ТЕМА 7. ПОНЯТТЯ ТА МЕТА ТЕСТУВАННЯ

1.1 Поняття та мети тестування

Існує декілька визначень понять, пов'язаних із тестуванням. З одного боку, тестування – це процес дослідження та випробування програмного продукту. Призначення тестування полягає в демонстрації розробникам і замовникам того, що програма відповідає вимогам, а також виявлення ситуацій, в яких поведінка програми є неправильною, небажаною або не відповідає специфікації. З іншого боку, тестування – це перевірка відповідності між реальною і очікуваною поведінкою програми. Воно здійснюється на кінцевому наборі тестів, обраному певним чином. Тестування в широкому сенсі – це одна з технік контролю якості, що включає в себе активності:

– з планування робіт (TestManagement);

– з проектування тестів (TestDesign);

– з виконання тестування (TestExecution);

– з аналізу отриманих результатів (TestAnalysis). Мети тестування полягають у наступному:

– підвищити ймовірність того, що інформаційна система, призначена для тестування, буде працювати правильно при будь-яких обставинах;

– підвищити ймовірність того, що інформаційна система, призначена для тестування, буде відповідати всім описаним вимогам;

– надати актуальну інформацію про стан продукту (інформаційної системи) на даний момент.

Існують різні способи класифікації тестування за метою, рівнем, призначенням, позитивністю, об'єктом та багатьма іншими характеристиками (Рис.1). Далі буде розглянуто найбільш використані способи класифікації видів тестування.

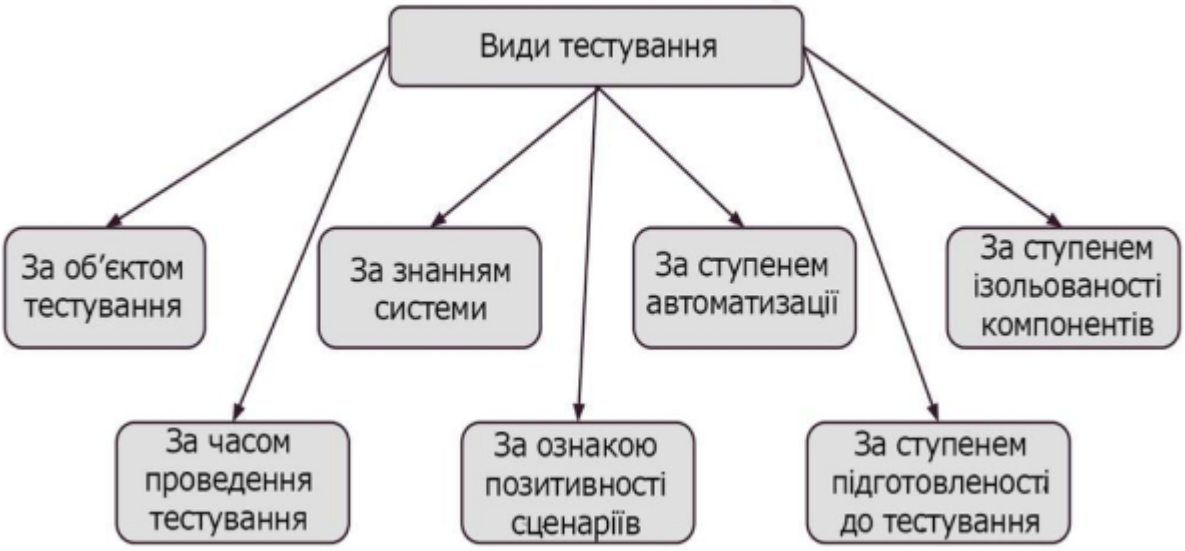

Рисунок 1 – Класифікація тестування

1.2 Класифікація за об'єктом тестування

За об'єктом тестування виділяють наступні види тестування:

- функціональне тестування;
- тестування продуктивності;
- тестування навантаження;
- стрес-тестування;
- тестування стабільності;
- конфігураційне тестування;
- юзабіліті-тестування;
- тестування інтерфейсу користувача;
- тестування безпеки;
- тестування локалізації;
- тестування сумісності.

Функціональне тестування розглядає заздалегідь вказане поведінку і ґрунтується на аналізі специфікацій функціональності компонента або системи в цілому

Тестування навантаження – це автоматизоване тестування, що імітує роботу певної кількості бізнес користувачів на якомусь загальному ресурсі.

