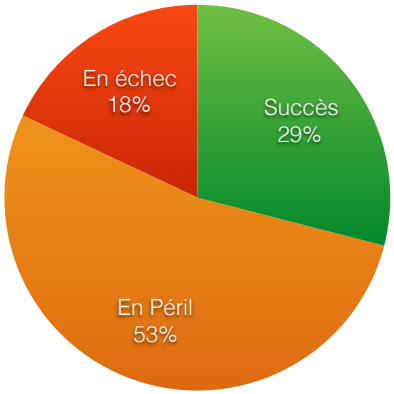
2 Du symptôme au problème

3 Visualiser le logiciel

4 Maîtriser l'évolution du logiciel

4

Le logiciel, cet objet complexe



[The Stanford Group, 2004]

[PDQA]

5

1,000,000,000 de lignes de code

x 2s = 2,000,000,000 secondes

/ 3600 = 560 heures

/ 8 = 70 jours

/ 20 = 3 mois !!

[PDQA]

6

Heureusement, on ne lit pas le code entièrement !

On lit les modèles de conception !

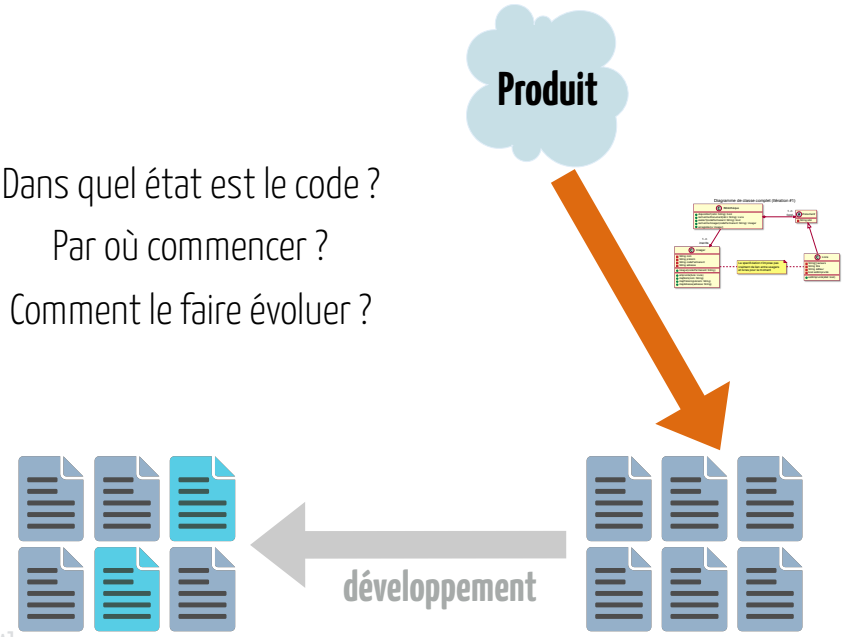
[PDQA]

7

Dans quel état est le code ?

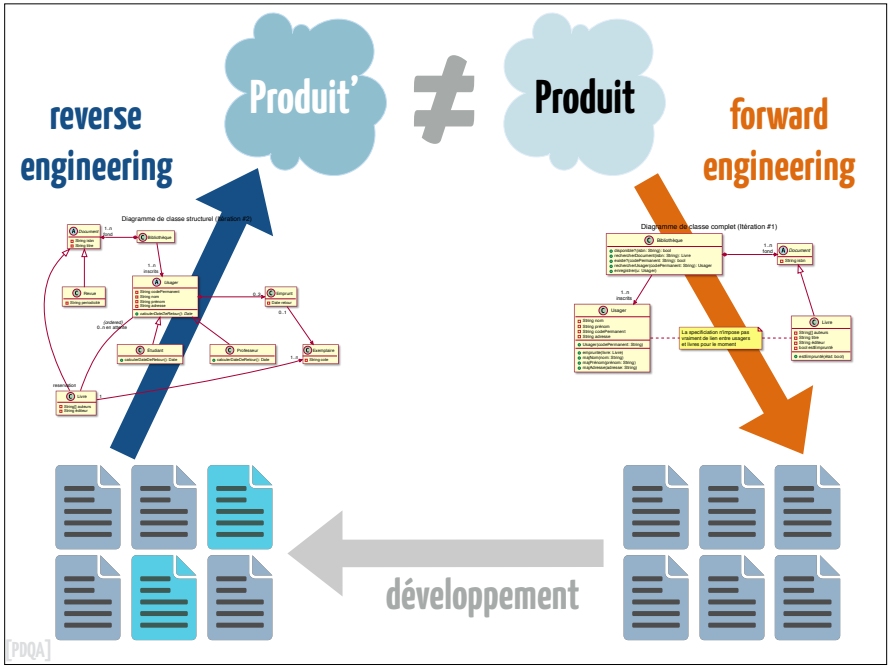
Par où commencer ?

Comment le faire évoluer ?



[PDQA]

8



9

Un vrai système contient beaucoup de détails !

Comment juger de sa qualité ?

[PDQA]

10

How the UML diagram describes the software

How the code is actually written

https://twitter.com/JobOpportunIT_/status/1106631344233295872?s=19

11

Quantifier le logiciel

1

12

You cannot **control**

what you cannot **measure**

Tom de Marco

[PDQA]

13

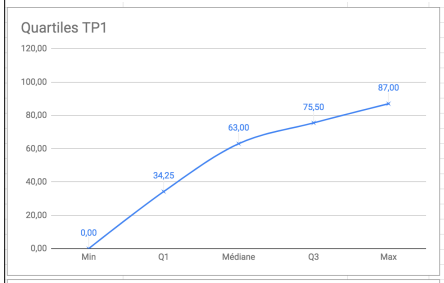
Métrique (définition)

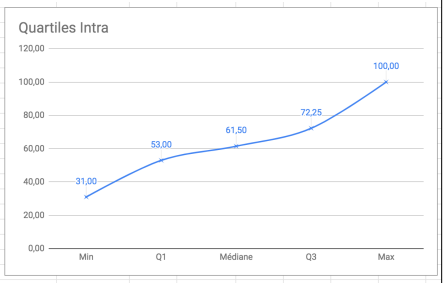
Une **métrique** est une **fonction** qui **assigne un nombre** à un **produit**, un **processus** ou une **ressource**.

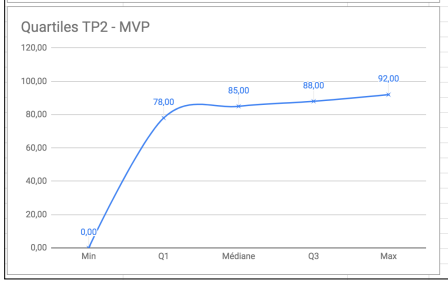
[PDQA]

14

Les métriques caractérisent des profils







Métriques
INF-5153
H19

15

Métrique Logicielle (définition)

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

[PDQA]

16

Exemples de métriques OO

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]

17

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

A part ça ... pas grand chose

[PDQA]

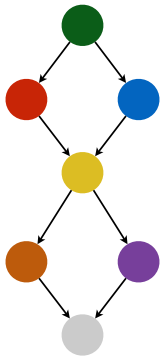
18

CYCLO(P) ?

