МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ

Методичні вказівки
до виконання лабораторних робіт
для студентів спеціальностей 123 «Комп’ютерна інженерія»
та 125 «Кібербезпека»

Київ 2022

УДК 681.3.07
Т36

Укладач Є.Є. Шабала, канд. техн. наук, доцент

Рецензент О.О. Терентьєв д-р техн. наук, професор

Відповідальний за випуск Ю.І. Хлапонін д-р техн. наук, професор

Затверджено на засіданні кафедри кібербезпеки та комп'ютерної інженерії , протокол № 2 від 22 вересня 2022 р.

В авторській редакції.

Тестування програмного забезпечення систем: методичні
Т36 вказівки / уклад. Є.Є. Шабала. –Київ: КНУБА, 2022. – 28с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт.

Призначені для студентів спеціальностями 123 «Комп'ютерна інженерія» та 125 «Кібербезпека» галузі знань 12 «Інформаційні технології».

ЗМІСТ

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1. **Лабораторні заняття** з дисципліні «Тестування програмного забезпечення систем» у студентів спеціальності 123 «Комп'ютерна інженерія», 125 «Кібербезпека» займають 30 годин і охоплюють розділи курсу, що пов'язані з проектуванням програмного забезпечення та методів тестування якості розробки програмного забезпечення.

2. **Мета лабораторних** занять полягає у розгляді та ознайомленні студентів з існуючими способами контролю якості розробки програмного забезпечення з позицій тестування програмного забезпечення.

3. Лабораторні роботи виконуються на лабораторних заняттях з дисципліни. Підготовка до лабораторних занять здійснюється студентами в часи самостійної роботи. Перелік і кількість задач для розв'язання визначається викладачем, який веде лабораторні заняття, відповідно до робочої програми дисципліни. Після виконання кожної роботи студенти складають звіт, котрий захищають. За результатами виконання та захисту роботи виставляються бали за спеціальною шкалою оцінювання, наведеною у робочій програмі. Бали, отримані за окремі роботи, формують загальну суму балів за дисципліну, яка враховується у підсумкову оцінку за модуль та семестр.

Проектування програмних продуктів, як і будь-яких інших складних систем, виконується поетапно з використанням блочно-ієрархічного підходу, який передбачає розробку продукту по частинах з наступною збіркою . На кожному етапі виконуються певні проектні операції, які відповідним чином документуються. Крім того програмний продукт повинен супроводжуватися різного роду програмою документацією (керівництво програміста, користувача, оператора).

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).

Необхідними умовами для тестування є наявність:

- об'єкта тестування, доступного для проведення досліджень;
- виконавця(ів).

Достатніми умовами для тестування є наявність:

- об'єкта тестування, доступного для проведення досліджень;
- виконавця(ів) (залежно від виду діяльності на різних фазах їм може бути як людина, так і машина або комбінація людина + машина);
- плану тестування;

- тест кейсів / тестів;
- звіту, що підтверджує виконання задач і досягнення цілей, по тестуванню об'єкта.

В методичних вказівках до лабораторних робіт з дисципліни „Тестування програмного забезпечення систем” викладені 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).

2. ЗАВДАННЯ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ

Лабораторна робота №1

Технічне завдання. Вибір моделі розробки забезпечення.

Мета: ознайомитися с правилами написання технічного завдання, навчитися застосовувати різні моделі життєвого циклу програмного забезпечення.

Теоретичні відомості

Модель розробки ПЗ (Software Development Model, SDM) - структура, яка систематизує різні види проектної діяльності, їх взаємодію і послідовність в процесі розробки ПЗ.

Вибір тієї або іншої моделі залежить від масштабу і складності проекту, предметної області, доступних ресурсів і безлічі інших чинників. Моделей розробки ПЗ багато, але в загальному випадку класичними можна вважати *водоспадну, v-подібну, ітераційну інкрементальну, спіральну і гнучку*.

Водоспадна модель (waterfall model) зараз представляє швидше історичний інтерес, тому що в сучасних проектах практично непридатна. Вона передбачає одноразове виконання кожної з фаз проекту, які, в свою чергу, суворо слідують один за одним.

V-образна модель є логічним розвитком Водоспадної. Можна помітити, що в загальному випадку як водоспадна, так і vобразна моделі життєвого циклу ПЗ можуть містити один і той же набір стадій, але принципова відмінність полягає в тому, як ця інформація використовується в процесі реалізації проекту.

Ітераційна Інкрементальна модель (iterative model, incremental model) є фундаментальною основою сучасного підходу до розробки ПЗ. Як випливає з назви моделі, їй властива певна подвійність:

- з точки зору життєвого циклу модель є ітераційною, тому що має на увазі багаторазове повторення одних і тих же стадій;
- з точки зору розвитку продукту (збільшення його корисних функцій) модель є інкрементальною.
- Ключовою особливістю даної моделі є розбиття проекту на відносно невеликі проміжки (ітерації), кожен з яких в загальному випадку може включати в себе всі класичні стадії, властиві Водоспадній і v-образній моделям.
- Результатом ітерації є приріст (інкремент) функціональності продукту, виражений в проміжному білді (build).

Спіральна модель (spiral model) являє собою окремий випадок ітераційної інкрементальної моделі, в якому особлива увага приділяється управління ризиками, особливо впливають на організацію процесу розробки проекту і контрольні точки.

