LOG8371 : Ingénierie de la qualité en logiciel

Système de contrôle de version et révision de code

Hiver 2017

Fabio Petrillo
Chargé de Cours

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Système de contrôle de version

- Un système de contrôle de version (VCS) est une méthodologie ou un outil qui vous permet de suivre les changements que vous effectuez aux fichiers de votre projet.
- VCS suit tous les changements pour nous, en conservant une copie de chaque modification apportée au code dans nos projets.
- une sauvegarde pour garder notre code source
- Le centre de dépôt (repository) est l'endroit où le système de contrôle de la version permet de suivre toutes les modifications que vous effectuez.
- Chaque changement de code dans un VCS - comme un dépôt de code est appelé un commit
- VCS centré (Subversion, CVS) -> un seul centre de dépôt (distant)
- Distribué VCS (Git, Mercurial) -> chacun possède son propre centre de dépôt qui a l'histoire complète du projet.

Avantages du contrôle de la version distribuée

- Peut collaborer sans autorité centrale
- Opérations déconnectées
- Facile à ramifier et à fusionner
- Définissez votre propre flux de travail
- Outils puissants de partage communautaire
- Un chemin plus facile à contribuer à la commitance

DVCS et communauté de Build

- Les développeurs peuvent facilement découvrir et forger des projets. En plus, il est simple pour les développeurs de partager leurs changements

“Le contrôle de la version distribuée vise à responsabiliser votre communauté et les personnes qui peuvent rejoindre votre communauté” - Mark Shuttleworth

Pourquoi le contrôle de la version?

VCS est devenu **essentiel** au développement de logiciels parce que:

- Ils permettent aux équipes de collaborer
- Ils gèrent les changements et permettent l'inspection
- Ils suivent la propriété
- Ils suivent l'évolution des changements
- Ils permettent la ramification
- Ils permettent une intégration continue

Comment fonctionne un DVCS tel que Git?

- Un DVCS fonctionne généralement au niveau d'un changeset
- Logiquement, un référentiel est constitué d'un premier vide
- État, suivi de plusieurs changesets
- changeset sont identifiées par une valeur de hachage SHA-1
- e.g., 0878a8189e6a3ae1ded86d9e9c7cbe3f

Que doit contenir un message de journal de validation?

- Crafting le message parfait d’un log est quelque chose d'art.
- Écrivez une simple et seule ligne, qui explique le commit, puis ajouter quelques phrases complètement pour expliquer votre commit.
- Gardez à l'esprit lorsque vous rédigez votre message de commit celui qui va le lire.
- Humains et analyseurs (systèmes de suivi des problèmes ou outils de dépôt de versions)
- Si un programme va lire vos messages de log, vous devez vous assurer d'inclure toutes les informations qu'il expecte. Il existe des outils permettant de lire les log de commit et de mettre à jour les outils tiers tels que les systèmes de suivi des tickets pour afficher l'activité des tâches.

Branches

- La version actuelle de votre repository est simplement un pointeur
- À la fin de l'arbre
- Le "trunk" par défaut de Git s'appelle "master"
- La pointe de la branche actuelle s'appelle "HEAD"
- Toute branche peut être désignée par son identifiant SHA-1
- La création de branches dans un DVCS est rapide, vous indiquez simplement un élément différent de l'arbre sur le disque

Quand créer une branche?

Changements expérimentaux: Créez une nouvelle branche pour faire votre travail, et vous pouvez y travailler séparément de toute modification qui est déployée immédiatement.

Nouvelles fonctionnalités: Chaque fois que vous commencez à travailler sur une nouvelle fonctionnalité, créez une nouvelle branche.

