УДК 004.9

С.Г. Удовенко[1], Н.О. Миронова[2], Т.В. Федорончак[2], К.К. Верещак[2]

[1] *Харківський національний економічний університет імені Семена Кузнеця, Харків*
[2] *Запорізький національний технічний університет, Запоріжжя*

ВИКОРИСТАННЯ ШАБЛОНІВ АВТОМАТИЧНОГО ТЕСТУВАННЯ В ПРОЕКТАХ З РОЗРОБКИ ВЕБ-ДОДАТКІВ

Запропоновано метод побудови ієрархічної структури багаторазових універсальних тестових шаблонів для заданої предметної області. Запропоновані шаблони використовуються для генерації тестових сценаріїв і можуть швидко адаптуватися до будь-якого типового проекту. Розроблено систему автоматичного тестування, що базується на сучасному фреймворку Selenium у поєднанні з мовою Python та бібліотекою тестування Python Unittest. Проведене дослідження показало практичну значимість нового методу генерації тестових сценаріїв завдяки скороченню витрат часу на складання тестів та проведення тестування.

Ключові слова: *система автоматичного тестування, веб-додаток, генерація тестових сценаріїв, тестування на основі моделей.*

Вступ

Важливим етапом розробки програмного забезпечення є етап тестування. Складність і різноманіття функцій сучасних програм вимагає застосування спеціальних методів аналізу стану і працездатності програмного забезпечення протягом усього циклу розробки, впровадження та супроводу. У широкому сенсі, тестування – це одна з технік контролю якості, яка включає планування, складання тестів, безпосереднє виконання тестування та аналіз отриманих результатів. На сьогодні забезпечення якості програмних продуктів, що контролюється на етапі тестування, є одним з провідних напрямів в IT-індустрії.

Автоматизація процесів тестування представляє особливий інтерес, тому що дозволяє значно скоротити час проходження системою повного комплексу тестових сценаріїв та мінімізувати при цьому кількість трудових витрат на цей процес. Автоматизовані системи тестування вже стали невід'ємною частиною процесів розробки програмних продуктів. Особливу увагу слід приділити області веб-розробки, яка стрімко розвивається та стає все більш поширеною областю в інженерії програмного забезпечення.

Складність тестування у веб-проектах обумовлюється наступними характеристиками: короткими циклами розробки, розподіленою багатоланковою архітектурою, складністю та динамічністю поведінки, кросплатформністю та кросбраузерністю [1, 2]. У проектах при розробці веб-додатків широко застосовується ітеративний підхід, що обумовлює доцільність використання регресійного тестування, тобто повторного тестування частин системи, що розробляється. Тому автоматичне тестування дозволяє значно полегшити та пришвидшити цей процес і скоротити витрати часу та людських ресурсів на нього [2, 3]. В області веб-розробки на сьогоднішній день існує велика кількість різних систем автоматизації тестування, що відрізняються функціональними можливостями, ступенем стандартизації, опрацюванням призначеного для користувача інтерфейсу, правилами ліцензування [4, 5].

Незважаючи на те, що в області тестування веб-додатків здійснюється велика кількість теоретичних та практичних досліджень, в ній досі залишається чимало важливих питань, що потребують вирішення [3, 6]. Одним з них є побудова систем автоматичного тестування веб-додатків та генерація тестових сценаріїв для них. Таким чином, розробка нового методу генерації тестових сценаріїв для автоматичного тестування веб-додатків є актуальною та доцільною.

Аналіз останніх досліджень і публікацій і постановка проблеми

До актуальних та нетривіальних задач тестування слід віднести складання тестових сценаріїв, які брали б до уваги специфічні вимоги до веб-додатків та дозволяли б ефективно виявляти помилки в розроблених кодах. Одним з різновидів автоматичного тестування є тестування на основі моделей, при якому варіанти тестування будуються із деякої моделі веб-додатку, що тестується [1, 2, 7].

Рис. 1 відображає загальне представлення процесу тестування на основі моделей [7]. Згідно з вимогами до веб-додатку, здійснюється розроблення моделі, яка відображає ті особливості системи, що підлягають тестуванню. Ця модель дозволяє генерувати набори варіантів сценаріїв тестування, які визначають умови, вхідні дані, очікувані результати роботи системи та оцінювання результатів тестування з використанням компаратора. Порівняння в компараторі очікуваних та реальних результатів дає можливість зробити висновок про наявність помилок в системі, надійність та відповідність системи моделі згідно з вимогами до неї. Крім того, може бути зроблено висновок про необхідність модифікації існуючої моделі чи додаткової генерації варіантів тестування.

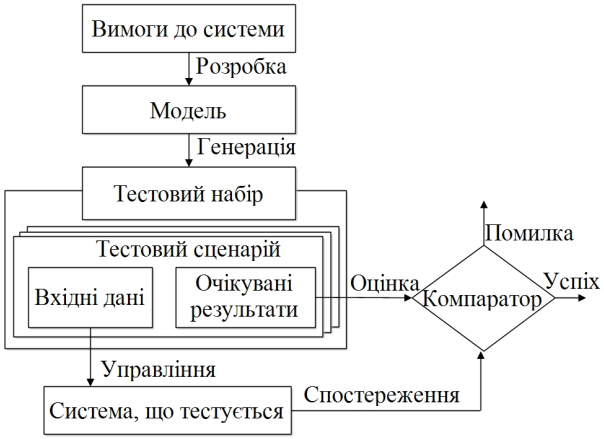

Рис. 1. Тестування на основі моделі веб-додатку

Ще однією особливістю тестування веб-додатків є необхідність зберігати інформацію про стан сесії користувача. В результаті цього тестовий сценарій має враховувати послідовність тестових випадків, тобто декількох дій в системі.