Розробка технічного завдання

Технічне завдання являє собою документ, в якому сформульовані основні цілі розробки, вимоги до програмного продукту, визначено терміни та етапи розробки та регламентований процес приймально-здавальних випробувань. У розробці технічного завдання беруть участь як представники замовника, так і представники виконавця. В основі цього документа лежать вихідні вимоги замовника, аналіз передових досягнень техніки, результати виконання науково-дослідних робіт, передпроектних досліджень, наукового прогнозування і т. п.

Порядок розробки технічного завдання

Розробка технічного завдання виконується в наступній послідовності. Насамперед, встановлюють набір виконуваних функцій, а також перелік і характеристики вихідних даних. Потім визначають перелік результатів, їх характеристики і способи подання. Далі уточнюють середовище функціонування програмного забезпечення: конкретну комплектацію і параметри технічних засобів, версію операційної системи і, можливо, версії і параметри іншого встановленого програмного забезпечення, з яким належить взаємодіяти майбутньому програмному продукту.

У випадках, коли розробляється програмне забезпечення збирає і зберігає деяку інформацію або включається в управління будь-яким технічним процесом, необхідно також чітко регламентувати дії програми у разі збоїв обладнання та енергопостачання.

Порядок виконання лабораторної роботи

1. Обрати модель життєвого розробки програмного забезпечення та обґрунтувати свій вибір.
2. Розробити технічне завдання на розробку програмного продукту на вибір студента: веб-сайт, комп’ютерна гра, мобільний додаток, база даних або будь-яке інше програмне забезпечення. Технічне завдання повинне містити наступні розділи:

ВСТУП

1. ПІДСТАВИ ДЛЯ РОЗРОБКИ
   1.1. Документ, на підставі якого ведеться розробка
   1.2. Організація, що затвердила підставу розробки, і дата його затвердження
   1.3. Найменування теми розробки
2. ПРИЗНАЧЕННЯ РОЗРОБКИ
   2.1 Критерії ефективності та якості програми
   2.2 Цілі розробки програми
3. ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик
3.1.1 Склад виконуваних функцій
3.1.2 Організація вхідних і вихідних даних
3.1.3 Тимчасові характеристики і кількість пам 'яті
3.2 Вимоги до надійності
3.2.1 Вимоги до надійного функціонування
3.2.2 Контроль вхідної і вихідної інформації
3.2.3 Час відновлення після відмови
3.3 Умови експлуатації
3.4 Вимоги до складу і параметрів технічних засобів
3.5 Вимоги до мов програмування
3.6 Вимоги до програмних засобів, які використовуються програмою
3.7 Вимоги до програмної документації
4. ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ
5. СТАДІЇ І ЕТАПИ РОЗРОБКИ
6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ
6.1 Види випробувань
6.2 Загальні вимоги до приймання
7. ЕТАПИ ВПРОВАДЖЕННЯ

Залежно від особливостей програми або програмного виробу допускається уточнювати зміст розділів, вводити нові розділи або об'єднувати окремі з них. При необхідності допускається в технічне завдання включати додатки.

4. Оформити звіт з виконання лабораторної роботи та захистити.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

1. Наведіть етапи розробки програмного забезпечення.
3. Моделі розробки програмного забезпечення.
4. Взаємодія тестувальників з іншими учасниками розробки програмного забезпечення.
5. Назвіть основні розділи технічного завдання.

Лабораторна робота №2
Етапи розробки програмного забезпечення

Мета: навчитися створювати формальні моделі і на їх основі визначати специфікації розроблюваного програмного забезпечення.

Теоретичні відомості

Розробка специфікацій

Розробка програмного забезпечення починається з аналізу вимог до нього. У результаті аналізу отримують специфікації розроблюваного програмного забезпечення, будують загальну модель його взаємодії з користувачем або іншими програмами і конкретизують його основні функції. При структурному підході до програмування на етапі аналізу і визначення специфікацій розробляють три типи моделей: моделі функцій, моделі даних і моделі потоків даних.

Оскільки різні моделі описують проектована програмне забезпечення з різних сторін, рекомендується використовувати відразу кілька моделей, що розробляються у вигляді діаграм, і пояснювати їх текстовими описами, словниками і т. п.

Структурний аналіз передбачає використання наступних видів моделей:

- діаграм потоків даних (DFD - Data Flow Diagrams), описують взаємодію джерел і споживачів інформації через процеси, які повинні бути реалізовані в системі;
- діаграм «сутність-зв'язок» (ERD - Entity-Relationship Diagrams), що описують бази даних розроблюваної системи;
- діаграм переходів станів (STD - State Transition Diagrams), що характеризують поведінку системи в часу;
- функціональних діаграм (методика SADT);
- специфікацій процесів;
- словника термінів.

Специфікації процесів зазвичай представляють у вигляді короткого текстового опису, схем алгоритмів, псевдокод, Flow-форм або діаграм Насс-Шнейдермана.

Словник термінів являє собою короткий опис основних понять, що використовуються при складанні специфікацій. Він повинен включати визначення основних понять предметної області, опис структур елементів даних, їх типів і форматів, а також усіх скорочень і умовних позначень

Діаграми переходів станів

За допомогою діаграм переходів станів можна моделювати подальше функціонування системи на основі її попереднього та поточного функціонування.

