МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ЗАСОБИ КЕРУВАННЯ ЯКІСТЮ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методичні вказівки та завдання
до виконання лабораторних робіт №1-7
для підготовки здобувачів другого магістерського рівня вищої освіти
спеціальності 121 «Інженерія програмного забезпечення»

Київ 2024

УДК 004.052
З

Укладачі: Т. А. Гончаренко, кандидат технічних наук, доцент
О. О. Маційовський, асистент

Рецензент О. Л. Соловей, канд техн. наук, доцент

Відповідальна за випуск Т. А. Гончаренко, канд. техн. наук, доцент

Затверджено на засіданні кафедри інформаційних технологій, протокол №9 від 27 березня 2024 року.

В авторській редакції.

З **Засоби керування якістю процесу розробки програмного забезпечення**: методичні вказівки та завдання до виконання лабораторних робіт/ уклад.: Гончаренко Т. А. та Маційовський О. О. – Київ: КНУБА, 2024. – 19 с.

Містять зміст, порядок оформлення, вказівки та завдання до виконання лабораторних робіт.

Призначено для здобувачів 12 галузі “Інформаційні технології” спеціальності 121 «Інженерія програмного забезпечення».

ЗМІСТ

Загальні положення

Методичні вказівки до виконання лабораторних робіт з дисципліни “Засоби керування якістю процесу розробки програмного забезпечення (ПЗ)” призначені для проведення занять з основних розділів навчальної дисципліни у відповідності до навчального плану підготовки бакалаврів спеціальності 121 «Інженерія програмного забезпечення».

Управління якістю процесів розробки програмного забезпечення в сучасному інформаційному середовищі є однією з найважливіших складових успіху будь-якого програмного проекту. Підтримка високої якості програмного забезпечення вимагає систематичного підходу та застосування передових методів управління та контролю.

Цикл розробки програмного забезпечення включає в себе різноманітні етапи, від аналізу вимог та проектування до реалізації та тестування. В кожному з цих етапів важливо забезпечити високу якість продукту та процесу розробки.

Мета лабораторних робіт з управління якістю програмного забезпечення - це дати студентам практичний досвід роботи з ключовими аспектами управління якістю. Кожна лабораторна робота спрямована на конкретний аспект управління якістю, починаючи від аналізу коду та тестування і закінчуючи аудитом та розробкою рекомендацій.

При виконанні лабораторних робіт студенти матимуть можливість ознайомитися з методами та інструментами управління якістю, розвинути практичні навички та вивчити кращі практики в цій сфері. Крім теоретичних знань, студенти отримають практичний досвід роботи з реальними проектами, що підготує їх до викликів сучасної індустрії програмного забезпечення.

Лабораторна робота №1. Аналіз якості вихідного коду.

Теоретичні відомості

Аналіз якості вихідного коду - це процес, що включає в себе ретельний огляд програмного коду з метою визначення його відповідності стандартам та кращим практикам програмування. Основні аспекти аналізу включають перевірку читабельності коду, ефективності виконання, безпеки та масштабованості.

Під час аналізу використовуються різні інструменти, такі як статичні аналізатори коду, які автоматично перевіряють код на наявність помилок та порушення стандартів. Також може бути використана ручна рецензія коду, коли інші розробники переглядають код з точки зору його якості та читабельності.

Приклад виконання лабораторної роботи

Припустимо, ми аналізуємо вихідний код веб-додатку із використанням мови програмування Python та фреймворку Django.

Спочатку робимо аналіз структури коду. Під час аналізу, ми виявили, що код складається з моделей Django, представлень, URL-маршрутів та шаблонів. Кожен модуль має чітко визначену відповідальність. Далі перевірка читабельності та стилю коду. Використовуючи інструменти, такі як PEP 8, для перевірки читабельності та відповідності стилю коду, були виправлені невідповідності стандартам.

За допомогою інструментів профілювання ми визначимо найбільш витратні за часом операції, та оптимізуємо їх. Наступним кроком буде виявлення можливих помилок та недоліків. Під час аналізу було виявлено недоліки у роботі з базою даних та потребу удосконалення системи авторизації

Задачі для розв'язання

Оберіть мову програмування та завантажте код для аналізу.

1. Вивчіть стандарт кодування для обраної мови (наприклад, PEP 8 для Python).
2. Використовуючи інструменти аналізу коду, перевірте відповідність коду стандартам якості.
3. Зробіть записи про виявлені недоліки та рекомендації щодо покращення.

