Enseigner la rétro-ingénierie, en s'interrogeant sur l'évolution du logiciel : retour d'expériences

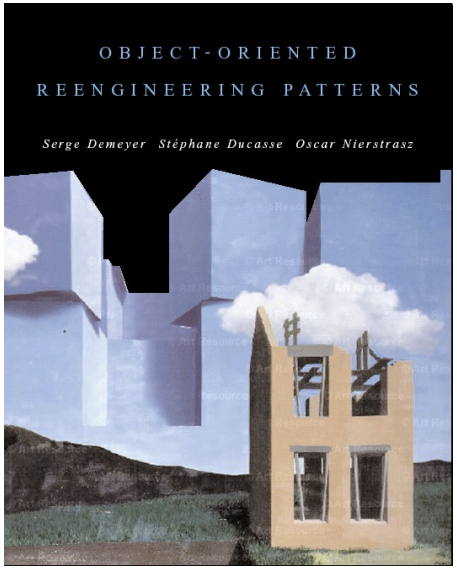
Mireille Blay-Fornarino, Sébastien Mosser & Xavier Blanc

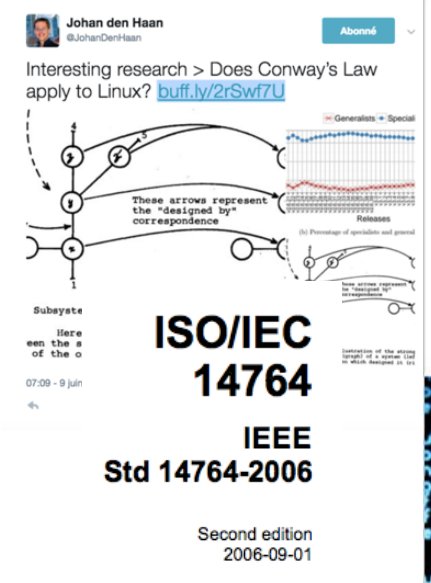
Maintenance Logicielle?

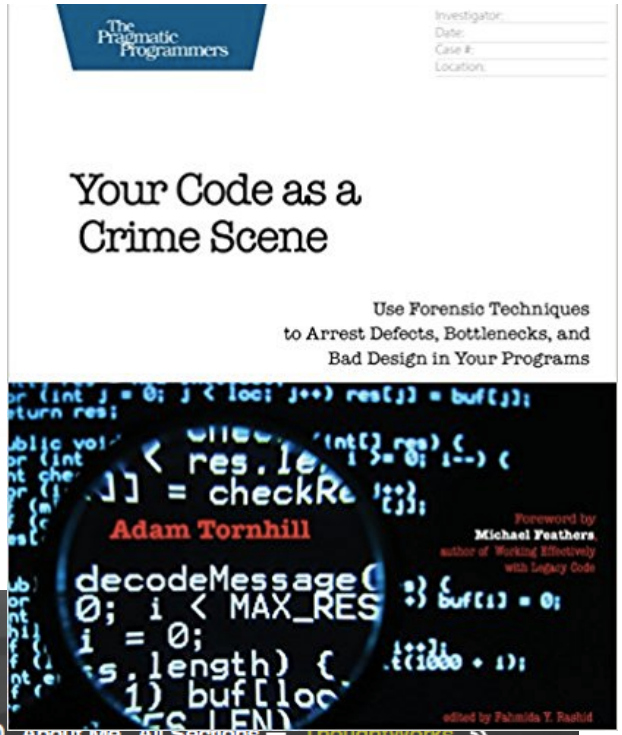
Enseigner la Maintenance Logicielle

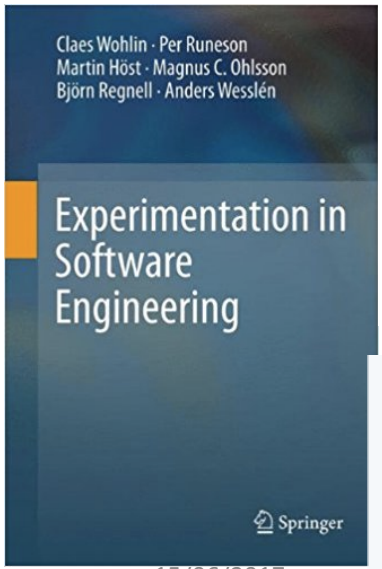
« réputée non-enseignable ».

Dédicace @pcollet









Analyse de code ?

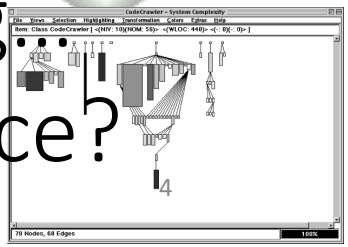
Code review

analysis of software system using **machine learning techniques**. 2017

Agilité …

Refactoring

Maintenance?



Maintenance logicielle dans l'entreprise ?

