Лекція №10 ПОіП « ТЕСТУВАННЯ ПРОГРАМНИХ ВИРОБІВ»
(Модуль 2 -)

План лекції.

Основні поняття і принципи тестування

Для створення програмних виробів застосовуються технології, які містять сукупність виробничих процесів, методів і засобів, призначених для розробки програм із заданими показниками якості. Будь-який виріб тестується на відповідність цим показникам.

Стосовно до програмного виробу тестування — це процес багаторазового виконання програми з метою виявлення помилок.

Основним методом виявлення помилок під час налагодження програм є їх тестування. При цьому витрати на тестування для виявлення помилок є найбільшими, досягають 30—40 % загальних витрат на розробку програм і значною мірою визначають якість створеного програмного продукту.

Особливостями тестування програмного виробу є [28]:

– відсутність еталона (програми), якому повинна відповідати програма, що тестується;

– висока складність програм і теоретична неможливість вичерпного тестування;

– практична неможливість створення єдиної методики тестування (формалізації процесу тестування) внаслідок різноманітності програмних виробів за їх складністю, функціональним призначенням, сферою використання і т. ін.

Для підвищення ефективності тестування розроблені різні методи систематичного і регламентованого тестування, враховуючі особливості програм, що створюються.

Основна мета тестування для виявлення помилок — визначення всіх відхилень результатів функціонування реальної програми від заданих еталонних значень. Ефективними є операції тестування, які виявляють максимальну кількість помилок у програмі, що обґрунтовує витрати на їх виконання.

Задачі тестування доцільно розділити на групи відповідно до трьох стадій тестування:

- тестування для виявлення помилок у програмі;
- тестування для діагностики і локалізації причин виявлених перекручень результатів;

– тестування для контролю виконаних коректувань програм і даних.

Після тестування для виявлення помилок застосовується тестування для їх діагностики і локалізації. Його цілі й методи дещо відрізняються від тих, які відповідають першій стадії тестування. На цій стадії найважливіше завдання — точно встановити місце перекручень програми або даних, що стало причиною відхилення результатів від еталонних. Тобто визначити, яку частину програми необхідно коректувати. На цій стадії витрати виправдані, а тестування вважається ефективним, якщо воно приводить до повної локалізації помилки, що підлягає виправленню.

Після локалізації і усунення виявлених помилок застосовується контрольне тестування, яке має підтвердити правильність виконаного коректування програми і відсутність раніше виявлених помилок, а також відсутність повторних помилок, які можуть з'явитися під час коректування.

Домашинний контроль програми не гарантує її придатність для роботи, тому необхідний контроль програми за результатами її виконання на ЕОМ. Для кожного класу задач можуть існувати свої особливі способи контролю програм на ЕОМ. Розглянемо один метод контролю, який можна вважати універсальним для всіх класів задач, — метод контрольних тестів.

Тестом називають інформацію, що складається з вхідних даних, спеціально підібраних для програми, що налагоджується, і з відповідних їм еталонних результатів (не тільки остаточних, а й проміжних), що використовуються надалі для контролю правильності роботи програми.

Тестування базується на наступних загальних принципах [28]:

Принцип 1. Процес тестування більш ефективний, якщо проводиться не автором програми.

З визначення тестування як процесу ясно, що тестування тим ефективніше, чим більше помилок виявлено. При створенні програми неможливо свідомо залишати в ній помилки, це означає, що програміст заздалегідь налаштований на їх відсутність, тому підсвідомо не намагається їх знайти.

Це не означає, що програміст не може тестувати свою програму. Що стосується налагодження, то його ефективніше виконує сам автор програми.

Принцип 2. Тестовий набір даних повинен включати два компоненти: опис вхідних даних і опис точного і коректного результату, відповідного цьому набору вхідних даних.

Складність реалізації принципу полягає в тому, що під час тестування програми (модуля) необхідно для кожного вхідного набору даних вручну розрахувати очікуваний результат або знайти припустимий інтервал зміни вихідних даних, а це дуже трудомісткий процес навіть для невеликих програм.

Принцип 3. Необхідно ретельно вивчати результати застосування кожного тесту.

Значну частину всіх помилок можна виявити внаслідок перших тестових прогонів.

Принцип 4. Тести для неправильних і непередбачених вхідних даних повинні розроблятися так само ретельно, як для правильних, передбачених.

