JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 15 (2020) 156-163
Received: 26 November 2019
Accepted: 12 December 2019

Comparative analysis of solutions used in automated testing

Analiza porównawcza rozwiązań wykorzystywanych w testowaniu automatycznym

Agnieszka Dorota Wac*, Tomasz Kamil Watras, Grzegorz Kozieł

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The article presents a comparative analysis of tools supporting the production of automated tests for specific test scenarios. Within the framework of the research, the described tools were installed and then their functionalities were described. Using each of the tools, author's automatic tests were written and performed. The tool discussion was examined both in terms of the process of creating the tests and the speed of their execution. In addition, the selected tools were evaluated and subjective evaluations of the comfort of using the tool were taken into account, taking into account aspects such as the ease of installation and configuration of the tool, the need for programming skills, the simplicity of creating tests, additional functionalities and access to documentation. The summary includes the evaluation of the compared applications, and also discusses specific cases for which the tool will work better.

Keywords: Selenium; Katalon Studio; SahiPro; TestComplete

Streszczenie
W artykule przedstawiona została analiza porównawcza narzędzi wspomagających wytwarzanie zautomatyzowanych testów dla sprecyzowanych scenariuszy testowych. W ramach badań oprogramowanie zostało zainstalowane, a następnie opisano jego funkcjonalności. Przy pomocy każdego z narzędzi zostały napisane i wykonane autorskie testy automatyczne. Omawianie programy zbadano zarówno pod kątem procesu tworzenia testów, jak i prędkości ich wykonywania. Dodatkowo zostały ocenione pod względem subiektywnej oceny komfortu używania danego narzędzia, biorąc pod uwagę aspekty takie, jak łatwość instalacji i konfiguracji, potrzeba znajomości programowania, prostota tworzenia testów, dodatkowe funkcjonalności oraz dostęp do dokumentacji. W podsumowaniu zawarta została ocena porównywanych aplikacji, a także omówione zostały konkretne przypadki, dla których dane oprogramowanie sprawdzi się lepiej.

Słowa kluczowe: Selenium; Katalon Studio; SahiPro; TestComplete

*Corresponding author
Email address: agnieszka.wac@pollub.edu.pl (A. D. Wac)

1. Wstęp

Szybki rozwój technologii informatycznych, duża konkurencja na rynku oraz presja ze strony klientów firm informatycznych wymuszają zwiększanie tempa wytwarzania i dostarczania oprogramowania. Zmiany standardów oraz wymagań wpływają na stosowanie coraz to innych, nowszych technologii. Zmiana metodyk zarządzania projektami i przejście w kierunku metodyk zwinnych zmusza twórców do większej elastyczności, ale też wprowadza więcej zmian w tworzonym rozwiązaniu.

Wszystkie wymienione zjawiska skutkują zwiększeniem ryzyka popełnienia błędu podczas procesu wytwarzania oprogramowania. Skutecznym sposobem redukowania liczby defektów oraz zwiększania niezawodności dostarczanych rozwiązań informatycznych jest testowanie. Stosowane jest ono w większości dużych projektów. Naturalnym wydaje się być wykonywanie testów manualnych. Jednak wysokie koszty pracy ludzkiej oraz brak odpowiedniej liczby wykwalifikowanych testerów skutkują wprowadzaniem testów automatycznych. Testy te są szczególnie opłacalne w przypadku wielokrotnego powtarzania tych samych przypadków testowych. Wówczas raz opracowany test automatyczny jest uruchamiany wielokrotnie bez konieczności angażowania zasobów ludzkich.

W niniejszym artykule przedstawiono narzędzia pozwalające na automatyzację testów funkcjonalnych aplikacji internetowych oraz takie, które wspomagają testowanie interfejsu programistycznego aplikacji opartych o architekturę REST (ang. representational state transfer).

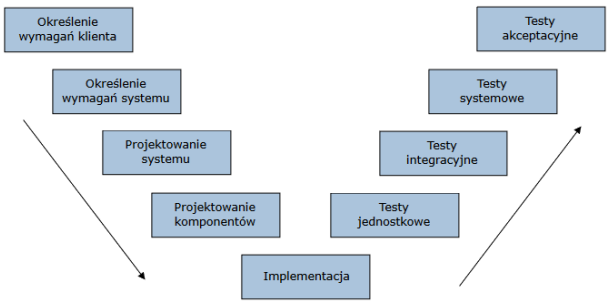
2. Modele wytwarzania oprogramowania

Zgodnie z podejściem ISTQB (International Software Testing Qualification Board - Stowarzyszenie Jakości Systemów Informatycznych) w różnych modelach wytwarzania oprogramowania stosowane są odmienne techniki planowania i realizacji testów [1].

Najbardziej rozpowszechnionym rozwiązaniem jest model wodospadu. Charakteryzuje się on tym, że każdy kolejny etap rozpoczyna się bezpośrednio po akceptacji ukończenia poprzedzającego [2]. Proces wytwórczy dzieli się na określone fazy: specyfikacja wymagań, projektowanie, implementacja, integracja, testy i weryfikacja, wdrożenie, utrzymanie. Podejście to sprawdza się dobrze w małych projektach, o ściśle okre-

ślonych wymaganiach, ponieważ pozwala na precyzyjne rozmieszczenie zasobów i weryfikację każdego etapu wytwarzania oprogramowania. Nie jest jednak polecane do dynamicznie zmieniających się oraz rozbudowywanych projektów. Dostarczenie gotowego produktu odbywa się dopiero w końcowych etapach.