Стресове тестування (StressTesting) дозволяє перевірити наскільки додаток і система в цілому працездатні в умовах стресу і також оцінити здатність системи до регенерації, тобто до повернення до нормального стану після припинення впливу стресу. Стресом в даному контексті може бути підвищення інтенсивності виконання операцій до дуже високих значень або аварійне зміна конфігурації сервера. Також одним із завдань при стресовому тестуванні може бути оцінка деградації продуктивності, таким чином цілі стресового тестування можуть перетинатися з цілями тестування продуктивності.

Тестування стабільності або надійності (Stability / ReliabilityTesting). Завданням тестування стабільності (надійності) є перевірка працездатності програми при тривалому (багатогодинному) тестуванні із середнім рівнем навантаження.

Конфігураційне тестування (ConfigurationTesting) - спеціальний вид тестування, спрямований на перевірку роботи програмного забезпечення при різних конфігураціях системи (заявлених платформах, підтримуваних драйвери, при різних конфігураціях комп'ютерів і т.д.)

Тестування зручності користування(юзабіліті) - це метод тестування, спрямований на встановлення ступеня зручності використання, навченості, зрозумілості і привабливості для користувачів продукту, що розробляється в контексті заданих умов. Сюди також входить тестування для користувача інтерфейсу (англ. UI Testing) - це вид тестування дослідження, виконуваного з метою визначення, чи зручний деякий штучний об'єкт (такий як веб- сторінка, призначений для користувача інтерфейс або пристрій) для його передбачуваного застосування. UsereXperience (UX) - відчуття, яке відчувається користувачем під час використання цифрового продукту, в той час як Userinterface - це інструмент, що дозволяє здійснювати інтеракцію «користувач - програмний ресурс».

Тестування безпеки - це стратегія тестування, яка використовується для перевірки безпеки системи, а також для аналізу ризиків, пов'язаних із забезпеченням цілісного підходу до захисту додатки, атак хакерів, вірусів, несанкціонованого доступу до конфіденційних даних.

Тестування взаємодії (InteroperabilityTesting) - це функціональне тестування, що перевіряє здатність додатку взаємодіяти з одним і більше компонентами або системами і включає в себе тестування сумісності (compatibilitytesting) і інтеграційне тестування.

1.3 Класифікація за рівнем знання про систему

За рівнем знання про систему виділяють наступні види тестування:

– чорний ящик(BlackBox);

– білий ящик (WhiteBox);

– сірий ящик (GreyBox).

BlackBox.

Summary: Ми не знаємо, як влаштована тестована система.

Тестування методом «чорного ящика» (рис.2), також відоме як тестування, засноване на специфікації або тестування поведінки - техніка тестування, заснована на роботі виключно з зовнішніми інтерфейсами тестової системи.

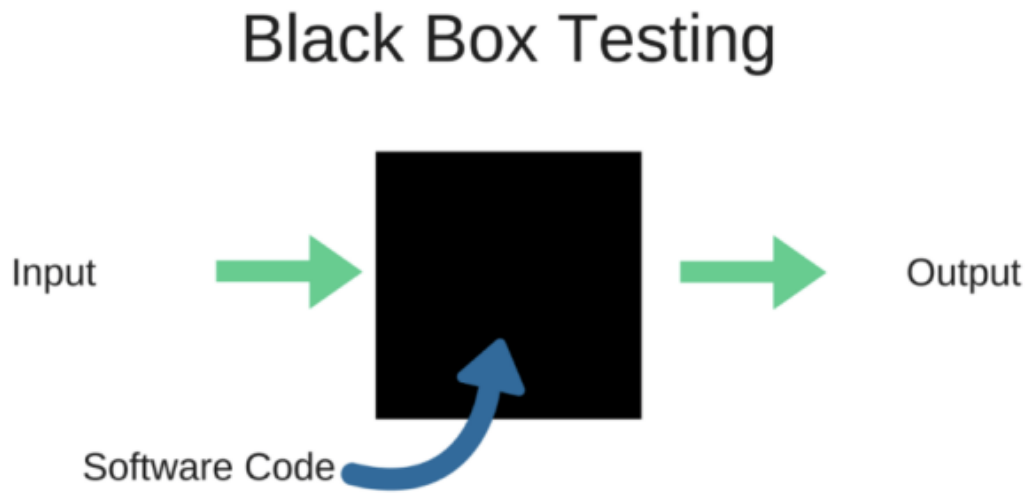

Рисунок 2 – Вид тестування BlackBox

Тестування чорного ящика - це:

– тестування, як функціональне, так і не функціональне, яке не передбачає знання внутрішнього устрою компонента або системи;

– тест-дизайн, заснований на техніці чорного ящика - процедура написання або вибору тест-кейсів на основі аналізу функціональної або не функціональної специфікації компонента або системи без знання її внутрішнього устрою.

Чому саме «чорний ящик»? Тестова програма для тестувальника - як чорний непрозорий ящик, змісту якого він не бачить.

Метою цієї техніки є пошук помилок в таких категоріях:

– неправильно реалізовані або відсутні функції;

– помилки інтерфейсу;

– помилки в структурах даних або організації доступу до зовнішніх баз даних;

– помилки поведінки або недостатня продуктивність системи.

Таким чином, ми не маємо уявлення про структуру та внутрішній устрій системи. Потрібно концентруватися на тому, що програма робить, а не на тому, як вона це робить.

Тестування чорного ящика може бути як функціональним, так і не функціональним. Функціональне тестування передбачає перевірку роботи функцій системи, а не функціональне - відповідно, загальні характеристики нашої програми.

Техніка чорного ящика може бути застосована на всіх рівнях тестування (від модульного до приймального), для яких існує специфікація. Наприклад, при здійсненні системного або інтеграційного тестування, вимоги або функціональна специфікація будуть основою для написання тест- кейсів.

Переваги:

– тестування проводиться з позиції кінцевого користувача і може допомогти виявити неточності і протиріччя в специфікації;

– тестувальнику немає необхідності знати мови програмування і заглиблюватися в особливості реалізації програми;

– тестування може проводитися фахівцями, незалежними від відділу розробки, що допомагає уникнути упередженого ставлення;

– можна починати писати тест-кейси, як тільки готова специфікація. Недоліки:

– тестується тільки дуже обмежена кількість шляхів виконання програми;

– без чіткої специфікації(а це швидше за реальність на багатьох проектах) досить важко скласти ефективні тест-кейси;

– деякі тести можуть виявитися надмірними, якщо вони вже були проведені розробником на рівні модульного тестування.

WhiteBox.

Summary: Нам відомі всі деталі реалізації програми, що тестується.

Тестування методом білого ящика (також: прозорого, відкритого, скляного ящика; засноване на коді або структурний тестування) – метод тестування програмного забезпечення, який передбачає, що внутрішня структура / пристрій / реалізація системи відомі тестувальникам (рис.3). Ми вибираємо вхідні значення, грунтуючись на знанні коду, який буде їх обробляти. Точно так само ми знаємо, яким повинен бути результат цієї обробки. Знання всіх особливостей програми, що тестується і її реалізації - обов'язкові для цієї техніки.

