DOI: https://doi.org/10.36910/6775-2524-0560-2023-53-22
УДК: 004.42
Міскевич Оксана Іванівна, асистент
https://orcid.org/0000-0002-5009-2391
Столенко Ігор Олексійович, бакалавр
Куліков Назар Олександрович, бакалавр
Луцький національний технічний університет, м. Луцьк, Україна

ФРЕЙМВОРК TESTNG ЯК ІНСТРУМЕНТ ДЛЯ НАПИСАННЯ ТЕСТІВ

Міскевич О.І., Столенко І.О., Куліков Н.О. Фреймворк testNG як інструмент для написання тестів. У даній статті розглянуто основні концепції, методології та техніки тестування для якості програмних продуктів та виявлення дефектів після перевірки програмного забезпечення. Адже вибір правильної методики є необхідністю до хорошого результату. Розробка автоматизованих тестів для тестування ПЗ на сьогодні є дуже актуальною та необхідною.

Поставлена задача вимагає вивчення: ключових питань тестування, класифікації видів тестування, вимог та фаз тестування, класів еквівалентності та основні види та стани тест-кейсів, які будемо розглядати в даній статті.

Розглянуто: основні атрибути Test Case; позитивні та негативні тест-кейси, які перевіряють дані та виняткові ситуації; високорівневі та низькорівневі тест-кейси і створення ефективних тестових кейсів. А також фреймворк TestNG, який допомагає написати тести на мові програмування Java.

Ключові слова: автоматизоване тестування, Test Case , Agile, Scrum, мануальне тестування, тестування інтерфейсу, testNG, Selenium.

Miskevych O., Stolenko I., Kulikov N. The testNG framework as a tool for writing tests. This article discusses the basic concepts, methodologies, and testing techniques for software product quality and defect detection after software verification. After all, choosing the right technique is a necessity for a good result. The development of automated tests for software testing is very relevant and necessary today.

The given task requires the study of: key issues of testing, classification of types of testing, requirements and phases of testing, equivalence classes and the main types and states of test cases, which will be considered in this article.

Considered: the main attributes of the Test Case; positive and negative test cases that test data and exceptional situations; high-level and low-level test cases and creating effective test cases. And also the TestNG framework, which helps to write tests in the Java programming language.

Keywords: automated testing, Test Case Agile, Scrum, manual testing, interface testing, testNG, Selenium.

Постановка проблеми.

Сучасний світ неможливо уявити без програмного забезпечення. Все, чим користується людина, від побутової техніки і автомобілів до смартфонів і комп'ютерів, базується на різноманітних програмних рішеннях. В умовах стрімкого розвитку інформаційних технологій, гаджети стали невід'ємною частиною нашого повсякденного життя. Вони полегшують нашу роботу, допомагають знайти інформацію, виконати обчислення та забезпечують безліч інших корисних функцій.

Однак, з масовим використанням програмного забезпечення, виникає основне питання: наскільки надійні та безпечні ці продукти? Якщо програмне забезпечення містить помилки або дефекти, це може призвести до серйозних проблем для користувачів. Щоб забезпечити високу якість та надійність програмного продукту, необхідне правильне тестування. Метою роботи є аналіз основних методологій розробки ПЗ для тестування.

Для досягнення мети було розглянуто основні концепції методологій на прикладі Agile та Scrum.

Викладання основного матеріалу. У сучасному інформаційному середовищі, де швидкість і адаптивність є ключовими чинниками успіху, методології розробки програмного забезпечення, такі як Agile та Scrum, стають все більш популярними. Хоча вони створювалися, як методології для розробки, їх можна пристосувати до будь-якої задачі. Давайте розглянемо основні концепції цих методологій та їх вплив на процес тестування.

Agile - це підхід , основним акцентом якого є гнучкість, відкритість та співпраця. Принципи Agile включають інкрементальну розробку, самоорганізацію команди, співпрацю з клієнтом. Це дозволяє швидко реагувати на зміни в вимогах та постійно вдосконалювати продукт.

Scrum - це популярна Agile-методологія, яка допомагає керувати проектами та розробляти програмні продукти. Для ефективного керування проектом визначаються чіткі ролі, події та артефакти.

У Scrum існують три ролі:

1. Product Owner, або власник продукту, який відповідає за управління вимогами та пріоритетами продукту.
2. Scrum Master, який забезпечує дотримання методології Scrum та допомагає команді вирішувати необхідні проблеми.
3. Development Team, або команда розробників, яка виконує усю розробку продукту.

Це розділення ролей сприяє чіткому розподілу обов'язків і ефективному виконанню завдань.

Спрінт - це інтервал часу, під час якого команда працює над конкретними поставленими завданнями. Тривалість цієї роботи від 2 до 4 тижнів. Готовий продукт є завершенням завдання.