Inne podejście prezentowane jest w modelu V. Jego nazwa pochodzi od specyficznego, równoległego sposobu wykonywania poszczególnych etapów, co prezentuje rysunek 1. W pierwszym etapie (koncepcyjnym) określone zostają wymagania i wybierane są idee możliwe do zrealizowania przy określonych zasobach i możliwościach technicznych. Wymaga to zaangażowania zarówno zespołów architektów, jak i programistów oraz testerów. W kolejnych fazach - implementacyjnej oraz weryfikacyjnej, tworzony i sprawdzany jest gotowy produkt. Dzięki ścisłej współpracy wewnątrz projektu możliwe jest wypracowanie odpowiednich rozwiązań już na etapie powstawania idei, a w efekcie zapobieganie błędom zanim zostaną one zaimplementowane.



Rysunek 1: Rozkład etapów modelu V

Modele sekwencyjne stosuje się w przypadku, gdy założenia projektu są z góry znane, proces budowania aplikacji dokładnie opisany, a klient oczekuje gotowego produktu. Coraz częściej jednak tworzone są systemy o dużym stopniu skomplikowania, rozwijane przez długi okres czasu. W tej sytuacji popularność zyskują techniki zwinne [3]. Opierają się one na iteracyjnym modelu wytwarzania oprogramowania. Techniki te kładą nacisk na współpracę zespołu z klientem, wytwarzanie funkcjonujących części aplikacji oraz szybkiego reagowania na konieczność wprowadzania zmian. Klient nie otrzymuje gotowego produktu, ale uczestniczy w procesie jego tworzenia, otrzymując kolejne działające elementy systemu. Konieczność ciągłej weryfikacji poprawności działania i integralności systemów sprzyja automatyzacji testów.

Do wyboru odpowiedniego modelu wytwarzania oprogramowania konieczna jest dokładna analiza specyfiki danego projektu. Należy odpowiednio wcześnie określić sposób dostarczania działających aplikacji, możliwości współpracy z klientem, a także rozważyć korzyści płynące z rzetelnego opracowania i utrzymywania dokumentacji oraz postepowania ściśle według wytycznych.

3. Testy

Bez względu na wybór modelu, system należy sprawdzić pod wieloma względami. Testowana powinna być poprawność działania poszczególnych elementów, ich zgodność z wymaganiami, ale również aspekty niefunkcjonalne, takie jak stabilność, niezawodność, wydajność, użyteczność, bezpieczeństwo, kompatybilność, utrzymywalność, przenaszalność [4].

3.1. Rodzaje testów

Wczesne wykrycie możliwych błędów i podatności pozwala na sprawniejsze ich wyeliminowanie. Wprowadzanie zmian na wysokim stopniu zaawansowania projektu zawsze wiąże się ze zwiększeniem kosztów i wydłużeniem czasu wykonania produktu. Dlatego też często łączone jest wykonywanie testów biało- i czarnoskrzynkowych. Pierwszy rodzaj skupia się wykryciu błędów metod poprzez wprowadzanie gotowych zestawów danych i wymuszanie przejścia przez wszystkie ścieżki systemu, co wymaga znajomości programowania. Testy czarnoskrzynkowe pozwalają na weryfikację czy oprogramowanie spełnia swoje założenia. Zadaniem testera jest dobranie wartości i wykonanie działań sprawdzających czy uzyskany wynik działania danej funkcjonalności jest zgodny z przewidywanym w dokumentacji.

Aby mieć pewność, że każdy wykryty błąd został naprawiony konieczne jest wykonanie testów potwierdzających – retestów. Pozwalają one na weryfikację poprawności usunięcia defektu. Jednocześnie dodanie nowej funkcjonalności lub rozbudowa już istniejącej, powinna wiązać się z przeprowadzeniem testów regresyjnych, których celem jest wykrycie potencjalnych błędów spowodowanych wprowadzeniem zmian w innych częściach funkcjonalności.

W sylabusie ISTQB wyróżnia się testy [1] modułowe, integracyjne, systemowe oraz akceptacyjne. Każdy z nich wymaga skonfigurowania odpowiedniego środowiska testowego. Odnoszą się one również do innych aspektów i funkcjonalności oprogramowania.

Testy modułowe (jednostkowe) mają na celu sprawdzenie działania odseparowanych modułów aplikacji. Umożliwiają wykrycie błędów na poziomie funkcji. Symulowanie ich działania w środowisku docelowym jest uzyskiwane przy pomocy zaślepek, sterowników, atrap systemu lub wirtualizacji usług [1].

W kolejnym etapie sprawdzana jest integracja modułów, czyli poprawność ich komunikacji. Wykonywane testy angażują nie tylko metody danego oprogramowania, ale również bazy danych, czy zewnętrzne serwisy. Sprawdzana jest prawidłowość w przesyłaniu komunikatów, ich kompletność, zabezpieczenia oraz synchronizacja.

Testy systemowe wymagają poprawnej i kompletnej konfiguracji całego systemu, odwzorującego środowisko produkcyjne lub docelowe. Pozwalają na sprawdzenie, czy stworzony system działa zgodnie z założeniami. Umożliwiają również wykrycie potencjalnych problemów współpracy ze środowiskiem docelowym, np.

w przypadku wykorzystywania innego systemu operacyjnego lub braku kompletnej konfiguracji.

Ostatnim etapem jest wykonanie testów akceptacyjnych. Ich celem nie musi być wyszukiwanie kolejnych defektów oprogramowania, ale sprawdzenie gotowości systemu do wdrożenia w docelowym środowisku. Wykonanie testów akceptacyjnych wiąże się z budowaniem zaufania do tworzonego oprogramowania. Umożliwia zapewnienie odbiorców końcowych, że wytworzony system działa w sposób przewidziany w specyfikacji i realizuje określone procesy biznesowe.

3.2. Testy manualne a automatyczne

Celem przeprowadzenia testów jest zapewnienie określonego poziomu jakości oprogramowania. Wiele czynności weryfikujących poprawność działania aplikacji może zostać zautomatyzowana, nadal jednak są aspekty, które wymagają ręcznej weryfikacji.