При обробці даних в програмі, що тестується, необхідно передбачити діагностику неприпустимих значень даних і повідомлення про причину неможливості їх обробки.

Принцип 5. Необхідно перевіряти не тільки, чи робить програма те, для чого вона призначена, але і чи не робить вона те, що не повинна робити.

Необхідно будь-яку програму перевірити на небажані побічні дії.

Принцип 6. Імовірність присутності не виявлених помилок у частині програми пропорційна до кількості помилок, уже виявлених у цій частині.

Ця властивість помилок групуватися пояснюється тим, що ті частини програми, де під час тестування виявлено більше помилок, або мають слабку ідеологію, або розроблялися програмістами більш низької кваліфікації. З цього принципу витікає практичний висновок: якщо в якій-небудь частині програми виявлено більше помилок, ніж в інших, то її необхідно тестувати більш ретельно.

Способи тестування

Контроль програми зводиться до того, щоб підібрати таку систему тестів, отримання правильних результатів для якої гарантувало б правильну роботу програми і для інших початкових даних із сфери, вказаної в технічному завданні. Для реалізації методу контрольних тестів мають бути виготовлені або заздалегідь відомі еталонні результати, на основі зіставлення яких з отриманими тестовими результатами можна було б зробити висновок про правильність роботи програми на даному тесті. Еталонні тестові результати для обчислювальних задач можна отримати, здійснюючи обчислення вручну, застосовуючи результати, здобуті раніше на інший ЕОМ, або по іншій програмі, або використовуючи відомі факти, властивості, фізичні закони.

Є три способи тестування*: алгоритмічне*, *аналітичне*, *змістовне.*

Алгоритмічне тестування застосовується для контролю етапів алгоритмізації та програмування. Програмісти проектують тести і готують еталонні результати на етапі алгоритмізації, а використовують їх на етапі налагодження.

Функціональне або *аналітичне тестування* використовується для контролю вибраного методу розв'язування задачі, правильності його роботи у заданих режимах із установленими діапазонами даних. Тести проектують і створюють на етапі проектування після вибору методу, а використовують їх на останньому етапі налагодження або для аналізу результатів пробного розрахунку; в ході тестування застосовуються ще й якісні оцінки результатів.

Змістовне тестування використовують для перевірки правильності постановки задачі, принципи якої формулюються в технічному завданні. Для контролю застосовуються, як правило, якісні оцінки і статистичні характеристики програми, фізичне значення отриманих результатів і т. ін. У проведенні змістовного тестування активну участь повинні брати замовники або майбутні користувачі програми.

Змістовні і аналітичні тести перевіряють правильність роботи програми загалом або великих її частин, в той час як алгоритмічні тести перевіряють роботу окремих блоків або операторів програми. Алгоритмічні тести, крім встановлення наявності помилок у програмі, мають давати певну інформацію також про їх місцезнаходження у

програмі, тобто локалізувати помилки на наступному етапі. Розробляючи систему тестів, треба намагатися, щоб успішний прогін її на ПК доводив відсутність помилок у програмі (або окремому її блоці), хоч виготовлення і прогін усіх тестів, необхідних для доказу, може зробити етап контролю досить довгим. Тому при розробці системи тестів постає задача мінімізації кількості необхідних тестових результатів, машинного часу і зусиль програміста. Здебільшого в разі використання методу тестування можна говорити про відсутність помилок лише для невеликих блоків (модулів) програми, а для цілої програми обмежуватися ймовірністю відсутності помилок у програмі.

Методи тестування

Методи тестування спрямовані на виявлення максимального числа помилок у найбільш важливих режимах функціонування програм при обмежених ресурсах.

Розглянемо детальніше методи тестування, що послідовно застосовуються: *статичний*, *детермінований*, *стохастичний* і *в реальному масштабі часу.*

Статичне тестування — найбільш формалізоване і автоматизовуване, базується на правилах структурної побудови програм і обробки даних. Перевірка виконання цих правил проводиться без виконання об'єктного коду програми формальним аналізом тексту програми мовою програмування. Статичне тестування реалізовується ручними методами тестування програм, написаних за вимогами структурного і модульного програмування. Це зробило програми зручними для читання і дало змогу провести тестування окремих модулів вручну без використання ЕОМ.

Використання ручних методів тестування досить ефективне, вони сприяють істотному збільшенню продуктивності і підвищенню надійності програм, дозволяють раніше виявити помилки, а отже зменшити вартість виправлення. У разі ручних методів тестування ймовірність того, що при виправленні помилок не вносяться нові помилки, набагато вища.

