3. Методи білого, чорного та сірого ящика.

3.1 Метод «білого ящика».

При тестуванні методом «білого ящика» розробник тесту має доступ до вихідного тестируемому кодом і може компонувати з ним код тестів. Така ситуація типова для модульного тестування, при якому тестуються тільки окремі частини системи. При тестуванні методом «білого ящика» може використовуватися знання про внутрішній устрій перевіряється ПЗ, у тому числі і для організації перевірки обробки допустимих, граничних і некоректних даних. Крім того, даний вид тестування дозволяє оцінити рівень покриття коду тестами.

Термін білий ящик означає, що при розробці тестових випадків тестувальники використовують будь-які доступні відомості про внутрішню структуру чи коді. Технології, застосовувані під час тестування білого ящика raquo ;, зазвичай називають технологіями статичного тестування.

Цей метод не ставить мети виявлення синтаксичних помилок, так як дефекти такого роду звичайно виявляє компілятор. Методи білого ящика спрямовані на локалізацію помилок, які складніше виявити, знайти і зафіксувати. З їх допомогою можна виявити логічні помилки і перевірити ступінь покриття тестами.

Тестові процедури, пов'язані з використанням стратегії білого ящика, використовують керуючу логіку процедур. Вони надають ряд послуг, в тому числі:

· Дають гарантію того, що всі незалежні шляхи в модулі перевірені принаймні один раз.

· Перевіряють всі логічні рішення на предмет того, істини вони чи хибні.

· Виконують всі цикли всередині операційних кордонів і з використанням граничних значень.

· Досліджують структури внутрішніх даних із цілі перевірки їх достовірності.

Тестування допомогою білого ящика, як правило, включає в себе стратегію модульного тестування, при якому тестування ведеться на модульному або функціональному рівні та роботи з тестування направлені на дослідження внутрішнього устрою модуля.

10

Даний тип тестування називають також модульним тестуванням, тестуванням прозорого ящика (clear box) або прозорим (translucent) тестуванням, оскільки співробітники, які проводять тестування, мають доступ до програмного коду і можуть бачити роботу програми зсередини. Даний підхід до тестування відомий також як структурний підхід.

На цьому рівні тестування перевіряється керуюча логіка, що виявляється на модульному рівні. Тестові драйвери використовуються для того, щоб всі шляхи в даному модулі були перевірені хоча б один раз, всі логічні рішення розглянуті у всіляких умовах, цикли були виконані з використанням верхніх і нижніх меж і роконтроліровани структури внутрішніх даних.

Методи тестування на основі стратегії білого ящика:

· Ввід невірних значень. При введенні невірних значень тестувальник змушує коди повернення показувати помилки і дивиться на реакцію коду. Це хороший спосіб моделювання певних подій, наприклад переповнення диска, нестачі пам'яті й т.д. Популярним методом є заміна alloc () функцією, яка повертає значення NULL в 10% випадків з метою з'ясування, скільки збоїв буде в результаті. Такий підхід ще називають тестуванням помилкових вхідних даних. При та...

кому тестуванні перевіряється обробка як вірних, так і невірних вхідних даних. Тестувальники можуть вибрати, які значення перевіряють діапазон вхідних/вихідних параметрів, а також значення, що виходять за межу діапазону.

· Модульне тестування. При створенні коду кожного модуля програмного продукту проводиться модульне тестування для перевірки того, що код працює вірно і коректно реалізує архітектуру. При модульному тестуванні новий код перевіряється на відповідність докладного опису архітектури; обстежуються шляху в коді, встановлюється, що екрани, спадаючі меню і повідомлення належним чином відформатовані; перевіряються діапазон і тип даних, що вводяться, а також те, що кожен блок коду, коли потрібно, генерує виключення і повертає помилки (Еггог returns). Тестування кожного модуля програмного продукту проводиться для того, щоб перевірити коректність алгоритмів і логіки і те, що програмний модуль задовольняє пропонованим вимогам і забезпечує необхідну функціональність. За підсумками модульного тестування фіксуються помилки, що відносяться до логіки програми, перевантаження і виходу з діапазону, часу роботи і витоку пам'яті.

· Тестування обробки помилок. При використанні цього методу зізнається, що нереально на практиці перевірити кожне можливе умова виникнення помилки

11

З цієї причини програма обробки помилок може згладити наслідки при виникненні несподіваних помилок.

Тестувальник зобов'язаний переконатися в тому, що додаток належним чином видає повідомлення про помилку. Так, додаток, що повідомляє про системну помилку, що виникла через проміжного програмного забезпечення представляє невелику цінність, як для кінцевого користувача, так і для тестувальника.

· Витік пам'яті. При тестуванні витоку пам'яті додаток досліджується з метою виявлення ситуацій, при яких додаток не звільняє виділену пам'ять, внаслідок чого знижується продуктивність або виникає тупикова ситуація. Дана технологія застосовується як для тестування версії додатка, так і для тестування готового програмного продукту. Можливе застосування інструментів тестування. Вони можуть стежити за використанням пам'яті програми протягом декількох годин або навіть днів, щоб перевірити, чи буде зростати обсяг використовуваної пам'яті. З їх допомогою можна також виявити ті оператори програми, які не звільняють виділену пам'ять.

· Комплексне тестування. Метою комплексного тестування є перевірка того, що кожен модуль програмного продукту коректно узгоджується з іншими модулями продукту. При комплексному тестуванні може використовуватися технологія обробки зверху вниз і знизу вгору, при якій кожен модуль, що є листом в дереві системи, інтегрується з наступним модулем нижчого або більш високого рівня, поки не буде створено дерево програмного продукту. Ця технологія тестування спрямована на перевірку не тільки тих параметрів, які передаються між двома компонентами, але й на перевірку глобальних параметрів і у випадку об'єктно-орієнтованого додатки, всіх класів верхнього рівня.

· Тестування ланцюжків. Тестування ланцюжків передбачає перевірку групи модулів, що складають функцію програмного продукту. Ці дії відомі ще як модульне тестування, з його допомогою забезпечується адекватне тестування компонентів системи. Дане тестування виявляє, чи достатньо надійно працюють модулі для того, щоб утворити єдиний модуль, і чи видає модуль програмного продукту точні і що погодяться.

· Дослідження покриття. При виборі інструмента для дослідження покриття важливо, щоб група тестування проаналізувала тип покриття, необхідний для програми. Дослідження покриття можна провести за допомогою різних технологій. Метод покриття операторів часто називають С1, що також означає покриття вузлів. Ці виміри показують, чи був перевірений кожен виконуваний оператор. Даний метод тестування зазвичай використовує програму протоколювання (profiler) продуктивності.

12

· Покриття рішень. Метод покриття рішень спрямований на визначення (у відсотковому співвідношенні) всіх можливих результатів рішень, які були перевірені за допомогою комплекту тестових процедур. Метод покриття рішень іноді відносять до покриттю гілок і називають С2. Він вимагає: щоб кожна точка входу і виходу в програмі була досягнута хоча б один раз, щоб всі можливі умови для рішень в програмі були перевірені не менше одного разу і щоб кожне рішення в програмі хоча б раз було протестовано при використанні всіх можливих результатів.

· Покриття умов. Покриття умов схоже на покриття рішень. Воно спрямоване на перевірку точності істинних або помилкових результатів кожного логічного виразу. Цей метод включає в себе тести, які перевіряють вираження незалежно один від одного. Результати цих перевірок аналогічні тим, що одержують при застосуванні методу покриття рішень, за винятком того, що метод пок риття рішень більш чутливий до керуючої логіці програми.