Sonar Smash

Speakers : Helen Wallace (Murex), James Mac Mahon (Murex)
Format : Conférence
Date : 18 avril 2019

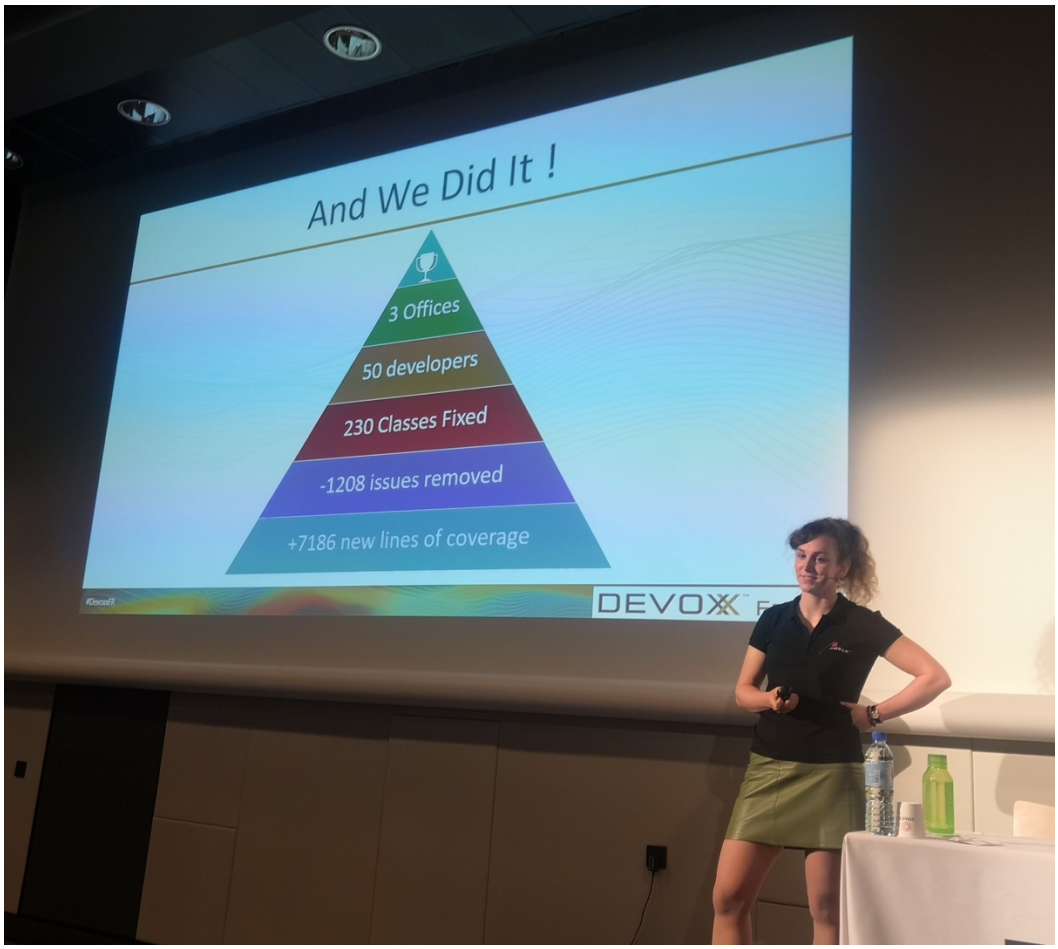
Fuelling the removal of technical debt through competition.

A Murex, la base de code a 15 ans. Le code est vieux, confus, difficile à comprendre malgré la compétence des développeurs.

Problématique : le code legacy (même d'un an) peut ne pas être testé et ne pas être clair.
Ils ont utilisé SonarQube pour détecter les défauts de qualité. D'autres outils sont utilisés : Semmle, JaCoCo (code coverage), Black Duck

Lorsqu'un développeur doit implémenter une nouvelle fonctionnalité, il doit modifier du code existant. Cela ne le rend pas particulièrement heureux. Pour comprendre le code, il veut regarder les tests … mais il n'y en a pas …
Le problème : il y'a de la dette technique.
Corriger une même règle en masse est fastidieux. C'est pourquoi ils ont souhaité faire en sorte que ce soit fun pour les développeurs.
Ils ont donc créé une compétition entre développeurs, avec pour récompense un repas gratuit.
Les développeurs ont formé des équipes.