Найчастіше моделі веб-додатків описують аспекти функціональних можливостей системи та відображають бажану поведінку [7]. Найбільш поширеними є підходи до моделювання веб-додатків, пов'язані з побудовою навігаційних графів, UML моделей та FSM моделей веб-додатку. Такі моделі можуть будуватися автоматично, наприклад, за допомогою веб-краулерів чи реверсивної інженерії.

Наприклад, в роботі [8] запропоновано будувати граф навігації по сторінкам веб-додатку та генерувати згідно з цим графом тестові маршрути, що використовують специфіку міжсторінкових переходів. В роботі [9] цю ідею було розвинуто. В ній використовується перевірка коректності навігації по веб-додатку для тестування функціональних вимог. Навігаційна структура додатку при цьому моделюється засобами UML-діаграм та XML-нотації. Однак недоліком такого підходу є дуже велика кількість комбінацій переходів між сторінками в складних веб-додатках.

В роботі [10] автори розширили традиційні методи тестування потоків даних на область веб-розробки. Вони запропонували розглядати компоненти веб-додатку як об'єкти, а для генерації тестових сценаріїв використовувати модель, що відображає потоки даних між цими об'єктами. В подальших своїх роботах вони розробили метод генерації тестів на основі комбінацій різних типів моделей: об'єктної діаграми, діаграми станів та навігаційної діаграми переходу по сторінках [11]. Однак в якості недоліку можна зазначити, що ці моделі для побудови потребують аналізу сирцевого коду веб-додатку.

В роботі [12] використовуються UML діаграми прецедентів та діаграми послідовності для створення ієрархічної моделі переходів між варіантами використання. В роботі [13] для моделювання поведінки веб-додатку використано традиційну UML діаграму послідовностей та розширену веб-діаграму, яка дозволяє відобразити архітектуру клієнтської частини додатку та залежності між веб-сторінками. В роботі [14] для опису структури веб-додатку застосовано моделі скінченого автомату (FSM) з обмеженнями. Автори використали ієрархічний підхід для моделювання складних веб-додатків. В моделях скінченого автомату кожна логічна сторінка представляє стан веб-додатку, а дії користувача та введені дані призводять до переходів між цими станами. В модель при цьому вводяться обмеження для зменшення простору станів та переходів між ними. Такий підхід знайшов подальший розвиток в роботах інших дослідників. В якості прикладів можна навести використання ймовірнісних моделей скінченого автомату [15], в яких переходи між станами асоційовані з імовірностями цих переходів, або застосування генетичних алгоритмів для генерації тестів з FSM моделі [16].

Як видно, зазвичай процес створення тестових сценаріїв є індивідуальним для кожного окремо взятого проекту та може здійснюватися паралельно з розробкою програмного продукту. Однак специфічною особливістю проектів з розробки веб-додатків є багаторазове повторення типових рішень, що мають мінімум функціональних відмінностей і розрізняються лише в своїй візуальній складовій. В якості прикладів таких проектів можна навести розробку інтернет-магазину, блогу, порталу новин тощо. За таких умов розробка автоматичних тестових сценаріїв для кожного окремого проекту є недоцільною з точки зору раціонального використання ресурсів.

Метою статті є розробка методу побудови ієрархічної структури універсальних шаблонів тестування предметної області для веб-додатків. Така структура шаблонів повинна швидко адаптуватися для будь-якого типового проекту та заданої предметної області. Розроблений метод повинен бути реалізований в системі автоматичного тестування веб-додатків.

Метод побудови ієрархічної структури універсальних шаблонів тестування

Суть концепції, що пропонується, полягає в побудові ієрархічної структури універсальних тестових сценаріїв (рис. 2). На верхньому рівні цієї структури знаходяться узагальнені тестові сценарії, характерні для заданої предметної області. Найнижчим рівнем ієрархії є область взаємодії з елементами візуального інтерфейсу конкретного проекту. При правильно вибудуваній ієрархії адаптація розроблених шаблонних сценаріїв під проект здійснюється саме на цьому рівні. Така ієрархічна структура тестових сценаріїв дозволяє здійснювати тестування на основі моделей, де враховуються функціональні вимоги до веб-додатків обраної типової предметної області та налаштування для конкретної реалізації. Після налаштування моделі під конкретний проект генеруються кінцеві тестові набори.

Ієрархічна структура успадкованих тестових сценаріїв передбачає поділ на узагальнене ядро та

змінну частину. Ядро містить опис тестових випадків, який залишається незмінним від проекту до проекту в рамках певної предметної області. Змінна частина містить відсилання до певних точок входу проекту, таких як адреси сторінок, посилання на активні елементи інтерфейсу користувача, посилання на блоки з певним типом інформації. Таку змінну частину в подальшому будемо називати словником проекту.

