3. http://www.cisco.com

4. Унифицированные коммуникации [Електронний ресурс] – Режим доступу : http://www.ixbt.com/comm/unicomm.shtml

5. Люгер Д. Искусственный интеллект: стратегии и методы решения сложных проблем / Люгер Д. ; [4-е узд. – М. : Издательский дом «Вильямс», 2003. – 864 с.

Статтю представляє: д.т.н. Поморова О.В.
Надійшла 14.2.2012 р.

УДК. 584. 326

М.П. ДИВАК, В.В ЧИЧА, Л.С. ОЛІЯРНИК
Тернопільський національний економічний університет

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕСТУВАННЯ ГРАФІЧНОГО КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ (GUI)

Розглянуто сучасні спеціалізовані засоби для тестування графічного користувацького інтерфейсу. Досліджено властивості існуючих систем, і на цій основі запропоновано та обґрунтовано власну автоматизовану систему тестування графічного користувацького інтерфейсу веб-додатків.

In the paper modern and specialized means for testing graphic user interface are analyzed. The properties of existing systems were investigated, and on this basis is proposed and proved the own automated system for testing graphical user interface of Web applications.

Ключові слова: графічний користувацький інтерфейс, тестовий сценарій, візуалізатор, ключове слово, покриття коду, діаграма компонент, програмне забезпечення (ПЗ).

Постановка проблеми у загальному вигляді та її зв'язки із важливими науковими і практичними завданнями. Останнім часом набули широкого поширення засоби автоматизованого тестування користувацького інтерфейсу, які знижують навантаження на тестувальника-оператора. Автоматизація тестування користувацького інтерфейсу – природний і необхідний спосіб тестування ПЗ (особливо на етапі регресійного тестування), оскільки скорочує витрати компанії-розробника.

Більшість відомих компаній, таких як SmartBear Software, HP, IBM, Telerik, ThoughtWorks, є розробниками спеціальних тестових «фреймворків», які імітують дії користувача, але вказані системи ще не достатньо відтестовані і мають не повний функціонал.

Аналіз останніх досліджень і публікацій, у яких започатковано розв'язання даної проблеми. Питання автоматизації тестування ПЗ у вітчизняній та зарубіжній літературі розглядалося багаторазово. Тема тестування програмного забезпечення одержала свій розвиток у працях відомих вітчизняних та іноземних вчених: Харченка В.С., Гуляєва В.А., Локазюка В.М., Савченка Ю.Г., Согомоняна Е.С., Романкевича О.М., Кривулі Г.Ф., Дрозда О.В., Ліпаєва В.В., Майєрса Г., Канера Сема, Соммервіла І., Бейзера Б. Проблеми процесу автоматизованого тестування графічного користувацького інтерфейсу розглянуто у працях Калинова А.Я., Косачова А.С., Посипкіна М.А., Соколова А.А.[1], Р. Робінсона [2]. У статті [1] викладено метод автоматичної генерації набору тестів для графічного інтерфейсу користувача, модельованого детермінованим кінцевим автоматом за допомогою UML діаграм. Метод полягає у побудові обходу графа станів системи із застосуванням не надлишкового алгоритму обходу та компіляції побудованого обходу у тестовий набір. У статті [2] наведено основні порівняльні характеристики програмних продуктів для тестування GUI, а саме: WinRunner, QA Run, Silk Test, Visual Test та Robot.

Аналіз вище зазначених праць свідчить про те, що фахівці у своїх дослідженнях здебільшого розглядають окремі аспекти проблеми автоматизованого тестування GUI. Зокрема, у статті [1] процес автоматичної генерації тестів є трудомістким і передбачає використання двох програмних продуктів, а саме Rational Rose та Rational Robot. На даний час вказаний продукт є застарілим і не здатен підтримувати більшість сучасних браузерів. Що до засобів, зазначених у статті [2], то використання їх також є трудомістким – часові витрати на «прогон» тестів вручну співпадають із часом їх написання.

Вимоги до програмної системи. Метою досліджень сучасних систем для автоматизованого тестування GUI веб-додатків (Ranorex Studio 3.1.1, TestComplete 8, Teleric Test Studio 2011.2 та Silenium IDE) є встановлення їхніх основних особливостей функціонування і на цій основі створення власної тестової системи, яка відзначається підвищеною продуктивністю. Поряд із цим, однією із основних не функціональних вимог створюваної системи є «кросбраузерність».