Модельована система в будь-який заданий момент часу знаходиться точно в одному з кінцевого безлічі станів. З плином часу вона може змінити свій стан, при цьому переходи між станами повинні бути точно визначені

Функціональні діаграми

Функціональні діаграми відображають взаємозв'яок функцій розроблюваного програмного забезпечення. Вони створюються на ранніх етапах проектування систем, для того щоб допомогти проектувальнику виявити основні функції і складові частини проектованої системи і, по можливості, виявити і усунути істотні помилки. Для створення функціональних діаграм пропонується використовувати методологію SADT.

Діаграми потоків даних

Для опису потоків інформації в системі застосовуються діаграми потоків даних (DFD - Data flow diagrams). DFD дозволяє описати необхідну поведінку системи у вигляді сукупності процесів, що взаємодіють за допомогою зв'язуючих їх потоків даних. DFD показує, як кожен з процесів перетворює свої вхідні потоки даних у вихідні потоки даних і яким чином процеси взаємодіють між собою.

Діаграми «сутність-зв'язок»

Діаграма сутність-зв'язок - інструмент розробки моделей даних, що забезпечує стандартний спосіб визначення даних і відношень між ними. Вона включає сутності та взаємозв'яок, що відображають основні бізнес-правила предметної області. Така діаграма не надто деталізована, в неї включаються основні сутності і зв'язки між ними, які задовольняють вимогам, що пред'являються до інформаційної системи.

Порядок виконання лабораторної роботи

1. На основі технічного завдання з лабораторної роботи № 1 виконати аналіз функціональних та експлуатаційних вимог до програмного продукту.

2. Визначити основні технічні рішення (вибір мови програмування, структура програмного продукту, склад функцій ПП, режими функціонування) і занести результати в документ, званий «Проектом»

3. Розробити діаграми потоків даних для розв'язуваної завдання.

4. Розробити діаграми «сутність-зв'язок», якщо програмний продукт містить базу даних.

5. Розробити функціональні діаграми.
6. Розробити діаграми переходів станів.
7. Розробити специфікації процесів.
8. Додати словник термінів.
9. Оформити результати, використовуючи MS Office у вигляді проекту.
10. Здати і захистити роботу.

Звіт з лабораторної роботи повинен складатися з:

1. Постановки завдання.
2. Документу «Проект», що містить:
   - специфікації процесів;
   - всі отримані діаграми;
   - словник термінів.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

1. Назвіть етапи розробки програмного забезпечення.
2. З чого складається структурний аналіз розробки програмного забезпечення?
3. У чому полягає постановка задачі та передпроектні дослідження?
4. Назвіть функціональні та експлуатаційні вимоги до програмному продукту.
5. Перелічіть складові проекту.

Лабораторна робота 3.
Тестова документація й артефакти тестування

Мета: ознайомлення та набуття практичних навичок роботи з тестовою документацією та складання тестової документації відповідно до вимог.

Теоретичні відомості

Розглянемо основну тестову документацію.

Тест (test) - набір з одного або декількох тест-кейсів.

Тест-кейс (test case) - набір вхідних даних, умов виконання та очікуваних результатів, розроблений з метою перевірки тієї чи іншої властивості або поведінки програмного засобу.

Життєвий цикл тест-кейсу наведений на Рис.1.

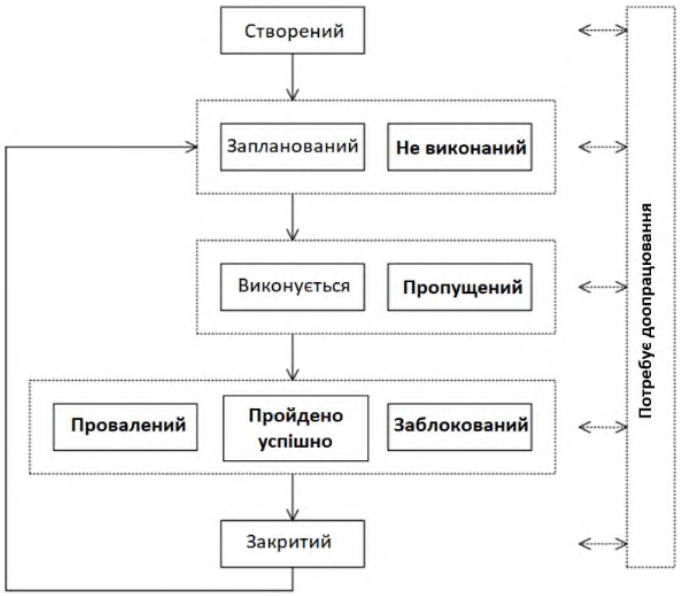

Рис. 1 Життєвий цикл тест-кейсу

Під тест-кейсом також може розумітися відповідний документ, який представляє формальну запис тест-кейса. Структура даного артефакту полягає в наступному: що треба зробити (Action) – очікуваний результат (Expected result) – фактичний результат (Test result).

Безпосередньо сам тестовий випадок складається з 3 частин:

1. PreConditions (передумови) – або список кроків, які призводять систему в стан, придатний для тестування, або список перевірок умов того, що система вже знаходитися в необхідному стані.
2. Test Case Description (опис тестового випадку) – список дій, за допомогою яких здійснюється основна перевірка функціоналу (після якої і звіряється фактичний результат з очікуваним).
3. PostConditions (післяумови) – список дій, які повертають систему в початковий стан.

Атрибути, які містить тест-кейс наведений на Рис.2.