Scrum включає кілька важливих подій для кращої організації роботи команди.

Розглянемо кожен з них:

Першою подією є планування Sprint Planning, під час якого вся команда детально обговорює та визначає завдання, які планує виконати протягом наступного спрінту. Важливо, що усі завдання вибираються з Product Backlog.

Другою подією є щоденні зустрічі Daily Scrum, які відбуваються щоденно під час спрінту, де команда оцінює свій прогрес та вирішує проблеми, які виникають. Кожен член команди зазвичай відповідає на три основні питання: "Що я виконав вчора?", "Над чим я працюю сьогодні?", "Чи є якісь перешкоди?"

Третя подією є перегляд Sprint Review, після завершення спрінту команда демонструє результати своєї роботи при зустрічі з клієнтом або стейкхолдерами та отримає зворотний зв'язок.

Четверта подія аналіз Sprint Retrospective, де команда аналізує минулий спрінт, виявляє, що пішло погано та добре і де є можливість покращення. Результати ретроспективи допомагають команді постійно вдосконалювати свій процес.

Використання Scrum у роботі з Jira: У багатьох організаціях для керування проектами і використання Scrum використовується програмне забезпечення Jira. Jira надає зручні інструменти для створення і управління Product Backlog, Sprint Backlog та відстеження інкрементів.

Product Backlog у Jira: Product Backlog - це список всіх завдань і вимог до продукту. В Jira ви можете створити проект і використовувати відповідний Backlog для реєстрації всіх вимог і завдань. Кожне завдання має опис, призначення і пріоритет, що допомагає команді розуміти, над чим працювати далі.

Sprint Backlog у Jira: Після планування спрінту команда вибирає завдання з Product Backlog і додає їх до Sprint Backlog, який в Jira представляє собою список завдань для поточного спрінту.

Increment у Jira: Increment - це готовий до випуску продукт, який створюється на кінець кожного спрінту. В Jira ви можете відстежувати роботу над проектом та перевіряти стан готовності інкременту.

Використання Agile та Scrum у роботі зі спеціальними інструментами, такими як Jira, допомагає командам ефективно керувати проектами та забезпечувати високу якість програмного продукту. Ці методології надають інструменти та процеси, що сприяють постійному вдосконаленню та надійності продукту, а також зростанню задоволеності клієнтів.

Процес тестування при розробці програмного забезпечення та його подальшому вдосконаленні відіграє важливу роль. Якість та надійність програмного продукту безумовно залежать від якості проведеного тестування. Існують два основних напрямки тестування: мануальне та автоматизоване.

Мануальне тестування є традиційним та найпоширенішим способом перевірки програмного забезпечення. В цьому методі тестувачі вручну виконують тести на програмі, і вони грають ключову роль у виявленні помилок та встановленні відповідності програми вимогам. Існують декілька важливих переваг мануального тестування:

Мануальне тестування надає можливість швидко адаптуватися до різних змін у програмі. Тестувальники можуть модифікувати написані тести та проводити нові перевірки без необхідності переробки автоматизованих тестів.

Мануальне тестування надає можливість для перевірки інтерфейсу та взаємодії програми з користувачем, емулювати реальні умови використання програми та переконатися у тісній співпраці з користувачем.

Мануальне тестування дозволяє виявляти неочевидні помилки та недоліки, на відміну від автоматизованого тестування, де можливість упустити помилку. Тестувальники можуть виявити проблеми, які потребують інтуїції та творчого підходу.

Автоматизоване тестування передбачає використання програмних скриптів та інструментів для автоматизації тестів.

Розглянемо автоматизоване тестування на прикладі .

Було проведено тестування реєстрації на сайті E katalog. Чи все працює коректно. Завдяки тест-кейсу ми проходили шлях реєстрації на сайті від початку до кінця і перевіряли чи все виконується правильно , чи немає багів.

Перший етап для автоматизованого тестування є написання тест-кейсів.

Таблиця 1 – Тест-кейси

ID	1
Precondition:	The user must be on HomePage.
Steps:	1.Click on the "Log in" button.
	2.Click on the "Or register" button.
	3.Enter your name on the "Name" field.
	4.Enter your email on the "Email" field.
	4.Enter your Password on the "Password" field and click register.
Expected result	User must see registration was successful.

Наступне - це підготовка тестового середовища. В нашому випадку це ITELLIJ IDEA на мові програмування Java.

```
@BeforeMethod
public void openBrowser() {
   driver = new ChromeDriver();
   driver.get(BASE_URL);
   driver.manage().window().maximize();
}

@AfterMethod
public void closeBrowser() {
   //log.info("<<<=== Teardown");
   //driver.quit();
}
```

Прописуємо анотації:

@BeforeMethod – ця анотація відкриває браузер Google Chrome. (Вона в коді виконується сама перша).