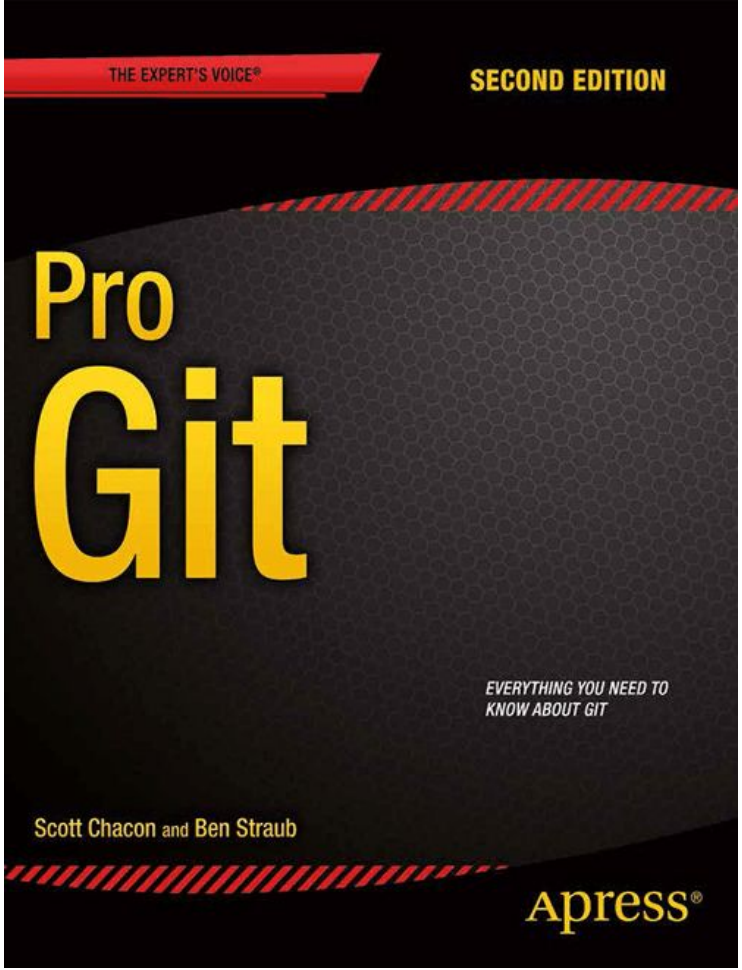
Correction de bugs: Créez une branche pour suivre vos modifications sur ce bug.

Fusionner (Merging)

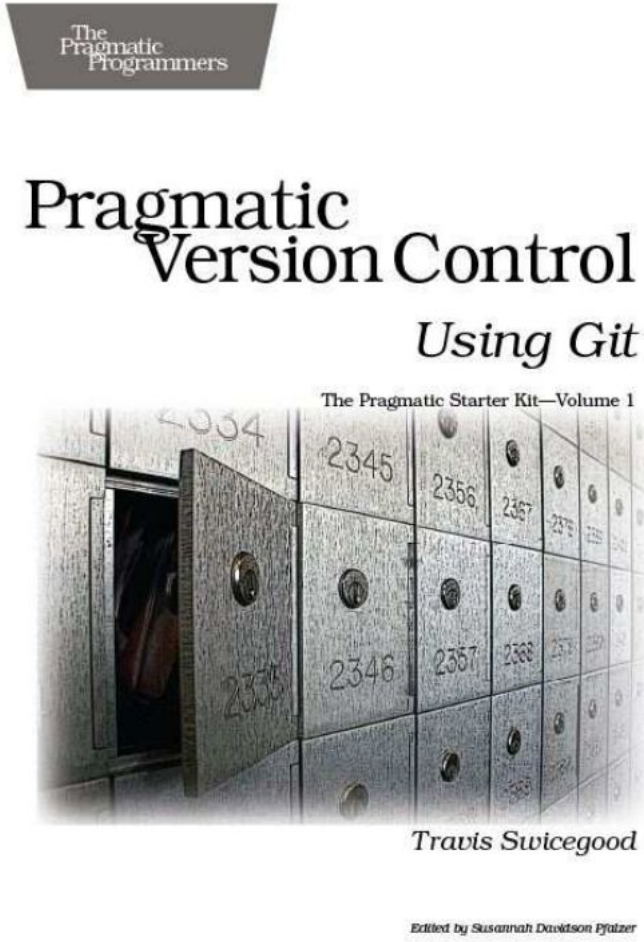
- DVCS consiste à fusionner
- Les fusionnements ne sont que le tissage de deux (ou plus) branches locales en un seul
- Cependant, contrairement à CVCS, vous n'avez pas à préciser de quoi vous fusionnez et vers quoi; Les arbres savaient automatiquement quel était leur point de partage dans le passé, et ils pouvaient le résoudre à partir de là.
- La fusion est beaucoup plus facile dans un DVCS comme Git