Основні методи ручного тестування: *інспекції початкового тексту* і *наскрізний перегляд*. Оператори і операнди тексту програми аналізуються в символьному вигляді, тому цей метод тестування іноді називають символічним тестуванням.

Детерміноване тестування є найбільш трудомістким і деталізованим, воно вимагає багаторазового виконання програми на ЕОМ з використанням визначених, спеціально підібраних тестових наборів даних. При детермінованому тестуванні контролюється кожна комбінація вхідних даних і відповідні їй результати для виявлення відхилень від еталонних значень, а також кожне затвердження у специфікації тестованої програми.

Детермінований метод тестування внаслідок трудомісткості можливо застосовувати для окремих модулів у процесі збирання програми або для невеликих і нескладних програмних комплексів.

При тестуванні програмного виробу неможливо перебрати всі комбінації початкових даних і проконтролювати результати функціонування на кожній з них, тому для комплексного тестування програмного виробу застосовується стохастичне тестування.

Стохастичне тестування передбачає використання як початкових даних множини випадкових величин з відповідними розподілами, а для порівняння отриманих результатів використовуються ті самі розподіли випадкових величин.

Стохастичне тестування застосовується, здебільшого, для виявлення помилок, а для діагностики і локалізації помилок необхідно використати детерміноване тестування з конкретними значеннями початкових даних зі сфери вже отриманих випадкових величин. Стохастичне тестування легше інших піддається автоматизації використанням генераторів випадкових величин.

Тестування в реальному масштабі часу є обов'язковим для програмних виробів, призначених для роботи в системах реального часу. У процесі такого тестування перевіряються результати обробки початкових даних з урахуванням часу їх надходження, тривалості і пріоритетності обробки, динаміки використання пам'яті і взаємодії з іншими програмами. Якщо виявлено відхилення результатів виконання програм від очікуваних, то для локалізації помилок фіксують час і переходять до детермінованого тестування.

Правила тестування

Досвід багатьох програмістів дозволив сформулювати деякі загальні правила тестування:

Прохід ділянок. Кожна лінійна ділянка програми повинна бути обов'язково пройдена при виконанні, принаймні, одного тесту. Оскільки це правило використовується для поблокового контролю, постає питання про автоматизацію обліку пройдених (і непройдених) ділянок програми.

За допомогою спеціально підібраних тестових вхідних даних можна перевірити роботу окремих блоків надійніше, ніж з реальними даними, які отримують блоки від сусідніх блоків, оскільки такі умовні дані дають змогу обійти виконання (і контроль) протестованих раніше ділянок або блоків. Такий незалежний контроль роботи окремих блоків буває можливий, якщо вони володіють властивістю модульності, яка закладається ще під час розробки алгоритму. Якщо виконання якоїсь ділянки змінює порядок виконання або характер роботи інших ділянок, може знадобитися багаторазова перевірка ділянок програми, і навіть перегляд всіх гілок програми; багаторазова перевірка потрібна, зокрема, і для ділянок, що містять змінні з індексами.

Точність перевірки. Контроль арифметичних блоків (як і інших блоків) проводиться шляхом порівняння результатів, отриманих при виконанні блоку, з еталонними результатами. Для арифметичних результатів додатково визначають точність, з якою необхідно порівнювати еталонні і тестові результати, щоб можна було пересвідчитися у правильності роботи блоку. Складність полягає в тому, що величини, які входять у певний арифметичний вираз, що перевіряється у блоці, залежно від співвідношення їхніх значень і характеру операцій, що виконуються з ними, вкладають різний внесок у результат.

Щоб бути упевненим в тому, що правильний числовий результат, отриманий на ПК, свідчить про правильність програми, необхідно стежити за проміжними результатами обчислень, які не повинні виходити за певний діапазон, що

встановлюється залежно від точності обчислень еталонних результатів. Виконання такої вимоги може призвести до необхідності багаторазової перевірки виразу для різних діапазонів даних.

Мінімальність обчислень. Коли тривалість роботи контрольованої програми залежить від яких-небудь параметрів, то під час контролю їх потрібно вибирати такими, щоб вони мінімізували кількість обчислень. До таких параметрів, наприклад, можна віднести крок або відрізок інтегрування, порядок матриці або кількість елементів вектора, довжину символьних рядків, точність для ітераційних обчислень і т. ін. Значення початкових даних треба вибирати такими, щоб виготовлення еталонних результатів вручну було достатньо легким. Наприклад, спочатку можна взяти дані цілочисельними або такими, щоб при перевірці виразів деякі їх доданки, вже перевірені раніше, оберталися на нуль.