P =

```
if( c1() )
   f1();
else
   f2();

if( c2() )
   f3();
else
   f4();
```

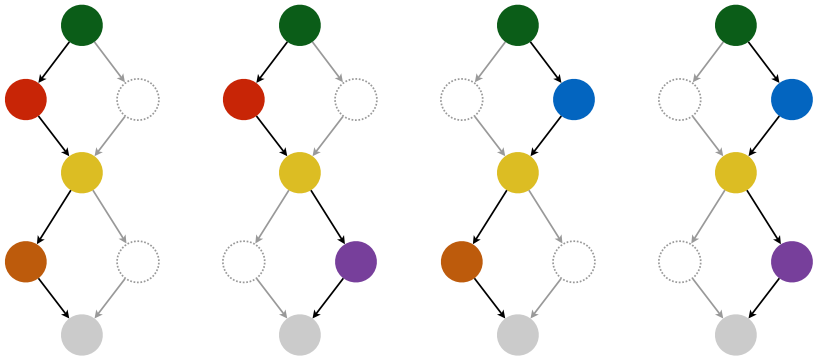
≡



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

19

CYCLO(P) = 4



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

20

Méthodes pondérées (WMC)

"**Weighted Method Count**" (WMC) fait la **somme pondérée des méthode d'une classe** (classiquement **avec CYCLO** pour pondérer)

Configurable, on peut changer la pondération au besoin

A part ça ... pas grand chose

[PDQA]

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 ?

[PDQA]

Couplage entre objets (CBO)

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

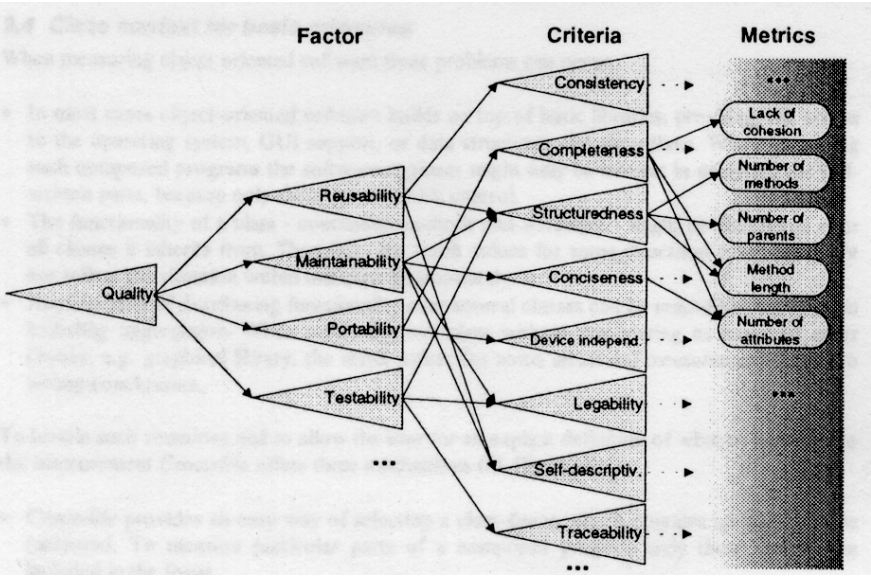
Prend en compte les vrais dépendances

Pas de mesure de l'intensité du couplage

[PDQA]

●●●

[PDQA]



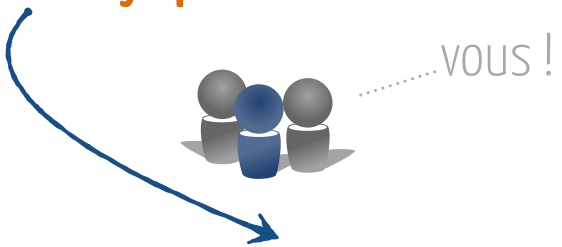
McCall, 1977

[PDQA]

25

Le problème #1 avec les métriques ...

Capture un symptôme



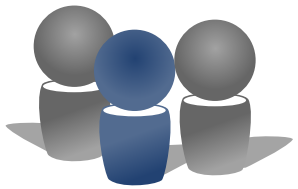
Comment en tirer le traitement ?

[PDQA]

26

Le problème #2 avec les métriques ...

SOLID GRASP DRY KISS



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

[PDQA]

27

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

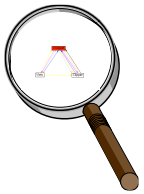
En quoi ça nous aide ?

[PDQA]

28

En quoi ça nous aide ?

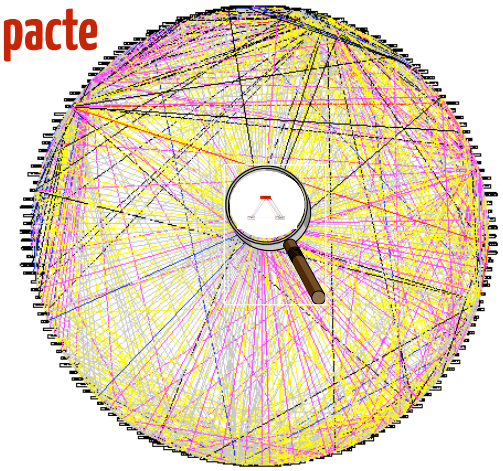
Si je change
juste cette méthode ...



[PDQA]

29

Mais qu'en fait ça impacte
33% du code



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

[PDQA]

30

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

Fréquents

Inévitables

Dispendieux

[PDQA]

31

Comment les éviter ?

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

[PDQA]

32

Du symptôme au problème

2

33

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]

34

Un **BLOB** :

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

[PDQA]

35

Un **BLOB** :

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

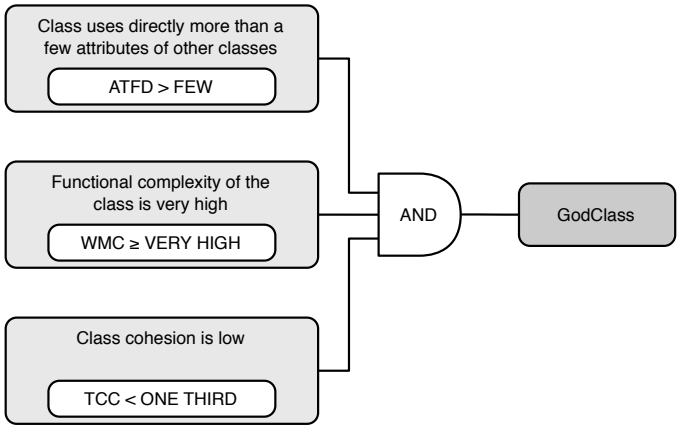
[PDQA]

36

[PDQA]

A God Class centralizes too much intelligence in the system.