Контрольні запитання

1) Які критерії визначають якість вихідного коду програмного забезпечення?
2) Які інструменти аналізу вихідного коду можна використовувати для оцінки його якості?
3) Які принципи Clean Code можна застосувати для покращення якості вихідного коду?

4) Як виявити потенційні проблеми в коді, такі як антипатерни та запахи коду?
5) Які аспекти кодового стилю слід враховувати під час аналізу якості вихідного коду?
6) Як можна оцінити читабельність вихідного коду та його розуміння для інших розробників?
7) Як визначити ефективність та продуктивність вихідного коду під час аналізу його якості?
8) Які вимоги до безпеки можна врахувати під час аналізу якості вихідного коду?
9) Які стратегії можна застосувати для рефакторингу вихідного коду на основі його аналізу?
10) Які переваги може мати покращення якості вихідного коду для розробки програмного забезпечення та кінцевих користувачів?

Лабораторна робота №2. Тестування в реальному середовищі

Теоретичні відомості

Тестування в реальному середовищі – це процес, що включає в себе виконання тестів програмного забезпечення в умовах, максимально наближених до реальних умов використання продукту. Це дозволяє виявити проблеми та недоліки, які можуть виникнути при реальному використанні програми користувачами.

Основні види тестування в реальному середовищі включають:

1. Тестування на реальних пристроях: Виконання тестів на реальних пристроях, таких як смартфони або планшети, для перевірки сумісності та функціональності програмного забезпечення (додаток А.1).
2. Тестування в реальних мережевих умовах: Виконання тестів в реальних мережевих умовах для перевірки продуктивності та надійності програмного забезпечення під час реального використання в мережі.
3. Тестування в реальних відомостях про користувачів: Виконання тестів з участю реальних користувачів для оцінки зручності використання та реакції на нові функції.

Приклад виконання лабораторної роботи

Припустимо, ми тестуємо веб-додаток, який використовується для онлайн-торгівлі, в реальних умовах.

Спочатку потрібно підготувати ізольоване тестове середовище, яке максимально відтворює умови реального використання додатку.

Після підготовки тестового середовища, потрібно виконати реальні сценарії використання. При виконання тестових сценаріїв, які моделюють реальні дії користувачів, включаючи пошук товарів, додавання їх до кошика та

оформлення замовлень, ми дізнаємось, про правильність роботи програмного забезпечення.

Наступним кроком потрібно провести аналіз продуктивності та надійності. Наприклад, з використанням додатку Selenium, після виміру часу відповіді сервера, ми виявили деякі проблеми з продуктивністю, які потребують подальшої оптимізації.

Наступним кроком буде збір фідбеку від користувачів. Для цього потрібно включити засоби для отримання фідбеку в додаток, та отримати інформацію від реальних користувачів про їхні враження, та пропозиції.

Задачі для розв'язання

1. Визначте функціонал програмного продукту для тестування.
2. Розробіть тест-кейси та визначте критерії успішності тестування (додаток А.2.).
3. Виконайте тестування в реальному середовищі, фіксуючи результати та виявлені помилки.
4. Створіть звіт про тестування, включаючи рекомендації щодо виправлення помилок.

Контрольні запитання

1) Що таке тестування в реальному середовищі і чому воно важливе для розробки програмного забезпечення?
2) Які основні відмінності між тестуванням у контрольованому середовищі та тестуванням в реальному середовищі?
3) Які типи тестів можуть бути проведені в реальному середовищі? (додаток А3)
4) Як визначити потрібний рівень тестування для конкретного продукту чи функції в реальному середовищі?
5) Як впливає реальне середовище на результати тестування та які чинники можуть впливати на його ефективність?
6) Які інструменти можуть бути використані для проведення тестування в реальному середовищі?
7) Як визначити обсяг та покриття тестування для забезпечення належної якості продукту в реальних умовах експлуатації?
8) Які можливі проблеми можуть виникнути під час тестування в реальному середовищі і як їх уникнути?
9) Як організувати та планувати процес тестування в реальному середовищі для забезпечення ефективного виконання?
10) Які можливі переваги та недоліки можуть виникнути в результаті тестування в реальному середовищі порівняно з іншими методами тестування?

Лабораторна робота №3. Впровадження змін згідно Scrum

Теоретичні відомості

Scrum – це гнучкий підхід до розробки програмного забезпечення, що базується на ітеративному та інкрементальному підходах. Його основні принципи включають в себе розподіл роботи на короткі ітерації, відомі як спринти, та активне співробітництво між учасниками команди.