Pulling and Pushing

- Nous n'avons pas parlé de la nature distribuée de DVCS
- Les changements circulent entre les dépôts par push et pull
- Puisque qu'un arbre DVCS est simplement un pointeur vers une branche ....
- Il y a trois cas à considérer pour comparer deux arbres:
  - Votre pointe est un ancêtre de l’actuelle
  - Mon pointe est un ancêtre de la votre
  - Aucun de nos pointe ne sont les ancêtres directs; Cependant, nous partageons tous deux un ancêtre commun



https://git-scm.com/book/en/v2

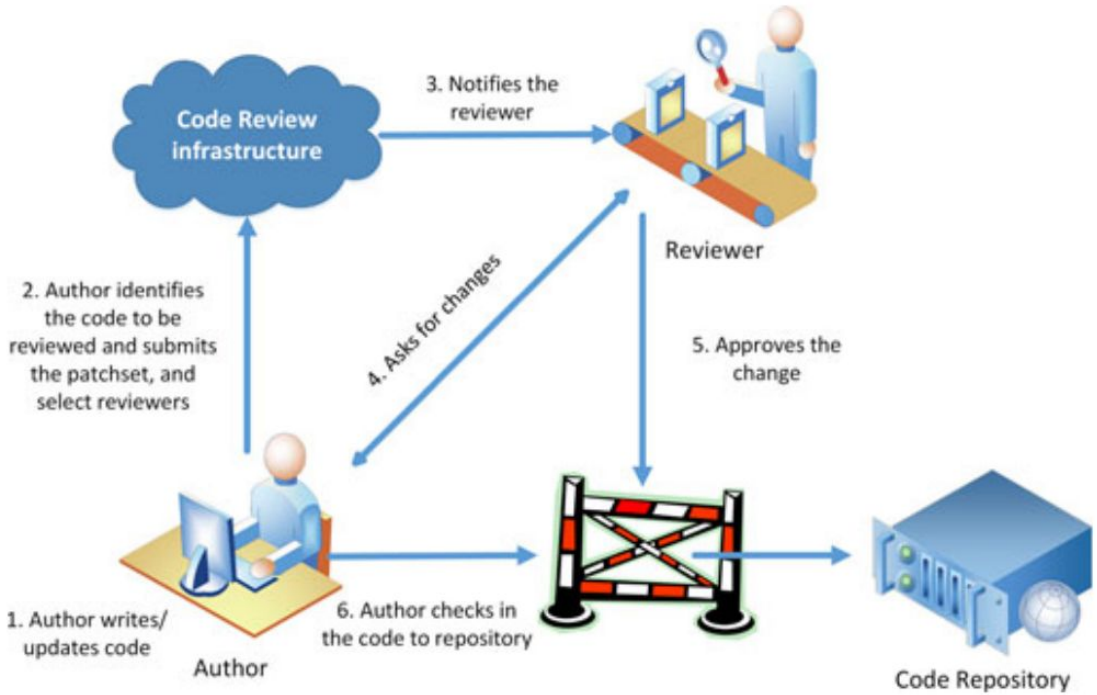


Examen de code

Qu'est-ce que l'examen de code?

- Lorsqu'un développeur écrit un code, un autre développeur est invité à passer en revue ce code
- Une critique approfondie ligne par ligne
- Se produit dans un contexte non menaçant
- L'objectif est la coopération, pas la recherche de pannes
- Souvent une partie intégrante du processus de codage
- Déboguer le code erroné de quelqu'un d'autre
    - Examen de code involontaire: pas si bon; Les émotions peuvent évaser