Testy manualne polegają na ręcznym uruchamianiu poszczególnych funkcji systemu przez testera. Pozwalają na wykrycie błędów działania aplikacji i interfejsów, a także ocenę takich aspektów systemu, jak łatwość obsługi, ergonomiczność i przydatność dla użytkownika. Zaletą testów manualnych jest to, że wykonywanie ich nie wymaga znajomości języków programowania, umożliwiają szybkie przetestowanie nowych lub zmieniających się funkcjonalności, a także wykonanie bardzo złożonych scenariuszy, nieopłacalnych do automatyzacji. Ich wadą jest konieczność bazowania na zasobach ludzkich, co powoduje wzrost kosztów i czasu potrzebnego do realizacji projektu.

Testy automatyczne opierają się na wykorzystaniu narzędzi i bibliotek do uruchamiania zaprogramowanych przypadków testowych. W początkowej fazie powstawania projektu wymagają większych kosztów niż testy manualne, jednak pozwalają na zapobieganie wielu błędom na późniejszych etapach, a także skrócenie czasu potrzebnego na przetestowanie niektórych aspektów oprogramowania dzięki możliwości ponownego użycia napisanych testów. Wymagają jednak ciągłej aktualizacji oraz weryfikacji działania, a także kontrolowania, czy ich wykonanie jest nadal zasadne i użyteczne. Automatyzacja jest najbardziej korzystna w przypadku konieczności wykonywania testów [5]:

- powtarzalnych,
- dla wielu różnych zestawów danych,
- na różnych platformach sprzętowych oraz konfiguracjach,
- czasochłonnych lub niemożliwych do ręcznej weryfikacji,
- wymagających wprowadzenia dużej ilości danych wejściowych,
- dla krytycznych funkcjonalności,
- wydajnościowych.

3.3. Narzędzia do testów funkcjonalnych aplikacji internetowych

SeleniumIDE jest to darmowy dodatek stworzony dla dwóch najpopularniejszych przeglądarek internetowych, czyli Chrome oraz Firefox. Jest on dostępny do pobrania z oficjalnych sklepów dodatków. Tester nie musi się przejmować konfiguracją środowiska testowego ani instalowaniem skomplikowanych narzędzi. Po zainstalowaniu dodatku, narzędzie jest gotowe do pracy. Dzięki prostemu interfejsowi, użytkownik jest w stanie stworzyć zautomatyzowane testy bez korzystania z dokumentacji. W celu ułatwienia zarządzania tworzonymi testami, ma możliwość tworzenia projektów, a także definiowania przypadków użycia. Program oferuje szerokie możliwości nagrywania i edycji testów oraz definiowania ich przy pomocy dostępnych akcji.

Selenium WebDriver jest to udostępniony dla użytkownika zestaw funkcji dostępnych w WebDriver API, dołączonego do silnika Selenium, które pozwalają między innymi na wyszukiwanie elementów strony za pomocą selektorów CSS, xpath oraz atrybutów dostępnych dla znaczników HTML. Narzędzie jest przeznaczone dla bardziej zaawansowanych użytkowników posiadających znacznie większą wiedzę na temat wytwarzania oprogramowania w wysokopoziomowych językach, wspieranych przez Selenium WebDriver, takich jak C#, Java, Ruby, Python oraz JavaScript [6]. Użytkownik samodzielnie definiuje wykonywane kroki testowe, sposób obsługi błędów oraz generowanie informacji na temat przeprowadzonych testów.

TestComplete jest to komercyjne rozwiązanie do wytwarzania testów automatycznych, przeznaczone dla bardziej wymagających użytkowników. Narzędzie oferuje automatyzację testów dla aplikacji internetowych, desktopowych oraz mobilnych wykorzystujących systemy operacyjne Android oraz iOS. Użytkownik jest w stanie nagrywać testy, które mogą być zapisane jako skrypt do ponownego użycia. Program oferuje także integrację z innymi produktami takimi jak Jira czy też systemy do wersjonowania wersji [7].

SahiPro, oferujące automatyzację testów dla aplikacji internetowych, jest kolejnym komercyjnym narzędziem. Pozwala na nagrywanie testów, które są następnie dostępne do edytowania przez użytkownika. Wygenerowane testy mogą być zapisane jako funkcje w języku JavaScript celem ponownego użycia. Program wspiera głównie przeglądarki Firefox oraz Internet Explorer.

Katalon Studio jest samodzielnym narzędziem oferującym możliwość automatyzowania testów tworzonych przez użytkownika. Jest to rozwiązanie darmowe [8], oparte na silniku Selenium. Wszystko to zostało obudowane w interfejs, który zawiera najpotrzebniejsze elementy do tworzenia testów. Narzędzie to oferuje automatyzację testów aplikacji internetowych, mobilnych oraz serwisów internetowych. Program nie wymaga od użytkownika znajomości języków programistycznych. Korzystanie z narzędzi deweloperskich w przeglądarce pomoże użytkownikom w sprawniejszym tworzeniu testów.

Tabela 1: Dostępne opcje narzędzi do testowania aplikacji internetowych

	Selenium IDE	Selenium WebDriver	Test Complete	Sahi-Pro	Katalon Studio
Licencja	Darmowa	Darmowa	Płatna	Płatna	Darmowa
Testy UI	Tak	Tak	Tak	Tak	Tak
Testy aplikacji mobilnych	Nie	Nie	Nie	Tak	Tak
Testy aplikacji desktopowych	Nie	Nie	Tak	Tak)	Tak
Testy Serwisów internetowych	Nie	Nie	Nie	Nie	Tak)
Nagrywanie testów	Tak	Nie	Tak	Tak	Tak
Tworzenie testów od podstaw	Tak	Tak	Tak	Tak	Tak
Modyfikacja testów	Tak	Tak	Tak	Tak	Tak
Raportowanie wyników	Tak	Możliwe	Tak	Tak	Tak

3.4. Narzędzia do testów REST API

Do testów REST API wybrane zostały trzy narzędzia: Katalon Studio, Postman i SoapUI. Wszystkie te programy są dostępne za darmo. Umożliwiają testowanie poprawności działania aplikacji serwisów internetowych.

