Міністерство освіти і науки України
Луцький національний технічний університет

ТЕСТУВАННЯ ТА ДІАГНОСТИКА ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ

Методичні вказівки
до виконання самостійної роботи
для здобувачів першого (бакалаврського) рівня вищої освіти
освітньої програми «Комп’ютерна інженерія»
галузь знань 12 Інформаційні технології
спеціальності 123 Комп'ютерна інженерія
денної та заочної форм навчання

Луцьк 2025

УДК 004.415.53 (07)
Т 36

Рекомендовано до видання вченою радою факультету КІТ ЛНТУ,
протокол № _____ від « ___ » _____ 20 ___ року.

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ

Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри комп'ютерної інженерії та безпеки ЛНТУ, протокол № _____ від « _____ » _____ 20 ___ року

Завідувач кафедри КІБ _____ Тарас ТЕРЛЕЦЬКИЙ

Укладачі: _____ Оксана МІСКЕВИЧ, старший викладач кафедри комп'ютерної інженерії та безпеки ЛНТУ
_____ Сергій ГРИНЮК, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

Рецензент: _____ Катерина МЕЛЬНИК, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

Відповідальний за випуск: _____ Тарас ТЕРЛЕЦЬКИЙ, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

Т-36 **Тестування та діагностика програмно-апаратних засобів.** Методичні вказівки до виконання самостійної роботи для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. О. І. Міскевич, С. В. Гринюк. Луцьк : ЛНТУ, 2025. 50 с.

Методичне видання складене відповідно до діючої програми курсу «**Тестування та діагностика програмно-апаратних засобів**» з метою використання студентами спеціальності 123 Комп'ютерна інженерія при вивченні даної дисципліни.

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Навчальна дисципліна «Тестування та діагностика програмно-апаратних засобів» дає студентам розширення та поглиблення теоретичних знань з питань функціонування комп'ютерних засобів; засвоєння основних понять та визначень в галузі тестування; ознайомлення студентів з технічними характеристиками основних компонентів обчислювальних систем; вивчення програмного забезпечення призначеного для тестування апаратних та програмних ресурсів; огляд різновидів тестування та критеріїв вибору тестів; отримання навиків застосування сучасних інформаційних технологій для аналізу та тестування; засвоєння студентами прийомів і навиків розв'язання завдань, які виникають в практичній роботі.

Основними завданнями курсу є: ознайомити студентів з технічними характеристиками та основними компонентами обчислювальних систем; вивчити програмне забезпечення, яке призначене для тестування апаратних та програмних ресурсів; отримати навики проведення комплексних перевірок роботоздатності обчислювальних систем; поглибити знання студентів про прийоми і навички розв'язання конкретних завдань, які виникають в практичній роботі; ознайомити студентів з сучасною літературою та методами отримання інформації за темою курсу.

Результати навчання є: вміти застосовувати знання технічних характеристик, конструктивних особливостей, призначення і правил експлуатації програмно-технічних засобів комп'ютерних систем та мереж для вирішення технічних задач спеціальності; вміти здійснювати пошук інформації в різних джерелах для розв'язання задач комп'ютерної інженерії; вміти ідентифікувати, класифікувати та описувати роботу комп'ютерних систем та їх компонентів; вирішувати проблеми у галузі комп'ютерних та інформаційних технологій.

Вивчення курсу повинно супроводжуватися виконанням практичних та самостійних робіт.

Самостійна робота студентів займає до 50 % загального обсягу навчального часу студента, відведеного для вивчення дисципліни, а також включає підготовку до заліку з дисципліни.

Самостійна робота студентів ставить за мету:

- розвиток творчих здібностей та активізацію розумової діяльності студентів;
- формування у студентів потреби самостійного поповнення знань;
- здобуття студентом додаткових вирішень питань з периферійними пристроями.

Завданням самостійної роботи студентів є наступне:

- навчити студентів самостійно працювати над літературою;
- творчо сприймати навчальний матеріал і його осмислювати;
- набути навички щоденної самостійної роботи в одержанні та узагальненні знань, вмінь.

Зміст самостійної роботи студентів з дисципліни визначається навчальною програмою дисципліни та робочою навчальною програмою вивчення дисципліни.

На самостійну роботу винесено:

- підготовка до лекцій;
- частина теоретичного матеріалу, менш складного за змістом;
- підготовка до практичних робіт;

Викладач може видавати студентам різні види завдань самостійної роботи:

- опрацювання інформації, отриманої безпосередньо на обов’язкових навчальних заняттях;
- робота з відповідними підручниками та особистим конспектом лекцій;
- самостійне вивчення окремих тем або питань;
- робота з довідковою літературою;
- творчі завдання (доповіді, проекти, огляди, тощо);
- виконання підготовчої роботи до практичних робіт.

Успішне виконання завдання самостійної роботи можливе за умови наявності у студентів певних навичок:

– вміння працювати з додатковими джерелами;

– проводити аналіз навчального матеріалу;

– навичок роботи з периферійними пристроями та їх програмним забезпеченням.

Виконання завдань з самостійної роботи контролюється після закінчення логічно завершеної частини лекцій та практичних робіт з дисципліни, і її результати враховуються при виставленні підсумкової оцінки.

Навчальний матеріал навчальної дисципліни, передбачений робочим навчальним планом для засвоєння студентами у процесі самостійної роботи, виноситься на підсумковий семестровий контроль – залік.

1. ОСНОВНІ ТЕРМІНИ ТЕСТУВАННЯ

Що таке тестування?

Перевірка відповідності між реальною та очікуваною поведінкою програми на кінцевому наборі тестів, обраному певним чином.

Цілі тестування.

Виявлення дефектів

Підвищення впевненості в рівні якості

Надання інформації для прийняття рішень

Запобігання дефектів

QA, QC, Testing.

Забезпечення якості (quality assurance) – частина менеджменту якості, спрямована на створення впевненості, що вимоги до якості будуть виконані.

Управління якістю (quality control) – частина менеджменту якості, спрямована на виконання вимог до якості.

Тестування (testing) – Процес, що містить в собі всі активності життєвого циклу, як динамічні, так і статичні, що стосуються планування, підготовки та оцінки програмного продукту і пов’язаних з цим результатів робіт з метою визначити, що вони відповідають описаним вимогам, показати, що вони підходять для заявлених цілей та визначення дефектів.

Менеджмент якості (quality management) – скоординована діяльність з керівництва та управління організацією, стосовно до якості.

Примітка – Керівництво та управління до якості зазвичай включає в себе розробку політики у сфері якості та цілей у сфері якості, планування якості, управління якістю, забезпечення якості та поліпшення якості.

Основні види тестування.

Функціональне тестування

Нефункціональне тестування

Тестування структури / архітектури програмного забезпечення (структурне тестування)

Тестування змін: підтверджувальне і регресійне тестування

Класифікація видів

Тестування: по знанню системи, по об’єкту тестування, по суб’єкту тестування, за часом проведення тестування, за критерієм “позитивності” сценаріїв, за ступенем ізольованості тестованих компонентів, за ступенем автоматизації тестування і за ступенем підготовки до тестування.

За якими атрибутам характеристик якості ми тестуємо?

У ПЗ є 6 характеристик якості:

Функціональні можливості (Functionality),

Надійність (Reliability),

Практичність (Usability),

Ефективність (Efficiencies),

Ремонтопридатність (Maintainability)

Мобільність (Portability).

У кожної характеристики є атрибути.

Відмінності Load and Performance testing.

Тестування навантаження (load testing) – вид тестування продуктивності, що проводиться з метою оцінити поведінку компонента або системи під збільшені навантаження (число одночасно працюючих користувачів і / або число транзакцій) для визначення максимального рівня навантаження компонента або системи.)

Тестування продуктивності (performance testing) – процес тестування з метою визначити продуктивність програмного продукту.

Load vs Performance Testing

Нам здалося, що Load і Performance Testing переслідують все ж одну і ту ж мету: перевірка продуктивності (часу відгуку) на різних навантаженнях. Власне тому ми й не стали розділяти їх. У той же час хто то може розділити. Головне все таки розуміти цілі того чи іншого виду тестування і постаратися їх досягти.

Нефункціональні види тестування.

Нефункціональне тестування – включає тестування навантаження, тестування продуктивності, стрес-тестування, тестування зручності використання, тестування

відновлення, тестування надійності і тестування переносимості. Це тестування того, “як” система працює.

Тестування структури / архітектури програмного забезпечення (структурне тестування)

Тестування змін: підтверджувальне і регресійне тестування.