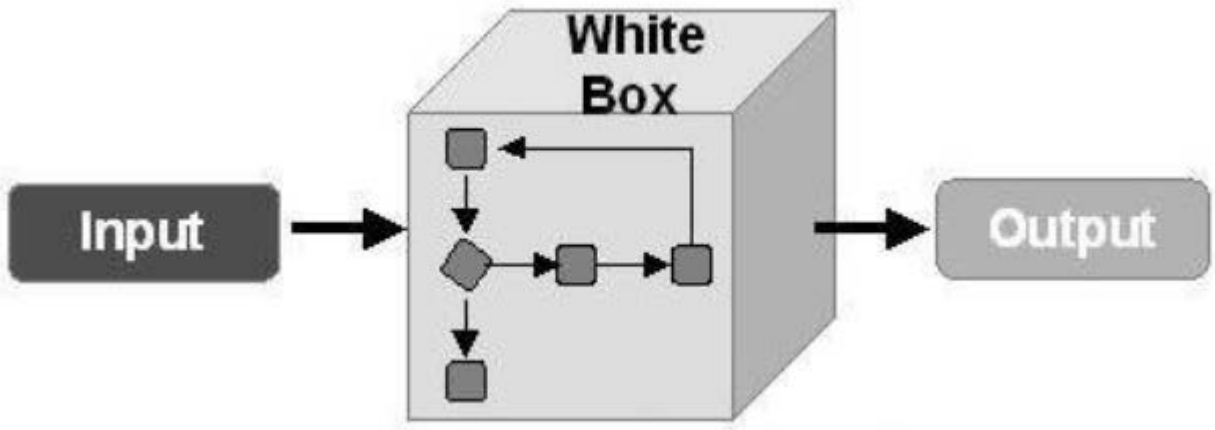

Рисунок 3 – Вид тестування WhiteBox

Тестування білого ящика - поглиблення під внутрішній устрій системи, за межі її зовнішніх інтерфейсів.

Тестування білого ящика - це:

– тестування, засноване на аналізі внутрішньої структури компонента або системи.

– тест-дизайн, заснований на техніці білого ящика - процедура написання або вибору тест-кейсів на основі аналізу внутрішнього устрою системи або компонента.

Техніка білого ящика може бути застосована на різних рівнях тестування - від модульного до системного, але головним чином застосовується саме для реалізації модульного тестування компонента його автором.

Переваги:

– тестування може проводитися на ранніх етапах: немає необхідності чекати створення призначеного для користувача інтерфейсу;

– можна провести більш ретельне тестування, з покриттям великої кількості шляхів виконання програми.

Недоліки:

– для виконання тестування білого ящика необхідна велика кількість спеціальних знань

– при використанні автоматизації тестування на цьому рівні, підтримка тестових скриптів може виявитися досить накладної, якщо програма часто змінюється.

Чому «білий ящик»? Тестова програма для тестувальника - прозорий ящик, вміст якого він прекрасно бачить.

GreyBox.

Summary: Нам відомі тільки деякі особливості реалізації тестованої системи.

Тестування методом сірого ящика - метод тестування програмного забезпечення, який передбачає, комбінацію WhiteBox і BlackBox підходів. Тобто, внутрішній устрій програми нам відомо лише частково. Передбачається, наприклад, доступ до внутрішньої структурі та алгоритмам роботи програмного забезпечення для написання максимально ефективних тест-кейсів, але саме тестування проводиться за допомогою техніки чорного ящика, тобто, з позиції користувача. Цю техніку тестування також називають методом напівпрозорого ящика: щось ми бачимо, а що - ні. Техніка сірого ящика може бути застосована на різних рівнях тестування - від модульного до системного, але головним чином застосовується на інтеграційному рівні для перевірки взаємодії різних модулів програми.

1.4 Класифікація за ступенем автоматизації

За ступенем автоматизації виділяють наступні види тестування:

– ручне;

– автоматизоване;

– напів-автоматизоване.

При ручному тестуванні (manualtesting) тестувальники вручну виконують тести, не використовуючи ніяких засобів автоматизації. Ручне тестування - самий низькорівневий і простий тип тестування, які не потребують великої кількості додаткових знань.

Проте, перед тим як автоматизувати тестування будь-якої програми, необхідно спочатку виконати серію тестів вручну. Мануальне тестування вимагає значних зусиль, але без нього ми не зможемо переконатися в тому, чи можлива автоматизація в принципі. Один з фундаментальних принципів тестування говорить: 100% автоматизація неможлива. Тому, ручне тестування - необхідність.

Автоматизоване тестування передбачає використання спеціального програмного забезпечення (крім тестового) для контролю виконання тестів і порівняння очікуваного фактичного результату роботи програми. Цей тип тестування допомагає автоматизувати часто повторювані, але необхідні для максимізації тестового покриття завдання.

Деякі завдання тестування, такі як низькорівневий регресійне тестування, можуть бути трудомісткою і вимагають багато часу якщо виконувати їх вручну. Крім того, мануальне тестування може недостатньо ефективно знаходити деякі класи помилок. У таких випадках автоматизація може допомогти заощадити час і зусилля проектної команди.

Після створення автоматизованих тестів, їх можна в будь-який момент запустити знову, причому запускаються і виконуються вони швидко і точно. Таким чином, якщо є необхідність частого повторного прогону тестів, значення автоматизації для спрощення супроводу проекту і зниження його вартості важко переоцінити. Адже навіть мінімальні патчі і зміни коду можуть стати причиною появи нових багів.