Достовірність еталонів. Треба звернути увагу і на достовірність процесу отримання еталонних результатів. Вони повинні обчислюватися не самим програмістом, а кимось іншим, щоб одні й ті самі помилки у завданні на програмування або в алгоритмі не проникли і в програму, і в еталонні результати. Якщо тести готує сам програміст, то є небезпека мимовільної підгонки обчислюваних значень під бажані, отримані раніше на машині. Як еталонні результати можна використати і дані, отримані під час прокручування програми.

Планування. Основою ефективного тестування є його плановість і систематичність. Тільки діючи за планом, що враховує структуру і особливості розроблюваної програми, можна сподіватися швидко виявити всі (або майже всі) помилки у програмі і переконливо продемонструвати замовникові правильність тестованої програми.

У разі планового підходу перевіряється блок за блоком (для алгоритмічного тестування), режим роботи програми за режимом (для функціонального і змістовного контролю). Плануються як загальний підхід до контролю програми і основні його принципи (стратегія контролю), так і особливості реалізації вибраних принципів, наприклад розв’язується задача мінімізації кількості тестів (тактика контролю). Таким чином, заздалегідь встановлюється, що необхідно проконтролювати і як це краще виконати.

Генерація тестових наборів даних

Детерміноване тестування, або тестування на певних вхідних значеннях, ґрунтується на двох підходах: *структурне тестування* (СТ) і *функціональне тестування* (ФТ).

Структурне тестування (стратегія «білого ящика») передбачає відповідно до логіки програми побудову таких вхідних наборів даних, які під час багаторазового виконання програми на ЕОМ забезпечили б виконання максимально можливої кількості маршрутів, всіх логічних розгалужень, циклів, усіх операторів, всіх умов (і їх комбінацій) і т. д.

Під час побудови тестових наборів даних за принципом «білого ящика» керуються такими критеріями: покриття операторів, покриття вузлів розгалуження, покриття умов, комбінаторне покриття умов.

Це означає, що кожний оператор і кожний вузол і розгалуження мають бути виконані хоча б один раз, перевірені всі умови та їх комбінації.

Функціональне тестування (стратегія «чорного ящика» або тестування за «входом-виходом») повністю абстрагується від логіки програми: передбачається, що програма — це «чорний ящик», а тестові дані вибираються на основі аналізу вхідних функціональних специфікацій програми.

До стратегії «чорного ящика» належать методи: *еквівалентного розбиття*; *аналізу граничних значень*; *функціональних діаграм.*

Метод еквівалентного розбиття полягає в тому, що виділяються класи еквівалентності шляхом аналізу вхідної умови і розбиттям її на дві або більше групи. Для будь-якої умови існують правильний клас еквівалентності (що представляє правильні вхідні дані програми) і неправильний, тобто помилкові вхідні значення. На основі класів еквівалентності будуються тестові набори. Причому для правильних класів еквівалентності кожний тест повинен покривати якомога більше правильних класів еквівалентності.

Для кожного неправильного класу еквівалентності будується принаймні один тестовий набір.

Аналіз граничних значень передбачає перевірку ситуацій, що виникають на межах і поблизу меж еквівалентного розбиття. Наприклад, якщо правильна область значень є –1,0 до +1,0, то треба передбачити тести –1,0, 1,0, –1,001 і 1,001.

Метод функціональних діаграм полягає в тому, що створюється набір тестів, що визначаються функціональними задачами та їх зв’язками в програмному комплексі. Ці тести мають забезпечити перевірку якості рішення функціональних задач, сформульованих у технічному завданні, перевірку стійкості функціонування за наявності аномалій на вході у програму, тобто в разі аномалій у зовнішньому середовищі. Це особливо важливо під час тестування великих програмних комплексів, коли помилки розв’язування однієї функціональної задачі можуть заблокувати виконання всіх програмних модулів, пов’язаних з нею функціонально.