Примітка: Регресійне тестування може виконуватися на всіх рівнях тестування і включає функціональне, нефункціональне і структурне тестування.

Тестування установки.

Тестування установки (installation testing) – Перевіряє працездатність методів установки, налаштування і видалення програми на всіх підтримуваних платформах.

Тестування документації, основні принципи.

Перевірка документації (documentation testing) – Перевіряє точність всієї документації користувача.

Системне тестування вимагає тестування документації. Для цього за допомогою документації перевіряють спосіб подання описаних раніше системних тестів. Наприклад, якщо запланований якийсь навантажувальний тест, то слід використовувати документацію для підбору конкретних варіантів. Крім того, шляхом безпосередньої інспекції необхідно перевірити точність і ясність викладу користувальницької документації. Будь-які процедури, які фігурують у документації, повинні кодуватися і пропускатися через програму.

Читаючи й аналізуючи документацію, слід передусім приділити увагу її точності, повноті, ясності, простоті використання і тому, наскільки вона відповідає духу програмного продукту.

Основні принципи Usability testing.

Оптимально, коли зручність використання тестують кінцеві користувачі, а не тестувальники. Завдання тестувальника може полягати в підготовці набору практичних значень, пов’язаних з реальною діяльністю, повторюваних тестових завдань, які повинен буде виконати кожен користувач. Проектуйте ці тестові сценарії так, щоб у процесі їх виконання користувач зіткнувся з усіма аспектами

програмного забезпечення, знайомлячись з ними в якомусь певному або випадковому порядку.

Важливу роль відіграє збір та аналіз докладних, точних даних. Реальним початком процесу збору даних є розробка детальних користувальницьких інструкцій і списку завдань. Закінчується ж цей процес зведенням в воєдино результатів спостережень, зроблених користувачами, або відповідей користувачів на анкети після проведення тестів.

Нарешті, результати тестування повинні бути правильно інтерпретовані, і на основі отриманих висновків розробники мають внести в ПЗ відповідні зміни.

Артефакти тестування.

Основними поняттями RUP (Rational Unified Process) є артефакт (artifact) і прецедент (precedent). Артефакти – це деякі продукти проекту, породжувані або використовувані в ньому при роботі над остаточним продуктом. Прецеденти – це послідовності дій, виконуваних системою для отримання спостережуваного результату.

2 варіант відповіді на питання:

У відповідність з процесами або методологіями розробки ПЗ, під час проведення тестування створюється і використовується певна кількість тестових артефактів (документи, моделі і т.д.). Найбільш поширеними тестовими артефактами є:

План тестування (Test Plan) – це документ описує весь обсяг робіт з тестування, починаючи з опису об’єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Набір тест кейсів і наборів (Test Case & Test suite) – це послідовність дій, за якою можна перевірити чи відповідає тестована функція встановленим вимогам.

Дефекти / Баг Репорт (Bug Reports / Defects) – це документи, що описують ситуацію або послідовність дій, яка призвела до некоректної роботи об’єкта тестування, із зазначенням причин і очікуваного результату.

Тест план і check-list.

План тестування (Test Plan) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об’єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Чек-ліст – це документ, що описує що має бути протестовано. При цьому чек-ліст може бути абсолютно різного рівня деталізації. На скільки детальним буде чек-лист залежить від вимог до звітності, рівня знання продукту співробітниками і складності продукту.

Чек-ліст потрібен для:

Не забути необхідні тести

Для поділу завдань за рівнем кваліфікації

Для збереження звітності і результатів тестування

Що має бути в Чек-листі:

Список перевірок (з необхідної ступенем деталізації)

Статус перевірок:

збірка, на якій проводилося тестування

тестове оточення (якщо є) тестувальник

Результат перевірки

Traceability matrix (намалюйте)

Матриця відповідності вимог – це двовимірна таблиця, яка містить відповідність функціональних вимог (functional requirements) продукту і підготовлених тестових сценаріїв (test cases). У заголовках колонок таблиці розташовані вимоги, а в заголовках рядків – тестові сценарії. На перетині – позначка, що означає, що вимогу поточної колонки покрито тестовим сценарієм поточного рядка.

Матриця відповідності вимог використовується QA-інженерами для валідації покриття продукту тестами. МВВ є невід’ємною частиною тест-плану (рисунок 2.1).

Sample traceability matrix

Requirement Identifiers	Reqs Tested	REQ1 UC 1.1	REQ1 UC 1.2	REQ1 UC 1.3	REQ1 UC 2.1	REQ1 UC 2.2	REQ1 UC 2.3.1	REQ1 UC 2.3.2	REQ1 UC 2.3.3	REQ1 UC 2.4	REQ1 UC 3.1	REQ1 UC 3.2	REQ1 TECH 1.1	REQ1 TECH 1.2	REQ1 TECH 1.3
Test Cases	321	3	2	3	1	1	1	1	1	1	2	3	1	1	1
Tested Implicitly	77														
TC1.1.1	1	x													
TC1.1.2	2		x	x											
TC1.1.3	2	x											x		
TC1.1.4	1			x											
TC1.1.5	2	x												x	
TC1.1.6	1		x												
TC1.1.7	1			x											
TC1.2.1	2				x		x								
TC1.2.2	2					x		x							
TC1.2.3	2								x	x					
TC1.3.1	1										x				
TC1.3.2	1										x				
TC1.3.3	1											x			
TC1.3.4	1											x			
TC1.3.5	1											x			
etc															
TC5.6.2	1														x

Рисунок 1.1 – Матриця відповідності вимог

Основні поля test case.

Тестовий сценарій (test case) – Набір вхідних значень, передумов виконання, очікуваних результатів і фактичних результатів, розроблений для певної мети або тестової умови, таких як виконання певного шляху програми або ж для перевірки відповідності певній вимозі.

Основні поля тестового сценарію: Набір вхідних значень, передумов виконання, очікуваних результатів і фактичних результатів.

Життєвий цикл бага (помилок)

Блок-схема, що показує основні статуси і можливі переходи від статусу до статусу в процесі його існування (рисунок 2.2)

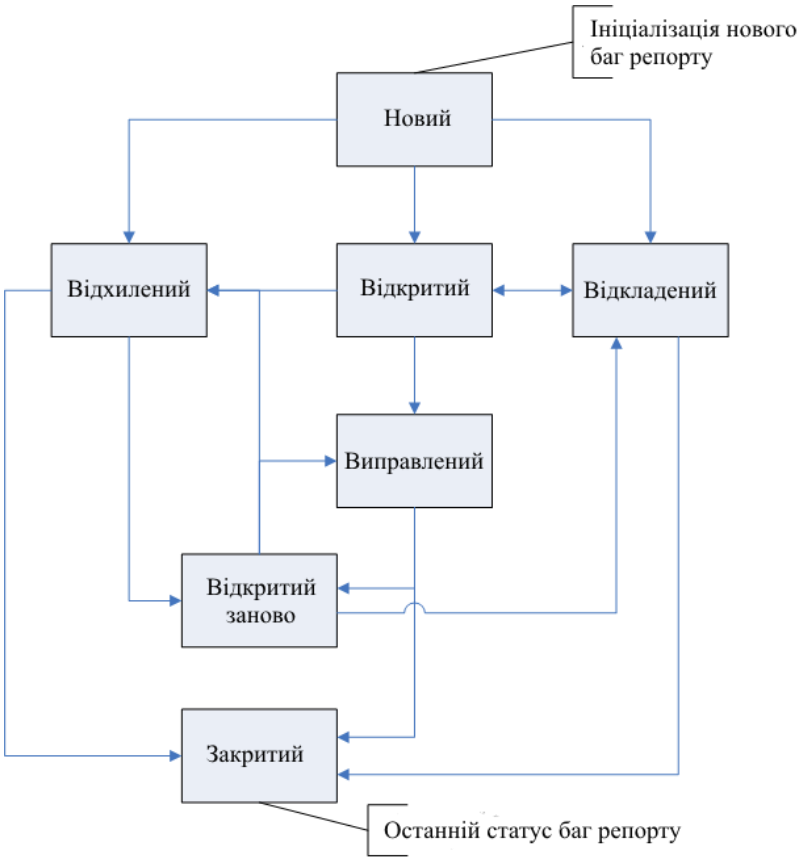

Рисунок 1.2 – Життєвий цикл бага

Опис даної схеми.