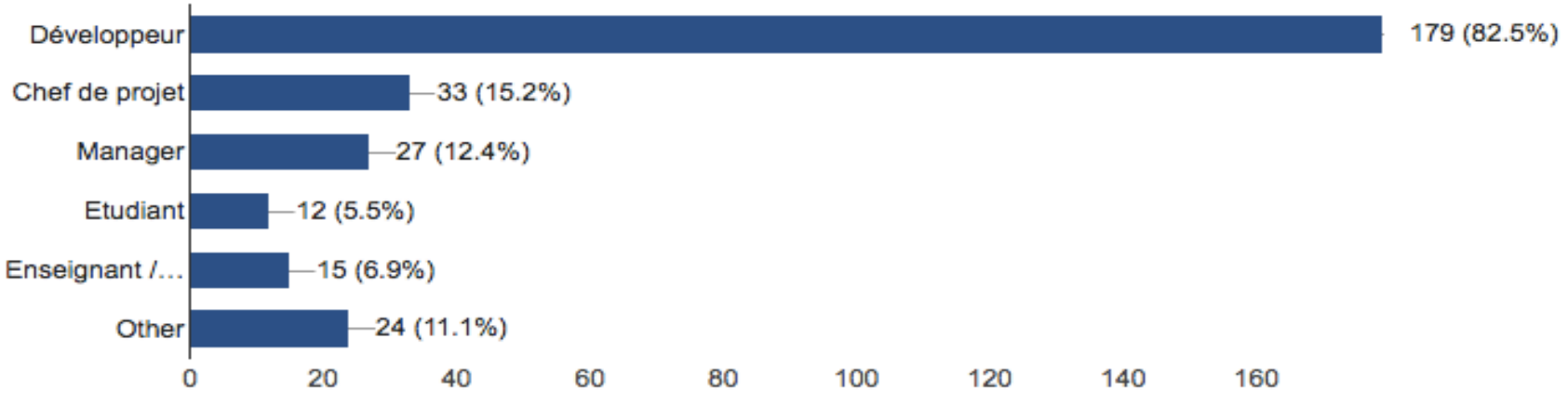
Un sondage…

juillet à octobre 2016

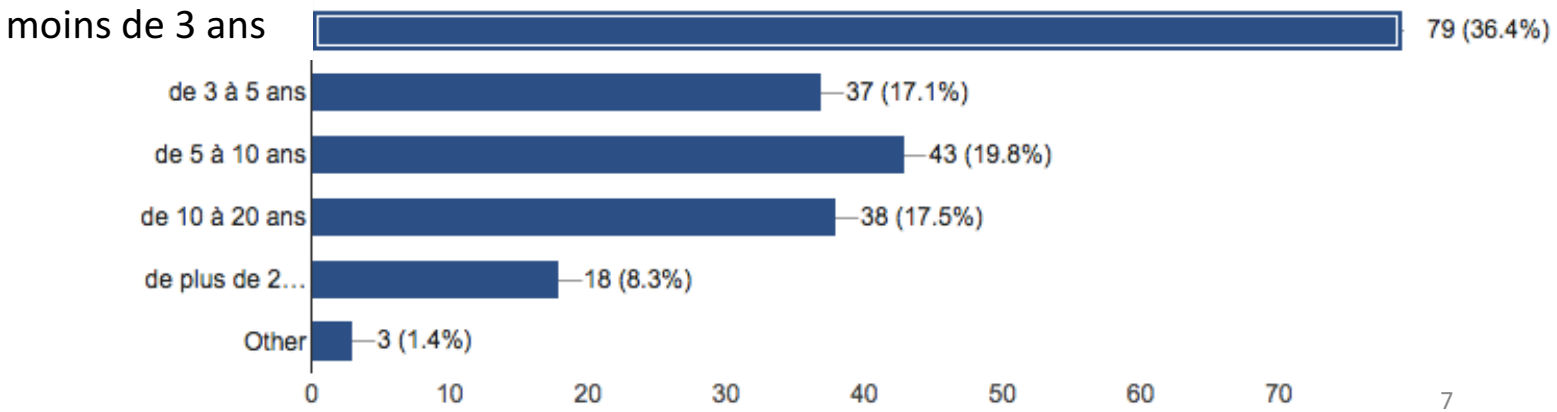
https://goo.gl/kYPZhX

Vous êtes (217 responses)

