МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО
«ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра економічної кібернетики та інженерії програмного забезпечення

МЕТОДИЧНІ ВКАЗІВКИ ДО КУРСОВОЇ РОБОТИ

з дисципліни «Якість програмного забезпечення та тестування»
для студентів денної та заочної форм навчання
за спеціальністю
121 − Інженерія програмного забезпечення

Запоріжжя
2021-2022 н. р.

Методичні вказівки до курсової роботи з дисципліни «Якість програмного забезпечення та тестування» для студентів денної та заочної форм навчання за спеціальністю 121 – Інженерія програмного забезпечення.

Укладач: к.т.н., доц. кафедри ЕКІПЗ Резніченко Ю.С.

Ухвалено на засіданні кафедри ЕКІПЗ
протокол № 1
від «30» 08 2021 р.
Зав. кафедри ЕКІПЗ
д.е.н., доц. Левицький С.І.

ЗМІСТ

ВСТУП

Тестування програмного забезпечення (ПЗ) – процес дослідження ПЗ з метою отримання інформації про якість програмного продукту, а саме відповідність специфікації, технічному завданню та вимогам замовника ПЗ.

Практичний підхід до тестування ПЗ особливу увагу приділяє процесам тестування на фоні стрімкого прискорення процесу розробки ПЗ. Цей підхід орієнтований на використання спеціалістами з тестування ПЗ тестових робіт. Швидкість та ефективність розробки ПЗ залежить від того, наскільки процес тестування вписується до загального життєвого циклу розробки ПЗ та від ефективності використання технології тестування.

Тестування − одна з технік контролю якості, що включає в себе діяльність з планування робіт (Test Management), проектуванню тестів (Test Design), виконанню тестування (Test Execution) та аналізу отриманих результатів (Test Analysis).

Необхідними умовами для тестування є наявність об'єкту тестування, доступного для проведення тестів та виконавця(ів) (як людина, так і машина або комбінація людина + машина). Достатніми умовами для тестування додатково до необхідних умов є наявність плану тестування, тест-кейсів/тестів, звіту, що підтверджує виконання задач та досягнення цілей з тестування об'єкту.

У даних методичних вказівках викладено основні принципи, технології тестування, вимоги до документів з тестування згідно відповідних стандартів:

- тест-план (тест план IEEE 829, тест план RUP, план приймально-здавальних випробувань RUP, план проведення навантажувального тестування);
- тест-дизайн специфікації (тест-дизайн специфікація MSF, тест-дизайн специфікація IEEE 829-1998);
- тестовий випадок (test case);
- звіт про помилку (bug report).

Основну увагу приділено налаштуванню процесу тестування, що дозволяє як можна скоріше реалізувати якісний програмний продукт.

1 МЕТА ТА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

Курсова робота (КР) з навчальної дисципліни «Якість програмного забезпечення та тестування» призначена для закріплення та практичного застосування основних стандартів, що використовуються на різних етапах життєвого циклу програмного забезпечення для оцінювання якості, існуючих моделей та метрик якості, а також основних методів модульного, комплексного тестування та сучасних програмних засобів інженерії якості.

Метою КР є підготовка фахівців, що володіють знаннями та навичками щодо визначення критеріїв якості програмного забезпечення, володіють методиками забезпечення якості та сучасними технологіями тестування програмних продуктів.

Основним завданням КР є сформувати у фахівців наступні компетентності: інтегральну − здатність розв’язувати складні спеціалізовані завдання або практичні проблеми інженерії програмного забезпечення, що характеризуються комплексністю та невизначеністю умов, із застосуванням теорій та методів інформаційних технологій; фахові − здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування; здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами; здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу; здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.

У результаті виконання КР фахівці повинні: знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення; знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення; вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання; застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення; застосовувати

на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення; мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення; мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації; знати та вміти застосовувати методи верифікації та валідації програмного забезпечення; знати підходи щодо оцінки та забезпечення якості програмного забезпечення.

КР є самостійною роботою студентів. Відповідальність за повноту та правильність теоретичних висновків та практичних результатів, а також їх опис несуть студенти, як автори власних КР.

2 ПОРЯДОК ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Основні етапи виконання КР:

a) виконання практичного завдання;

b) оформлення пояснювальної записки;

c) захист роботи.

3 ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

3.1 Розробка тест-плану

Тестування програмного забезпечення (Software Testing) – це перевірка відповідності між реальною і очікуваною поведінкою програми, що здійснюється на кінцевому наборі тестів, обраних певним чином. (IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004) У більш широкому змісті, тестування - це одна з технік контролю якості, що включає в себе діяльність з планування робіт (Test Management), проектуванню тестів (Test Design), виконанню тестування (Test Execution) і аналізу отриманих результатів (Test Analysis).

Верифікація (verification) програми і її компонентів з метою визначення чи задовольняють результати поточного етапу розробки умовам, сформованим на початку цього етапу (IEEE). Тобто чи виконуються цілі, терміни, задачі з розробки проекту, визначені на початку поточної фази.