Et cela a marché. La preuve en quelques chiffres :



Comment créer sa propre compétition ?

- Trouver une récompense
- Avoir du feedback tout au long
- Pouvoir faire tourner les compétitions (pour casser la routine)

Quick win : corriger 3000 jours de technical debt est dissuasif.
Il faut donc d'abord commencer par filtrer ses métriques. Se concentrer par exemple sur les règles portant sur les exceptions.

Outils :

- IntelliJ et Eclipse
- Walkmod : open source tool to fix java coding style issues.
- Detection du code mort

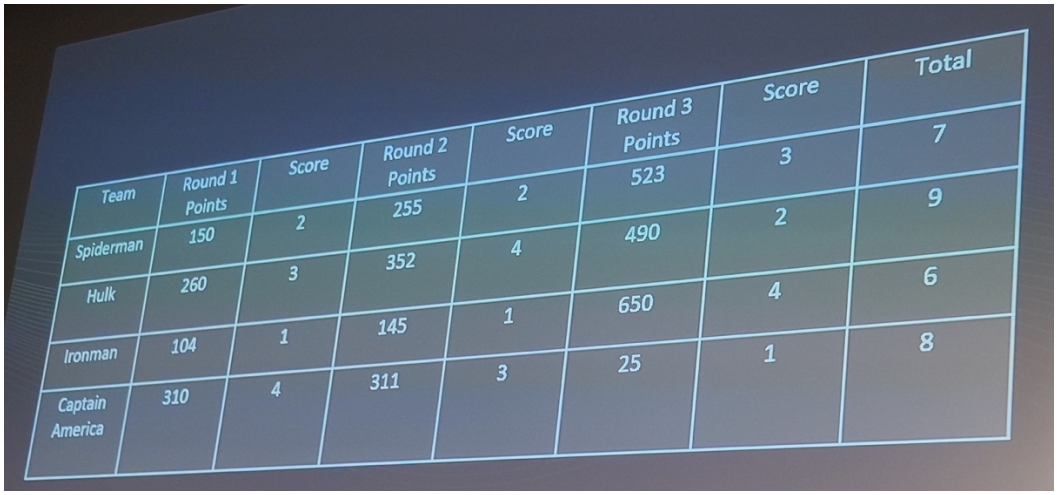
Mécanisme :

- L'approche « Corriger une classe comportant de nombreuses violations de règles » est à privilégier par rapport à corriger toutes les violations d'une même règle sur toutes les classes => se concentrer sur une classe, voir un package
- 2 types d'intervention : corriger les défauts et ajouter des TU

Score calculé sur la base : (issuesRemoved x 5) + (linesOfCode x changeInCoverage x 2)

