Engineer

Code review

~~ɓɔ~~ en andens gennemgang af en ændring, før den flettes ind

Hvad det er

Et code review er det trin, hvor en anden end forfatteren læser en ændring, før den bliver en del af kodebasen. Googles guide til virksomhedens udviklingspraksis definerer det som en proces, hvor en anden end kodens forfatter eller forfattere undersøger koden, og kapitlet om emnet i *Software Engineering at Google* tilføjer, at det ofte sker, før koden bliver ført ind i kodebasen. På dansk siger mange også kodegennemgang, men i praksis skifter navnet mest med værktøjet. På GitHub hedder det et pull request-review, hos Google et precommit-review, og i Linux-kernen foregår det over e-mail, hvor en reviewer, der er tilfreds med en patch, tilbyder et Reviewed-by-tag til den.

Googles guide oplister, hvad en reviewer ser efter: design, funktionalitet, kompleksitet, tests, navngivning, kommentarer, stil og dokumentation. Bag hvert ord står et spørgsmål. Opfører koden sig, som forfatteren sandsynligvis havde tænkt sig, og er den opførsel god for dens brugere? Kunne den være enklere? Følger der korrekte og veldesignede automatiske tests med? Har forfatteren opdateret den dokumentation, der beskriver den?

Mekanikken er stort set den samme overalt. Forfatteren sender en diff med en beskrivelse, reviewerne kommenterer enkelte linjer, forfatteren svarer eller uploader en ny version, og runden gentages, indtil en reviewer godkender ændringen. Hos Google hedder godkendelsen LGTM, en forkortelse af *looks good to me*, og bogen beskriver den som en tilladelsesbit: der kræves som standard kun ét LGTM, og sammen med en godkendelse fra en kodeejer og et readability-tjek giver det lov til at committe ændringen. GitHub afslutter hvert review med én af tre beslutninger. Comment giver generel feedback uden at godkende eller bede om ændringer, Approve signalerer, at ændringerne er klar til at blive merget, og Request changes markerer feedback, som forfatteren bør tage hånd om før merge. Alle med læseadgang kan reviewe og kommentere, og

administratorer af et repository kan kræve godkendelser, før en pull request bliver merget. Googles guide kalder hver ændring en CL, og derfor hedder en af dens sider Small CLs.

Brug begrebet i en beslutning

Brug »Hvad er code review?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Ét team, der tager en idé fra skitse til noget, jeres kunder kan logge ind på, i stedet for at kaste den mellem tre leverandører.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har ført en GIS-kortapp til 10x omsætningsvækst — 10x omsætningsvækst, drevet af kortproduktet

Har ført en GIS-kortapp til 10x omsætningsvækst Kontakt os →

Hvorfor det betyder noget

Bogen nævner seks fordele ved et velfungerende review. Det kontrollerer, at koden er korrekt, sikrer, at andre udviklere kan forstå ændringen, holder kodebasen ensartet, fremmer teamets fælles ejerskab, spreder viden og efterlader en historisk optegnelse. Korrekthed er den oplagte, og forfatterne henviser til en undersøgelse hos IBM, udgivet i 1986, som viste, at fejl, der blev fundet tidligere, tog kortere tid at rette. Bogen tilføjer den betingelse, der betyder noget: tiden, der blev brugt på review, betalte sig kun, så længe processen forblev let. Statisk analyse og automatiske tests bærer i dag en stor del af korrekthedsarbejdet, og det har ifølge bogen mindsket behovet for at lade mennesker stå for det.

Forståelse er den fordel, som værktøjer ikke overtager. En reviewer kan gøre det, som selv den dygtigste udvikler ikke kan gøre ved sit eget arbejde, nemlig læse det uden forfatterens perspektiv, og et review er ofte den første prøve på, om en ændring giver mening for andre. Det betyder noget, fordi kode bliver læst mange flere gange, end den bliver skrevet. De kulturelle fordele er mere stille. Review minder udvikleren om, at koden ikke er ens egen, men en del af et fælles foretagende, det lader eksperter give deres viden om et område videre, og fordi hver ændring beholder sit review, kan alle senere finde ud af, hvornår et mønster kom ind, og læse det review, der bragte det med.

