Piszę testy automatyczne, kim jestem?

Praca jako tester automatyzujący to ścieżka rozwoju którą wybiera coraz więcej osób. Co robi automatyk, jakie są pułapki i korzyści z tej specjalizacji?

1. Wprowadzenie

W świecie IT, który nas obecnie otacza, istnieje pewna tendencja skłaniająca testerów oprogramowania do pójścia w kierunku automatyzacji testowania. Inaczej mówiąc, testerzy manualni chcą po prostu jak najszybciej dostać możliwość pisania testów automatycznych.

Takie podejście nie jest wcale złe, ponieważ poszerzanie swojej wiedzy oraz nabieranie umiejętności nikomu nie zaszkodziło, a wręcz przeciwnie, pomogło. Nierzadko się zdarza, że tester otrzymuje (na własną prośbę) możliwość automatyzacji przypadków testowych. Istnieją też takie projekty, gdzie ten rodzaj testów jest wymagany jako część produktu. Wtedy albo zatrudniamy osobę, która potrafi automatyzować, albo zajmuje się tym tester, który pracuje przy projekcie.

Szansa automatyzowania testów jest celem, do którego dąży wielu z nas. Ale okazuje się, że pierwsze kroki z jakimi przyjdzie nam się zmierzyć nie są takie proste jak wypowiedzenie słów: „będę pisał automaty!!!". Niestety, ale można wpaść tutaj w pewną pułapkę. Może się okazać (co często ma miejsce) że utrzymanie skryptów automatyzujących testy oraz ich odporność na najdrobniejsze zmiany nie będą już takie proste.

2. Piszę testy automatyczne, nie testuję manualnie?

Załóżmy, że jesteśmy w sytuacji, gdy piszemy już tylko testy automatyczne, musimy o nie dbać oraz przeglądać ich wyniki. Powstał projekt, który zawiera te testy.

Testerzy manualni mają na ogół taką cechę, że dość wnikliwie przyglądają się temu co testują. Oprócz sprawdzania wymagań, testują często eksploracyjnie i takie postępowanie jest dla nich normą. Niestety, jeśli już piszemy automatyczny test, to sprawdza on jedną konkretną ścieżkę. Okazuje się też, że napisanie jednego testu trwa nieraz cały dzień. Nie idzie to tak szybko, jak testowanie manualne. Taki stan rzeczy może skłonić testera do zadania sobie pytania: *„Czy ja mam dobre tempo? Czy ja nie piszę za wolno?* Przecież będzie około 200 testów regresji." Okazuje się jednak, że tak to niestety wygląda w praktyce. Testy manualne znajdują błędy, a my pisząc nowe testy pokrywamy taką ścieżkę, która ma zwracać pozytywny wynik. Testowana aplikacja na ogół „żyje" (wciąż jest rozwijana). Oprócz tego, że piszemy nowe testy, musimy dostosować stare do zmieniającej się aplikacji. Pojawia się kolejne pytanie: *„Co zrobić, żeby pogodzić dostosowanie testów do aplikacji, która żyje oraz pisać nowe testy tak, aby zmieścić się w rozsądnym czasie?".*

Tester, który nie ma obok siebie osoby doświadczonej w tworzeniu skryptów testowych, został tak naprawdę rzucony na głęboką wodę. Nie ma przed sobą łatwego zadania. Może popełnić wiele błędów, ponieważ brak mu doświadczenia oraz znajomości dobrych praktyk. Automatyzacja nie jest możliwa do osiągnięcia bez dobrej znajomości narzędzia - frameworka, który umożliwia oskryptowanie przypadków testowych. Nie jest to również możliwe bez znajomości podstaw języka programowania, w którym pisane są testy automatyczne.

3. Jaka jest moja rola w projekcie?

Osoba, która zajmuje się już tylko automatyzacją przypadków testowych, boryka się z problemami opisanymi w poprzednim rozdziale. Wspomniane problemy skłaniają człowieka do myślenia nad tym, czy pracuje efektywnie, czy czas który poświęca na pisanie nowych skryptów oraz na utrzymanie starych testów jest rozsądny.

Tester, który zaczął przygodę z testami automatycznymi ostatecznie dochodzi do wniosku, że tak naprawdę już nie testuje, natomiast zajmuje się tylko pisaniem skryptów oraz ich utrzymaniem. Taka sytuacja zmusza człowieka do zadawania sobie pytania: *„Czy ja jestem testerem czy już nie? Nie piszę już przypadków, nie testuję tak jak to było to za czasów wykonywania scenariuszy testowych tylko manualnie. Muszę pisać skrypty i je utrzymywać i niezbyt szybko to idzie. To nie jest testowanie! Czy to co robię jest dobre? Czy tak powinno być, czy tak to wygląda w innej firmie?”*.

Początki nie muszą być łatwe i na ogół nie są ale nie powinniśmy się poddawać. W IT ceni się osoby o otwartych umysłach, chcące się uczyć oraz mówiące o problemach, które pojawiają się podczas pracy. Zakomunikowanie w swoim zespole, że istnieje coś z czym nie dajemy sobie rady nie jest wstydem, a wręcz przeciwnie. Jeśli zaczynamy nowy rozdział w karierze, to mogą się pojawić nieprzewidziane sytuacje. Problemy powinny być zgłaszane jak najwcześniej. Może się okazać, że problem utrzymania testów automatycznych nie istnieje tylko po stronie osoby automatyzującej przypadki. Może się również okazać, że proces testowy nie jest dobrze dobrany do projektu i należy zmienić podejście do tworzonych do tej pory testów wykonywanych automatycznie.