Рис. 2. Ієрархічна структура універсальних тестових сценаріїв

Таким чином, структура системи автоматичного тестування з використанням таких шаблонів матиме вигляд, представлений на рис. 3.

Ядро системи тестування в свою чергу ділиться на дві основні складові: набір шаблонів тестових сценаріїв і бібліотеку операторів. Такий поділ потрібен для досягнення необхідного рівня абстракції і виключення дублювання коду в разі повторення одних і тих же кроків в різних тестових сценаріях.

Тестовий сценарій – це сукупність тестових випадків, що забезпечують комплексну перевірку працездатності окремого аспекту системи, що тестується. Кожен тестовий сценарій зберігається в окремому файлі і є автономною функціональною одиницею. Склад і функціональні характеристики тестових сценаріїв залежать від конкретної предметної області.

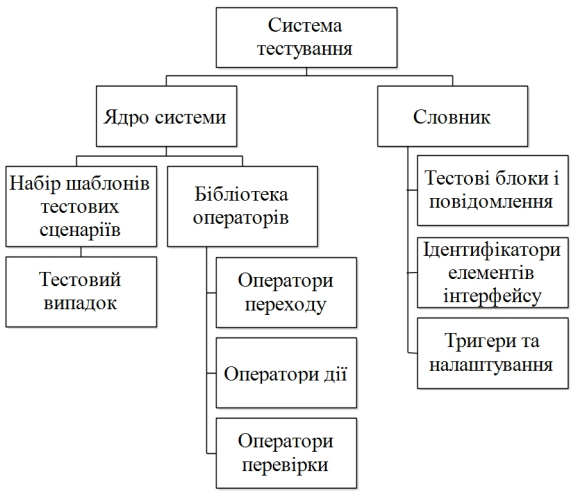

Рис. 3. Структура системи автоматичного тестування з використанням шаблонів

Тестовий випадок – це послідовність дій гіпотетичного користувача, що впливає на поточний стан тестованої системи певним чином і має конкретний очікуваний результат. Порівняння очікуваного результату з фактичним є основним фактором перевірки адекватності поведінки системи, що тестується. До компонентів системи пред'являються дві вимоги:

– універсальність тестових сценаріїв ядра, тобто шаблонів, від яких залежить повнота покриття тестовими випадками предметної області;

– мінімізація кількості елементів словника проекту, що безпосередньо впливає на обсяг роботи з формування словника під новий проект.

Бібліотека операторів є сукупністю повторюваних дій, які можуть бути використані в різних тестових сценаріях. Оператори безпосередньо взаємодіють з даними зі словника проекту з метою отримання доступу до необхідних елементів інтерфейсу користувача. Оператори діляться на три основні категорії:

– оператори переходу, що використовуються для переміщення між різними сторінками вебдодатка. Основна вимога до даних операторів – повна імітація дії гіпотетичного користувача. Система тестування може користуватися лише тими елементами управління, які доступні реальному користувачеві. Зазвичай будь-якої тестовий випадок починається з виклику оператора переходу до відповідного даному випадку розділу системи;

– оператори дії, представляють собою сукупність варіантів взаємодії з органами управління користувальницького інтерфейсу (кліки по активним елементам, введення даних з клавіатури та інші дії користувача);

– оператори перевірки, що представляють собою сукупність різних методів перевірки поточного стану системи. До таких методів перевірки відносяться: наявність на сторінці певного елемента, вміст текстового блоку, порівняння поточного значення певних параметрів системи з раніше збереженими значеннями та інше.

Словник проекту використовується для налаштування тестової системи під потреби конкретного проекту. Він складається зі специфічних для даного проекту налаштувань: найменувань та посилань на елементи графічного інтерфейсу, відповідей в результаті виконання операцій, опису структур даних аналізованих об'єктів та інше. В загальному вигляді словник проекту можна розділити на три основні групи:

– ідентифікатори елементів інтерфейсу (для ідентифікації елементів призначеного для користувача інтерфейсу використовуються CSS-селектори);

– текстові блоки і повідомлення (набір текстових даних, характерних для тих чи інших тестових випадків, за наявністю або відсутністю яких тестова система може визначити чи є виконання тестового випадку успішним);

– тригери та налаштування (параметри, які задають правила проведення тестів та характеризуються булевими значеннями, що визначають тестові випадки даного проекту і схему аналізу).

На основі запропонованої концепції можна сформулювати метод побудови ієрархічної структури універсальних шаблонів тестування предметної області, що передбачає реалізацію наступних етапів:

Етап 1. Вибір предметної області, типової для розробки веб-додатків, та визначення основних функціональних вимог до веб-додатку.

Етап 2. Формування ядра системи, тобто опис універсальних тестових сценаріїв, що моделюють визначені функціональні вимоги, та створення бібліотеки операторів, що визначають сукупність часто повторюваних дій.

Етап 3. Визначення словника, тобто змінної частини, що відноситься до конкретної реалізації, а саме: адрес веб-сторінок, ідентифікаторів елементів користувацького інтерфейсу, параметрів проекту тощо.

Етап 4. Інтеграція розроблених шаблонів до програмної системи автоматичного тестування.

Етап 5. Налаштування словника під конкретний проект розробки веб-додатку.