Упорядкування при тестуванні. Для тестування застосовуються методи, що передбачають упорядкування та систематизацію тестів за різними стратегіями і параметрами, і методи неврегульованого тестування. У разі неврегульованого тестування початкові дані, що імітують зовнішнє середовище, випадковим чином генеруються у всьому діапазоні можливої зміни параметрів, проводиться випадковий перебір значень у довільних комбінаціях різних величин. При цьому багато які значення початкових даних характеризуються малою ймовірністю виявлення помилок і не виправдовують витрат на виконання тестування. Крім того, можливо поява даних, логічно суперечливих. Проте дані, найбільш важливі з позиції реального використання програм і можливості виявлення помилок, можуть виявитися не охопленими в процесі тестування. При реально існуючих обмеженнях на об’єми тестування його неврегульоване застосування виявляється малоефективним і майже не знаходить застосування.

Прагнення до раціонального використання обмежених ресурсів приводить до систематизації процесу і методів тестування. Методи впорядкованого тестування базуються на виділенні чинників і параметрів, що дають змогу ефективно розподіляти ресурси тестування з урахуванням їх впливу на якість програм. Систематизація може значно змінюватися залежно від етапів тестування, однак можна виділити декілька загальних принципів, на базі яких будуються основні методи тестування. Для упорядкування операцій тестування використовується інформація про структуру програми і процес обробки інформації, про характер зміни і взаємозв'язку змінних, про найбільш вірогідні і важливі комбінації початкових даних, про характеристики помилок і ймовірність їх виявлення і т. ін. Таким чином обмежені ресурси тестування використовуються передусім для виявлення найбільш небезпечних помилок в найбільш важливих режимах функціонування програм. З цією метою послідовно застосовуються методи тестування: статичний, детермінований, стохастичний і в реальному масштабі часу.

У реальних умовах на вхід програми можуть потрапити сильно перекручені або помилкові дані. Програми повинні зберігати свою працездатність після надходження таких даних або відновлювати її при подальшому надходженні даних, що змінюються в заданих межах. Тому тестування необхідно провести не тільки в разі коректних початкових даних, а й у разі перекручених. У зв'язку з цим впорядкований підхід до тестування може базуватися на критеріях досягнення максимальної коректності або максимальної надійності програм.

Типи тестів

Розглянутий вид контролю можна назвати тестуванням основних функціональних можливостей програми — *основний тест*. Крім основного тесту можна виокремити такі види тестів.

Вироджений тест слугує для перевірки правильності виконання зовнішніх функцій програми, наприклад, звернення до неї і вихід з неї.

Тест граничних значень перевіряє роботу програми для граничних значень параметрів, що визначають обчислювальний процес, оскільки для граничних значень параметра робота програми має особливий характер, що вимагає і особливого контролю.

Аварійний тест перевіряє реакцію програми на виникнення різного роду аварійних ситуацій в програмі, зокрема, викликаних неправильними початковими даними. Тобто перевіряється закінчення її роботи або виправлення неправильних початкових даних.

Крім автономних тестів, призначених для контролю окремих блоків програми, існують стикові і комплексні тести. *Стикові тести* призначаються для перевірки взаємозв'язку (стиковки) вже налагоджених частин програми. *Комплексні тести* перевіряють правильність роботи всіх або більшості частин програми після їх об'єднання.

Загальні характеристики об'єктів на етапах тестування

Об’єкти і технології тестування змінюються залежно від етапу створення програми [14]. З урахуванням цього можна виокремити такі об’єкти:

- специфікації вимог на програмні модулі, групи програм і комплекс;
- програмні модулі, підготовлені до налагодження;
- групи програм, що розв’язують функціональні задачі;
- комплекс програм, для якого виконуються всі види налагодження;
- програмний виріб, що підлягає випробуванням перед передачею його в експлуатацію;
- програмний продукт, що супроводжується до завершення його життєвого циклу.

Ці об’єкти різняться складністю для тестування, рівнем теоретичної розробки методів і наявним станом автоматизації процесу тестування.

Розглянемо основні задачі об’єктів поетапного тестування, а також категорії тестів, що застосовуються (рис. 6.1). Відповідно до задач для кожного об’єкта можна виокремити кілька категорій тестів, кожна з них призначена для виявлення помилок певного класу.

Тестування специфікацій вимог. Програмні специфікації вимог або формалізовані технічні завдання на комплекси програм та їх компоненти використовуються для опису основних функцій програм і їх взаємодії. Програмні специфікації на модулі і групи програм визначають функції цих компонент і зв’язки між ними з управління і з інформації. Основна мета тестування специфікації вимог:

- перевірка повноти і взаємної відповідності функцій, що реалізуються програмними компонентами;
- перевірка відповідності інформації на входах і виходах програмних модулів і груп програм, що взаємодіють;
- результат тестування специфікацій — перевірка їх коректності в межах загального опису функцій і взаємодії відповідних програмних компонентів.

