Engineer

Defekt

🐞 en brist som hindrar en produkt från att uppfylla kraven

Vad det är

En defekt är något som är fel i ett arbetsresultat och som får resultatet att bete sig annorlunda än det var tänkt, eller att utelämna en del av det som det skulle göra. De flesta programmerare säger bugg, och testare och standarder skriver på engelska också fault eller flaw.

Det viktiga ordet är arbetsresultat. En defekt kan sitta i källkod, och den kan lika gärna sitta i ett krav som motsäger ett annat krav, i en design som inte bär den last den ritades för, i en konfiguration, i ett test som förväntar sig fel värde eller i en manual som beskriver en knapp som produkten tog bort för två versioner sedan. Var den än sitter hör defekten till själva saken, och den finns där oavsett om någon har märkt den.

Testarnas ordförråd håller isär tre närliggande ord, och skillnaden gör nytta. En människa begår ett misstag, på engelska error: läser en specifikation fel, utgår från något som inte stämmer eller slinter på tangentbordet. Misstaget lämnar en defekt efter sig i det som personen tog fram. När den defekta delen körs under förhållanden som blottar den beter sig produkten fel, och det synliga felbeteendet heter på engelska failure. Kedjan går åt ett håll och kan stanna vid vilken länk som helst. En defekt i en gren som aldrig körs, eller en som kräver indata som ingen skickar, kan ligga i en kodbas i åratal utan att orsaka ett enda felbeteende.

Vad som räknas som fel beror på vad som förväntades, och förväntningarna sträcker sig längre än den skrivna specifikationen. Kvalitetsmodellen för produkter i ISO/IEC 25010 beskriver funktionell lämplighet som i vilken grad en produkt erbjuder funktioner som uppfyller uttalade och underförstådda behov när den används under angivna förhållanden. Modellen delar upp det i fullständighet (täcker funktionerna de angivna uppgifterna och de tänkta

användarnas mål), korrekthet (är resultaten exakta) och ändamålsenlighet (hjälper funktionerna till att få uppgiften gjord). En felaktig summa är ett problem med korrektheten; en rapport som ingen kan exportera är ett problem med fullständigheten. Samma modell behandlar tillförlitlighet som en egen egenskap, med bl.a. felfrihet, tillgänglighet, feltolerans och återställningsförmåga bland delarna. En produkt som räknar fram rätt svar och sedan kraschar två gånger om dagen har alltså fortfarande defekter.

En regression är en defekt med en historia: något som fungerade och som en senare ändring förstörde. Den har fått ett eget namn eftersom den bär på en ledtråd som en ursprunglig defekt saknar, nämligen en senaste version som veterligen var bra.

Använd begreppet i ett beslut

Använd ”Vad är en defekt?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Ett team som tar en idé från en skiss till något dina kunder kan logga in i, i stället för att lämna den vidare mellan tre leverantörer.

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. Ledde GIS-kartappen till 10 gånger högre intäkter — 10x intäktstillväxt, med kartprodukten i spetsen

Ledde GIS-kartappen till 10 gånger högre intäkter Hör av er →

Varför det spelar roll

Vad en defekt kostar beror mest på var den hittas. Den som skriver koden och själv ser felet betalar en minut, och inga spår blir kvar. Hittas den i en granskning kostar den ett samtal. Hittas den av en kund kostar den själva felbeteendet, supportärendet, utredningen, rättningen och en del förtroende, och om den gjorde verklig skada har den blivit en del av en incident. Typsystem, lintverktyg, tester och kodgranskning arbetar alla längs samma linje och försöker flytta ögonblicket då defekten upptäcks närmare ögonblicket då den skrevs.

Defekter kommer också i familjer. Misstaget bakom en defekt har oftast släktingar: samma missförstådda gränssnitt som anropas på samma sätt på sex ställen, samma förväxling av enheter som kopieras in i varje ny modul, samma okontrollerade antagande i varje hanterare av anrop. Den fruktbara frågan efter en defekt handlar mindre om var den trasiga raden finns än om vad som gjorde raden lätt att skriva, och var samma sak har hänt någon annanstans.

Och defekter kommer tillbaka. Koden ändras hela tiden, och en rättad defekt kan återvända genom en omstrukturering, en sammanslagning eller en uppgradering av ett beroende. Ett test som skrivs för att fallera på defekten och gå igenom med rättningen gör rättningen till en bestående egenskap hos bygget. Dyker en regression upp ändå blir versionshistoriken ett sökområde: `git bisect` gör en binärsökning över de commits som ligger mellan en version som veterligen är bra och en som veterligen är dålig, så att ett spann på tusen commits krymper till den enda som förde in buggen på ungefär tio testkörningar.