Етап 6. Тестування веб-додатку за допомогою згенерованих із шаблонів кінцевих тестових сценаріїв.

Програмна реалізація системи автоматичного тестування

Можна сформулювати такі основні вимоги до програмної платформи для реалізації запропонованого методу:

– наявність гнучкої системи команд і операторів;

– можливість автоматичної адаптації до мінливих умов тестування за допомогою циклів і розгалужень;

– підтримка ієрархічних структур і успадкування;

– підтримка оголошень функцій і базові можливості повторного використання коду.

За своєю суттю ці вимоги зводяться до того, що обрана для реалізації програмна платформа повинна мати базові можливості типової скриптової мови програмування. Однак слід взяти до уваги той факт, що цільовими користувачами такої платформи є не програмісти, а фахівці в області забезпечення якості. Отже розроблені шаблони тестування повинні бути простими та зручними для користувачів, які не є фахівцями в області розробки програмного забезпечення і не мають глибоких знань мов програмування.

З урахуванням перелічених вимог був проведений аналіз існуючих на сьогоднішній день засобів автоматизації тестування з метою вибору оптимальної платформи для реалізації методу. У число розглянутих програмних пакетів потрапили такі системи: Watir, Selenium, Ranorex, TestComplete, IBM Rational Robot. За результатами аналізу можливостей цих систем оптимальною платформою для реалізації поставлених завдань слід вважати вільний фреймворк для тестування веб-додатків Selenium в поєднанні з мовою програмування Python. Selenium є проектом з відкритим вихідним кодом, що має велике співтовариство користувачів. Він розробляється з 2004 року і на сьогодні є найбільш поширеним засобом для тестування веб-додатків і, крім того, є базою для побудови багатьох інших засобів тестування, в тому числі комерційних [5, 17].

Фреймворк Selenium складається з двох частин: Selenium WebDriver та Selenium IDE.

Selenium WebDriver є драйвером браузера, тобто програмною бібліотекою, яка дозволяє розробляти програми, що керують поведінкою браузера, імітуючи дії користувача при роботі з веб-додатками. Selenium WebDriver був прийнятий за основу при розробці стандарту інтерфейсу для управління браузером організацією W3C. По суті цей продукт на сьогоднішній день є стандартом для інтерфейсів управління браузерами. Для взаємодії з браузерами WebDriver за допомогою API безпосередньо викликає команди браузера, завдяки чому реалізується спосіб взаємодії з браузером, що є близьким до дій реального користувача.

Selenium IDE – це зручний інструмент для швидкої розробки невеликих тестових сценаріїв в браузері Firefox, що використовується для розвідувального тестування та пошуку помилок. Selenium IDE підтримує запис сценаріїв тестування в графічному режимі з подальшим експортом в одну із запропонованих мов програмування: Java, Python, Ruby. В той же час реалізувати складну логіку або перевірку в сценаріях IDE майже неможливо.

Як Selenium WebDriver, так і Selenium IDE, є безкоштовними кросплатформними програмними засобами і задовольняють всім вимогам до програмної платформи для реалізації універсальних шаблонів автоматичного тестування веб-додатків.

До недоліків проекту Selenium можна віднести:

– більш примітивний інтерфейс записи сценаріїв в порівнянні з комерційними платними аналогами;

– відсутність вбудованої системи складання звітів за результатами тестування.

Розглянемо архітектуру запропонованої системи автоматичного тестування з використанням універсальних шаблонів тестових сценаріїв. Система складається з наступних компонентів: Selenium WebDriver, Selenium IDE та Python Selenium API. Фреймворк Selenium в поєднанні з гнучкістю і широкими можливостями мови програмування Python надає всі необхідні умови для реалізації запропонованої концепції системи тестування веб-додатків. У складі мови програмування Python є вбудована бібліотека Unit testing framwork (скорочено unittest) для створення тестових сценаріїв. Ця бібліотека може бути використана як основа для створення шаблонів відповідно до запропонованого методу.

Для роботи з WebDriver необхідно мати 3 основних програмних компоненти (рис. 4):

– браузер, роботу якого користувач хоче автоматизувати, що встановлений на певній ОС і має свої налаштування (за замовчуванням або згідно з індивідуальними вимогами);

– драйвер браузера (веб-сервер, який запускає браузер і надсилає йому команди управління);

– тестовий скрипт, який містить набір команд мовою Python для драйвера браузера з використанням клієнтських бібліотек для прив'язки скрипту до інтерфейсу Selenium WebDriver.