Lanza, Marinescu 2006

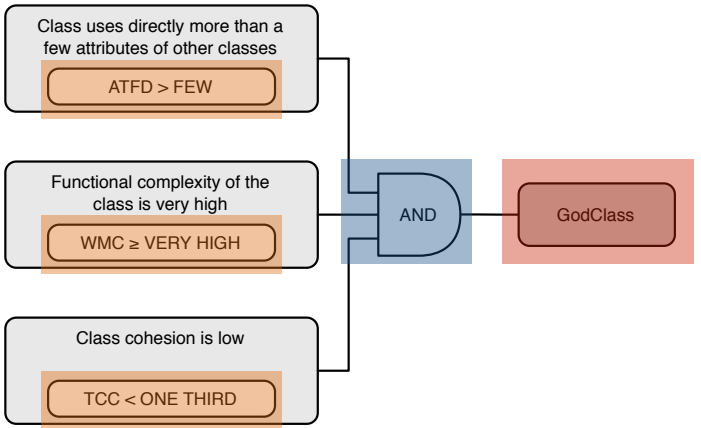


Access To Foreign Data (ATFD) Weighted Method Count (WMC) Tight Class Cohesion (TCC)

37

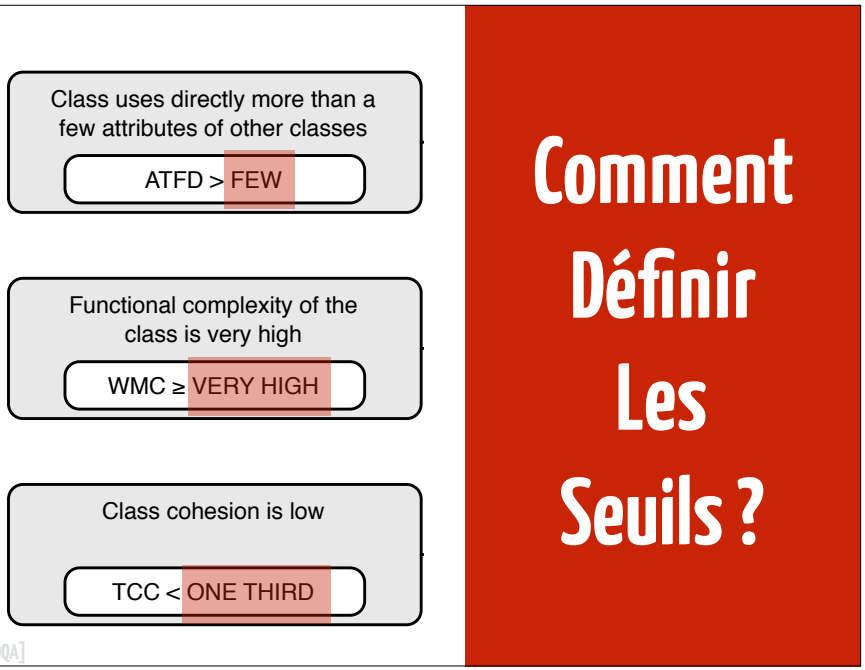
A God Class centralizes too much intelligence in the system.

Lanza, Marinescu 2006



[PDQA] Composition de métrique Problème de conception

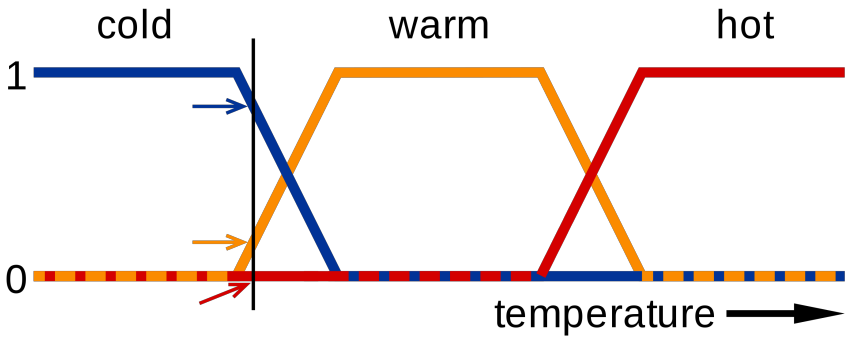
38



[PDQA]

39

Définition des seuils



Principes de logique dite “floue”

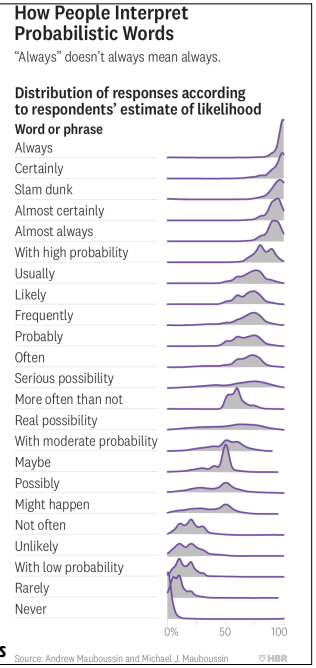
https://en.wikipedia.org/wiki/Fuzzy_logic

40

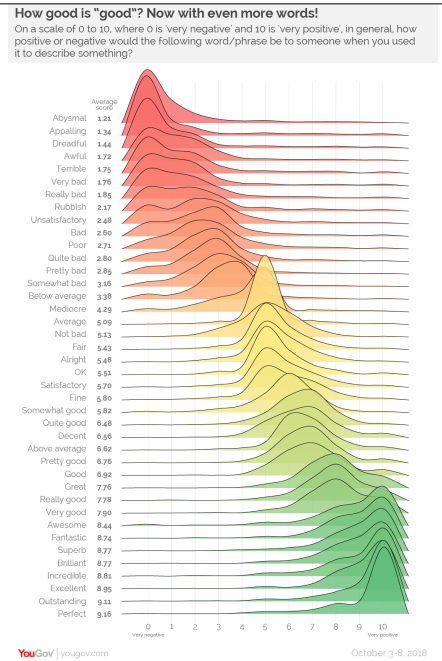
Il est toujours difficile de quantifier les seuils de detection

Qui pour les définir ?

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



41

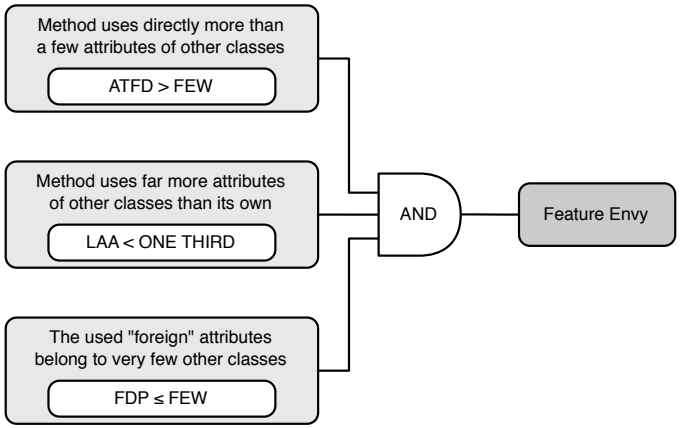


42

[PDQA]

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

Lanza, Marinescu 2006



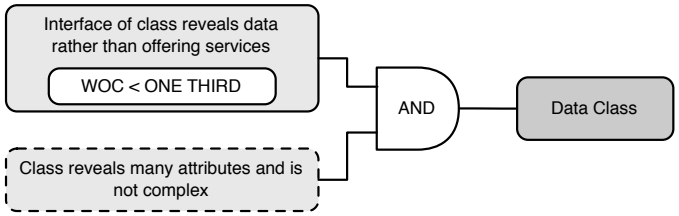
Locality of Attribute Access (LAA) Foreign Data Providers (FDP)

43

[PDQA]

Data Classes are dumb data holders.