Рис. 6.1. Загальна схема тестування програмного модуля

Тест перевірки повноти і узгодженості функцій програмних компонент на рівні специфікацій призначений для виявлення помилок функціональних компонент при їх представленні програмними специфікаціями. Тестування доцільно провести по спадному методу, починаючи від специфікації комплексу або групи програм. Тестування виконують зазвичай вручну переглядом програмних специфікацій за маршрутами виконання груп програм при вирішенні певних функціональних задач. Коригуванню підлягають програмні специфікації модулів і груп програм або вводяться нові компоненти комплексу програм з функціями, яких бракує.

Тест перевірки узгодженості інтерфейсу в специфікаціях програмних компонент застосовується для виявлення помилок в описах змінних і передачах управління при взаємодії модулів і груп програм. За структурною схемою комплексу або групи програм встановлюється перелік модулів, що послідовно викликаються, і перевіряється коректність описів викликів у програмних специфікаціях.

Тестування програмних модулів — найбільш формалізований і автоматизований процес. Основна задача — перевірка обробки програмними модулями інформації, що надходить, і коректності вихідних даних відповідно до функцій, заданих у специфікаціях. При цьому повинна бути перевірена коректність структури модулів і їх основних конструктивних компонент: циклів, блоків, перемикачів.

Тест перевірки структури програмного модуля призначений для виявлення помилок у схемі прийняття рішень і логіці функціонування модуля. Перевіряються маршрути обробки інформації у модулі і правильність їх реалізації залежно від початкових даних. Повнота тесту визначається критеріями виділення маршрутів для тестування та мірою покриття маршрутів виконання програми. Виокремлення і

упорядкування маршрутів для тестування може бути автоматизоване, так само як і контроль повноти охоплення маршрутів тестами.

Тест перевірки обчислень і перетворення даних програмним модулем служить для виявлення помилок в обчислювальній частині програми. Для цього виділяються компоненти програмного модуля і маршрути обробки даних, що містять обчислення і перетворення змінних. До складу тесту входять набори значень початкових змінних у всій сфері їх визначення. При цьому виокремлюються комбінації змінних у критичних і особливих точках, а також поєднання суперечливих і перекручених даних, коли найчастіше виявляються помилки. Еталонами слугують результати попередніх розрахунків за формулами, закладеними в модулі. Виокремлення обчислювальної частини програми, відповідних маршрутів, а також деяких особливих значень змінних може бути частково автоматизовано.

Тест перевірки повноти функцій, що виконуються модулем, призначений для виявлення помилок у розробленій програмі відносно всіх функцій, заданих в програмній специфікації. Для перевірки функцій використовуються результати тестування структури і обчислень, які доповнюються тестами для перевірки ще не контрольованих функцій, які звичайно містять різні перекручення й аномальні дані. Порівняння функцій створеного програмного модуля з програмною специфікацією важко автоматизувати, тож воно виконується зазвичай уручну.

Тестування функціональних груп програм призначене для перевірки коректності рішення великих автономних функціональних задач, управляючих та інформаційних зв’язків між модулями, а також коректності функцій, що обчислюються в процесі обробки інформації. При цьому зростає складність об’єктів, що тестуються і, відповідно, обсяг тестів. Детермінованим тестуванням перевіряються структура груп програм і основні маршрути обробки інформації. Іноді основні результати дістають методами стохастичного тестування, які слабко формалізовані й автоматизовані.

Тест перевірки структури групи програм застосовується для виявлення помилок реальної структурної побудови групи програм і його відповідності специфікації. Перевіряється правильність викликів програмних модулів і повернення управління під час взаємодії у групі програм. Структура, що реалізується, може бути побудована автоматизовано за паспортами розроблених програмних модулів.

Тест перевірки міжмодульного інтерфейсу в групі програм призначений для виявлення помилок інформаційних зв’язків між модулями в групі та іншими групами програм під час взаємодії з нею. Кожна змінна і масив змінних перевіряються на тотожність описів у всіх модулях, де вони використовуються, а також на відповідність початковим програмним специфікаціям. Всі реалізовані інформаційні зв’язки можуть бути встановлені, впорядковані і узагальнені автоматизовано.

