DevOps

Revue de code

Thomas Ropars

Email: thomas.ropars@univ-grenoble-alpes.fr

Website: tropars.github.io

Remerciements

Le contenu de ce cours est très largement inspiré du cours de Eddie Kohler:

- Code reviews

Dans ce cours

- Objectifs de la revue de code

- Conseils et bonnes pratiques

Objectifs de la revue de code dans un projet

Amélioration de la qualité du code

- Assurer le respect des standards/conventions
- Proposer des solutions alternatives
- Identifier des bugs
- Proposer des optimisations

Impact sur le travail en équipe

- Permettre à tout le monde de suivre les évolutions du projet
    - Assurer que tout le monde à une compréhension globale
    - Éviter les redondances
- Créer une dynamique de travail en équipe

Guide de bonne conduite

- **Pensez aux conséquences possibles de vos commentaires**
    - Sur le code
    - Sur le temps et la motivation des contributeurs

- **Concentrez-vous sur ce compte**
    - les questions de goûts sont secondaires
    - Des outils existent pour vérifier/corriger les problèmes stylistiques

Des conseils

Pour tout le monde

- Relire les choses importantes, laisser des outils faire le reste
- Tout le monde doit participer aux tâches de relecture de code
- Tout le code doit être relu
- Adopter une attitude **positive**

Pour les relecteurs

- Faire des revues de code souvent, mais avec des sessions courtes
- Ce n'est pas un problème de dire que tout va bien
- Utiliser une checklist

Pour les contributeurs (ex: créateurs d'une *merge request*)

- Soumettre un code court
- Fournir des éléments de contexte

Checklist

Général

- Est-ce que le code fonctionne? Est-ce qu'il répond à l'objectif? Est-ce que la logique est correcte?
- Le code est-il facile à comprendre?
- Le code est-il conforme aux conventions?
- Y a-t-il du code inutile/redondant?
- Le code est-il suffisamment modulaire?
- Y a-t-il du code introduit pour débugger qui devrait être supprimé?

Robustesse

- Les données d'entrées sont-elles vérifiées?
    - Type, taille, format, valeur
- Les erreurs et exceptions sont-elles traitées?
- Le cas de valeurs non valides pour les paramètres est-il traité?

Checklist (suite)

Documentation

- Le travail a-t-il été commenté? Les commentaires décrivent-ils les intentions?
- Toutes les fonctions sont-elles commentées?
- La prise en charge de cas pathologiques est-elle documentée?

Tests

- Est-ce que du code de test a été fourni?
- Les tests unitaires vérifient que le code répond à l'objectif?

Actual Studies

- Average defect detection rates
  - Unit testing: 25%
  - Function testing: 35%
  - Integration testing:45%
  - **Design and code inspections: 55% and 60%.**

- 11 programs developed by the same group of people
  - First 5 without reviews: average 4.5 errors per 100 lines of code
  - Next 6 with reviews: average 0.82 errors per 100 lines of code
  - Errors reduced by **> 80 percent.**
    - IBM's Orbit project: 500,000 lines, 11 levels of inspections. Delivered early and 1 % of the errors that would normally be expected.
    - After AT&T introduced reviews, study with > 200 people reported a +14% productivity, -90% defects.
      - (From Steve McConnell's *Code Complete*)

7

Credits: M. Stepp

Études sur l'utilité des revues de code

Étude auprès d'employés de Google

- Le détection de problèmes est importante mais pas centrale
    - Même si important pour éviter des problèmes majeurs
- La revue de code améliore la clarté et la maintenabilité du code
- Aspect *éducatif* important
    - Conventions à suivre
    - API à utiliser
    - etc

Étude auprès d'employés de Microsoft

- Conclusions similaires
- Retours utiles pour repenser la conception d'une solution (même si la solution est déjà correcte)

Crédits: Étude @Google et Étude @Microsoft

Références

- Les notes de cours de E. Kohler
- Les notes de cours de M. Stepp
- La checklist de Frog Creek
- Une liste très complète de ressources sur le sujet