Основні етапи впровадження змін за методологією Scrum включають:

1. Планування спринту: Визначення завдань, які будуть виконуватися протягом спринту.
2. Розробка: Виконання завдань згідно з планом спринту.
3. Спринт-ретроспектива: Оцінка роботи команди після завершення спринту та виявлення можливих шляхів покращення.

Приклад виконання лабораторної роботи

При виконанні лабораторної роботи, впровадження змін згідно Scrum, ми виконуємо роботу з впровадженням змін у веб-додатку відповідно до цієї методології.

Для початку виконання лабораторної роботи, потрібно сформувати Scrum-команди. Наприклад наша команда складається з 5 розробників, Scrum-майстра та продуктового власника. Ми провели зустріч для організації роботи та встановлення пріоритетів. Наступним кроком буде планування спринту. Під час планування спринту, ми обговорили та прийняли 10 задач для виконання протягом двотижневого спринту. Кожна задача була оцінена відповідно до складності.

Реалізація змін. Команда приступила до виконання задач за методологією Scrum. Ми проводили щоденні стендапи для відстеження прогресу та вирішення можливих проблем. Останнім кроком буде спринтовий огляд. Після завершення спринту, ми провели огляд виконаних робіт зі Scrum-командою, продуктовим власником та іншими зацікавленими сторонами. Обговорили досягнення та можливі покращення.

Задачі для розв'язання

1. Визначте конкретні зміни у функціоналі програмного продукту.
2. Створіть Backlog змін та розподіліть їх за ітераціями.
3. Застосуйте елементи Scrum, такі як Sprint Planning та Daily Standups, для впровадження змін.
4. Слідкуйте за прогресом та оцінюйте вплив змін на якість продукту.

Контрольні запитання

1) Що таке методологія Scrum та які принципи лежать в її основі?
2) Які ключові ролі і відповідальності учасників в Scrum-команді?
3) Які основні артефакти використовуються в Scrum для управління проектом?

4) Як відбуваються Scrum-події (спринти, планування спринтів, ретроспективи тощо) та яка їх роль у процесі розробки?
5) Як визначити обсяг та тривалість спринта для проекту?
6) Як здійснюється планування та прийняття змін в ході реалізації проекту за методологією Scrum?
7) Як забезпечити комунікацію та співпрацю між учасниками Scrum-команди для досягнення спільних цілей?
8) Як визначити та оцінити ризики, пов'язані з впровадженням Scrum у власній організації?
9) Які можливі перешкоди можуть виникнути під час впровадження Scrum та як їх уникнути або подолати?
10) Які переваги та виклики можуть виникнути після успішного впровадження Scrum у робочий процес команди?

Лабораторна робота №4. Оцінка продуктивності та реакції користувачів

Теоретичні відомості

Оцінка продуктивності - це процес вимірювання та аналізу продуктивності програмного забезпечення в реальних умовах використання. Вона включає в себе оцінку швидкості виконання операцій, часу реакції системи на дії користувачів, а також використання ресурсів комп'ютера.

Збір фідбеку від користувачів - це процес отримання, аналізу та використання відгуків користувачів щодо якості та функціональності програмного забезпечення. Це може включати в себе анкетування, вивчення відгуків у соціальних мережах та збір аналітичних даних про використання програми.

Приклад виконання лабораторної роботи

Наприклад ми будемо оцінювати продуктивність та реакцію користувачів інтернет-магазину. Спочатку ми проводимо аналіз продуктивності. Виміряємо час завантаження головної сторінки та сторінок категорії товарів. Далі зберемо інформацію про час відклику сервера під час різних запитів: пошук, додавання товару в кошик, оплата.

Наступним кроком буде збір реакції користувачів. Проведемо опитування або спостереження за користувачами, під час їх перебування на сайті. Аналіз відгуків користувачів у відгуках на сайті, соціальних мережах, форумів. Визначимо основні скарги та пропозиції щодо роботи інтернет-магазину.

Далі оцінка результатів. Порівняємо отримані дані з рекомендаціями та стандартами щодо продуктивності інтернет-магазинів. Визначимо основні проблеми та вразливі місця, які впливають на продуктивність інтернет-магазину, та задоволення користувачів.

Останнім кроком є розробка рекомендації. Встановлення конкретних кроків для покращення продуктивності магазину, таких як оптимізація шаблонів, кешування сторінок, використання CDN тощо. Розробка стратегії поступового впровадження рекомендацій з урахуванням витрат та строків виконання.