Review koster også tid, og langsomt review koster mere. Googles guide om hastighed siger, at når reviews er langsomme, falder hele teamets tempo, udviklerne begynder at protestere mod processen, og kodens sundhed lider, fordi der opstår pres for at lade svagere ændringer slippe igennem, og

oprydninger bliver modarbejdet. De fleste klager over, at en reviewer er for streng, forsvinder ifølge guiden, når den samme reviewer blot svarer hurtigere. Dens standard for godkendelse er pragmatisk: reviewere bør hælde mod at godkende en ændring, når den helt sikkert forbedrer systemets samlede kodesundhed, også selvom den ikke er perfekt. Guiden kalder det det øverste princip blandt sine regler og samler begrundelsen i én linje: »der findes ikke perfekt kode – der findes kun bedre kode«.

Sådan ser det ud i praksis

Forestil dig en webbutik, hvor priserne er gemt som hele øre. En forfatter tilføjer en forårskampagne og sender denne ændring til review:

```
--- 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)
```

Revieweren efterlader fire kommentarer:

```
shop/pricing.py, linje 9
  De andre rabatkoder og deres satser ligger i VOUCHERS i
  shop/vouchers.py. Kan SPRING komme med der? Så bliver næste
  kampagne en ændring af data og ikke af denne funktion.

shop/pricing.py, linje 10
  Priserne er hele øre alle andre steder i modulet, og at gange
  med 0.9 kan efterlade en brøkdel af en øre. De andre rabatter
  går gennem round_money() i shop/money.py.

shop/pricing.py, linje 11
  Nit: fragten er ikke med i rabatten. Hvis det er med vilje,
  ville en kort kommentar om det hjælpe den næste læser.

Generelt
  Der er ingen test med en rabatkode i kurven. Kan du tilføje
  én for SPRING og én, der viser, at en ukendt kode lader
  prisen være?
```

Hver kommentar handler om koden og ikke om personen, og hver af dem siger hvorfor. Den første, den anden og den sidste har ingen mærkat, så de forventes at blive håndteret; den tredje begynder med Nit:, som Googles guide bruger om noget mindre, som forfatteren teknisk set burde gøre, men som ikke vil påvirke tingene ret meget. Guidens andre mærkater er Optional, eller Consider, for en god idé, der ikke er påkrævet, og FYI for en tanke, som forfatteren ikke forventes at handle på i denne ændring. Uden mærkater, advarer den, kan forfattere læse hver kommentar som et krav. På GitHub ender denne runde med Request changes. Forfatteren flytter SPRING ind i tabellen, sender rabatten gennem afrundingsfunktionen, tilføjer kommentaren og begge tests og beder om et nyt kig; anden runde ender med Approve, eller hos Google med LGTM. Guiden beder reviewere om at svare på en anmodning om review inden for højst én arbejdsdag og ikke at afbryde en fokuseret opgave for at gøre det, men at vente på en naturlig pause.

Kommentarerne handler hverken om formatering, importrækkefølge eller om testene består, for det burde en maskine have sagt, før revieweren åbnede diffen. Bogen beskriver presubmits, der kører, når en ændring første gang sendes til review, og som kan afvise den, før nogen læser den; Linux-kernen beder forfattere om at køre dens stiltjek, `scripts/checkpatch.pl`, før de indsender. Den kontrolmotor, der blev bygget til et repository, fra 2025, tager den idé langt: 268 registrerede commit-tjek, 277 lint-regler og 15 egne ESLint-regler plus 146 tests af selve tjekkene. Tjekkene kontrollerer blandt andet, at en migration kan rulles tilbage, og at en oversættelsesnøgle findes i begge locales, og enhver commit med præfikset fix: skal have en test med, der fejler uden den. Tjekkene omsatte mekaniske regler til kode, så reviewerne kunne bruge mere tid på design. Siden nævner også prisen: det er langsomt at committe, og en dårligt skrevet kontrol er virkelig frustrerende at arbejde uden om.