Існує широкий спектр інструментів для автоматизованого тестування GUI веб-додатків. Оптимально організована система повинна відповідати таким вимогам:

1. Запис та відтворення тестів
2. Відлагодження помилок
3. Безоплатність
4. Підтримка тестування веб-додатків:
    1) Internet Explorer

2) Mozilla Firefox
3) Google Chrome

5. Наявність візуалізатора
6. Можливість використання ключових слів при написанні тесту
7. Інтеграція з Visual Studio
8. Генерація коду на C #[3].

Результати досліджень існуючих систем автоматизованого тестування наведено у табл.1., де «+» позначено наявність даної можливості у вказаній системі, а «-» - її відсутність.

Таблиця 1

Порівняльна характеристика сучасних систем автоматизованого тестування

Система тестування / Характеристики	TestComplete	Ranorex Studio	Test Studio	Selenium IDE
Запис та відтворення тестів	+	+	+	+
Відлагодження помилки	-	+	+	+
Безоплатність	-	-	-	+
Підтримка тестування веб-додатків: Internet Explorer Mozilla Firefox Google Chrome	+ + + -	+ + + -	+ + + +	+ - + -
Наявність візуалізатора	+	+	+	-
Можливість використання ключових слів при написанні тесту	+	+	+	-
Інтеграція з Visual Studio	-	+	+	-
Генерація коду на C #	-	+	+	-

Згідно із табл. 1, жодна із систем, що досліджувалися, не відповідає сформульованим вимогам. Вказані додатки корисні лише своєю універсальністю, тобто вони за достатньо короткий момент часу здатні створити велику кількість тестів за допомогою функції record&playback. Але отримані скріпти доволі часто мають проблеми зі стійкістю і є достатньо складними.

Тому метою даної роботи є розробка продукту, який би найкращим чином відповідав вимогам, наведеним у табл.1.

Кількісні характеристики якості програмної системи для автоматизованого тестування GUI. У стандарті ISO/IEC 9126 сформульовано вимоги щодо якості ПЗ. Проте, враховуючи специфіку вказаного ПЗ, необхідно сформулювати кількісні характеристики оцінки якості тестових систем цих оцінок якості:

1. Покриття тестових сценаріїв.
2. Покриття коду.
3. Досяжність завершеності тестування.

Показник покриття тестових сценаріїв характеризує кількість покритих елементів програмного забезпечення в результаті тестування. Найчастіше даний показник визначають у відсотках. Вказаний показник обчислюємо за формулою:

$$TV = (V - DV)/V \quad (1)$$

де TV – оцінка ступеня тестування (вимірюється у відсотках); V – максимальна кількість тестів, що покривають роботу усієї системи, що тестується; DV – максимальна кількість тестів, які залишилися не покритими після прогону усієї множини тестів T = $\{t_i: i=1,..,n\}$. Величина DV монотонно зменшується від 1 до 0 із збільшенням кількості тестів.

Показник покриття коду відображає відсоток вихідного коду програми, який був протестований, тобто це щільність покриття тестами коду, що виконується. Покриття коду включає в себе декілька видів, а саме: покриття операторів, умов, шляхів та функцій. У нашому випадку тестування сайту використовуватимемо покриття операторів, тобто чи кожен рядок коду був виконаним і відтестованим [3]. Даний показник обчислюємо лише тоді, коли є доступ до структури (коду) системи, що тестується.

Показник досяжності завершеності тестування обчислюємо з метою визначення коли потрібно припинити процес тестування, тобто чи програма є достатньо відтестованою й готовою до функціонування за призначенням. Критерій завершеності тестування має такий вигляд:

$$TV > L,$$

де L – рівень тестованості ($0 < L < 1$) [2].

Архітектура та алгоритм функціонування. Перш ніж ознайомитися з роботою системи, слід звернути увагу на сайт, який буде тестуватися з використанням даної програми.

Сайт являє собою MVC-додаток (фреймворк ASP.NET MVC 3). MVC спрямований на відділення бізнес-логіки від користувальницького інтерфейсу, щоб розробники могли легко змінювати окремі частини додатку, не вносячи зміни в інші. В архітектурі MVC модель надає дані і правила бізнес-логіки, представлення відповідає за користувальницький інтерфейс (наприклад, текст, поля вводу), а контролер забезпечує взаємодію між моделлю та представленням. Розуміння роботи сайту потрібно для розробки тестових сценаріїв та дослідження самого тестування.