Задачі для розв'язання

1. Встановіть інструменти для моніторингу продуктивності.
2. Проведіть тести на продуктивність під час пікового навантаження.
3. Зберіть фідбек від користувачів, використовуючи опитування чи звернення до служби підтримки.
4. Аналізуйте результати та розробіть план подальших заходів для покращення якості.

Контрольні запитання

1) Чому оцінка продуктивності та реакції користувачів є важливою частиною процесу розробки програмного забезпечення?
2) Які методи використовуються для вимірювання продуктивності програмного забезпечення?
3) Я к визначити критерії оцінки реакції користувачів на продукт?
4) Які метрики можуть використовуватися для оцінки реакції користувачів на продукт?
5) Як здійснюється збір даних про реакцію користувачів на продукт?
6) Як аналізувати результати тестування продуктивності та реакції користувачів?
7) Які можливі проблеми можуть виникнути під час оцінки продуктивності та реакції користувачів і як їх вирішити?
8) Як зробити висновки на основі результатів оцінки продуктивності та реакції користувачів?
9) Які можливі стратегії покращення продуктивності та реакції користувачів можна розглянути на основі отриманих даних?
10) Які переваги може мати вдосконалення продуктивності та реакції користувачів для команди розробників та кінцевих користувачів?

Лабораторна робота №5. Управління виявленими помилками

Теоретичні відомості

Управління виявленими помилками є ключовим аспектом розробки програмного забезпечення, оскільки допомагає забезпечити високу якість та надійність продукту. Цей процес включає в себе кілька етапів:

1. Ідентифікація помилок: Помилки можуть виявитися під час розробки, тестування або експлуатації програмного забезпечення. Важливо правильно ідентифікувати їх та документувати для подальшого виправлення.

2. Реєстрація в системі відстеження помилок: Кожна виявлена помилка реєструється у спеціальній системі, такій як Bug Tracker, де вказується її опис, причина виникнення, стан та пріоритет.
3. Відстеження та аналіз: Помилки відстежуються з моменту їх виявлення до моменту виправлення. Проводиться аналіз причин їх виникнення для попередження подібних ситуацій у майбутньому.
4. Виправлення та тестування: Після ідентифікації та виправлення помилок проводиться тестування, щоб переконатися, що виправлення було успішним та не викликало нових проблем.
5. Документування та звітність: Всі кроки, пов'язані з управлінням помилками, повинні бути детально задокументовані для подальшого аналізу та використання в майбутніх проектах.

Управління виявленими помилками допомагає забезпечити стабільність та надійність програмного продукту, підвищуючи задоволеність користувачів та зменшуючи витрати на підтримку.

Приклад виконання лабораторної роботи

Припустимо, ми аналізуємо та виправляємо помилки, які виявлені у веб-додатку під час тестування.

Ідентифікація помилок: Ми використовували інструменти для виявлення помилок, такі як журнали помилок та звіти про відстеження помилок, щоб ідентифікувати проблеми у функціональності та стабільності додатку.

Реєстрація та відстеження помилок: Кожна виявлена помилка була документована у системі відстеження помилок, де їй був присвоєний унікальний ідентифікатор. Ми відстежували статус кожної помилки від її виявлення до виправлення.

Виправлення та перевірка помилок: Наша команда розробників виправила кожну виявлену помилку та перевірила її виправлення шляхом проведення тестів, щоб переконатися, що проблема вирішена та не виникли нові проблеми.

Аналіз та уроки з кожної помилки: Після виправлення кожної помилки ми проводили аналіз, щоб з'ясувати причини, які призвели до її виникнення, та розробляли план дій для уникнення схожих проблем у майбутньому.

Задачі для розв'язання

1. Аналізуйте виявлені помилки під час тестування та експлуатації.
2. Визначте серйозність кожної помилки та її вплив на користувачів.
3. Розробіть план виправлення помилок, визначте пріоритети та строки виправлення.

Контрольні запитання

1) Чому важливо виявлення та управління помилками під час розробки програмного забезпечення?
2) Які основні кроки включає процес управління виявленими

помилками?
3) Як класифікувати виявлені помилки за їхньою серйозністю та впливом на продукт?
4) Як здійснюється відстеження та документування виявлених помилок?
5) Як визначити пріоритетність виявлених помилок для їх подальшого виправлення?
6) Як впливають виявлені помилки на графік та обсяг робіт з розробки програмного забезпечення?
7) Як забезпечити ефективне виправлення виявлених помилок у встановлені терміни?
8) Як оцінити ефективність процесу управління виявленими помилками та зробити відповідні висновки?
9) Які можливі стратегії запобігання виявленню помилок можна врахувати при розробці програмного забезпечення?
10) Як впливає процес управління виявленими помилками на якість та користування програмним забезпеченням?