Тест перевірки виконання обмежень з використання пам’яті і тривалості виконання групи програм призначений для виявлення помилок використання реальних ресурсів ЕОМ у типових режимах виконання програм, у критичних режимах, а також у разі різних пошкоджень вхідної інформації. Для виявлення виходу використовуваного обсягу пам’яті за межі, встановлені в специфікаціях і в описах змінних. Оцінюється ймовірність перевищення заданих меж тривалості обробки даних. Перевірки

проводяться, в основному, стохастичні, тому великого значення набувають автоматизоване формування тестів і автоматична обробка результатів тестування.

Тест перевірки повноти рішення функціональних задач групою програм служить для завершального контролю групи програм, перевірки міжмодульного інтерфейсу всередині групи програм, перевірки функціонування групи у всіх заданих режимах і виявлення помилок, що залишилися після попередніх категорій тестів. Тестування завершується реєстрацією отриманих результатів і оформляється протоколом тестування для передачі групи програм на комплексування.

Тестування комплексу програм під час налагодження — найбільш складний процес, який використовує переважно стохастичне тестування і в реальному масштабі часу з основною метою перевірки коректності та надійності функціонування всього комплексу програм при правильних і пошкоджених початкових даних, перевірки надійності функціонування всього комплексу програм у реальних і критичних умовах і ефективності засобів програмного захисту і відновлення. Формалізація процесу тестування на цьому етапі найважча.

Тест перевірки повноти рішення функціональних задач комплексом програм за типових початкових даних призначений для виявлення помилок функціонування в типових умовах, визначених технічним завданням на комплекс програм. Для систем реального часу тест містить, в основному, динамічні і стохастичні дані. Ці дані імітуються моделями реальних об'єктів. Результати тестування обробляються і порівнюються з еталонами, в основному, автоматично. Для модулів, в яких помилки виявлено, потрібно провести додаткове тестування.

Тест перевірки функціонування програм в критичних ситуаціях за умовами і логікою розв'язання задач призначений для виявлення помилок при виконанні програм в ситуаціях, які не часто реалізовуються, але важливі для функціонування системи обробки інформації, коли можна очікувати пошкодження результатів. Такі стресові випадки підготовляються вручну або передбачається їх реалізація у складі даних стохастичних і динамічних тестів.

Тест перевірки коректності використання ресурсів пам'яті і продуктивності обчислювальної системи служить для виявлення помилок виконання програм за браком пам'яті (наприклад, буферних накопичувачів) і продуктивності. Тестування робиться, здебільшого, у стохастичному режимі в реальному масштабі часу за підготовленими сценаріями, що імітують дефіцит одного з ресурсів системи. Внаслідок тестування встановлюються реальні характеристики комплексу програм на вибраній обчислювальній системі. У разі короткочасних перевантажень пам'яті або продуктивності має забезпечуватися захист від відмов і відновлення нормального режиму при подальшому зниженні завантаження.

Тест перевірки паралельного виконання програм використовується для виявлення помилок, зумовлених неузгодженим використанням початкових і проміжних даних, а також пристроїв обчислювальної системи при паралельному виконанні програм. Тестування проводиться переважно стохастично. Особливо важливо виявити всі тупикові ситуації, коли паралельні програми звертаються до одних і тих же ресурсів обчислювальної системи і не можуть продовжити обчислення до їх звільнення. Стохастичне тестування програм, що паралельно виконуються,

вимагає значної кількості випадкових і детермінованих початкових даних, які підготовлюються автоматично.

Тест перевірки ефективності захисту від перекручень початкових даних служить для виявлення помилок в програмах, які виявляються при помилкових або пошкоджених даних. При тестуванні виявляються ситуації порушення працездатності комплексу програм і зниження якості його функціонування в залежності від інтенсивності перекручень.

Тест визначення надійності комплексу програм призначений для вимірювання основних показників надійності під час реального функціонування комплексу програм. У процесі тестування за типових і критичних умов визначаються значення таких показників, як наробка на відказ, тривалість відновлення, коефіцієнт готовності й інших. Імітація початкових даних, реєстрація відказів і умов порушення роботоздатності може проводитися автоматично.

