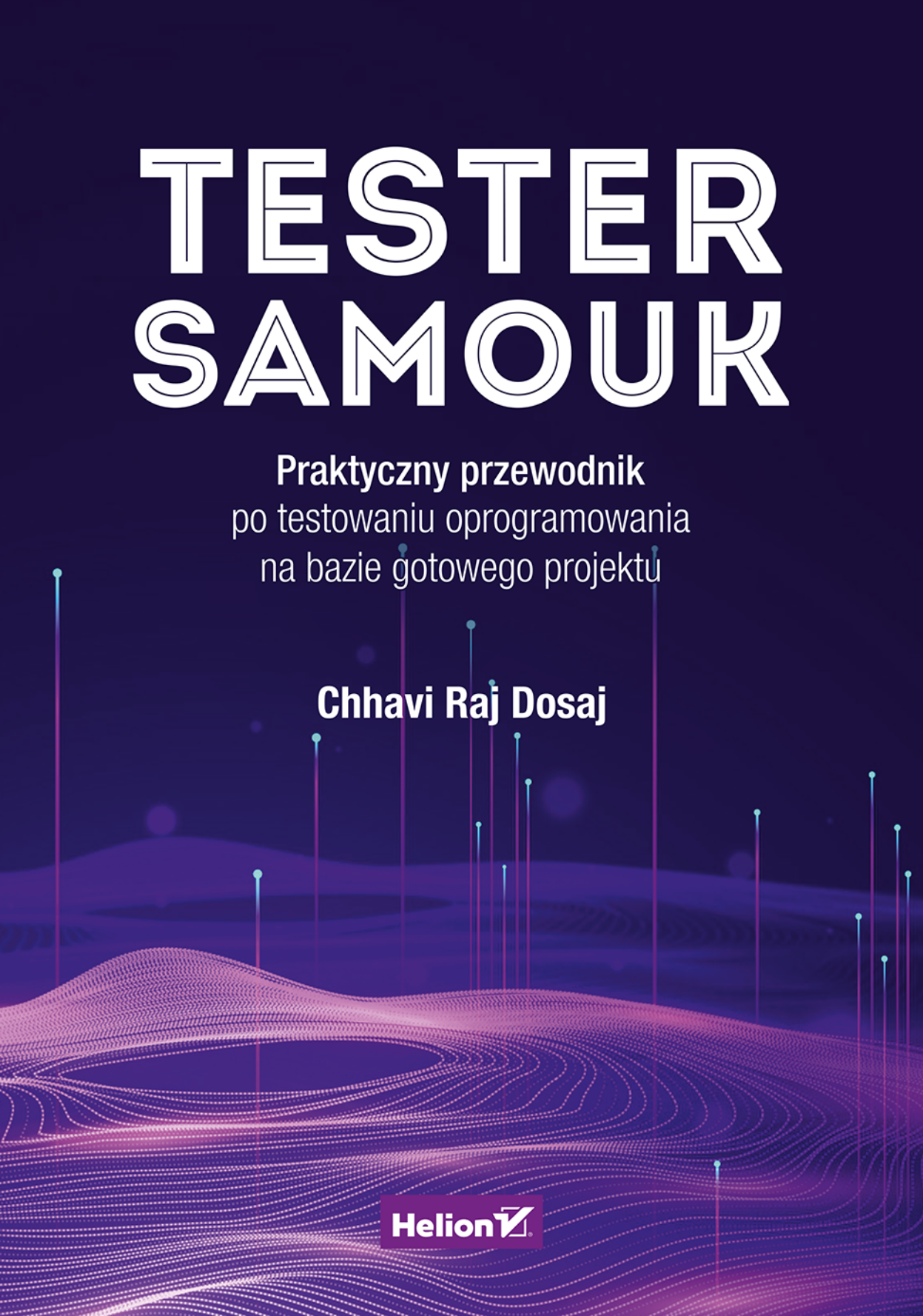
TESTER SAMOUK

Praktyczny przewodnik
po testowaniu oprogramowania
na bazie gotowego projektu

Chhavi Raj Dosaj



Helion

Tytuł oryginału: The Self-Taught Software Tester: A Step By Step Guide
to Learn Software Testing Using Real-Life Project

Tłumaczenie: Robert Górczyński

ISBN: 978-83-289-0971-7

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/tessam
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

Rozdział 1

Czym jest testowanie oprogramowania?