Валідація (validation) - це визначення відповідності розроблювального ПЗ очікуванням і потребам користувача, вимогам до системи (BS7925-1).

План тестування (Test Plan) - це документ, що описує обсяг робіт з по тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи устаткування, спеціальних знань, а також оцінки ризиків з варіантами їхнього рішення.

Тест дизайн (Test Design) - це етап процесу тестування ПЗ, на якому проектуються і створюються тестові випадки (тест кейси), відповідно до визначених раніше критерій якості і цілей тестування.

Тестовий випадок (Test Case) - це сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції або її частини, що тестуються.

Звіт про помилку/Дефект Репорт (Bug Report) - це документ, що описує ситуацію або послідовність дій, що призвели до некоректної роботи об'єкта тестування, із вказівкою причин і очікуваного результату.

Тестове Покриття (Test Coverage) - це одна з метрик оцінки якості тестування, що представляє із себе щільність покриття тестами вимог або коду, що виконується.

Деталізація Тест Кейсів (Test Case Detalization) - це рівень деталізації опису тестових кроків і необхідного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття.

Час Проходження Тест Кейса (Test Case Pass Time) - це час від початку проходження кроків тест кейса до одержання результату тесту.

Залежно від переслідуваних цілей види тестування можна умовно розділити на наступні типи: функціональне, не функціональне, пов'язані зі змінами.

Функціональні тести ґрунтуються на функціях, виконуваних системою, і можуть проводитися на всіх рівнях тестування (компонентному, інтеграційному, системному, приймальному). Як правило, ці функції описуються у вимогах, функціональних специфікаціях або у виді випадків використання системи (use cases).

Тестування функціональності може проводиться в двох аспектах: вимоги та бізнеси-процеси.

Тестування в перспективі «вимоги» використовує специфікацію функціональних вимог до системи як основу для дизайну тестових випадків (Test Cases). У цьому випадку необхідно зробити перелік того, що буде тестуватися, а що ні, визначають пріоритетність вимог на основі ризиків (якщо це не зроблено в документі з вимогами), а на основі цього переліку пріоритетів формуються тестові сценарії (test cases). Це дозволяє сфокусуватися при тестуванні на важливішому функціоналі.

Тестування в перспективі «бізнеси-процеси» використовує знання цих самих бізнесів-процесів, що описують сценарії щоденного використання системи. У цій перспективі тестові сценарії (test scripts), як правило, ґрунтуються на випадках використання системи (use cases).

Тестування безпеки (Security and Access Control Testing) – це перевірка безпеки системи, а також аналіз ризиків, пов'язаних із забезпеченням цілісного підходу до захисту додатка, атак хакерів, вірусів, несанкціонованого доступу до

конфіденційних даних. Тестування безпеки може виконуватися як автоматизовано так і в ручну, включаючи перевірку як позитивних, так і негативних тестових випадків. Ґрунтується на трьох основних принципах - це конфіденційність, цілісність і доступність (confidentiality, integrity, availability).

Тестування взаємодії (Interoperability Testing) – це функціональне тестування, що перевіряє здатність додатка взаємодіяти з одним і більш компонентами або системами, що включає в себе тестування сумісності (compatibility testing) і інтеграційне тестування (integration testing).

Задачею тестування продуктивності (Performance testing) є визначення масштабованості додатка під навантаженням, при цьому відбувається:

- вимір часу виконання обраних операцій при визначених інтенсивностях виконання цих операцій;
- визначення кількості користувачів, що одночасно працюють з додатком;
- визначення границь прийнятної продуктивності при збільшенні навантаження (при збільшенні інтенсивності виконання цих операцій);
- дослідження продуктивності на високих, граничних, стресових навантаженнях.

Навантажувальне тестування (Load vs Performance Testing). В англомовній термінології ви можете так само знайти ще один вид тестування - Load Testing - тестування реакції системи на зміну навантаження (у межах припустимого). Load і Performance переслідують одну й теж саму мету: перевірка продуктивності (часів відгуку) на різних навантаженнях. Власне тому їх не розділяють.

Стресове тестування (Stress Testing) 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 у даному контексті може бути підвищення інтенсивності виконання операцій до дуже високих значень або аварійна зміна конфігурації сервера. Також однієї з задач при стресовому тестуванні може бути оцінка деградації продуктивності, у такий спосіб мети стресового тестування можуть перетинатися з цілями тестування продуктивності.

Тестування стабільності або надійності (Stability / Reliability Testing) – це перевірка працездатності додатка при тривалому (багатогодинному) тестуванні із

середнім рівнем навантаження. Часи виконання операцій можуть грати в даному виді тестування другорядну роль. При цьому на перше місце виходить відсутність витоків пам'яті, перезапусків серверів під навантаженням і інші аспекти, що впливають саме на стабільність роботи.