Існує кілька основних видів автоматизованого тестування:

– автоматизація тестування коду (Code-driventesting) - тестування на рівні програмних модулів, класів і бібліотек (фактично, автоматичні юніт- тести);

– автоматизація тестування графічного інтерфейсу користувача (Graphical user interface testing) - спеціальна програма (фреймворк автоматизації тестування) дозволяє генерувати користувальницькі події - натискання клавіш, кліки мишкою, і відслідковувати реакцію програми на ці дії - чи відповідає вона специфікації.

– автоматизація тестування API (Application Programming Interface) - програмного інтерфейсу програми. Тестуються інтерфейси, призначені для взаємодії, наприклад, з іншими програмами або з користувачем. Тут знову ж таки, як правило, використовуються спеціальні фреймворки.

Для складання автоматизованих тестів, QA-фахівець повинен вміти програмувати. Автоматичні тести - це повноцінні програми, просто призначені для тестування.

Коли, що і як автоматизувати і чи автоматизувати взагалі - дуже важливі питання, відповіді на які повинна дати команда розробки. Вибір правильних елементів програми для автоматизації в великій мірі буде визначати успіх автоматизації тестування в принципі. Потрібно уникати автоматизації тестування ділянок коду, які можуть часто змінюватися.

На реальних проектах часто використовується комбінація ручного і автоматизованого тестування, причому рівень автоматизації буде залежати як від типу проекту, так і від особливостей постановки виробничих процесів в компанії.

1.5 Класифікація за ступенем ізольованості компонентів

За ступенем ізольованості компонентів виділяють наступні види тестування:

- модульне тестування;
- інтеграційне тестування;
- системне тестування.

Модульне тестування (UnitTesting). Компонентне (модульне) тестування перевіряє функціональність і шукає дефекти в частинах додатка, які доступні і можуть бути протестовані по-окремо (модулі програм, об'єкти, класи, функції і т.д.).

Інтеграційний тестування (IntegrationTesting). Перевіряється взаємодія між компонентами системи після проведення компонентного тестування.

Системне тестування (SystemTesting). Основним завданням системного тестування є перевірка як функціональних, так і не функціональних вимог в системі в цілому. При цьому виявляються дефекти, такі як неправильне використання ресурсів системи, непередбачені комбінації даних користувача рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або неправильна функціональність, незручність використання і т.д.

1.6 Класифікація за часом проведення тестування

За часом проведення тестування виділяють наступні види тестування:

– альфа-тестування:

– димове тестування;

– тестування нової функції;

– підтверджуюче тестування;

– регресійне тестування;

– приймальне тестування;

– бета-тестування.

Альфа-тестування (alphatesting) - це вид приймального тестування, яке зазвичай проводиться на пізній стадії розробки продукту і включає імітацію реального використання продукту штатними розробниками або командою тестувальників. Зазвичай альфа тестування полягає в систематичній перевірці всіх функцій програми з використанням технік тестування «білого ящика» і «чорного ящика».

Переваги альфа-тестування:

– забезпечує краще уявлення про надійність програмного забезпечення на ранній стадії;

– Допомагає моделювати поведінку користувача і навколишнє середовище в режимі реального часу;

– виявляє багато серйозних помилок;

– дає можливість раннього виявлення помилокщодо дизайну і функціональності.

Недоліки альфа-тестування:

– функціональність не може бути перевірена на всю глибину, оскільки програмне забезпечення все ще знаходиться на стадії розробки;

– іноді розробники і тестувальники незадоволені результатами альфа- тестування.

Бета-тестування (betatesting) - інтенсивне використання майже готової версії продукту з метою виявлення максимального числа помилок в його роботі для їх подальшого усунення перед остаточним виходом (релізом) продукту на ринок, до масового споживача. Бета-тестування є реально працюючу версію програми з повним функціоналом. І завдання бета-тестів - оцінити можливості і стабільність роботи програми з точки зору її майбутніх користувачів.

На відміну від альфа-тестування, проведеного силами штатних розробників або тестувальників, бета-тестування передбачає залучення добровольців з числа звичайних майбутніх користувачів продукту, яким доступна згадана попередня версія продукту (так звана бета-версія).

