УДК 004.415.538

Жарков А. В.
Овчинников К. В.

ТЕХНОЛОГІЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ТЕСТУВАННЯ РОЗГАЛУЖЕНИХ КЛІЄНТ-СЕРВЕРНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Вінницький національний технічний університет

Анотація
Наведено основні переваги систем автоматизованого тестування. Розроблені тестові класи для програмного забезпечення Kenshoo Social, з використанням середовища NetBeans та веб-додатку Jenkins.
Ключові слова: автоматизоване тестування, програмне забезпечення, клієнтська частина, серверна частина.

Abstract
The main advantages of automated testing systems are given. Developed test classes for Kenshoo Social software, using the NetBeans environment and the Jenkins web application.
Keywords: automated testing, software, front end, back end.

Вступ

Автоматизоване тестування є невід'ємною частиною сучасного підходу до розробки програмного забезпечення (ПЗ). Організація системи автоматичного тестування дозволяє ідентифікувати помилки функціональності і дизайну додатка на всіх етапах його створення[1]. Озвучимо лише найочевидніші переваги, які надає застосування систем автоматизованого тестування. При будь-яких змінах, внесених програмістом в код, запускається тестовий сценарій, який перевіряє коректну роботи попереднього і нового функціоналу додатку. Таким чином, розробник завжди впевнений у тому, що його дії не спричинили помилок в роботі програми. А якщо помилка виникла, то звіт про проходження тестів дозволить точно ідентифікувати місце і причину збою.

Автоматизоване тестування значно скорочує час розробки ПЗ, так як замінює процес ручного тестування. Крім того, автономні тести здатні працювати для різних форм і додатків. Таким чином, один раз написаний тест може служити інструментом для пошуку помилок багаторазово. Випадковий підхід генерації даних, що використовуються в автоматизованих тестах, дозволяє знайти непередбачені помилки. Дуже часто машина здатна згенерувати такий тестовий приклад, який людина-тестувальник не змогла б навіть уявити.

Історія проходження тестів на різних етапах розробки дає цінну статистичну інформацію, а саме: середню кількість помилок при реалізації однієї форми, проходження одного спринту (якщо мова йде про компанії, що підтримують гнучкі методології розробки) за весь час створення продукту; час, витрачений на виправлення помилок; типи помилок, їх процентне співвідношення. Такого роду інформація може бути представлена замовнику, а також активно використовуватися при подальшій розробці, щоб на її підставі поліпшити процес, усуваючи недоліки попередніх релізів. Крім того, інформація може бути використана при проходженні перевірок ПЗ на відповідність певному міжнародному стандарту[2].

Не дивлячись на всі перераховані переваги автоматизованого тестування, в більшості IT компаній практика ручного тестування не перетворилося в атавізм. Це відбувається через те, що створення подібної системи є нетривіальною і дуже специфічною задачею. Залежно від функціональності розроблюваного ПЗ, мов і засобів програмування, задача організації системи кардинально змінюється і ускладнюється[3].

Результати дослідження

В тезисі представлені розроблені тестові класи для програмного забезпечення Kenshoo Social. Дане програмне забезпечення – це складна клієнт серверна система, яка складається з клієнтської частини – front end, та серверної частини – back end. Для розробки і контролю за тестами було використано середовище NetBeans та веб додаток Jenkins. NetBeans використовується на етапі створення, відлагодження тестів. Jenkins – це веб - додаток на який мають доступ розробники. За його

допомогою кожен хто приймає участь в проекті може слідкувати за станом проекту в цілому, за виконанням тестів та рівнем покриття тестами програмного коду. Back end, серверна частина програмного забезпечення Kenshoo Social, складається з шести модулів (рисунок 1). Кожен з них має свою незмінну структуру (рисунок 2).

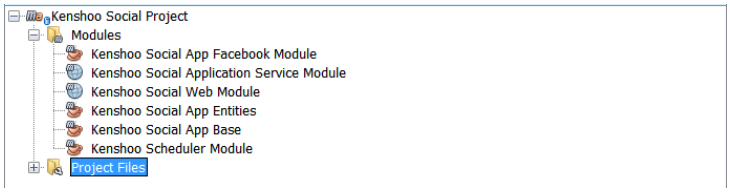


Рисунок 1 – Модулі програмного забезпечення Kenshoo Social

В кожному модулі обов'язково присутні дві папки – Source Packages та Test Packages. Перша містить функціонал програмного забезпечення, а друга тести до кожного класу цього програмного забезпечення. Таким чином забезпечується стабільна робота продукту в процесі розробки, швидке реагування на зміну в коді, відлагодження проблемних місць. Тестове покриття дає змогу розробникам виявити слабкі місця.