Об'ємне тестування (Volume Testing) -це одержання оцінки продуктивності при збільшенні обсягів даних у базі дані додатки, при цьому відбувається:

- вимір часу виконання обраних операцій при визначених інтенсивностях виконання цих операцій;
- може вироблятися визначення кількості користувачів, що одночасно працюють з додатком.

Тестування установки (Installation Testing) направлено на перевірку успішної інсталяції і настроювання, а також відновлення або видалення ПЗ. Інсталяція відбувається автоматично вручну та за допомогою візардів.

Тестування зручності користування (Usability Testing) - це метод тестування, спрямований на встановлення ступеня зручності використання, навчання, зрозумілості і привабливості для користувачів розроблювального продукту в контексті заданих умов.

Тестування, що пов'язане зі змінами.

Димове тестування (Smoke Testing) спрямовано на поверхневу перевірку всіх модулів додатка на предмет працездатності і наявність швидкого знаходження критичних і дефектів, що блокують. За результатами димового тестування робиться висновок про те, чи приймається чи ні установлена версія ПЗ в тестування, експлуатацію або на постачання замовникові. Для полегшення роботи, економії часу і людських ресурсів рекомендується автоматизувати димові тести.

Регресійне тестування (Regression Testing) спрямовано на перевірку змін, зроблених у додатку або навколишнім середовищі (налагодження дефекту, злиття коду, міграція на іншу операційну систему, базу даних, веб сервер або сервер додатка), для підтвердження того факту, що існуюча раніше функціональність працює як і колись. Регресійними можуть бути тести як функціональні, так і не функціональні.

Як правило, для регресійного тестування використовуються тест кейси, написані на ранніх стадіях розробки і тестування. Це дає гарантію того, що зміни в новій версії додатка не зашкодили вже існуючу функціональність. Рекомендується робити автоматизацію регресійних тестів, для прискорення наступного процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення.

3 основних типи регресійного тестування:

- регресія помилок - багів (Bug regression) - спроба довести, що виправлена помилка насправді не виправлена;
- регресія старих помилок - багів (Old bugs regression) - спроба довести, що недавня зміна коду або даних зламало виправлення старих помилок, тобто старі помилки - баги стали знову відтворюватися;
- регресія побічного ефекту (Side effect regression) - спроба довести, що недавня зміна коду або даних зламало інші частини розроблювального додатка.

Тестування зборки (Build Verification Test). Тестування спрямоване на визначення відповідності, випущеної версії, критеріям якості для початку тестування. По своїм цілям є аналогом Димового Тестування, спрямованого на приймання нової версії в подальше тестування або експлуатацію. Вглибину воно може проникати далі, залежно від вимог до якості випущеної версії.

Санітарне тестування або перевірка погодженості/справності (Sanity Testing). Вузькоспеціалізоване тестування достатнє для доказу того, що конкретна функція працює згідно заявленим у специфікації вимогам. Є підмножиною регресійного тестування. Використовується для визначення працездатності визначеної частини додатка після змін зроблених у ньому або навколишньому середовищу. Звичайно виконується вручну.

3.2 Розробка тестових випадків

Тестування на різних рівнях виробляється протягом усього життєвого циклу розробки і супроводу ПЗ. Рівень тестування визначає те, над чим виробляються тести: над окремим модулем, групою модулів або системою, у

цілому. Проведення тестування на всіх рівнях системи - це основа успішної реалізації і здачі проекту.

Рівні тестування:

- компонентне або модульне тестування (Component Testing or Unit Testing);
- інтеграційне тестування (Integration Testing);
- системне тестування (System Testing);
- приймальне тестування (Acceptance Testing).

Компонентне (модульне) тестування (Component or Unit Testing) перевіряє функціональність і шукає дефекти в частинах додатка, які доступні і можуть бути протестовані окремо (модулі програм, об'єкти, класи, функції тощо). Звичайно компонентне (модульне) тестування проводиться викликаючи код, який необхідно перевірити і за підтримкою середовищ розробки, таких як фреймворки (frameworks - каркаси) для модульного тестування або інструменти для налагодження. Усі знайдені дефекти, як правило виправляються в коді без формального їхнього опису в системі менеджменту помилок/дефектів - багів (Bug Tracking System).

Один з найбільш ефективних підходів до компонентного (модульного) тестування - це підготовка автоматизованих тестів до початку основного кодування (розробки) програмного забезпечення. Це називається «розробка від тестування» (test-driven development) або «підхід тестування спочатку» (test first approach). При цьому підході створюються й інтегруються невеликі частини коду, напроти яких запускаються тести, написані до початку кодування. Розробка ведеться доти поки всі тести не будуть успішними.

Інтеграційне тестування (Integration Testing) призначене для перевірки зв'язку між компонентами, а також взаємодії з різними частинами системи (операційною системою, устаткуванням або зв'язком між різними системами).

Рівні інтеграційного тестування:

- компонентний інтеграційний рівень (Component Integration testing). Перевіряється взаємодія між компонентами системи після проведення компонентного тестування;
- системний інтеграційний рівень (System Integration Testing). Перевіряється взаємодія між різними системами після проведення системного тестування.