http://code.google.com/p/rietveld/downloads/detail?name=Mondrian2006.pdf

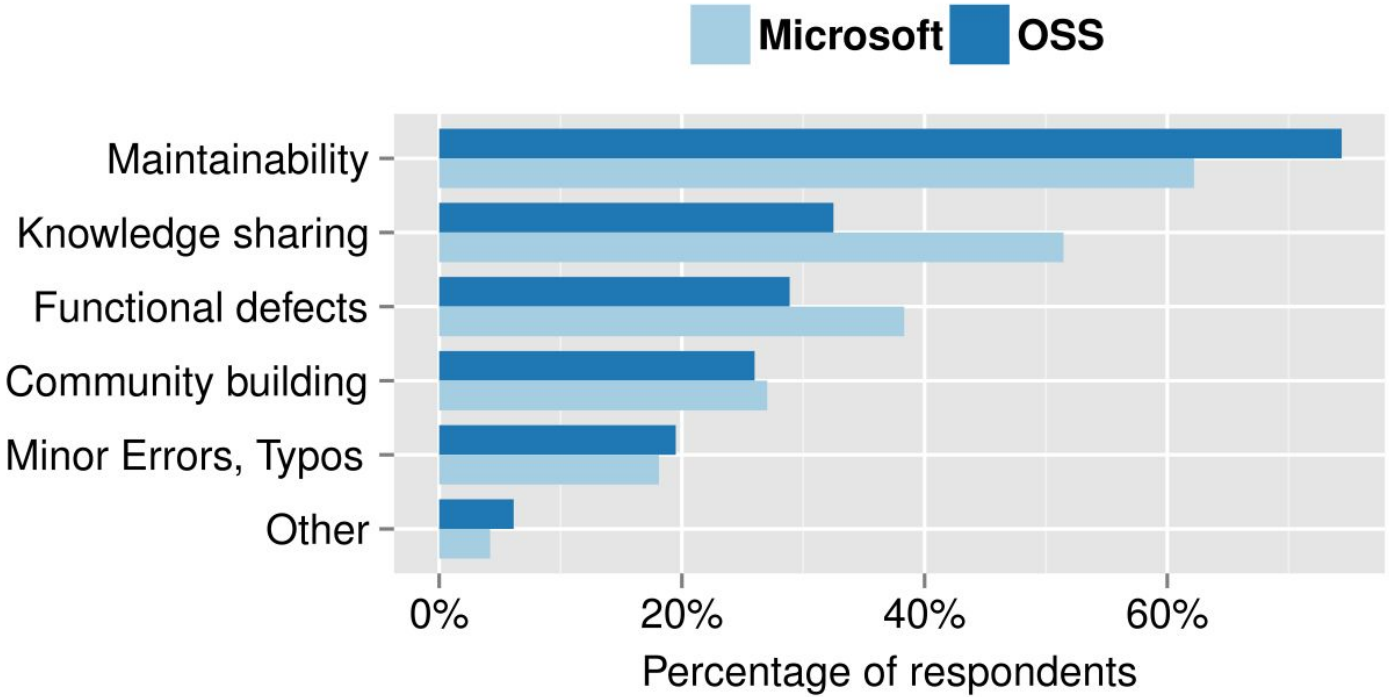


A. Bosu, J. C. Carver, C. Bird, J. Orbeck and C. Chockley, "Process Aspects and Social Dynamics of Contemporary Code Review: Insights from Open Source Development and Industrial Practice at Microsoft," in *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 56-75, Jan. 1 2017.

Pourquoi l'examen de code?

