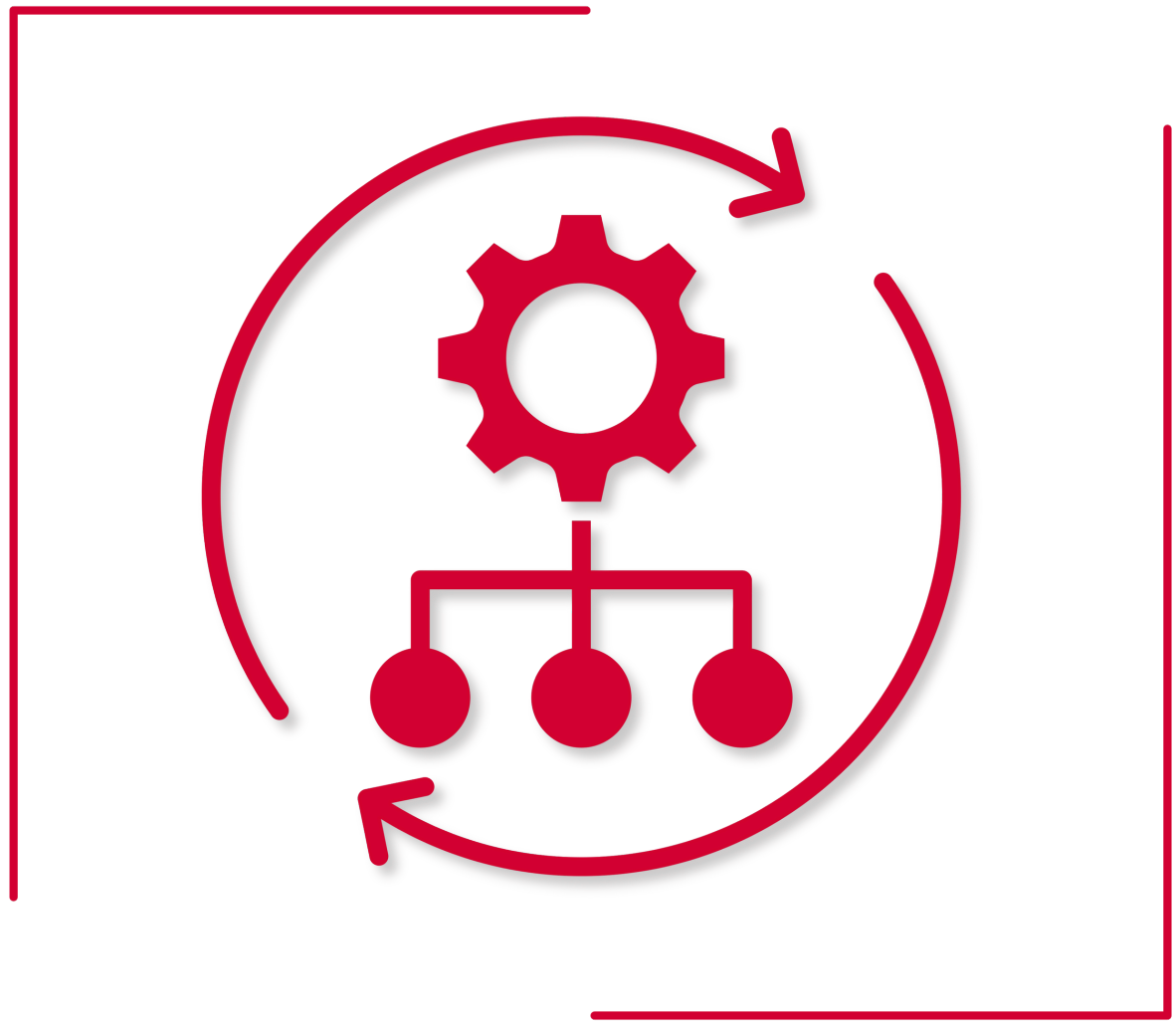
QESTIT



CHECKLISTA

12 PUNKTER FÖR BRA AUTOMATISERADE TESTER

Introduktion

Det finns flera syften eller mål med att skapa automatiserade tester. Att skapa dessa tester är ofta ganska enkelt, men att skapa "bra" automatiserade tester är en betydligt större utmaning som kräver omfattande erfarenhet och medveten övning. Här har vi sammanställt några övergripande mål som behöver uppnås för att de automatiserade testerna ska vara effektiva och av hög kvalitet. Dessa mål har sedan brutits ner i totalt 12 specifika punkter. Ett automatiserat test anses vara "bra" när samtliga 12 punkter är uppfyllda.

MÅL PÅ HÖG NIVÅ

- Tester bör hjälpa oss att förbättra kvaliteten.
- Tester bör hjälpa oss att förstå systemet.
- Tester bör reducera (och inte öka) risk.
- Tester bör vara enkla att köra.
- Tester bör vara enkla att skriva och underhålla.
- Tester bör kräva minimalt underhåll när systemet utvecklas runt dem.

TESTER BÖR HJÄLPA OSS ATT FÖRBÄTTRA KVALITETEN

1 TEST SOM SPECIFIKATION

Om du använder TDD (Testdriven utveckling) eller BDD (Beteendedriven utveckling), där tester skrivs först, ger detta dig ett verktyg för att definiera vad systemet ska göra innan själva utvecklingen påbörjas. Genom att tänka igenom olika scenarier och omvandla dem till tester kan vi identifiera områden där kraven är oklara eller motsägelsefulla. Denna analys bidrar till att förbättra specifikationens kvalitet, vilket i sin tur leder till en bättre design och därmed högre programvarukvalitet.

2 DEFEKTAVSTÖTANDE

Automatiserade tester kan upptäcka buggar, men deras främsta syfte är inte att hitta fel. Istället handlar testautomation om att förhindra att buggar uppstår. Se automatiserade tester som en barriär mot defekter, som hindrar tidigare åtgärdade fel från att återkomma i programvaran. När vi har omfattande och välfungerande regressionstester på plats, minimerar vi risken för buggar eftersom testerna identifierar problem innan koden ens checkas in.

