Engineer

Kodegjennomgang

~~ɓə~~ en annens gjennomgang av en endring før den flettes inn

Hva det er

En kodegjennomgang er steget der noen andre enn forfatteren leser en endring før den blir en del av kodebasen. Googles veiledning om selskapets utviklingspraksis definerer det som en prosess der noen andre enn kodens forfatter eller forfattere undersøker koden, og kapitlet om emnet i *Software Engineering at Google* legger til at dette ofte skjer før koden føres inn i kodebasen. Mange norske utviklere bruker også det engelske code review, og ellers avhenger navnet mest av verktøyet. På GitHub er det en pull request-gjennomgang, hos Google en precommit-gjennomgang, og i Linux-kjernen skjer det over e-post, der en gjennomgåer som er fornøyd med en patch, tilbyr en Reviewed-by-tag til den.

Googles veiledning lister opp hva en gjennomgåer ser etter: design, funksjonalitet, kompleksitet, tester, navngiving, kommentarer, stil og dokumentasjon. Bak hvert ord ligger et spørsmål. Oppfører koden seg slik forfatteren sannsynligvis mente, og er den oppførselen god for brukerne? Kunne den vært enklere? Følger det med korrekte og godt utformede automatiske tester? Har forfatteren oppdatert dokumentasjonen som beskriver den?

Mekanikken er stort sett den samme overalt. Forfatteren sender en diff med en beskrivelse, gjennomgåerne kommenterer enkeltlinjer, forfatteren svarer eller laster opp en ny versjon, og runden gjentas til en gjennomgåer godkjenner endringen. Hos Google kalles godkjenningen LGTM, en forkortelse for *looks good to me*, og boken beskriver den som en tillatelsesbit: som standard kreves bare én LGTM, og sammen med godkjenning fra en kodeeier og en readability-kontroll gir den lov til å committe endringen. GitHub avslutter hver gjennomgang med én av tre avgjørelser. Comment gir generell tilbakemelding uten å godkjenne eller be om endringer, Approve signaliserer at endringene er klare til å flettes inn, og Request changes markerer tilbakemeldinger forfatteren bør ta tak i før innflettingen. Alle med lesetilgang kan gjennomgå og kommentere, og

administratorer av et repo kan kreve godkjenninger før en pull request flettes inn. Googles veiledning kaller hver endring en CL, og derfor heter en av sidene Small CLs.

Bruk begrepet i en beslutning

Bruk «Hva er kodegjennomgang?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Ett team som tar en idé fra skisse til noe kundene dine kan logge inn på, i stedet for at den sendes videre mellom tre leverandører.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Ledet en GIS-kartapp til 10 ganger mer omsetning — 10x inntektsvekst, ledet fra kartproduktet

Ledet en GIS-kartapp til 10 ganger mer omsetning Ta kontakt →

Hvorfor det er viktig

Boken nevner seks fordeler ved en velfungerende kodegjennomgang. Den kontrollerer at koden er korrekt, sørger for at andre utviklere forstår endringen, holder kodebasen konsistent, fremmer felles eierskap i teamet, sprer kunnskap og etterlater en historisk oversikt. Korrekthet er den åpenbare, og forfatterne viser til en studie hos IBM, publisert i 1986, som fant at feil som ble oppdaget tidligere, tok kortere tid å rette. Boken legger til betingelsen som teller: tiden som ble brukt på gjennomgang, lønte seg bare så lenge prosessen forble lett. Statisk analyse og automatiske tester bærer i dag mye av korrekthetsarbeidet, og ifølge boken har det redusert behovet for å la mennesker stå for det.

Forståelse er fordelen verktøyene ikke tar over. En gjennomgåer kan gjøre det selv den beste utvikleren ikke kan gjøre med sitt eget arbeid, nemlig lese det uten forfatterens perspektiv, og en gjennomgang er ofte den første prøven på om en endring gir mening for andre. Det betyr noe fordi kode blir lest mange flere ganger enn den blir skrevet. De kulturelle fordelene er mer stillferdige. Gjennomgangen minner utvikleren på at koden ikke er ens egen, men en del av et felles foretak, den lar eksperter gi videre det de kan om et område, og fordi hver endring beholder sin gjennomgang, kan alle senere finne ut når et mønster kom inn, og lese gjennomgangen som brakte det med seg.