W tym rozdziale wyjaśnię, czym jest testowanie oprogramowania i dlaczego ma tak duże znaczenie.

Testowanie pomaga w wyszukiwaniu błędów w oprogramowaniu, zanim ujawnią się one w środowisku produkcyjnym

W dniu 4 czerwca 1996 roku nienazwana rakieta typu Ariane 5 została wystrzelona z Gujany Francuskiej przez Europejską Agencję Kosmiczną, a po zaledwie 37 sekundach lotu uległa zniszczeniu. To był pierwszy lot tej rakiety po ponad dziesięciu latach pracy, które pochłonęły kwotę 7 miliardów dolarów. Zniszczona rakieta i przenoszony przez nią ładunek były warte 500 milionów dolarów. Komisja śledcza zajęła się zbadaniem przyczyn eksplozji i w ciągu dwóch tygodni wydała raport. Okazało się, że przyczyną awarii rakiety był błąd oprogramowania w inercyjnym układzie odniesienia. Ten błąd doprowadził do zboczenia rakiety z jej pionowego kursu wznoszenia. To z kolei spowodowało zainicjowanie procedury samozniszczenia, zanim nieprzewidywalny tor lotu mógłby doprowadzić do znacznie poważniejszych problemów. Taki scenariusz nie został wzięty pod uwagę podczas przeprowadzania testów oprogramowania użytego w rakiecie. Gdyby bowiem został uwzględniony w testach, zostałby wcześniej wychwycony, co pozwoliłoby na usunięcie błędu.

Podstawowym celem testowania jest wyszukiwanie błędów, zanim doprowadzą one do awarii w środowisku produkcyjnym. Dlatego też testy są opracowywane w sposób, który umożliwi wychwycenie jak największej liczby usterek, zanim system trafi do środowiska produkcyjnego.

Testowanie gwarantuje, że system spełnia wymagania użytkownika

Jeden z urzędów komunikacji zdecydował się na zaimplementowanie nowego oprogramowania przeznaczonego do rejestrowania pojazdów. To nowe narzędzie zostało opracowane w taki sposób, aby wyeliminować całą ręczną pracę związaną z wydawaniem praw jazdy. W efekcie miał zostać skrócony czas obsługi obywatela. Do opracowania tego systemu została wybrana znana firma zajmująca się tworzeniem oprogramowania. Na początkowym etapie pracy urząd ściśle współdziałał z wykonawcą, aby dokładnie określić wymagania stawiane nowemu systemowi. Jednak na etapie testowania okazało się, że system nie był w stanie obsłużyć wielu funkcjonalności, do których realizacji był przeznaczony. Powodem takiego stanu rzeczy było to, że programiści nie zrozumieli poprawnie wymagań systemu, a tym samym nieprawidłowo zaimplementowali jego funkcjonalności.

Jednym z ważniejszych celów testowania jest zagwarantowanie, że wymagania użytkownika zostaną spełnione. W przeciwnym razie oprogramowanie nie będzie działać zgodnie z oczekiwaniami. Gdy rozpoczynają się prace nad projektem, wymagania przedstawione przez użytkownika stają się podstawą dla specyfikacji na przykład w postaci dokumentu wymagań biznesowych (ang. *business requirements document*, BRD). Tego rodzaju dokument służy jako punkt wyjścia do utworzenia przypadków testowych wykorzystywanych następnie podczas testowania projektu. *Przypadki testowe mogą być później wykonywane, aby zapewnić sprawdzenie wszystkich wymagań biznesowych za pomocą testów.* Dzięki temu testowanie może zagwarantować, że gdy z oprogramowania będą korzystać rzeczywiści użytkownicy, będzie ono działało zgodnie z oczekiwaniami.

Testowanie pomaga zminimalizować ryzyko

Jedna z firm tworzących oprogramowanie jest zaangażowana w opracowywanie automatycznego systemu kontroli lotu oraz systemu gier komputerowych. Testowanie obu tych systemów nie będzie odbywało się w dokładnie ten sam sposób, ponieważ niebezpieczeństwo związane z awarią systemu kontroli lotu jest bez porównania większe niż w przypadku systemu gier komputerowych.

