| Must have рівень:   1. Зроби порівняння статичних та динамічних технік тестування. Наведи переваги та можливі обмеження при використанні кожної з них.  |  | Статистична техніка тестування | Динамічна техніка тестування | | --- | --- | --- | | Основна інформація | Перевірка артефактів (документації, коду, вимог) без виконання програми. Використовуються рецензії, аналіз, інспекції, статичні аналізатори коду. | Перевірка ПЗ під час виконання. Тестувальники запускають систему, виконують тест-кейси, перевіряють фактичну поведінку. | | Перевага №1 | Дефекти виявляються на ранніх етапах (ще до запуску коду). | Дозволяє перевірити фактичну поведінку системи та функціонал. | | Перевага №2 | Допомагає знайти дефекти, які важко виявити під час виконання (наприклад, у вимогах або дизайні). | Виявляє дефекти, що проявляються тільки під час роботи (помилки продуктивності, інтеграційні проблеми). | | Перевага №3 | Знижує витрати і час на виправлення, бо баги виявляються раніше. | Дозволяє перевірити відповідність системи бізнес-вимогам у реальних сценаріях. | | Перевага №4 | Покращує якість документації, дизайну і підтримку коду. | Дозволяє знайти баги, які залежать від середовища (наприклад, браузера, пристрою, ОС). | | Перевага №5 | Покращує комунікацію в команді (обговорення вимог, коду, дизайну). | Підтверджує, що система працює так, як очікується кінцевими користувачами. | | Обмеження №1 | Не можна перевірити фактичну роботу системи (наприклад, як завантажується сторінка). | Вимагає наявності працюючої системи або її компонентів. | | Обмеження №2 | Не виявляє помилки продуктивності, інтеграції або роботи в реальному середовищі. | Зазвичай дорожче і довше (потрібне середовище, дані, сценарії, підтримка). | | Обмеження №3 | Залежить від кваліфікації рецензентів/аналітиків. | Дефекти можуть бути знайдені пізніше, що підвищує вартість виправлення. | | Обмеження №4 | Може пропустити дефекти, які видно лише при запуску програми. | Може не виявити проблем у вимогах, архітектурі або документації. | | Висновок | **Статичне тестування** ефективне для раннього виявлення дефектів, пов’язаних із вимогами, дизайном і кодом. Воно дешевше на старті та попереджає появу багів у майбутньому. | **Динамічне тестування** є критично важливим для перевірки реальної роботи продукту, його функціональності, продуктивності та відповідності бізнес-вимогам. |   Обидві техніки доповнюють одна одну: статичне тестування зменшує кількість помилок до запуску, а динамічне - гарантує, що продукт працює як очікується у реальному середовищі. |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Середній рівень:   1. Виконай завдання попереднього рівня. 2. Наступне твердження стосується покриття рішень:   *Коли код має одну ‘IF” умову, не має циклів (LOOP) або перемикачів (CASE), будь-який тест, який ми виконаємо, дасть результат 50% покриття рішень (decision coverage).*  Яке твердження є коректним?   * 1. **Коректно. Будь-який тест кейс надає 100% покриття тверджень, таким чином покриває 50% рішень.**   2. Коректно. Результат будь-якого тесту умови IF буде або правдими, або ні.   3. Некоректно. Один тест може гарантувати 25% перевірки рішень в цьому випадку.   4. Некоректно, бо занадто загальне твердження. Ми не можемо знати, чи є воно коректним, бо це залежить від тестованого ПЗ.   Я обрала відповідь “а”, бо при одній умові IF існує два можливих шляхи виконання (True та False). Один тест покриває лише один із них, тобто 50% рішень. При цьому всі виконані інструкції в рамках цього шляху будуть покриті і це дає 100% покриття тверджень. Спробувала проілюструвати, для себе я зрозуміла, що згідно з умовою завдання, якщо умова істинна (True) - ніяких дій немає і ми одразу йдемо до кінця. Якщо умова хибна (False), то виконується action, а потім програма також завершується.     1. Є псевдокод: Switch PC on -> Start MS Word -> IF MS Word starts THEN -> Write a poem -> Close MS Word.   Скільки тест кейсів знадобиться, щоб перевірити його функціонал?   * 1. **1 – для покриття операторів, 2 – для покриття рішень**   2. 1 – для покриття операторів, 1 – для покриття рішень   3. 2 – для покриття операторів, 2 – для покриття рішень   4. 2 – для покриття операторів, 1 – для покриття рішень   Потрібно пройти всі кроки хоча б один раз.  Якщо MS Word запускається успішно - ми виконаємо всі дії: ввімкнення ПК, запуск, написання вірша, закриття Word.  Отже, 1 тест кейса достатньо, щоб покрити всіх операторів.  Рішення (decision coverage):  Є одна умова: “Чи стартував MS Word?”  Для покриття умов потрібно:  Випадок True ( я позначила як yes) - Word запустився Випадок False (позначила для себе як no) - Word не запустився.  Тобто потрібно 2 тест кейси.     1. Скільки потрібно тестів для перевірки тверджень коду:      * 1. **2**   2. 1   3. 3   4. 4   Тут є 2 умовні оператори (IF):   1. IF P+Q > 100 2. IF P > 50   Для покриття всіх операторів достатньо 2 тестів: один, де умови істинні - виконаються всі Print і один, де умови хибні - не виконається жоден Print.  **Я вже побачила, що намалювала діаграму так, ніби друге порівняння P > 50 виконується лише якщо перша умова (P+Q > 100) = False.**  **Але це два незалежні IF, тобто друге рішення (P > 50) треба перевіряти завжди, незалежно від першого. Спробувала перемалювати.** |