Підходи до інтеграційного тестування:

- знизу нагору (Bottom Up Integration). Усі низькорівневі модулі, процедури або функції збираються разом і потім тестуються. Після чого збирається наступний рівень модулів для проведення інтеграційного тестування. Даний підхід вважається корисним, якщо всі або практично всі модулі, рівня, що розробляється готові. Також даний підхід допомагає визначити за результатами тестування рівень готовності додатка;
- зверху вниз (Top Down Integration). Спочатку тестуються усі високорівневі модулі, і поступово один за іншим додаються низькорівневі. Усі модулі більш низького рівня симулюються заглушками з аналогічною функціональністю, потім в міру готовності вони заміняються реальними активними компонентами;
- великий вибух ("Big Bang" Integration). Всі або практично всі розроблені модулі збираються разом у вигляді закінченої системи або її основної частини, і потім проводиться інтеграційне тестування. Такий підхід дуже гарний для збереження часу. Однак якщо тест кейси і їхні результати записані не вірно, то сам процес інтеграції сильно ускладниться, що стане перешкодою для команди тестування при досягненні основної мети інтеграційного тестування.

Основною задачею системного тестування є перевірка як функціональних, так і не функціональних вимог у системі в цілому. При цьому виявляються дефекти, такі як невірне використання ресурсів системи, непередбачені комбінації даних користувальницького рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або невірна функціональність, незручність використання тощо. Для мінімізації ризиків, пов'язаних з особливостями поведінки в системі в будь-якому середовищі, під час тестування рекомендується використовувати оточення максимальне наближене до того, на яке буде встановлений продукт після видачі.

Можна виділити два підходи до системного тестування:

- на базі вимог (requirements based).Для кожної вимоги пишуться тестові випадки (test cases), що перевіряють виконання даної вимоги;
- на базі випадків використання (use case based). На основі представлення

про способи використання продукту створюються випадки використання системи (Use Cases). По конкретному випадку використання можна визначити один або більш сценаріїв. На перевірку кожного сценарію пишуться тест кейси (test cases), які мають бути протестовані.

Формальний процес приймального тестування, що 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.

Приймальне тестування виконується на підставі набору типових тестових випадків і сценаріїв, розроблених на підставі вимог до даного додатка.

Рішення про проведення приймального тестування приймається, коли:

- продукт досяг необхідного рівня якості;
- замовник ознайомлений із планом приймальних робіт (Product Acceptance Plan) або іншим документом, де описаний набір дій, пов'язаних із проведенням приймального тестування, дата проведення, відповідальні особи тощо.

Фаза приймального тестування триває доти, поки замовник не виносить рішення про відправлення додатка на доробку або видачі додатка.

Тестовий випадок (test case) - сукупність вхідних даних тесту, умови виконання і очікуваних результатів, які розроблені для конкретної мети. Тестовий випадок - це найменша одиниця тестування, яку можна самостійно виконати від початку до кінця. Шаблони тестового випадку і зразок їх заповнення представлені у додатку В.

Розглянемо особливості заповнення полів шаблону тестування.

Ідентифікатор тестового випадку - включає номер версії тесту.

Власник тесту – ПІБ особи, що експлуатує тест (воно може не співпадати з ПІБ автора тесту).

Дата останнього перегляду – ця інформація визначає актуальність тесту.

Назва тесту - опис назви тесту, що дозволяє його легко знайти і зрозуміти його призначення. Не рекомендується вживати назви, що не несуть ніякого сенсового навантаження, наприклад, "xxxLLL0123.tst".

Місцезнаходження тесту – повна назва шляху, розташування на диску ЕОМ.

Технічна вимога, що тестується - унікальний ідентифікатор, який відображається в документах технічних вимог.

Мета тестування - формулювання того, що має досягти тест.

Конфігурація засобів тестування - специфікація вводу/виводу, умови випробувань.

Налаштування на прогін тесту - процедура подібна методиці тестування. Вона передбачає опис дій тестувальника і очікуваних результатів. Якщо настроювання автоматизовані, це виглядає так: run setupSC03.pl.

Методика тестування - опис дій тестувальника і очікуваних результатів.

Взаємозалежність тестових випадків – ідентифікація будь-якого тестового випадку. Для того, щоб виконання даного тесту починалося при означених умовах, необхідно здійснити прогін попередніх тестів.

Очистка тесту – якщо система була переведена в нестійкий стан або дані були зруйнованими, очистка дозволяє усунути подібні ситуації.

3.3 Техніка тест-дизайну

Тест-дизайн – це етап процесу тестування ПЗ, на якому проектуються та створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріїв якості і цілей тестування.

План роботи над тест дизайном:

- аналіз існуючих проектних артефактів: документація (специфікації, вимоги, плани), моделі, що виконується код, тощо;
- написання специфікації з тест дизайну (Test Design Specification);
- проектування і створення тестових випадків (Test Cases).