Kodegjennomgang koster også tid, og treg gjennomgang koster mer. Googles veiledning om hastighet sier at når gjennomgangene er trege, faller tempoet i hele teamet, utviklerne begynner å protestere mot prosessen, og kodehelsen lider, fordi det oppstår press for å slippe svakere endringer gjennom, og opprydding blir motarbeidet. De fleste klager på at en gjennomgåer er for

streng, forsvinner ifølge veiledningen når den samme gjennomgåeren bare svarer raskere. Standarden for godkjenning er pragmatisk: gjennomgåere bør heller godkjenne en endring når den helt klart forbedrer den samlede kodehelsen i systemet, selv om den ikke er perfekt. Veiledningen kaller dette det overordnede prinsippet blant reglene sine og samler begrunnelsen i én linje: «det finnes ikke perfekt kode – det finnes bare bedre kode».

Slik ser det ut i praksis

Tenk deg en nettbutikk der prisene føres i hele øre. En forfatter legger til en vårkampanje og sender denne endringen til gjennomgang:

```
--- 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)
```

Gjennomgåeren legger igjen fire kommentarer:

```
shop/pricing.py, linje 9
  De andre rabattkodene og satsene deres ligger i VOUCHERS i
  shop/vouchers.py. Kan SPRING flyttes dit også? Da blir neste
  kampanje en endring i data, ikke i denne funksjonen.

shop/pricing.py, linje 10
  Prisene er hele øre overalt ellers i modulen, og å gange
  med 0.9 kan gi en brøkdel av et øre. De andre rabattene
  går gjennom round_money() i shop/money.py.

shop/pricing.py, linje 11
  Nit: frakten er ikke med i rabatten. Hvis det er med vilje,
  vil en kort kommentar om det hjelpe neste leser.

Generelt
  Det mangler en test med rabattkode i handlekurven. Kan du
  legge til én for SPRING og én som viser at en ukjent kode
    lar prisen være uendret?
```

Hver kommentar handler om koden og ikke om personen, og hver av dem sier hvorfor. Den første, den andre og den siste har ingen merkelapp, så de forventes fulgt opp; den tredje begynner med Nit:, som Googles veiledning bruker om noe lite som forfatteren teknisk sett burde gjøre, men som ikke vil påvirke noe særlig. Veiledningens andre merkelapper er Optional, eller Consider, for en god idé som ikke er påkrevd, og FYI for en tanke forfatteren ikke forventes å handle på i denne endringen. Uten merkelapper, advarer den, kan forfattere lese hver kommentar som et krav. På GitHub ender denne runden med Request changes. Forfatteren flytter SPRING inn i tabellen, sender rabatten gjennom avrundingsfunksjonen, legger til kommentaren og begge testene og ber om en ny titt; andre runde ender med Approve, eller hos Google med LGTM. Veiledningen ber gjennomgåere svare på en forespørsel om gjennomgang innen høyst én arbeidsdag, og ikke avbryte en konsentrert oppgave for å gjøre det, men vente på en naturlig pause.

Kommentarene handler verken om formatering, importrekkefølge eller om testene består, for det burde en maskin ha sagt før gjennomgåeren åpnet diffen. Boken beskriver presubmits som kjører når en endring først sendes til gjennomgang, og som kan avvise den før noen leser den; Linux-kjernen ber forfattere kjøre kjernens stilkontroll, `scripts/checkpatch.pl`, før de sender inn. Kontrollmotoren som ble bygd for et repo, fra 2025, tar den ideen langt: 268 registrerte commit-kontroller, 277 lint-regler og 15 egne ESLint-regler, pluss 146 tester av selve kontrollene. Kontrollene sjekker blant annet at en migrering kan rulles tilbake og at en oversettelsesnøkkel finnes i begge språkversjonene, og enhver commit med prefikset fix: må ha med en test som feiler uten den. Kontrollene kodet inn mekaniske regler, slik at gjennomgåerne kunne bruke mer tid på design. Siden sier også hva det koster: det går tregt å committe, og en dårlig skrevet kontroll er virkelig irriterende å jobbe seg rundt.