Лабораторна робота №6. Аудит якості коду та процесів розробки

Теоретичні відомості

Аудит якості коду та процесів розробки - це комплексний процес перевірки та оцінки якості вихідного коду та процесів розробки програмного забезпечення. Основні етапи аудиту включають в себе:

1. Оцінка якості коду: Аналіз структури, читабельності, ефективності та безпеки програмного коду з метою виявлення недоліків та вдосконалення практик програмування.
2. Перевірка відповідності стандартам: Перевірка коду на відповідність встановленим стандартам програмування та кращим практикам розробки.
3. Оцінка процесів розробки: Аналіз ефективності та ефективності процесів розробки програмного забезпечення, таких як методології розробки, управління проектами та комунікації в команді.
4. Визначення рекомендацій та покращень: На основі результатів аудиту розробляються конкретні рекомендації та стратегії для вдосконалення якості коду та процесів розробки.
5. Впровадження виправлень: Розроблені рекомендації виконуються та впроваджуються у робочі процеси з метою поліпшення якості та ефективності розробки програмного забезпечення.

Аудит якості коду та процесів розробки допомагає виявити та виправити недоліки, що сприяє підвищенню ефективності та якості роботи команди розробників.

Приклад виконання лабораторної роботи

Припустимо, що ми аудитуємо якість коду та процесів розробки великого проекту з використанням мови програмування Java та фреймворку Spring.

Аналіз коду на відповідність стандартам: Ми використовували інструменти для аналізу коду на відповідність стандартам кодування Java, таким як Checkstyle, і виправляли виявлені невідповідності.

Перевірка ефективності та безпеки коду: Ми використовували інструменти для аналізу ефективності та безпеки коду, такі як FindBugs та SonarQube, і виправляли виявлені проблеми з ефективністю та безпекою.

Оцінка процесів розробки: Ми аналізували процеси розробки, включаючи керування версіями, тестування та впровадження, та розробляли рекомендації для їх поліпшення.

Розробка рекомендацій та планування дій: На основі результатів аудиту ми розробили конкретні рекомендації для вдосконалення якості коду та процесів розробки і склали план дій для їх впровадження.

Задачі для розв'язання

1. Заплануйте аудит якості коду та процесів розробки.
2. Проведіть аудит, використовуючи стандарти та методики, враховуючи вимоги якості та ефективність процесів.
3. Створіть звіт з рекомендаціями для подальшого вдосконалення.
4. Розробіть план дій для впровадження рекомендацій

Контрольні запитання

1) Що таке аудит якості коду та процесів розробки та чому він важливий для проектів з розробки програмного забезпечення?
2) Які основні цілі та завдання проведення аудиту якості коду та процесів розробки?
3) Які критерії та метрики можуть використовуватися для оцінки якості коду?
4) Які методи можна використовувати для проведення аудиту якості коду та процесів розробки?
5) Які можливі проблеми можуть бути виявлені під час аудиту якості коду та процесів розробки?
6) Які кроки потрібно вжити для виправлення виявлених проблем після проведення аудиту?
7) Як забезпечити ефективне впровадження рекомендацій, отриманих під час аудиту?
8) Як оцінюється ефективність аудиту якості коду та процесів розробки?
9) Які можливі переваги може мати виконання аудиту якості коду та процесів розробки для команди розробників та продукту?
10) Як аудит якості коду та процесів розробки впливає на загальну якість та довгостроковий успіх проекту з розробки програмного забезпечення?