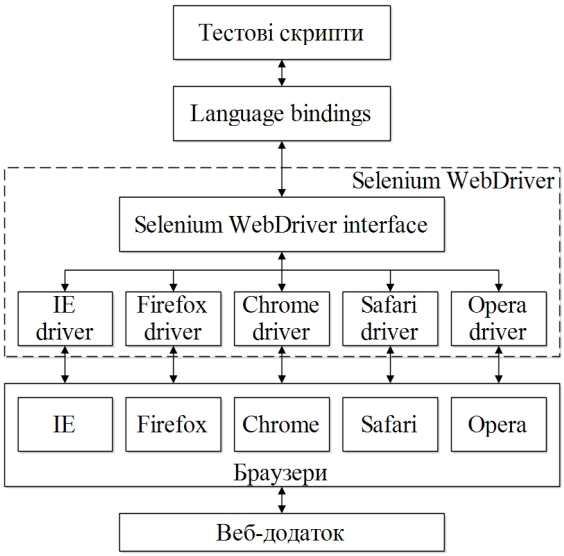

Рис. 4. Архітектура системи
з використанням Selenium WebDriver

В мові Selenium використовується набір команд для драйвера браузера. Послідовність таких команд є тестовим сценарієм. У Selenium існує широкий вибір команд для максимально повного тестування веб-додатків. З використанням цих команд можна виконати прості перевірки наявності елементів інтерфейсу користувача або певного контенту, роботи гіперпосилань, полів введення, меню, табличних даних тощо. Команди Selenium також підтримують перевірку розмірів вікна, позиції курсору миші, роботу з діалоговими вікнами, елементами Ajax, обробку подій та інші функції, необхідні для тестування сучасних веб-додатків. Прив'язка мови Selenium до мови Python здійснюється за допомогою простого API для написання тестів функціональності або тестів відповідності вимогам.

Використаний в системі фреймворк тестування юнітів Python Unittest підтримує автоматизацію тестів, використання загального коду для налаштування і виконання тестів, об'єднання тестів в колекції та виведення інформації. Система з використанням Python Unittest підтримує деякі важливі концепції за допомогою компонентів, наведених на рис. 5: *випробувальний стенд* (test fixture), що здійснює підготовку до виконання одного або кількох тестів, а також всі необхідні дії з очищення (наприклад, створення тимчасових баз даних або запуск серверного процесу); *тестовий випадок* (test case) – мінімальний юніт тестування, що перевіряє відповіді для різних наборів даних; *набір тестів* (test suite), що є набором тестових випадків, що використовується для об'єднання тестів, які повинні бути виконані спільно; *виконавець тестів* (test runner), що є компонентом, який керує виконанням тестів і надає користувачеві результат.

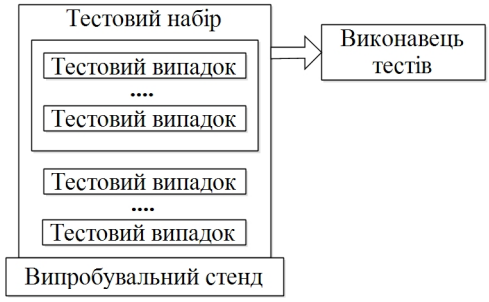

Рис. 5. Архітектура системи
з використанням Python Unittest

Виконавець може використовувати графічний або текстовий інтерфейс або повертати спеціальне значення, яке повідомляє про результати виконання тестів. Система сценаріїв передбачає два основні режими експлуатації:

– запуск на локальній машині тестувальника (у такому випадку локальна машина є тестовим стендом і запуск тестових сценаріїв здійснюється з командного рядка в консолі операційної системи, а результат роботи також буде відображено в консолі);

– запуск на серверному тестовому стенді в Selenium Grid (в цьому випадку запуск і збір результатів тестування здійснюється планувальником тестового стенда);

– локальний запуск тестових сценаріїв здійснюється командою python –m unittest <назва_ тестового сценарію> (при бажанні можна уточнити конкретний тестовий випадок в рамках заданого сценарію).

Після виконання тестового сценарію в консолі відображається статистика проходження тестів по кожному тестовому випадку та підсумкові значення кількості позитивних та негативних результатів тестів. У разі невдалого завершення тестових сценаріїв система наводить детальний опис відповідної помилки.

Приклад реалізації та практичне дослідження

В результаті практичного дослідження була розроблена та реалізована система шаблонів тестових сценаріїв для автоматизації тестування веб-додатків типу «інтернет-магазин». Приклад застосування даних тестових сценаріїв системи шаблонів наведено для магазину, що знаходиться за адресою: https://tochkazp.com.ua.

Можна виділити наступні шаблони тестових сценаріїв, що відображають функціональні вимоги до веб-додатку (на прикладі предметної області інтернет-магазину), наведено в табл. 1. У табл. 2 наведено деякі з визначених операторів для тестування інтернет-магазинів. Приклад декількох типових параметрів зі словника проекту для інтернет-магазину наведені в табл. 3.

Таблиця 1

Шаблони тестових сценаріїв

Назва сценарію	Опис сценарію
test_catalog	Тестування коректності відображення сторінок каталогу та наявності всієї інформації про товари
test_catalog_filter	Тестування системи фільтрів і сортувань в каталозі товарів
test_cart	Тестування можливості додавання товарів в корзину та зміни кількості товарів в корзині
test_order	Тестування системи замовлень товару з різними параметрами доставки, оплати, контактними даними
test_registration	Тестування системи реєстрації, авторизація, відновлення паролів
test_profile	Тестування сторінок профілю користувача

