УДК 004.415.53

ДОСЛІДЖЕННЯ ІНСТУРМЕНТУ JENKINS ДЛЯ РЕГРЕСІЙНОГО ТЕСТУВАННЯ

Омельницький А. А.
email: andrii.omelnytskyi@nure.ua
Харківський національний університет радіоелектроніки, каф. АПОТ
м. Харків, Україна

This work was developed to introduce the importance of the Jenkins tool for executing regression testing of software for computer systems. The basic goals and features were considered, and their practical usage was demonstrated. The importance of using Jenkins in different types and approaches to software testing was analyzed in relation to the software development life cycle. The results highlight the various features that can be used for executing regression testing in real commercial projects. This work demonstrates all main functions of Jenkins that allows create comfortable environment of software for computer systems.

Створення сучасного та багатофункціонального програмного забезпечення вимагає залучення значних ресурсів та можливостей з боку команди розробки: планування системи, аналіз вимог, проектування архітектури, програмування, тестування та розгортання [1]. Ключовим етапом виготовлення додатку є його тестування, яке дозволяє виявити всі наявні відхилення у роботі перед його комерційним запуском. Розробка якісного додатку є однією із основних вимог, адже конкурентний продукт не повинен мати відхилень фактичного результату роботи від очікуваного та не бути вразливим до хакерських атак.

Існує багато типів тестування, що дозволяють виявляти критичні дефекти у роботі застосунку, наприклад як тестування функціоналу, навантаження, безпеки та подібного. Серед основних видів виділяють також регресійне тестування, що спрямоване на перевірку функціоналу застосунку після внесених змін в коді програми.

Модифікація існуючого коду програми для, наприклад виправлення помилок, зазвичай викликає появу більшої кількості збоїв у роботі системи [2]. Тому впровадження якісного регресійного тестування є запорукою раннього виявлення багів застосунку ще на етапі його розробки та імплементації.

Безпосереднє виконання регресійного тестування можливо двома способами: ручним та автоматизованим. Під час ручного регресійного тестування спеціалісти із забезпечення якості програмного забезпечення вручну перевіряють коректність роботи додатку після внесених покращень. Автоматизоване тестування відбувається завдяки раніше створених автоматизованих тестів, що програмно імітують взаємодію реальних користувачів із сайтом. Використання мануального способу для впровадження регресійного тестування має безліч недоліків, серед яких

людський фактор та об’єм часу, що потрібен для його реалізації. Тому більшість IT-проектів використовує автоматизований підхід до виконання повторного тестування своїх програмних продуктів.

Для злагодженої роботи автоматизованих тестів більшість компаній використовує Jenkins. Jenkins – це платформа із відкритим кодом для безперервної інтеграції та доставки. Функціонал додатку дозволяє автоматизувати робочі процеси, налагодити кращу роботу завдяки доступним плагінам та впровадити виконання задач на різних машинах [3].

Використання даного інструменту командою тестувальників для впровадження регресивного тестування має безліч переваг та плюсів. Серед основних – це автоматизація запуску тестів. Під час розробки нового функціоналу додатку та налагодження роботи вже існуючих модулів важливо проводити постійне регресійне тестування, яке допоможе виявити наявні баги в системі. Як було сказано раніше, регресійне тестування можливо провести завдяки наявним тестам, проте аби їх не запускати вручну, спеціалісти із якості програмного забезпечення можуть залучити Jenkins для автоматичного запуску всіх наявних тестів.

Використання Jenkins дозволяє налагодити автоматичний запуск пробігу тестів за розкладом. Наприклад, команда QA може налагодити запуск тестів ввечері, тим самим виконати регресійне тестування автоматизовано вночі, без залучення додаткових ресурсів із боку інженерів. Тим самим на ранок команда буде мати звіт, що міститиме в собі всю інформацію про результати пробігу всіх тестів в системі.

Також завдяки Jenkins можливо налаштувати паралельний пробіг тестів [4]. Великі та складні проекти вимагають створення великої кількості автоматизованих тестів, їх запуск один за одним може зайняти значний об’єм часу для їх виконання. Тому паралельний пробіг тестів зменшить загальний час пробігу збірки та забезпечить швидше виконання регресивного тестування додатку.

Jenkins підтримує використання різноманітних плагінів, що впроваджують зручнішу та ефективнішу інтеграцію платформи в робочі процеси компанії. Наприклад, завдяки наявним плагінам додатку можливо створити зручні звіти з результатами пробігу тестів прямо в Jenkins. Такі плагіни покращують аналіз створюваних звітів, додаючи до них скріншоти, графіки, діаграми і подібного.

Якщо аналіз звіту безпосередньо в Jenkins не є зручним для тестувальника, дана платформа підтримує інтеграцію із більшістю сучасними трекінговими системами, такими як Allure, Jira, Zephyr, TestRail і подібного. Більшість із вище згаданих платформ дозволяють генерацію унікального API для взаємодії їх системою. Jenkins підтримує використання цих API для побудови зв’язку між двома системами, тому цілком реально підняти середовище, яке після закінчення пробігу тестів в Jenkins буде

надсилати результати в інші системи для створення більш зручних та ефективних звітів.

Більше того, завдяки Jenkins можливо управляти базою даних системи, автоматизувавши видалення зайвої інформації в базі, що з'являється після пробігу автоматизованих тестів.

Тож, як бачимо, Jenkins має широке коло можливостей для автоматизації виконання регресійного тестування, від запуску та паралелізації тестів до надсилання звітів в сторонні системи. Використання даного інструменту запроваджує швидше, якісніше та зручніше регресійне тестування всього створюваного додатку.

Список використаних джерел:

1. Software Development Life Cycle. вебсайт. URL: https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/ (дата звернення 03.03.2025).

2. lenford J. M., Badgett T., Sandler C. The Art of Software Testing. 2012. P. 134.

3. Laster B. Jenkins 2: Up and Running. 2018. P. 38-40.

4. Shkil O., Miroshnyk M., Rakhlis D., Trifanov O. Data structures for deductive simulation of HDL conditional operators. *Innovative technologies and scientific solutions for industries*. 2023. No 3(25). Pp. 98–113. DOI: https://doi.org/10.30837/ITSSI.2023.25.098

5. Шкіль О., Рахліс Д., Філіпенко І., Корнієнко В. і Рожнова Т. Автоматизоване проєктування вбудованих систем цифрового оброблення сигналів на платформі SoC. *Сучасний стан наукових досліджень та технологій в промисловості*. 2024. No 1(27). С. 192–203. DOI: https://doi.org/10.30837/ITSSI.2024.27.192