Engineer

God praxis i mjukvaruutveckling

beprövade arbetssätt för ett visst sammanhang

Vad det är

God praxis i mjukvaruutveckling är ett arbetssätt som har visat sig värdefullt nog för att vara det vanliga valet under vissa förhållanden. Att återställa en säkerhetskopia i en separat miljö är god praxis när man behöver veta hur fort verksamheten kan komma igång igen. Ett avslutat kopieringsjobb visar att kopian togs. En återställning visar om den går att använda, hur lång tid arbetet tar och vilka moment som ännu kräver improvisation.

En rekommendation kan komma från en forskargrupp, ett projekt eller ett standardiseringsorgan. Ofta bygger den på förutsättningar. Teamet kanske har en egen miljö för utrullning eller tillgång till en kollega som kan granska förändringen. Ett råd kan användas som vägledning eller bli en gemensam regel. I ett avtal kan det finnas ytterligare krav. Varifrån regeln kommer påverkar vem som kan medge ett undantag och under vilka villkor.

NIST:s ramverk för säker programvaruutveckling beskriver i version 1.1 arbetssätt, uppgifter och exempel på genomförande. Det går att lägga in arbetssätten i olika utvecklingsprocesser. Upplägget gör det lättare att skilja målet från en viss metod. Att kunna visa att någon har granskat en ändring är ett resultat. En inställning i ett visst verktyg är ett sätt att nå dit. Ett annat team kan kontrollera om dess egen rutin ger motsvarande belägg och uppfyller gällande krav.

Uttrycket kan också användas formellt. IETF:s serie Best Current Practice har ett förfarande som beskrivs i RFC 2026. RFC 8174 förklarar när vissa ord med versaler har den betydelse som anges i konventionen. Ett företag kan i stället mena ett väl beprövat råd när det skriver ”god praxis”. Läs källan, syftet och tillämpningsområdet innan rekommendationen behandlas som ett krav.

Använd begreppet i ett beslut

Använd ”Vad är god praxis i mjukvaruutveckling?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Testade, släppta och möjliga att backa, så att ändringar når kunderna en tisdagseftermiddag i stället för sent en fredagskväll.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Automatiserade GIS-driftsättning i GitHub Actions — driftsättning, data och rapportering, helt utan manuellt arbete

Automatiserade GIS-driftsättning i GitHub Actions Hör av er →

Varför det spelar roll

Ett tydligt arbetssätt besparar teamet samma diskussion vid varje ändring. En ny utvecklare vet vad som ska bifogas när en ändring skickas för granskning. Driften får uppgifter som går att jämföra mellan utrullningar och team som delar en tjänst kan vara överens om vad som kräver ett meddelande. ”Granska noga” lämnar kriterierna till var och en. Att visa att kunden inte kan debiteras två gånger för samma beställning ger granskarna ett beteende att pröva.

En känd metod gör också undantagen synliga. Tät utrullning av små ändringar kan fungera för många tjänster. Vid en uppdatering av fast programvara kan en tekniker behöva isolera en enhet, uppdatera den och prova den på plats. Små ändringar kan fortfarande förbättra konstruktionen, testerna och ordningen för utrullningen. Tidsplanen för den fysiska insatsen behöver ta hänsyn till hur apparaten går att återställa och vem som kan göra det.

DORA beskriver utveckling mot en gemensam huvudgren genom frekvent integrering av små ändringar. Det ger en riktning när ett team lägger mycket tid på sammanslagning av långlivade grenar. Teamet behöver också egna observationer. Hur lång tid tar det att få in en ändring? Vad blir avvisat? Vilka fel syns först efter utrullning? En kortare grenlista hjälper föga om arbetet per gren har ökat.

Så ser det ut i praktiken

Tänk dig ett team som underhåller ett internt rapportverktyg. En andra person ska granska varje ändring och den gemensamma testsviten ska gå igenom. Månadsrapporten är försenad på grund av ett filterfel. Testsviten tar timmar eftersom den också kör omfattande prov för en annan export. Ändringen berör bara filtret och kan dras tillbaka innan rapporten har skickats.

Teamet kontrollerar först gällande krav. Behövs ett godkännande före utskick måste den snabbare vägen ändå ordna det. Därefter avgränsar teamet risken. Rätt poster ska komma med, grupperingen ska finnas kvar och uppgifter från andra avdelningar ska utelämnas. Utvecklaren kör avgränsade tester med kända data och jämför raderna med det förväntade utfallet. En kollega granskar ändringen och jämförelsen. Teamet behåller den föregående rapporten och anger vem som kan stoppa utskicket. Hela testsviten körs senare för att ge underlag till en varaktig förbättring.

Notera den vanliga regeln, skälet till undantaget, vilka skydd som finns kvar och när undantaget löper ut. Mät hur lång tid arbetet tog och kontrollera vad resultatet blev. Om samma problem återkommer kan teamet ta fram en snabb, permanent kontroll för just rapporten. Pröva den mot en avsiktligt felaktig månad och en post från en annan avdelning. Tidtagning ensam visar inte om de felen hade upptäckts.

Fallet om en obligatorisk kontroll vid commit med 22 linters visar hur ett projekt gjorde sina kontroller bindande. Fallet om tillgänglighetsfel som de automatiska testerna missade visar vikten av att undersöka den del av användarens upplevelse som kontrollerna inte fångar.

Var det går fel

Checklistan blir målet. En ifylld lista kan fortfarande följas av det fel man försökte förebygga. Välj ett tillfälle då arbetssättet borde göra skillnad och kontrollera utfallet. Går de återställda data att öppna?

Verktyget får stå för arbetssättet. En plattform för kodgranskning skapar inte tid för att pröva ett beslut. Ange resultatet som plattformen ska underlätta och kontrollera att människor får de uppgifter de behöver i tid.

Undantaget blir en vana. En väg som fungerar i ett nödläge blir lätt återanvänd. Ange vem som ansvarar för den, hur risken begränsas och när den ska tas upp igen. Upprepning är skäl att undersöka och förbättra den ordinarie processen.

Källans villkor följer inte med. Tät utrullning på en nätbaserad tjänst kan kräva små resurser samtidigt som samma upplägg blir dyrt på en fysisk anläggning. Jämför problemen, begränsa försöket och kom överens i förväg om hur ett gott och ett dåligt utfall ser ut.

Läs mer

- NIST:s ramverk för säker programvaruutveckling, version 1.1 — arbetssätt, uppgifter och exempel på genomförande.
- RFC 2026, avsnitt 5 — betydelsen av Best Current Practice i IETF:s process.
- RFC 8174 — när versalerna anger ordens särskilda normativa betydelse.
- DORA om utveckling på huvudgrenen — små ändringar, tät integrering och sätt att följa effekten.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är god praxis i mjukvaruutveckling?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/engineering-best-practices/