Техніки тест-дизайну:

1. Еквівалентний поділ (Equivalence Partitioning - EP). Як приклад, у вас є діапазон припустимих значень від 1 до 10, ви повинні вибрати одне вірне значення усередині інтервалу, скажемо, 5, і одне невірне значення поза інтервалом - 0.

2. Аналіз граничних значень (Boundary Value Analysis - BVA). Якщо взяти приклад вище, як значення для позитивного тестування виберемо мінімальну і максимальну границі (1 і 10), і значення більше і менше границь (0 і 11). Аналіз Граничних значень може бути застосований до полів, записів, файлів, або до будь-якого роду сутностей, що мають обмеження.
3. Причина/Наслідок (Cause/Effect - CE). Це, як правило, уведення комбінацій умов (причин), для одержання відповіді від системи (Наслідок). Наприклад, ви перевіряєте можливість додавати клієнта, використовуючи визначену екранну форму. Для цього вам необхідно буде ввести кілька полів, таких як "Ім'я", "Адреса", "Номер Телефону" а потім, натиснути кнопку "Додати" - ця "Причина". Після натискання кнопки "Додати", система додає клієнта в базу даних і показує його номер на екрані - це "Наслідок".
4. Передбачення помилки (Error Guessing - EG). Це коли тест аналітик використовує свої знання системи і здатність до інтерпретації специфікації на предмет того, щоб "вгадати" при яких вхідних умовах система може видати помилку. Наприклад, специфікація говорить: "користувач має увести код". Тест аналітик, буде думати: "Що, якщо я не введу код?", "Що, якщо я введу неправильний код? ", і так далі. Це і є здогадування помилки.
5. Вичерпне тестування (Exhaustive Testing - ET) - це крайній випадок. У межах цієї техніки ви повинні перевірити всі можливі комбінації вхідних значень, і в принципі, це повинно знайти всі проблеми. На практиці застосування цього методу не представляється можливим, через величезну кількість вхідних значень.

План розробки тест випадків - кейсів містить:

1. Аналіз вимог.
2. Визначення набору тестових даних на підставі EP, BVA, EG.
3. Розробка шаблона тесту на підставі CE.
4. Написання тест-кейсів на підставі первинних, тестових даних і кроків тесту.

3.4 Розробка звітів про дефекти

Звіт про помилки/дефекти (Bug Report, Баг репорт) - це технічний документ і мова опису має бути технічною. Повинна використовуватися правильна термінологія при використанні назв елементів інтерфейсу користувача (editbox, listbox, combobox, link, text area, button, menu, popup menu, title bar, system tray, тощо.), дій користувача (click link, press the button, select menu item,тощо) і отриманих результатах (window is opened, error message is displayed, system crashed, тощо).

Обов'язковими полями баг репорту є: короткий опис (Bug Summary), серйозність (Severity), кроки до відтворення (Steps to reproduce), результат (Actual Result), очікуваний результат (Expected Result). Нижче приведені вимоги і приклади по заповненню цих полів.

Короткий опис. Зміст усього баг репорту. Наприклад:

1. Додаток зависає, при спробі збереження текстового файлу розміром більше 50Мб.
2. Дані на формі «Профайл» не зберігаються після натискання кнопки «Зберегти».

При написанні баг-репорту використовується принцип «Де? Що? Коли?».

Де?: У якому місці інтерфейсу користувача або архітектури програмного продукту знаходиться проблема. Причому, починайте пропозицію з іменника, а не з прийменника (предлог).

Що?: Що відбувається або не відбувається відповідно до специфікації або вашій уяві про нормальну роботу програмного продукту. При цьому вказуйте на наявність або відсутність об'єкта проблеми, а не на його зміст (його вказують в описі). Якщо зміст проблеми варіюється, усі відомі варіанти вказуються в описі.

Коли?: У який момент роботи програмного продукту, або при якій події або при яких умовах проблема виявляється.

Серйозність. Якщо проблема знайдена в ключовій функціональності додатку і після її виникнення додаток стає цілком недоступний, і подальша робота з ним неможлива, то вона є помилкою, що блокує. Звичайно всі проблеми, що блокують, знаходяться під час первинної перевірки нової версії продукту

(Build Verification Test, Smoke Test), тому що їхня наявність не дозволяє повноцінно проводити тестування. Якщо ж тестування може бути продовжено, то серйозність даного дефекту буде критична.

Кроки до відтворення / Результат / Очікуваний результат. Дуже важливо чітко описати всі кроки, зі згадуємо усіх вводи даних (імені користувача, даних для заповнення форми) і проміжних результатів.

Наприклад, кроки до відтворення:

1. Увійдіть у систему: Користувач Тестер1, пароль xxxXXX > Вхід у систему здійснений.
2. Кликніть линк Профайл > Сторінка Профайл відкрилася.
3. Уведіть Нове ім'я користувача: Тестер2.
4. Натисніть кнопку Зберегти.

Результат. На екрані з'явилася помилка. Нове ім'я користувача не був збережено.

