МЕТОД ПІДВИЩЕННЯ ЯКОСТІ ВЕБ-ДОДАТКІВ НА ОСНОВІ АВТОМАТНОГО ПІДХОДУ

Артюхов Валерій Євгенович,
студент групи 316-І, спеціальності 122 «Комп'ютерні науки», факультету Інформатики, математики та економіки
Постильна Олена Олексіївна,
кандидат педагогічних наук
Мелітопольський державний педагогічний університет імені Богдана Хмельницького

Анотація. Протягом останніх років суттєво зросла як складність веб-додатків, так і вимоги до їх надійності. Найчастіше через стислі терміни розробки й часто мінливих вимог замовника перевірка якості веб-додатків здійснюється вручну без формальних критеріїв. Відсутність моделей і формальної специфікації робить процес перевірки якості трудомістким та не дозволяє гарантувати відсутність помилок. У статті запропоновано метод автоматичного виділення явних станів і побудови автоматної моделі для веб-додатків за допомогою динамічного аналізу. Наявність автоматної моделі дозволить формалізувати вимоги специфікації та використовувати Model Checking для автоматизації процесу тестування. Також розроблений алгоритм пошуку схожих станів, завдяки якому навіть для складних веб-додатків, можливо автоматично будувати моделі з невеликою кількістю станів, які будуть зрозумілі розробникові.

Ключові слова. Model Checking, тестування, алгоритм, стан, автоматна модель, веб-додаток.

Складність веб-додатків стрімко зростає так само, як і вимоги до їх надійності та безпеки. Питання перевірки якості веб-додатків є актуальним, оскільки безліч програм реалізуються у вигляді веб-додатків і багато з них пов'язані з фінансовими трансакціями та конфіденційною інформацією: Інтернет –магазини (Amazon.com), Інтернет-банкінг, поштові клієнти (GMail), соціальні мережі (Facebook), системи документообігу (Alfresco) і багато інших.

Веб-додаток – клієнт-серверний додаток, в якому клієнтом виступає браузер, а сервером – веб-сервер. Логіка веб-додатка розподілена між сервером і клієнтом, обмін інформацією відбувається мережею [1]. Тестування веб-додатків так само, як і тестування користувальницьких інтерфейсів, важко автоматизувати, оскільки це вимагає емуляції дій користувача, таких як введення тексту, заповнення форм, переходи за посиланнями. Для складних веб-додатків різних послідовностей дій користувача може бути нескінченно багато, тому перевірити всі варіанти взаємодії користувача з системою неможливо. На практиці найчастіше перевіряється тільки невелика підмножина можливих варіантів дій користувача, що залишає можливість появи різноманітних критичних помилок.

У статті пропонується метод автоматизації тестування веб-додатків, заснований на автоматному підході. Кінцеві автомати дозволяють зручно описувати взаємодію системи з користувачем [2, 3]. Стан веб-додатка – веб-

сторінка, яку бачить користувач. Вхідні впливу, які можуть змінювати стан додатка – це дії користувача та відповіді сервера на AJAX-запити. Стан програми може змінюватися за рахунок переходу на нову веб-сторінку або динамічною зміною структури (DOM-дерева) поточної сторінки.

Вихідний код програми є найбільш точним описом логіки роботи, але це уявлення незручно для тестування, але він не дозволяє зрозуміти високорівневу логіку програми. З цією метою в статті запропоновано метод автоматичної побудови автоматних моделей існуючих веб-додатків.

У своїй роботі Alalfi M.H. автор досліджує 24 різні методи тестування та верифікації веб-сайтів [4, с. 267]. У більшості підходів пропонується починати розробку з побудови моделі або будувати модель вручну [5, 6, 7]. A Sylvain Halle наполягає на створенні моделі програми вручну та застосовуванні Model Checking для перевірки властивостей цієї моделі [8, с. 236]. Ці підходи не застосовні до існуючих складним веб-додатків, оскільки побудова моделі вручну занадто складна та не може гарантувати відповідність моделі коду програми. Більш того, при змінах коду програми доведеться вручну оновлювати саму модель.

У статті Haydar M. описаний метод побудови кінцевого автомата, який буде описувати задану властивість веб-додатка [9, с. 411]. Отримані моделі можуть складатися з тисяч станів і не можуть бути використані як зрозумілий опис веб-додатка. Цей підхід має на увазі будування моделі повністю автоматично наявної програми і потім, вручну проаналізувавши модель, описувати властивості веб-додатків, для перевірки яких можна буде використовувати Model Checking.

Спроба автоматизувати верифікацію та побудову моделей зроблена і колективом японських науковців [10, с. 355], але зі значними обмеженнями: підтримуються тільки додатки, розроблені з використанням фреймворку Struts і технології Java Server Page templates, і не підтримуються AJAX-додатки. Запропонований в даній роботі підхід застосовується до значно ширшого кола веб-додатків і не накладає обмеження на використовувані технології. У тому числі принципово важлива підтримка технології AJAX, яка повсюдно використовується для створення динамічних та інтерактивних програм.