Ryzyko jest nieodłącznym aspektem podczas tworzenia oprogramowania. Jeżeli system ulegnie awarii, skutkiem może być strata czasu, pieniędzy lub nawet obrażenia bądź śmierć człowieka, w zależności od tego, gdzie jest używane oprogramowanie. Ta niepewność nabiera coraz większego znaczenia wraz ze wzrostem poziomu złożoności systemu

i powagi konsekwencji jego awarii. Intuicyjnie można powiedzieć, że istnieje większe prawdopodobieństwo awarii znacznie bardziej skomplikowanego systemu i wówczas skutki tej awarii również będą poważniejsze. Co zostanie przetestowane oraz w jaki sposób odbędą się testy, ma bezpośredni związek ze wspomnianym ryzykiem. Większe ryzyko narzuca bardziej rygorystyczne i dokładniejsze testowanie.

Ryzyko produktu oznacza, że kompilacja oprogramowania nie spełnia celów, do których osiągnięcia została przygotowana. To może być skutkiem błędu w projekcie bądź nieprawidłowego opracowania funkcjonalności. Powszechnie spotykanym przykładem jest tutaj podatne na awarie oprogramowanie lub istnienie błędu w oprogramowaniu, który może doprowadzić do powstania szkody po stronie użytkownika.

Na początkowym etapie pracy nad projektem następuje określenie ryzyka produktu. Powstają wówczas przypadki testowe, które mają zapewnić pokrycie testami obarczonych ryzykiem fragmentów produktu oraz odpowiedni poziom testowania. Dzięki temu można mieć pewność, że ryzyko zostało zminimalizowane do akceptowalnego poziomu (bardziej rygorystyczne testy w przypadku elementów charakteryzujących się dużym ryzykiem i mniej rygorystyczne dla elementów o niewielkim ryzyku).

Testowanie zmniejsza ryzyko produktu, ponieważ większość problemów związanych z systemem zostaje odkrytych dzięki testom. Dlatego też zmniejsza się niebezpieczeństwo awarii systemu, gdy będzie on używany w środowisku produkcyjnym.

Testowanie pozwala określić poziom jakości oprogramowania i pomaga w podejmowaniu decyzji

W styczniu 2018 roku firma Apple ogłosiła, że wprowadzenie niektórych funkcji w systemie operacyjnym iOS 12 może się opóźnić nawet o rok. Na etapie testowania odkryto wiele problemów związanych z jakością oprogramowania w systemie iOS. Apple poważnie potraktował te informacje, ponieważ kładzie ogromny nacisk na jakość dostarczanego oprogramowania.

Interesariusze często chcą poznać poziom jakości oprogramowania, zanim podejmą decyzję o jego udostępnieniu użytkownikom. Informacje na temat jakości oprogramowania są dostarczane przez wyniki przeprowadzanych testów. Jednym z *kluczowych wskaźników poziomu jakości jest liczba znalezionych usterek* na etapie testowania oraz *liczba pozostałych usterek* wciąż wymagających usunięcia.

Na jakość wpływa także poziom spełnienia oczekiwań. Z jednej strony mamy pewne oczekiwania (wymagania) projektu, z drugiej zaś produkt, który powinien spełniać te oczekiwania. Testy pomagają ustalić, w jakim stopniu produkt był w stanie zbliżyć się do tych oczekiwań. Tego rodzaju informacje ułatwiają interesariuszom podejmowanie decyzji.

Testowanie gwarantuje, że system działa zgodnie z oczekiwaniami

Duży bank australijski uaktualnił swój system, aby pliki podatkowe i finansowe mogły być przez klientów biznesowych przekazywane bezpośrednio do systemu. W fazie początkowych testów testerzy przekazywali pliki w różnych formatach, aby zweryfikować poprawność działania systemu. System bez żadnych problemów poradził sobie z przetworzeniem tych plików, więc pracujący nad nim zespół był przekonany, że ta funkcjonalność działa prawidłowo. Następnie podczas systemu przez klientów biznesowych system ulegał awarii, gdy użytkownicy próbowali przekazywać rzeczywiste pliki podatkowe i finansowe. Okazało się, że rozmiar tych plików zwykle przekraczał 10 MB, natomiast podczas testów wykorzystywano małe pliki, o wielkości wynoszącej zaledwie kilka kB, co spowodowało, że problem nie został wcześniej wykryty.