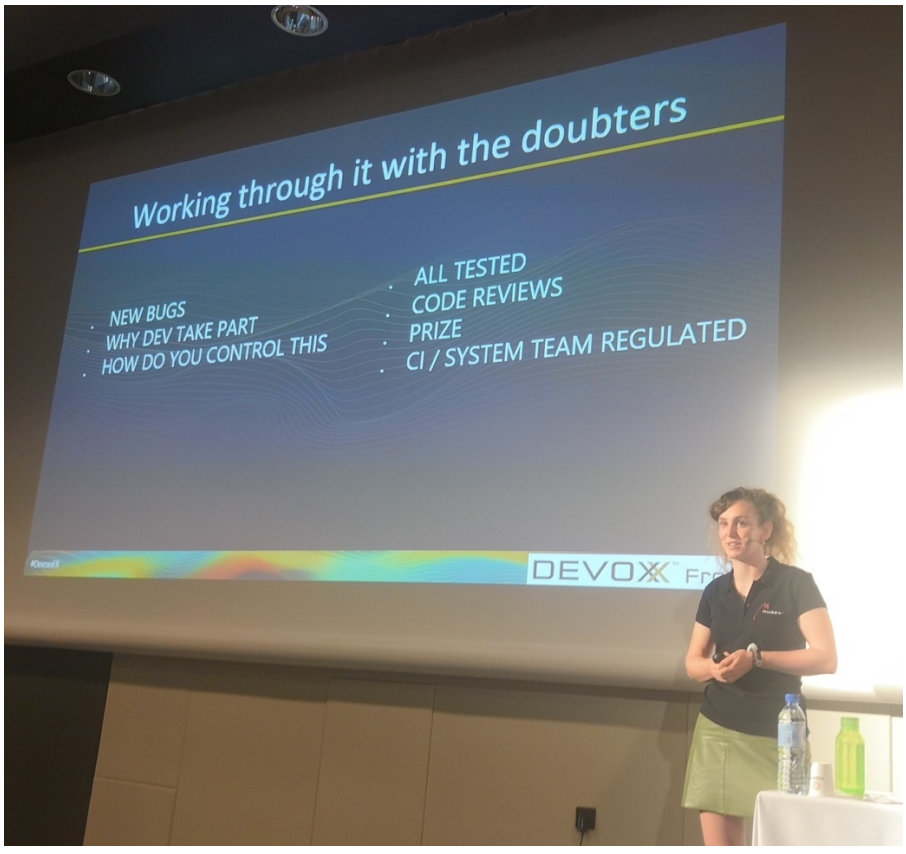
C’est dangereux de corriger tout seul. Cela fonctionne mieux en équipe : mélange de développeurs seniors et de juniors. Fait plus sens.

Tout le monde aime les comebacks. Il ne faut pas abandonner même si la compétition est difficile.



Team	Round 1 Points	Score	Round 2 Points	Score	Round 3 Points	Score	Total
Spiderman	150	2	255	2	523	3	7
Hulk	260	3	352	4	490	2	9
Ironman	104	1	145	1	650	4	6
Captain America	310	4	311	3	25	1	8

A la fin d’un sprint (ex : 2 semaines), celui qui a le plus de points augmente son score de 3 points, l’équipe suivante gagne 2 points. De ce fait, il n’y pas tellement d’écart de points et on peut revenir facilement dans la compétition.
Pour calculer automatique le score, ils utilisent une API proposée par SonarQube mais qui est mal ou peu documentée.



Quand corriger ? par exemple en faisant un break lors de la résolution d'un problème compliqué de threads partis en deadlock.

