Nouvelle génération de tests pour projets Java

Speakers : Vincent Massol (xWiki)
Format : Conférence
Date : 20 avril 2018

Vincent Massol travaille chez xWiki et fait partie de l'équipe des Cast Codeurs.
En 1999, Vincent a créé le projet Cactus. 18 ans après, il reste passionné par les tests.

Contexte 1 - XWiki

Les outils/méthodes présentés dans ce talk ont été appliqués sur l'application XWiki.
Application de 14 ans avec 10-15 commiters.
Pour le build, XWiki utilise Maven et les pipelines Jenkins.

Contexte 2 - STAMP

STAMP : Automatic Test Amplification
Projet de recherche Européen de 3 ans
XWiki participe au projet

Etat de l'Art

Sur XWiki :

- 1O414 tests automatisés (2,5h)
    - Unit Test avec Mockito
    - Tests d'intégration
    - Tests Selenium

Amène plusieurs questions :

- Est-ce que ma couverture est suffisante ?
- Comment gérer la compatibilité ascendante ?
- Est-ce que mes tests sont de qualité ?
- Comment tester l'application dans différents environnements de déploiement ?

Couverture de code

- Utilisation de Jacoco et Clover
- Stratégie à « Ratchet effect »
    - Le build Maven échoue si le nouveau code dégrade le taux de couverture de code du code existant. Seuil à paramétrer, que le développeur peut monter au fur et à mesure qu'il ajoute des tests et augment ainsi la couverture de tests.
    - Améliore la couverture de tests
    - A quand même quelques soucis : ne prend pas en compte les tests fonctionnels

- Global Clover TPC : pipeline calculant 1x par mois la couverture globale de tous les projets. Utilise Clover car Vincent n'a pas réussi avec Jacoco.