Рис.2 Атрибути тест-кейсу

Тест-план (Test Plan, план тестування) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об’єктів, стратегій, розкладів, критеріїв початку та закінчення тестування об’єкту, що тестується – до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Як ми можемо бачити, тест-план є важливою складовою будь-якого грамотно організованого процесу тестування, оскільки містить в собі всю необхідну інформацію, що описує даний процес. Але в більшості випадків, з якими нам доведеться зіткнутися, тест-план буде грати більш формальну роль, але, все ж, його присутність має багато переваг. Наприклад:

- Можливість виставлення пріоритетів для задач з тестування.
- Побудова стратегії тестування, погодженої з усією командою.
- Можливість вести облік всіх необхідних ресурсів, як технічних, так і людських.
- Планування використання ресурсів на тестування.
- Прорахунок ризиків, можливих при проведенні тестування.

Залежно від конкретизації завдань, що описуються, тест-план може мати два рівня деталізації: майстер тест-план та детальний тест-план.

Детальний тест-план містить у собі завдання тестування для кожної команди, для кожного релізу або ітерації проекту. Створюється детальний тест-план або для

декомпозованої частини проекту, або для невеликих проектів. Він може складатися з:

- Перелік областей тестування з пріоритетами.
- Стратегії тестування.
- Переліку можливих ризиків.
- Перелік необхідних ресурсів.
- Плану виконання проекту.

Майстер тест-план у свою чергу створюється або для організації процесу тестування між декількома командами, які тестують один проект, але мають різні завдання, або для проекту, який складається з безлічі ітерацій, які пов’язує якась загальна інформація, повторення якої в кожному релізі займає надто багато часу.
У майстер тест-план входить:

- Загальна інформація про проект (посилання на документацію, баг-трекери, і т.д.)
- Положення, що описують процес тестування, заклади дефектів і т.д.

Критерії готовності продукту до випуску.
Для полегшення життя тестувальникам, існують кілька шаблонів тест-планів (**IEEE, RUP**).

Порядок виконання роботи

1. Розробити тест-план згідно шаблону IEEE або RUP. Приклад надається викладачем.
2. Створити декілька тест-кейсів для обраної теми згідно Рис. 2.
3. Оформити звіт з виконання лабораторної роботи та захистити.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання
1. Що таке тест, тест-кейс?
2. Життєвий цикл тест-кейсу
3. Які атрибути містить тест-кейс?
4. Що таке тест-план та його компоненти?

Лабораторна робота № 4
Розробка та проведення тесту користувацького інтерфейсу

Мета – ознайомитися з особливостями складання звіту про дефекти. Набути навичок проведення автоматизованого GUI тестування та UI тестування інтерактивного прототипу веб-додатку. Освоїти використання схеми зв'язку проблем в розробці програмного забезпечення.

Теоретичні відомості

Тестування графічного інтерфейсу - це тип тестування програмного забезпечення, який перевіряє графічний інтерфейс користувача програмного забезпечення. Мета тестування графічного інтерфейсу користувача (GUI) полягає у забезпеченні функціональних можливостей програмних додатків відповідно до специфікацій шляхом перевірки екранів та елементів керування, таких як меню, кнопки, піктограми тощо. GUI - це те, що бачить користувач.

Користувач не бачить вихідний код. Інтерфейс видно користувачеві. Особливо увага приділяється структурі дизайну, зображенням того, чи працюють вони належним чином чи ні.

Тестування графічного інтерфейсу в основному передбачає:

- Тестування розміру, положення, ширини, висоти елементів.
- Тестування повідомлень про помилки, які відображаються.
- Тестування різних розділів екрана.
- Тестування шрифту, читабельний він чи ні.
- Тестування екрану в різних роздільних здатностях за допомогою збільшення та зменшення масштабу, як 640 x 480, 600x800 тощо.
- Тестування вирівнювання текстів та інших елементів, таких як піктограми, кнопки тощо, знаходяться у належному місці чи ні.
- Тестування кольорів шрифтів.
- Тестування кольорів повідомлень про помилки, попереджувальних повідомлень.
- Перевірка того, чи має зображення хорошу чіткість чи ні.
- Тестування вирівнювання зображень.
- Перевірка правопису.
- Перевірка привабливості інтерфейсу.
- Тестування смуг прокрутки відповідно до розміру сторінки, якщо така є.
- Тестування відключених полів, якщо такі є.
- Тестування розміру зображень.
- Перевірка заголовків, чи правильно він вирівняний чи ні.

- Тестування кольору гіперпосилання.

Таблиця 1 - Етапи тестування зручності використання інтерфейсу користувача

Етап	Час проведення	Мета та дії
Дослідницьке тестування	Проводиться після формулювання вимог до системи та розроблення прототипу інтерфейсу	Основна мета - провести високорівневе дослідження інтерфейсу і виявити, чи довзоляє він з достатньою ефективністю розв'язувати задачі користувача;
Оцінювальне тестування	Проводиться після розробки низькорівневих вимог та деталізованого прототипу ІК	Поглиблює дослідницьке тестування та має ту ж мету; проводяться кількісні вимірювання характеристик ІК: вимірюється кількість звертань до системи допомоги по відношенню до кількості виконаних операцій, кількість помилкових операцій, час усунення наслідків помилкових операцій і т.і.
Валідаційне тестування	Проводиться ближче до етапу завершення розробки	Проводиться аналіз відповідності ІК стандартам, які регламентують питання зручності інтерфейсу, проводиться загальне тестування всіх компонентів ІК (програмна реалізація, система допомоги, керівництво користувача) з позиції кінцевого користувача, а також перевіряється відсутність дефектів зручності використання ІК, виявлених на попередніх етапах
Порівняльне тестування	Може проводитись на будь-якому етапі розробки ІК	Порівнюються два або більше варіантів реалізації ІК