3 UTPEKNING AV DEFEKTER

Om de automatiserade testerna är små och fokuserade, där varje test endast kontrollerar ett enda beteende, kan vi snabbt identifiera var felet ligger baserat på vilket test som misslyckas. För att uppnå detta behöver vi skriva tester som täcker alla möjliga scenarier och varje del av programvaran. Det är också viktigt att testerna själva inte innehåller några fel, vilket gör det avgörande att hålla dem så enkla och lättbegripliga som möjligt. Tester bör därför vara små, ha låg komplexitet, följa ett konsekvent format och endast testa ett beteende i taget.

TESTER BÖR HJÄLPA OSS ATT FÖRSTÅ SYSTEMET

4 TESTER SOM DOKUMENTATION

Automatiserade tester kan tydligt demonstrera hur koden är tänkt att fungera genom att ange det förväntade resultatet av specifika scenarier. Om vi vill förstå hur systemet utför en viss uppgift, kan vi köra testet med felsökaren och steg för steg följa koden för att se hur den arbetar. Enhetstesterna fungerar därmed som en form av dokumentation för systemet, som både beskriver och bekräftar dess beteende.

TESTER BÖR REDUCERA (INTE ÖKA) RISK

5 TESTER SOM SÄKERHETSNÄT

Att ändra äldre kod kan vara riskfyllt eftersom vi ofta inte vet vad som kan gå fel eller om vi råkar förstöra något. Detta kräver att vi arbetar långsamt och noggrant, med mycket manuell analys innan vi gör några förändringar. Däremot, när vi arbetar med kod som har en automatiserad testsvit, kan vi arbeta betydligt snabbare. Testerna fångar upp oväntade sidoeffekter av förändringar och informerar oss om något har gått sönder. På så sätt fungerar de automatiserade testerna som ett säkerhetsnät, vilket ger oss tryggheten att våga göra förändringar.

