JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 28-31
Wysłane: 2018-11-26
Przyjęte: 2018-12-02

Analiza możliwości testowania aplikacji typu SPA na przykładzie narzędzi Selenium i Protractor

Mateusz Szpinda*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Testy automatyczne mają na celu weryfikowanie funkcjonalności systemu na poziomie interfejsu użytkownika końcowego. W artykule porównane zostały narzędzia Selenium WebDriver oraz Protractor. Do przeprowadzenia analizy porównawczej narzędzi ustalono pięć kryteriów porównawczych na podstawie, których przeprowadzony został proces analitycznej hierarchizacji. Na podstawie otrzymanych obliczeń zostały wyciągnięte wnioski dotyczące tego, która platforma jest lepszym wyborem pod względem testowania aplikacji Single Page Application.

Słowa kluczowe: testy automatyczne; Selenium; Protractor; Single Page Application; Angular

*Autor do korespondencji.

Adres e-mail: mateusz.szpinda94@gmail.com

Analysis of the possibilities of testing SPA applications on the example of Selenium and Protractor tools

Mateusz Szpinda*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Automated tests are designed to verify the functionality of the system at the level of the end-user interface. The article compared popular tools Selenium WebDriver and Protractor. To conduct a comparative analysis tools have been established five benchmarks under which it was carried out the analytical hierarchy proces. Based on the obtained calculations, conclusions have been drawn regarding which platform is a better choice in terms of testing the application of the Single Page Application.

Keywords: automated tests; Selenium; Protractor; Single Page Application; Angular

*Corresponding author.

E-mail address: mateusz.szpinda94@gmail.com

1. Wstęp

Testowanie oprogramowania jest nieodzownym elementem podczas procesu wytwarzania jak najwyższej, jakości oprogramowania. Testy umożliwiają wykrycie błędów we wczesnej fazie rozwoju oprogramowania [1]. Próby zminimalizowania ilości błędów występujących w wersjach produkcyjnych systemów informatycznych od lat skutkowały ewolucją procesu testowania. Szczególnym rodzajem testów, na które w coraz większym stopniu kładzie się nacisk są testy automatyczne, które znacząco przyspieszają proces testowania. Głównym celem stosowania testów automatycznych jest zmniejszenie nakładu pracy, jaką trzeba wykonać by przetestować tworzoną aplikacje [2].

2. Przegląd literatury

Porównanie Selenium i Protractor nie było jak dotąd poruszane w literaturze naukowej. Główną tego przyczyną jest fakt, że Protractor jest nowym narzędziem na rynku służącym do testowania aplikacji internetowych. Można natomiast znaleźć pozycje przedstawiające tematykę testowania aplikacji opartych na frameworku Angular oraz porównania między innymi Selenium WebDriver z innymi technologiami wspierającymi testy automatyczne.

Tobias Palmer oraz Markus Waltre przedstawiają architekturę aplikacji Single Page Application oraz opisują testowanie SPA na przykładzie narzędzia Protractor [3].

Harsh Bajaj w swoim artykule pisał na temat wyboru technologii służących do wykonywania testów automatycznych [4]. Szukając najlepszego rozwiązania brał pod uwagę takie kryteria jak wspierane systemy i przeglądarki, łatwość wdrożenia w technologie oraz łatwość tworzenia skryptów testowych.

Badanie zostanie przeprowadzone przy pomocy jednej z metod analizy wielokryterialnej zwanej procesem analitycznej hierarchizacji. Paweł Cabała w swoim artykule przedstawia założenia teoretyczne tej metody [5]. Dodatkowo został przedstawiony praktyczny przykład zastosowania procesu.

3. Procedura badawcza

3.1. Cel

Celem pracy jest analiza możliwości testowania aplikacji internetowych typu Single Page Application (SPA) na przykładzie narzędzi Selenium WebDriver oraz Protractor. Oba narzędzia służą do przeprowadzania testów automatycznych aplikacji internetowych. Analiza zostanie

przeprowadzona przy pomocy procesu analitycznej hierarchizacji [5]. W tym celu zostały określone kryteria porównawcze takie jak dokumentacja techniczna, wspierane platformy, możliwości testowania SPA, wyszukiwanie elementów oraz szybkość wykonywania testów.

3.2. Technologie

3.2.1. Selenium

Selenium jest jedną z najpopularniejszych platform służących do wykonywania testów automatycznych stron internetowych. Framework powstał w 2004 roku, którego twórcą był Jason Huggins. Narzędzie umożliwia pracę w wielu popularnych językach programowania takich jak Java, C#, JavaScript, Python itp. W skład Selenium wchodzą następujące narzędzia: Selenium IDE, Selenium RC, Selenium WebDriver oraz Selenium Grid [6].