Інструменти тестування графічного інтерфейсу

Нижче наведено список популярних засобів тестування графічного інтерфейсу :

- Ranorex
- Selenium
- QTP
- SilkTest
- TestComplete
- Squish GUI Tester

Завдання:

1. Створити декілька тест-кейсів для тестування графічного інтерфейсу обраного дадатку.
2. Провести тестування, обравши будь-який засіб тестування, який запропоновано в даній лабораторній роботі (або обрати самостійно).
3. Оформити звіт з виконання лабораторної роботи та захистити.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні запитання

1. В чому суть UI тестування?
2. Наведіть список необхідних перевірок тестування GUI.
3. Перелічіть інструментальні засоби GUI тестування.

Лабораторна робота 5. Основний інструментарій для аналізу якості при роботі з програмним забезпеченням

Теоретичні відомості

Якість програмного забезпечення (Software quality) асоціація IEEE визначає як ступінь відповідності програмного забезпечення встановленій комбінації властивостей. У Міжнародному стандарті якості ISO 9000:2007 це поняття визначається як сукупність характеристик ПЗ, які забезпечують встановлені та очікувані вимоги .

Програмне забезпечення Confluence, розроблене Atlassian, є ефективним програмним забезпеченням для спільної роботи команд що забезпечує загальну платформу для команд для спільної роботи та ефективного обміну інформацією. Atlassian JIRA - це програмне забезпечення для відстеження випусків та проектів для планування, відстеження та управління проектами. JIRA в основному використовується гнучкими командами розробників для налаштування робочих процесів, співпраці команд та випуску програмного забезпечення.

Завдання:

1. Перейти на сайт JIRA (https://www.atlassian.com/software/jira).
2. Вибрати варіант «Завантажити JIRA безкоштовно» («get it free»).
3. Зареєструватися на реальний email з яким-небудь доменом.
4. Підтвердити за допомогою email свій акаунт.
5. Вибрати опцію створення нового проекту та вибрати тип проекту (Scrum), натиснути «Ок».
6. Задати ім’я наприклад, Test Project (абревіатура проекту створюється автоматом). Проект буде додано до списку наявних проектів, тепер до нього можна додавати завдання і т. д.

По завершенню цих завдань вийде такий результат, як представлено на Рис.3

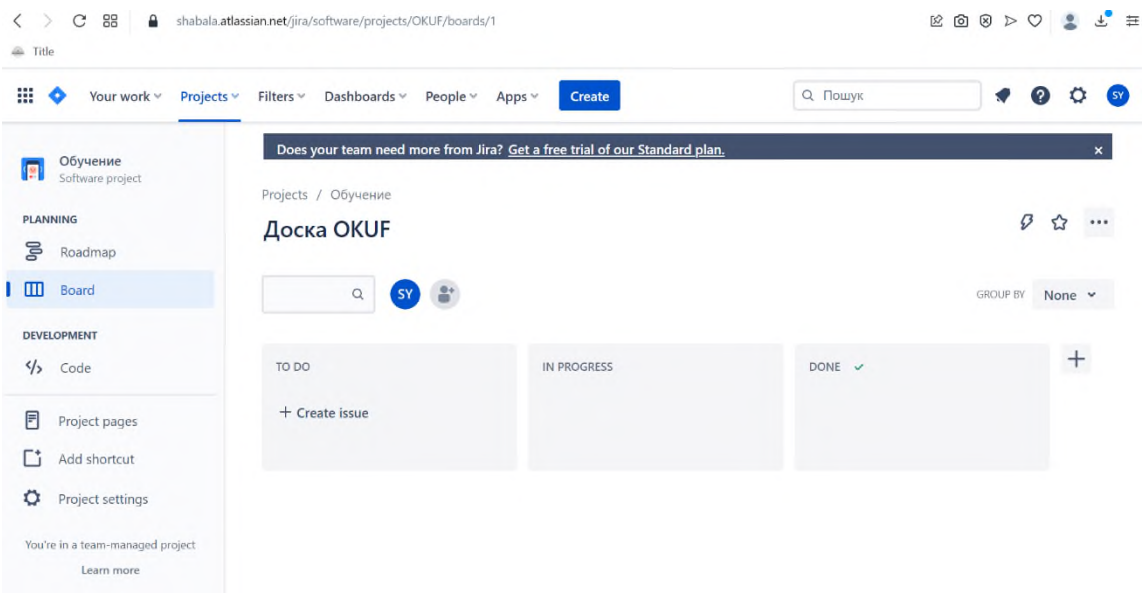

Рис.3 Вікно створеного проекту

7. Створити в цьому проекті декілька задач (issue):
– вибрати проект, тип задачі;
– сформулювати sammary (короткий опис) та опис (description), при цьому можна форматувати текст за необхідності та додавати додатки (можна прикріпити скрін, відео і т. д.);
– дати опис середовища (оточення, environment) – ті умови, за яких відтворюється цей баг, наприклад;
– натиснути Create (створити). При цьому задача додається до проекту та її можна переглянути, редагувати, видалити і т. д.
8. Створити спринт, указавши дату його завершення та додавши до нього задачі, створені в попередньому пункті (для цього послідовно необхідно натиснути Create Board –> Create Sprint –> Start Spring і виконати необхідні настроювання).