Припустимо ви знайшли баг і зареєстрували його в баг трекінг системі. Відповідно до нашої блок-схемі він отримає статус “Новий”. Тестувальник, відповідальний за валідацію нових баг репортів, або координатор проекту (в залежності від розподілу ролей у команді) може перевести його в один з наступних статусів:

“Відхилений”, якщо даний баг невалідний або повторний, або ж його просто не змогли відтворити;

“Відкладений”, якщо даний баг не потрібно виправляти в даній ітерації· “Відкритий”, якщо виправити бага необхідно.

Розглянемо тепер по порядку кожен з варіантів.

1. Відхилений. У цьому випадку ви можете або посперечатися про долю вашого багрепорта, змінивши статус на “Відкритий заново” або закрити його – статус “Закритий”.

2. Відкладений. Баг репорт в статусі “Відкладений” можна перевести в статус “Відкритий”, коли буде потрібно виправлення або в статус “Закритий”, якщо вже не буде потрібно.

3. Відкритий. Саме в такому стані розробник отримує баг репорт для виправлення. Він може відхилити (подальші дії дивіться в пункті 1) або виправити баг. Баг репорт в статусі “Виправлений” перекладається на тестувальника для перевірки. У разі якщо проблема все ще відтворюється, виставляється статус “Відкритий заново” і баг репорт направляється назад на доопрацювання до розробника. Якщо ж виправлення було успішним, то баг репорт переводиться в статус “Закритий”.

Необхідно зазначити, що дана схема сильно спрощена. Для більшої наочності і, можливо, зручності роботи на проекті, ви можете додати додаткові статуси і переходи, тим більше, що сучасні баг трекінгові системи дозволяють це робити. Правда майте на увазі, що заплутані схеми переходів і зайві статуси можуть значно ускладнити життя.

Примітка 1: в деяких системах баг трекінгу створений баг репорт відразу отримує статус “Відкритий” без додаткової валідації.

Примітка 2: багато баг трекінгових систем дозволяють перевідкривати закриті баги, проте особисто я проти такої практики, тому й не описував подібний перехід у вище поданому життєвому циклі.

Примітка 3: Розглянутий вище життєвий цикл заснований на тому, що в команді є хтось, відповідальний за призначення баг репортів. У випадку, якщо такої ролі на проекті немає, то баги призначаються розробниками самостійно, і тоді для уникнення плутаниці, є сенс ввести ще один проміжний статус “У розробці” (In progress), що показує, що даний баг репорт вже призначений і знаходиться на стадії виправлення.

Bug report, Основні поля bug report.

Звіт про баг (bug report): Див. Звіт про дефект.

Звіт про дефект (defect report): Документ, що містить звіт про будь-які нестачі в компоненті або системі, який може привести компонент або систему до неможливості виконати потрібну опцію.

Різні багтрекінгові системи, пропонують нам різні поля для заповнення та різні структури опису дефектів. Наведена нижче приклад – те, що рекомендують використовувати у вигляді шаблону баг репорту.

Шапка

Короткий опис (Summary) – Короткий опис проблеми, явно вказує на причину і тип помилкової ситуації.

Проект (Project) – Назва тестованого проекту.

Компонент додатку (Component) Назва частини або функції продукту.

Номер версії (Version) – Версія на якій була знайдена помилка.

Серйозність (Severity) – Найбільш поширена п’ятирівнева система градації серйозності дефекту: S1 Блокуючий (Blocker), S2 Критичний (Critical), S3 Значний (Major), S4 Незначний (Minor), S5 Тривіальний (Trivial).

Пріоритет (Priority) – Пріоритет дефекту: P1 Високий (High), P2 Середній (Medium), P3 Низький (Low).

Статус (Status) – Статус бага. Залежить від використовуваної процедури та життєвого циклу бага (bug workflow and life cycle).

Автор (Author) – Автор баг репорту.

Призначений на (Assigned To) – Ім’я співробітника, призначеного на вирішення проблеми.

Оточення

ОС / Сервіс Пак і т.д. / Браузера + версія / … Інформація про оточення, на якому був знайдений баг: операційна система, сервіс пак, для WEB тестування – ім’я і версія браузера і т.д.

Опис

Кроки відтворення (Steps to Reproduce) – Кроки, за якими можна легко відтворити ситуацію, що призвела до помилки.

Фактичний Результат (Result) – Результат, отриманий після проходження кроків до відтворення

Очікуваний результат (Expected Result) – Очікуваний правильний результат Доповнення

Прикріплений файл (Attachment) – Файл з логами, скріншот або будь-який інший документ, який може допомогти прояснити причину помилки або вказати на спосіб вирішення проблеми

Пріоритет і Серйозність

Серйозність (Severity) – це атрибут, що характеризує вплив дефекту на працездатність програми.

Пріоритет (Priority) – це атрибут, який вказує на черговість виконання завдання або усунення дефекту. Можна сказати, що це інструмент менеджера з планування робіт. Чим вище пріоритет, тим швидше потрібно виправити дефект.

Градація серйозних дефектів (Severity)

S1 Блокуюча (Blocker)

Блокуюча помилка, що приводить додаток в неробочий стан, в результаті якого подальша робота з тестованої системою або її ключовими функціями стає неможлива. Рішення проблеми необхідно для подальшого функціонування системи.
S2 Критична (Critical)

Критична помилка, неправильно працює ключова бізнес логіка, діра в системі безпеки, проблема, яка призвела до тимчасового падіння сервера або приводить в неробочий стан деяку частину системи, без можливості вирішення проблеми, використовуючи інші вхідні точки. Рішення проблеми необхідно для подальшої роботи з ключовими функціями тестируемой системою.

S3 Значна (Major)

Значна помилка, частина основний бізнес логіки працює некоректно. Помилка не критична або є можливість для роботи з тестованої функцією, використовуючи інші вхідні точки.

S4 Незначна (Minor)

Незначна помилка, що не порушує бізнес логіку частини програми, що тестується, очевидна проблема для користувача інтерфейсу.

S5 Тривіальна (Trivial)

Тривіальна помилка, яка не стосується бізнес логіки додатка, погано відтворена проблема, малопомітна за допомогою користувальницького інтерфейсу, проблема сторонніх бібліотек або сервісів, проблема, не надає ніякого впливу на загальну якість продукту.

Градація Пріоритету дефекту (Priority)

P1 Високий (High)

Помилка повинна бути виправлена якомога швидше, так як її наявність є критичною для проекту.

P2 Середній (Medium)

Помилка повинна бути виправлена, її наявність не є критичною, але вимагає обов'язкового рішення.

P3 Низький (Low)

Помилка повинна бути виправлена, її наявність не є критичною, і не вимагає термінового вирішення.

Порядок виправлення помилок з їх пріоритетами:

High -> Medium -> Low

Назвіть баг з вищим пріоритетом і низькою серйозністю і навпаки

Наприклад, якщо на головній сторінці замість "Зробити Яндекс стартовою сторінкою" був би напис з орфографічною помилкою: "Зробити Я**п**дек стартовою сторінкою" (вищий пріоритет – низька серйозність)

Навпаки – помилка на рівні архітектури, виправлення якої на поточний момент дорожче, ніж існування з нею (низький пріоритет – висока серйозність)

Use case відміну від test case

Сценарій використання системи (use case): Послідовність операцій у взаємодії актора і компонента або системи зі значним результатом, при якій актором може бути як користувач, так і все, що може обмінюватися інформацією з системою.

Тестовий сценарій (test case): Набір вхідних значень, передумов виконання, очікуваних результатів і пострезультатів виконання, розроблений для певної мети або тестової умови, такої як виконання певного шляху програми або ж для перевірки відповідності певним вимогам.

Верифікація та валідація.

Верифікація (verification): Підтвердження наданням об'єктивних доказів того, що встановлені вимоги були виконані.

Примітки

1. Термін "верифікований" використовують для позначення відповідного статусу.

2. Діяльність з підтвердження вимоги може включати в себе:

Здійснення альтернативних розрахунків;

Порівняння специфікації на новий проект з аналогічною документацією на проект;

Проведення випробувань і демонстрацій;

Аналіз документів до їх випуску.

Валідація (validation): Підтвердження наданням об'єктивних доказів того, що вимоги, призначені для конкретного використання або застосування, виконані.

Примітки

1. Термін "валідація" використовують для позначення відповідного статусу.

2. Умови застосування можуть бути реальними або змодельований.

Об'єктивне свідчення (objective evidence): Дані, що підтверджують наявність або істинність чого-небудь.

Вимога (requirement): Потреба або очікування, яке встановлено, зазвичай передбачається чи є обов'язковим.

Специфікація (specification): Документ, що встановлює вимоги.

Випробування (test): Визначення однієї або декількох характеристик відповідно до встановленої процедури.