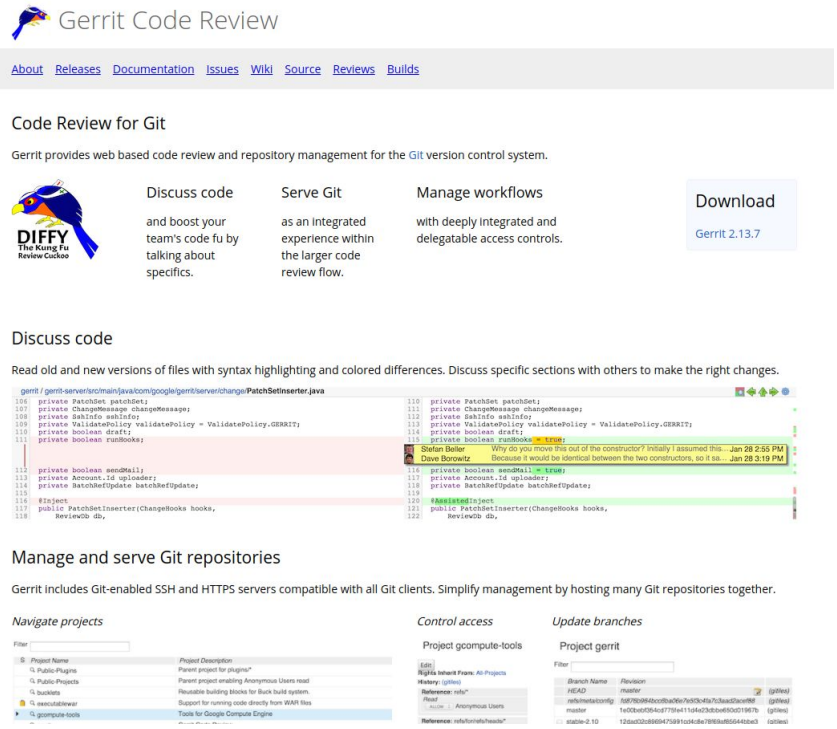
- Quatre yeux attrapent plus de bugs: capturez les bogues tôt pour économiser des heures de débogage
- Appliquer les normes de codage: garder la lisibilité globale et la qualité du code élevée
- Mentorat de nouveaux développeurs: Apprenez des erreurs sans diviser l’ équipe.
- Établir des relations de confiance: préparer plus de délégation
- Bonne alternative à la programmation par paire: asynchrone et à travers les lieux
- Propriété de code collectif
  - Toute l'équipe est responsable d'un changement
  - Déclencher la discussion sur le code et l'architecture
- Construire un modèle de confiance dans l’équipes
  - Déléguer le processus d'approbation
  - Autoriser la «promotion des rôles élus» et assurer que les gens se sentent engagés

Pourquoi l'examen de code?



A. Bosu, J. C. Carver, C. Bird, J. Orbeck and C. Chockley, "Process Aspects and Social Dynamics of Contemporary Code Review: Insights from Open Source Development and Industrial Practice at Microsoft," in *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 56-75, Jan. 1 2017.

Examen du Code Gerrit



https://www.gerritcodereview.com/

Examen du Code Gerrit

- Gerrit fournit une analyse de code basée sur le Web et une gestion de dépôt pour le système de contrôle de version Git.
- Discuter le code
- Gerrit fournit un serveur git intégré
- Contrôle d'accès et flux de travail
- Gerrit a été développé chez Google par Shawn Pearce (fondateur de JGit) pour le développement du projet Android.
- À partir d'un ensemble de patches pour Rietveld, également un outil d'analyse de logiciel, il est devenu une copie et s'est transformé en un projet distinct lorsque les patches ACL ne seraient pas fusionnés dans Rietveld par son auteur, Guido van Rossum.

Une branche Une caractéristique

- La branche Master contient uniquement les modifications révisées et approuvées: le master passe du bon au meilleur état après chaque changement (approuvé).
- Chaque branche de fonction est basée sur une branche principale: point de départ stable
- Un changement peut vraiment être abandonné car
  - Aucun autre changement approuvé ne peut dépendre d'un changement non encore approuvé
  - Gerrit rejettera automatiquement un changement successeur d'un changement abandonné

