DOI: https://doi.org/10.32839/2304-5809/2019-11-75-144
УДК 004.415.5

Єгорова О.В., Бичок В.П.
Черкаський державний технологічний університет

ПРОГРАМНІ ЗАСОБИ ДЛЯ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Анотація. У статті розглядаються особливості використання методу білої скриньки для тестування програмного забезпечення. Наведено основні різновиди методу білої скриньки. Наведено основні напрями автоматизації тестування програмного забезпечення. Виконано аналітичний огляд систем автоматизованого тестування програмного забезпечення. Інформаційні системи для тестування програмного забезпечення допомагають керувати процесом тестування, відслідковуванням помилок та формуванням звітності. Більшість таких систем є кросплатформеною, має відкритий вхідний код, зручний користувацький інтерфейс та гарно інтегрується з іншими системами, проте потребує наявності у тестувальника навичок програмування для написання спеціальних фреймів і бібліотек.

Ключові слова: тестування, програмне забезпечення, біла скринька, код.

Yehorova Olha, Bychok Vladyslav
Cherkasy State Technological University

APPLICATIONS FOR SOFTWARE TESTING

Summary. The paper the features of using the white box method to test software was considered. The white box test method of testing software is based on the assumption that the tester knows the internal structure of the program, investigates the internal elements of the program and the relationships between them. The object of testing is not the external but internal behavior of the program. This technique allows to detect implementation errors such as poor management of the code system by analyzing the internal work of the software. The white box software testing method can be applied at the integration, modular, and system levels. In the general case, testing software using the white box method can detect errors in hidden code when removing unnecessary lines, provides the opportunity to use side effects and ensure completeness of testing by writing a test script. The main variants of the white box method are given: testing of program control streams, testing of program data flows and mutation testing. Modern research in the field of software testing automation is devoted to the use of algorithmic models, neural networks and test language processing methods. Test methods for program control flow are based on the use of test completeness criteria. The basic directions of software testing automation are given. Analytical review of information systems for automation of testing management, information systems for automated testing, information systems for cross-browser testing, information systems for load testing, information systems for error tracking and information systems for API testing have done. Software testing information systems help manage the testing, tracking of errors and generate reports. Most of these systems are cross platform, have open source code and user-friendly interface, well integrate with other systems, but tester must have programming skills to write special frames and libraries.

Keywords: testing, software, white box, code.

Постановка проблеми. Перед розробниками програмного забезпечення завжди поставали проблеми впорядкування дій з тестування у вигляді зв'язаного процесу для раціонального розподілу ресурсів, а також прийняття обґрунтованих рішень щодо початку та завершення тестування програмного забезпечення. Одна із передумов виникнення такої проблеми полягає у прагненні виробників програмного забезпечення до скорочення витрат на розробку програного продукту шляхом обмеження використання фінансових і трудових витрат на виконання повноцінного тестування. Іншою передумовою є посилення ймовірності прояву ризику порушення термінів і перевищення вартості тестування, внаслідок обмеженості проектних ресурсів, що негативно впливає не лише на процеси організації і управління тестуванням, але і на оптимальний обсяг тестування та впорядкування цілей тестування.

Аналіз останніх досліджень та публікацій. Протягом декількох років спостерігається значний інтерес до дослідження стратегій тестування програмного забезпечення. Так, в [1] розглядаються проблеми автоматизації тестування програмного забезпечення. Використання алгоритмічних моделей знань для автоматизації тестування програмних продуктів показано в [2]. Метод ідентифікації прихованих помилок програмного забезпечення на основі нейромережевих інформаційних технологій запропоновано в [3]. Аналітичний огляд методів, стратегій та інструментальних засобів для тестування web-орієнтованих системи виконано в [4]. Оптимізації автоматизованого тестування програмного забезпечення із використанням методів обробки мови тестування присвячена робота [5].

Виділення не вирішених раніше частин загальної проблеми. Аналіз проблеми, останніх досліджень та публікацій свідчить про те, що особливості реалізації та виконання тестування програмного забезпечення за принципом білої скриньки із використанням інформаційних системах для автоматизованого тестування програмного забезпечення майже не визначені.