Граничні умови, класи еквівалентності.

Граничне значення (boundary value): Вхідні значення або вихідні дані, яке перебуває на межі еквівалентної області або на найменшій відстані від обох сторін грані, наприклад, мінімальне або максимальне значення області.

Еквівалентна область (equivalence partition): Частина області вхідних або вихідних даних, для якої поведінка компонента або системи, ґрунтуючись на специфікації, вважається однаковою.

Різниця між тестуванням desktop and web.

Одні з думок:

1.Веб-тестування зачіпає ряд питань, які зазвичай не виникають при традиційному тестуванні десктоп додатків. Прикладами спеціалізованого веб-тестування можуть служити такі речі як: тестування сумісності браузера, тестування Web доступності, перевірка на «мертві» посилання, а також відстеження повідомлень між клієнтом і сервером.

2. Проблеми продуктивності і безпеки у веб-додатку будуть іншими, ніж в десктоп додатках. Існують відмінності в клієнтській базі, в тому, як розгорнуто додаток, і як часто воно використовується. А також відрізняються сервісна модель та обслуговування веб-додатків.

3. Дурниці це все, висмоктані з пальця, с урахуванням розвитку інтернет технологій ніякої різниці немає.

Підтримка користувачів (Support).

Причин може бути декілька:

«Тестувальник – всезнайка» – він знає більше за всіх про сам продукт і про його вразливості.

«Тестувальник – винтик-шпунтик» – оперативно зреагує на озвучену проблему.

«Тестувальник – шукач» – він вживе заходів, щоб виправити цю проблему в найкоротші терміни, або знайде того, хто це зробить.

«Тестувальник – провідник» – може розібратися і пояснити користувачеві на зрозумілою для нього мовою, як виправити проблему, навіть якщо вона не на боці продукту.

Тестувальник = техпідтримка – ідеал для невеликих і середніх компаній. У великих виникнуть проблеми, там краще набирати вже спеціально навчених людей. Ідеальної схеми не існує, і як би ми не намагалися охопити всі проблеми, кожному необхідно зібрати свій конструктор з підходів та методик. Адже кожна компанія і її продукти унікальні, і навряд чи хтось знає, що підходить саме вам.

Відмінність Sanity від Smoke.

Тест працездатності (sanity test): Див. Димове тестування.

Димове тестування (smoke test): Вибірка із загального числа запланованих тестових сценаріїв, що покриває основну функціональність компонента або системи. Проводиться з метою упевнитися, що базові функції програми в цілому працюють коректно, без заглиблення в деталі. Щоденна збірка і димове тестування є передовими практичними методами.

Вхідний тест (intake test): Спеціальний тип димового тестування для прийняття рішення, чи готовий компонент або система готові для подальшого детального тестування. Зазвичай починається на початку фази тестування.

Інша думка:

Відмінність санітарного тестування від димового (Sanity vs Smoke testing)

У деяких джерелах помилково вважають, що санітарне та димове тестування – це одне і теж. Ми ж вважаємо, що ці види тестування мають “вектора руху”, напрямки в різні боки. На відміну від димового (Smoke testing), санітарне тестування (Sanity testing) направлено вглиб функції, що перевіряється, в той час як димове направлено вшир, для покриття тестами якомога більшої функціоналу в найкоротші терміни.

Розробка Agile і Waterfall.

Водоспадна модель (англ. waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад. ВИМОГИ – ПРОЕКТУВАННЯ – РЕАЛІЗАЦІЯ – ВЕРИФІКАЦІЯ – СУПРОВОДЖЕННЯ.

Яка система розробки використовується на проекті зараз.

Яка система розробки використовується знає кожен особисто.

Які ролі на проекті займає Junior, Middle, Senior.

Грейди в IT – це розподіл спеціалістів за рівнем їхньої компетенції, досвіду та ступенем відповідальності. Грейдування існує, щоб структурувати заробітну плату, розуміти, який існує кар'єрний ріст, позначити обов'язки та очікування від працівників, залежно від їхньої позиції.

— Intern (Стажер). Це фахівець з початковим рівнем знань, зазвичай студенти або недавні випускники, які ще не встигли отримати достатньо практичного досвіду. Часто вони безкоштовно стажуються в компаніях, щоб попрактикуватися і з досвідом роботи вже шукати оплачувану вакансію. Або ж сама компанія може після успішного стажування запропонувати їм оффер (унікальна пропозиція);

— Junior (Молодший спеціаліст). Це новачок, у якого вже є певний досвід роботи (зазвичай менше 2 років). Але він не може взяти на себе багато відповідальності, йому потрібне постійне наставництво і керівництво. Зазвичай він виконує завдання з низьким рівнем складності під наглядом більш досвідчених колег;

— Middle (Спеціаліст). Досвідчений співробітник, у якого за плечима від 2 до 5 років досвіду. Він уже готовий брати на себе відповідальність і може працювати автономно. Він справляється із завданнями середньої складності, бере участь у проєктах і може допомагати молодшим колегам;

— Senior (Старший спеціаліст). Сеньйор – це висококваліфікований співробітник, який уже понад 5 років працює у своїй сфері. Він має глибокі знання та значний досвід, здатний розв'язувати складні задачі, наставляти інших працівників, брати участь у плануванні проєктів та ухвалювати важливі технічні рішення;

— Lead (Провідний спеціаліст). Team Lead – нескладно здогадатися, що це та людина, яка керує командою або проєктом. Вона відповідає за координацію роботи команди, розподіл завдань, контроль якості та терміни виконання. Він може брати участь у стратегічному плануванні та ухвалювати рішення на рівні всієї компанії;

— Junior (Молодший розробник). Це найменш досвідчений фахівець із практичним досвідом менше 2 років. Зазвичай, який знає основи комп'ютерного програмування і може писати базовий код як мінімум однією мовою програмування. Уже на цьому рівні він має чудово розбиратися в комп'ютерах з програмно-апаратного забезпечення. Молодший розробник має постійно розвиватися, вчитися, багато запитувати, бути гнучким в освоєнні нових інструментів і вміти ефективно працювати над помилками;

— Middle (Середній розробник). Розробники рівня Middle є більш досвідченими, ніж Junior, оскільки вони вже працювали над кількома проєктами та мають досвід від 2 до 4 років. Вони здатні самостійно організувати середовище розробки, можуть розв'язувати різні задачі, працюючи як у команді, так і самостійно, а також бути наставниками для молодших розробників. Такі розробники беруть на себе більше відповідальності, коли йдеться про експлуатацію, але вони все одно потребують допомоги від Senior, якщо зіштовхнуться із завданням, яке раніше не вирішували;

— Senior (Старший розробник). Щоб стати senior-розробником, необхідно мати досвід програмування від 5 до 8 років. З таким досвідом уже не страшно самостійно розв'язувати питання різної складності, розуміти всю суть проєкту, над яким працює команда, розумітися на його аспектах, пропонувати нові підходи, що дадуть змогу поліпшити розробку з урахуванням потреб бізнесу. Крім хороших hard skills, senior-розробники мають бути крутими в комунікації, проявляти максимальну ініціативність і мати навички, щоб бути хорошими наставниками для інших фахівців.

— Lead (Провідний розробник) . Провідний розробник – це повноцінний архітектор проєктів, які може працювати як над великими серверними, так і над зовнішніми версіями важливих проєктів, а також грамотно поставити більш дрібні завдання. У таких фахівців понад 8 років досвіду. Щоб стати хорошим спеціалістом, потрібно відмінно розбиратися в зборі вимог, перевірці, управлінні, архітектурі, коді, інтеграції та тестуванні. Також не менш важливими є гнучкі навички – потрібно бути ефективним менеджером, який постійно піднімає планку програмного

забезпечення для кількох команд, володіти відмінними навичками ведення переговорів, щоб змусити команду йти за ним і його ідеями.

Різниця між error, bug, failure. Помилка (error) Дія людини, яке призводить до неправильного результату.

Баг (bug): Див. Дефект.

Дефект (defect): Вада в компоненті або системі, яка може привести компонент або систему до неможливості виконати необхідну функцію, наприклад невірний оператор або визначення даних. Дефект, виявлений під час виконання, може призвести до відмов компонента або системи.

Відмова (failure): Відхилення компонента або системи від очікуваного виконання, експлуатації або результату.

2. ПЕРЕЛІК ПИТАНЬ ДО САМОСТІЙНОЇ РОБОТИ

Тема 1. Основи та ключові питання тестування.

Тема для самостійного опрацювання: Введення в тестування програмного забезпечення.