@AfterMethod – ця анотація закриває браузер. (Вона в коді виконується сама остання після проходження Авто-тесту).

Наступний наш крок - через Xpath витягуємо елементи на веб-сторінці , які нам потрібні для проходження нашого тесту.

Наприклад: В пункті один нашого тест-кейсу необхідно натиснути на кнопку зареєструватися. Створюємо змінну і даємо назву, зазвичай називають, аналогічно, як і кнопку. Робимо такі змінні приватними і константами.

```
private static final String LOG_IN = "//div[@class=\"header_action_login\"]";
```

Рис.1. Створення змінної та назви

Після створення змінною прописуємо метод натискання на цю кнопку, щоб наш код був коротким і зрозумілим.

```
public HomePage clickLogIn() {
    waitUntilElementToBeClickable(LOG_IN).click();
    return this;
}
```

Рис.2. Короткий код програми

Тут використовуємо вейтер - це очікування перед тим, як програма продовжить виконання, щоб переконатися, що все готово.

Після створення всіх змінних і методів , які прописані в тест-кейсі, викликаємо ці методи для того, аби наш код перевірив чи все працює коректно на веб-сторінці.

```
@Test
public void Registration() {
    HomePage homePage = new HomePage(driver);

    homePage.clickLogIn()
            .clickRegestrationButton()
            .inputNameInTheField(NAME)
            .inputEmailInTheField(EMAIL)
            .inputPasswordField(PASSWORD)
            .clickButtonRegestr()
            .clickConfirm();

    Assert.assertTrue(homePage.textIsDisplayed());
    Assert.assertTrue(homePage.buttonIsDisplayed());
}
```

Рис.3. Виконання коду

Якщо після виконання нашого коду виводиться "Passed," це підтверджує успішне проходження тесту. У випадку, коли цього не сталося, це свідчить про виявлення багу.

В цьому автотесті використано Selenium.

Selenium - це набір інструментів і бібліотек для автоматизації веб-додатків. Він дозволяє програмістам і тестерам створювати скрипти, які автоматично взаємодіють з веб-додатками, виконуючи такі дії, як натискання на кнопки, заповнення форм, перевірка вмісту сторінок і багато інших. Selenium може бути використаний для тестування веб-додатків, веб-скрапінгу, а також для автоматизації завдань, які вимагають взаємодії з веб-сайтами.

Selenium надає можливість вибору мови програмування для написання автоматизованих тестів або скриптів, і підтримує багато різних браузерів, таких як Chrome, Firefox, Edge, інші. Selenium також має можливість працювати в різних операційних системах.

Основні компоненти Selenium включають у себе:

Selenium WebDriver: Це API, яке надає можливість програмно керувати веб-браузерами. Ви можете використовувати WebDriver для виконання дій на сторінках веб-сайтів, таких як кліки, введення тексту, отримання і перевірки даних тощо.

Selenium IDE: це розширення для браузера, яке дозволяє записувати та відтворювати дії користувача на сторінках веб-сайтів. Ви можете використовувати Selenium IDE для створення автоматизованих тестів без програмування.

Selenium Grid: цей компонент дозволяє запускати тести на різних браузерах та операційних системах паралельно, що полегшує тестування на різних конфігураціях.

Selenium є популярним інструментом в сфері тестування програмного забезпечення та веб-розробки, і він допомагає автоматизувати багато рутинних завдань, пов'язаних з взаємодією з веб-додатками.

Писати на фреймфорці TestNG.

TestNG (Test Next Generation) - це фреймворк для тестування програмного забезпечення, який використовується для написання і виконання автоматизованих тестів в Java. TestNG розшифровується як "Test Next Generation" і був створений як більш потужний і гнучкий аналог JUnit і TestNG, із підтримкою більш широкого спектру тестових сценаріїв. Основні особливості та переваги TestNG включають у себе:

Анотації: TestNG використовує анотації (маркери) для вказівки методів, які слід виконати під час тестування, такі як @Test, @BeforeMethod, @AfterMethod і інші. Це дозволяє легко визначити порядок виконання тестових методів та конфігураційні дії перед і після тестування.

Групи тестів: Ви можете групувати тести в TestNG і виконувати лише певні групи тестів або виключити окремі групи з тестування. Це дозволяє вам легко організовувати ваші тести та виконувати їх партіями.

Параметризація: TestNG підтримує параметризацію тестів, що дозволяє вам виконувати один і той же тест з різними вхідними даними.

Залежності між тестами: Ви можете визначати залежності між тестами, що вказує порядок виконання тестів і гарантує, що попередні тести завершаться успішно перед виконанням наступних.

Розширені можливості налаштування: TestNG надає багато можливостей для налаштування тестового оточення, такі як робота з XML-файлами конфігурації тестів і визначення параметрів.

