Engineer

Дефект

вада, через яку продукт не відповідає вимогам

Що це таке

Дефект — це щось неправильне всередині результату роботи, через що той поводиться не так, як задумано, або не робить частини того, що мав робити. Програмісти здебільшого кажуть «баг», у розмові можна почути «вада» чи просто «помилка», а в стандартах і підручниках із тестування трапляються ще англійські fault і flaw.

Головне слово тут — *результат роботи*. Дефект може сидіти у вихідному коді, але так само у вимозі, що суперечить іншій вимозі, в архітектурі, яка не витримує навантаження, під яке її проєктували, у файлі конфігурації, у тесті, що очікує хибне значення, чи в інструкції, яка описує кнопку, прибрану з продукту два релізи тому. Хоч би де він був, дефект належить самій речі й існує незалежно від того, помітив його хтось чи ні.

Тестування розрізняє три речі, які в побутовій мові злипаються в одну «помилку». Людина помиляється (error): хибно прочитала специфікацію, зробила неправильне припущення, натиснула не ту клавішу. Ця помилка залишає дефект у тому, що людина створила. Коли дефектна частина виконується за умов, які його оголюють, продукт поводиться неправильно, і саме цю видиму неправильну поведінку називають відмовою (failure). Ланцюжок рухається тільки в один бік і може обірватися на будь-якій ланці. Дефект у гілці коду, до якої виконання ніколи не доходить, або такий, що потребує вхідних даних, яких ніхто не надсилає, може роками лежати в кодовій базі, так і не спричинивши жодної відмови.

Що вважати неправильним, залежить від того, чого чекали, а очікування сягають далі за написану специфікацію. Модель якості продукту в ISO/IEC 25010 описує функціональну придатність (functional suitability) як ступінь, до якого продукт надає функції, що задовольняють заявлені й неявні потреби

під час використання за визначених умов. Цю характеристику поділено на повноту (чи охоплюють функції визначені завдання й цілі користувачів), коректність (чи точні результати) і доцільність (чи допомагають функції виконати завдання). Хибна сума — проблема коректності; звіт, який неможливо експортувати, — проблема повноти. Надійність у моделі стоїть окремою характеристикою, і серед її складників є безвідмовність, доступність (availability), стійкість до відмов і відновлюваність, тож продукт, який рахує правильно, а потім двічі на день падає, однаково має дефекти.

Регресія — це дефект з історією: щось працювало, а пізніша зміна його зламала. Окрему назву вона отримала тому, що несе підказку, якої бракує первинному дефекту, а саме останню версію, про яку відомо, що вона була справною.

Застосуйте поняття до рішення

Застосуйте «Що таке дефект?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Одна команда веде ідею від ескізу до продукту, у який ваші клієнти вже можуть увійти, а не перекидає роботу між трьома підрядниками.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Збільшила дохід GIS-застосунку в 10 разів — 10x зростання виторгу завдяки картографічному продукту

Збільшила дохід GIS-застосунку в 10 разів Зв'язатися з нами →

Чому це важливо

Ціна дефекту найбільше залежить від того, де його знайшли. Якщо його впіймав сам автор коду, це коштує хвилину й не лишає сліду. На рев'ю він коштує одну розмову. Якщо ж його знайшов клієнт, рахунок охоплює саму відмову, звернення до підтримки, розслідування, виправлення й частину довіри, а коли дефект завдав реальної шкоди, він уже став частиною інциденту. Системи типів, лінтери, тести й рев'ю коду працюють на одному відтинку: вони намагаються наблизити момент виявлення до моменту, коли дефект з'явився в коді.

Дефекти ще й ходять родинами. У помилки, яка породила один, зазвичай є родичі: той самий неправильно зрозумілий інтерфейс, однаково викликаний у шести місцях, та сама плутанина з одиницями виміру, скопійована в кожен новий модуль, те саме неперевірене припущення в кожному обробнику запитів. Після дефекту корисніше питати не стільки, де стоїть хибний рядок, скільки що зробило цей рядок легким для написання і де ще сталося те саме.

І дефекти повертаються. Код змінюється постійно, і виправлений дефект може прийти знову через рефакторинг, злиття гілок чи оновлення залежності. Тест, написаний так, щоб падати на дефекті й проходити на виправленні, перетворює виправлення на тривку властивість збірки. Якщо регресія все-таки з'являється, історія версій стає простором пошуку: `git bisect` виконує двійковий пошук серед комітів між відомою справною і відомою зламаною версією, тож діапазон із 1 000 комітів звужується до того, що вніс баг, приблизно за десять запусків тестів.