Очікуваний результат. Сторінка профайл перевантажилася. Нове значення імені користувача збережено.

Серйозність (Severity) - це атрибут, що характеризує вплив дефекту на працездатність додатка.

Пріоритет (Priority) - це атрибут, що вказує на черговість виконання задачі або усунення дефекту. Можна сказати, що це інструмент менеджера по плануванню робіт. Чим вище пріоритет, тим швидше потрібно виправити дефект.

Градація серйозності дефекту (Severity)

- S1 Що Блокує (Blocker) – ця помилка, приводить додаток у неробочий стан, у результаті якого подальша робота з системою, що тестується або її ключовими функціями стає неможлива. Рішення проблеми необхідно для подальшого функціонування системи.
- S2 Критична (Critical) – неправильно працює ключова бізнес логіка, діра в системі безпеки, проблема, що призвела до тимчасового падіння сервера або, що приводить у неробочий стан деяку частину системи, без можливості рішення проблеми, використовуючи інші вхідні крапки. Рішення проблеми необхідно для подальшої роботи з ключовими функціями системи, що тестуються.

- S3 Значна (Major) - частина основний бізнес логіки працює некоректно. Помилка не критична або є можливість для роботи з функцією, що тестується, використовуючи інші вхідні крапки.
- S4 Незначна (Minor) - не порушує бізнес логіку частини додатка, що тестується, очевидна проблема інтерфейсу користувача.
- S5 Тривіальна (Trivial) - не стосується бізнес логіки додатка, погано відтворена проблема, малопомітна по засобах інтерфейсу користувача, проблема сторонніх бібліотек або сервісів, проблема, що не робить ніякого впливу на загальну якість продукту.

Градація пріоритету дефекту (Priority).

- P1 Високий (High). Помилка повинна бути виправлена якнайшвидше, тому що її наявність є критичною для проекту.
- P2 Середній (Medium). Помилка повинна бути виправлена, її наявність не є критичної, але вимагає обов'язкового рішення.
- P3 Низький (Low). Помилка повинна бути виправлена, її наявність не є критичної, і не вимагає термінового рішення.

Порядок виправлення помилок за пріоритетами: High > Medium > Low

Наявність відкритих дефектів P1, P2 і S1, S2, вважається неприйнятним для проекту. Усі подібні ситуації вимагають термінового рішення і йдуть під контроль до менеджерів проекту.

Наявність строго обмеженої кількості відкритих помилок P3 і S3, S4, S5 не є критичним для проекту і допускається у додатку. Кількість же відкритих помилок залежить від розміру проекту і встановлених критеріїв якості.

4 ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальна записка до КР оформлюється згідно ДСТУ 3008-2015 та ДСТУ ГОСТ 2.105-95, має бути виконана українською мовою.

Пояснювальна записка до КР складається з таких розділів:

- титульний лист;
- зміст;
- вступ;
- виконання практичного завдання (основний текст);
- висновки;
- перелік посилань;
- додатки (якщо необхідно).

Усі матеріали, що додаються до розділів, нумеруються та підшиваються (м’яка палітурка). Титульний лист, зміст, вступ, висновки, перелік використаних посилань, додатки не нумеруються, їх назви вважаються заголовками цих розділів.

Розділи та підрозділи повинні мати заголовки. Пункти та підпункти можуть мати заголовки. Заголовки розділів слід розташовувати посередині рядка та друкувати великими літерами без крапок у кінці. Абзацний відступ має бути однаковим в усьому тексті пояснювальної записки.

Приклад титульного аркушу наведено у додатку А.

Зміст розташовують безпосередньо після титульного аркушу, починаючи з нової сторінки. Він має містити усі розділи та підрозділи та номери сторінок початку матеріалу. Традиційно зміст формується шляхом автогенерації.

У вступі необхідно стисло обґрунтувати актуальність обраної теми КР, на основі стислої оцінки сучасного стану досліджуваної проблеми. Рекомендований обсяг вступу – 1 сторінка.

Основний текст пояснювальної записки, як правило, містить опис виконання практичного завдання. Назви підрозділів бажано формулювати так, щоб вони відображали їх зміст. Розділи можуть розбиватися на підрозділи, пункти та підпункти. Кожен пункт та підпункт містить закінчену

інформацію.

Висновки розміщують безпосередньо після основної частини пояснювальної записки з нової сторінки. Рекомендовано почати розділ фразою «Отже, у результаті виконання курсової роботи було…» та перерахувати усі види діяльності щодо виконаної роботи (вивчено, розроблено, спроектовано, реалізовано тощо.). Рекомендований обсяг вступу – 1 сторінка.

Перелік використаних джерел, на які є посилання у відповідних розділах, наводять у кінці роботи, з нової сторінки. Опис посилань оформлюється згідно з ДСТУ ГОСТ 7.1:2006 або ДСТУ 8302:2015.