W Internecie istnieje sporo wpisów na temat dobrych praktyk związanych z automatyzacją testowania. Można znaleźć literaturę związaną z tym zagadnieniem. Istnieją portale takie jak *Facebook*, *LinkedIn* lub inne, gdzie ludzie opisują swoje problemy i otrzymują na nie odpowiedzi. Często również w pracy mamy kogoś, kogo można poprosić o pomoc. Tak więc zamiast narzekać, powinniśmy zgłaszać problemy albo próbować je samodzielnie rozwiązywać. Dobre testy automatyczne oraz dobry proces testowy wnoszą wiele wartości, z których skorzysta cały zespół i klient zamawiający oprogramowanie.

4. Dobre praktyki

4.1 Podstawy programowania oraz znajomość frameworka testowego

Tester automatyzujący musi dobrze znać podstawy języka programowania, w którym pisze testy. Oprócz tego powinien też dobrze znać metody, które dostarcza framework wykorzystany do automatyzacji. Dobra znajomość metod języka oraz frameworka sprawi, że nie będziemy pisać zbędnych metod czy algorytmów, co może czasami przypominać „wyważanie otwartych drzwi”. Dobrze jest wiedzieć, z czego można skorzystać oraz co oferuje zastosowany język programowania oraz framework. Taka wiedza pozwala na duże usprawnienie, przyśpieszenie pracy i przede wszystkim przynosi satysfakcję tworzącemu skrypty.

4.2 Niezależność testów

Test automatyczny nie powinien zależeć od zmian wprowadzonych przez inny test. Musi być na nie odporny. W przeciwnym przypadku, gdy taki test się nie powiedzie, to kolejne zależne od niego

testy automatyczne również się nie powiodą, a to z kolei daje niewiarygodne wyniki. Niezależność pozwala nam na uruchamianie różnych testów równolegle, o ile umożliwia to sam zastosowany framework np. gdy chcemy uruchomić testy na różnych platformach/urządzeniach, strategia często stosowana podczas testowania aplikacji mobilnych.

4.3 Korzystanie ze sprawdzonych wzorców

Używanie sprawdzonych wzorców np. *Page Object Pattern.* Takie podejście zakłada, że każda strona/ ekran/ wspólne elementy na ekranach aplikacji są osobnym obiektem. Ułatwia to utrzymanie testów, ponieważ pisane w taki sposób selektory oraz metody mogą być wielokrotnie użyte. W książce pt. „*Ciągłe dostarczanie oprogramowania*" autorów J. Humble'a, D. Farley'a w rozdziale 8 zaprezentowany jest wzorzec, który stanowi uogólnienie podejścia *Page Object Pattern*. Autorzy przedstawili podejście, które dzieli ekran na sterowniki w taki sposób, aby można było redukować poziom zależności testów od GUI. Zachęcam do lektury osoby pragnące pogłębić wiedzę w tym obszarze.

4.4 Lokalizacja elementów za pomocą umownych atrybutów

Lokalizacja elementów, na których wykonujemy akcje powinna się odbywać za pomocą dedykowanych atrybutów, np. *id* lub klasy zaczynające się od wyrażenia *e2e*. Dodawanie takich atrybutów można wymusić podejściem procesowym lub podczas tworzenia GUI. Wówczas programiści będą wiedzieli, że dany atrybut nie może zostać zmieniony ani usunięty.

4.5 Asercje w testach automatycznych

Podejście dotyczące asercji w testowaniu automatycznym powinno być dobrze przemyślane. Nie będzie to możliwe, jeśli asercje danego języka nie są znane osobie piszącej testy automatyczne. Asercje wykorzystuje się głównie do wykrycia sytuacji, w której program dociera do niepożądanego miejsca (oczekiwany rezultat jest inny niż spodziewany) lub do sprawdzenia wyświetlania elementów na ekranie (np. *element.isDisplayed()*). Jeżeli dana asercja powoduje przerwanie programu w przypadku niepowodzenia (zwrócona wartość *false*), to nie powinniśmy jej stosować tam, gdzie nie jest potrzebna (np. weryfikowanie poprawności argumentów metod publicznych).

4.6 Wszyscy odpowiadamy za jakość

Nikt nie powiedział, że za testy automatyczne albo ich wyniki powinni być odpowiedzialni tylko testerzy. Niestety, takie przekonanie funkcjonuje czasami w zespołach. Nie jest ono ani dobre ani właściwe. Jakość to w projekcie nasze dobro wspólne, do którego powinniśmy wszyscy dążyć. Jeśli tempo zmian aplikacji uniemożliwia osobie automatyzującej pisanie i jednoczesne utrzymanie skryptów, to sam proces utrzymania części skryptów można przenieść na programistów. Proces wytwórczy powinien być ciągle udoskonalany, a to oznacza, że w zespole powinniśmy definiować zmiany i wprowadzać korekty.

5. Podsumowanie

Głównym celem artykułu było przedstawienie perspektywy testera, który zaczyna przygodę z testami automatycznymi i nie zajmuje się testami manualnymi. Taka zmiana nie jest łatwa na samym początku. Daje sporo do myślenia nad swoją rolą w projekcie. W takim przypadku istnieją tylko dwie drogi:

- idziemy za ciosem i doprowadzimy siebie oraz testy automatyczne do poziomu, z którego będziemy zadowoleni,
- będziemy unikać automatycznych testów.

Publikacja została napisana na bazie własnych doświadczeń.

Recenzenci: Anna Justyna Rejrat, Klaudia Chmura, Łukasz Pawluś

Marek Żukowicz jest absolwentem matematyki na Uniwersytecie Rzeszowskim. Jest testerem oprogramowania w firmie Ailleron. Jego zainteresowania skupiają się wokół testowania, matematyki, AI, zastosowania modeli matematycznych w procesie testowania.