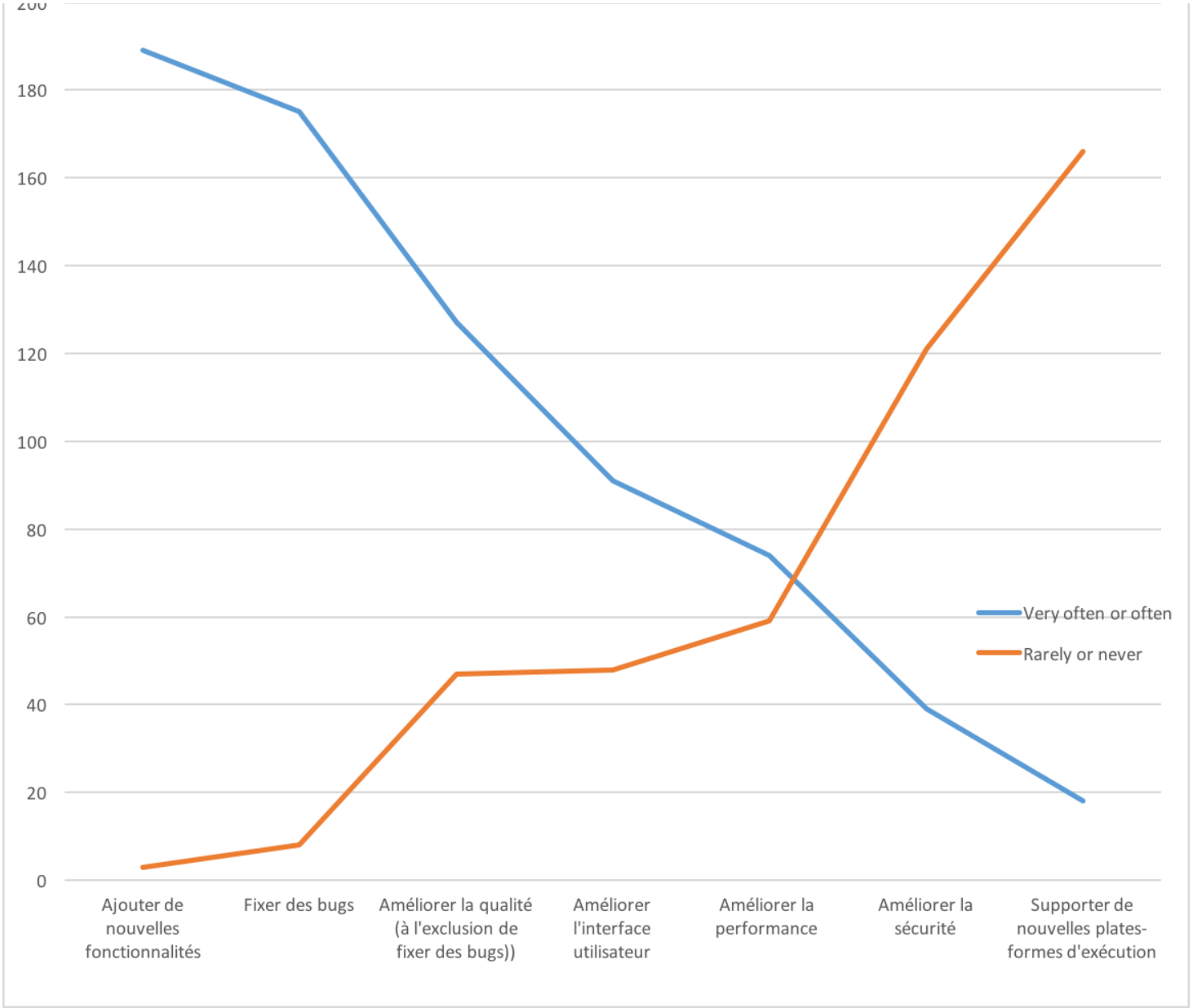
217 réponses

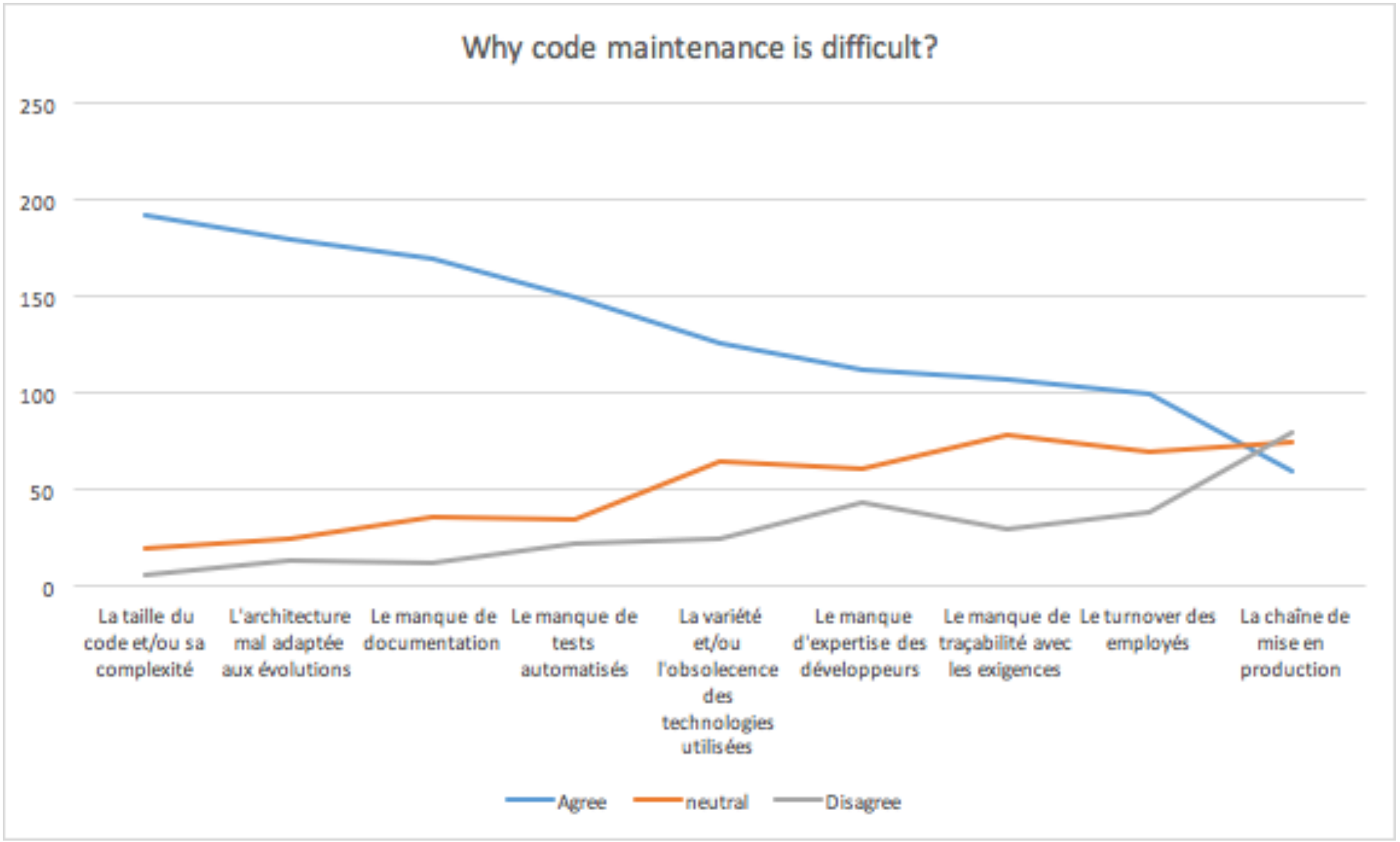


Vous avez une expérience professionnelle de (217 responses)