Таблиця 2

Оператори системи

Оператор		Опис оператору
Оператори переходу	goto_index	Перехід на головну сторінку магазину
	goto_category	Перехід на сторінку категорії товарів
	goto_product_page	Перехід на сторінку товару
	goto_cart	Перехід на сторінку перегляду вмісту кошика
	goto_order	Перехід на сторінку оформлення замовлення
	goto_login	Перехід на сторінку авторизації
	goto_profile	Перехід на сторінку профілю
Оператори дії	auth_login	Вхід на сайт
	auth_logout	Вихід з сайту
	set_filter_price	Установка значень фільтра за ціною
	set_filter_select	Установка значення довільного фільтра зі списком вибору
	submit_filter	Застосування встановлених фільтрів
	cart_add_list	Додавання товару в кошик в списку товарів
	cart_add_page	Додавання товару в кошик на сторінці товару
	cart_remove	Видалення товару з кошика
	order_fill_contact	Заповнення контактної інформації на сторінці оформлення замовлення
	order_fill_payment	Заповнення інформації про спосіб оплати на сторінці оформлення замовлення
	order_fill_delivery	Заповнення інформації про спосіб доставки на сторінці оформлення замовлення
	order_submit	Відправка даних про замовлення
Оператори перевірки	check_product_list_item	Перевірка структури даних про товари на сторінках зі списком товарів
	check_product_page	Перевірка структури даних про товар на сторінці товару
	check_cart_update	Перевірка зміни кількості товарів в кошику
	check_order_success	Перевірка наявності повідомлення про успішне оформлення замовлення
	check_auth	Перевірка стану авторизації

Таблиця 3

Параметри із словника проекту

Параметр	Опис
ELEMENT_LIST_PRODUCT	Селектор блоку товару в списку товарів
ELEMENT_LIST_PRODUCT_PRICE	Селектор елемента з ціною товару в списку товарів
FILTER_AVAILABLE	Ознака використання в поточному проекті фільтрів по параметрам
FILTER_PRICE_FROM	Селектор елемента управління фільтром за ціною (нижній поріг)
FILTER_PRICE_TO	Селектор елемента управління фільтром за ціною (верхній поріг)
FILTER_FORM	Селектор форми з фільтрами
FILTER_SUBMIT	Селектор кнопки застосування параметрів фільтрів
CART_COUNT	Селектор блоку з даними про кількість товарів у кошику
CART_LINK	Селектор посилання на сторінку з вмістом кошика

Для нового проекту вхідними даними для розробленої системи автоматичного тестування є файл словника проекту. Словник керує поведінкою тестових сценаріїв: визначає які функціональні вимоги підлягають тестуванню, параметри доступу до управляючих елементів інтерфейсу користувача та очікувані результати проходження тестів.

Вихідними даними системи тестування є звіт про проходження тестових сценаріїв. Звіт формується в кількох форматах: спливаюче повідомлення в інтерфейсі IDE PyCharm, лог тестів в консолі IDE або у вигляді файлів, в разі запуску системи тестування з передачею спеціальних параметрів. На рис. 6 наведено приклад виконання тестового сценарію, що складається з 3-х тестових випадків. Результат перевірки кожного тестового випадку відображається крапкою у випадку вдалого проходження, чи літерою «F» (від англ. Fail) – у разі невиконання умов тестового випадку. Також для невдалого тесту відображається детальний опис місця виникнення помилки. Крім наведеного прикладу тестування інтернет-магазину https://tochkazp.com.ua тестуванню підлягали ще два проекти з розробки інтернет-магазинів, що на даний час ще не введено в експлуатацію, а знаходяться на етапі реалізації. Як було відзначено, особливістю проектів з розробки веб-додатків є багаторазове повторення типових рішень, які мають досить схожий функціонал і відрізняються візуальним представленням та користувальницьким інтерфейсом. Це дозволило авторам запропонувати концепцію побу-

дови універсальних шаблонів для автоматичного тестування. Для запропонованої концепції було розроблено метод побудови ієрархічної структури універсальних шаблонів тестування предметної області, обрано сучасну програмну платформу для реалізації, розроблено архітектуру системи автоматичного тестування та виконано розробку системи шаблонів автоматичних тестів. Робота створеної системи досліджена на прикладі тестування інтернет-магазинів. В результаті експериментального дослідження визначено, що використання розробленої системи дозволяє скоротити витрати часу на тестування веб-додатку у 8–10 разів в порівнянні з ручним тестуванням. Розроблені шаблони тестування є універсальними і можуть адаптуватися під різні реалізації інтернет-магазинів згідно з конкретними параметрами проектів. Отримані результати свідчать про практичну значимість виконаної розробки і доцільність подальших досліджень в даному напрямку.

Подальші кроки розвитку даної системи можуть бути спрямовані на вдосконалення системи побудови звітів про проходження тестів на основі аналізу відгуків та потреб реальних користувачів системи. Також в процесі введення в експлуатацію може бути розширена система операторів ядра системи після отримання більшого обсягу інформації про статистику використання неврахованих в процесі розробки поширених модулів і функцій в даній предметній області.