Сайт продажу квитків «Event» реалізує наступні функції (в дужках позначено роль користувача):

- Реєстрація користувача («User»)
- Видалення користувача («Administrator»)
- Створення події («Administrator»)
- Редагування події («Administrator»)
- Видалення користувача («Administrator»)
- Резервування квитка користувачем («User»)

Перед виконанням будь-яких дій на сайті, необхідно авторизуватися.

Особливості роботи розробленої програми для тестування такі: програма при запуску виконує послідовність дій для перевірки користувацького інтерфейсу у трьох браузерах (Mozilla Firefox, Google Chrome, Internet Explorer). Крім технічного характеру, програма має також пізнавальний характер – дозволяє користувачу ознайомитися як працювати з сайтом. Особливістю даної програми є те, що усі дії виконуються автоматично. Користувачу потрібно лише перевірити доступ до Інтернет, запустити дану програму, і подивитися результати виконання. Блок-схему функціонування програми та її компонент наведено на рис. 1 та рис. 2.

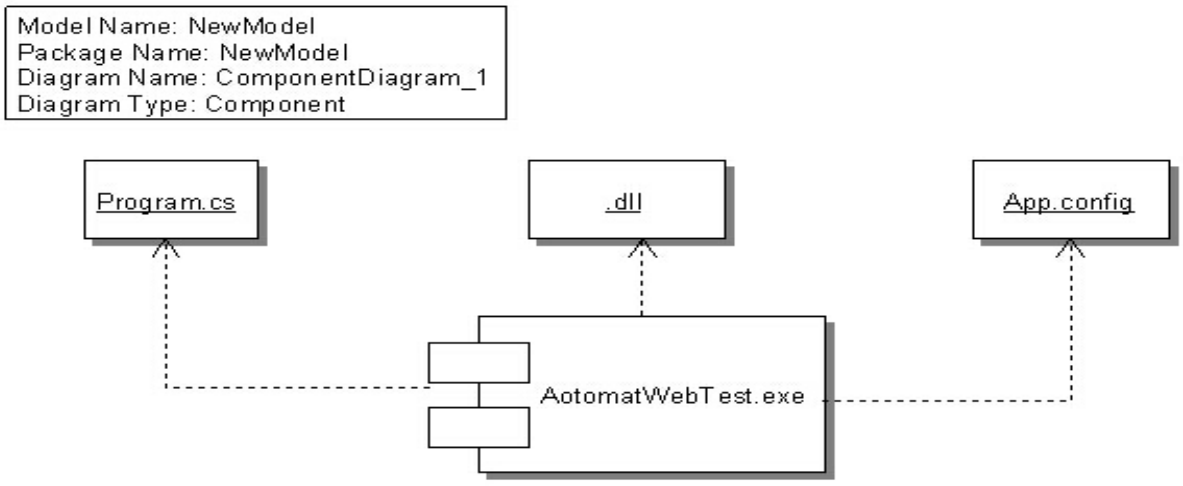

Рис. 1. UML-діаграма компонент

Діаграма компонентів описує особливості фізичного представлення системи. Вона визначає архітектуру розроблюваної системи, встановивши залежність між програмними компонентами [5]. Наявність подібної залежності (рис. 1) означає, що внесення змін до вихідного тексту програми, динамічної бібліотеки чи файлу конфігурації призводить до змін самої компоненти.

Блок-схема, яка представлена на рис. 2, представляє роботу системи автоматизованого тестування. Тестування відбувається у трьох браузерах і при виникненні помилки в тексті, система виводить текст помилки й припиняє свою роботу. У протилежному випадку система після виконання усіх тестів виводить результати й завершує свою роботу.

Дослідження ефективності запропонованої системи. Для тестування будь-якого програмного продукту для тестування важливо не лише наявність багатьох властивостей, які покращують його роботу, але й висока продуктивність даної системи – це можливість за найкоротший час виявити найбільшу кількість помилок.

Так, як згідно з табл. 1 кожна система має функцію запису й відтворення тестів, у ході виконання роботи у кожній системі було проведено запис тестових сценаріїв, і навмисно було зроблено помилки. Відсоток виявлення помилок кожною системою представлено на рис. 3.

