| 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 | Статичне тестування є методикою тестування програмного забезпечення, при якій ПЗ тестується без запуску коду. Воно являє собою процес або техніку, які виконуються для пошуку потенційних дефектів в програмному забезпеченні. Це також процес виявлення і усунення помилок і дефектів в різних супровідних документах, таких як специфікації вимог до програмного забезпечення і т. ін. | Метод динамічного тестування – тип тестування, який перевіряє функціональність програми, коли код виконується. Простіше кажучи, динамічне тестування виконується шляхом фактичного використання програми і визначення того, чи працює функціональність так, як очікується. | | Перевага №1 | Знижує вартість виправлення дефектів, виявляючи їх на ранніх етапах. | Забезпечує високу якість ПЗ. | | Перевага №2 | Покращує обізнаність про проблеми якості ПЗ. | Тестує ПЗ з точки зору користувача. | | Перевага №3 | Покращує процес розробки завдяки зворотньому зв'язку. | Виявляє складні помилки. | | Перевага №4 | Зменшує зусилля на виправлення помилок(якщо помилки виявлені до того, як код буде інтегровано та протестовано в реальному середовищі, процес виправлення стає менш трудомістким, оскільки не потрібно змінювати великі обсяги коду або переписувати значні частини програми). | Може бути автоматизовані. | | Перевага №5 | Полегшує обмін критичною інформацією серед команди. | - | | Обмеження №1 | Займає багато часу. | Може займати багато часу. | | Обмеження №2 | Не може виявити помилки, що виникають під час фактичного виконання програми. | Високі витрати. | | Обмеження №3 | Обмежені можливості автоматизації. | Виконується після завершення кодування, і знайдені баги виявляються пізніше в життєвому циклі розробки. | | Висновок | Статичні методики ефективні для раннього виявлення дефектів, що допомагає знижувати витрати на їх виправлення та покращує якість програмного забезпечення на початкових етапах розробки. Однак, вони не можуть виявити проблеми, що виникають під час фактичного виконання програми, і вимагають часу для аналізу коду та документації. | Динамічні методики дозволяють оцінювати функціональність та продуктивність програмного забезпечення в реальних умовах, забезпечуючи високу якість кінцевого продукту. Вони ефективні для виявлення складних помилок, але можуть займати багато часу і потребують значних ресурсів. Крім того, баги виявляються на пізніх етапах розробки, що може підвищити витрати на їх виправлення. | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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.   **Відповідь:** Як на мене, коректним є перше твердження**(a)**, оскільки при наявності однієї умови "IF" і відсутності циклів або перемикачів, тестування дасть або істинний, або хибний результат. Це забезпечує 100% покриття тверджень, але лише 50% покриття рішень, оскільки кожне рішення має дві гілки, які потрібно перевірити для повного покриття.   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   **Моя відповідь варіант a:**  1 тест-кейс для покриття операторів: потрібно виконати всю послідовність дій.  2 тест-кейси для покриття рішень: один тест для випадку, коли MS Word успішно запускається (і вірш пишеться), і один для випадку, коли MS Word не запускається (вірш не пишеться).   1. Скільки потрібно тестів для перевірки тверджень коду:     1. 2    2. 1    3. 3    4. 4   **Відповідь:** Я вважаю, достатньо буде створити **3(c)** тест-кейси для перевірки тверджень коду:  1. P+Q≤100 і P≤50  2. P+Q>100  3. P>50 |
| Програма максимум:   1. Виконай завдання двох попередніх рівнів. 2. Продовжуємо розвивати стартап для застосунку, який дозволяє обмінюватися фотографіями котиків.   Є алгоритм:  *Запитай, якого улюбленця має користувач.*  *Якщо користувач відповість, що має кота, то запитай, яка порода його улюбленця: «короткошерста чи довгошерста?»*  *Якщо клієнт відповість «довгошерста», то запитай: «ви бажаєте отримати контакти найближчого грумера?»*  *Якщо клієнт відповість «так», то скажи: «Надайте адресу найближчої котячої перукарні»*  *Інакше*  *Скажи: «Запропонуй магазин з товарами по догляду за шерстю»*  *Закінчити*  *Інакше*  *Скажи «Запропонуй обрати магазин із зоотоварами»*  *Закінчити*  *Якщо клієнт не має кота*  *Скажи “Коли вирішите завести улюбленця – приходьте”*  *Закінчити*  Завдання:   1. Намалюй схему алгоритму (в інструменті на вибір, наприклад, у вбудованому Google Docs редакторі, [figjam](https://www.figma.com/figjam/) чи [miro](https://miro.com/)) 2. Який потрібен мінімальний набір тест-кейсів, щоб переконатися, що всі запитання були поставлені, всі комбінації були пройдені та всі відповіді були отримані? |