Motiver les développeurs

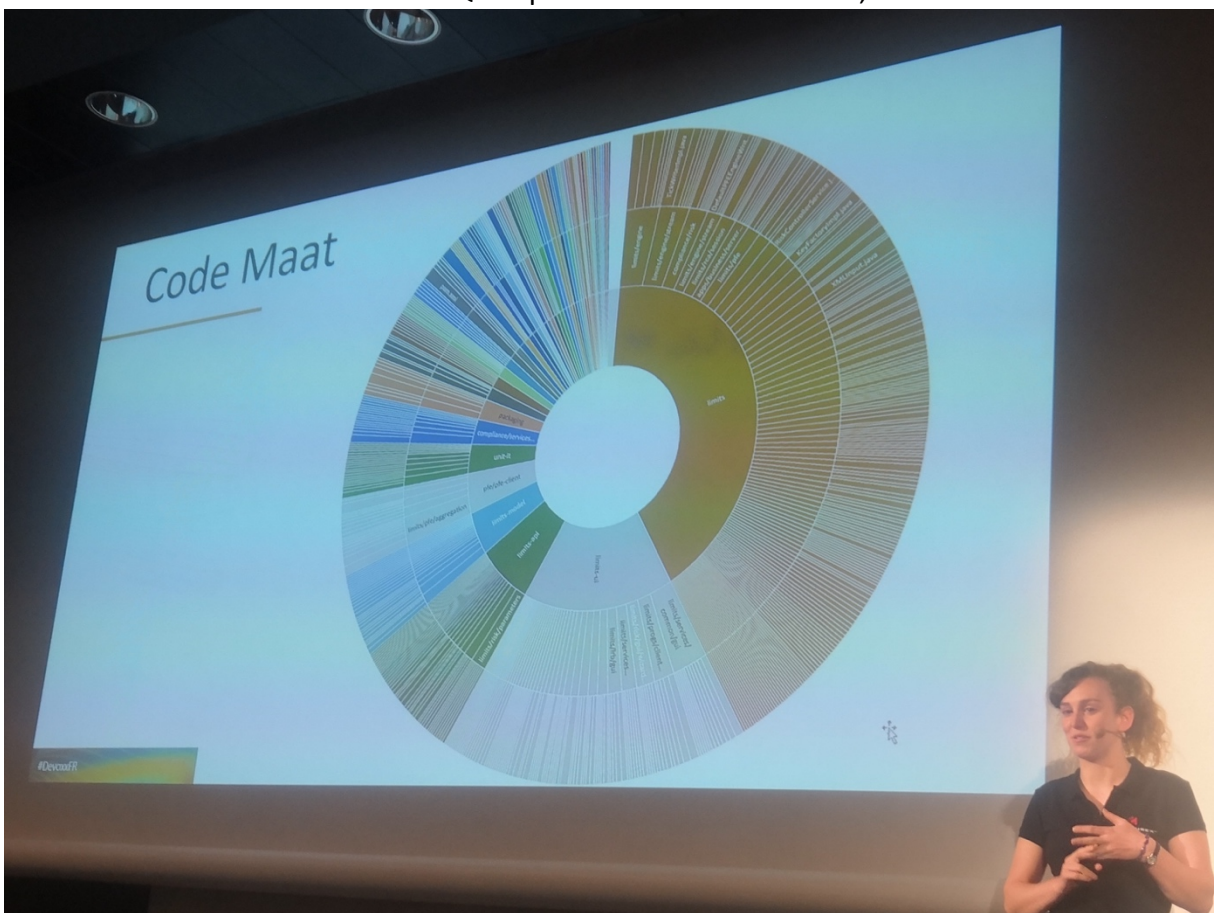
Il va falloir convaincre les développeurs qui doutent de l'intérêt de ce genre de compétition et de l'intérêt de résorber la dette technique. Nécessité d'avoir un avocat pour motiver les dévs.

La motivation est quelque chose de très puissant. Helen prend l'exemple de personnes se prenant en photo avant et après le nettoyage des déchets d'un site / plage.
Prix : free lunch et un trophée
Nécessité de rendre visible la compétition : implication des managers et de l'architecte
Faire partie d'une équipe aide à s'auto-motiver
Les développeurs adore coder.
La compétition doit être fun.

Que souhaitent les développeurs ?
Quitter le boulot à 16h00 le vendredi et aller au pub