6 INGEN TESTRISK

Det är viktigt att vi inte introducerar nya problem i systemet som en följd av våra automatiserade tester. För att undvika detta bör testkoden hållas strikt åtskild från produktionskoden, vilket hjälper till att undvika testspecifika beroenden i systemet, särskilt när det gäller enhetstester. All testspecifik kod och de bibliotek som används i testerna bör endast inkluderas under testbyggen och i testmiljöer. Testberoenden och testkod ska aldrig finnas med i den färdiga produktionskoden.

TESTER BÖR VARA ENKLA ATT KÖRA

Det finns fyra specifika punkter som gör automatiska tester enkla att köra. Med dessa fyra punkter uppfyllda är det bara att klicka på en knapp (eller ännu hellre trigga igång automatiskt) för att få den värdefulla feedback som testerna ger:

- Testerna måste vara helt automatiserade så att de kan köras utan ansträngning.
- Testerna måste vara repeterbara så att de kan köras flera gånger med samma resultat.
- Testerna måste vara självutvärderande så att de kan upptäcka och rapportera fel utan manuell inspektion.
- Varje test ska vara oberoende så att det kan köras för sig själv.

7 FULLT AUTOMATISERADE TESTER

Ett test som kan köras utan någon form av manuellt ingripande är ett helt automatiserat test. Att uppfylla den här punkten är en förutsättning för att kunna uppfylla de andra.

8 SJÄLVUTVÄRDERANDE

Vi måste vara noga med att inte introducera nya problem i systemet på grund av automatiserade tester. För att undvika att skapa testspecifika beroenden i systemet, särskilt när det gäller enhetstestkoden, är det viktigt att hålla testkoden helt separat från produktionskoden. All testspecifik kod och de bibliotek som används i testerna ska endast inkluderas under testbyggen och i testmiljöer. Testberoenden och testkod får aldrig finnas med i den slutliga produktionskoden.

9 REPETERBARA TESTER

Ett repeterbart test kan köras många gånger i rad och ger exakt samma resultat utan någon mänsklig inblandning/analys mellan körningarna.

TESTER BÖR VARA ENKLA ATT SKRIVA OCH UNDERHÅLLA

När vi ändrar beteendet i en del av systemet bör vi förvänta oss att endast ett fåtal tester påverkas av dessa modifieringar. En av fördelarna med testautomation är att förenkla förändringar, och därför bör vi alltid arbeta för att säkerställa att våra tester inte motverkar detta syfte genom att göra förändringar mer komplicerade. Tester bör kräva minimalt underhåll när systemet utvecklas och förändras runt dem.

10 ENKLA TESTER

Vi bör prioritera testernas fokus snarare än själva kodningen av dem. Det innebär att testerna ska vara så enkla som möjligt – lätta att läsa, skriva och underhålla. Målet är att verifiera ett enda tillstånd per test, genom att skapa en separat testmetod för varje unik kombination av förutsättningar. Varje testmetod bör dessutom utvärdera systemet genom en enda specifik väg i systemet.

11 UTTRYCKSFULLA TESTER

Ett bibliotek med hjälpmetoder som bygger upp ett domänspecifikt testspråk möjliggör för personen som skriver testkoden att uttrycka de koncept som den vill testa, utan att behöva översätta sina tankar till mycket mer detaljerad kod.

12 DELA UPP PROBLEMEN

Håll testkoden separat från produktionskoden (behåll strukturen och logiken från produktionskoden men i en parallell struktur). Varje test bör fokusera på ett enda problem för att undvika komplicerade och otydliga tester.

Vi hjälper företag att prioritera, planera och designa sitt kvalitetssäkringsarbete med expertkompetens inom test, prestanda, automatisering, säkerhet, ledning, kravhantering och UX-design.

info@qestit.se

08-501 108 90

Wallingatan 2 - 111 60 Stockholm

QESTIT

qestit.com