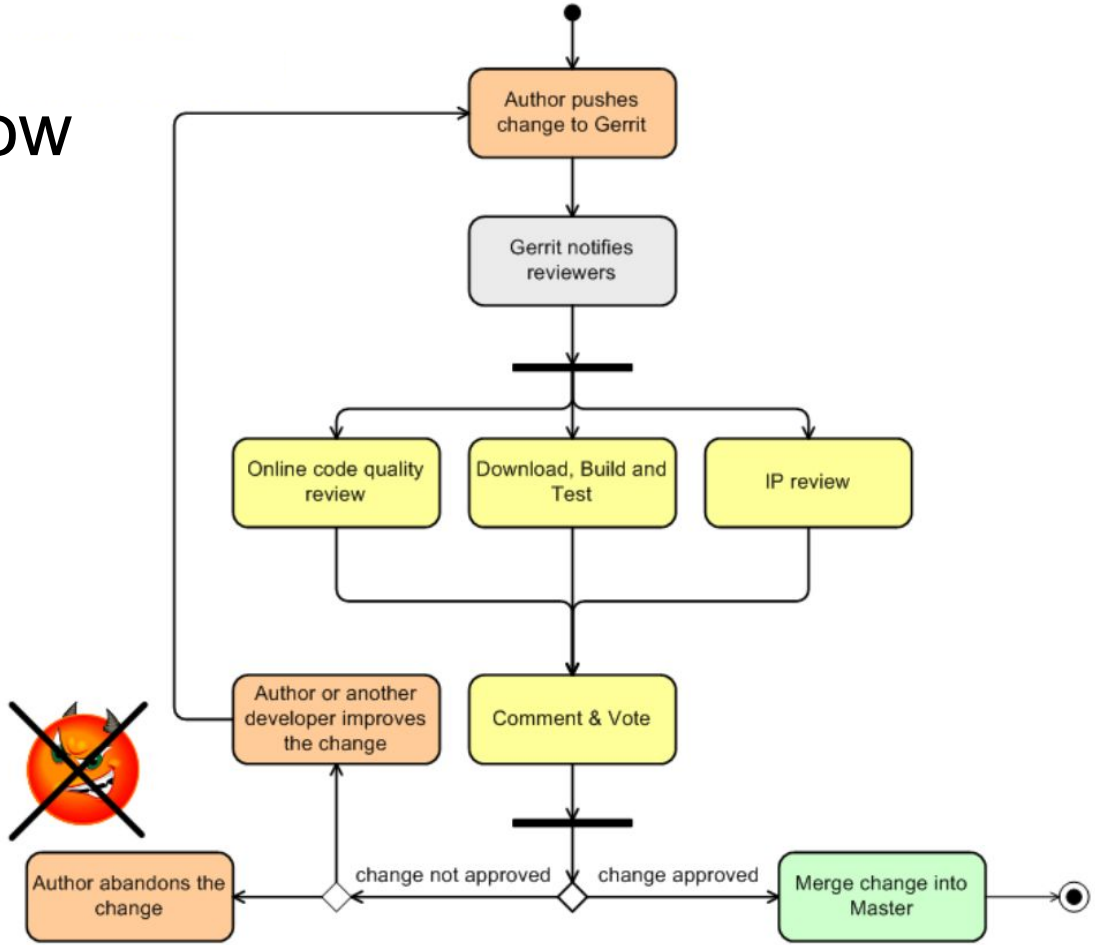
Gerrit: Roles de Project

- **Contributor**: Proposer des changements et de nouvelles idées
- **Reviewer**: Discuter et voter: -1 (choses à améliorer) a +1 (semble bon)
- **Committer**
  - Membre permanent du projet
  - Valider les changements, de-1 (Ne fonctionne pas) to +1 (fonctionne)
  - Peut utiliser veto (-2) ou approver (+2) a changer
  - Peut fusionner le code dans les branches du projet

Glossaire de la revue de code

- **Change**: Commit soumis pour examen
- **Patch Set**: Ensemble de modifications apportées à un changement
- **Review**: La notation d'un changement basé sur un ensemble de patch
  - -2: veto, Ne fusionnez pas
  - -1: it's OK, Plus de patchs sont nécessaires
  - 0: Juste des commentaires, pas de score
  - +1: C'est bon, mais j'ai besoin de quelqu'un d'autre pour le regarder
  - +2: il est parfait, s'il vous plaît fusionner
- **Verified**: Vérification si un changement fonctionne ou non
- **Submit change**: Soumettre un changement à travers l'interface graphique
- **Merge change**: Accepte un changement et passe à travers Git
- **Abandon change**: Ne fusionnez pas le changement et rangez le indéfiniment