Веб-додатком є вихідний код веб-сторінки, на якій знаходиться користувач. Події – дії користувача, що включають в себе елементи керування, введення тексту та заповнення форм.

Для автоматичної побудови моделі веб-додатку необхідно виявити максимальну кількість можливих станів. Дане завдання нерозв’язане в загальному випадку, так як різних комбінацій може бути багато. Ми пропонуємо розглянути алгоритм заснований на випадковій генерації дій користувача:

1. Вихідний код сторінки аналізується та складається перелік всіх можливих дій користувача actionlist, на які реагує дана сторінка додатку.

2. Вибирається випадкова дія зі списку можливих дій actionlist і виконується програмою, яка емулює дії користувача.

3. Всі дії запам’ятовуються у вигляді трійок <state1, action, state2>, де state1 – стан додатка до виконання дії, state2 – стан додатка після виконання дії, action – позначення виконаної дії.

4. Перейти до кроку 1, якщо за останні N ітерацій було виявлено хоча б одну нову подію, яка не зустрічалася раніше (N – підбирається під конкретну задачу). Дана умова зупинки буде спрацьовувати в тому випадку, якщо додаток було виявлено або якщо додаток потрапив в якийсь тупиковий стан, який не дозволяє перейти до нового, а тільки повертатися в попередній.

Вирішити проблему надмірно великої кількості станів можна об'єднанням схожих станів. Під станом ми маємо на увазі вихідний HTML–код сторінки. Робота з HTML-кодом як з DOM – деревом і введемо наступне рекурсивне визначення схожості: вершини DOM-дерева A і B схожі, тобто similar (A, B) == True якщо і тільки якщо вони одного типу, мають однаковий набір атрибутів і однакову кількість підверхівок, де кожна підверхівка A схожа з відповідною подверхівкою B.

При порівнянні верхівок текстові значення верхівок ігноруються, порівнюються тільки структури DOM-дерев. Наприклад, верхівки <p> text </ p> і <p> othertext </ p> будуть визнані схожими. Також при порівнянні двох DOM-дерев виконується два додаткових крока: фільтрація верхівок за їх типом і згортання дерев. Ті верхівки, які не впливають безпосередньо на поведінку веб-сторінки і обробку дій користувача, фільтруються: link, script, meta та інші. Згортання дерева дозволяє вважати схожими сторінки, які відрізняються тільки кількістю однотипних елементів, але не відрізняються своєю поведінкою. Прикладом таких веб-сторінок може бути список листів у поштовій скриньці або список результатів пошуку. Список DOM-верхівок x1, ..., xn буде згорнуто, якщо "i, j Î [1; n] similar (xi, xj) == True && xi.parent == xj.parent. У цьому випадку список верхівок буде замінений на список з однією верхівкою x1. Згортання виконується наступним алгоритмом:

1. Виконувати обхід DOM–дерева, починаючи з листя.

2. Для даної верхівки скласти список підверхівок listc.

3. Порівняти всі можливі пари xi, xj, listc і, якщо similar (xi, xj) == True, видалити підверхівку xj.

Два стани веб-додатка будуть схожими, якщо кореневі верхівки відповідних DOM-дерев, після кроків фільтрації та згортання будуть схожими.

Для апробації запропонованого підходу розроблено набір інструментів на мові Python 2.7. Для роботи з веб-сторінками та емуляції дій користувача використовується фреймворк Selenium [11, с. 272]. Для укладання графа та подання автоматної моделі у вигляді PNG-зображення використовується фреймворк GraphViz. Розроблений в рамках даного дослідження інструмент для заданого веб-додатка автоматично створює модель у вигляді кінцевого автомата, який описує виявлені стани програми та можливі переходи між ними. Мітки на станах відображають заголовки сторінок, а мітки на переходах дії, які призводять до зміни станів, записані в форматі: «натисніть на об'єкт L» або «введіть текст A в поле B». Вказівки на об'єкти всередині сторінки записані на мові XPath. На рис. 1 наведені приклади результату роботи інструменту для веб-додатків TadaList.com і m.Facebook.com.

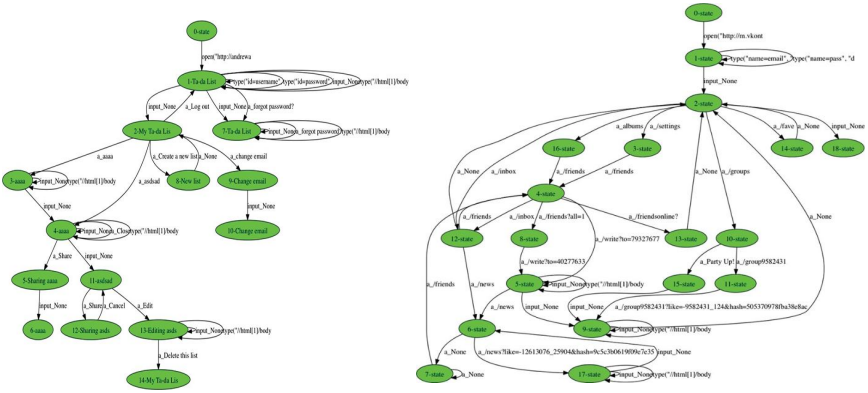

Рис 1. Автоматично побудовані моделі веб-додатків.