Vissa defekter är dessutom säkerhetsproblem. Common Weakness Enumeration från MITRE katalogiserar svagheter, som den beskriver som tillstånd i en komponent i mjukvara, firmware, hårdvara eller en tjänst som under vissa omständigheter kan bidra till att sårbarheter uppstår. Många poster i den listan är vanliga defekter som en angripare har användning för: en gränskontroll som saknas, användarindata som skickas in i en databasfråga utan att escapas, ett steg i behörighetskontrollen som hoppas över på en enda route.

Så ser det ut i praktiken

De defekter som kostade mest på den här webbplatsen var de som varje automatisk signal kallade felfria. En granskning hittade elva tillgänglighetsbrister som de automatiska reglerna hade godkänt, eftersom var och en var en egenskap hos hur en sida renderades snarare än hos vad uppmärkningen sade. Två av dem satt i själva kontrollerna. Varje fönster som layoutkontrollen öppnade var 900 pixlar högt, så en telefon i liggande läge på 852 × 393, med sju pixlar mellan två element, hade aldrig blivit granskad; och regelmotorn hade kört 70 av sina 105 regler. Varje defekt fördes in i en förteckning tillsammans med den regel den gav upphov till, så att arbetet stänger en hel klass av misstag och inte bara ett enskilt fall. När layoutkontrollen väl var lagad fallerade den 128 gånger på tre språk.

En annan defekt gav inga symtom alls. Webbplatsen har ett mörkt tema som standard och ett ljust tema för läsare vars system ber om det, och dess helsidesomslag består av tre lager i det mörka temat. En liten strukturell kontroll som räknar de lager som varje tema ritar ut fann att det ljusa temat bara ritade två redan vid sin första körning. Överlägget hade saknats sedan det ljusa temat skrevs. Sidorna renderades utan ett enda felmeddelande, och en bakgrund med ett lager för lite ser ut som ett medvetet val, så i månader fanns det ingen anledning att titta. Rättningen var en enda regel; kontrollen är det som håller den rättad.

Att hålla rättningar på plats är jobbet för repots vaktmotor, där en regel knyter varje buggrättning till ett bevis: en commit som är märkt som rättning måste ha med sig ett test som fallerar utan den. Vakterna har också 146 egna tester, eftersom en felaktig kontroll kan släppa igenom ändringarna den skulle avvisa och verka fungera.

Var det går fel

Det enskilda fallet rättas och klassen står öppen. Den rapporterade raden rättas, ärendet stängs, och tre syskon till samma misstag ligger och väntar på andra ställen i koden. Efter varje rättning, sök efter det mönster som gav upphov till den, och där en maskin kan känna igen mönstret, gör sökningen till en kontroll.

En rättad bugg kommer tillbaka. En omstrukturering ett halvår senare återställer det gamla beteendet, eftersom ingenting dokumenterade vad rättningen skyddade. Skriv testet före rättningen, se det fallera på den gamla koden och behåll det i testsviten för gott.

Kontrollen bär själv på defekten. En regeluppsättning som i tysthet läser in en bråkdel av sina regler, en lista med testfönster som alla har samma höjd, ett assert som prövar vad koden gör i stället för vad den borde göra: alla rapporterar framgång medan defekten ligger fullt synlig. Ge kontrollerna egna tester, och läs i verktygets egen utdata vilka regler som faktiskt kördes.

Täckning får stå för korrekthet. En testsvit kan köra varje rad och ändå godkänna ett felaktigt resultat när inget assert tittar på utfallet, och kod som uppfyller sin skrivna specifikation kan ändå missa ett underförstått behov. Använd kodtäckning för att hitta kod som inget test når, och bedöm ett test efter de defekter det skulle fånga.

”Fungerar som avsett” stänger en verklig rapport. Beteendet stämmer med specifikationen, rapporten avvisas, och det defekta arbetsresultatet var specifikationen hela tiden. När problemet sitter i kravet eller i designen, registrera defekten mot dem, så att rättningen sker vid källan.

Läs mer

- Defect — definitionen från ISTQB som används inom mjukvarutestning, med synonymer och de närliggande termerna error och failure.
- ISO/IEC 25010 — en översikt över kvalitetsmodellen för produkter och delegenskaperna under varje egenskap, bl.a. funktionell lämplighet och tillförlitlighet.
- git-bisect — manualen till Git om binärsökningen som hittar den commit som förde in en bugg, också med automatiska körningar via ett testskript.
- About CWE — vad Common Weakness Enumeration är, hur listan över svagheter i mjukvara och hårdvara är ordnad och vem som driver den.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en defekt?” 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/defect/