Lanza, Marinescu 2006

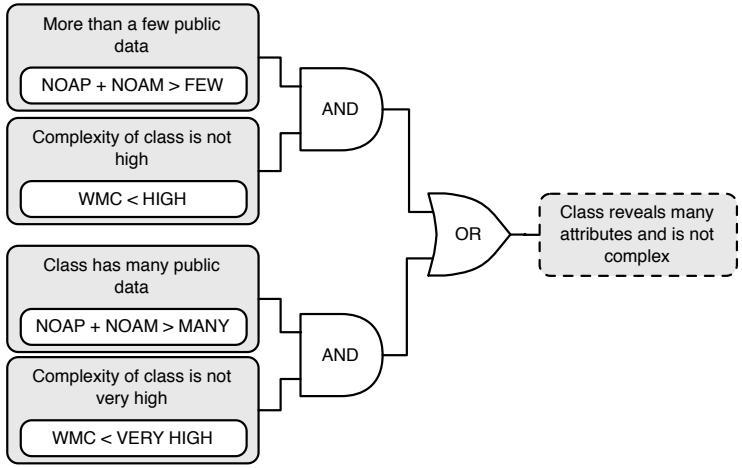


Weight of a Class (WOC)

44

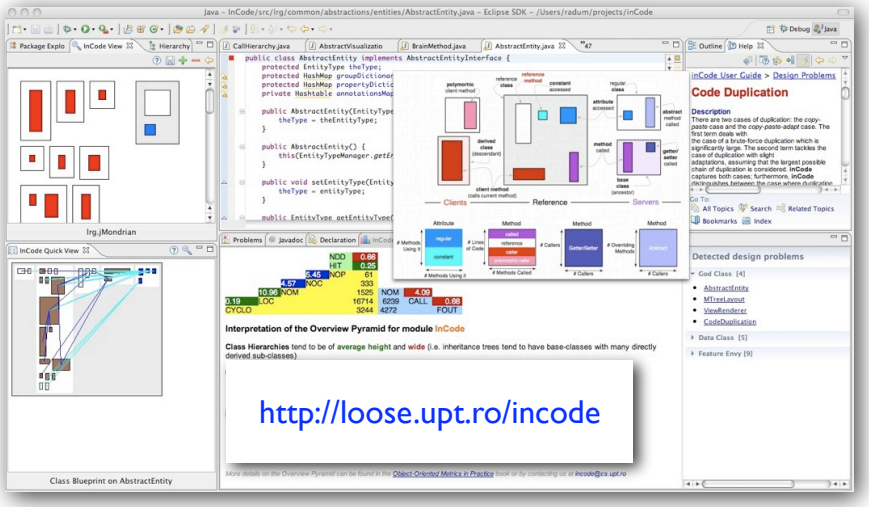
Data Classes are dumb data holders.

Lanza, Marinescu 2006



Number of Added Package (NOAP) Number of Added Method (NOAM)

L'analyse de qualité fait partie du cycle de vie

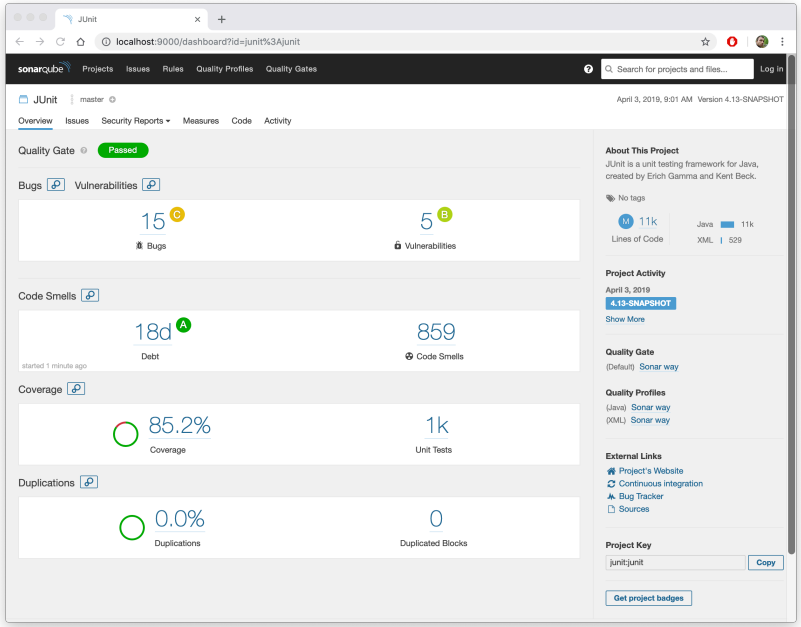


Qui dit **métrique** dit **bon sens** ...

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



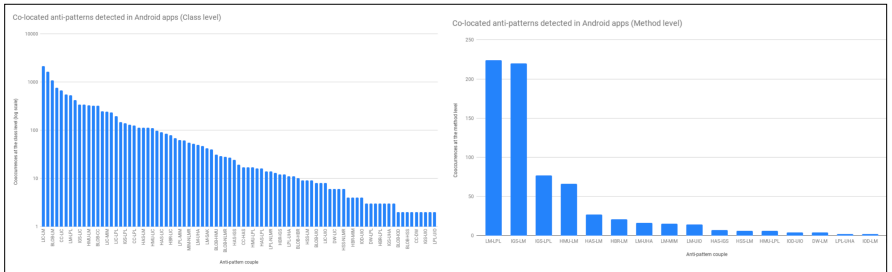
49

Exemple Académique : RepositoryMiner

```
public class Main {

   public static void main(String[] args) throws IOException {

      RepositoryMiner rm = new RepositoryMiner();
      rm.setRepositoryPath("/home/felipe/git/junit4");
      rm.setScm(new GitSCM());

      List<IMetric> metrics = Arrays.asList(new LOC(), new CYCLO());
      rm.setMetrics(metrics);

      List<ICodeSmell> codeSmells = Arrays.asList(new GodClass(), new LongMethod());
      rm.setCodeSmells(codeSmells);

      rm.mine();
   }

}
```

https://github.com/visminer/repositoryminer

50



(a) Overlapping anti-patterns detected at the class level

(b) Overlapping anti-patterns detected at the method level

https://ace-design.github.io/

May
Conference

A Delta-oriented Approach to Support the Safe Reuse of Black-box Code Rewriters.
Benjamin Benni, Sébastien Mosser, Naouel Moha, and Michel Riveill.
17th International Conference on Software Reuse (ICSR).

51

Peut-on mesurer la beauté d'une oeuvre en comptant le nombre de couleurs utilisées ?