У додатках розміщують матеріал, який доповнює пояснювальну записку, але включення його до основної частини може змінити упорядкований та логічний виклад роботи, не може бути послідовно розміщений у основній частині пояснювальної записки через великий обсяг або способи відтворення.

Як правило, до додатків включають таблиці, додаткові ілюстрації тощо. Кожний додаток має починатися з нової сторінки, має заголовок, надрукований угорі малими літерами (крім першої великої) симетрично до тексту сторінки. Додатки слід позначати послідовно великими літерами українського алфавіту, окрім літер Ґ, Є, Ї, Й, Ь. Ілюстрації та таблиці, що є в тексті додатку, нумерують у його межах додатку, наприклад, «Рисунок А.1».

Сторінки пояснювальної записки слід нумерувати арабськими цифрами, дотримуючись наскрізної нумерації упродовж усього тексту. Номер сторінки проставляють у правому верхньому куті сторінки без крапки у кінці. Першою сторінкою пояснювальної записки є титульний аркуш. Номер сторінки на ньому не проставляють.

Ілюстрації та таблиці, розміщені на окремих сторінках, включають до загальної нумерації сторінок пояснювальної записки. Перелік посилань та додатки включають до наскрізної нумерації.

Кількість ілюстрацій, які містяться у пояснювальній записці, визначаєтьсяїї змістом. Вона має бути достатньою для забезпечення ясності та конкретності тексту. Основними видами ілюстрацій є рисунки, графіки,

схеми, діаграми, фотографії. Їх слід розміщувати у роботі безпосередньо після тексту, де вони згадуються вперше або на наступній сторінці. Перед рисунком та після нього слід залишати пустий рядок.

На всі ілюстрації мають бути посилання у тексті. У місці, де згадуються ілюстрації, розміщують посилання у вигляді виразу у круглих дужках, наприклад, (рис. 3.1) або використовують звороти типу «як це показано (видно) на (з) рисунку 3.3».

Підпис розміщують під ілюстрацією відцентровано. Він складається зі слова «Рисунок», номера та назви ілюстрації. Номер ілюстрації складається з номера розділу та порядкового номера ілюстрації у цьому розділі, наприклад, «Рисунок 2.1». За потреби під назвою ілюстрації розміщують пояснювальні дані (підрисунковий текст). Їх не рекомендується наводити безпосередньо на рисунку.

Цифровий матеріал, як правило, оформлюють у вигляді таблиць. Таблицю слід розташовувати безпосередньо після тексту, у якому вона згадується вперше, або на наступній сторінці. На всі таблиці слід наводити посилання в тексті пояснювальної записки.

Таблиця має графи (вертикальні колонки) та рядки (горизонтальні строки). Зверху граф розташований заголовок, що складається із заголовків граф та, у разі необхідності, підзаголовків граф. Заголовок відокремлюється лінією від решти таблиці.

Заголовки граф та строк слід писати з прописної літери, а підзаголовки граф – зі строкової літери, якщо вони складають одне речення з заголовком, або з прописної літери, якщо вони мають самостійне значення. В кінці заголовків і підзаголовків крапки не ставляться. Таблиці зліва, справа та знизу, як правило, обмежують лініями.

Таблиці нумеруються арабськими цифрами наскрізною нумерацією. Допускається нумерація в межах розділу, при цьому її номер складається з номера розділу та порядкового номеру таблиці, які розділяються крапками.

При необхідності таблиця може мати назву, яка розміщується над нею. Якщо рядки або графи таблиці виходять за межі формату сторінки, таблицю поділяють на частини, переносячи частину таблиці на наступну

сторінку, повторюючи в кожній частині її головку. При поділі таблиці на частини допускається її головку заміняти відповідно номерами граф чи рядків, нумеруючи їх арабськими цифрами у першій частині таблиці. Слово «Таблиця...» вказують один раз зліва над першою частиною таблиці, над іншими частинами пишуть: «Продовження таблиці . . .» з зазначенням її номера.

Перенесення таблиці на наступну сторінку можливе тільки тоді, коли її не вдається розмістити на одному аркуші.

Якщо після відповідного тексту необхідно розмістити таблицю, але вона виходить за межі цього аркушу та може бути розташована на одному аркушу, то таблицю розміщують на початку наступного аркушу, а вільне місце даного аркушу заповнюють текстом з наступного аркуша.

За потреби у тексті роботи можуть бути наведені переліки. Перед ними ставиться двокрапка. Перед кожною позицією переліку слід ставити дефіс або арабські цифри чи малі літери українського алфавіту з дужкою (перший рівень деталізації). Для подальшої деталізації переліку можна використовувати подвійні дефіси "– –", арабські цифри (після літер), літери (після цифр), номери з крапкою.