TestNG став популярним фреймворком для тестування в Java, особливо в контексті автоматизованого тестування, і він використовується в багатьох проектах для написання та виконання тестів.

Результати дослідження. Автоматизоване тестування сьогодні є необхідністю для перевірки результатів та недопущення повторних помилок.

Автоматизовані тести виконуються значно швидше, ніж мануальні. Це дозволяє зекономити час та ресурси під час тестування.

Автоматизовані тести можуть бути повторно виконані без змін, що дозволяє легко перевіряти функціональність програми під час розробки.

Автоматизоване тестування ідеально підходить для регресійного тестування, тобто перевірки, чи не було порушено раніше працюючих функцій при внесенні нових змін в код.

Наш автотест на реєстрацію був пройдений успішно і багів було не виявленно. Тест пройдено декілька разів для кращого запобігання багів.

Висновки. Аналіз наукових досліджень і публікацій вказує на те, що автоматизоване тестування програмного забезпечення надає можливість кожному розробнику та тестувальнику підвищити ефективність і якість процесу тестування. Воно спрощує виконання багатьох рутинних завдань, сприяє швидкому виявленню помилок і реагуванню на зміни у вимогах до програмного продукту. Цей метод також допомагає покращити співпрацю між розробниками та тестувальниками, зменшити тривалість тестового циклу та знизити загальні витрати на тестування.

Вибір мови програмування для реалізації автоматизованих тестів впав на Java з численних причин. По-перше, Java вже визнана світовою спільнотою розробників як мова програмування, яка демонструє надійність та стабільність. По-друге, Java має велику кількість бібліотек та інструментів для автоматизованого тестування, що спрощує процес створення та підтримки тестових сценаріїв. Нарешті, використання Java дозволяє створювати переносимий код, що робить автоматизоване тестування більш гнучким і адаптованим до різних платформ.

Для перевірки функціональності програмного забезпечення був створений тест-кейс, який спочатку проходився вручну, за допомогою мануального тестування, і потім виконувався автоматизованими тестами. Ця автоматизація була призначена для спрощення процесу тестування та зменшення часу, потрібного для виконання тестових сценаріїв. Результати автоматизованих тестів дозволили оперативно перевірити відповідність роботи модуля реєстрації сайту E-katalog.

Після аналізу отриманих результатів можна впевнено заявити, що розроблене програмне забезпечення працює бездоганно, що підтверджує його надійність та високу функціональність.

Список бібліографічного опису

1. Міскевич О. Дослідження загроз від кібератак та захист персональної інформації. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2021. № 45. С. 84-89. https://doi.org/10.36910/6775-2524-0560-2021-45-12
2. Міскевич О., Каган І., Рожко О. Як обрати оптимальний ноутбук для навчання . Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2021. №43. С. 92-96. https://doi.org/10.36910/6775-2524-0560-2021-43-15
3. Христинець Н., Міскевич О., Мазуренко В. Технології Blockchain для оптимізації процесів документообігу. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2020. №40. С. 153-157.
4. Міскевич О., Багнюк , Н., Христинець , Н., Марчевська О. Автоматизація виявлення дефектної продукції методами машинного навчання . Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2020. №39.С. 175-180.
5. Грицюк Ю.І. Аналіз вимог до програмного забезпечення: навчальний посібник. Львів : Видавництво Львівської політехніки. 2018. 456 с.
6. Тестування та якість програмного забезпечення: URL: https://kpi-fictip32.github.io/Blog/s07/-software_quality.html. (дата звернення: 10.07.2023)

References

7. Miskevych O. Study of threats from cyberattacks and protection of personal information. Computer-integrated technologies: education, science, production. 2021. №. 45. P. 84-89. https://doi.org/10.36910/6775-2524-0560-2021-45-12
8. Miskevich O., Kagan I., Rozhko O. How to choose the optimal laptop for learning. Computer-integrated technologies: education, science, production. 2021. № 43. P. 92-96. https://doi.org/10.36910/6775-2524-0560-2021-43-15
9. Khrystinets N., Miskevich O., Mazurenko V. Blockchain technologies for optimizing document flow processes. Computer-integrated technologies: education, science, production. 2020. № 40. P. 153-157.
10. Miskevych O., Bagniuk, N., Khrystinets, N., Marchevska O. Automation of detection of defective products using machine learning methods. Computer-integrated technologies: education, science, production. 2020. №. 39. S. 175-180.
11. Hrytsyuk Y.I. Analysis of software requirements: a tutorial. Lviv: Lviv Polytechnic Publishing House. 2018. 456 p.
12. Software testing and quality: URL: https://kpi-fictip32.github.io/Blog/s07/-software_quality.html. (date of application: 10.07.2023)