[PDQA]

52

Visualiser le logiciel

3

53

Les métriques seules sont peu utile

Metric	Value	Remarks
No. of Lines of Code	223,068	including comments
No. of Source Files	1,209	*.java files
No. of Packages	99	–
No. of Classes	1,393	including 140 inner classes
No. of Methods	9,561	including accessor methods
No. of Attributes	3,358	all variables including static and local variables

On peut les combiner,
mais le résultat reste binaire

[PDQA]

54

Épidémie de Choléra à Londres (1854)

Quartier	|Puits|	|Malade|
A	1	Aucun
B	2	Aucun
C	1	Élevé
D	0	Élevé
E	0	Faible
F	2	Élevé
G	1	Faible
H	1	Faible

55

1854,
London,
cholera
epidemic

Quartier

Puits

[PDQA]

56

1854, London, cholera epidemic

Foyer Infectieux

Puits

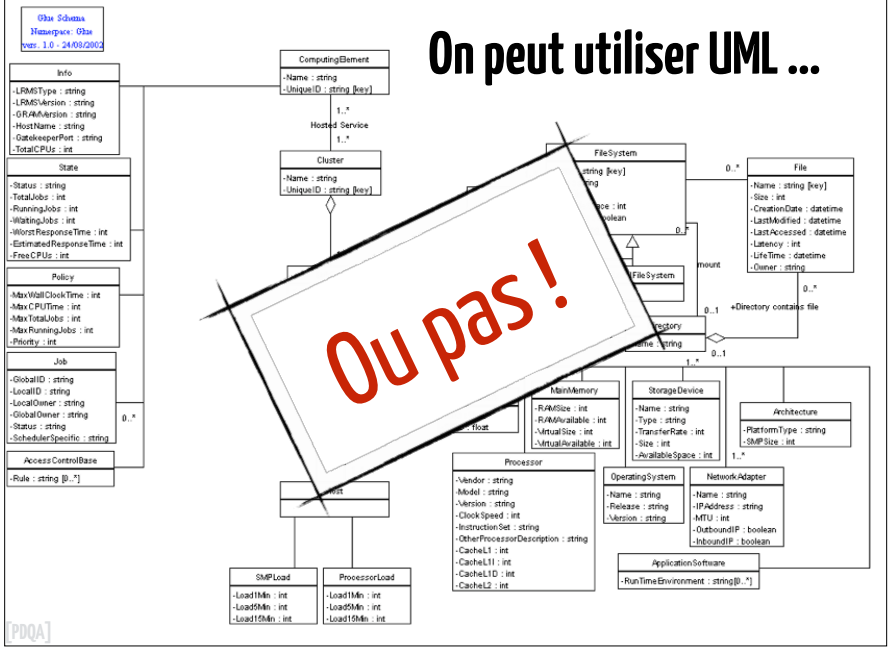
57

Épidémie de Choléra à Londres (1854)

Quartier	|Puits|	|Malade|
A	1	Aucun
B	2	Aucun
C	1	Élevé
D	3	Élevé
E	0	Faible
F	2	Élevé
G	1	Faible
H	1	Faible

La bonne visualisation est la distance au puit

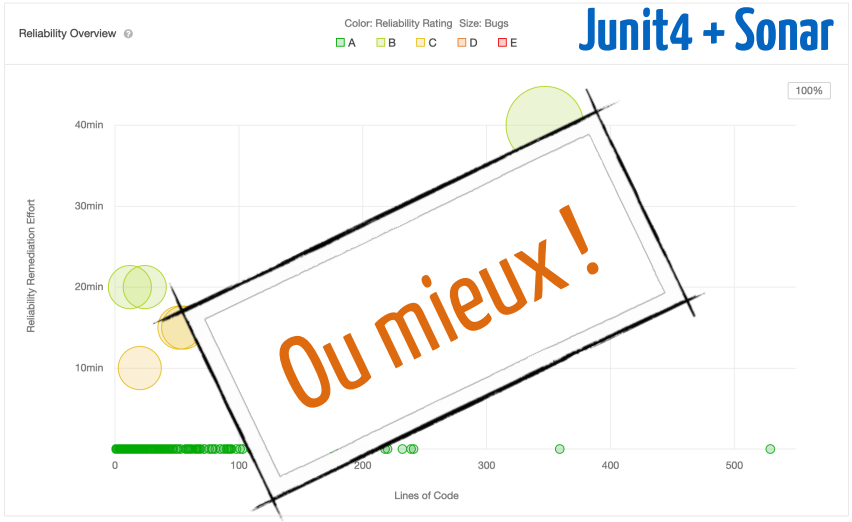
58



On peut utiliser UML ...

Ou pas !

59

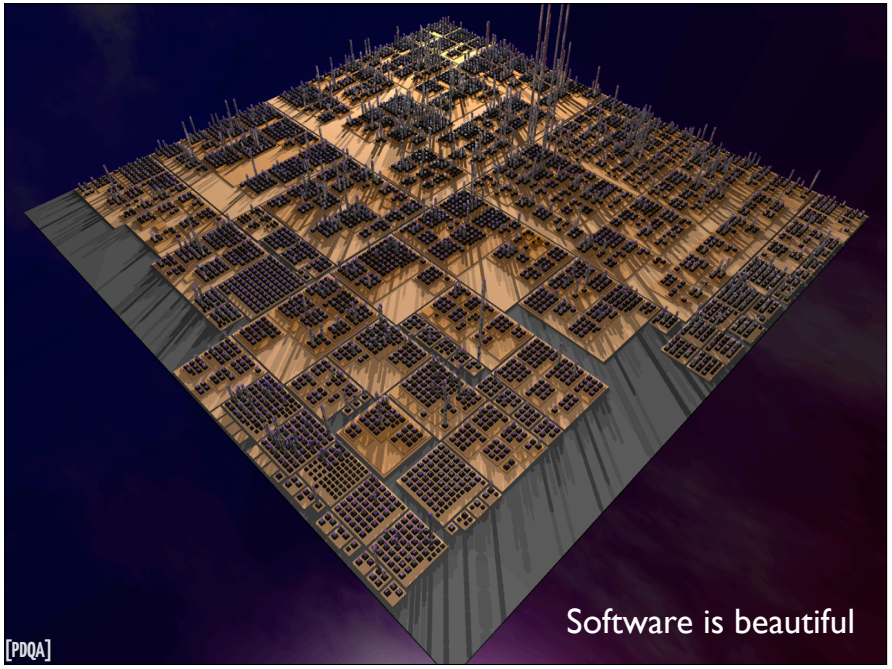


Junit4 + Sonar

Ou mieux !

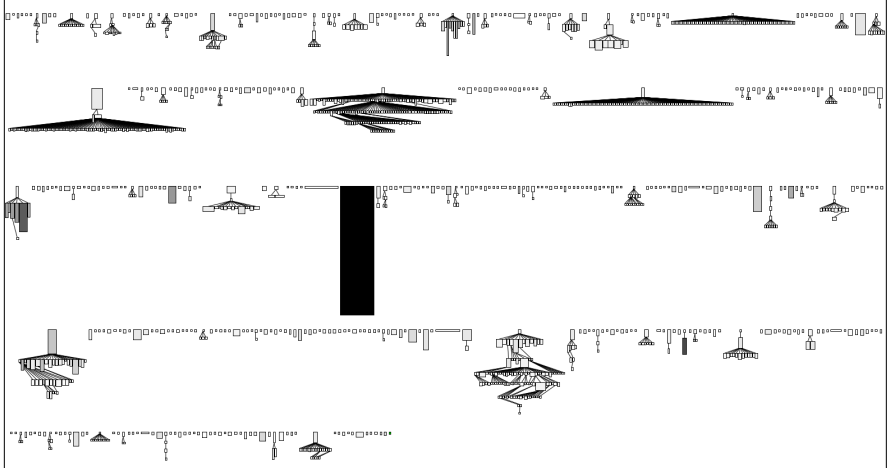
On peut utiliser des graphes ...