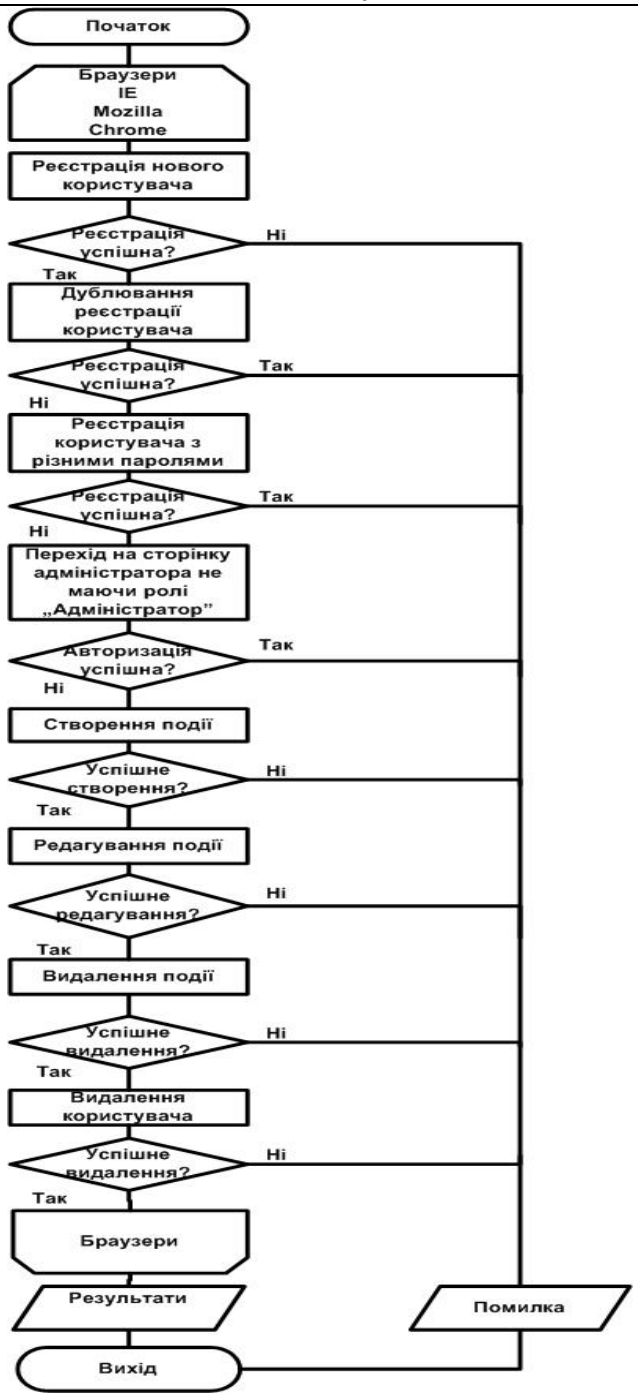

Рис. 2. Узагальнена блок-схема алгоритму програми

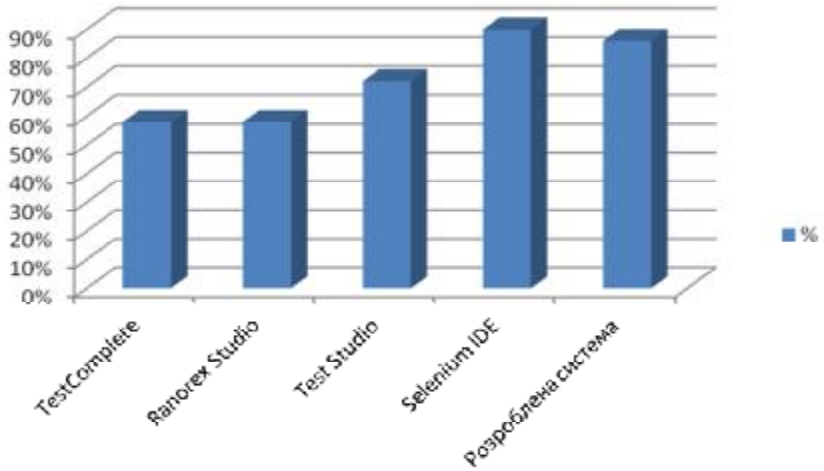

Рис. 3. Відсоток виявлення помилок кожною із досліджених систем

Згідно з проведеним дослідженням, у кожній із систем було проведено запис дев'яти тестових сценаріїв (*V*=9). У семи сценаріях було навмисно зроблено помилки, тобто в семи випадках із дев'яти мало б виникнути «падіння» тесту. Система TestComplete виявила лише чотири помилки, Ranorex Studio – теж чотири, Test Studio – п'ять, Silenium IDE – усі сім помилок, але звіт по одній з них не відповідав дійсності «падіння» тесту. Що ж до розробленої системи, то вона виявила шість помилок із семи. У ході дослідження також обчислено показник покриття тестових сценаріїв, покриття коду та обраховано критерій завершеності тестування. Згідно з проведеними обчисленнями, автоматизоване тестування для даного сайту є доцільним: покриття майже усіх тестових сценаріїв (DV=9-1), Загальна кількість рядків коду – 472, з них протестовано 295. Показник покриття коду – 63% та показник завершеності тестування – близько 89%.