Такими добровольцями (їх називають бета-тестерами) часто рухає цікавість до нового продукту - цікавість, заради задоволення якого вони цілком згодні миритися з можливістю випробувати наслідки ще не знайдених (а тому і не виправлених) помилок. Крім цікавості, мотивація може бути обумовлена бажанням вплинути на процес розробки і в підсумку отримувати більш задовольняє їхні потреби продукт і багатьом іншим. Дуже добре, якщо це люди, які вже мають досвід роботи з програмами такого типу, а ще краще з попередньою версією цієї ж програми. Зазвичай у компаній вже є певне коло осіб, з якими вони постійно співпрацюють.

Бета-тестування може бути:

– закритим: програма тестується в невеликій групі користувачів за запрошеннями;

– відкритим: цей варіант дозволяє протестувати додаток в більшій групі і отримати великий обсяг зворотного зв'язку; будь-який користувач зможе приєднатися до відкритого бета-тестування і відправити особистий відгук.

Переваги бета-тестування:

– знижує ризик виходу продукту з ладу за допомогою валідації клієнта;

– бета-тестування дозволяє компанії тестувати інфраструктуру після запуску;

– підвищує якість продукції завдяки зворотного зв'язку з клієнтами;

– є економічним методом збору даних в порівнянні з аналогічними методами;

– створює доброзичливість з клієнтами і підвищує задоволеність клієнтів.

Недоліки бета-тестування:

– управління тестуванням – проблема; у порівнянні з іншими типами тестування, які зазвичай виконуються всередині компанії в контрольованому середовищі, бета-тестування виконується в реальному світі, де у компанії рідко є контроль;

– пошук правильних користувачів бета-версії і підтримання їх участі може викликати труднощі.

Фази тестування:

1. Пре-альфа: програмне забезпечення є прототипом. Інтерфейс завершений, але не всі функції завершені. На цьому етапі програмне забезпечення не публікується.

2. Альфа: розробка програмного забезпечення підійшла до кінця і програмне забезпечення внутрішньо перевірено на наявність помилок і проблем.

3. Бета: програмне забезпечення стабільно і випущено обмеженому числу користувачів. Мета полягає в тому, щоб отримати відгуки клієнтів про продукт і внести відповідні зміни в програмне забезпечення.

4. Реліз-кандидат: на основі відгуків бета-тесту вносяться зміни в програмне забезпечення і перевіряються виправлення помилок. На цьому етапі функціональність радикально не змінюється. Реліз-кандидат також виставляється громадськості.

5. Випуск (реліз): все працює, програмне забезпечення випущено для загального користування.

Димове (Smoke) тестування розглядається як короткий цикл тестів, що виконується для підтвердження того, що після складання коду (нового або відредагованого) встановлюється додаток, стартує і виконує основні функції.

Тестування збірки або BuildVerificationTest - тестування спрямоване на визначення відповідності, випущеної версії, критеріям якості для початку тестування. За своїми цілями є аналогом димових Тестування, спрямованого на приймання нової версії в подальше тестування або експлуатацію. Вглиб воно може проникати далі, в залежності від вимог до якості випущеної версії.

Регресійне тестування - це вид тестування спрямований на перевірку змін, зроблених в додатку або навколишньому середовищу (лагодження дефекту, злиття коду, міграція на іншу операційну систему, базу даних, веб сервер або сервер додатки), для підтвердження того факту, що існуюча раніше функціональність працює як і раніше. Регресійний можуть бути як функціональні, так і не функціональні тести.

Повторне тестування - тестування, під час якого виконуються тестові сценарії, які виявили помилки під час останнього запуску, для підтвердження успішності виправлення цих помилок. У чому різниця між regressiontesting і re-testing? Re-testing - перевіряється виправлення багів. Regressiontesting - перевіряється те, що виправлення багів не вплинуло на інші модулі системи і не викликало нових багів.

Приймальне тестування (AcceptanceTesting). Формальний процес тестування, який перевіряє відповідність системи вимогам і проводиться з метою:

– визначення чи задовольняє система приймальним критеріям;

– винесення рішення замовником або іншою уповноваженою особою приймається додаток чи ні.

1.7 Класифікація за ознакою позитивності сценарію

За ознакою позитивності сценарію виділяють наступні види тестування:

– позитивнее тестування;

– негативнее тестування.