Hvor det går galt

Ændringen er for stor til at læse. Googles guide kalder 100 linjer en normalt rimelig størrelse for en ændring og 1.000 linjer normalt for meget, men overlader vurderingen til revieweren, og den bemærker, at 200 linjer i én fil kan være i orden, når de samme 200 linjer spredt over 50 filer ikke ville være det. Små ændringer bliver reviewet hurtigere og grundigere og indfører sjældnere fejl; bogen tilføjer, at reviewere med rette kan afvise en ændring som for stor til ét review. Del arbejdet op: stabl små ændringer oven på hinanden, send grupper af filer til forskellige reviewere, hold refaktoreringer adskilt fra nye funktioner og fejlrettelser, og lad hver ændrings tests ligge i samme ændring.

Reviewet ligger i dagevis. Hver dag, en ændring venter, forsinker det arbejde, der bygger på den, og en reviewer, der kun svarer med få dages mellemrum og med store ønsker, får processen til at føles hårdere, end den er. Svar inden for én arbejdsdag; bogens forventning er feedback inden for 24 arbejdstimer, og hvis det fulde review må vente, bør revieweren sige, at ændringen er set. Send alle kommentarerne på én gang i stedet for lidt ad gangen, for få ting irriterer en forfatter mere end en ny runde uafhængig feedback, efter at den forrige er taget hånd om.

Smag bliver fremført som et krav. En reviewer, der foretrækker en anden løkke eller et andet navn og blokerer ændringen for det, lærer forfatterne at holde op med at forbedre koden. Googles guide fastlægger rækkefølgen: tekniske fakta og data vejer tungere end meninger og personlige præferencer, stilguiden er den absolutte autoritet i spørgsmål om stil, og hvor flere tilgange er lige gode, vinder forfatterens præference. Mærk en præference som Nit eller Optional, så forfatteren kan skelne den fra et krav.

Kommentaren handler om personen. Guidens eget eksempel på en dårlig kommentar spørger, hvorfor forfatteren brugte tråde, når der åbenlyst ikke er nogen gevinst ved samtidighed; den gode udgave siger, at samtidighedsmodellen tilføjer kompleksitet uden nogen ydelsesgevinst, som revieweren kan se, så koden bør være enkelttrådet. Kommentér koden, aldrig udvikleren, og forklar begrundelsen bag ønsket.

Mennesker tjekker det, en maskine kunne. Reviewere, der bruger deres opmærksomhed på mellemrum og importrækkefølge, overser det designspørgsmål, som kun et menneske kan besvare. Flyt hver regel, der kan skrives ned, ind i en linter, et formateringsværktøj eller et presubmit-tjek, og lad revieweren beholde vurderingerne.

Svaret bliver i review-værktøjet. En reviewer spørger, hvad en kodeblok er til, forfatteren forklarer det i tråden, revieweren godkender, og den næste, der læser koden, ser aldrig forklaringen. Googles guide siger, at et sådant spørgsmål som regel bør føre til, at koden bliver skrevet om, så den er tydeligere, eller i det mindste til en kommentar i koden. Linux-kernen siger det samme til forfatterne: spørgsmål i et review, der ikke fører til en kodeændring, bør næsten helt sikkert give anledning til en kommentar eller en post i changeloggen.

Læs mere

- Introduction — åbningssiden i Googles guide til code review: hvad reviewere ser efter, hvordan man vælger reviewere, og hvornår parprogrammering tæller som review.
- The Standard of Code Review — det øverste princip, mentorskab gennem review, og hvordan konflikter mellem forfatter og reviewer bliver løst.
- Small CLs — hvorfor små ændringer får bedre reviews, hvornår en ændring er for stor, og fire måder at dele den op på.
- Code Review — Tom Manshreck og Caitlin Sadowskis kapitel i *Software Engineering at Google*: fordelene ved review, LGTM og ejerskab samt god praksis.
- Pull request reviews — GitHubs reference om at reviewe, godkende og bede om ændringer og om at kræve godkendelser før en merge.
- Submitting patches: the essential guide to getting your code into the kernel — hvordan Linux-kernen reviewer over e-mail, hvad et Reviewed-by-tag erklærer, og hvordan forfattere bør svare reviewerne.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er code review?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/code-review/