Postman jest najbardziej popularną z testowanych aplikacji. Początkowo dostępny jako rozszerzenie do przeglądarki Chrome, aktualnie udostępnia również oprogramowanie na systemy Windows. Wyróżnia się przejrzystym i czytelnym interfejsem. Wspomaga również pracę zespołów programistycznych i testerskich, umożliwiając współdzielenie tworzonych projektów [9] oraz szeroki zakres integracji z innymi aplikacjami. Jego zaletą jest dostarczenie przez twórców dokładnej dokumentacji oraz istnienie dużego wsparcia społeczności użytkowników.

Katalon Studio udostępnia funkcjonalność testowania serwisów internetowych, pomimo, że nie jest narzędziem dedykowanym do tego celu. Nie wymaga znajomości języków programowania do tworzenia przypadków testowych, ale umożliwia również generowanie skryptów w języku Groovy. Zaletą narzędzia Katalon Studio jest możliwość integracji z wieloma aplikacjami wspomagającymi tworzenie projektów, takimi jak: Jira, Jenkins, Bamboo, a nawet Postman. Dla mniej zaawansowanych użytkowników udostępniony został pakiet przykładowych projektów, ułatwiający szybkie zapoznanie się z narzędziem.

Dzięki prostemu interfejsowi graficznemu i funkcjonalnościom na poziomie klasy korporacyjnej, SoapUI pozwala na szybkie i proste tworzenie zautomatyzowanych testów funkcjonalnych, regresyjnych oraz obciążeniowych. Ze względu na ściśle określoną strukturę tworzonego projektu umożliwia parametryzację każdego poziomu testów. Dostarcza użytkownikowi narzędzi do budowania skomplikowanych scenariuszy testowych, a jednocześnie nie wymaga od niego wiedzy z zakresu programowania, ponieważ definiowanie kolejnych kroków odbywa się przez dodawanie asercji. Najciekawszą z funkcjonalności udostępnianych przez to narzędzie jest możliwość generowania testów obciążeniowych i bezpieczeństwa. Ich wyniki przedstawiane są w czytelnej postaci wykresów i raportu [10].

4. Wyniki badania

Proces badania możliwości testowanego oprogramowania pozwolił na ocenę jego przydatności pod względem funkcjonalnym i niefunkcjonalnym.

4.1. Metodyka badawcza

Dla każdego narzędzia do testów aplikacji internetowych przygotowano następujące zasady podczas wykonywania scenariuszy testowych:

- za pomocą każdego narzędzia zostało stworzone 10 kont pocztowych,
- nazwa tworzonego konta pocztowego jest w formacie: MGR_NazwaNarzedzia_NrKonta,
- **NazwaNarzedzia** oznacza uproszczoną nazwę testowanego narzędzia, a **NrKonta** kolejno tworzone konto od 0 do 10. Konto posiadające „0” jako **NrKonta** oznacza, że dane konto powstało podczas nagrywania procesu testowego.
- podczas tworzenia nowego konta pocztowego na portalu Wirtualnej Polski, istnieje szansa na to, że system będzie wymagał od użytkownika uzupełnienia mechanizmu zabezpieczającego CAPTCHA. Takie próby zaraportowano i następnie zatrzymano, po czym proces ponawiano,
- po stworzeniu testu poprzez nagranie, odtwarzany jest proces w celu zweryfikowania poprawności pobierania elementów przez badany program.

Stworzono dwa przykładowe scenariusze testowe, które są przedstawione w tabelach 2 oraz 3. Testy były tworzone i wykonywane na komputerze stacjonarnym.

Tabela 2: Scenariusz tworzenia konta pocztowego

Nazwa		Tworzenie konta pocztowego
Cel		Sprawdzenie poprawności tworzenia nowego konta pocztowego
Warunki wstępne		Wylogowany użytkownik z domeny pocztowej
Wymagania		Brak danego konta w systemie
Tworzenie konta pocztowego		
LP	**Opis wykonanych czynności**	**Oczekiwany Wynik**
1.	Wejście na stronę www.wp.pl	Przeniesienie do witryny portalu WP
2.	Kliknięcie w odnośnik poczty email	Przeniesienie na witrynę poczty email portalu WP
3.	Kliknięcie w odnośnik zakładania nowego konta	Przeniesienie na witrynę zakładania konta pocztowego
4.	Uzupełnienie danych	Odpowiednie wypełnienie formularza
5.	Zaznaczenie elementu CAPTCHA	Zaznaczenie pola wyboru CAPTCHA
6.	Kliknięcie przycisku do utworzenia konta	

Tabela 3: Scenariusz logowania do poczty

Nazwa		Zalogowanie się do konta pocztowego
Cel		Sprawdzenie poprawności logowania do konta pocztowego
Warunki wstępne		Wylogowany użytkownik z domeny pocztowej
Wymagania		Powinno istnieć konto w systemie
Logowanie do konta pocztowego		
LP	**Opis wykonanych czynności**	**Oczekiwany Wynik**
1.	Wejście na stronę www.wp.pl	Przeniesienie do witryny portalu WP
2.	Kliknięcie w odnośnik poczty email	Przeniesienie na witrynę poczty email portalu WP
3.	Uzupełnienie danych logowania	Uzupełnione wartości
4.	Uzupełnienie danych	Odpowiednie wypełnienie formularza
5.	Kliknięcie przycisku do logowania	Przejście do witryny poczty użytkownika.
6.	Sprawdzenie czy istnieje przycisk Wyloguj	Istnieje przycisk do wylogowania
7.	Wylogowanie użytkownika poprzez naciśnięcie przycisku wyloguj	

Tworzenie testów dla zdefiniowanych scenariuszy testowych wykonano na zainstalowanych i skonfigurowanym środowisku. Rozpoczęto od tworzenia testów dla scenariusza służącego do tworzenia nowych kont pocztowych na portalu Wirtualna Polska.