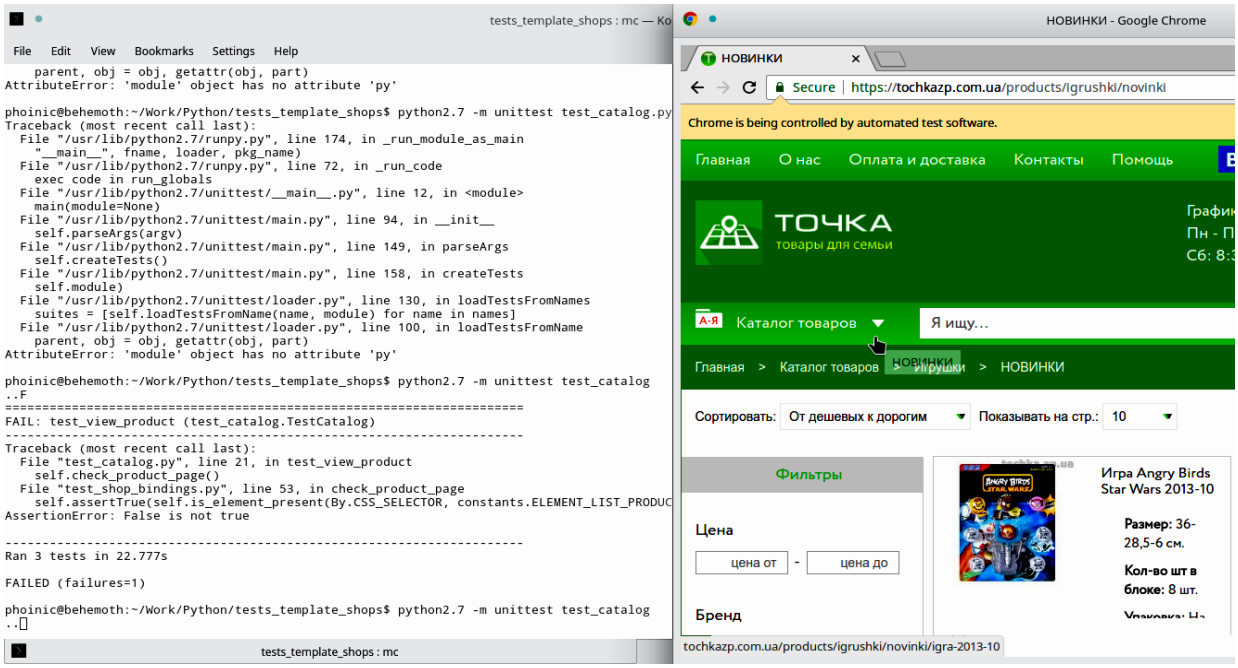

Рис. 6. Приклад роботи системи автоматичного тестування

Висновки

В роботі розглянуто актуальне на сьогоднішній день завдання автоматичного тестування веб-додатків. Проведений огляд сучасних програмних засобів та методів автоматизації тестування веб-додатків показав, що одним з невирішених питань в цій галузі є спрощення та прискорення процесу генерації тестових сценаріїв. Найбільш поширеним підходом до цього питання є тестування на основі моделей. Однак існуючі методи направлені на побудову окремих моделей для кожного нового проекту. Такий підхід не є раціональним з точки зору використання ресурсів для IT-компаній, які регулярно виконують типові веб-проекти. Визначені особливості дозволили запропонувати концепцію шаблонів автоматичного тестування предметної області.

В роботі розроблено метод побудови ієрархічної структури універсальних шаблонів тестування предметної області. Ці шаблони можна повторно використовувати та швидко адаптувати під будь-який типовий проект розробки веб-додатків в заданій предметній області. Розроблений метод використовує структуру, яка є моделлю, де враховуються функціональні вимоги до веб-додатків обраної типової предметної області та налаштування для конкретної реалізації. Така ієрархічна структура успадкованих тестових сценаріїв поділяється на узагальнене ядро та змінну частину, яка називається словником. Кінцеві тестові набори сценаріїв автоматично генеруються після налаштування під конкретний проект нижнього рівня ієрархічної моделі.

За результатами аналізу можливостей сучасних засобів автоматизації тестування програмного забезпечення було обрано оптимальну платформу для реалізації системи автоматичного тестування з використанням запропонованого методу. Розроблена система використовує вільний фреймворк для тестування веб-додатків Selenium в поєднанні з мовою програмування Python та фреймворк тестування юнітів Python Unittest. Розроблена система є цілком працездатною та дозволяє прискорити процес генерації тестових сценаріїв завдяки запропонованому методу. Створені шаблони дозволяють моделювати функціо-

нальні вимоги предметної області веб-додатку у вигляді наслідуваної ієрархічної структури.

Для практичного дослідження розробленого методу було вирішено прикладне завдання тестування інтернет-магазинів. Проведене дослідження показало практичну значимість запропонованої концепції через скорочення витрат часу на складання тестів і проведення тестування веб-додатків. Розроблена система автоматичного тестування є легкою в використанні для фахівців в області забезпечення якості: налаштування шаблонів під проект та генерація тестових сценаріїв є простим та зручним для користувачів.

Список літератури

1. Li, Y. F. Two decades of web application testing – A survey of recent advances [Text] / Y. F. Li, P. K. Das, D. L. Dowe // Information Systems. – 2014. – Vol. 43. – P. 20–54.

2. Sampath, S. Advances in web application testing, 2010-2014 [Text] / S. Sampath, S. Sprenkle // Advances in Computers. – 2016. – Vol. 101. – P. 155–191.

3. Dogan, S. Web application testing: A systematic literature review [Text] / S. Dogan, A. Betin-Can, V. Garousi // Journal of Systems and Software. – 2014. – Vol. 91. – P. 174–201.