Позитивне тестування - це тестування з застосуванням сценаріїв, які відповідають нормальному (штатним, очікуваному) поведінки системи. З його допомогою ми можемо визначити, що система робить те, для чого і була створена.

Негативним називають тестування, в рамках якого застосовуються сценарії, що відповідає позаштатному поведінки тестованої системи. Це можуть бути наприклад, виняткові ситуації або невірні дані.

Негативне тестування направлено на перевірку стійкості системи до різних впливів, валідації невірних даних, обробки виняткових ситуацій.

Сценарії позитивного тестування, в свою чергу, спрямовані на перевірку роботи системи з тими типами даних, для яких вона розроблялася.

Створення позитивних сценаріїв (тест-кейсів), як правило, передує створенню негативних тест-кейсів. Спочатку ми перевіряємо роботу системи, коли наш умовний користувач працює з системою "правильно", а потім приступаємо до перевірки відгуку системи на користувача, що допускає різні помилки (введення невірних даних наприклад). І система повинна бути готова відповісти на невірний запит. Це і є мета негативного тестування.

1.8 Класифікація за ступенем підготовленості до тестування

За ступенем підготовленості до тестування виділяють наступні види тестування:

– тестування на підставі документації;

– інтуітивне тестування.

Інтуїтивне тестування (ad-hoctesting) - вид тестування, який виконується без підготовки до тестів, без визначення очікуваних результатів, проектування тестових сценаріїв. Це неформальне, імпровізаційне тестування. Воно не вимагає ніякої документації, планування, процесів яких слід дотримуватися у виконанні. Також на даний вид тестування не пишуться тест-кейси, що в свою чергу може викликати певні труднощі в спробах відтворити дефект в системі. Такий вид найчастіше може дати відразу більше результату ніж тестування за заздалегідь визначеними сценаріями. Це обумовлено тим, що тестувальник на перших кроках приступає до тестування основного функціоналу і виконує нестандартні перевірки, точніше деякі з його перевірок будуть нестандартними.

Часто інтуїтивне тестування плутають з дослідним. Якщо говорити про ad-hoctesting і дослідне тестування, то ad -hoctesting - це більш інтуїтивне і безладне тестування, коли тестувальник просто йде і перевіряє, що йому хочеться. У нього немає певної мети, структури тестів в голові, якоїсь системи. У свою чергу дослідне тестування більш структуроване. Зазвичай тестувальник знає, що йому потрібно перевірити, у нього в голові є мета і якась система проведення тестів. Хоч тести в цьому випадку не обов'язково повинні бути оформлені у вигляді тест кейсів.

Ad-hoc тестування виконується, коли мало часу на точне і послідовне тестування. При цьому тестувальник покладається на своє загальне уявлення про програму і здоровий глузд. Тестування ad-hoc має сенс тільки в разі якщо тестувальник володіє загальною інформацією про продукт. Якщо людина зовсім не знатиме продукт, то витратить час на його вивчення, особливо якщо проект дуже складний і великий. Тому потрібно гарне уявлення про цілі проекту, його призначенням і основним функціям і можливостям. А далі вже можна приступати до ad-hoctesting.

Види інтуїтивного тестування:

– buddytesting (спільне тестування) - коли 2 людини, як правило розробник + тестувальник, працюють паралельно і знаходять дефекти в одному і тому ж модулі. Такий вид тестування допомагає тестувальника виконувати необхідні перевірки, а програмісту - фіксувати баги на ранніх етапах.

– pairtesting (парне тестування) - коли 2 тестувальника перевіряють один модуль і допомагають один одному. Наприклад один може шукати дефекти, а другий їх

документувати. Таким чином у одного тестера буде функція, того, хто знаходить, в іншого – того, хто описує.

– monkeytesting - довільне тестування програми з метою її зламати. Основні переваги ad-hoctesting:

– немає необхідності витрачати час для підготовки документації;

– найважливіші дефекти виявляються на ранніх етапах;

– часто застосовується коли беруть нового співробітника: методом ad-hoc людина схоплює за 3 дні те, що по тест кейсам розбирала б тиждень - це називається форсоване навчання нових співробітників;

– можливість знайти хитрі дефекти, які не можна було б знайти при використанні стандартних сценаріїв перевірок.