Engineer

Revue de code

~~ɓə~~ faire relire un changement par un tiers avant intégration

Ce que c’est

Une revue de code est l’étape où une personne autre que l’auteur lit un changement avant qu’il n’intègre la base de code. Le guide des pratiques d’ingénierie de Google la définit comme l’examen du code par toute personne autre que celles qui l’ont écrit, et le chapitre consacré au sujet dans *Software Engineering at Google* ajoute que cela se produit souvent avant l’entrée du code dans la base. La plupart des équipes parlent de la pratique au pluriel, les revues de code ; les noms changent avec l’outil. Sur GitHub, c’est une revue de pull request, chez Google une revue precommit, et dans le noyau Linux elle se déroule par courriel, où une personne satisfaite du patch propose une étiquette Reviewed-by.

Le guide de Google énumère ce qu’examine la personne chargée de la revue : conception, fonctionnement, complexité, tests, nommage, commentaires, style et documentation. Derrière chaque mot se trouve une question. Le code fait-il ce que son auteur voulait, et est-ce bon pour les personnes qui l’utilisent ? Pourrait-il être plus simple ? Existe-t-il des tests automatisés, et sont-ils corrects et raisonnablement conçus ? La documentation qui décrit le code a-t-elle été mise à jour ?

Le mécanisme est presque partout le même. L’auteur envoie un diff et une description, les personnes chargées de la revue commentent des lignes précises, l’auteur répond ou envoie une nouvelle version, et le cycle recommence jusqu’à l’acceptation du changement. Chez Google, cette acceptation s’appelle LGTM, abréviation de *looks good to me*, « cela me paraît bon », et le livre la décrit comme un bit de permission : un seul LGTM est requis en l’absence de règle contraire et, avec l’approbation d’un responsable du code et un contrôle de lisibilité, il permet de créer le commit. GitHub termine chaque revue par l’une de trois décisions. Comment apporte des remarques sans approuver ni demander des changements, Approve indique que le changement peut être fusionné tel

quel, et Request changes désigne les points à traiter avant. Toute personne ayant un accès en lecture au dépôt peut participer aux revues et commenter, et les administrateurs peuvent exiger une approbation avant fusion. Le guide de Google appelle chaque changement une CL, d'où le titre Small CLs de l'une de ses pages.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une revue de code ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Une seule équipe transforme une idée en un produit auquel vos clients peuvent se connecter, sans le faire passer entre trois prestataires.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Application GIS : revenus multipliés par 10 — 10x croissance du chiffre d'affaires portée par le produit cartographique

Application GIS : revenus multipliés par 10 Contactez-nous →

Pourquoi c'est important

Le livre énumère six bénéfices d'une revue bien menée. Elle vérifie que le code est correct, s'assure que les autres ingénieurs comprennent le changement, impose une cohérence dans toute la base de code, favorise la responsabilité collective de l'équipe, diffuse les connaissances et laisse un historique. La correction est le bénéfice évident, et les auteurs citent une étude d'IBM publiée en 1986, selon laquelle les défauts découverts plus tôt demandaient moins de temps à corriger. Le livre ajoute la condition décisive : le temps investi dans la revue ne rapportait que tant que le processus restait léger. L'analyse statique et les tests automatisés assument aujourd'hui une grande partie du contrôle de correction, ce qui, selon le livre, a réduit le besoin de le confier à des personnes.

La compréhension est le bénéfice qu'aucun outil ne remplace. Aussi compétent soit un ingénieur, il ne peut pas lire son propre travail avec le regard d'une personne extérieure, alors que quelqu'un qui le revoit le peut ; la revue est souvent la première occasion où une autre personne que l'auteur vérifie qu'un changement a du sens. C'est important, car le code se lit bien plus souvent qu'il ne s'écrit. Les bénéfices culturels sont plus discrets. La revue rappelle que le code n'appartient pas à son auteur mais à une entreprise commune, permet aux spécialistes de transmettre leur connaissance d'un domaine et, comme chaque changement conserve sa revue, permet ensuite de retrouver l'arrivée d'un pattern et de lire la revue qui l'a introduit.