Nagranie testu w SeleniumIDE wygenerowało odpowiednie kroki testowe. Kompletny scenariusz zduplikowano i zmodyfikowano wartości dotyczące tworzonego konta. Uruchomiono scenariusz testowy, który wykonał zdefiniowane wcześniej testy. Czasy ich wykonywania przedstawia tabela 4.

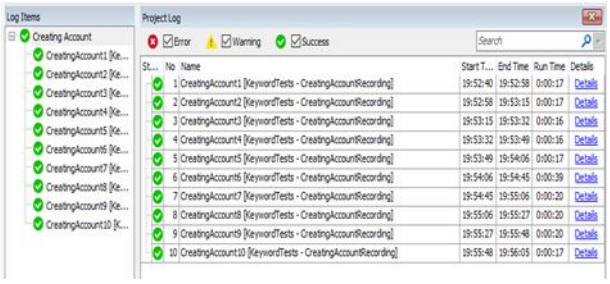
Tabela 4: Czasy tworzenia nowych kont za pomocą SeleniumIDE

Test	1	2	3	4	5	6	7	8	9	10	Ręczne nagrywanie
Czas [s]	9	10	9	10	9	9	9	10	10	10	29

Wyznaczone czasy w testach automatycznych oscylowały przy wynikach od 9 do 10 sekund, dając średni czas 9.5s. Przeprowadzając ręczne nagrywanie osiągnięto wynik 29s.

SeleniumWebDriver został wykorzystany dla języka C#, co wymagało użycia zintegrowanego środowiska deweloperskiego jakim jest Visual Studio 2017 w wersji Community. Po stworzeniu projektu, pobraniu odpowiedniej biblioteki oraz pobraniu sterownika przeglądarki Chrome w wersji 77, przystąpiono do zdefiniowania kroków testowych. Głównym ciałem klasy jest klasa testowa, która posiada w swojej definicji metadane pozwalające na parametryzację tego przypadku testowego.

Po poprawnym skonfigurowaniu kroków, przeprowadzono testy dla scenariusza tworzenia nowych kont, które się nie powiodły. System detekcji automatyzacji Captcha działa prawidłowo zapobiegając używania zautomatyzowanych testów, co niestety uniemożliwiło poprawne przeprowadzenie tego scenariusza testowego.



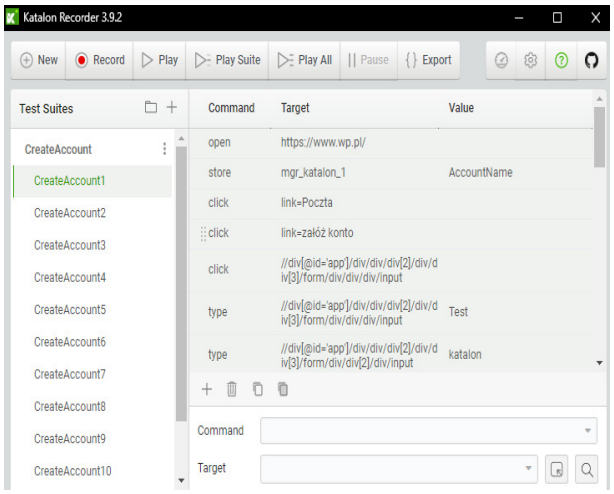
Rysunek 1: Raport po wykonaniu scenariusza testowego

W aplikacji TestComplete stworzono projekt i przystąpiono do nagrania testu. Po wykonaniu scenariusza, otrzymano wymagane kroki testowe. Następnie stworzono scenariusz testowy wewnątrz programu i zdefiniowano testy dla poszczególnych nowych kont. Po uruchomieniu i pomyślnym zakończeniu został wygenerowany raport widoczny na rysunku 2.

Uzyskane czasy oscylowały w przedziale 16-20 sekund, ze średnią 17.77s. Test numer 6, którego czas wyniósł 39 sekund został świadomie odrzucony podczas analizy, ponieważ system portalu wykrył zautomatyzowany system tworzenia i wymagał uzupełnienia elementu Captcha.

W programie SahiPro, po uruchomieniu narzędzia wybrano kontroler, za pomocą którego nagrano test według scenariusza do tworzenia nowych kont pocztowych. Następnie wygenerowane kroki testowe zostały umieszczone w odpowiedniej funkcji. Po wykonaniu tego skryptu, żaden test się nie powiódł. Wynika to z tego powodu, że testowane narzędzie wymusza korzystanie z autorskiego proxy i powoduje wykrycie przez system Captcha, przez co testy się nie powiodły.

W Katalon Studio stworzono projekt i za pomocą dostępnego rozszerzenia do przeglądarki rozpoczęto nagrywanie testów. Nagrany test został zduplikowany, a następnie zmodyfikowany pod wymagane dane testowe, co przedstawia rysunek 3.



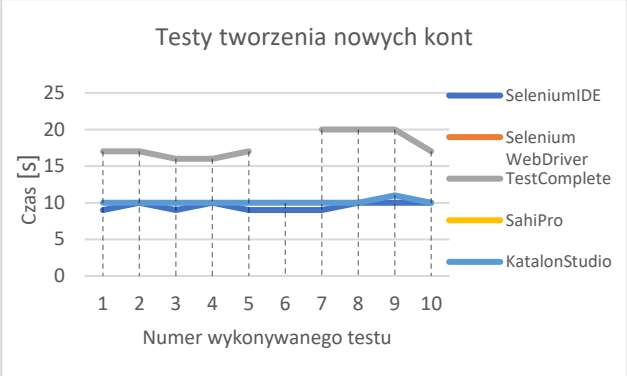
Rysunek 2: Stworzony scenariusz testowy w KatalonStudio

Uruchomiony scenariusz testowy pomyślnie utworzył konta pocztowe, a czas przeprowadzenia testów przedstawia tabela 5.