3.2.2. Protractor

Protractor jest darmowym narzędziem służącym do tworzenia testów automatycznych. Został opracowany przez zespół programistów firmy Google Inc. Protractor jest napisany w języku JavaScript oraz jest oparty o Selenium WebDriver. Zaletą narzędzia jest fakt, że oferuje dedykowane funkcje służące do testowania aplikacji Angular.js oraz Angular [3]. Protractor współpracuje z takimi frameworkami jak Jasmine, Mocha itp.

3.3. Aplikacja testowa

W celu przeprowadzenia procedury porównawczej powstała aplikacja internetowa symulująca działanie sklepu internetowego. Część serwerowa została napisana w środowisku uruchomieniowym Node.js z wykorzystaniem frameworka Express oraz nierelacyjnej bazy danych MongoDB zgodnie z wzorcem REST (ang. Reprezentational State Transfer). Za warstwę klienta odpowiada framework Angular 5, Angular Material oraz bootstrap 4. System można podzielić na dwa moduły: część przeznaczoną dla użytkownika oraz dla administratora systemu. Część użytkownika umożliwia korzystanie z następujących funkcjonalności: rejestracja, logowanie, edycja danych osobowych, dodawanie produktów do koszyka, dodawanie produktów do listy życzeń. Część administracyjna udostępnia następujące funkcjonalności: zmiana statusów zamówień, funkcje CRUD (ang. create, read, update and delete) na katalogach, produktach oraz użytkownikach.

3.4. Kryteria

3.4.1. Dokumentacja

Istotnym punktem podczas wyboru nowej technologii, z której programista ma korzystać jest dokumentacja techniczna. Dobrze sporządzona dokumentacja cechuje się tym, że zawiera prawidłowo opisane funkcjonalności oraz praktyczne przykłady ich użycia. Dokumentacja powinna być przygotowana w sposób uporządkowany, intuicyjny, aby programista korzystający z niej mógł znaleźć poszukiwane informacje przy jak mniejszym nakładzie czasu.

3.4.2. Wspierane platformy i języki

Niewątpliwie wsparcie wielu platform bądź języków programowania jest istotnym elementem podczas wyboru odpowiedniego narzędzia [7]. Istnieją narzędzia, które wymuszają na programiście korzystanie z danego języka z powodu braku wsparcia innych platform. W przypadku projektów o dużej skali, w której wykorzystywane są różne języki programowania istotne jest, aby narzędzia testowe były dokładnie w tym języku, co dana część projektu.

3.4.3. Możliwości testowania SPA

W aplikacjach webowych typu Single Page Application przepływ danych miedzy klientem a serwerem funkcjonuje w sposób asynchroniczny za pomocą protokołu http [8]. Narzędzia służące do testowania tego typu aplikacji mają za zadanie obsłużyć asynchroniczne funkcjonowanie aplikacji. Głównym problemem obsługi tego typu zdarzeń jest nieprzewidywalność komponentów aplikacji. Czas od momentu wysłania żądania do serwera, w celu pobrania bądź wysłania danych, do momentu otrzymania odpowiedzi jest nieznany i zależny między innymi od takich czynników jak odległość serwera, ilość danych oraz poziom zaawansowania ich przetworzenia.

3.4.4. Wyszukiwanie elementów

Wyszukiwanie elementów na stronie jest istotnym kryterium podczas badania oraz korzystania z narzędzi służących do tworzenia testów automatycznych. Są one podstawowymi funkcjonalnościami, jakie oferują narzędzia, które są wykorzystywane podczas każdego scenariusza testowego. Za ich pomocą symulowane są interakcje ze stroną naśladujące zachowanie użytkownika. Ponadto znajdowanie elementów na stronie może służyć kontrolowaniu poprawności wyświetlanych danych, sprawdzaniu czy dane się pojawiły bądź oczekiwaniu na element.

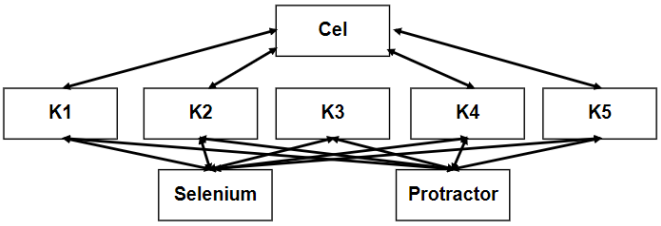
3.4.5. Szybkość

Czas, w jakim testy zostaną wykonane jest jednym z kluczowych kryteriów porównawczych narzędzi służących do testowania aplikacji SPA [9]. Testy automatyczne mają za zadanie symulować zachowanie użytkownika aplikacji, a co za tym idzie, wykonanie ich może być bardzo czasochłonne, szczególnie w przypadku systemów o dużej skali.