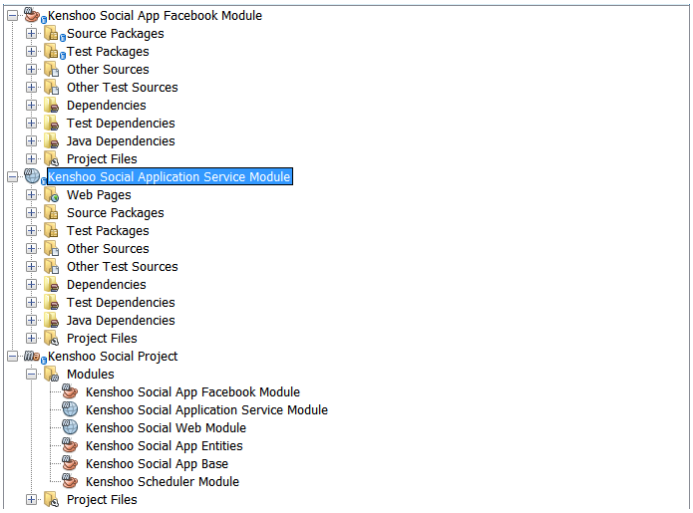


Рисунок 2 – Основна структура кожного модуля

Даний програмний продукт має декілька різновидів тестування – mock tests, integration tests. Перший вид тестує програмний продукт локально. Другий вид з використанням дійсних значень, які записані в реальній базі даних. В цьому випадку відбувається тестування на реальних вхідних даних з реальною базою даних, а отриманий результат звіряється з очікуваним.

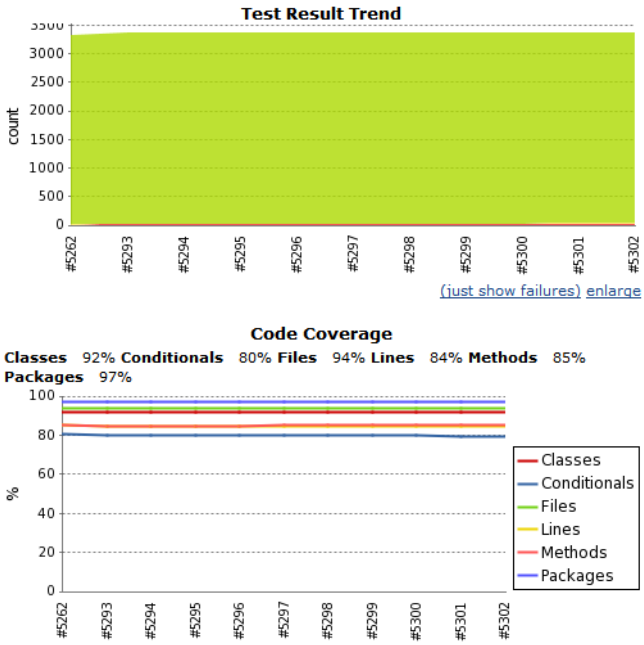
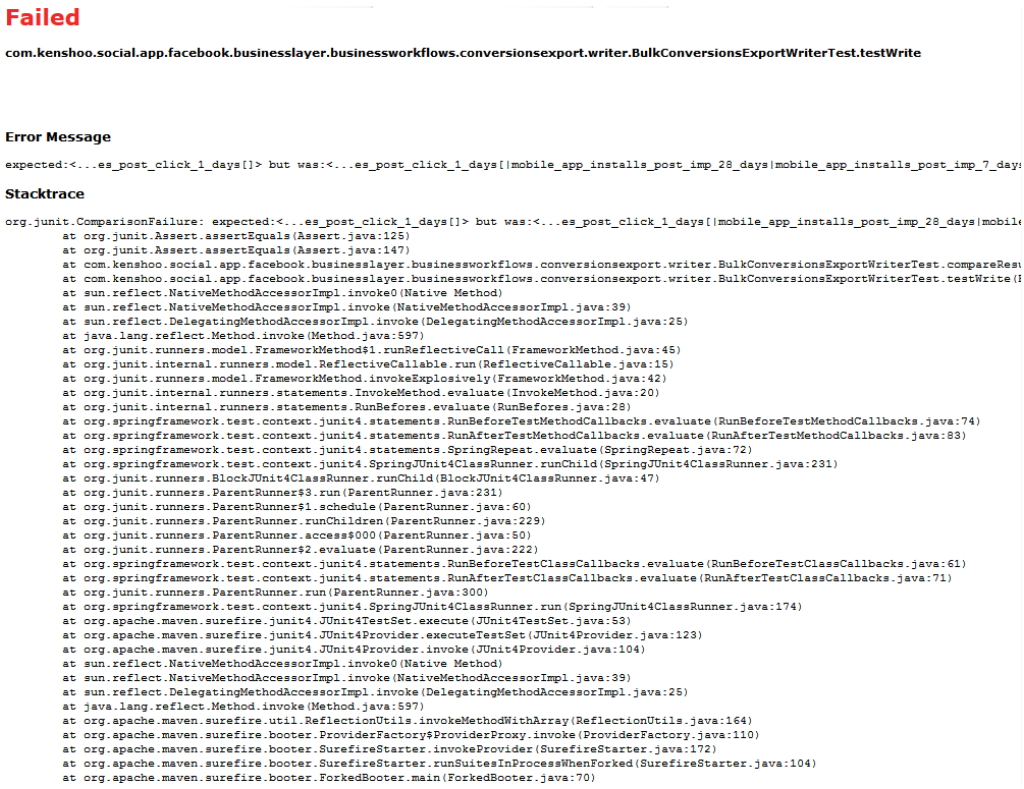