Gerrit Workflow



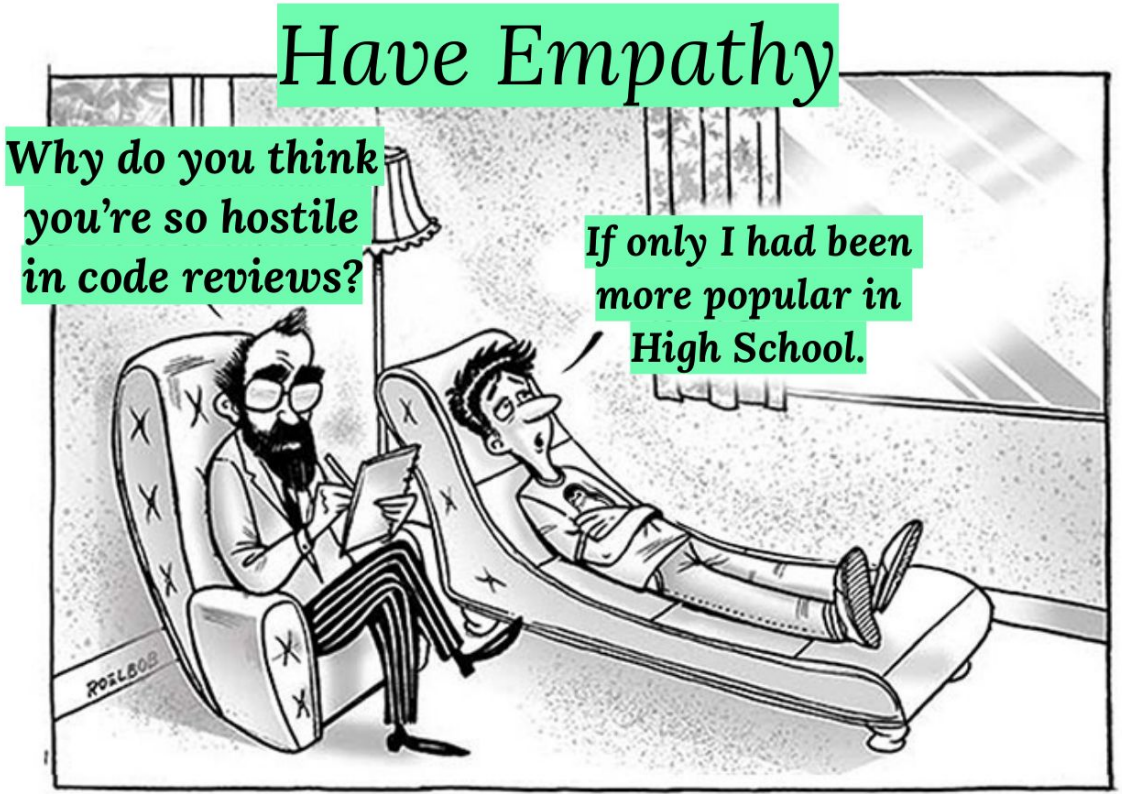
Frustrations de révision du code

- Ajoutez une demande de temps
- Ajoutez un processus
- Peut entraîner des tensions d'équipe
- Les développeurs «intelligents» pensent qu'ils n'en ont pas besoin

Comment développer une culture d'examen de code?

- S'engager à **toujours** examiner avant de fusionner
- Présentez les faits
- Ne le forcer pas
- De nouvelles idées prennent du temps au root
- Les codes d'évaluation doivent être universels
  - Peu importe la façon dont Senior / Junior vous êtes
  - Seuls les développeurs seniors examinent == bottleneck
  - L'inégalité engendre l'insatisfaction
  - Décomposer les anciennes barrières et les quo de statut
- L'examen du code est effectué par vos peers, et non par la gestion.
- L'échec est inévitable
- Les erreurs deviennent équipe, pas de responsabilité individuelle

Comment être un excellent examinateur de code



Comment être un excellent examinateur de code

- Soyez objectif
  - “Cette méthode manque d'un docstring”
  - ~~“Vous avez oublié d'écrire un docstring”~~
- Posez des questions ne donne pas de réponses
  - “Est-ce qu'il serait plus logique si... ?”
  - “Qu'est-ce que vous pensez a propos... ? ”
- Offrir des suggestions
  - “Il **pourrait** être plus facile de”
  - “Nous avons **tendance** à le faire de cette façon”
- Évitez ces termes comme: simplement, facilement, juste, évidemment, bien en fait...
- Complétez le bon travail et les bonnes idées
- Ne soyez pas un perfectionniste
- Sauvegardez le "cosmetiques" pour le dernier, après l'impression d'une architecture, d'un design, ou d'autres problèmes importants à grande échelle

Comment être un excellent examinateur de code

- Évitez Burn Out
- Compléter dans 24-48 heures
- Définition terminé

Prochain cours
Construction, intégration continue, livraison continue