3.5. Proces hierarchizacji

Pierwszy etap polega na utworzeniu hierarchicznej struktury procesu decyzyjnego. Na najwyższym poziomie struktury hierarchicznej powinien znajdować się główny cel, który zostaje rozkładany na niezależne od siebie kryteria oceny, wyznaczone przez decydenta. Kryteria te znajdują się na następnym poziomie hierarchii. Na najniższym poziomie struktury hierarchicznej znajdują się rozpatrywane warianty

decyzyjne. Schemat struktury przedstawiony został na rysunku 1.



Rys. 1. Struktura hierarchiczna problemu

Kryteriami wyboru odpowiedniego narzędzia do testowania są następujące pozycje:

- dokumentacja techniczna (K1);
- wspierane platformy programistyczne (K2);
- praca z asynchronicznym działaniem aplikacji (K3);
- szybkość wykonywanych testów (K4);
- łatwość wyszukiwania elementów na stronie (K5).

Tabela 1. Skala Saaty`ego 0

Wartość	**Stopień ważności (ocena obiektu A względem B)**
1	Jednakowa ważność (A jest jednakowo preferowane z B)
3	Nieznaczna ważność (A jest bardziej preferowane niż B)
5	Wyraźna ważność (A jest dużo bardziej preferowane niż B)
7	Bardzo wyraźna ważność (A jest bardzo silnie preferowane w porównaniu z B)
9	Absolutna ważność (A jest ekstremalnie preferowane w porównaniu z B)
(2,4,6,8)	Wartości pośrednie

Kolejnym etapem procesu jest porównywanie parami kryteriów oceny. W tabeli 1 przedstawiono dziewięciostopniową skalę preferencji na podstawie, której dokonano porównań ważności ustalonych kryteriów. Ustalono, że K1 jest najmniej istotne spośród wszystkich kryteriów. Określono, że K2 ma większą wagę od K1, a zdecydowanie mniejsze od pozostałych. Uznano, że K3 ma większą wagę od pozostałych opcji. W dalszej kolejności porównywano K4 z K1 (ocena 6), K2 (ocena 3), K3 (ocena 1/4), oraz K5 (ocena 1/3). Kryterium K5 okazało się mniej ważne tylko od K3 (ocena 1/4). Powyższe oceny pozwoliły na budowę macierzy porównań parami (A), która została przedstawiona w tabeli 2.

W następnym kroku macierz A jest normalizowana w macierz B. Do wykonania tego kroku skorzystano z poniższego wzoru:

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \quad (1)$$

gdzie: n- liczba porównywanych elementów.

Tabela 2. Wyznaczenie wag i wskaźników spójności dla kryteriów

Macierz **A**

	K1	K2	K3	K4	K5
K1	1	1/3	1/7	1/6	1/5
K2	3	1	1/7	1/5	1/3
K3	7	6	1,	4	4
K4	6	5	1/4	1	1/3
K5	5	3	1/4	3	1

Macierz **B**

	K1	K2	K3	K4	K5
K1	0,0455	0,0217	0,0800	0,0199	0,0216
K2	0,1364	0,0652	0,0800	0,0239	0,0576
K3	0,3182	0,3913	0,5600	0,4781	0,6906
K4	0,2727	0,3261	0,1400	0,1195	0,0576
K5	0,2273	0,1957	0,1400	0,3586	0,1727

	w	**Aw**	**Aw/w**		
K1	**0,0377**	0,1895	5,0210		
K2	**0,0726**	0,3651	1,1232	L	7,3737
K3	**0,4876**	2,7955	16,5540	CI	0,5934
K4	**0,1832**	0,9675	4,1567	CR	8,6%
K5	**0,2188**	1,2968	10,0137		
	1,0000				

Macierz B umożliwia wyznaczenie wektorów preferencji badanych elementów. W tym celu skorzystano z poniższego wzoru:

$$w_i = \frac{1}{n}\sum_{j=1}^{n} b_{ij} \quad (2)$$

gdzie : n- liczba porównywanych elementów.

Z powyższych rezultatów widać, że największą wagę otrzymało kryterium K3 (0.4876). Następnie znalazły się K5 (0,2188), K4 (0,1832), K2 (0,0726). Najniższą wagę posiada kryterium K1 (0,0377).

Metoda AHP (ang. Analytic Hierarchy Process) umożliwia weryfikację spójności porównań parami. W tym celu należy obliczyć maksymalną wartość własnej macierzy, która została obliczona za pomocą poniższego wzoru:

$$\lambda_{max} = \frac{1}{n}\sum_{i=1}^{n} \frac{(Aw)_i}{w_i} \quad (3)$$

gdzie: λ_{max}- maksymalna własność macierzy, n- liczba porównywanych elementów, w_i- wektor preferencji, Aw- wektor preferencji macierzy A

Wartość wskaźnika konsekwentności (CR, ang. consistency ratio) nie przekracza 10 %, co oznacza, że przy porównywaniu ze sobą elementów zachowano logikę ocen.