Запитання для самоконтролю

1. Що таке тестування програмного забезпечення?
2. Які основні етапи побудови і розвитку ПЗ?
3. Що таке «парадокс тестування»?
4. Що таке «Оракул» в тестуванні?

Література : [1], [2].

Тема 2. Ознаки класифікації видів тестування та якість продукту·

Тема для самостійного опрацювання: Тестування за ступенем автоматизації та ізольованості компонентів.

Запитання для самоконтролю

1. Описати тестування за ступенем автоматизації? Їх переваги та недоліки.
2. Що таке інтеграційне тестування?
3. Як працює системне тестування?
4. Що таке модульне тестування?

Література : [1], [2], [3].

Тема 3. Вимоги та Фази тестування. Класи еквівалентності·

Тема для самостійного опрацювання: Поділ на класи еквівалентності.

Запитання для самоконтролю

1. Що таке клас еквівалентності?
2. Як відбувається техніка поділу на класи еквівалентності?
3. Навести приклад поділу на класи еквівалентності.

Література : [1], [2], [4].

Тема 4. Основні види та стани тест кейсу·

Тема для самостійного опрацювання: Навички, які необхідні для написання тест-кейсів.

Запитання для самоконтролю

1. Що таке тест-кейс?
2. Як працює високорівневий тест-кейс?
3. Як правильно написати низькорівневий тест-кейс?
4. Навести приклади та порядок виправлення помилок у тест-кейсах.
5. Як підвищити ефективність тестових кейсів.

Література : [1], [2], [4].

Тема 5. Виникнення дефектів та баг-репорти·

Тема для самостійного опрацювання: Порядок виправлення помилок за їх пріоритетами·

Запитання для самоконтролю

1. Що таке пріоритет ?
2. Який порядок виправлення помилок з їх пріоритетами?
3. Навести приклади виправлення помилок за пріоритетами.

Література : [1], [2], [6].

Тема 6. Класифікація апаратних, програмних засобів контролю та діагностування.

Тема для самостійного опрацювання: Пошук та усунення несправностей ПК.

Запитання для самоконтролю

1. Які основні функції мікросхеми BIOS?
2. Які основні причини збою програмного забезпечення?
3. Які компоненти ПК можуть давати збій в роботі і залишатися непоміченими?

Література : [1], [2], [8].

Тема 7. Загальні поняття надійності.

Тема для самостійного опрацювання: Методи оцінки надійності.

Запитання для самоконтролю

1. Як визначають надійність програмних комплексів?
2. У чому полягають особливості системного підходу до забезпечення надійності?
3. Які є методи оцінки надійності?
4. Охарактеризуйте поняття «відновлювані об‘єкти» та «невідновлювані об‘єкти».

Література : [1], [2], [10].

Тема 8. Пошкодження та відмови. Класифікація відмов.

Тема для самостійного опрацювання: Методи введення надлишковості.

Запитання для самоконтролю

1. Що таке надлишковість?
2. Охарактеризуйте поняття часова, інформаційна та структурна надлишковість.
3. Як впливає складність об‘єкта на його надійність?
4. В чому полягає поняття «технічний ресурс виробу» ?

Література : [1], [2], [11].

Тема 9. Етапи аналізу та показники надійності ПЗ.

Тема для самостійного опрацювання: Розрахунок основних показників надійності ПЗ.

Запитання для самоконтролю

1. Що таке імовірність відмови?
2. Як розрахувати інтенсивність відмов системи?
3. Що таке середній час відновлення?
4. Як розрахувати середню важкість помилок ?

Література : [1], [2], [11].

Тема 10. Фактори, що впливають на надійність ЕОМ та комп'ютерних систем.

Тема для самостійного опрацювання: Фактори зовнішнього впливу на якість електроживлення

Запитання для самоконтролю

1. Які є основні фактори зовнішнього впливу на якість електроживлення?
2. Які є засоби захисту системи електроживлення ЕОМ?
3. Яка можлива електробезпека при монтажі, обриві або відсутності заземлення?
4. Назвіть основні правила охорони праці під час експлуатації ЕОМ.

Література : [1], [2], [12].

Тема 11. Задачі та методи діагностування.

Тема для самостійного опрацювання: Завдання технічної діагностик.

Запитання для самоконтролю

1. В чому суть технічної діагностики?
2. Що є об'єктом технічного контролю?
3. Які завдання розв'язує технічна діагностика?

Література : [1], [2], [13].

Тема 12. Основні види датчиків інтернету речей.

Тема для самостійного опрацювання: Загроза і безпека в Інтернеті речей

Запитання для самоконтролю

1. Які основні аспекти необхідно врахувати при розгортанні IoT?
2. Чому захист IoT важливий?
3. Чим захист IoT відрізняється від захисту інформаційних технологій?
4. Які є методи та інструкції для вирішення питань безпеки?

Література : [1], [2], [14].

3. ВАРІАНТИ ЗАВДАНЬ ДЛЯ МОДУЛЬНОЇ КОНТРОЛЬНОЇ РОБОТИ

ВАРІАНТ №1

1. Суть тестового контролю.
2. Що таке POST та Post-карта.
3. Класифікація апаратних засобів.
4. Назвіть прилади з ручним керуванням. Дайте їм коротку характеристику.
5. Опишіть елементи логічного аналізатора.
6. Опишіть принцип роботи BIOS.

ВАРІАНТ №2

1. Тестовий контроль це.
2. Критерії вибору тестів та Проблема оракула.
3. Описати зовнішні апаратні засоби .
4. Для чого призначений логічний пробник.
5. Опишіть типи логічного аналізатора.
6. Програми для тестування оперативної пам’яті.

ВАРІАНТ №3

1. Як вибираються тести при деструктивному та конструктивному підходах.
2. Ефективність тестування та Тестування для виявлення дефектів
3. Критеріїв класифікації апаратури контролю.
4. Струмовий зонд.
5. Опишіть режими логічного аналізатора.
6. Програми для тестування процесора.

ВАРІАНТ №4

1. Основні задачі тестування.
2. Тестування чорного ящика?
3. Етапи контролю і діагностування.
4. Багатоконтактний логічний пробник.
5. Опишіть основні характеристики логічного аналізатора.
6. Інтерфейс та можливості тестуючої програми Victoria.

ВАРІАНТ №5

1. Що таке помилка та несправність.
2. Теоретичні і практичні обмеження тестування та Проблема нездійсненних шляхів.
3. Групи діагностичних програм. Опишіть.
4. Логічний пульсатор.
5. Опишіть способи запуску реєстрації ЛА.
6. Опишіть програму Memtest.

ВАРІАНТ №6

1. Тестування - процес ітераційний. Опишіть.
2. Що тестується автоматично при включенні комп’ютера.
3. Метод діагностування за допомогою схем вбудованого контролю. Опишіть.
4. Струмовий зонд.
5. У якій формі виводиться інформація на екран логічного аналізатора.
6. Aida. Яку інформацію Ви можете дізнатися з цієї програми.

ВАРІАНТ №7

1. Мета тестування.
2. Альфа- і бета- тестування.
3. Для виявлення несправностей схем контролю застосовують:
4. Прилади з ручним керуванням. Опишіть коротко.
5. Сигнатурний аналізатор. Його призначення та принцип роботи.
6. Вбудований засіб перевірки оперативної пам'яті

ВАРІАНТ №8

1. Тестовий контроль це –
2. Модульне, інтеграційне і системне тестування.
3. Властивості Схеми контролю, що самоперевіряються.
4. Струмовий зонд.
5. Призначення логічного аналізатора.

6. Що таке BIOS? Типи мікросхем для підпрограм BIOS (які були, які є на сьогоднішній день).

ВАРІАНТ №9

1. Що таке збій та відмова.
2. Невідкладні та терпимі умови залежно від серйозності помилки.
3. Критерії класифікації апаратури контролю.
4. Логічний пульсатор.
5. Що таке сигнатура та сигнатурний аналізатор
6. Можливості тестуючої програми HDDScan.

ВАРІАНТ №10

1. Як вибираються тести при деструктивному та конструктивному підходах.
2. Тестування білого ящика.
3. Основні випадки, коли необхідне діагностичне ПЗ.
4. Багатоконтактний логічний пробник **.**
5. Опишіть елементи логічного аналізатора.
6. Тест стабільності систем. Яка програма надає такі можливості. Опишіть програму.

ВАРІАНТ №11

1. Суть тестового контролю.
2. Що таке POST та Post-карта.
3. Класифікація апаратних засобів.
4. Назвіть прилади з ручним керуванням. Дайте їм коротку характеристику.
5. Опишіть елементи логічного аналізатора.
6. Опишіть принцип роботи BIOS.