Tabela 5: Czasy tworzenia nowych kont w Katalon Studio

Test	1	2	3	4	5	6	7	8	9	10	Ręczne nagrywanie
Czas [s]	10	10	10	10	10	10	10	10	11	10	29

Uzyskane czasy kształtują się na poziomie 10s, w jednym przypadku wyniósł on 11s.
Porównano również czasy wykonania scenariusza w poszczególnych narzędziach, co przedstawia wykres na rysunku 4. Test numer 6 dla narzędzia TestComplete nie został uwzględniony ze względu na opisany wcześniej błąd.



Rysunek 3: Wykres czasów wykonywania scenariusza tworzenia nowych kont w poszczególnych narzędziach

W SeleniumIDE nagrano test kierując się scenariuszem testowym. Następnie wygenerowany test zduplikowano i zmodyfikowano pod wymagane dane. Po uruchomieniu scenariusza testowego, poszczególne przypadki wykonywały się w czasie przedstawionym w tabeli 6. Logowanie ręczne i wpisanie skomplikowanego hasła zajęło 14s. Dzięki automatyzacji czas ten spadł do średniej wartości 4.8s.

Tabela 6: Czasy logowania do kont za pomocą SeleniumIDE

Test	1	2	3	4	5	6	7	8	9	10	Ręczne logowanie
Czas [s]	5	5	5	5	5	5	4	4	4	6	14

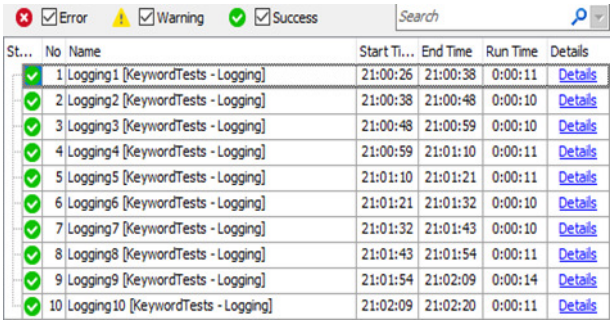
Po ręcznym utworzeniu kont dla tego narzędzia, został opracowany automatyczny test wykonujący proces logowania.

Czasy wykonania poszczególnych testów są przedstawione poniżej (tabela 7). Średni czas wyniósł 11.03 sekundy.

Tabela 7: Czasy wykonywania testów logowania w Visual Studio 2017 przy użyciu Selenium WebDriver

Test	1	2	3	4	5	6	7	8	9	10
Czas [s]	12248	12575	12227	12852	8044	8009	8533	13856	9786	12250

W TestComplete do wcześniej utworzonego projektu, dodano kolejny projekt testowy pod nazwą *Logging*, którego celem jest wykonanie scenariusza służącego do zweryfikowania poprawności logowania w systemie.



Rysunek 4: Raport dotyczący wykonanych testów logowania w TestComplete

Wykonanie testów wygenerowało raport, na podstawie którego możliwe jest sprawdzenie czy ukończyły się one prawidłowo, co ukazano na rysunku 5. Średni czas logowania wyniósł 11.3s.

W SahiPro przystąpiono do procesu nagrywania testu logowania użytkownika. Następnie zmodyfikowano wygenerowany skrypt w sposób umożliwiający parametryzację.

Proces wykonywania testów przeszedł częściowo pomyślnie. Ostatni krok testowy się nie powiódł z powodu błędu działania serwera wykorzystywanego przez firmę SahiPro. Wykonane testy zostały przy średnim czasie 15.59 sekundy, co jest przedstawione w tabeli 8.

Tabela 8: Czasy logowania przy użyciu SahiPro

Test	1	2	3	4	5	6	7	8
Czas [s]	11,45	15,12	17,36	17,15	16,49	15,67	16,89	14,6

W Katalon Studio nagrano proces logowania użytkownika. Stworzono scenariusz testowy w programie, dodano do niego zduplikowane i zmodyfikowane testy. Po uruchomieniu i pomyślnym ukończeniu scenariusza, zebrano czasy wykonywania poszczególnych testów do tabeli 9. Średni czas wyniósł 5.9s.

Tabela 9: Czasy logowania przy użyciu Katalon Studio

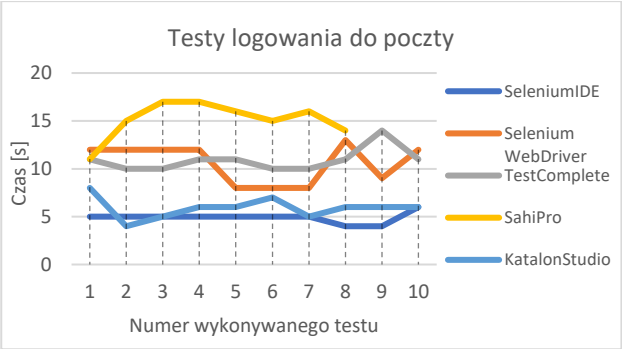
Test	1	2	3	4	5	6	7	8	9	10	Ręczne logowanie
Czas [s]	8	4	5	6	6	7	5	6	6	6	16

Porównano również czasy wykonania scenariusza w poszczególnych narzędziach, co przedstawia wykres na rysunku 6.

Wykonywanie testów REST API odbyło się przy pomocy darmowego zewnętrznego serwisu http://petstore.swagger.io/. Każde testowane narzędzie wykonało zdefiniowany w przypadkach testowych zestaw zapytań w celu oceny jego możliwości.

W ramach testów REST API zbadane zostały funkcjonalności takie, jak:

- tworzenie żądań GET z parametrami,
- tworzenie żądań POST z parametrami w ciele zapytania,
- parametryzacja testów,
- generowanie skryptów do testów automatycznych.