Лабораторна робота №7. Створення рекомендацій з покращення якості продукту

Теоретичні відомості

Створення рекомендацій з покращення якості продукту є важливим етапом у розвитку програмного забезпечення. Цей процес включає в себе кілька ключових кроків:

1. Аналіз поточного стану продукту: Проведення детального аналізу функціональності, продуктивності та інших аспектів продукту для визначення його сильних та слабких сторін.
2. Визначення потреб користувачів: Вивчення потреб та вимог користувачів щодо функціональності, зручності використання та інших аспектів продукту.
3. Оцінка конкурентоспроможності: Аналіз конкурентного середовища та порівняння продукту з конкурентами з метою визначення його переваг та недоліків.
4. Розробка конкретних рекомендацій: На основі отриманих даних розробляються конкретні рекомендації з покращення якості продукту, які можуть стосуватися функціональних змін, інтерфейсу користувача, продуктивності та інших аспектів.
5. Створення плану дій: Розробка конкретного плану дій для впровадження рекомендацій з метою покращення якості продукту.
6. Впровадження та оцінка результатів: Виконання запланованих заходів з покращення та оцінка їх ефективності у покращенні якості та конкурентоспроможності продукту.

Створення рекомендацій з покращення якості продукту допомагає розробникам зрозуміти потреби користувачів та ринкові тенденції, що дозволяє створити продукт, який задовольняє вимоги споживачів та відповідає найсучаснішим стандартам.

Приклад виконання лабораторної роботи

Проведемо аналіз продуктивності інтернет-магазину та визначимо, як користувачі сприймають швидкість його роботи. Розробимо рекомендації щодо покращення продуктивності та сприйняття користувачами швидкості роботи магазину.

Аналіз продуктивності:

Виміряємо час завантаження головної сторінки та сторінок категорій/товарів.

Зберемо інформації про час відклику сервера під час різних типів запитів (пошук, додавання товару до кошика, оплата тощо).

Оцінимо пропускну спроможність сервера та швидкість завантаження зображень та інших ресурсів.

Збір реакції користувачів:
Проведемо опитування або спостереження за користувачами під час їхнього перебування на сайті.
Аналізуємо відгуки користувачів у відгуках на сайті, соціальних мережах та форумах.
Визначимо основні скарги та пропозицій щодо швидкості роботи магазину.
Оцінка результатів:
Порівняємо отримані дані з рекомендаціями та стандартами щодо продуктивності інтернет-магазинів.
Визначимо основні проблеми та вразливі місця, які впливають на продуктивність магазину та задоволення користувачів.
Розробка рекомендацій:
Встановимо конкретні кроки для покращення продуктивності магазину, таких як оптимізація шаблонів, кешування сторінок, використання CDN тощо.
Розробимо стратегію поступового впровадження рекомендацій з урахуванням витрат та строків виконання.

Висновок:

Аналіз та оцінка продуктивності та сприйняття користувачами швидкості роботи інтернет-магазину є важливим етапом у процесі його розробки та підтримки. Ця лабораторна робота дозволила ідентифікувати проблеми та розробити конкретні рекомендації щодо їх вирішення. Покращення продуктивності та сприйняття користувачами швидкості роботи магазину позитивно вплине на їхнє задоволення від використання магазину та сприятиме підвищенню конверсії та заробітку.

Задачі для розв'язання

1. Вивчіть аналіз коду, результати тестування та аудиту, щоб зрозуміти сильні та слабкі сторони продукту.
2. Визначте конкретні області, де можна покращити якість продукту.
3. Розробіть рекомендації з покращення якості, зосереджуючись на конкретних аспектах, таких як безпека, швидкість або зручність використання.
4. Визначте конкретні кроки для впровадження кожної рекомендації.

Контрольні запитання

1) Чому створення рекомендацій з покращення якості продукту є важливою частиною процесу розробки програмного забезпечення?
2) Які основні критерії використовуються для оцінки якості програмного продукту?
3) Які методи можна використовувати для збору відгуків користувачів про продукт?

4) Як виявити слабкі місця та проблеми в продукті за допомогою аналізу відгуків користувачів?
5) Як враховувати потреби та очікування користувачів під час розробки рекомендацій?
6) Як вибрати пріоритетні напрями покращення на основі аналізу відгуків користувачів?
7) Які можливі стратегії покращення якості продукту можна розглянути на основі отриманих даних?
8) Як забезпечити ефективну комунікацію та співпрацю між розробниками та користувачами під час розробки рекомендацій?
9) Як оцінити вплив запропонованих змін на якість та використовуваність продукту?
10) Як впливають рекомендації з покращення на користувачів та бізнес продукту?