Тест оцінки ефективності захисту від збоїв апаратури і невиявлених помилок програм служить для перевірки якості засобів програмного контролю і оперативного відновлення за різних пошкоджень функціонування. При цьому оцінюються ймовірність виявлення ситуації відказу і середня тривалість відновлення. Спеціалізований тест оцінки ефективності захисту дозволяє визначити ймовірність виявлення кожного виду збоїв і відповідну йому тривалість відновлення нормального функціонування. Таким чином виявляються помилки програм контролю або програм відновлення, а також визначаються їхні динамічні характеристики.

Тестування комплексу програм при випробуваннях призначене, в основному, для перевірки відповідності технічному завданню і оцінювання придатності комплексу програм до регулярної експлуатації та супроводу. Перевірка придатності до супроводження охоплює тестування настроювання версій на умови конкретного застосування і аналіз зручності модифікування версій комплексу програм.

Тест випробувань на відповідність комплексу програм технічному завданню служить для паспортизації створеного комплексу як завершеного програмного продукту. Тестування забезпечує перевірку характеристик функціонування програм на кожну вимогу технічного завдання, оскільки для генерації тестів застосовуються переважно автоматичні імітатори, характеристики яких визначаються до початку тестування і відрізняються від характеристик реальних джерел інформації та об’єктів управління.

Тест перевірки зручності експлуатації і взаємодії людини з комплексом програм призначений для виявлення помилок представлення, які важко формалізуються, вихідних і результативних даних. Під час тестування оцінюються об’єм, зручність представлення і контролю початкових даних, зручність їх аналізу і використання, перевіряються динамічні характеристики введення і відображення даних в реальному часі, виявляються помилки розподілу функцій системи, що автоматизуються між людиною і ЕОМ, а також оцінюється можливість рішення задач обслуговуючим персоналом системи.

Тест перевірки зручності установлення і підготовки робочої версії служить для виявлення помилок методів і засобів настроювання адаптації комплексу програм до конкретних умов застосування. Для перевірки засобів адаптації створюються

спеціальні тести, що охоплюють найбільш типові режими використання комплексу програм.

Тест перевірки роботи комплексу програм при певних конфігураціях обладнання використовується для виявлення помилок, що виявляються при зміні складу або характеристик компонент обчислювальної системи. Число можливих конфігурацій обладнання може бути надто велике, щоб можна було усі протестувати під час підготовки програмного виробу. Тести мають забезпечувати перевірку цих засобів автоматизованої адаптації у всіх допустимих комплектаціях обладнання, а також тестування адаптованих версій.

Тест перевірки коректності документації призначений для виявлення помилок відповідності реального комплексу програм всій супроводжуючій його конструкторській та експлуатаційній документації. Тому ця категорія тестів є завершальною і призначена для встановлення якості документації. Найбільш повно в документації перевіряються контрольні приклади, що входять в інструкції з експлуатації. Здійснюється перевірка відповідності документації стандартам на проектування і оформлення програм, дотримання яких дозволяє підвищувати якість програм, і тільки тестуванням документації можна встановити міру дотримання стандартів.

Тест перевірки зручності супроводу і модифікації програм повинен забезпечувати виявлення тих помилок побудови комплексу програм і його компонент, які утруднюють їх зміну у процесі супроводження. Зручність супроводження закладається при системному і структурному проектуванні програм, при створенні модульно-ієрархічної структури на всіх рівнях ієрархії програм і даних. Основна мета таких структур полягає у можливості заміни або зміни окремих компонент програми без внесення змін в інші компоненти.

Тестування комплексу програм під час супроводження здійснюється з використанням практично всіх перелічених категорій тестів, характерних для розробки і випробувань комплексу програм. Взагалі супроводження є практично повторенням всього процесу створення програм або його окремих етапів.

Перераховані категорії тестів є основними і для комплексу програм реального часу, однак їх застосування залежить від класу програм, що розробляються. Організація та ефективне проведення такого загального систематичного тестування вимагають великих витрат і високої кваліфікації фахівців. Систематизація категорій тестів дає змогу послідовно зосередити увагу на виявленні помилок певних класів. Упорядковане застосування категорій тестів дозволяє ефективно використати обмежені ресурси тестування. Повні витрати на всі тести бувають виправдані і необхідні, коли комплекс програм виконує особливо відповідальні функції і його недостатня якість загрожує великими збитками або аваріями.

Контрольні питання

1. Мета тестування програм.
2. Зміст функціонального та структурного тестування.
3. Принципи тестування.
4. Чи обов'язкова наявність еталонних результатів у тесті?
5. Який тест називається вдалим?
6. Типи тестів для функціонального та структурного тестування.