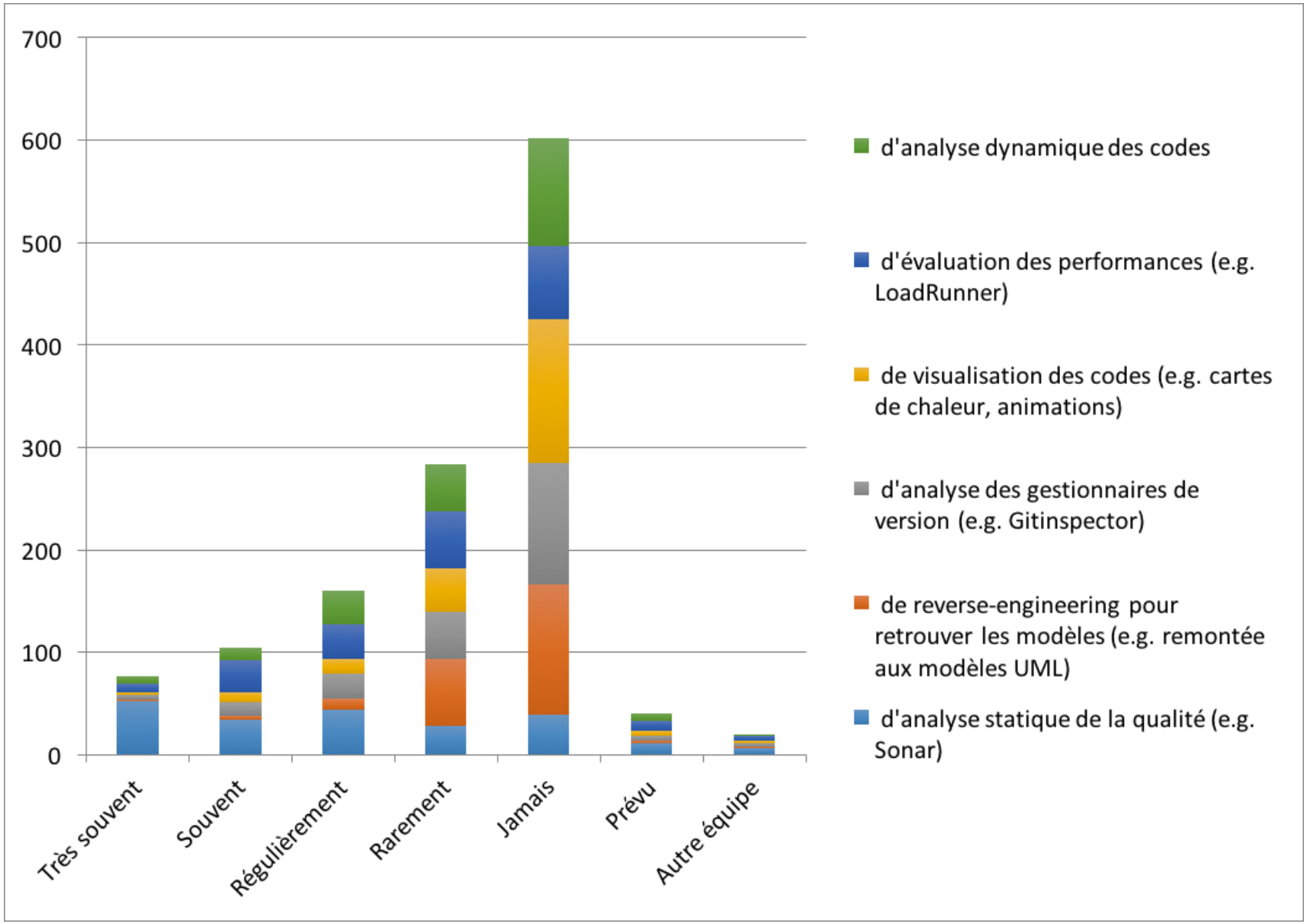


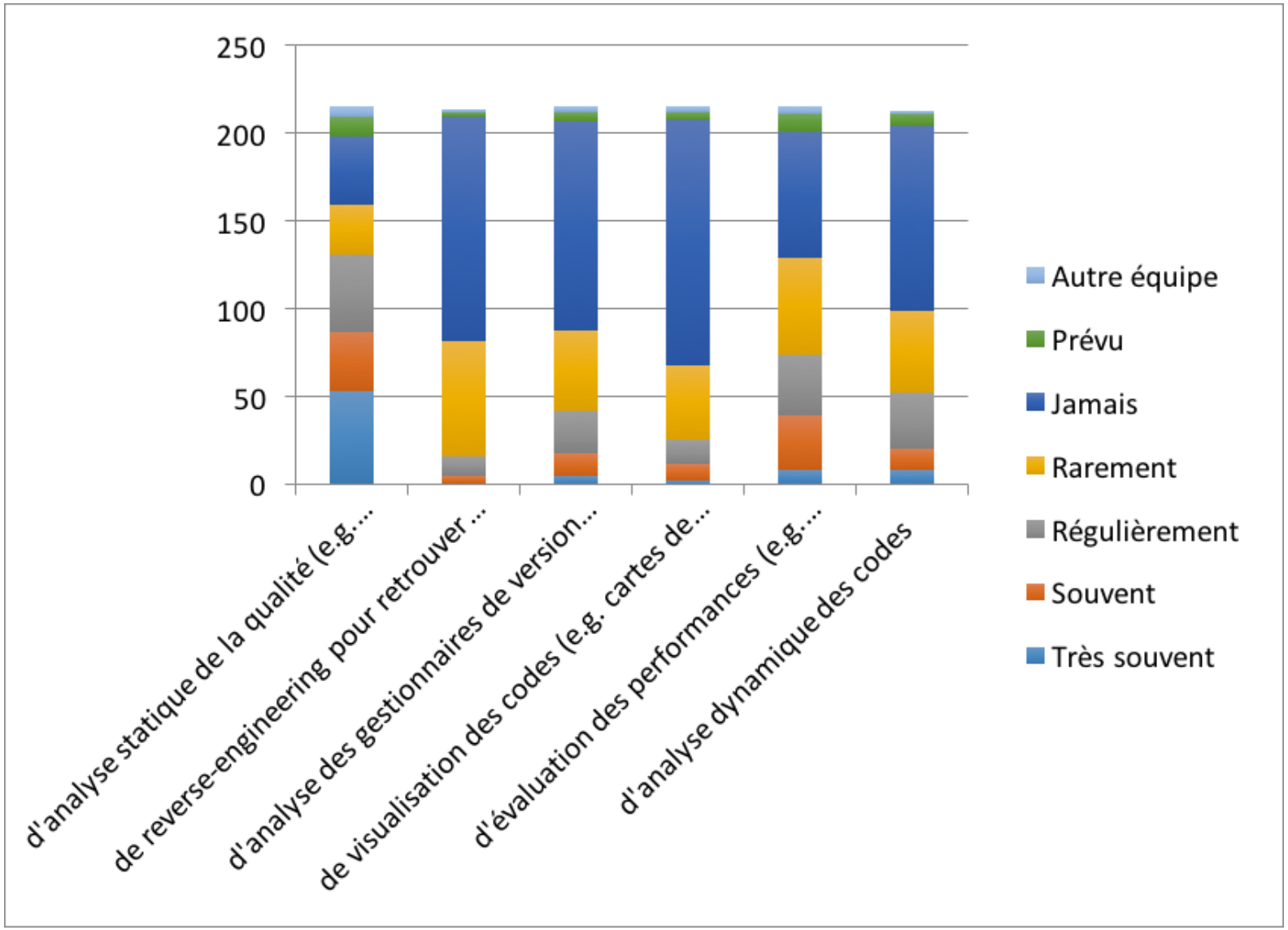
Vous modifiez votre code pour :




Why code maintenance is difficult?
250
200
150
100
50
0
La taille du code et/ou sa complexité
L'architecture mal adaptée aux évolutions
Le manque de documentation
Le manque de tests automatisés
La variété et/ou l'obsolecence des technologies utilisées
Le manque d'expertise des développeurs
Le manque de traçabilité avec les exigences
Le turnover des employés
La chaîne de mise en production
Agree
neutral
Disagree

Les outils que vous utilisez, servent à :


700
600
500
400
300
200
100
0
Très souvent
Souvent
Régulièrement
Rarement
Jamais
Prévu
Autre équipe
d'analyse dynamique des codes
d'évaluation des performances (e.g. LoadRunner)
de visualisation des codes (e.g. cartes de chaleur, animations)
d'analyse des gestionnaires de version (e.g. Gitinspector)
de reverse-engineering pour retrouver les modèles (e.g. remontée aux modèles UML)
d'analyse statique de la qualité (e.g. Sonar)


250
200
150
100
50
0
d'analyse statique de la qualité (e.g....
de reverse-engineering pour retrouver ...
d'analyse des gestionnaires de version...
de visualisation des codes (e.g. cartes de...
d'évaluation des performances (e.g....
d'analyse dynamique des codes
Autre équipe
Prévu
Jamais
Rarement
Régulièrement
Souvent
Très souvent

Les outils que vous utilisez, servent à :

Et alors …

Il serait intéressant de prévoir des études quantitatifs, questionnaires contrôlés, etc.