Kolejnym etapem procesu analitycznej hierarchizacji jest porównanie parami narzędzi Selenium oraz Protractor biorąc pod uwagę każde z kryteriów osobno. Końcowe wyniki oraz obliczone wektory preferencji oznaczone literą v zostały zaprezentowane w tabeli 3. Względem K1 lepsze okazało się narzędzie Protractor (0,67). Biorąc pod uwagę K2 wyższą wagę otrzymało Selenium WebDriver (0,86). W następnych kryteriach narzędziem o wyższej wadze okazał się Protractor, dla K3 (0,8), dla K4 (0,67) oraz K5 (0,75).

Tabela 3. Ocena narzędzi ze względu na poszczególne kryteria

K1-dokumentacja					
	I	II	macierz B		v
I	1,00	1/2	0,33	0,33	**0,33**
II	2,00	1,00	0,67	0,67	**0,67**

K2-wspierane platformy					
	I	II	macierz B		v
I	1	6	0,86	0,86	**0,86**
II	1/6	1	0,14	0,14	**0,14**

K3-asynchroniczność					
	I	II	macierz B		v
I	1	1/4	0,20	0,20	**0,20**
II	4	1	0,80	0,80	**0,80**

K4-szybkość					
	I	II	macierz B		v
I	1	1/2	0,33	0,33	**0,33**
II	2	1	0,67	0,67	**0,67**

K5-wyszukiwanie elementów					
	I	II	macierz B		v
I	1	1/3	0,25	0,25	**0,25**
II	3	1	0,75	0,75	**0,75**

4. Wyniki

W tabeli 4 przedstawiano dane związane z wyznaczeniem ostatecznych wag oraz wariantów. Końcowo wyższą ocenę otrzymał Protractor, którego waga wyniosła 0,61. Ponadto suma wag poszczególnych kryteriów oraz suma ostatecznych wag wyniosła 1, co potwierdza poprawność obliczeń.

Tabela 4. Wyznaczenie ostatecznych wag i wariantów

Kryteria	K1	K2	K3	K4	K5	Suma
Wagi	0,0377	0,0726	0,4876	0,1832	0,2188	1,0000
Warianty	Wagi wariantów względem kryteriów					Ostateczne wagi
	K1	K2	K3	K4	K5	
Selenium	0,33	0,86	0,20	0,33	0,25	**0,39**
Protractor	0,67	0,14	0,80	0,67	0,75	**0,61**
Suma	1,00	1,00	1,00	1,00	1,00	1,00

5. Podsumowanie

Biorąc pod uwagę wyniki przeprowadzonego badania oraz wybrane kryteria porównawcze uznano, że lepszym wyborem pod względem testowania aplikacji napisanej przy użyciu frameworka Angular będzie Protractor. Ponadto biorąc pod uwagę każde z kryteriów osobno, wykazano, że we wszystkich zestawieniach nieznacznie bądź wyraźnie lepszym wyborem jest Protractor. Głównym powodem tego jest fakt, że Protractor jest dedykowanym narzędziem do aplikacji Single Page Application dającym większe możliwości oraz posiadającym wbudowane mechanizmy ułatwiające testowanie aplikacji.

Literatura

[1] Jureczko M.: Testowanie oprogramowania, Politechnika Wrocławska, 2011

[2] Test Automation Tool Comparison – HP UFT/QTP vs. Selenium, https://www.aspiresys.com/WhitePapers/QTPvsSelenium.pdf [20.11.2018]

[3] Palmér T., Walter M.: Automated end-to-end user testing on single page web applications

[4] Bajaj H.: Choosing the right automation tool and framework is critical to project success

[5] Cabała P.: Proces analitycznej hierarchizacji w ocenie wariantów rozwiązań projektowych, Quarterly Journal–No, 2018, 24 [20.11.2018]

[6] Umesh N., Saraswat A: Automation Testing: An Introduction to Selenium. International Journal of Computer Applications, 119(3), 2015

[7] Smilgin R.: Zawód tester, Wydawnictwo Naukowe PWN, 2016

[8] Scott E.: SPA design and architecture: understanding single page web applications. Manning Publications Co., 2015

[9] Drążek K., Skublewska-Paszkowska M.: Porównanie wydajności testów automatycznych napisanych w technologii SeleniumWebDriver i HP UFT

Saaty T. L.: How to make a decision: The Analytic Hierarchy Process, European journal of operational research, 1990, 48.1: 9-26