ВАРІАНТ №12

1. Тестовий контроль це.
2. Критерії вибору тестів та Проблема оракула.
3. Описати зовнішні апаратні засоби .
4. Для чого призначений логічний пробник.
5. Опишіть типи логічного аналізатора.

6. Програми для тестування оперативної пам’яті.

ВАРІАНТ №13

1. Як вибираються тести при деструктивному та конструктивному підходах.
2. Ефективність тестування та Тестування для виявлення дефектів
3. Критеріїв класифікації апаратури контролю.
4. Струмовий зонд.
5. Опишіть режими логічного аналізатора.
6. Програми для тестування процесора.

ВАРІАНТ №14

1. Основні задачі тестування.
2. Тестування чорного ящика?
3. Етапи контролю і діагностування.
4. Багатоконтактний логічний пробник.
5. Опишіть основні характеристики логічного аналізатора.
6. Інтерфейс та можливості тестуючої програми Victoria.

ВАРІАНТ №15

1. Що таке помилка та несправність.
2. Теоретичні і практичні обмеження тестування та Проблема нездійсненних шляхів.
3. Групи діагностичних програм. Опишіть.
4. Логічний пульсатор.
5. Опишіть способи запуску реєстрації ЛА.
6. Опишіть програму Memtest.

ВАРІАНТ №16

1. Тестування - процес ітераційний. Опишіть.
2. Що тестується автоматично при включенні комп’ютера.
3. Метод діагностування за допомогою схем вбудованого контролю. Опишіть.
4. Струмовий зонд.
5. У якій формі виводиться інформація на екран ЛА
6. Aida. Яку інформацію Ви можете дізнатися з цієї програми.

ВАРІАНТ №17

1. Мета тестування.
2. Альфа- і бета- тестування.
3. Для виявлення несправностей схем контролю застосовують:
4. Прилади з ручним керуванням. Опишіть коротко.
5. Сигнатурний аналізатор. Його призначення та принцип роботи.
6. Можливості тестуючої програми Checklt Diagnostics.

ВАРІАНТ №18

1. Тестовий контроль це –
2. Модульне, інтеграційне і системне тестування.
3. Властивості Схеми контролю, що самоперевіряються.
4. Струмовий зонд.
5. Призначення логічного аналізатора.
6. Що таке BIOS? Типи мікросхем для підпрограм BIOS (які були, які є на сьогоднішній день).

ВАРІАНТ №19

1. Що таке збій та відмова.
2. Невідкладні та терпимі умови залежно від серйозності помилки.
3. Критерії класифікації апаратури контролю.
4. Логічний пульсатор.
5. Що таке сигнатура та сигнатурний аналізатор
6. Можливості тестуючої програми HDDScan.

ВАРІАНТ №20

1. Як вибираються тести при деструктивному та конструктивному підходах.
2. Тестування білого ящика.
3. Основні випадки, коли необхідне діагностичне ПЗ.
4. Багатоконтактний логічний пробник **.**
5. Опишіть елементи логічного аналізатора. Тест стабільності систем.

4. ОРІЄНТОВНІ ТЕСТОВІ ЗАВДАННЯ

Виберіть одну вірну відповідь з декількох запропонованих:

Тестування програмного забезпечення - це:

A. Пошук і виправлення всіх помилок в програмі.

B. Перевірка програми на наявність критичних помилок в коді.

C. Перевірка та порівняння поведінки програми з вимогами та очікуваним результатом.

D. Освоєння бюджету замовника.

Оберіть вірне твердження стосовно якості:

A. Якісна програма - це програма, яка не містить помилок.

B. Якісний дизайн, який подобається більшості користувачів продукту.

C. Сукупність властивостей продукції, які обумовлюють її придатність, задовольнити певні потреби відповідно до призначення.

D. Програма є якісною, якщо її протестували.

Оберіть вірне твердження стосовно визначень QA і QC?

A. Різниці немає. Це одне і теж поняття.

B. QA інженер орієнтований на пошук дефектів, в той час як QC орієнтований на покращення процесу.

C. QA - комплекс заходів, що охоплює всі технологічні аспекти на всіх етапах розробки. QC - процес контролю відповідності продукту що розробляється, до заявлених вимог.

D. QC - недопущення появи дефектів на етапі розробки, QA - виявлення і усунення дефектів на етапі розробки.

Оберіть правильне твердження стосовно валідації:

A. Аналіз продукту на відповідність до технічних вимог.

B. Аналіз продукту на відповідність до очікувань кінцевого користувача.

C. Аналіз продукту на наявність в ньому помилок.

D. Валідація не виконується під час тестування програмного забезпечення.

Оберіть правильне твердження стосовно верифікації:

A. Аналіз продукту на відповідність до технічних вимог.

B. Верифікація не виконується під час тестування програмного забезпечення.

C. Аналіз продукту на відповідність до очікувань кінцевого користувача.

D. Верифікація здійснюється по остаточному завершенню етапу розробки програмного забезпечення.

Принципом тестування не є:

A. Тестування може показати наявність помилок, але не довести їх відсутність.

B. Можна провести вичерпне тестування, яке б показало всі можливі комбінації введення даних від користувача та станів системи.

C. Тестування повинно починатись якомога раніше в життєвому циклі розробки програмного забезпечення.

D. Той факт, що не було знайдено жодної помилки, ще не гарантує якість програмного забезпечення.

Чому потрібно тестувати програмне забезпечення?

A. Для того, щоб переконатись, що програма виконує ті функції, які вона має виконувати і вирішує ті задачі, які повинна вирішувати.

B. Помилка в програмному забезпеченні може призвести до втрат грошей та часу.

C. Помилки можуть нести небезпеку здоров'ю та життю користувачів.

D. Все перелічене вище.

Життєвий цикл програмного забезпечення –

A. Період часу, який починається з моменту прийняття рішення про необхідність створення програмного продукту і закінчується в момент його повного вилучення з експлуатації.

B. Процес побудови і розвитку програмного забезпечення.

C. Це процес дослідження ПЗ з метою виявлення помилок і перевірки якості.

Етапи життєвого циклу ПЗ:

A. Аналіз вимог, Планування, Дизайн і розробка, Впровадження, Тестування, Оцінка, Реліз, Підтримка.

B. Планування, Аналіз вимог, Дизайн і розробка, Впровадження, Тестування, Оцінка, Реліз, Підтримка.

C. Планування, Аналіз вимог, Впровадження, Тестування, Оцінка, Реліз, Підтримка.

Тестування ПЗ:

A. Перевірка відповідності між реальною і очікуваною поведінкою програми.

B. Це процес дослідження ПЗ з метою виявлення помилок і перевірки якості.

C. Тестування – це одна з технік контролю якості, що включає в себе активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) та аналізу отриманих результатів (Test Analysis).

D. Всі відповіді вірні.

Верифікація:

A. Визначення відповідності ПЗ очікуванням і потребам користувача, вимогам до системи.

B. Це процес оцінки системи або її компонентів з метою визначення чи задовольняють результати поточного етапу розробки умовам, сформованим на початку цього етапу.

C. Це процес дослідження ПЗ з метою виявлення помилок і перевірки якості чи виконуються цілі, терміни, завдання, по розробці проекту, визначені на початку поточної фази.

Валідація:

A. Це визначення відповідності ПЗ очікуванням і потребам користувача, вимогам до системи.

B. Це процес дослідження ПЗ з метою виявлення помилок і перевірки якості програма.

Необхідні умови тестування?

A. Наявність тест кейсів / тестів.

B. Наявність плану тестування.

C. Наявність виконавця (ів) (людина або машина, або комбінація людина + машина).

D. Наявність об’єкта тестування, доступного для проведення випробувань.

Достатні умови тестування:

A. Наявність об’єкта тестування, доступного для проведення випробувань.

B. Наявність виконавця (ів) (людина або машина, або комбінація людина + машина).

C. Наявність плану тестування.

D. Наявність тест кейсів / тестів.

E. Наявність звіту, що підтверджує виконання завдань і досягнення цілей, з тестування об’єкта.

Тестовий випадок (Test Case):

A. Це артефакт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції, що тестується або її частини.

B. Це факт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції, що тестується або її частини.

C. Це документ, що описує весь обсяг робіт з тестування.

Баг / Дефект Репорт:

A. Це документ, що описує весь обсяг робіт з тестування.

B. Це документ, що описує ситуацію або послідовність дій, що призвела до некоректної роботи об’єкта тестування, із зазначенням причин і очікуваного результату.