Посилання в тексті роботи на джерела слід зазначати порядковим номером за переліком посилань, виділеним квадратними дужками, наприклад, «... у роботах [1–5] ...». При посиланнях на розділи, підрозділи, пункти, підпункти, ілюстрації, таблиці, формули, рівняння, додатки зазначають їх повні номери. При посиланнях слід писати: «... у розділі 4 ...», «... дивися 2.1 ...», «... за 3.3.4 ...», «... відповідно до 2.3.2.1 ...», «... на рисунку 1.2 ... (рис. 2.1)», «... у таблиці 3.2 ... (табл. 1.2)», «... (див. 3.2) ...», «... за формулою (3.1) ...», «... у рівняннях (1.23) – (1.25) ...», «... у Додатку Б».

5 ВАРІАНТИ ЗАВДАНЬ НА КУРСОВУ РОБОТУ

У переліку наведено орієнтовний перелік тем до виконання практичного завдання:

1. Аналіз якості та тестування музичного плеєру.
2. Аналіз якості та тестування гри «Шашки».
3. Аналіз якості та тестування демонстраційної програми «Фрактали».
4. Аналіз якості та тестування демонстраційного додатку сортування масивів.
5. Аналіз якості та тестування програми для роботи з базою даних.
6. Аналіз якості та тестування програми парсингу інформації з інтернет-сторінок.
7. Аналіз якості та тестування програми для автоматизація роботи фітнес-клубу.
8. Аналіз якості та тестування програми «Економічний калькулятор».
9. Аналіз якості та тестування інтерактивної карти промисловості України.
10. Аналіз якості та тестування гри «Судоку».

6 ПОРЯДОК ЗАХИСТУ КУРСОВОЇ РОБОТИ

До захисту КР студентів допускає викладач, який керує курсовим проектуванням. Необхідними умовами допуску до захисту є:

а) наявність своєчасно виконаного практичного завдання;

б) наявність своєчасно сформованої за стандартом пояснювальної записки.

Захист відбувається перед комісією та передбачає доповіді студентів з презентацією/демонстрацією про основні теоретичні висновки та практичні результати власної КР. Орієнтовний час доповіді − 5 хвилин. Після доповіді студенти повинні відповісти на поставлені запитання.

КР оцінюється за наступними критеріями:

- практичне завдання – до 60 балів;
- пояснювальна записка – до 20 балів;
- захист − до 20 балів.

7 ПЕРЕЛІК РЕКОМЕНДОВАНИХ ПОСИЛАНЬ

1. Білас О.Є. Якість програмного забезпечення та тестування: навчальний посібник / О.Є. Білас. – Львів: Видавництво Львівської політехніки, 2011 –216 с.
2. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення: навчальний посібник / А.С. Авраменко, В.С. Авраменко, Г.В. Косенюк. – Черкаси: ЧНУ ім. Б. Хмельницького, 2017. – 284 с.
3. Золотухіна О.А., Негоденко О.В., Резник С.Ю., Разіна С.Я. Якість та тестування інформаційних систем: навчальний посібник / О.А. Золотухіна, О.В. Негоденко, С.Ю. Резник, С.Я. Разіна. − Київ: ННІІТ ДУТ, 2020. –128 с.
4. Смагіна О.О. Якість програмного забезпечення та тестування : навчальний посібник/ О.О. Смагіна, С.О. Переяславська. – Старобільськ: ДЗ «ЛНУ ім. Т. Шевченка», 2021. – 286 с.
5. Дідковська М.В. Тестування: Основні визначення, аксіоми та принципи. Текст лекцій. Частина І / М.В. Дідковська, Ю.О. Тимошенко. – ННК НТУУ «КПІ», 2010 – 62 с.
6. Дідковська М.В. Тестування: Критерії та методи. Текст лекцій. Частина ІІ / М.В. Дідковська. – ННК НТУУ «КПІ», 2010 – 90 с
7. ISTQB Стандартний глосарій термінів, що використовуються у тестуванні програмного забезпечення. – 2014. – 73 с.
8. Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення: навчальний посібник / І.Л. Бородкіна, Г.О. Бородкін – Центр навчальної літератури. − 2018. – 204 с.
9. Бандура В. В. Архітектура та проектування програмного забезпечення : конспект лекцій / В. В. Бандура, Р. І. Храбатин. –

ІваноФранківськ : ІФНТУНГ, 2012. – 240 с.

10. Карпенко М.Ю., Манакова Н.О., Гавриленко І.О. Технології створення програмних продуктів та інформаційних систем: навчальний посібник / М.Ю. Карпенко, Н.О. Манакова, І.О. Гавриленко. − Харків: Нац. універ. міськ. госп. ім. О.М. Бекетова, 2017. − 93 с.

ДОДАТОК А

Зразок титульного аркушу

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ПрАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра економічної кібернетики та інженерії програмного забезпечення

До захисту допущена
Зав. кафедрою _____
д.е.н., доц. Левицький С.І.

КУРСОВА РОБОТА
з дисципліни «Якість програмного забезпечення та тестування»
на тему «Аналіз якості та тестування редактору блок-схем»

Виконав
ст. гр. ІПЗ–118 _____ Костенко А.О.

Науковий керівник
к.т.н., доц. _____ Резніченко Ю.С.

Запоріжжя
2021