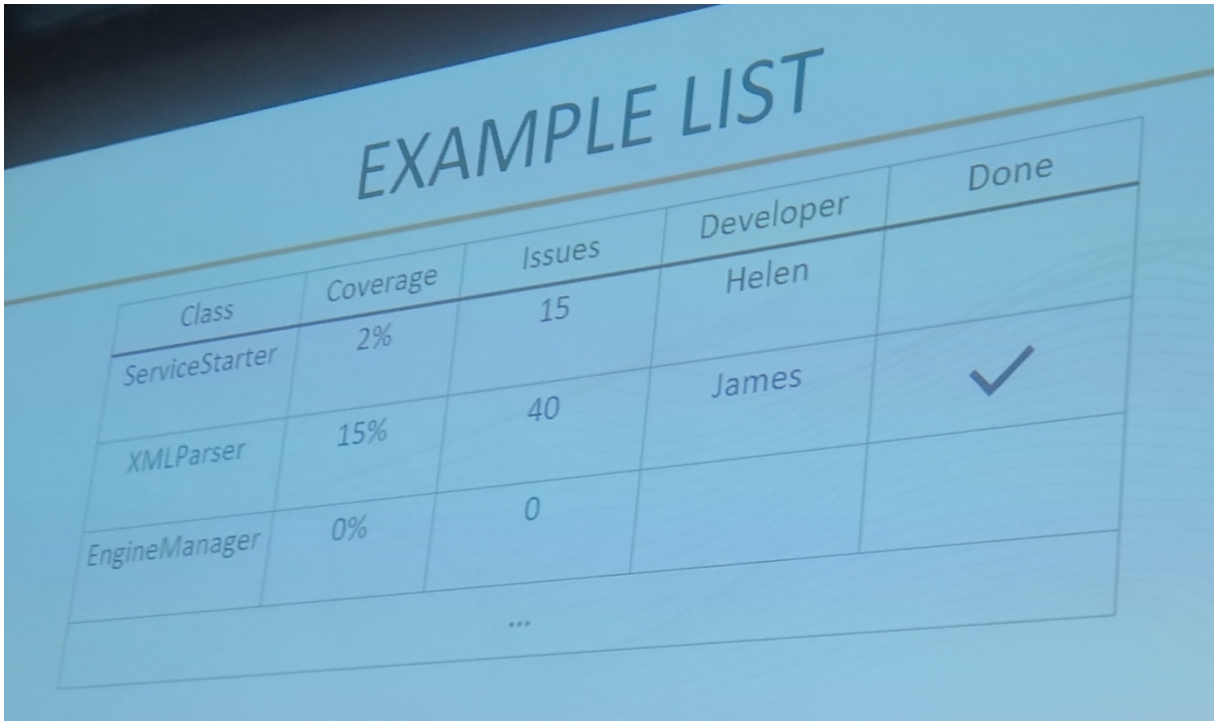
Focusing effort

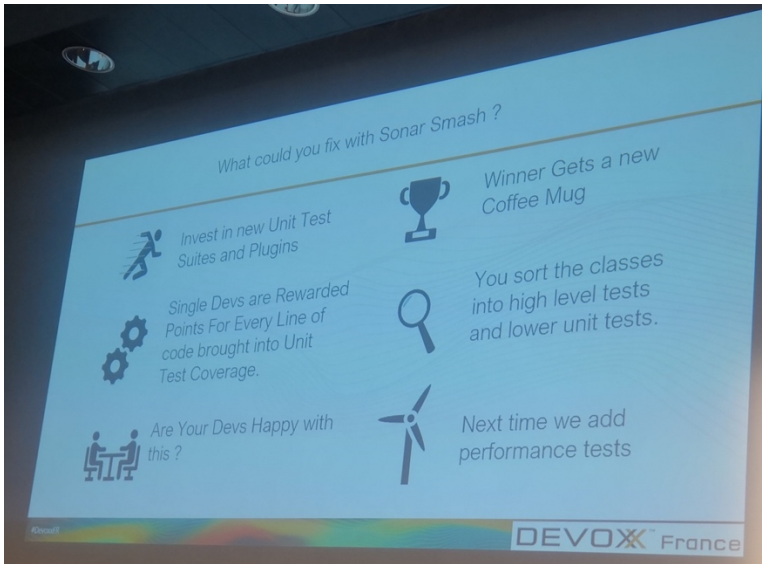
Par où commencer ? utiliser SonarQube pour sélectionner le code, utiliser Code Maat



Difficulté : annoncer aux développeurs qu'il faut recommencer la compétition dès la semaine prochaine.

Les membres non techniques de l'équipe ont souhaité participer.



Les leçons qu'ils ont apprises en organisant et participant à Sonar Smash :

- Les nouveaux tests ont permis de détecter de très vieux bugs
- Plaisir à travailler avec le code qui a été amélioré
- Refactorer permet de s'améliorer en tant que développeur