Мета статті. Головною метою цієї статті є визначення особливостей тестування програмного забезпечення із використанням методу білої скриньки та огляд інструментальних засобів для автоматизації тестування програмного забезпечення.

Виклад основного матеріалу. Метод тестування програмного забезпечення за принци-

пом білої скриньки (прозорої, відкритої, скляної скриньки) – це метод тестування внутрішньої поведінки програмного компонента або системи. Використання даного методу базується на припущенні, що тестувальнику відома внутрішня структура програмної системи і результат тестування. Об'єктом дослідження є внутрішні елементи програми і зв'язки між ними. Тестування методом білої скриньки використовується з метою доведення коректності побудови всіх елементів програмної системи та правильності їх взаємодії один з одним.

Існує декілька різновидів методу тестування програмного забезпечення за принципом білої скриньки: тестування потоків керування програми, тестування потоків даних програми та мутаційне тестування [6-9].

Тестування потоків керування програми передбачає представлення логіки програми у вигляді графа, в якому вершини відповідають операторам, а гілки – взаємозв'язкам між ними.

Тестування потоків даних програми орієнтоване на вивчення переходів і взаємозв'язків між вершинами у яких відбувається ініціалізація та використання змінних.

Мутаційне тестування передбачає навмисне внесення помилок до вихідного коду програми з метою порівняння роботи вихідної програми і програми мутанта.

Методи тестування на основі потоку керування програми базуються на використанні критеріїв оцінювання повноти тестування [6-9].

Метод тестування на основі покриття операторів базується на використанні критерію покриття операторів, який передбачає виконання кожного оператора програми щонайменше один раз.

Метод тестування на основі покриття рішень базується на використанні критерію покриття рішень, який передбачає, що кожна гілка алгоритму має бути пройдена хоча б один раз.

Метод тестування на основі покриття шляхів базується на використанні критерію покриття шляхів, який передбачає, що кожна гілка алгоритму має бути пройдена хоча б один раз.

Метод тестування на основі покриття умов базується на використанні критерію покриття умов, який передбачає, що кожна найпростіша умова має бути перевірена на правильних та хибних значеннях хоча б один раз.

Метод тестування на основі покриття рішень/умов базується на використанні критерію покриття умов/рішень, який передбачає розробку тест-кейсів таким чином, щоб результат кожної умови, кожне рішення і кожен оператор виконувалися хоча б один раз.

Метод тестування на основі комбінаторного покриття умов базується на використанні комбінаторного критерію покриття умов/рішень, який вимагає, щоб всі можливі комбінації результатів умов в кожному рішенні, а також кожен оператор виконався хоча б один раз.

При тестуванні програмного із використанням методу білої скриньки мають місце наступні аспекти:

– в більшості випадків кількість помилок є найменшою в «центрі» і найбільшою на «периферії» програми;

– апріорні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними, тому типовим може стати маршрут, за яким модель обчислень буде опрацьована неякісно;

– при записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних);

– деякі результати роботи програми залежать не від вихідних даних, а від внутрішніх станів програми.

Як наслідок, тестування програмного забезпечення із використанням методу білої скриньки дозволяє виявляти помилки в прихованому коді при видаленні зайвих рядків, надає можливість використати побічні ефекти та забезпечити повноту тестування шляхом написання тестового сценарію.

Разом з тим, тестування програмного забезпечення із використанням методу білої скриньки має такі недоліки:

– відносно витратний процес, який потребує залучення висококваліфікованих фахівців;

– залишає багато недосліджених шляхів, оскільки ретельну перевірку всіх можливих прихованих помилок виконати дуже складно;

– залишає непоміченою деяку частина пропущеного коду.

Сьогодні існує декілька напрямів автоматизації тестування програмного забезпечення:

– автоматизація управління тестуванням;

– автоматизоване тестування;

– автоматизація кросбраузерного тестування;

– автоматизація навантажувального тестування;

– автоматизація відслідковування помилок;

– автоматизація тестування API.

До найбільш популярних інструментів для управління тестуванням програмного забезпечення належать TestRail, qTest та PractiTEST.