Для TadaList за 9 хвилин роботи інструментом було виконано 250 випадкових дій на сторінці додатків, виявлено 216 різних станів і переходів між ними. Після застосування алгоритму пошуку і злиття схожих станів була отримана модель, що складається з 15 станів. Для m.Facebook.com за 6 хвилин було виконано 200 випадкових дій, що дозволило виявити 170 різних станів і переходів між ними. Алгоритм пошуку та злиття схожих станів дозволив спростити модель до 19 станів.

У роботі зроблена спроба автоматизувати процес перевірки якості веб-додатків. Запропоновано метод автоматичної побудови автоматних моделей існуючих веб-додатків з урахуванням специфіки заданої області. Моделі можуть бути використані для формалізації вимог специфікації й автоматичної перевірки виконання вимог за допомогою верифікатора.

Також наявність моделі може істотно автоматизувати процес створення набору тестів, що дозволить перевіряти задані частини веб-додатка.

Література:

1. Wikipedia, Web-Application article. [Електронний ресурс] – Режим доступу: http://en.wikipedia.org/wiki/Web_application
2. Шалыто А. А., Туккель Н. И. Программирование с явным выделением состояний // Мир ПК. 2001. № 8. С. 116–121; № 9. С. 132–138.
3. Программирование мобильных устройств. – [Електронний ресурс] – Режим доступу: http://is.ifmo.ru/science/MD–Mobile.pdf.
4. Alalfi, M.H., Cordy, J. R.,Dean, T. R. Modelling methods for web application verifica- tion and testing: state of the art. Softw. Test., Verif. Reliab. (2009) 265–296.
5. Hassan A. E., Holt R. C. Architecture recovery of web applications. Proceedings of the 24th International Conference on Software Engineering ICSE, ACM Press: New York, NY, USA, 2002; 349–359.

6. Antoniol G., Di Penta M., Zazzara M. UnderstandingWeb Applications through Dy- namic Analysis. Proceedings of the 12th International Workshop on Program Compre- hension IWPC, 2004; 120–131.
7. Di Lucca G. A., Di Penta M. Integrating Static and Dynamic Analysis to improve the Comprehension of Existing Web Applications. Proceedings of the Seventh IEEE International Symposium on Web Site Evolution WSE, IEEE Computer Society: Wash- ington, DC, USA, 2005; 87–94.
8. Sylvain Halle, Taylor Ettema, Chris Bunch, Tevfik Bultan. Eliminating navigation errors in web applications via Model Checking and runtime enforcement of navigation state machines. ASE 2010: 235–244.
9. Haydar M. Formal Framework for Automated Analysis and Verification of Web-Based Applications. In ASE(2004) 410–413.
10. Atsuto Kubo, Hironori Washizaki, Yoshiaki Fukazawa. Automatic Extraction and Veri- fication of Page Transitions in a Web Application // Apsec. 14th Asia-Pacific Software Engineering Conference. 2007. Pp. 350–357.
11. Antawan Holmes, Marc Kellogg. Automating Functional Tests Using Selenium // Pro- ceedings of the conference on AGILE 2006. July 23–28, 2006. Pp. 270–275.
12. Шаров С.В., Постильна О.О. Інформатизація освіти та виховання як вектор розвитку сучасного суспільства. Науковий вісник мелітопольського державного педагогічного університету. Серія: Педагогіка. – Вип. 1(17) – Мелітополь: ФО-П Однорог Т.В. – 2017. – С. 199-204.

ВИКОРИСТАННЯ МЕСЕНДЖЕРІВ В ОСВІТІ

Бабаєв Ігор Віталійович,
Студент 4 курсу «Документознавство та інформаційна діяльність»
Донецького національного університету імені Василя Стуса, м. Вінниця
Січко Тетяна Василівн,а
к.т.н., доцент кафедри прикладної математики
та теорії систем управління
Донецького національного університету імені Василя Стуса, м. Вінниця

Анотація. У тезах доповіді викладено основні аспекти використання месенджерів та ботів. Розкрито перспективи їх використання на всіх рівнях освіти. Наведено порівняння з традиційним програмним забезпеченням.

Ключові слова: освіта, месенджери, чат-боти, комунікація, програмне забезпечення.

Миттєві повідомлення або, повніше, система обміну миттєвими повідомленнями (англ. Instant messaging, скорочено IM) — телекомунікаційна служба для обміну текстовими повідомленнями між комп'ютерами або іншими пристроями користувачів через комп'ютерні мережі (як правило через Інтернет). Зазвичай і від початку, це були невеликі текстові повідомлення. Але з розвитком у систему були додані й інші функції, такі як передавання файлів, зображень, звукових сигналів та повідомлень, відео, а також здійснення спільних дій, таких як малювання або ігри.