- Quasi **absence d'utilisation d'outils** sauf la qualité par « Metrics »
- 2 principales difficultés : (i) la **taille** et la complexité des codes et (ii) **l'architecture** mal adaptée
- Importance de la maintenance dédiée à l'ajout de fonctionnalités et la correction de bugs.

→Conduire une réflexion sur la notion de « Métriques »
→Introduire d'autres formes d'analyse
→Travailler sur des projets réels de « grande » taille
→Mettre le focus sur le **rétro-ingénierie**

Le module

Contexte d'enseignement

- Niveau M2 :
  - Formations différentes : étudiants de master et Polytech en Sciences Informatiques.
  - Pré-requis : Avancés en GL et Programmation

=> Hétérogénéité

- 2016-2017 : 23 étudiants (*35 en 2017-18*)…
- Durée 7 semaines avec un créneau de 4h chaque semaine
- Une évaluation en 8^{e} semaine.

=> Durée limitée

Un apprentissage dirigé par des études de cas

- Indispensable de s'impliquer pour apprendre
  - Une étude de cas par groupe de 3 ou 4 étudiants
- Un objectif de « rétro-ingénierie »
  - Intéressant voire ludique
  - Atteignable
  - Formateur
- Savoir poser une question & développer une démarche pour y répondre

▶️ Demander aux étudiants d'élaborer eux-mêmes leur question

Interrogations sur les bonnes pratiques du GL

- Inspirations : Les lois sur l'évolution du logiciel, les principes agiles, …
  - Comment ≪ vérifier ≫ la véracité des lois de Lehman? Quid des changements de méthodologie? …
- Sujets étudiés :
  - *Quels impacts sur la « qualité » des codes des approches test en premier (TDD) ou test en dernier (TL) ?*
  - Peut-on *vérifier la loi de Conway*[1] sur des codes « open-source » ?
  - Y-a-t-il corrélation entre des demandes d'utilisateurs et l'évolution d'un jeu (Terasology)?
  - Comment identifier des dépendances entre composants exprimés dans des langages différents? utilisant des approches événementielles?
  - Y-a-t-il perte de qualité des codes avec les ajouts de fonctionnalités?
  - Qui doit corriger les bugs?
  - How to do feature-driven comparisons ?

1- « les organisations qui définissent des systèmes ... sont contraintes de les produire sous des designs qui sont des copies de la structure de communication de leur organisation. »

Artefacts produits

- Des chapitres d'un livre (gitbooks)
  i. Problématique sous la forme d'une ≪ question bien posée ≫ et incluant un état de l'arté
  ii. la démarche proposée pour répondre à la question (métriques, outils, codes choisis et pourquoi, méthode)
  iii. les résultats brutes (chiffres, visualisations),
  iv. l'analyse des résultats,
  v. la conclusion (analyse critique, perspectives).
- Les codes éventuels (scripts) ou mode d'emplois

- Deux exposés :
  - Présentation des objectifs et des outils utilisés => fertilisation croisée
  - Présentation des résultats

Quels impacts sur la « qualité » des codes des approches test en premier (TDD) ou test en dernier (TL) ?

- « As Test-Driven Development is really driven by the tests, we expect TDD projects to have a high code coverage, of at least 80% and higher than TL projects. This method involves an important refactoring phase, so cleaning the code is an important part of it. Because of this, we expect a better code quality but also more commits about refactoring and less about fixing or patching bugs. »

•Alexandre Cazala <alexandre.cazala@gmail.com>
•Nicolas Lecourtois <lecourtoisn@gmail.com>
•Lisa Joanno <lisa.joanno@gmail.com>
•Pierre Massanès <pierre.massanes@gmail.com>

Quels impacts sur la « qualité » des codes des approches test en premier (TDD) ou test en dernier (TL) ?

- « As Test-Driven Development is really driven by the tests, we expect TDD projects to have a high **code coverage**, of at least 80% and higher than TL projects. This method involves an important refactoring phase, so cleaning the code is an important part of it. Because of this, we expect a **better code quality** but also **more commits about refactoring and less about fixing** or patching bugs. »