Деякі дефекти водночас є проблемами безпеки. Common Weakness Enumeration, перелік, який веде MITRE, каталогізує слабкі місця (weaknesses), а проєкт описує їх як стани компонента програмного забезпечення, мікропрограми, апаратури чи сервісу, що за певних обставин можуть сприяти появі вразливостей. Чимало пунктів цього переліку — звичайні дефекти, для яких зловмисник знайшов застосування: пропущена перевірка меж, введення користувача, передане в запит без екранування, крок авторизації, пропущений на одному маршруті.

Як це виглядає на практиці

Найдорожчими дефектами на цьому сайті виявилися ті, які кожен автоматичний сигнал вважав нормальними. Під час аудиту знайшлося одинадцять дефектів доступності, які автоматичні правила пропустили, бо кожен проявлявся у відображеній сторінці, а в HTML-розмітці його не було видно. Два з них сиділи в самих перевірках. Кожне вікно, яке відкривала перевірка макета, мало висоту 900 пікселів, тому телефон, повернутий горизонтально, 852 × 393, із сімома пікселями між двома елементами, жодного разу не потрапив у поле зору; а рушій правил виконував 70 зі своїх 105 правил. Кожен дефект записано в реєстр поруч із правилом, яке він породив, тож робота усуває цілу категорію помилок, а не один її прояв. Щойно саму перевірку макета полагодили, вона провалилася 128 разів на сторінках трьома мовами.

Другий дефект не мав жодного симптому. За замовчуванням сайт показує темну тему, а світлу — читачам, чия система про неї просить, і повносторінкове тло в темній темі складається з трьох шарів. Невелика структурна перевірка, що рахує шари, які малює кожна тема, виявила, що світла тема малює лише два, і зробила це вже під час першого запуску. Накладки бракувало відтоді, як написали світлу тему. Ніщо не кидало помилки, кожна сторінка відображалася, а тло з одним шаром менше легко

прийняти за свідомий вибір, тож місяцями ніхто не мав причини придивитися. На виправлення пішло одне правило, а виправленим його тримає саме перевірка.

Утримувати виправлення на місці — завдання системи запобіжників репозиторію, де одне правило пов'язує кожне виправлення бага з доказом: коміт-виправлення має приносити тест, що провалюється без цього виправлення. Самі запобіжники мають ще 146 власних тестів, бо несправний запобіжник може пропустити зміни, які мав би відхилити, і здаватися справним.

Де виникають проблеми

Випадок виправлено, а клас лишився відкритим. Рядок зі звіту виправляють, заявку закривають, а три родичі тієї самої помилки чекають в інших місцях коду. Після кожного виправлення шукайте шаблон, що його породив, а там, де машина здатна розпізнати шаблон, перетворіть пошук на перевірку.

Виправлений баг повертається. Через пів року рефакторинг відновлює стару поведінку, бо ніде не записано, від чого захищало виправлення. Пишіть тест до виправлення, переконайтеся, що на старому коді він падає, і залиште його в наборі тестів назавжди.

Дефект сидить у самій перевірці. Набір правил, що мовчки завантажує лише частину своїх правил, список тестових вікон однакової висоти, твердження (assertion), яке перевіряє, що код робить, а не що мав би робити: кожне з них звітує про успіх, поки дефект лежить на видноті. Дайте перевіркам власні тести й з'ясовуйте з виводу самого інструмента, які правила справді виконалися.

Покриття підміняє коректність. Набір тестів може пройти кожен рядок і все одно прийняти хибний результат, якщо жодне твердження не дивиться на підсумок, а код, що виконує свою письмову специфікацію, може однаково схибити щодо неявної потреби. Використовуйте покриття, щоб знаходити код, який ніщо не тестує, а цінність тесту оцінюйте за дефектами, які він здатен упіймати.

«Працює як задумано» закриває справжній звіт. Поведінка збігається зі специфікацією, звіт відхиляють, а дефектним результатом роботи весь цей час була сама специфікація. Коли проблема живе у вимозі чи в архітектурі, реєструйте дефект саме там, щоб виправлення відбулося біля джерела.

Що почитати далі

- Defect — визначення ISTQB, прийняте в тестуванні програмного забезпечення, із синонімами та пов’язаними термінами error і failure.
- ISO/IEC 25010 — огляд моделі якості продукту й підхарактеристик кожної характеристики, серед них функціональна придатність і надійність.
- git-bisect — посібник Git із двійкового пошуку коміту, що вніс баг, зокрема з автоматичними запусками тестового скрипту.
- About CWE — що таке Common Weakness Enumeration, як упорядковано її перелік слабких місць у програмах і апаратурі та хто його веде.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке дефект?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/defect/