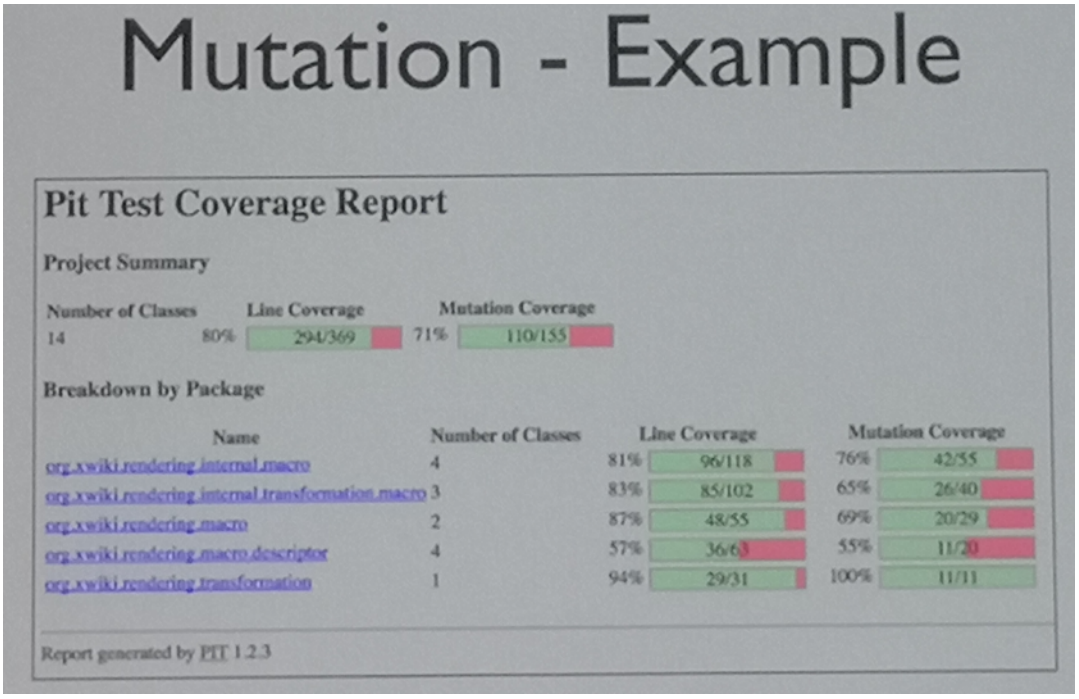
Backward Compatibility

- Pour les APIs (et les SPIs)
- Utilise le plugin Maven Revapi
    - Supporte la compatibilité de source et de binaires
- Stratégie 1 :
    - Casse le build en cas de compatibilité cassée
    - On peut ignorer ce break en ajoutant une ignore list et en prévenant les utilisateurs
- Stratégie 2 :
    - Notion de module legacy
    - Utilisation de l'annotation @Deprecated
    - On change le code pour ne plus utiliser le code marqué @Deprecated
    - Une fois le refactoring réalisé, on déplace le code déprécié dans un module à part
    - Utilisation de AspecjJ pour les modules Legacy pour generate and aspectified API
    - 2 versions de XWiki : avec et sans legacy
- Stratégie 3
    - Difficulté de designer une API stable dès la 1ière fois
    - Ajout d'une annotation @Unstable + @Since sur les nouvelles APIs (pendant 1 an)
    - Le build échoue si @Unstable et pas de @Since
    - Après un an, le build échoue si le @Unstable est toujours là

Mutation Testing

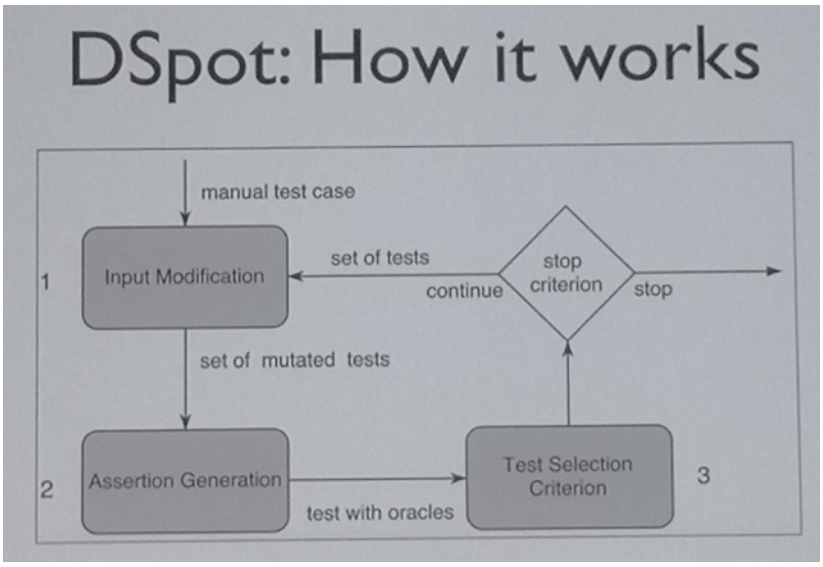
- Stratégie de tests récente
- Utilisation des projets PIT et Descartes (STAMP)
- Concepts
    - Modifier le code testé et exécuter les tests
    - Exemple : on remplace un > par un <
    - Lorsque les tests continuent de passer, c'est qu'ils n'ont pas détecté la mutation.
    - On obtient à la fin un score de mutation
- Projet Descarte
    - Essaie de trouver des mutations efficaces pour accélérer le temps de mutation testing
- Limitations
    - Très lent de faire beaucoup de mutation
    - Pleins de cas ne sont pas intéressants

    - Le support du multi module Maven n'est pas pris en charge
        - Création du projet PITmp
        - 7 heures sur le module xwiki-rendering
- Stratégie XWiki
    - Mettre un seuil dans le build
    - Faire échouer le build si la qualité des tests (mutation score) baisse
    - Le nouveau code doit être de même qualité ou de qualité supérieure au code existant
    - Pour aller plus loin : Faire exécuter les tests que sur le code qui a bougé
- Vincent recommande de n'utiliser la technique du Mutation Testing que si le code source a déjà une bonne couverture de tests