Rysunek 5: Wykres czasów wykonywania scenariusza logowania w poszczególnych narzędziach

4.2. Analiza porównawcza narzędzi do testów funkcjonalnych aplikacji internetowych

Narzędzia zostały porównane pod wieloma aspektami w skali od 1 do 5 kolejno oznaczając jako bardzo słaba ocena, słaba, średnia, dobra oraz bardzo dobra ocena. Porównanie zebrano w postaci tabeli.

Tabela 10: Porównanie narzędzi do automatyzacji testów

	SeleniumIDE	**Selenium WebDriver**	**TestComplete**	**SahiPro**	**Katalon Studio**
Początkowa instalacja	Wystarcza zainstalowanie rozszerzenia w przeglądarce 5	Potrzeba dodatkowego środowiska Pobranie odpowiednich bibliotek 3	Wystarczy instalacja samodzielnego narzędzia 5	Instalacja narzędzia poprzez użycie odpowiednich komend 3	Pobranie spakowanego narzędzia i rozpakowanie go. Dodatkowa instalacja rozszerzenia w przeglądarce 4
Ilość dodatkowej konfiguracji	Po uruchomieniu wystarczy stworzyć projekt 5	Pobranie odpowiednich bibliotek, sterowników przeglądarek 3	Wystarczy stworzenie projektu 5	Uruchomienie przeglądarki i kontrolera 5	Wystarczy stworzenie projektu 5
Czas wdrożenia	Intuicyjne GUI, pozwala na natychmiastowe tworzenie testów 5	Należy znać jeden ze wspieranych języków programowania 3	Nagrywanie testów nie wymaga większej wiedzy na temat używania narzędzia 4	Nagrywanie testów nie wymaga większej wiedzy na temat używania narzędzia 4	Nagrywanie testów nie wymaga większej wiedzy na temat używania narzędzia 4
Znajomość programowania	Wymaga podstawowej znajomości programowania przy edycji wygenerowanego skryptu 3	Wymaga zaawansowanej znajomości programowania 2	Wymaga podstawowej znajomości programowania przy edycji wygenerowanego skryptu 3	Do edytowania wygenerowanego skryptu wymaga podstawowej znajomości JavaScript 3	Wymaga podstawowej znajomości programowania przy edycji wygenerowane-go skryptu 3
Generowanie raportu	Brak możliwości generowania raportu 1	Brak domyślnego mechanizmu, dostępne jedynie przez środowisko IDE 3	Możliwość wygenerowania raportu ze szczegółami 5	Możliwość wygenerowania raportu ze szczegółami 5	Możliwość wygenerowania raportu ze szczegółami 5
Nagrywanie scenariuszy	Jest to podstawowa funkcja narzędzia 5	Brak możliwości nagrywania 1	Jest to podstawowa funkcja narzędzia 4	Jest to podstawowa funkcja narzędzia 5	Jest to podstawowa funkcja narzędzia 5
Generowanie skryptów z nagranych scenariuszy	Po zapisaniu projektu można edytować plik, w którym znajduje się skrypt napisany w Selenese 2	Brak możliwości nagrywania 1	Możliwość wygenerowania skryptu z nagranego testu szczegółami 5	Możliwość wygenerowania skryptu z nagranego testu szczegółami poprzez edycje 3	Możliwość wygenerowania skryptu z nagranego testu szczegółami 5
Możliwość ponownego użycia skryptów	Można używać dany test poprzez skopiowanie jego zawartości 4	Jest możliwość wydzielenia odpowiednich metod 4	Jest możliwość wydzielenia odpowiednich skryptów 4	Jest możliwość wydzielenia odpowiednich metod 4	Jest możliwość wydzielenia odpowiednich metod 4
Integracja z innymi narzędziami	Brak możliwości integracji z innymi narzędziami 1	Możliwe integracje zależne od używanego środowiska 5	Domyślna integracja z Jira, CI/CD oraz repozytoriami GIT 4	Skomplikowane korzystanie z używanych bibliotek 2	Możliwość integracji z CI/CD, repozytoriami GIT i wieloma innymi 4
Średnia ocena	3,44	2,78	4,33	3,78	4,33

4.3. Analiza porównawcza narzędzi do testów REST API

W tabeli 11 zebrane zostało podsumowanie najważniejszych aspektów i funkcjonalności dla testowanych narzędzi REST API.

Tabela 11: Porównanie narzędzi do testów REST API

	Katalon Studio	Postman	SoapUI
Czas wytworzenia testu	Szybkie, nieskomplikowany interfejs.	Bardzo szybkie, tworzenie testów jest proste i intuicyjne.	Stworzenie testu wymaga dokładnego zapoznania się z interfejsem.
Znajomość programowania	Przydatne do weryfikacji poprawności odpowiedzi i pisania testów w Groovy, nie wymagane przez dostępność gotowych fragmentów kodów lub dodawania z poziomu odpowiedzi.	Przydatna, ale nie wymagana ze względu na możliwość generowania gotowych fragmentów kodu, skrypty tworzone w JavaScript.	Wymagane do tworzenia skryptów w Groovy.
Intuicyjność	Bardzo intuicyjny interfejs, duża ilość ułatwień i podpowiedzi.	Bardzo intuicyjny interfejs, pozwala na pracę bez konieczności wcześniejszego wdrożenia.	Wymaga poznania struktury aplikacji, mało intuicyjny interfejs.
Prostota / zaawansowanie	Łączy przyjazny interfejs z dużą liczbą funkcjonalności.	Bardzo proste narzędzie.	Narzędzie idealne dla bardziej zaawansowanych użytkowników.
Parametryzacja	Możliwość tworzenia zaawansowanych skryptów do zapytań i testów.	Konfiguracja środowisk uruchomieniowych z określonymi parametrami.	Parametryzacja na każdym poziomie testów.
Tworzenie testów automatycznych	Przy pomocy edytora lub skryptów.	Bardzo szybkie, przypisane do zapytania.	Testy grupowane i segregowane w określonej strukturze, tworzone przez asercje.
Jakość raportu z testów	Tylko podstawowe informacje.	Raport czytelny i szczegółowy.	Zawiera oczekiwane informacje, ale przedstawione w mniej przystępnej formie.
Integracja z innymi narzędziami	Największe możliwości integracji, już z poziomu aplikacji.	Szerokie możliwości integracji.	Integracja z użyciem SoapUI CLI.
Współdzielenie projektów	Wyłącznie przez GIT/SVN.	Możliwość współdzielenia przestrzeni roboczych z różnymi uprawnieniami, system komentarzy.	Brak.
Tworzenie serwisów mock	Brak.	Dostępne.	Dostępne.
Dodatkowe zalety	Narzędzie współpracuje z dużą ilością zewnętrznych aplikacji, nie jest dedykowane wyłącznie do testów serwisów REST.	Monitor – uruchamianie testów według określonego harmonogram.	Testy bezpieczeństwa i obciążeniowe.