C. Це етап процесу тестування ПЗ.

D. Всі відповіді вірні.

Тестове Покриття:

A. Це етап процесу тестування ПЗ.

B. Це рівень деталізації опису тестових кроків і необхідного результату.

C. Це одна з метрик оцінки якості тестування, що представляє із себе щільність покриття тестами.

D. Це час від початку проходження кроків тест кейса до отримання результату тесту.

Специфікація Тест Кейсів:

A. Це рівень деталізації опису тестових кроків і необхідного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття.

B. Це час від початку проходження кроків тест кейса до отримання результату тесту.

C. Це етап процесу тестування ПЗ.

Проблема оракула. «Оракул» в тестуванні:

A. Вдосконалення якості розробленого програмного забезпечення системи за допомогою виявлення дефектів, не знайдених раніше всіма іншими видами перевірок.

B. Спостереження за поведінкою програми, виконуваної в цілях тестування із заданими параметрами, за заданим сценарієм або з іншими заданими початковими умовами чи цілями тестування.

C. Це будь-який агент (людина або програма), що оцінює поведінку програми і формує висновок про результат тесту (тест пройдено чи ні). Цей висновок істотно залежить від трактування поняття «відмова» і «дефект» в конкретному контексті.

Функціональне тестування:

A. Виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог.

B. Це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням.

C. Це просто форма тестування продуктивності.

Тестування продуктивності:

A. Виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог.

B. Це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням.

C. Це просто форма тестування продуктивності.

Навантажувальне тестування:

A. Виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог.

B. Це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням.

C. Це просто форма тестування продуктивності.

Стрес-тестування:

A. Виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог.

B. Це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням.

C. Це просто форма тестування продуктивності.

D. Використовується для встановлення межі пропускної здатності програми. Цей тип тестування проводиться для визначення надійності системи при екстремальних або непропорційних навантаженнях.

Тестування стабільності:

A. Виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог.

B. Це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням.

C. Проводиться з метою переконатися в тому, що програма витримає очікуване навантаження протягом тривалого часу.

D. Використовується для встановлення межі пропускної здатності програми. Цей тип тестування проводиться для визначення надійності системи при екстремальних або непропорційних навантаженнях.

Об’ємне тестування:

A. Виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог.

B. Це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням.

C. Тестування проводиться зі збільшенням не навантаження і часу роботи, а кількості використовуваних даних, які зберігаються і використовуються в додатку.

D. Використовується для встановлення межі пропускної здатності програми. Цей тип тестування проводиться для визначення надійності системи при екстремальних або непропорційних навантаженнях.

Тестування білого ящика:

A. Під час тестування білого ящика використовуються метрики покриття коду.

B. Розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

C. Об'єктом тестування тут є не зовнішня, а внутрішня поведінка програми. розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

D. Це типово для юніт-тестування (unit testing), при якому тестуються тільки окремі частини системи.

E. Всі відповіді вірні.

Тестування чорного ящика:

A. Тестер має доступ до ПЗ тільки через ті ж інтерфейси, що й замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншому комп'ютеру або іншому процесу підключитися до системи для тестування.

B. Розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

C. Об'єктом тестування тут є не зовнішня, а внутрішня поведінка програми. розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

D. Це типово для юніт-тестування (unit testing), при якому тестуються тільки окремі частини системи.

E. Всі відповіді вірні.

Тестування сірого ящика:

A. Тестер має доступ до ПЗ тільки через ті ж інтерфейси, що й замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншому комп'ютеру або іншому процесу підключитися до системи для тестування.

B. Розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

C. Об'єктом тестування тут є не зовнішня, а внутрішня поведінка програми. розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

D. Це типово для юніт-тестування (unit testing), при якому тестуються тільки окремі частини системи.

E. Розробник тесту має доступ до вихідного коду, але при безпосередньому виконанні тестів доступ до коду, як правило, не потрібний.

Недоліки тестування «білої скриньки»:

A. Кількість незалежних маршрутів може бути дуже велика.

B. Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

C. У програмі можуть бути пропущені деякі маршрути.

D. Не можна виявити помилки, поява яких залежить від даних.

E. Всі відповіді правильні.

Переваги тестування «білої скриньки»:

A. Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

B. Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

C. При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

D. Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

E. Кожна з цих причин є аргументом для проведення тестування за принципом «білої скриньки». Тести «чорної скриньки» не зможуть реагувати на помилки таких типів.

F. Всі відповіді правильні.

На етапі статичного тестування перевіряється ?

A. Вся документація, отримана як результат життєвого циклу програми.

B. Технічне завдання.

C. Специфікація.

D. Вихідний текст програми на мові програмування.

E. Всі варіанти правильні.

На етапі динамічного тестування перевіряється ?

A. Вся документація, отримана як результат життєвого циклу програми.

B. Безпосереднього виконання програми.

C. Специфікація.

D. Вихідний текст програми на мові програмування.

E. Всі варіанти правильні.

Тестові скрипти:

A. Вся документація, отримана як результат життєвого циклу програми.

B. Пишуться для безпосереднього виконання програми.

C. Пишуться для перевірки компонентів, в яких найбільш висока ймовірність появи відмов або вчасно не знайдена помилка може бути дорогою.

D. Вихідний текст програми на мові програмування.

За ступенем автоматизації тестування є:

A. Ручне тестування.

B. Автоматизоване тестування.

C. Напівавтоматизоване тестування.

D. Всі відповіді правильні.

За ступенем ізольованості компонентів тестування є:

A. Компонентне.

B. Інтеграційне.

C. Системне.

D. Всі відповіді правильні.

Різниця між error, bug, failure:

A. Error - помилка дії людини, bug - дефект у системі, failure - відхилення системи.

B. Error - помилка дії програми, bug - дефект у системі, failure - помилка системи.

C. Error - помилка дії програми, bug - дефект у системі, failure - помилка експлуатації.

Що таке тестування?

A. Перевірка поведінки програми на кінцевому наборі тестів, обраному певним чином.

B. Перевірка відповідності між реальною та очікуваною поведінкою програми на кінцевому наборі тестів, обраному певним чином.

C. Перевірка відповідності між поведінкою програми на кінцевому наборі тестів, обраному певним чином.

Тестування установки?

A. Перевіряє працездатність методів установки, налаштування і видалення програми на всіх підтримуваних платформах.

B. Перевіряє працездатність методів установки.

C. Налаштування і видалення програми на всіх підтримуваних платформах.

Тестування документації:

A. Перевірка документації (documentation testing) – Перевіряє план та всю документацію.

B. Перевірка документації (documentation testing) – Перевіряє документацію користувача.

C. Перевірка документації (documentation testing) – Перевіряє точність всієї документації користувача.

Артефакти тестування:

A. Набір вхідних значень, передумов виконання, очікуваних результатів.

B. Набір вхідних значень, передумов виконання і фактичних результатів.

C. Набір вхідних значень, передумов виконання, очікуваних результатів і фактичних результатів.

D. Всі відповіді правильні.

Use case:

A. Послідовність операцій у взаємодії актора і компонента , при якій актором може бути як користувач та обмінюватися інформацією з системою.

B. Послідовність операцій системи зі значущим результатом, при якій актором може бути як користувач, так і все, що може обмінюватися інформацією з системою.

C. Послідовність операцій у взаємодії актора і компонента або системи зі значущим результатом, при якій актором може бути як користувач, так і все, що може обмінюватися інформацією з системою.

D. Всі відповіді правильні.

Основні поля test case:

A. Альфа-тестування.

B. Артефакти – це деякі продукти проекту, які використані в ньому при роботі.

C. Артефакти – це деякі продукти проекту, породжувані або використовувані в ньому при роботі над остаточним продуктом.

D. Артефакти – це деякі продукти проекту, породжувані або використовувані в ньому при роботі.

E. Всі відповіді правильні.

Цілі тестування:

A. Виявлення дефектів, Підвищення впевненості в рівні якості, Надання інформації для прийняття рішень.

B. Виявлення дефектів, Надання інформації для прийняття рішень, Запобігання дефектів.

C. Виявлення дефектів, Підвищення впевненості в рівні якості, Надання інформації для прийняття рішень, Запобігання дефектів.

Відмінності Load and Performance testing:

A. Load testing - вид тестування продуктивності, performance testing – процес тестування з метою визначити продуктивність ПЗ.