Enrichissement des tests

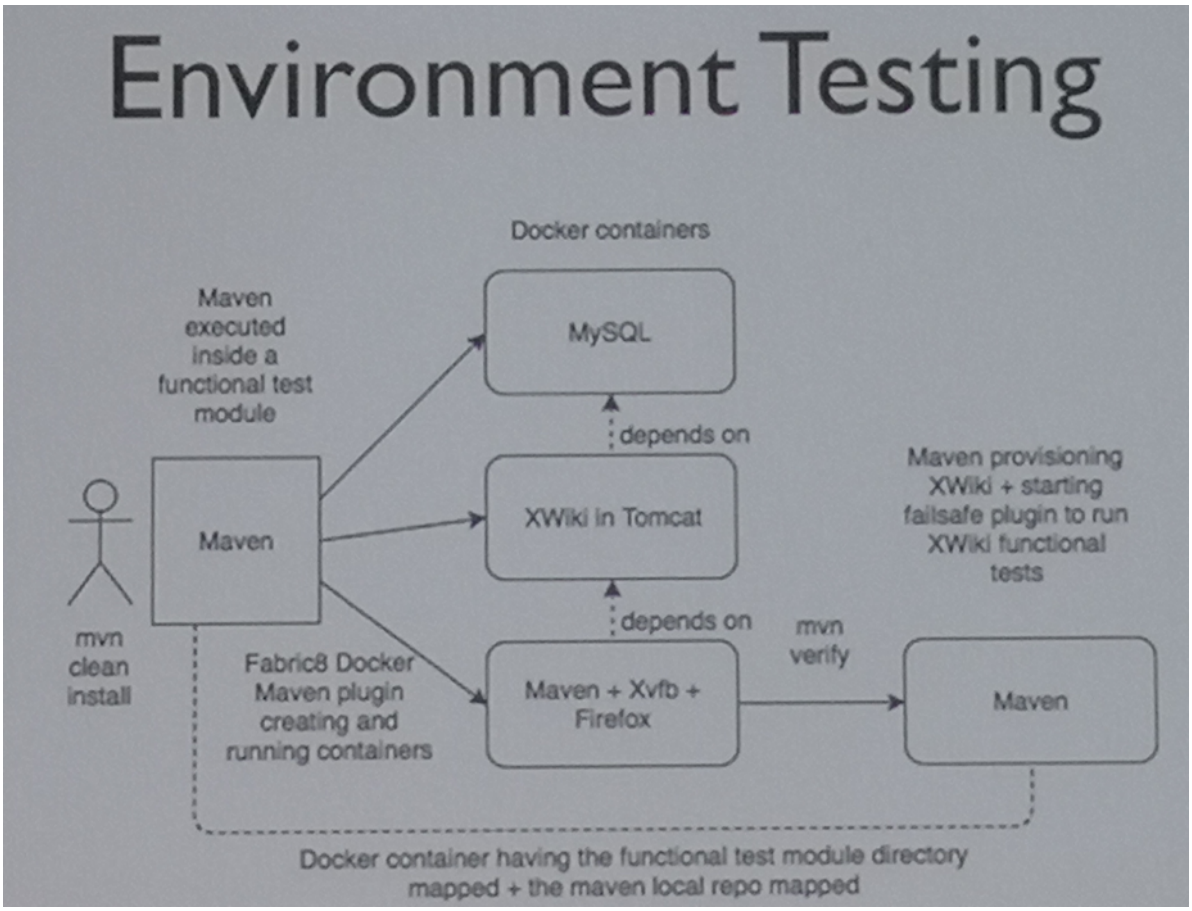
- Utilisation de **DSpot** pour enrichir les tests :
  - DSport capture l'état des objets sur lesquels les asserts sont fait (genre de logger)
  - Il retire les assert
  - Puis il ajoute les assert sur l'état capturé
  - On lance les mutation testing et on conserve les assertions qui tuent le plus de mutants



- Stratégie (pas encore mise en œuvre)
  - DSpot est très lent
  - Le faire exécuter sur le CI de temps en temps
  - Générer les tests dans un autre répertoire

Environment Testing

- Exécuter des tests fonctionnels sur plusieurs environnements :
  - Plusieurs bases : MySQL, Oracle …
  - Plusieurs OS
  - Plusieurs Servlet Containers
  - Plusieurs navigateurs
- Nécessité de lancer ces tests localement
- Utilisation du Docker Maven plugin de Fabric8
- Utilisation en cours de mise en place :
  - Le plugin Maven crée puis exécute les conteneurs
  - Exemple de conteneurs : MySQL, XWiki dans Tomcat, Maven+Xvfb+Firefox
  - Le 3ième conteneur lance un maven verify avec des tests Selenium/WebDriver
- Difficulté à configurer les paramètres mémoires



Autres tests à mettre en place :

- Tests de performance
- Stress tests
- Usability testing