Рисунок 3 – Графіки, які відображають рівень покриття коду тестами

Для зручності управління проектом було використано додаток Jenkins (рисунок 4). Всі дані про проходження тестів, покриття тестами, готовність проекту відображаються на сервері, на якому білдиться проект. В колонці Build History ми бачимо останні 11 білдів. На рисунку 4 зображені дані останнього білда під номером 5302. Вкладка Latest Test Result підказує нам що 1 тест не пройшов, отже там виникла помилка. На рисунку 3 зображено графіки покриття коду тестами. В середньому покриття тестами становить 80%. Можна бути впевненим, що будь яка зміна в коді відобразиться на тестах.

Рисунок 4 – Додаток Jenkins. Результат проходження тестів

Перейдемо по гіперпосиланню Latest Test Result (рисунок 5). Тут відображаються stacktrace всіх тестів які не пройшли. Stacktrace – це інформація про помилку, включаючи назву помилки в якому місці коду вона відбулась і виконані перед цим кроки. Отже, наша помилка має назву ComparisonFailure. Вона виникає тоді коли ми порівнюємо результат, який повертається з методу, що тестується, з очікуваним значенням.



Failed

com.kenshoo.social.app.facebook.businesslayer.businessworkflows.conversionsexport.writer.BulkConversionsExportWriterTest.testWrite

Error Message

```
expected:<...es_post_click_1_days[]> but was:<...es_post_click_1_days[|mobile_app_installs_post_imp_28_days|mobile_app_installs_post_imp_7_days
```