Інформаційна система TestRail [10] призначена для ручного тестування програмного забезпечення із використання готових чек-листів. TestRail має вбудований редактор клавіатури із зрозумілим JavaScript-інтерфейсом, дозволяє організовувати тестування за методом Drag'n'drop та додавати нові перевірки вже в процесі виконання тестів.

В [11] знаходимо опис інформаційної системи для управління тестуванням програмного забезпечення qTest, що орієнтована на команди, які дотримуються основних принципів DevOpsi Agile. Функції даного програмного продукту зосереджені на виконанні дослідницького та сесійного тестування, плануванні автоматизації тестування з інтеграцією CI-платформ, управлінні гнучкими тестами та формуванні аналітичної звітності.

Віртуальна платформа PractiTEST [12] орієнтована на виконання end-to-end тестування і надає можливість команді розробників та тестувальників відслідковувати в деталях всі етапи складання та перевірки програмного забезпечення. Інформаційна система PractiTEST дозволяє багаторазово використовувати тести і редагувати сценаріїв перевірки при взаємодії із різними вебпродуктами, а також має ефективні засоби візуалізація даних на основі сучасних інформаційних панелей та банерів.

Варто зазначити, що інформаційні системи для управління тестуванням програмного забезпечення qTest і TestRail гарно інтегруються з ін-

шими інструментами тест-менеджменту, зокрема, Jira, Jenkinsi GitHub, а інформаційна система PractiTEST інтегрується із системами відслідковування помилок за допомогою особистого API.

Для автоматизованого тестування програмного забезпечення використовуються інформаційні системи SQUISH, RANOREX, QTP, Selenium, Katalon Studio та Watir. Так, інформаційні системи QTP (Quick Test Professional) [13] і Squish [14] призначені для автоматизованого тестування графічного інтерфейсу, проте QTP орієнтована на виконання автоматизованих регресійних тестів, а Squish – на перевірку людино-машинних інтерфейсів.

Інструменти із відкритим кодом Selenium, Katalon Studio та Watir використовуються для автоматизації процесу тестування веб-додатків.

Найбільш широкий спектр задач реалізовано у фреймворку Selenium [15], який підтримується декількома операційними системи (Windows, Mac, Linux) та багатьма браузерами (Chrome, Firefox, IE, і браузерами Headless). Скрипти для даного фрейму можна написати на таких мовах програмування як Java, Groovy, Python, C#, PHP, Ruby і Perl.

Характерною особливістю інструменту Katalon Studio [16], який походить від фреймворку Seleniumі Appium, є наявність функції Katalon Analytics, яка дозволяє користувачу одержати вичерпну інформацію про перебіг процесу тестування шляхом побудови метрик, діаграм та графіків.

Інформаційна система Watir [17] підтримує функцію керування поточним тестування та інтегрована з інструментами BBD, зокрема, RSpec, Cucumber та Test / Unit, а для автоматизації тестування веб-додатків використовує бібліотеки Ruby.

Більшість інформаційних систем для автоматизованого тестування програмного забезпечення потребують наявності у тестувальнику навичок програмування для написання спеціальних фреймів і бібліотек, які забезпечать виконання окремих функцій в процесі тестування.

Типовими інструменти кросбраузерного тестування є інформаційні системи LAMBDATEST, Browsera та BROWSERSHOTS.

За допомогою масштабованої хмарної кросбраузерної платформи LAMBDATEST [18] можна виконувати тестування сайтів і веб-додатків, архітектура яких взаємодіє із об'єктами хмарної інфраструктури. Функції даного програмного продукту дозволяють автоматизувати тестування веб-додатків за допомогою інформаційної системи Selenium, перевіряти сумісність з інтерактивним браузером Live Interactive Browser та розпаралелювати процес тестування.

Програмні продукти Browsera [19] і BROWSERSHOTS [20] призначені для тестування на сумісність з браузерами. Функції даних програмних засобів спрямовані на перевірку відображення веб-сайтів в структурі браузерів. Крім того, інформаційна система Browsera дозволяє збирати і зберігати всі помилки сценаріїв, одночасно переглядати одразу декілька версій браузерів та в режимі он-лайн виконувати порівняння макетів при відображенні на різних розширеннях.

Інструменти для навантажувальних перевірок призначені для виконання тестування завантаженості і продуктивності під час використання програмних веб-продуктів, які розробляються. Серед них найбільш поширеними є інформаційні системи WebLOAD, WAPT і LoadUI Pro.