B. Performance testing – процес тестування з метою визначити продуктивність, load testing - тестування навантаженості.

C. Load testing – вид тестування продуктивності, performance testing - тестування навантаженості.

Серйозність:

A. Це атрибут, що характеризує вплив дефекту на працездатність програми.

B. Це атрибут, що характеризує якість програми.

C. Це атрибут, що характеризує вплив дефекту на програму.

Основні види тестування:

A. Тестування змін, Тестування структури, Нефункціональне тестування, Функціональне тестування.

B. Функціональне тестування, Нефункціональне тестування, Тестування змін.

C. Тестування змін, функціональне та нефункціональне тестування.

5. КРИТЕРІЇ ОЦІНЮВАННЯ

Порядок поточного оцінювання знань.

Оцінка з поточного контролю визначається як середня арифметична оцінка з усіх навчальних занять та розраховується при оцінюванні після проведення останнього у семестрі навчального заняття.

Контроль за виконанням практичних робіт.

Протягом семестру студенти повинні виконати практичні роботи передбачені робочою програмою дисципліни.

Порядок оцінювання наступний:

— якщо робота виконана грамотно і акуратно оформлена, під час захисту роботи студент дає вірні відповіді на запитання, демонструє знання підручників, посібників, викладає в логічній послідовності теоретичний матеріал – 90-100 балів;

— якщо виконана лабораторна робота грамотно і акуратно оформлена, під час захисту лабораторної роботи студент дає вірні відповіді на запитання, демонструє знання підручників, посібників але ним допущені незначні помилки у формуванні термінів, розрахунків – 75-89 балів;

— якщо робота виконана не в повному обсязі або із значними помилками, та не оформлена відповідним чином або студент дає невірні відповіді на запитання, допускає помилки у формуванні термінів, категорій, розрахунків – 60-74 балів;

— якщо робота не виконана, студент частково володіє навчальним матеріалом та не в змозі викласти зміст більшості питань – 1-35.

Контроль за виконанням модульних завдань.

При виконанні завдань модульних контрольних робіт оцінюванню підлягають теоретичні знання та практичні навички, яких набули студенти після опанування матеріалу дисципліни.

Протягом семестру проводиться дві модульні контрольні роботи щодо перевірки рівня засвоєння знань студентами. Модульні контрольні завдання містять теоретичні питання. Контрольні завдання складені з урахуванням вимоги однакової складності для всіх студентів.

Модульний контроль містить 60 тестових завдань. Відповідь на кожне питання модульного контролю оцінюється в 0,5 (вірна) або 0(не вірна) балів. Загальна сума балів складає 100 балів.

Тестові завдання охоплюють теоретичний та практичний матеріали тем, який вивчаються в межах навчальної дисципліни «Тестування та діагностика програмно-апаратних засобів» та згруповані за двома модулями, кожен з яких складається з тестових завдань різного рівня складності.

Модульний контроль проводиться у терміни згідно з графіком освітнього процесу в очній формі та/або з використання технологій дистанційного навчання. Завдання модульного контролю виконується кожним здобувачем вищої освіти індивідуально.

Здобувач вищої освіти, який на складання модульного контролю не з'явився:

— з поважної причини – допускається до його складання за згодою декана факультету;

— без поважної причини – за модульний контроль отримує 0 балів.

Здобувач вищої освіти вважається допущеним до семестрового контролю (екзамену/заліку), якщо він виконав усі види робіт, передбачені цією робочою програмою.

Здобувачі вищої освіти, які до початку екзаменаційної сесії хоча б з одного контрольного заходу по модульному контролю мають менше 60 балів, не одержують заліку як такі, що не виконали програми з освітнього компонента.

Контроль за виконанням завдань для самостійного опрацювання.

При контролі виконання завдань для самостійного опрацювання оцінці підлягає самостійне опрацювання окремих питань.

Навчальний матеріал дисципліни, передбачений для засвоєння здобувачем освіти під час самостійної роботи, повинен виноситись на підсумковий контроль разом із навчальним матеріалом, що опрацьовувався під час проведення навчальних занять.

Порядок підсумкового оцінювання знань.

Здобувач освіти вважається допущеним до семестрового контролю (заліку) з освітнього компонента, якщо він виконав усі види робіт, передбачені робочою програмою навчальної дисципліни. Підсумковий контроль проводиться з метою оцінювання результатів навчання здобувача освіти на певному рівні вищої освіти або на окремих його завершальних етапах за національною шкалою і шкалою ЄКТС. Підсумковий контроль включає семестровий контроль успішності здобувача освіти.

Семестровий контроль проводиться у формі заліку.

Підсумковий бал (за 100-бальною шкалою) з дисципліни «Тестування та діагностика програмно-апаратних засобів» визначається як середньозважена величина, залежно від питомої ваги кожної складової залікового кредиту.

Оцінювання знань здобувачів вищої освіти здійснюється відповідно до загальних критеріїв паралельно за:

– позитивні оцінки – «зараховано», негативні оцінки – «незараховано»;

– 100-бальною накопичувальною шкалою ЄКТС.

Така система оцінювання знань здобувачів вищої освіти, що ґрунтується на поєднанні модульних технологій навчання та залікових освітніх одиниць (залікових кредитів), дозволяє здійснювати оцінювання більш гнучко, об’єктивно і сприяє систематичній та активній самостійній роботі здобувачів вищої освіти впродовж усього навчання, стимулює виявлення і розвиток їх творчих здібностей, забезпечує змагальність між здобувачами вищої освіти у навчанні тощо.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Міскевич О. І. Тестування та діагностика програмно-апаратних засобів. Конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спец. 123 Комп'ютерна інженерія денної та заочної форм навчання. Луцьк: ЛНТУ, 2024. 84 с.

2. Міскевич О. І. Тестування та діагностика програмно-апаратних засобів : метод. вказівки до практичних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спец. 123 Комп'ютерна інженерія денної та заочної форм навчання. Луцьк : ЛНТУ, 2024. 60 с.

3. Чернящук Н.Л., Христинець Н.А. Архітектура комп'ютерів: конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології» спец. 123 Комп'ютерна інженерія денної та заочної форм навчання. Луцьк : Луцький НТУ, 2020. 108 с.

4. Міскевич О., Столенко І., Куліков Н. Фреймворк testNG як інструмент для написання тестів. *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*. Луцьк : Видавництво ЛНТУ, 2023. Вип.53. С.152–157.

5. Міскевич О. Аналіз роботи мережевих утиліт в командному вікні Windows. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*· Луцьк : Видавництво ЛНТУ. 2023. Вип. 50. С. 84–89.

6. Міскевич О., Каган І., Рожко О. Як обрати оптимальний ноутбук для навчання. *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*. Луцьк : Видавництво ЛНТУ. 2021. Вип.43. С. 92–96.

7. Михалик А. В., Христинець Н. А., Міскевич О. І. Продуктивність технології Crossfire X при навантаженні відеоадаптерів мікропроцесорів AMD. *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*. Луцьк : Видавництво ЛНТУ. 2020. Вип. 39. С. 213–217.

8. Добірка корисних подкастів для IT-фахівців: URL: https://training.epam.ua/#!/News/399?lang=ua (дата звернення: 10.01.2025).

9. Про тестування та тестувальників без міфів та стереотипів: URL: https://training.epam.ua/#!/News/-318?lang=ua. (дата звернення: 10.02.2025).

10. USB Enabling Connections USB (universal serial bus). URL: http://www.usb.org/home. (дата звернення: 15.01.2025).

11. Офіційний сайт Cisco. URL: https://www.netacad.com/ru /courses/all-courses. (дата звернення: 12.02.2025).

12. Діагностичні програми. URL: https://biblprog.org.ua/ua/diagnostic/ (дата звернення: 12.02.2025).

13. Офіційний сайт Prometheus. URL : https://courses.prometheus.org.ua/ (дата звернення: 10.02.2025).

14. Офіційний сайт Дія. Освіта. URL: https://osvita.diia.gov.ua/ (дата звернення: 10.02.2025).

Т-36 **Тестування та діагностика програмно-апаратних засобів.** Методичні вказівки до виконання самостійної роботи для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. О. І. Міскевич, С. В. Гринюк. Луцьк : ЛНТУ, 2025. 50 с.

Комп’ютерний набір: О. І. Міскевич

Редактор: О. І. Міскевич

Підп. до друку «___» _____ 2025р.
Формат 60х84/16. Папір офс. Гарнітура Таймс.
Ум. друк. арк. ____. Тираж 10 прим. Зам. ____

Відділ іміджу та промоції
Луцького національного технічного університету
43018, м. Луцьк, вул. Львівська, 75
Друк – ВІП ЛНТУ