Stacktrace

```
org.junit.ComparisonFailure: expected:<...es_post_click_1_days[]> but was:<...es_post_click_1_days[|mobile_app_installs_post_imp_28_days|mobile
        at org.junit.Assert.assertEquals(Assert.java:125)
        at org.junit.Assert.assertEquals(Assert.java:147)
        at com.kenshoo.social.app.facebook.businesslayer.businessworkflows.conversionsexport.writer.BulkConversionsExportWriterTest.compareRes
        at com.kenshoo.social.app.facebook.businesslayer.businessworkflows.conversionsexport.writer.BulkConversionsExportWriterTest.testWrite(
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:597)
        at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:45)
        at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:15)
        at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:42)
        at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:20)
        at org.junit.internal.runners.statements.RunBefores.evaluate(RunBefores.java:28)
        at org.springframework.test.context.junit4.statements.RunBeforeTestMethodCallbacks.evaluate(RunBeforeTestMethodCallbacks.java:74)
        at org.springframework.test.context.junit4.statements.RunAfterTestMethodCallbacks.evaluate(RunAfterTestMethodCallbacks.java:83)
        at org.springframework.test.context.junit4.statements.SpringRepeat.evaluate(SpringRepeat.java:72)
        at org.springframework.test.context.junit4.SpringJUnit4ClassRunner.runChild(SpringJUnit4ClassRunner.java:231)
        at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:47)
        at org.junit.runners.ParentRunner$3.run(ParentRunner.java:231)
        at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:60)
        at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:229)
        at org.junit.runners.ParentRunner.access$000(ParentRunner.java:50)
        at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:222)
        at org.springframework.test.context.junit4.statements.RunBeforeTestClassCallbacks.evaluate(RunBeforeTestClassCallbacks.java:61)
        at org.springframework.test.context.junit4.statements.RunAfterTestClassCallbacks.evaluate(RunAfterTestClassCallbacks.java:71)
        at org.junit.runners.ParentRunner.run(ParentRunner.java:300)
        at org.springframework.test.context.junit4.SpringJUnit4ClassRunner.run(SpringJUnit4ClassRunner.java:174)
        at org.apache.maven.surefire.junit4.JUnit4TestSet.execute(JUnit4TestSet.java:53)
        at org.apache.maven.surefire.junit4.JUnit4Provider.executeTestSet(JUnit4Provider.java:123)
        at org.apache.maven.surefire.junit4.JUnit4Provider.invoke(JUnit4Provider.java:104)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:597)
        at org.apache.maven.surefire.util.ReflectionUtils.invokeMethodWithArray(ReflectionUtils.java:164)
        at org.apache.maven.surefire.booter.ProviderFactory$ProviderProxy.invoke(ProviderFactory.java:110)
        at org.apache.maven.surefire.booter.SurefireStarter.invokeProvider(SurefireStarter.java:172)
        at org.apache.maven.surefire.booter.SurefireStarter.runSuitesInProcessWhenForked(SurefireStarter.java:104)
        at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:70)
```

Рисунок 5 – Результат переходу по ссилці Latest Test Result

Така помилка може означати тільки одне – відбулись зміни в коді. Для виправлення помилки необхідно звернутись до документації яка описує даний метод, що тестується. Якщо результат, що повертається, є правильним, відповідає документації, змінюємо значення з яким перевіряємо.

Висновки

В тезисі представлені розроблені тестові класи для програмного забезпечення Kenshoo Social. Дане програмне забезпечення – це складна клієнт серверна система, яка складається з клієнтської частини – front end, та серверної частини – back end. Для розробки і контролю за тестами було використано середовище NetBeans та веб-додаток Jenkins. NetBeans використовується на етапі створення і відлагодження тестів. Jenkins – це веб-додаток на який мають доступ розробники. За його допомогою кожен хто приймає участь в проекті може слідкувати за станом проекту в цілому, за виконанням тестів та рівнем покриття тестами програмного коду.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Система автоматичного тестування програм під ОС Android : веб-сайт URL: https://core.ac.uk/download/pdf/38409131.pdf (дата звернення: 24.02.2021)
2. Тестування : веб-сайт URL: https://learn.ztu.edu.ua/pluginfile.php/6545/mod_resource/content/1/Тема%204.pdf (дата звернення: 24.02.2021)
3. Автоматизоване регресійне тестування обчислювальних алгоритмів на основі нейронних мереж : веб-сайт URL: https://cyberleninka.ru/article/n/avtomatizovane-regresiyne-testuvannya-obchislyuvalnih-algoritmiv-na-osnovi-neyronnih-merezh (дата звернення: 24.02.2021)

Жарков Анатолій Володимирович — студент групи 1АКІТ-17б, факультет комп'ютерних систем і автоматики, Вінницький національний технічний університет, с. Вила, Вінницького р-ну, Вінницької обл., e-mail: fkca.1akitjav@gmail.com

Науковий керівник: ***Овчинников Костянтин Вячеславович*** — канд. техн. наук, доцент кафедри автоматизації і інтелектуальних інформаційних технологій, Вінницький національний технічний університет

Zharkov Anatoliy — Department of Computer System and Automation, Vinnytsia National Technical University, village Vyla, Vinnytsia district, Vinnytsia region, e-mail: fkca.1akitjav@gmail.com

Scientific supervisor: ***Ovchynnykov Kostiantyn V.*** — Cand. Sc. (Eng), Assistant Professor of Automation and Intelligent Information Technology department, Vinnytsia National Technical University, Vinnytsia