La revue prend aussi du temps, et une revue lente coûte davantage. Le guide de Google sur la vitesse indique que des revues lentes ralentissent toute l'équipe, suscitent du ressentiment chez les développeurs et nuisent à la santé du code, car elles poussent à accepter des changements de moindre qualité et découragent les nettoyages. Il ajoute que la plupart des plaintes contre une personne trop stricte en revue disparaissent lorsque cette même personne répond simplement plus vite. Son critère d'approbation est pragmatique : dès qu'un changement laisse clairement le code du système en meilleur état qu'avant, il faut plutôt l'approuver, même s'il n'est pas parfait. Le guide en fait son principe prioritaire et en donne la raison en une phrase : « le code "parfait" n'existe pas ; il n'existe que du meilleur code ».

À quoi cela ressemble en pratique

Prenons une boutique dont les prix sont enregistrés en pence entiers. Un auteur ajoute une campagne de printemps et soumet ce changement à la revue :

```
--- a/shop/pricing.py
+++ b/shop/pricing.py
@@ -8,2 +8,4 @@ def order_total(basket):
     subtotal = sum(line.price * line.quantity for line in basket.lines)
+    if basket.voucher == "SPRING":
+        subtotal = subtotal * 0.9
     return subtotal + shipping_for(basket)
```

La personne chargée de la revue laisse quatre commentaires :

```
shop/pricing.py, ligne 9
  Les autres bons de réduction et leurs taux sont dans VOUCHERS, dans
  shop/vouchers.py. SPRING pourrait-il y aller aussi ? La prochaine
  campagne modifierait alors les données, et non cette fonction.

shop/pricing.py, ligne 10
  Partout ailleurs dans ce module, les prix sont en pence entiers.
  Multiplier par 0.9 peut laisser une fraction de penny. Les autres
  remises passent par round_money() dans shop/money.py.

shop/pricing.py, ligne 11
  Nit: les frais de port sont exclus de la remise. Si c'est voulu,
  un bref commentaire le précisant aiderait la personne qui le lira ensuite.

Général
```

```
Aucun test ne porte sur un panier avec un bon de réduction.
Pourriez-vous en ajouter un pour SPRING, et un montrant qu'un code
inconnu ne change rien ?
```

Chaque commentaire porte sur le code plutôt que sur la personne et explique pourquoi. Les deux premiers et le dernier n'ont pas d'étiquette : il faut donc les traiter. Le troisième commence par Nit:, que le guide de Google emploie pour un petit point que l'auteur devrait corriger mais qui change peu de choses. Ses autres étiquettes sont Optional, ou Consider, pour une bonne idée qui n'est pas obligatoire, et FYI pour une réflexion qui n'exige pas d'action dans ce changement. Sans étiquettes, prévient-il, les auteurs risquent de lire chaque commentaire comme une obligation. Sur GitHub, ce tour se termine par Request changes. L'auteur déplace SPRING dans la table, fait passer la remise par la fonction auxiliaire d'arrondi, ajoute le commentaire et les deux tests, puis demande un nouvel examen ; le second tour se termine par Approve, ou LGTM chez Google. Le guide demande de répondre à une demande de revue sous un jour ouvré au maximum, sans interrompre une tâche qui exige de la concentration, mais en attendant une pause naturelle.

Aucun commentaire ne porte sur la mise en forme, l'ordre des imports ou la réussite des tests, car une machine aurait dû le signaler avant l'ouverture du diff. Le livre décrit des contrôles presubmit qui s'exécutent lors du premier envoi d'un changement en revue et peuvent le rejeter avant que quiconque ne le lise ; le noyau demande aux auteurs d'exécuter son contrôleur de style, `scripts/checkpatch.pl` , avant l'envoi. Le moteur de contrôles bloquants construit pour un dépôt, construit dans le cadre d'un poste occupé depuis 2025, pousse cette idée très loin : 268 contrôles de commit enregistrés, 277 règles de lint et 15 règles ESLint sur mesure, plus 146 tests des contrôles eux-mêmes. Ils vérifient notamment qu'une migration peut être annulée et qu'une clé de traduction existe dans les deux locales, et tout commit marqué comme correction doit inclure un test qui échoue sans elle. Les contrôles traduisaient les règles mécaniques en code, ce qui permettait aux personnes chargées de la revue de consacrer plus de temps à la conception. Elle précise aussi le coût : créer un commit est lent et contourner un contrôle mal écrit est exaspérant.