Na etapie początkowych testów często udaje się wychwycić usterki. Jeżeli jednak nie zostaną wykryte, wcale nie oznacza to, że projekt będzie działał zgodnie z oczekiwaniami. Zanim oprogramowanie zostanie przekazane do środowiska produkcyjnego, konieczne jest przeprowadzenie *testów na poziomie użytkownika*. To pomoże nie tylko wykryć ewentualne usterki, ale również zdobyć większe zaufanie użytkowników.

Mamy różne fazy testowania oprogramowania, z których każda ma odmienne cele. Zadaniem fazy testów akceptacyjnych przeprowadzanych przez użytkowników, odbywającej się po wszystkich pozostałych etapach testowania, jest sprawdzenie, czy projekt działa zgodnie z przeznaczeniem. W niniejszej książce dokładnie omówię poszczególne etapy testowania oraz wyjaśnię ich cel.

Testowanie symulujące rzeczywiste scenariusze

W dniu 1 lipca 2012 roku rząd australijski wprowadził nowy system internetowy przeznaczony do obsługi imigrantów. Ponieważ ten system działał w trybie obsługi klientów według kolejności zgłoszenia, rankiem 1 lipca wszyscy zainteresowani ruszyli do komputerów. Wprawdzie byli w stanie uzyskać dostęp do systemu, ale w przypadku wielu osób próba wysłania zgłoszenia kończyła się niepowodzeniem. Wydajność działania aplikacji internetowej okazała się niewystarczająca w przypadku dużego obciążenia i witryna musiała zostać wyłączona. Przeprowadzenie testów z użyciem systemu oprogramowania mogło ujawnić tylko jedną bądź kilka usterek. *Testowanie nie może pokazać, czy oprogramowanie jest pozbawione od błędów*. Pod względem funkcjonalnym system działał bez zastrzeżeń. Jednak testowanie niefunkcjonalne nie zostało przeprowadzane poprawnie i dlatego nie sprawdzono działania systemu w warunkach dużego obciążenia. Szczytowe obciążenie okazało się za duże na możliwości tej witryny i doprowadziło do jej niedostępności przez wiele godzin.

Na etapie testowania sprawdza się również wydajność, aby zweryfikować działanie w sytuacji odzwierciedlającej rzeczywisty scenariusz, np. ilu jednocześnie użytkowników może się zalogować bądź przeprowadzać określoną operację. Dzięki temu jest możliwe wyszukiwanie problemów związanych z wydajnością działania mocno obciążonego systemu.

Na podstawie przedstawionych przykładów można stwierdzić, że testowanie to sposób na *ocenę jakości oprogramowania* i *zminimalizowanie ryzyka awarii oprogramowania* po jego przekazaniu do środowiska produkcyjnego.

Przedstawione scenariusze pokazują również, że jeśli oprogramowanie zostanie poprawnie przetestowane, wówczas można uniknąć awarii po wdrożeniu, choć rzadko tak się dzieje.

Jednym z powodów jest brak możliwość przetestowania wszystkiego. W przypadku większości systemów niemożliwe jest dokładne przetestowanie wszystkiego, to zaś nie pozwala sprawdzić wszystkich kombinacji danych wejściowych i warunków wstępnych. Posłużę się teraz prostym przykładem w celu wyjaśnienia dokładnego testowania. Załóżmy, że testowane oprogramowanie ma pole o wielkości 1, które akceptuje tylko jedną z 26 wielkich liter łacińskich. Przeanalizujmy teraz zastosowanie techniki dokładnego przetestowania oprogramowania, które zawiera takie pole. W omawianym przykładzie wymagane są poprawne testy sprawdzające użycie w tym polu wszystkich 26 wielkich liter alfabetu łacińskiego. Ponadto trzeba upewnić się, że zostaną odrzucone wszystkie niepoprawne znaki. Dlatego wymagane są testy dotyczące cyfr od 0 do 9, 26 małych liter alfabetu łacińskiego i 32 znaków specjalnych (łącznie ze spacją). To oznacza w sumie 94 przypadki testowe potrzebne do dokładnego przetestowania pola o wielkości jednego znaku. Pomyśl teraz o przetestowaniu w ten sposób oprogramowania mającego 10 pól danych wejściowych, z których każde może przyjąć 5 różnych wartości. Jeżeli chcesz przetestować wszystkie warianty poprawnych danych wejściowych, to byłoby to 10 do potęgi 5 (10^5), czyli $10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 = 100\,000$ przypadków testowych. Istnieje znikome prawdopodobieństwo przeprowadzenia takiej liczby testów oprogramowania.