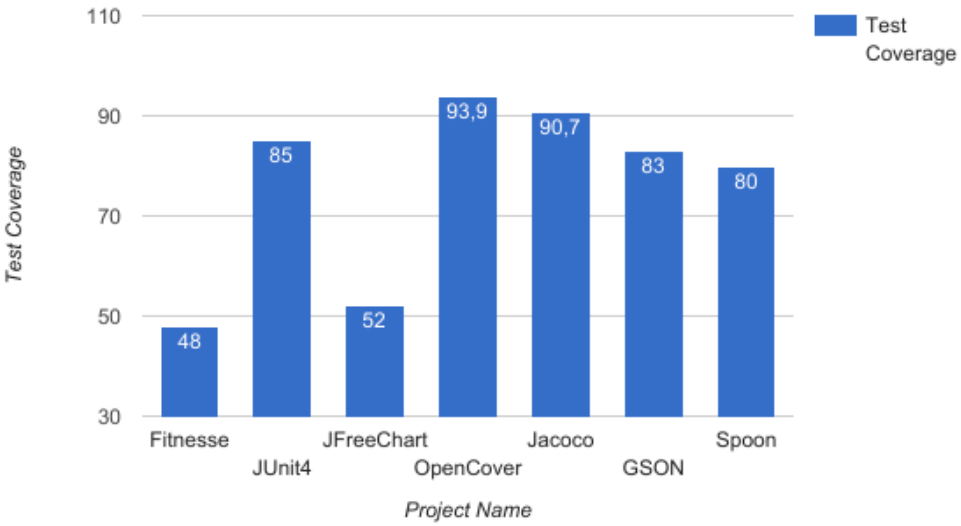
•Alexandre Cazala <alexandre.cazala@gmail.com>
•Nicolas Lecourtois <lecourtoisn@gmail.com>
•Lisa Joanno <lisa.joanno@gmail.com>
•Pierre Massanès <pierre.massanes@gmail.com>

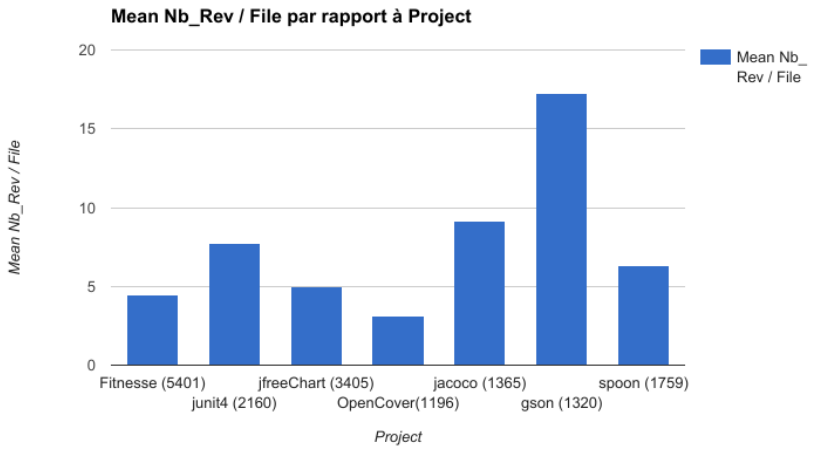
Metrics	Test-Driven Development				Test-Last Development		
	Fitnesse	JUnit4	JFreeChart	OpenCover	Spoon	GSON	JaCoCo
Code Coverage	48%	85%	45%	93.9%	90.7%	83%	80%
Sonar issues	1927	833	5039	286	2341	592	200
Complexity	8612	2061	19323	1568	7635	1945	1962
Code Age	48.3%	21.5%	18.1%	83.6%	9%	50%	35%
Average number of reviews/files	4.46	7.72	4.97	3.15	6.35	17.26	9.12
% "Fix" Commit	6%	7%	1%	7%	30%	6%	2%
% "Refactor" Commit	1%	1%	0%	2%	7%	0%	0%
% "Test" commit	14%	14%	7%	9%	12%	13%	11%

JaCoCo : couverture de tests

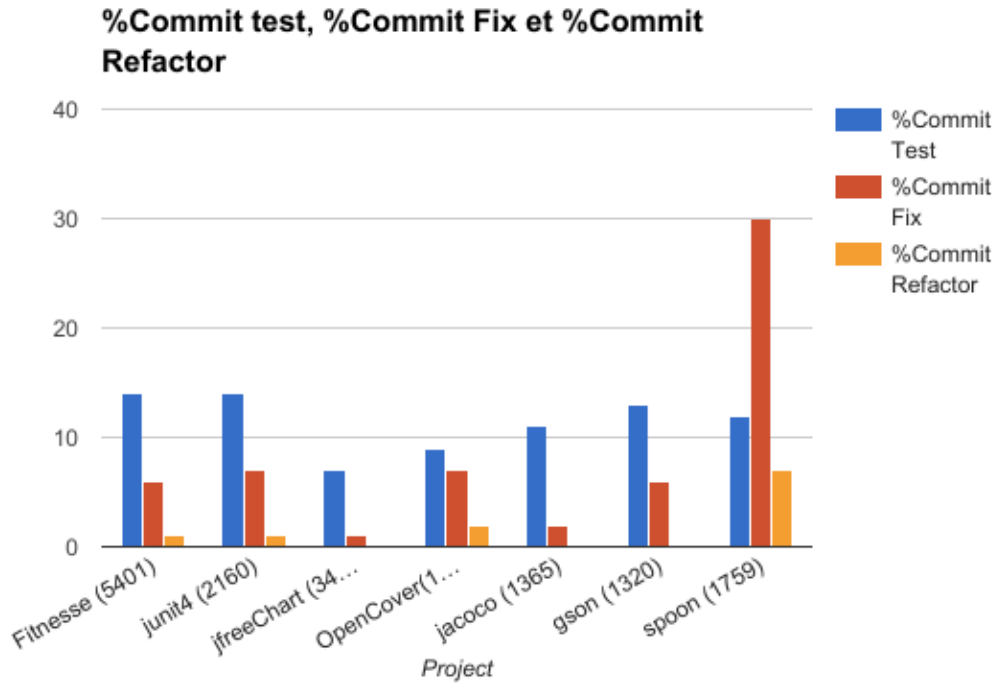
Sonar : issues, métriques et visualisation