Où les problèmes surviennent

Le changement est trop gros pour être lu. Le guide de Google juge généralement raisonnable un changement de 100 lignes et généralement trop gros un changement de 1 000 lignes, tout en laissant la décision à la personne qui le revoit. Il précise aussi que 200 lignes dans un fichier peuvent convenir, là où les

mêmes 200 lignes réparties sur 50 fichiers ne conviendraient pas. Les petits changements sont revus plus vite et plus soigneusement et risquent moins d'introduire des bugs ; le livre ajoute qu'il est légitime de refuser un changement trop gros pour une seule revue. Divisez le travail : empilez de petits changements, envoyez des groupes de fichiers à des personnes différentes, séparez les refactorisations des nouvelles fonctions et des corrections de bugs, et gardez les tests de chaque changement dans ce même changement.

La revue attend pendant des jours. Chaque journée d'attente retarde le travail qui dépend du changement, et une réponse tous les quelques jours avec des demandes importantes fait paraître le processus plus sévère qu'il ne l'est. Répondez sous un jour ouvré ; le livre attend un retour sous 24 heures ouvrées et, si la revue complète doit attendre, la personne qui la prend en charge doit confirmer avoir vu le changement. Envoyez tous les commentaires ensemble plutôt qu'au compte-gouttes : peu de choses agacent autant un auteur qu'un nouveau tour de remarques portant sur d'autres points après avoir traité le précédent.

Un goût personnel devient une exigence. Une personne qui préfère une autre boucle ou un autre nom et bloque le changement pour cette raison apprend aux auteurs à cesser d'améliorer le code. Le guide de Google fixe l'ordre : faits et données priment sur les opinions ou les goûts, le guide de style a le dernier mot en matière de style et, si plusieurs approches sont aussi valables, l'auteur choisit. Étiquetez une préférence Nit ou Optional pour la distinguer d'une exigence.

Le commentaire vise la personne. Le mauvais exemple du guide demande pourquoi l'auteur a utilisé des threads alors que la concurrence n'apporte manifestement aucun bénéfice ; le bon indique que le modèle de concurrence ajoute de la complexité sans gain de performance visible pour la personne qui fait la revue, et que le code devrait donc être monothread. Commentez le code, jamais la personne qui le développe, et expliquez le raisonnement derrière la demande.

Des personnes vérifient ce qu'une machine pourrait contrôler. Consacrer son attention aux espaces et à l'ordre des imports fait manquer la question de conception que seule une personne peut trancher. Transférez chaque règle que l'on peut formaliser vers un linter, un outil de formatage ou un contrôle presubmit, et réservez le jugement à la personne chargée de la revue.

La réponse reste dans l'outil de revue. Une personne demande à quoi sert un bloc de code, l'auteur l'explique dans le fil, la revue est approuvée et la prochaine personne qui lit le code ne voit jamais l'explication. Le guide de Google indique qu'une telle question devrait généralement conduire à réécrire le code plus clairement ou, au minimum, à y ajouter un commentaire. Le noyau fait la même remarque aux auteurs : quand la question d'une revue ne change aucun code, sa réponse doit en règle générale finir dans un commentaire ou dans le journal des changements.

Pour aller plus loin

- Introduction — la page d'ouverture du guide de revue de Google : points examinés, choix des personnes chargées de la revue et cas où la programmation en binôme en tient lieu.
- Le critère de la revue de code — le principe prioritaire, le mentorat par la revue et la résolution des conflits entre auteur et personne chargée de la revue.
- Petites CL — pourquoi les petits changements sont mieux revus, quelle taille est excessive et quatre façons de diviser un changement.
- Revue de code — le chapitre de Tom Manshreck et Caitlin Sadowski dans *Software Engineering at Google* : bénéfices, LGTM, responsabilité sur le code et bonnes pratiques.
- Revues de pull requests — la référence GitHub pour revoir, approuver, demander des changements et exiger des approbations avant fusion.
- Soumettre des patches : le guide essentiel pour intégrer votre code au noyau — la revue par courriel dans le noyau Linux, ce qu'atteste une étiquette Reviewed-by et comment les auteurs doivent répondre.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'une revue de code ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/code-review/