Hvor det går galt

Endringen er for stor til å lese. Googles veiledning kaller 100 linjer vanligvis en rimelig størrelse for en endring og 1 000 linjer vanligvis for stort, men overlater vurderingen til gjennomgåeren, og den påpeker at 200 linjer i én fil kan være greit, mens de samme 200 linjene spredt over 50 filer ikke ville vært det. Små endringer blir gjennomgått raskere og grundigere og introduserer sjeldnere feil; boken legger til at gjennomgåere med rette kan avvise en endring som for stor for én gjennomgang. Del opp arbeidet: stable små endringer oppå hverandre, send grupper av filer til ulike gjennomgåere, hold refaktoreringer adskilt fra nye funksjoner og feilrettinger, og la hver endrings tester ligge i samme endring.

Gjennomgangen blir liggende i dagevis. Hver dag en endring venter, forsinker arbeidet som bygger på den, og en gjennomgåer som bare svarer med noen dagers mellomrom og med store krav, får prosessen til å føles strengere enn den er. Svar innen én arbeidsdag; bokens forventning er tilbakemelding innen 24 arbeidstimer, og hvis den fulle gjennomgangen må vente, bør gjennomgåeren si at endringen er sett. Send alle kommentarene samlet i stedet for stykkevis, for få ting irriterer en forfatter mer enn en ny runde urelaterte tilbakemeldinger etter at den forrige er tatt hånd om.

Smak blir framstilt som krav. En gjennomgåer som foretrekker en annen løkke eller et annet navn og blokkerer endringen for det, lærer forfatterne å slutte å forbedre koden. Googles veiledning fastsetter rekkefølgen: tekniske fakta og data veier tyngre enn meninger og personlige preferanser, stilguiden er den absolutte autoriteten i spørsmål om stil, og der flere tilnærminger er like gode, vinner forfatterens preferanse. Merk en preferanse som Nit eller Optional, slik at forfatteren kan skille den fra et krav.

Kommentaren handler om personen. Veiledningens eget eksempel på en dårlig kommentar spør hvorfor forfatteren brukte tråder når det åpenbart ikke er noen gevinst ved samtidighet; den gode versjonen sier at samtidighetsmodellen tilfører kompleksitet uten noen ytelsesgevinst gjennomgåeren kan se, så koden bør være enkelttrådet. Kommenter koden, aldri utvikleren, og forklar begrunnelsen bak ønsket.

Mennesker sjekker det en maskin kunne. Gjennomgåere som bruker oppmerksomheten på mellomrom og importrekkefølge, overser designspørsmålet bare et menneske kan svare på. Flytt hver regel som kan skrives ned, inn i en linter, et formateringsverktøy eller en presubmit-kontroll, og la gjennomgåeren beholde skjønnsvurderingene.

Svaret blir værende i gjennomgangsverktøyet. En gjennomgåer spør hva en kodeblokk er til, forfatteren forklarer i tråden, gjennomgåeren godkjenner, og den neste som leser koden, ser aldri forklaringen. Googles veiledning sier at et slikt spørsmål som regel bør føre til at koden skrives om så den blir tydeligere, eller i det minste til en kommentar i koden. Linux-kjernen sier det samme til forfatterne: spørsmål i en gjennomgang som ikke fører til en kodeendring, bør nesten helt sikkert føre til en kommentar eller en oppføring i endringsloggen.

Les mer

- Introduction — åpningssiden i Googles veiledning for kodegjennomgang: hva gjennomgåere ser etter, hvordan man velger dem, og når parprogrammering teller som gjennomgang.
- The Standard of Code Review — det overordnede prinsippet, veiledning gjennom gjennomganger og hvordan konflikter mellom forfatter og gjennomgåer løses.
- Small CLs — hvorfor små endringer får bedre gjennomganger, når en endring er for stor, og fire måter å dele den opp på.
- Code Review — Tom Manshreck og Caitlin Sadowskis kapittel i *Software Engineering at Google*: fordelene ved gjennomgang, LGTM og eierskap, og god praksis.
- Pull request reviews — GitHubs referanse om å gjennomgå, godkjenne og be om endringer og om å kreve godkjenninger før innfletting.
- Submitting patches: the essential guide to getting your code into the kernel — hvordan Linux-kjernen gjennomgår over e-post, hva en Reviewed-by-tag erklærer, og hvordan forfattere bør svare gjennomgåerne.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er kodegjennomgang?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/code-review/