Code maat : analyse de dépôts Git





=> le calcul sur Spoon pour les commits "Fix" vient sûrement du fait qu'on utilise le terme "fix" dans les commits pour clore les tickets que l'on ouvre sur Github. @simon.urli



•Alexandre Cazala <alexandre.cazala@gmail.com>
•Nicolas Lecourtois <lecourtoisn@gmail.com>
•Lisa Joanno <lisa.joanno@gmail.com>
•Pierre Massanès <pierre.massanes@gmail.com>

Visualisation : SoftVis3D

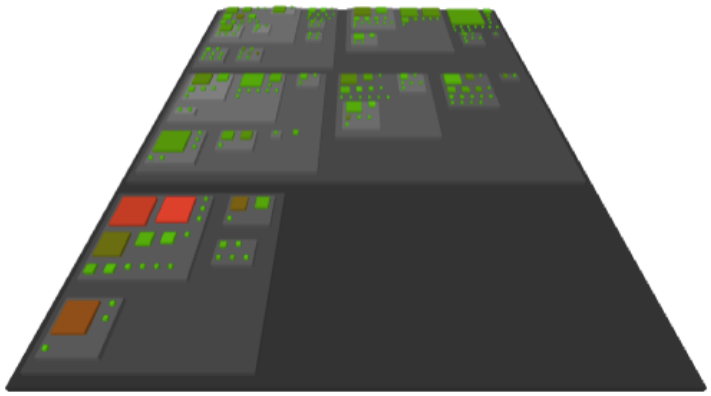


Figure 4 : SoftVis3D results for JUnit4

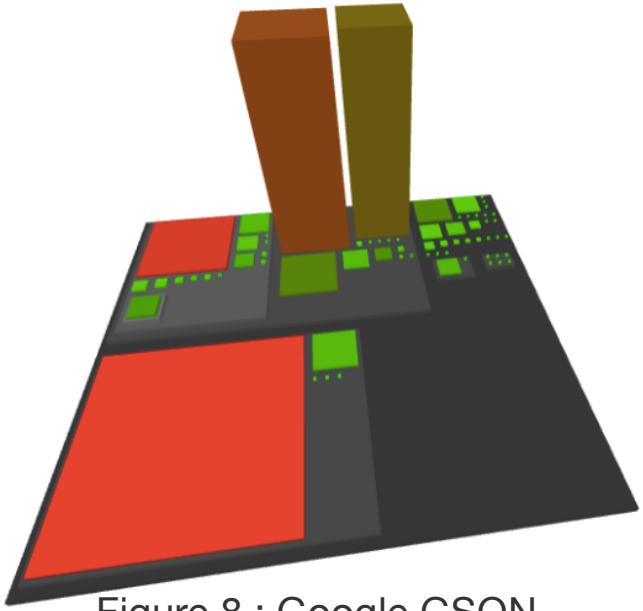


Figure 8 : Google GSON

For our project, we use it to judge the global **cleanliness** of a project. We used the complexity as footprint, the number of duplicated lines as height, and number of Sonar issues as the color.

•Alexandre Cazala <alexandre.cazala@gmail.com>
•Nicolas Lecourtois <lecourtoisn@gmail.com>
•Lisa Joanno <lisa.joanno@gmail.com>
•Pierre Massanès <pierre.massanes@gmail.com>

Rétrospective :
Des projets dirigés ou pas par les étudiants

+ Exploration de questions de GL et de différents outils
+ Motivation, Intérêt, créativité.
− Difficultés
  - à formuler les questions
  - à proposer une méthode d'analyse et à la généraliser
  - à « guider » sans imposer une direction 7 projets « étudiants »
− Mais aussi Déstabilisation de certains groupes.

Rétrospective :
Des projets dirigés ~~ou pas~~ par les étudiants

1. Leur proposer des sujets précis qui pourront servir d'exemple
   - Mais encourager les propositions propres
2. Forcer dès le début la lecture d'un article scientifique
3. Renforcer le « coaching »

Le retour des étudiants aurait été intéressant pour savoir ce qu'ils ont pensé de l'expérience.

Conclusion

- Un enseignement par étude de cas sur des questions liées au GL
- Quelques cours :
  - *Comprendre un logiciel en regardant son « histoire »* par Xavier Blanc
  - *Détection et analyse de l'impact des défauts de code dans les applications mobiles* par Naouel Moha
  - *Spoon*, Sébastien Mosser
- Une première expérience à améliorer
- Quel impact de cet enseignement sur leur appréhension du développement?

http://mireilleblayfornarino.i3s.unice.fr/teaching:reverse:2016

An Appropriate Use of Metrics

Management love their metrics. The thinking goes something like this, "We need a number to measure how we're doing. ***Numbers focus people and help us measure success****." Whilst well intentioned,* ***management by numbers*** *unintuitively leads to problematic behavior and* ***ultimately detracts from broader project and organizational goals.*** *Metrics inherently aren't a bad thing; just often, inappropriately used. This essay demonstrates many of the issues caused by management's traditional use of metrics and offers an alternative to address these dysfunctions.*

MARTINFOWLER.COM