Функції програмного продукту WebLOAD [21] спрямовані на створення користувацького навантаження як у хмарі, так і у локальному середовищі. При цьому, якщо тестування відбувається у хмарі, то використовується навантаження від Amazon EC2.

Програмний продукт WAPT [22] використовується для виконання стрес-тестування веборієнтованих інформаційних систем всередині операційної системи Windows.

Програмний продукт LoadUI Pro [23] призначений для виконання навантажувальних випробувань веб-компонентів. Функції даної інформаційної системи орієнтовані на одночасне створення декількох стратегій та реалізацію декількох сценаріїв тестування, а також багаторазове використання тестів SoapUi Pro.

Для відслідковування помилок під час тестування програмного забезпечення використовуються інформаційні системи Redmine, The Bug Genie, BugNET та інші.

Інформаційна система Redmine [24] використовується для контролю за процесом тестування програмного забезпечення. Даний програмний продукт надає багатофункціональний контроль доступу до задач та здатний одночасно підтримувати декілька баз даних.

Програмний продукт The Bug Genie [25] дозволяє відслідковувати помилки у веб-інтерфейсі, складати звіти про проблеми та управляти задачами.

На переваги інформаційні системи для відслідковування помилок під час тестування програмного забезпечення вказують відкритий початковий код та на наявність зручного і зрозумілого графічного інтерфейсу.

Типовими прикладами інструментів тестування API є SoapUI та WebInject. Кросплатформений програмний продукт SoapUI [26] використовується переважно для виконання навантажувального та функціонального тестування. За допомогою інформаційної системи WebInject [27] можна виконувати перевірки окремих компонентів систем, які мають HTTP-інтерфейс, а також регресійні і приймальні тести. Дана система вдало поєднує виконання мобільних і десктопних GUI-тестів з веб-тестуванням.

Висновки і пропозиції. Розглянуто особливості використання методу тестування програмного забезпечення за принципом білої скриньки. Наведено основні різновиди методу білої скриньки. Виконано аналітичний огляд систем автоматизованого тестування програмного забезпечення. Такі інформаційні системи допомагають керувати процесом тестування, відслідковувати помилки та формувати звітність. Як правило, вони є кросплатформенними, мають відкритий вхідний код, зручний користувацький інтерфейс та гарно інтегруються з іншими системами, проте потребують наявності у тестувальника навичок програмування для написання спеціальних фреймів і бібліотек.

Список літератури:

1. Кравчук С.О. Проблеми автоматизації тестування програмного забезпечення. *Актуальні задачі сучасних технологій*: матеріали VII міжнар. наук.-техн. конф. мол. учен. та студ., м. Тернопіль, 28-29 листопада 2018 р. Тернопіль, 2018. С. 95.
2. Буров Є.В. Інтелектуальна система автоматизованого тестування програмного продукту з використанням алгоритмічних моделей. *Вісник Національного університету «Львівська політехніка»*. 2011. № 699: Інформаційні системи та мережі. С. 21–30.
3. Говорущенко Т.О. Проблеми реалізації методу ідентифікації прихованих помилок програмного забезпечення на основі нейромережних інформаційних технологій. *Радіоелектронні і комп'ютерні системи*. 2008. № 7. С. 107–112.
4. Ali K., Xiaoling X. A reliable and an efficient web testing system. *International Journal of Software Engineering & Applications*. 2019. Vol. 10. № 1. P. 1–16. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3341095 (дата звернення: 16.11.2019).
5. Mann M., Sangwan O.P., Tomar P. Automated software test optimization using test language processing. *The International Arab Journal of Information Technology*. 2019. Vol. 16. № 3. P. 348–356.
6. Тамре Л. Введение в тестирование программного обеспечения. Москва : Вильямс, 2003. 359 с.
7. Василенко Н.В. Модели оценки надежности программного обеспечения. *Вестник Новгородского государственного университета им. Ярослава Мудрого*. 2004. № 28. С. 126–132.
8. Бирюков С.В. Анализ стратегий тестирования программного обеспечения. *Известия ЮФУ. Технические науки*. 2008. Вып. 78. № 1. С. 59–63.
9. Канер С., Фолк Д., Кек Нгуен Е. Тестирование программного обеспечения. Киев : ДиаСофт, 2000. 544 с.
10. Test Case Management Software – TestRail. URL: https://www.gurock.com/testrail?utm_source=adwords&utm_medium=cpc&utm_campaign=europe_en_generic&utm_content=testpad&gclid=Cj0KCQiAw4jvBRCJARIsAHYewPPo21kSi2XJH3gafMJ1fmoK-HHv0HBUotizADsmbEKZP1Fw5k-nDH8aAr4QEALw_wcB (дата звернення: 15.11.2019).
11. qTest – Modern Softw are Testing Tools Platform. URL: https://www.qasymphony.com/software-testing-tools/ (дата звернення: 15.11.2019).
12. PractiTEST QA Test Management Tool. URL: https://www.practitest.com/?utm_medium=listings&utm_source=guru&utm_campaign=testing+tools (дата звернення: 15.11.2019).
13. Automate Functional Testing QTP. URL: https://www.microfocus.com/ru-ru/products/uft-one/overview (дата звернення: 17.11.2019).
14. Automated GUI Testing – Squish. URL: https://www.froglogic.com/squish/?utm_source=guru99&utm_content=testing-tools (дата звернення: 15.11.2019).
15. SeleniumHQ Browsers Automations. URL: http://www.seleniumhq.org/ (дата звернення: 15.11.2019).
16. Katalon | Simplify Web, API, mobile. URL: https://www.katalon.com/ (дата звернення: 19.11.2019).
17. Watir is... URL: http://watir.com/ (дата звернення: 19.11.2019).
18. LAMBDATEST – Cross Browser Testing Cloud. URL: https://www.lambdatest.com/?utm_source=Guru99-2&utm_medium=Listing&utm_campaign=Softwaretestinngtools&utm_term=listing (дата звернення: 20.11.2019).
19. Browsera. URL: http://www.browsera.com/web_sites (дата звернення: 23.11.2019).
20. BROWSERSHOTS. URL: http://browsershots.org/ (дата звернення: 23.11.2019).
21. WebLOAD – Website and applications. URL: https://www.radview.com/webload-download/ (дата звернення: 27.11.2019).
22. WAPT 10: Performance testing tool for web and mobile applications. URL: https://www.loadtestingtool.com/download.shtml (дата звернення: 27.11.2019).
23. LoadUI Pro. URL: https://www.loadui.org/downloads/download-loadui-pro.html (дата звернення: 27.11.2019).
24. Redmine. URL: http://www.redmine.org/projects/redmine (дата звернення: 29.11.2019).
25. The Bug Genie. URL: http://www.thebuggenie.com/#tab_active (дата звернення: 29.11.2019).
26. SoapUI. URL: https://www.soapui.org/downloads/download-soapui-pro-trial.html (дата звернення: 29.11.2019).
27. WebInject – Web/HTTP Test Tool. URL: http://www.webinject.org/ (дата звернення: 29.11.2019).

References:

1. Kravchuk, S.O. (2018). Problemy avtomatyzatsii testuvannia prohramnoho zabezpechennia [The problem of software testing automatyion]. Proceedings of the *Aktualni zadachi suchasnykh tekhnolohii: VII mizhnarodna naukovo-tekhnichna konferentsiia molodykh uchenykh ta studentiv* (Ukraine, Ternopil, November 28-29, 2019). Ternopil, p. 95.
2. Burov, Ye.V. (2011). Intelektualna systema avtomatyzovanoho testuvannia prohramnoho produktu z vykorystanniam alhorytmichnykh modelei [Intelligent system for automated testing of software using algorithmic models]. *Visnyk Natsionalnoho universytetu «Lvivska politekhnika»*, no. 699: Informatsiini systemy ta merezhi, pp. 21–30.
3. Hovorushchenko, T.O. (2008). Problemy realizatsii metodu identyfikatsii prykhovanykh pomylok prohramnoho zabezpechennia na osnovi neiromerezhnykh informatsiinykh tekhnolohii [Problems of implementation of the method of identification of hidden errors of software on the basis of neural network information technologies]. *Radioelektronni i komp'iuterni systemy*, no. 7, pp. 107–112.
4. Ali, K., & Xiaoling, X. (2019). A reliable and an efficient web testing system. *International Journal of Software Engineering & Applications*, vol. 10, no. 1, pp. 1–16. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3341095 (accessed 16.11.2019).
5. Mann, M., Sangwan, O.P., & Tomar, P. (2019). Automated software test optimization using test language processing. *The International Arab Journal of Information Technology*, vol. 16, no. 3, pp. 348–356.
6. Tamre, L. (2003). Vvedeniye v testirovaniye programmnogo obespecheniya [Introduction to Software Testing]. Moscow: Vil'yams. (in Russian)
7. Vasilenko, N.V. (2004). Modeli otsenki nadezhnosti programmnogo obespecheniya [Software reliability assessment models]. *Vestnik Novgorodskogo gosudarstvennogo universiteta im. Yaroslava Mudrogo*, no. 28, pp. 126–132.
8. Biryukov, S.V. (2008). Analiz strategiy testirovaniya programmnogo obespecheniya [Software Testing Strategies Analysis]. *Izvestiya YUFU. Tekhnicheskiye nauki*, vol. 78, no. 1, pp. 59–63.

9. Kaner, C., Folk, D., & Kek Nguyen, Ye. (2000). Testirovaniye programmnogo obespecheniya [Sofrware testing]. Kyiv: DiASoft. (in Russian)
10. Test Case Management Software – TestRail. Available at: https://www.gurock.com/testrail?utm_source=adwords&utm_medium=cpc&utm_campaign=europe_en_generic&utm_content=testpad&gclid=Cj0KCQiAw4jvBRCJARIsAHYewPPo21kSi2XJH3gafMJ1fmoK-HHv0HBUotizADsmbEKZP1Fw5k-nDH8aAr4QEALw_wcB (accessed 15.11.2019).
11. qTest – Modern Softw are Testing Tools Platform. Available at: https://www.qasymphony.com/software-testing-tools/ (accessed 15.11.2019).
12. PractiTEST QA Test Management Tool. Available at: https://www.practitest.com/?utm_medium=listings&utm_source=guru&utm_campaign=testing+tools (accessed 15.11.2019).
13. Automate Functional Testing QTP. Available at: https://www.microfocus.com/ru-ru/products/uft-one/overview (accessed 17.11.2019).
14. Automated GUI Testing – Squish. Available at: https://www.froglogic.com/squish/?utm_source=guru99&utm_content=testing-tools (accessed 15.11.2019).
15. SeleniumHQ Browsers Automations. Available at: http://www.seleniumhq.org/ (accessed 15.11.2019).
16. Katalon | Simplify Web, API, mobile. Available at: https://www.katalon.com/ (accessed 19.11.2019).
17. Watir is... Available at: http://watir.com/ (accessed 19.11.2019).
18. LAMBDATEST – Cross Browser Testing Cloud. Available at: https://www.lambdatest.com/?utm_source=Guru99-2&utm_medium=Listing&utm_ campaign=Softwaretestinngtools&utm_term=listing (accessed 20.11.2019).
19. Browsera. Available at: http://www.browsera.com/web_sites (accessed.11.2019).
20. BROWSERSHOTS. Available at: http://browsershots.org/ (accessed 23.11.2019).
21. WebLOAD – Website and applications. Available at: https://www.radview.com/webload-download/ (accessed 27.11.2019).
22. WAPT 10: Performance testing tool for web and mobile applications. Available at: https://www.loadtestingtool.com/download.shtml (accessed 27.11.2019).
23. LoadUI Pro. Available at: https://www.loadui.org/downloads/download-loadui-pro.html (accessed 27.11.2019).
24. Redmine. Available at: http://www.redmine.org/projects/redmine (accessed 29.11.2019).
25. The Bug Genie. Available at: http://www.thebuggenie.com/#tab_active (accessed 29.11.2019).
26. SoapUI. Available at: https://www.soapui.org/downloads/download-soapui-pro-trial.html (accessed 29.11.2019).
27. WebInject – Web/HTTP Test Tool. Available at: http://www.webinject.org/ (accessed 29.11.2019).