60



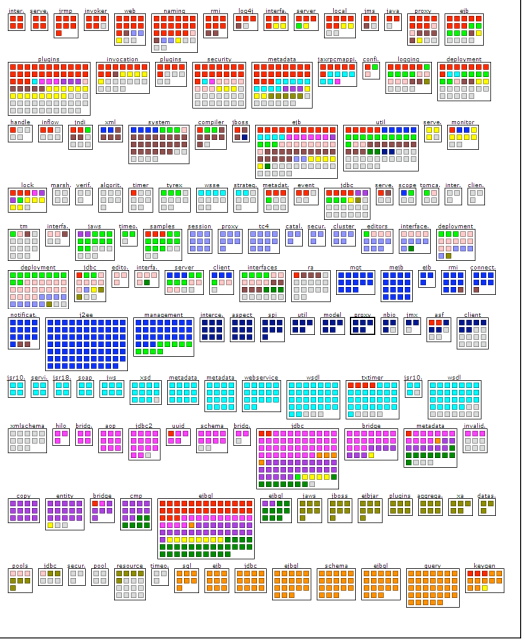
61

La visualisation rend la conception visible



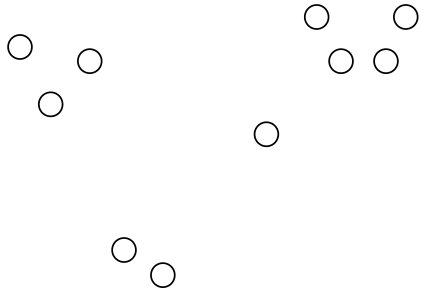
62

L'être humain est biologiquement capable de reconnaitre des motifs.



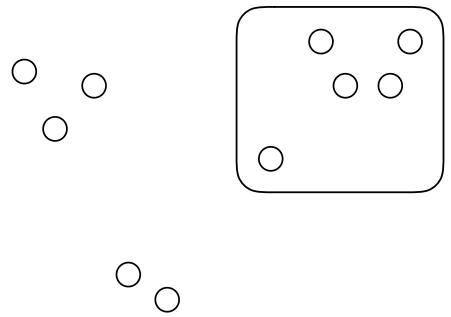
63

Combien de groupes voyez vous ?



64

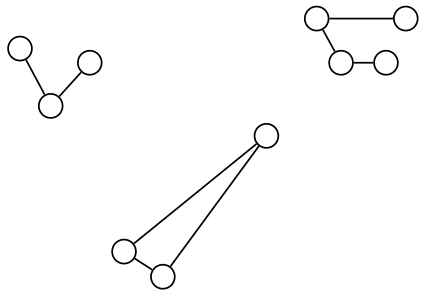
Combien de groupes voyez vous ?



[PDQA]

65

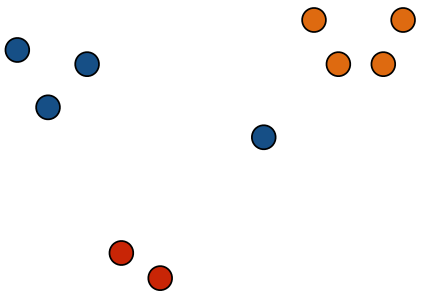
Combien de groupes voyez vous ?



[PDQA]

66

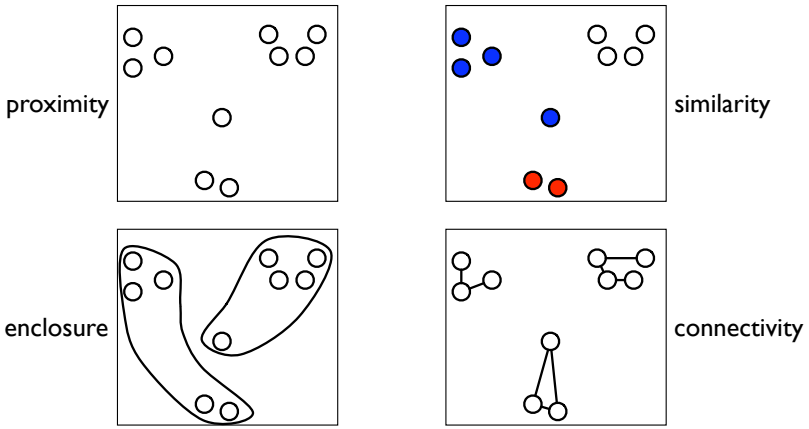
Combien de groupes voyez vous ?



[PDQA]

67

Principe de Gestalt



[PDQA]

68

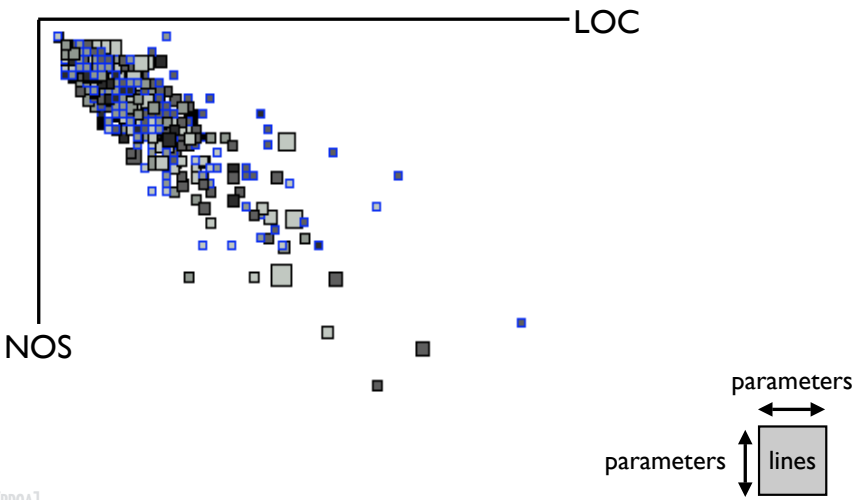
Visualiser = Exploiter la Pré-attention

8789364082376403128764532984732984732094873290845
389274-0329874-32874-2319847509834098340983240983 2
049823-0984903281453209481-0839393947896587436598

[PDQA]

69

A simple & powerful concept

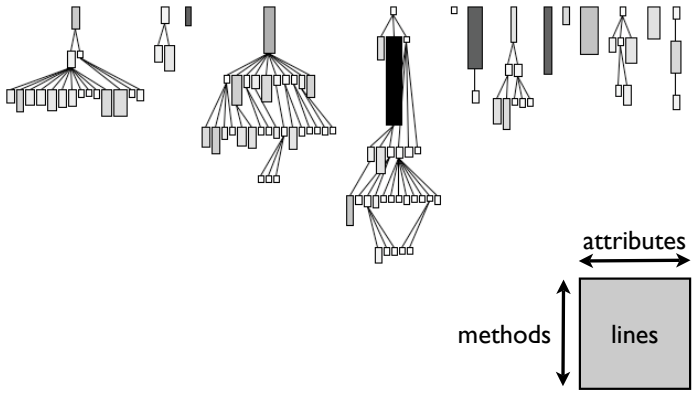


[PDQA]

70

System Complexity shows class hierarchies.