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

1. Навіщо використовувати програмне забезпечення JIRA?
2. Як називається спеціальна стартова сторінка в JIRA?
3. Що відображає пункт меню «Projects» в JIRA?
4. Що відображається в JIRA, якщо зайти під обліковим записом?
5. Чи можна імпортувати проект, якщо він є десь в іншій системі?

Лабораторна робота № 6
Пошук та документування дефектів

Мета: набуття практичних навичок застосування різних технік ручного тестування та опису знайдених дефектів у вигляді баг-репортів.

Теоретичні відомості

Дефект (баг bug, помилка) – відхилення фактичного результату від очікувань користувача, сформованих на основі вимог, специфікацій, іншої документації чи досвіду та здорового глузду.

Дефекти, виявлені тестувальником, мають бути коректними та зрозуміло описані, щоб розробник зміг відтворити цей дефект і усунути його. Опис кожного дефекту зберігається у спеціалізованій – багтрекінговій – системі (наприклад, JIRA, Bugzilla, Mantis, Redmine та ін) або попередньо створеному у програмному середовищі Microsoft Excel файлі.

Опис дефекту включає такі обов'язкові поля:

1. Headline – назва дефекту.
2. Severity - ступінь критичності (важливість дефекту).
3. Description – алгоритм відтворення.
4. Result – фактичний результат.
5. Expected result – очікуваний результат.
6. Attachment – прикріплені файли (додаток).

У багтрекінгових системах для кожного дефекту автоматично генерується його унікальний номер, у разі використання Microsoft Excel номер дефекту необхідно надавати вручну. Вимогу специфікації, що порушує виявлений дефект, можна додатково винести у примітку. Додатково в описі дефекту може бути вказана Priority – ступінь терміновості виправлення дефекту розробником.

Розглянемо докладно кожну категорію опису дефекту.

Headline (назва дефекту). Мета складання заголовка дефекту – надати коротку і водночас зрозумілу інформацію у тому, де, що у внаслідок чого сталося. Характеристиками якісного заголовка є стисліть, інформативність, точна ідентифікація проблеми.

Заголовок дефекту повинен відповідати на три запитання:

1. Де? Де інтерфейсу користувача перебуває проблема. У цій частині заголовка слід додатково вказати особливості тесту, якщо це допоможе розібратися в проблемі (версія операційної системи, браузера, сторонніх додатків, які мають відношення до тестованого програмного засобу).

2. Що? Що відбувається або не відбувається відповідно до специфікації або уявлення про нормальну роботу програмного продукту. При цьому необхідно вказувати на наявність проблеми, а не її зміст (його вказують в описі). Якщо зміст проблеми варіюється, всі відомі варіанти зазначаються в описі.

3. Коли (за яких умов)? У який момент роботи програми чи настання якої події проблема проявляється.

Приклад: у програмі є діалог "Перетворити дані" з кнопкою "Перетворити". При натисканні цієї кнопки з'являється повідомлення про помилку "Помилка 315". Заголовок дефекту за описаною методикою складається так:

Де?: Діалог "Перетворити дані".

Що?: Відобразиться повідомлення про помилку.

Коли?: Натиснувши кнопку «Перетворити».

Підсумковий заголовок буде мати такий вигляд: Діалог "Перетворити дані" показує повідомлення про помилку при натисканні кнопки "Перетворити".

Заберемо зайві слова, додамо код помилки для зручності пошуку: Діалог "Перетворити дані": повідомлення про помилку 315 при натисканні кнопки "Перетворити".

Severity (ступінь критичності). Ступінь критичності (серйозності, важливості) показує ступінь шкоди, що завдається проекту існуванням дефекту.

У загальному випадку виділяють такі градації критичності дефектів (Таблиця 2): Critical (критичний), Major (значний), Average (середньої значимості), Minor (незначний), Enhancement ( ).

Таблиця 2 – Критичність дефектів

Severity	**Опис**	**Приклад**
Critical (критичний)	Функціональна помилка, яка блокує роботу частини функціоналу або всього додатка. Функціональна помилка, яка порушує ключову (з погляду кінцевого користувача або бізнесу замовника) функціональність програми.	Заблоковано вкладку категорія меню). Неправильно підраховується підсумкова сума під час введення знижковий промокод. Розкривається конфіденційна інформація.
Major (значний)	Функціональна помилка, яка порушує нормальну роботу програми, але не блокує роботу частини функціоналу загалом.	Неможливо завантажити відеофайли на персональній сторінці. Не працює система e-mail нотифікації. Не працює інтеграція з соціальними мережами.

		Необхідно перезапускати додаток під час виконання типових сценаріїв роботи
Average (середньої значимості)	Не дуже важлива функціональна помилка. Критичні дефекти GUI.	Не працює сортування. "Поїхав" текст за межі вікна
Minor (незначний)	Рідко зустрічаються незначні функціональні дефекти. 90% дефектів GUI.	Введені необов'язкові для заповнення поля, яких немає в специфікації. Граматичні, пунктуаційні помилки.
Enhancement (пропозиція щодо покращення)	Функціональні пропозиції, поради щодо покращення дизайну (оформлення), навігації та ін. Такий дефект є необов'язковим для виправлення.	Додати кнопку «Вгору» на довгі форми. Збільшити розмір шрифту.