4. Monier, M. Evaluation of automated web testing tools [Text] / M. Monier, M. M. El-mahdy // International Journal of Computer Applications Technology and Research. – 2015. – Vol. 4, Issue 5. – P. 405–408.

5. Kumar, Y. Comparative study of automated testing tools: Selenium, SoapUI, HP Unified Functional Testing and Test Complete [Text] / Y. Kumar // Journal of Emerging Technologies and Innovative Research. – 2015. – Vol. 2, N. 9. – P. 42–48.

6. Garousi, V. A systematic mapping study of web application testing [Text] / V. Garousi, A. Mesbah, A. Betin-Can, S. Mirshokraie // Information and Software Technology. – 2013. – Vol. 55, Issue 8. – P. 1374–1396.

7. Rafique, N. Model based testing in web applications / N. Rafique, N. Rashid, S. Awan, Z. Nayyar // Int. Journal of Scientific Eng. and Research. – 2014. – Vol. 2, Issue 1. – P. 56–60.

8. Tung, Y. H. A novel approach to automatic test case generation for web applications [Text] / Y. H. Tung, S. S. Tseng, T. J. Lee, J. F. Weng // 10th International Conference on Quality. – 2010. – P. 399–404.

9. Garcia, B. Automated functional testing based on the navigation of web applications [Text] / B. Garcia, J. C. Duenas // Proceedings of the 7th International Workshop on Automated Specification and Verification of Web Systems, EPTCS 61. – 2011. – P. 49–65.

10. Liu, C. H. Object-based data flow testing of web applications [Text] / C.H. Liu, D.C. Kung, P. Hsia // Proc. of First Asia-Pacific Conference on Quality Software. – 2000. – P. 7–16.

11. Kung, D. C. An object-oriented web test model for testing Web applications [Text] / D.C. Kung, C.H. Liu, P. Hsia // Twelfth International Conference on Software Engineering and Knowledge Engineering. – 2000. – P. 537–542.

12. Li, L. A UML-based approach to testing web applications [Text] / L. Li, H. Miao, Z. Qian // Int. Symp. on Computer Science and Comp. Technology. – 2008. – P. 397–40.

13. Suhag, V. Model based test cases generation for web applications [Text] / V. Suhag, R. Bhatia // International Journal of Computer Applications. – 2014. – Vol. 92. – P. 23–31.

14. Andrews, A. Testing web applications by modeling with FSMs [Text] / A. Andrews, J. Offutt, R. Alexander // Software and System Modeling. – 2005. – Vol. 4, n. 3. – P. 326–345.

15. Qian, Z. Towards testing web applications: a PFSM-based approach / Z. Qian, H. Miao // Advanced Materials Research. – 2011. – Vol. 1. – P. 220–224.

16. Kalaji, A.S. An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models / A.S. Kalaji, R.M. Hierons, S. Swift // Information and Software Technology. – 2011. – Vol. 53. – P. 1297–1318.

17. Satheesh, A. Comparative study of open source automated web testing tools: Selenium and Sahi [Text] / A. Satheesh, M. Singh // Indian Journal of Science and Technology. – 2017. – Vol. 10(13). – P. 1–9.

Надійшла до редколегії 14.08.2017

Рецензент: д-р техн. наук, проф. Є.В. Бодянський, Харківський національний університет радіоелектроніки, Харків.

ПРИМЕНЕНИЕ ШАБЛОНОВ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ В ПРОЕКТАХ ПО РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ

С.Г. Удовенко, Н.А. Миронова, Т.В. Федорончак, К.К. Верещак

Предложен метод построения иерархической структуры многоразовых универсальных тестовых шаблонов для заданной предметной области. Предложенные шаблоны используются для генерации тестовых сценариев и могут быстро адаптироваться к любому типовому проекту. Разработана система автоматического тестирования, основанная на современном фреймворке Selenium в сочетании с языком Python и библиотекой тестирования Python Unittest. Проведенное исследование показало практическую значимость нового метода генерации тестовых сценариев благодаря сокращению затрат времени на составление тестов и проведения тестирования.

***Ключевые слова**: система автоматического тестирования, веб-приложение, генерация тестовых сценариев, тестирование на основе моделей*

USE OF AUTOMATED TESTING TEMPLATES FOR WEB APPLICATIONS DEVELOPMENT PROJECTS

S.G. Udovenko, N.O. Myronova, T.V. Fedoronchak, K.K. Vereschak

A specific feature of web application development projects is the multiple repetition of typical solutions that have rather similar functionality and differ in their visual representation and user interface. This has allowed proposing the concept of universal templates for automated testing system. The purpose of the work is to develop a method for constructing a hierarchical structure of reusable universal testing templates for given subject area, which can quickly adapt to any typical project. The proposed method can be referred to as model based testing technique, since the proposed templates model the functional requirements of the subject area of the web application in the form of the inherited hierarchical structure. The developed automated testing system is based on the up-to-date Selenium and Python Unittest frameworks and uses developed method of hierarchical structure of reusable universal testing templates for generation of test scenarios. The conducted research showed the practical significance of the proposed concept through the speed-up and reduction of time and human resources costs for the preparation of tests and their conduction.

***Keywords**: automatic test system, web application, test cases generation, model based testing.*