Lanza, Ducasse, 2003

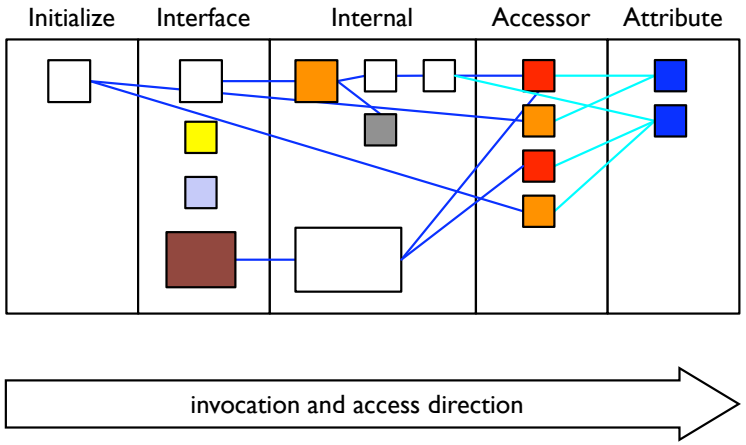


[PDQA]

71

Class Blueprint shows class internals.

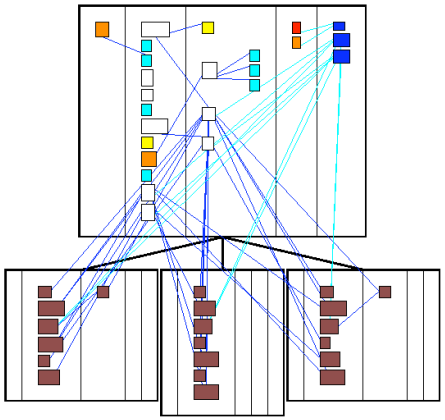
Lanza, Ducasse, 2005



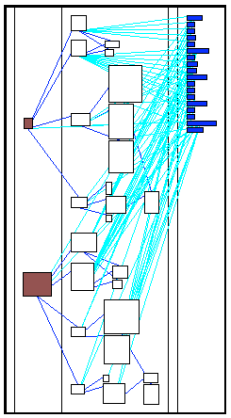
[PDQA]

72

Class Blueprint reveals patterns.



twin classes



schizophrenic class

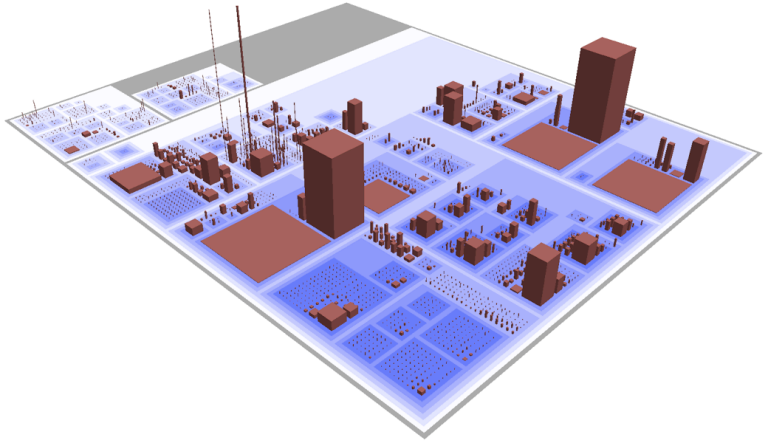
[PDQA]

Maîtriser l'évolution du logiciel

4

Code City shows where your code lives.

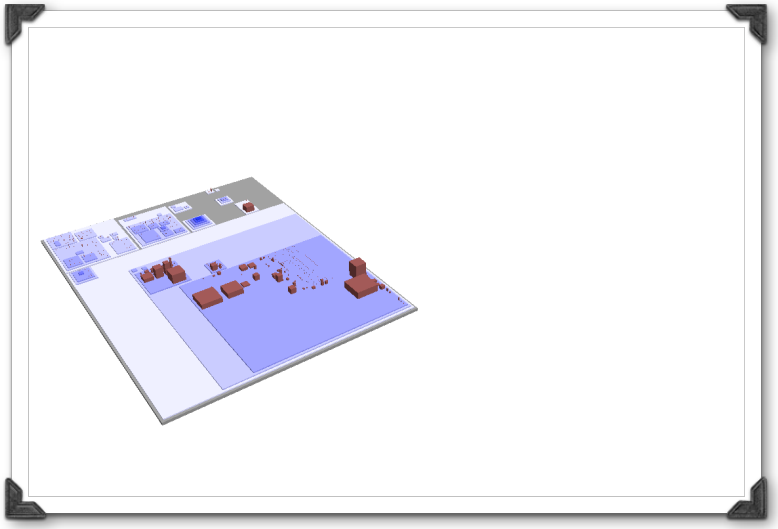
Wettel, Lanza, 2007



classes are buildings grouped in quarters of packages

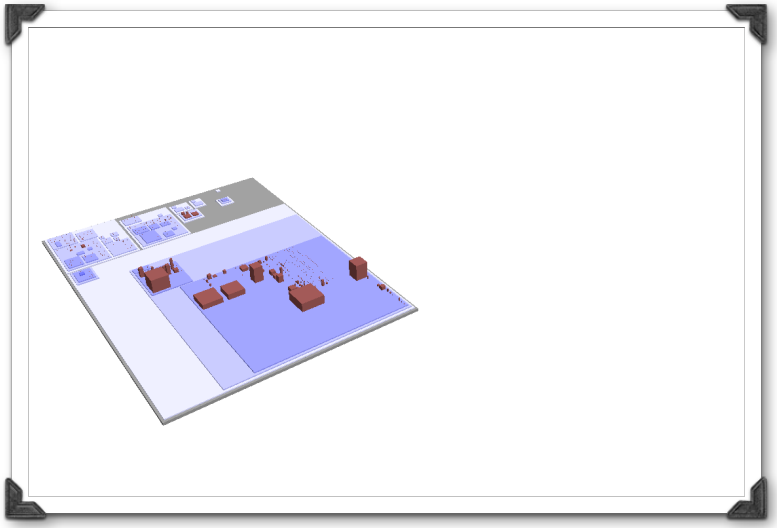
[PDQA]

Jmol - The Time Machine



[PDQA]

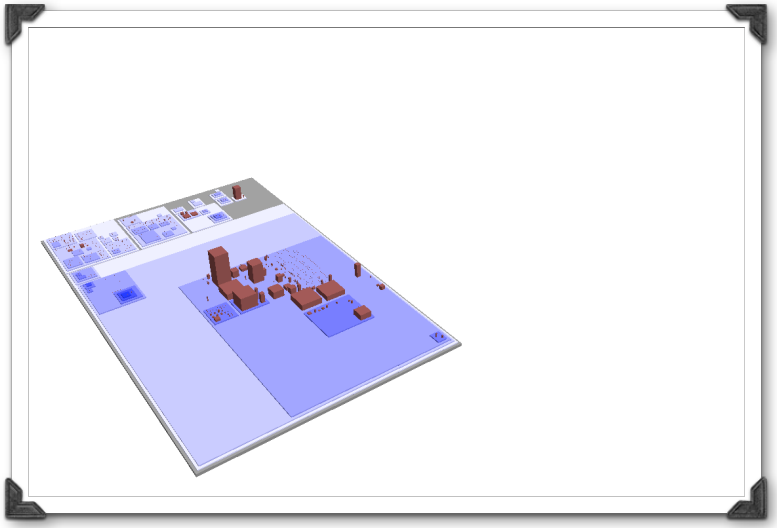
Jmol - The Time Machine



[PDQA]