Description (алгоритм відтворення). Мета складання алгоритму відтворення дефекту – послідовно описати кроки для повторення дефекту. Description має бути оформлений у вигляді списку переліку дій:

1. Крок #1.
2. Крок #2.

…

n. Крок #n

У випадку, якщо для відтворення дефекту потрібен ряд початкових умов (наприклад, має бути створено певний набір документів, користувач або група користувачів з особливими правами), то ці передумови повинні бути винесені на початок опису:

Передумови.

1. Крок #1.
2. Крок #2.

…

n. Крок #n.

При складанні Description необхідно слідувати наведеним нижче рекомендацій.

Description – це чіткий алгоритм, у якому вітаються короткі, зрозумілі фрази та нумерація.

Не можна використовувати особисті пропозиції формату «Я думаю, що так буде краще», «Я зайшов на сторінку…» і т.д.

Можна використовувати спеціальні символи "+", "=", "<>" які допоможуть зробити подібність навігації: File > Open, DOC+XLS. Однак не рекомендується писати кроки в рядок через символи переходу: це ускладнює сприйняття дефекту. Слід вказувати специфічні умови відтворення дефекту, якщо такі є. Наприклад, під яким користувачем ви працюєте (якщо це важливо).

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.

Result (фактичний результат). Мета написання Result – чітко описати отриманий результат.

Expected result (очікуваний результат). Мета написання Expected result – навести аргументи розробникам, як саме має працювати програма.

У Expected result має бути чітке обґрунтування, чому саме так має працювати додаток. Найкраще, якщо в ньому наведено посилання на конкретний пункт специфікації.

Якщо в Expected result наводиться посилання на специфікацію, сам тестувальник додатково цитує текст специфікації, щоб скоротити час розробника на аналіз документа та однозначно вказати на спосіб вирішення проблеми.

Якщо функція працює, але некоректно, то в Expected Result обов'язково має бути опис того, як вона має працювати коректно.

Якщо зроблено помилку в написі або інтерфейсі проекту, необхідно грамотно та повністю вказати, як вона має бути виправлена – написати напис без помилки або описати потрібні зміни інтерфейсу. Expected Result завжди слід заповнювати. Не варто покладатися на очевидність уявлень про правильну поведінку програми.

Текст Expected result рекомендується писати закінченими повними безособовими пропозиціями.

Attachment (додатки). Attachment – це прикріплений до дефекту файл, додатковий опис: скріншот, файли, необхідні для відтворення дефекти, логи програми, відео помилки і т.д. Attachment є допоміжним засобом передачі про проблему. Для всіх GUI Дефектів Attachment обов'язкові.

Якщо до дефекту прикріплюється файл, про це обов'язково має бути вказано в описі дефекту ("See the file/screenshot/log/video attached"). Прикріплений файл не повинен бути надто великим за розміром (особливо це стосується відео: до 10 мегабайт), а також має ставитись саме до описаної помилки (наприклад, з лога програми варто скопіювати в файл, що прикріплюється тільки дані про помилку). Ім'я файлу скріншота рекомендується робити числовим, нейтральним – 1.png, 25.png і т.п.

Скріншот повинен містити такі елементи: сама помилка, виділення місця помилки, стрілка до прямокутника, опис помилки: «Спостерігається результат» та/або «Очікуваний результат». Текст на скріншоті також необхідний виділити: обвести в прямокутник і набрати шрифтом, що помітно відрізняється шрифт програми. Якісно підготовлений скріншот має давати можливість зрозуміти зміст дефекту без необхідності читати його опис.

Порядок виконання роботи

1. Провести тестування обраного об’єкта з використанням різних технік, знайти якомога більше багів.
2. Проаналізувати тестові випадки.
3. Усі знайдені баги занести в JIRA у свій акаунт.
4.При написанні звіту навести скріншоти створених у JIRA баг-репортів.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

Що таке дефект?

Перелічити переваги і недоліки тестування методом «білого ящика».

Перелічити переваги і недоліки тестування методом «чорного ящика».

Навіщо потрібен Attachment при опісі дефекту?

Лабораторна робота № 7
Документування результатів тестування

Мета: Скласти підсумковий звіт про результати тестування обраного об’єкту тетування.

Теоретичні відомості

Підсумковий звіт про тестування - це документ, який містить короткий опис усіх випробувальних заходів та остаточні результати випробувань проекту тестування. Підсумковий звіт про тестування - це оцінка того, наскільки якісно проведено тестування. На основі звіту про тестування зацікавлені сторони можуть оцінити якість протестованого продукту та прийняти рішення щодо випуску програмного забезпечення.

Що містить звіт про тестування?

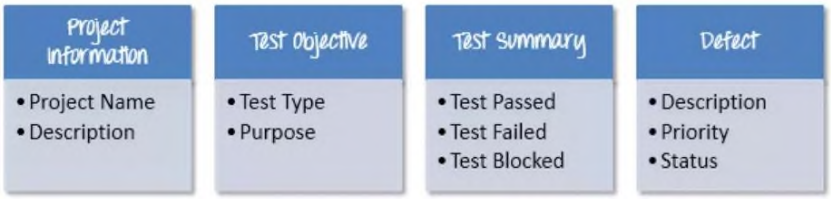

Рис. 4 Компоненти звіту тестування

Інформація про проект

Вся інформація про проект, така як назва проекту, назва продукту та версія, повинна бути описана у звіті про випробування. Наприклад, інформація про проект буде такою:

Project Overview				
PROJECT BASIC INFORMATION				
Project Name	Guru99 Bank			
Name of product (Product Number)	Banking website www.demo.guru99.com			
Product Description	The banking website			
Project Description	<Mision of project> Conduct testing to verify the quality of this website Ensure the website is released without any defects <Project's output product> Test Summary Report & Evaluation			
	Project Type	Testing/Verification		
Project Duration	Start date	10/1/2013	End date	10/31/2013

Рис. 5 Інформація про проект

Мета тесту

Звіт про випробування повинен містити цілі кожного етапу тестування, такі як модульний тест, тест продуктивності, системний тест ... тощо.