Додаток

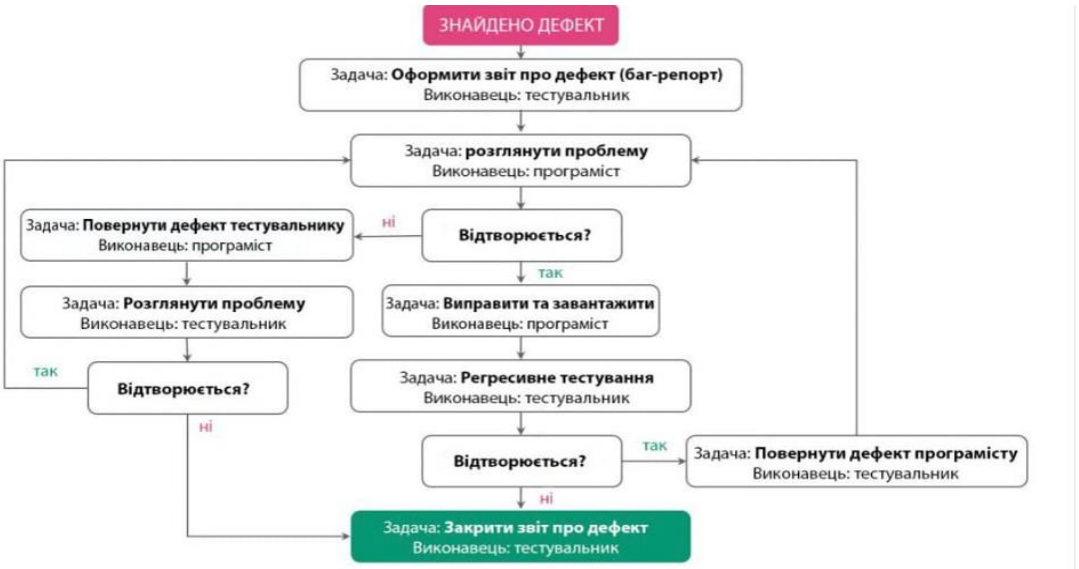

Рисунок А.1 Життєвий цикл багу

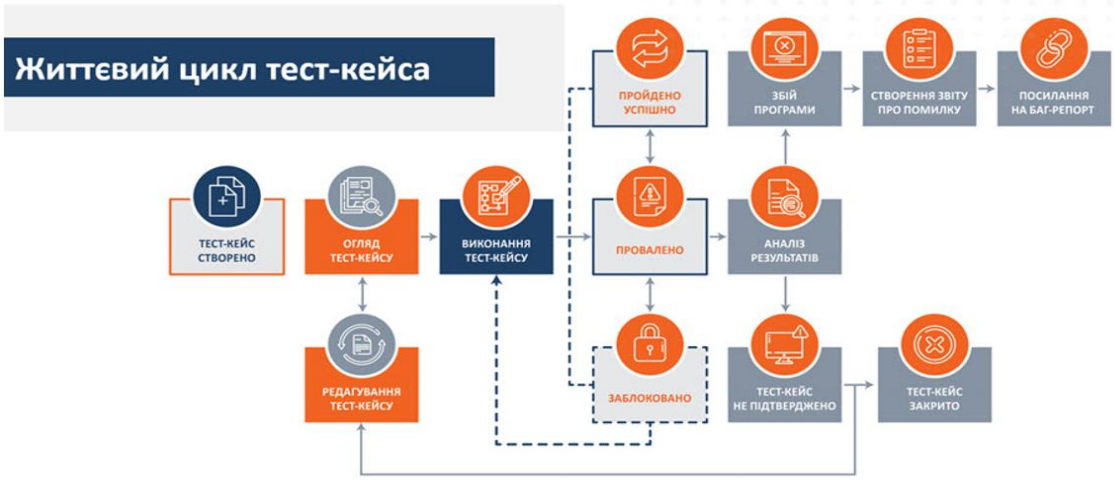

Рисунок А.2 Життєвий цикл тест-кейсу

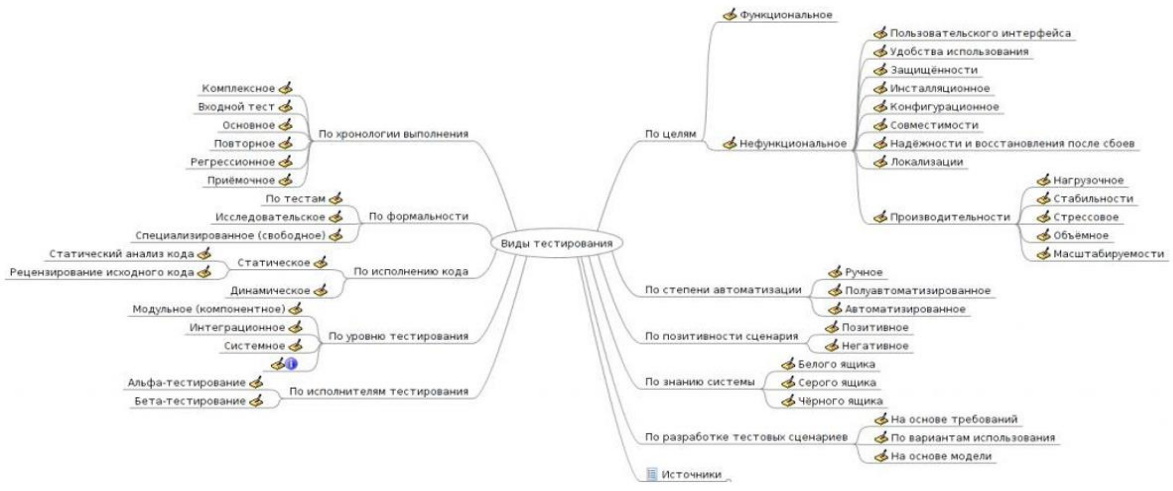

Рисунок А.3 Класифікація та види тестування

СПИСОК ЛІТЕРАТУРИ

1. Chaos Engineering: System Resiliency in Practice 1st Edition (Casey Rosenthal) [Електронний ресурс]. – Режим доступу: https://balka-book.com/ua/razrabotka_programnogo_obespecheniya-366/chaos_engineering_system_resiliency_in_practice_1st_edition-114661
2. Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing 1st Edition (Elisabeth Hendrickson) [Електронний ресурс]. – Режим доступу:https://dokumen.pub/explore-it-reduce-risk-and-increase-confidence-with-exploratory-testing-9781937785024-1937785025.html
3. Effective Software Testing: A Developer's Gui [Електронний ресурс]. – Режим доступу: https://kupichitay.com.ua/product/effective-software-testing-a-developers-guide/
4. Системи управління якістю [Електронний ресурс]. – Режим доступу: https://book.sumy.ua/sistemi-upravlinnya-yakistyu/
5. Testing Computer Software [Електронний ресурс]. – Режим доступу:https://books.google.com.ua/books/about/Testing_Computer_Software.html?id=Q-hTDwAAQBAJ&redir_esc=y
6. Testing Computer Software, 2nd Edition [Електронний ресурс]. – Режим доступу: https://kupichitay.com.ua/product/software-testing-ron-patton/
7. Foundations of Software Testing: ISTQB Certification [Електронний ресурс]. – Режим доступу: https://www.yakaboo.ua/ua/foundations-of-software-testing-istqb-certification.html?gad_source=1&gclid=Cj0KCQjw2a6wBhCVARIsABPeH1tlhzMFSjt2VgvmlHuETWCFt2d0sHRdn0gTPbtykIAAF0HD6Qi43loaArtbEALw_wcB