77

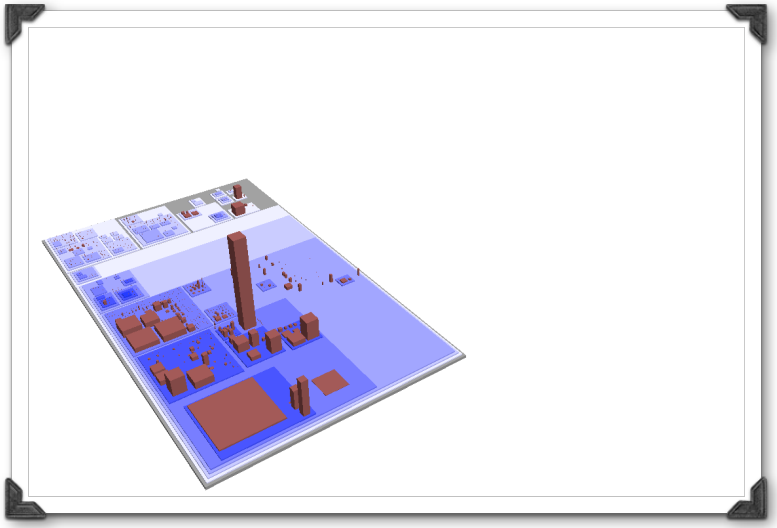
Jmol - The Time Machine



[PDQA]

78

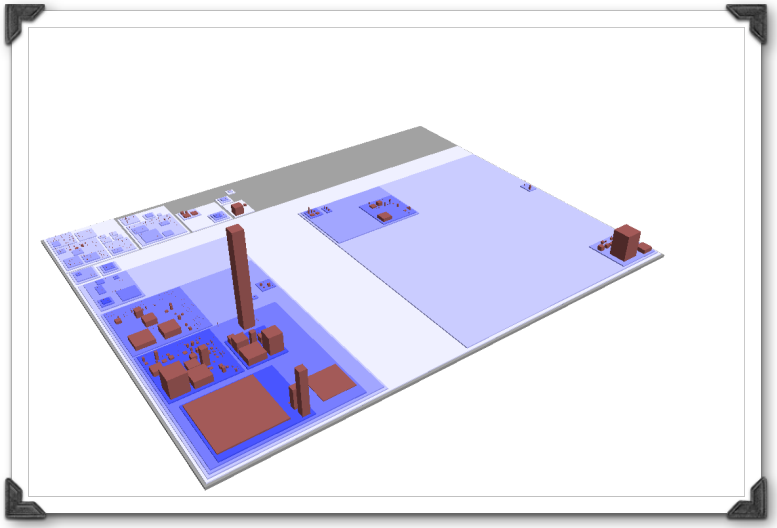
Jmol - The Time Machine



[PDQA]

79

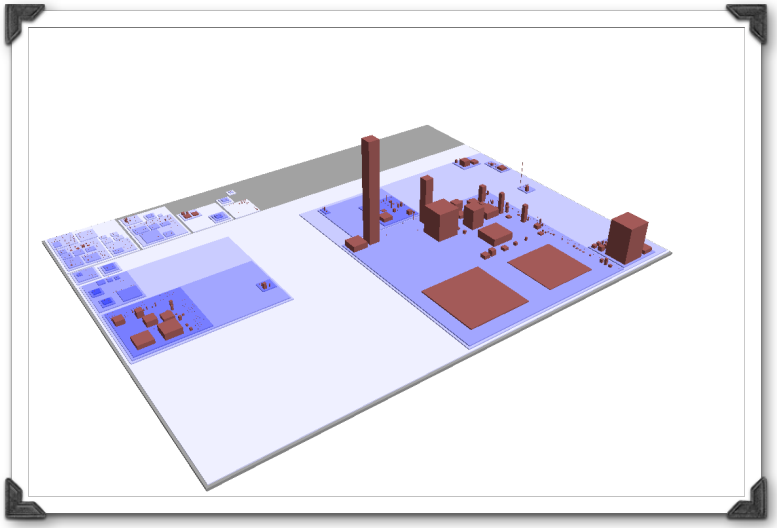
Jmol - The Time Machine



[PDQA]

80

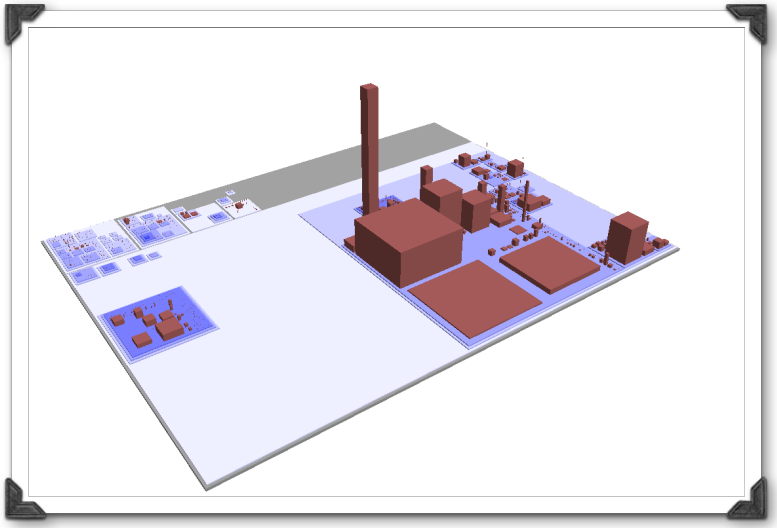
Jmol - The Time Machine



[PDQA]

81

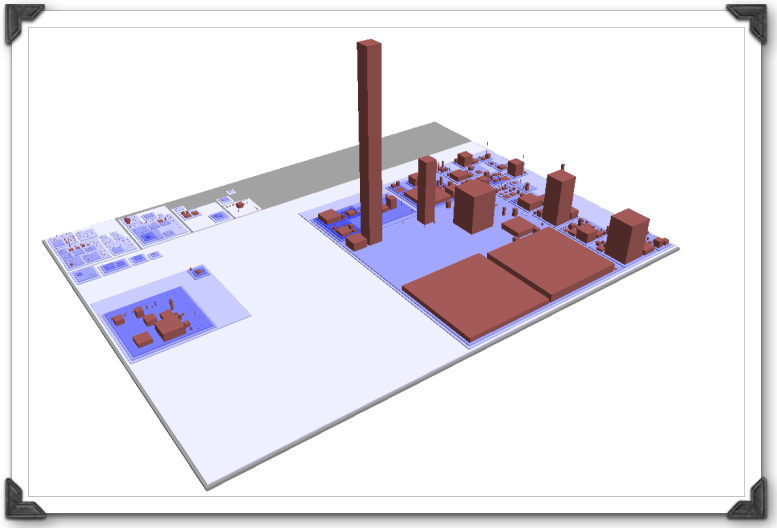
Jmol - The Time Machine



[PDQA]

82

Jmol - The Time Machine



[PDQA]

83