Висновки

У ході виконання роботи було досліджено чотири системи автоматизованого тестування GUI і створено нову систему, яка уможливлює автоматичне виконання тестування сайту продажу квитків через Інтернет, тобто система автоматично виконує функції взаємодії через GUI користувача. Показано, що запропонована система за критеріями якості переважає існуючі системи тестування.

Література

1. Автоматическая генерация тестов для графического пользовательского интерфейса по UML диаграммам действий / А.Я. Калинов, А.С. Косачёв, М.А. Посыпкин, А.А. Соколов // Труды Института системного программирования РАН. – 2004. – Т. 8. – Ч. 1.

2. Robinson Ray, AUTOMATION TEST TOOLS, Date Created: 1st March 2001, Last Updated: 11th Sept 2001.

3. www.wikipedia.org

4. Котляров В.П. Основы тестирования програмного обеспечения / В.П. Котляров, Т.В. Колякова. – М. : Бином, 2006. – 285 с.

5. Weilkiens T. Systems Engineering with SysMLUML Modeling, Analysis, Design. – Denise E. M. Penrose, 2007. – 320.

Рецензент: дт.н. Троцишин І.В.
Надійшла 13.2.2012 р.

УДК 004.492.3

С.М. ЛИСЕНКО, А.В. КРАСІЙ, В.В. МЕЛЬНИК
Хмельницький національний університет

ПОБУДОВА ПРОЦЕСУ ВИЯВЛЕННЯ ТРОЯНСЬКИХ ПРОГРАМ

В роботі досліджено достовірність сучасних засобів антивірусного діагностування та виявлено недоліки їх функціонування. Розроблено модель життєвого циклу троянських. Розглянуто особливості побудови процесу діагностування комп'ютерних систем на наявність троянських програм у режимах монітора та сканера.

In the article the main principles of antiviral diagnosis are proposed. Life circle Trojans model was developed. Main ideas of Trojan detection in monitor and scanner modes are presented.

Ключові слова: троянські програми, антивірусне діагностування комп'ютерних систем.

Вступ

Об'єднання комп'ютерних систем (КС) в локальні мережі та підключення їх до глобальної мережі Internet породжує багато проблем, пов'язаних з їх функціонуванням та використанням. Досить значні проблеми для роботи КС в локальних мережах створюють комп'ютерні мережні віруси, зокрема, їх наявність приводить до неправильного функціонування програмного та апаратного забезпечення.

Проблемі антивірусного діагностування комп'ютерних систем, що функціонують у режимі віддаленого доступу, коли ймовірність проникнення в систему шкідливих програмних об'єктів з боку віддалених комп'ютерних систем особливо висока, приділяється значна увага. Аналіз останніх результатів антивірусного діагностування КС показує динамічний ріст кількості комп'ютерних вірусних програм – програм чи деякої сукупності виконуваного коду, які можуть створювати свої копії, впроваджувати їх у файли, системні області комп'ютерної системи, зберігати здатність до розповсюдження, виконувати деструктивні дії. Серед них особливе місце займає окрема множина вірусних програм – троянські програми, які на відміну від класичних вірусних програм проникають у комп'ютерні системи з метою викрадення конфіденційної інформації, що становлять особливу небезпеку [1] і при цьому не створюють своїх копій.

Проведений аналіз АПЗ показав, що усі вони мають засоби діагностування КС на наявність троянських програм з використанням ІТ на основі сигнатурного аналізу, контрольних сум та евристичних аналізаторів. Проте деталі евристичних аналізаторів діагностування нових ТП недоступні та закриті для дослідження та вивчення. З огляду на це оцінка достовірності та ефективності антивірусного діагностування може бути здійснена шляхом дослідження результатів їх тестування.

Для оцінки достовірності антивірусного діагностування сучасних АПЗ дослідження результатів