8. Fifty Quick Ideas to Improve Your Tests [Електронний ресурс]. – Режим доступу: https://books.google.com.ua/books/about/Fifty_Quick_Ideas_to_Improve_Your_Tests.html?id=Re3FsgEACAAJ&redir_esc=y
9. Selenium Framework Design in Keyword-Driven Testing: Automate Your Test Using Selenium and Appium [Електронний ресурс]. – Режим доступу:https://dokumen.pub/selenium-framework-design-in-keyword-driven-testing-automate-your-test-using-selenium-and-appium-english-edition-9789389328202-9389328209.html
10. Robot Framework Test Automation [Електронний ресурс]. – Режим доступу: https://api.pageplace.de/preview/DT0400.9781783283040_A24172264/preview-9781783283040_A24172264.pdf

Навчально-методичне видання

ЗАСОБИ КЕРУВАННЯ ЯКІСТЮ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методичні вказівки та завдання

до виконання лабораторних робіт №1-7

для підготовки здобувачів другого магістерського рівня вищої освіти спеціальності 121 «Інженерія програмного забезпечення»

Укладачі: Т. А. Гончаренко, кандидат технічних наук, доцент
О. О. Маційовський, асистент

Комп'ютерне верстання

Підписано до друку 22.02.2024 Формат 60 × 84 1/ 16

Ум. друк. арк. 1,16. Обл.-вид. арк. 1,25.

Електронний документ. Вид № 59/III-17.

Видавець і виготовлювач

Київський національний університет будівництва і архітектури

Повітрофлотський проспект, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб'єктів

Видавничої справи ДК №808 від 13.02.20