Підсумок тесту

Цей розділ включає підсумок випробувальної діяльності загалом і включає:

- Кількість виконаних тестів
- Кількість тестових кейсів «проходить»
- Кількість тестових випадків «не вдається»
- Відсоток прохідності
- Відсоток відмов
- Коментарі

Ця інформація повинна відображатися візуально за допомогою кольорового індикатора , графіку та виділеної таблиці .

Дефект

Однією з найважливіших відомостей у звіті про випробування є дефекти. Звіт повинен містити наступну інформацію:

- Загальна кількість помилок
- Статус помилок (відкрито, закрито,….)
- Кількість помилок, відкритих, вирішених, закритих
- Розподіл за ступенем тяжкості та пріоритетом

Як і підсумок тесту, ви можете включити деякі прості показники, такі як щільність дефектів,% виправлених дефектів.

Наприклад: Щільність дефектів - 20 дефектів / 1000 рядків середнього коду

Загалом виправлено 90% дефектів

Ви можете представити дані як наступний графік:

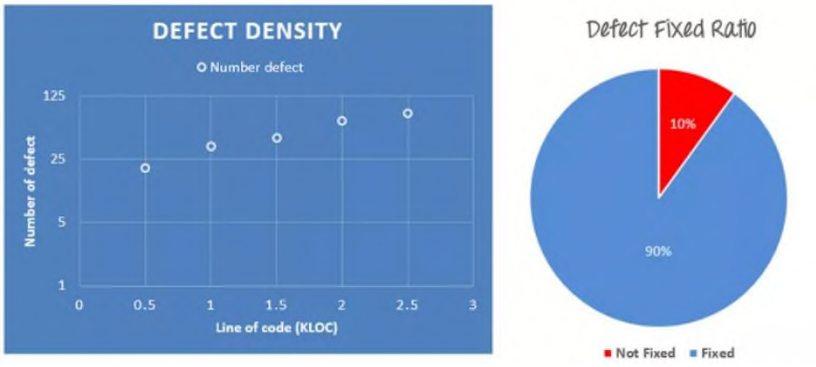

Рис. 6 Представлення сумарного звіту про дефекти

Завдання

1. Розробити підсумковий тест про результати тестування обраного об’єкту.

Захист звіту з лабораторної роботи полягає в пред'явленні викладачеві отриманих результатів (на екрані монітора), демонстрації отриманих навичок і відповідях на питання викладача.

Контрольні питання

1. Яка структура підсумкового звіту про результати тестування?
2. Мета розробки підсумкового звіту про результати тестування.
3. Що міститься у розділі «Інформація про проект»?

3. СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Сучасні технології автоматизованого проектування і верифікації програм: Конспект лекцій [Електронний ресурс] : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення», спеціалізації «Інженерія програмного забезпечення комп'ютерних систем» / КПІ ім. Ігоря Сікорського; уклад.: Я. Ю. Дорогий, О. О. Дорога-Іванюк. – Електронні текстові дані (1 файл: 3,9 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 89 с.
2. Сучасні технології автоматизованого проектування та верифікації програм. Тестування програмного забезпечення. Методичні вказівки до виконання лабораторних робіт. [Електронне видання] / Уклад.: Я.Ю. Дорогий, О.О. Дорога-Іванюк. – К.: НТУУ «КПІ», 2020. – 87 с.
3. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284 с.
4. Говорущенко Т.О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення : монографія / Говорущенко Т. О. Хмельницький : ХНУ, 2017. 310 с.
5. Тестування інтерфейсу користувача. Електронна підтримка та сучасні інформаційні технології у інтерфейсах користувача Електронний ресурс – [Режим доступу: https://msn.khnu.km.ua/pluginfile.php/278098/mod_resource/content/3/Лекція7.pdf
6. Повне керівництво з тестування графічного інтерфейсу: Підручник з тестування інтерфейсу користувача Електронний ресурс – [Режим доступу: https://uk.myservername.com/gui-testing-tutorial
7. Підсумковий звіт про тестові звіти: Навчіться на прикладі та шаблоні Електронний ресурс – [Режим доступу: https://uk.csstricks.net/8223011-test-summary-reports-tutorial-learn-with-example-and-template

Навчально-методичне видання

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ

Методичні вказівки
до виконання лабораторних робіт
для студентів спеціальностей 123 «Комп'ютерна інженерія»
та 125 «Кібербезпека»

Укладач **Шабала** Євгенія Євгенівна

Комп'ютерне верстання *М.М. Влаенко*

Підписано до друку 23.09.2022 Формат 60 х 84 1/ 16
Ум. друк. арк. 1,63. Обл.-вид. арк. 0,96.
Електронний документ. Вид № 59/ІІІ-17.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Повітрофлотський проспект, 31, Київ, Україна, 03680

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.