Nawet w przypadku małych systemów istnieje bardzo wiele możliwych wariantów danych. Nie można z całą pewnością powiedzieć, czy istnieją jakiekolwiek usterki w oprogramowaniu, jeżeli nie zostaną wypróbowane wszystkie możliwe warianty danych wejściowych. Jednak niemożliwe jest przetestowanie wszystkich potencjalnych wariantów danych wejściowych i okoliczności.

Dlatego *określa się ryzyko i priorytety w celu skoncentrowania się na najważniejszych aspektach, które powinny być przetestowane*. W dalszej części książki dokładnie omówię te ryzyka i priorytety. Ich zastosowanie ma duże znaczenie dla zagwarantowania, że jako pierwsza zostanie przetestowana najważniejsza funkcjonalność systemu.

*Testowanie jest również zależne od kontekstu***.** To oznacza, że w poszczególnych okolicznościach wymagane są odmienne rodzaje testów. Na przykład witryna internetowa, w której będą jedynie wyświetlane informacje, będzie testowana w zupełnie inny sposób niż sklep internetowy, w którym można kupować produkty, podając dane karty kredytowej. System kontroli ruchu lotniczego musi zostać przetestowany znacznie bardziej rygorystycznie niż serwis społecznościowy. Czynnik ryzyka będzie miał ogromny wpływ na wybór typu niezbędnych testów. Im większe niebezpieczeństwo strat, tym więcej trzeba zainwestować w testowanie oprogramowania przed jego wdrożeniem.

Dopiero płacąc za błędy, uświadamiamy sobie ich cenę!

Sławomir Kuligowski

O jakości projektu programistycznego decyduje praca testera. Taka osoba musi łączyć solidną wiedzę ze znajomością cyklu życia danego projektu. Bezcenne jest również doświadczenie nabywane w ramach testowania różnego rodzaju oprogramowania. Myślisz o podjęciu pracy odpowiedzialnej, dobrze płatnej i bardzo satysfakcjonującej? Zostań testerem oprogramowania!

Ta książka pomoże Ci się przygotować do pracy na stanowisku testera oprogramowania, a także zapewni wsparcie podczas wykonywania nowych zadań. Znajdziesz tu dokładne omówienie podstaw testowania, jego różnych rodzajów i poziomów w odniesieniu do zastosowania na różnych etapach pracy nad projektem. Kolejne rozdziały, na podstawie rzeczywistego projektu programistycznego, pokazują znacznie szerszy kontekst pracy testera. Zapoznasz się ze szczegółowymi informacjami o poszczególnych fazach testowania, ich planowaniu, przeprowadzaniu i monitorowaniu. Dowiesz się również, w jaki sposób skorzystać z wniosków z testów podczas planowania rozwoju projektu.

W książce między innymi:

- czym jest testowanie, na czym polega i w jaki sposób należy je przygotować
- jakich umiejętności wymaga się od testera i jak wygląda struktura zespołu testującego
- dopasowanie strategii testowania do kontekstu projektu oprogramowania
- plan testowania: zakres, podejścia, wymagania
- dokumentacja tworzona podczas testowania, raporty i wnioski

Chhavi Raj Dosaj jest wybitnym testerem z ponad dwudziestoletnim doświadczeniem. Z jego wiedzy korzystały takie podmioty jak American Express, Lehman Brothers, Macquarie Securities, Deutsche Bank, Reserve Bank of Australia i wiele innych. Jest też rozchwytywanym trenerem, a także autorem książek i materiałów przygotowujących do egzaminów ISTQB.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-0971-7

9 788328 909717

Cena: 59,00 zł