5. Wnioski

Celem niniejszego artykułu była analiza porównawcza rozwiązań wykorzystywanych w testowaniu automatycznym. Na podstawie przeprowadzonych badań należy podkreślić, że nie można wyznaczyć najlepszego rozwiązania do tworzenia zautomatyzowanych testów. Najważniejszym elementem wyboru odpowiedniego narzędzia, jest określenie indywidualnych wymagań, sposobu testowania produktu, parametryzacji testów, powtarzalności. Po ustaleniu tych elementów można przystąpić do wyboru.

Dla twórców własnego oprogramowania, którzy posiadają kontrolę nad używanymi funkcjonalnościami, lepszym rozwiązaniem może się okazać pisanie własnych testów przy wykorzystaniu Selenium WebDriver. Dostarcza ono możliwość integracji ciągłej integracji i wdrażania oprogramowania przez użycie używanego zintegrowanego środowiska deweloperskiego. Przez opcję definiowania danych, które mają zostać użyte podczas testów oraz stworzenia własnego systemu raportowania jest to narzędzie odpowiadające bardziej wymagającym zespołom.

Subiektywnie oceniając uniwersalnym narzędziem, które sprawdzi się w większości sytuacji i ma najmniejszy próg wejścia technicznego, będzie narzędzie oferowane przez firmę SmartBear. Kolejnym rozwiązaniem będzie zarówno dodatek do przeglądarki z narzędzia KatalonStudio oraz SeleniumIDE. Pisanie własnych rozwiązań przy użyciu Selenium WebDriver jest przeznaczone dla specyficznych rozwiązań, które przyciągną użytkowników, którzy chcą dowolnie kontrolować proces testowy. SahiPro jest to narzędzie oferujące najsłabszą dokumentację, proces tworzenia testów oraz obsługę techniczną.

W przypadku testów serwisów internetowych nie jest możliwe jednoznaczne określenie rankingu testowanych aplikacji. Każda z nich posiadała zarówno funkcjonalności wyróżniające ją na tle pozostałych, jak i elementy, których brak zdecydowanie wpływał niekorzystnie na jej odbiór przez użytkownika. Wykonanie testów trzech aplikacji: Postman, SoapUI oraz Katalon Studio pozwoliło jednak na subiektywne stwierdzenie najważniejszych elementów przydatnych podczas tworzenia testów aplikacji REST:

- możliwość szybkiego tworzenia zapytań,
- dostępność nieskomplikowanej metody automatyzacji i parametryzacji testów,
- generowanie czytelnych raportów z przeprowadzonych weryfikacji,
- możliwość integracji z innymi narzędziami.

Główną zaletą aplikacji Postman była prostota i przejrzystość interfejsu. Narzędzie to sprawdzi się idealnie podczas pracy w średnich zespołach testerskich, ponieważ umożliwia udostępnianie przestrzeni roboczych, a także pozwala na definiowanie zestawów zmiennych i parametrów dla różnych środowisk uruchomieniowych. Nie jest jednak najlepszym wyborem dla zaawansowanych użytkowników, ponieważ nie pozwala na zaawansowaną edycję skryptów. Z tym aspektem radzi sobie bardzo dobrze Katalon Studio, które umożliwia edycję zarówno zapytań, jak i testów. Jest jednak bardzo wąsko wyspecjalizowane i wymaga integracji z wieloma innymi narzędziami (takimi jak systemy kontroli wersji i współdzielenia projektów, generowania raportów, itp.). Nie pozwala również na tworzenie sztucznych API dzięki serwerom typu mock. SoapUI jest zdecydowanie narzędziem przeznaczonym dla bardziej zaawansowanych użytkowników. Świadczy o tym poziom skomplikowania interfejsu oraz tworzenia testów automatycznych. Z kolei dużą zaletą tego narzędzia jest możliwość parametryzacji testów na różnych, niezależnych poziomach.

Literatura

[1] Stowarzyszenie Jakości Systemów Informatycznych. Certyfikowany tester. Sylabus poziomu podstawowego certyfikatu ISTQB 2018. Wersja 1.0.1. 2019

[2] R. Pawlak. Testowanie oprogramowania. Podręcznik dla początkujących. Helion, 2014.

[3] https://www.unity.pl/blog/dlaczego-innowacyjnego-projektu-nie-zrealizujesz-za-pomoca-modelu-waterfall/ [10.11.2019]

[4] ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

[5] https://smartbear.com/learn/automated-testing/best-practices-for-automation [10.11.2019]

[6] https://www.seleniumhq.org/docs/03_webdriver.jsp [10.11.2019]

[7] https://smartbear.com/product/testcomplete/features/ [10.11.2019]

[8] https://docs.katalon.com/katalon-studio/docs/overview.html [10.11.2019]

[9] https://www.getpostman.com/product/workspaces [10.11.2019]

[10] https://kordiankicza.pl/soapui-dla-zielonych-cz-1-tworzymy-pierwszy-projekt / [10.11.2019]