Engineer

God praksis i programvareutvikling

utprøvde arbeidsmåter for bestemte vilkår

Hva det er

God praksis i programvareutvikling er en arbeidsmåte som har vist seg nyttig nok til å bli det vanlige valget under bestemte forhold. Å gjenopprette en sikkerhetskopi i et eget miljø er nyttig når bedriften må vite hvor fort driften kan komme i gang igjen. En fullført kopiering viser at jobben kjørte. Gjenoppretting viser om de lagrede dataene kan brukes, hvor lang tid det tar, og hvilke trinn som fortsatt må improviseres.

Et råd kan komme fra et forskningsmiljø, et prosjekt eller en standardiseringsorganisasjon. Rådet bygger ofte på noen forutsetninger. Teamet har kanskje et eget miljø å sette i produksjon fra eller en person som kan kontrollere endringen. Teamet kan innføre anbefalingen som en regel eller bruke den som veiledning. Det kan også gjelde et krav i en kontrakt. Opphavet til en regel avgjør hvem som kan gjøre unntak og på hvilke vilkår.

NISTs rammeverk for sikker programvareutvikling beskriver i versjon 1.1 praksiser, oppgaver og eksempler på gjennomføring. Rammeverket er laget for å kunne innpasses i ulike utviklingsløp. Det gir et nyttig skille mellom formålet og en konkret løsning. Å kunne dokumentere at noen har gjennomgått en endring, er et resultat. En innstilling i et bestemt verktøy er én måte å oppnå det på. Team kan undersøke om sine egne verktøy gir tilsvarende dokumentasjon og beskytter det som faktisk står på spill.

Uttrykket brukes også formelt. IETF utgir serien Best Current Practice gjennom en prosess som beskrives i RFC 2026. RFC 8174 forklarer når bestemte ord skrevet med store bokstaver har den særskilte normative betydningen kodifisert i konvensjonen. Et team kan ellers bruke «god praksis» i betydningen veletablert råd. Les hvor anbefalingen kommer fra, når den gjelder og hva den ber om.

Bruk begrepet i en beslutning

Bruk «Hva er god praksis i programvareutvikling?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Testet, utgitt og mulig å reversere, slik at endringer når kundene en tirsdag ettermiddag i stedet for sent en fredagskveld.

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. Automatiserte GIS-SaaS-utrulling i GitHub Actions — produksjonssetting, data og rapportering, alt uten manuelle grep

Automatiserte GIS-SaaS-utrulling i GitHub Actions Ta kontakt →

Hvorfor det er viktig

En tydelig arbeidsmåte gjør at teamet slipper å ta samme diskusjon for hver endring. Nye utviklere vet hva som skal følge en forespørsel om gjennomgang. Driften får sammenlignbare opplysninger om produksjonssettinger, og team som deler en tjeneste, vet hva de må varsle hverandre om. «Sjekk koden grundig» etterlater standarden til hver enkelt. En demonstrasjon av at kunden ikke kan bli belastet to ganger for samme bestilling, gir et observerbart resultat.

En kjent praksis gjør også avvikene synlige. Det kan være lurt å sette små programvareendringer i produksjon ofte. En oppdatering av fastvare kan kreve at en tekniker isolerer maskinen, kjører oppdateringen og kontrollerer den på stedet. Små endringer kan fremdeles være nyttige for utformingen, utprøvingen og rekkefølgen. Hyppigheten må likevel ta hensyn til hvor og når en enhet faktisk kan gjenopprettes.

DORA beskriver utvikling på en felles hovedgren som hyppig integrasjon av små endringer. Det er et aktuelt spor for team som bruker mye tid på å slå sammen endringer fra gamle grener. De må fortsatt undersøke egne forhold. Hvor lang er ventetiden før endringer kommer inn? Hva blir avvist? Hvor ofte oppstår det problemer etter integrering eller produksjonssetting? En kortere liste over grener kan bety at det ligger mer arbeid i hver gren.

Slik ser det ut i praksis

Tenk deg et team som vedlikeholder et internt rapportverktøy. En annen person skal gjennomgå hver endring, og den felles testpakken skal bestås. Månedsrapporten har blitt forsinket fordi et filter viser feil rader. Deler av testpakken bruker flere timer på en annen eksport. Filteret kan rettes og den tidligere versjonen hentes fram før noen mottar rapporten.

Teamet finner først ut hvilke krav som faktisk gjelder. Hvis en bestemt godkjenning kreves før utsending, skal den også innhentes når det haster. Deretter avgrenser de risikoen ved endringen. Rapporten skal ta med riktige rader, beholde grupperingen og utelate data fra andre avdelinger. Utvikleren tester med kjente data og sammenligner resultatet med de forventede radene. En kollega vurderer både koden og resultatet. Teamet beholder den gamle rapporten og utpeker en person som kan stoppe utsendingen. Hele testpakken kjøres senere for å gi grunnlag for en varig forbedring.

Notatet om unntaket viser den ordinære regelen, hvorfor saken kvalifiserer, hvilke vern som gjelder, og når unntaket utløper. Skriv hva som ble resultatet, og mål den faktiske gevinsten. Gjentar problemet seg, kan teamet lage en raskere kontroll for akkurat den rapporten. De kan prøve den med vilje mot en feil måned og data fra en annen avdeling. Hvis testen avslører feilene, vet teamet mer enn om det bare hadde målt kjøretiden.

Saken om kontrollen av en commit med 22 linters viser hvordan avtalte kontroller ble håndhevet i et prosjekt. Saken om tilgjengelighetsfeil som automatiske kontroller slapp gjennom viser hvor nyttig det er å undersøke den delen av brukeropplevelsen som en slik kontroll ikke kan bekrefte.

Hvor det går galt

Sjekklisten blir målet. En ferdig utfylt liste kan eksistere samtidig med feilen den skulle forebygge. Velg et tilfelle der praksisen tydelig skal gjøre en forskjell. Åpne et utvalg av de gjenopprettede dataene, eller sjekk hvem som faktisk fikk den nye versjonen.

Verktøyet blir tatt for arbeidsmåten. Et nytt system for gjennomgang gir ikke folk mer tid til å utfordre et valg. Beskriv hvilket resultat verktøyet skal støtte, og undersøk om fagfolk i praksis får opplysningene de trenger tidsnok.

Unntaket blir en vane. En nødløsning som fungerer, blir gjerne brukt igjen. Sett en tidsgrense og pek ut noen som eier risikoen. Et gjentatt unntak gir grunn til å forbedre den ordinære prosessen på en synlig måte.

Kilden kopieres uten vilkårene. Hyppig utrulling på en nettjeneste har andre kostnader enn oppdatering på et anlegg. Sammenlign utfordringene, prøv metoden i liten skala, og bestem på forhånd hva som taler for å fortsette.

Les mer

- NISTs rammeverk for sikker programvareutvikling, versjon 1.1 — arbeidspraksiser, oppgaver og eksempler på gjennomføring.
- RFC 2026, avsnitt 5 — betydningen av Best Current Practice i IETFs standardiseringsarbeid.
- RFC 8174 — når relevante ord med store bokstaver har sin normative betydning.
- DORA om utvikling på hovedgrenen — små endringer, hyppig integrasjon og måter å undersøke virkningen på.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er god praksis i programvareutvikling?» 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/engineering-best-practices/