

Spis treści

O autorze	XV
O recenzencie technicznym	XVII
Podziękowania	XIX
Wprowadzenie	XXI
Kto powinien przeczytać tę książkę?	XXII
Jak zorganizowana jest ta książka?	XXII
Część I: „Dlaczego” oraz „Co”	XXIII
Część II: „Jak”	XXIII
 CZĘŚĆ I. „DLACZEGO” ORAZ „CO”	1
1. Wartość automatyzacji testów	3
Dlaczego potrzebujemy automatyzacji testów?	3
Od modelu kaskadowego do zwinnego tworzenia oprogramowania	4
Koszt złożoności oprogramowania	5
Utrzymywanie stałego kosztu	6
Refaktoryzacja	8
Ciągłe doskonalenie	9
 2. Od testowania ręcznego do automatycznego	11
Podejście pierwsze: nagrywanie i odtwarzanie	11
Uzyskiwanie maksimum korzyści z automatyzacji testów	13
Różnice pomiędzy testami manualnymi i automatycznymi	16
Testowanie eksploracyjne	16
Rozważania dotyczące testowania automatycznego	18

3. Ludzie i narzędzia	27
Wybieranie właściwych narzędzi	27
Kto powinien pisać testy?	28
Promowanie testerów manualnych lub niedoświadczonych deweloperów do rangi deweloperów automatyzacji	28
Dzielenie pracy między testerów manualnych i deweloperów automatyzacji	31
Korzystanie z dedykowanego zespołu automatyzacji	33
Dedykowany deweloper automatyzacji wewnątrz każdego zespołu	34
Dawanie deweloperom pełnej odpowiedzialności za automatyzację	35
Różnorodność narzędzi	35
Klasyfikacja narzędzi	36
IDE i języki programowania	36
Biblioteki testowania (jednostkowego)	39
Biblioteki w stylu BDD	41
Technologie zapewniające interakcję z testowanym systemem	42
Pakiety do zarządzania testami	53
Narzędzia kompilacji oraz potoki ciągłej integracji lub ciągłego dostarczania	54
Inne czynniki mające znaczenie przy wybieraniu narzędzi	54
4. Osiąganie pełnego pokrycia	57
W jaki sposób mierzymy pokrycie?	58
Procent przypadków testów manualnych pokrytych przez automatyzację	58
Procent pokrytych funkcji	59
Procent pokrycia kodu	60
Uzyskiwanie korzyści przed osiągnięciem pełnego pokrycia	63
Co robimy po osiągnięciu pełnego pokrycia?	64
W jaki sposób uzyskać 100% pokrycia?	65
Odwracanie koła	66
Mapa drogowa prowadząca do pomyślnego projektu automatyzacji	69
Kiedy rozpocząć pracę nad progresją?	70
Nadawanie priorytetu pracy w celu zlikwidowania luki w regresji	71
5. Procesy biznesowe	73
Regularne uruchamianie testów	73
Najprostsze podejście	74
Testowanie nocne	74
Obsługiwanie błędów wykrywanych przez automatyzację	75
Zachowywanie testów kończących się niepowodzeniem	76
Wykluczanie testów kończących się niepowodzeniem	77
Tworzenie obejść w teście	78
Traktowanie wszystkich niepowodzeń automatyzacji jako błędów krytycznych	80
Ciągła integracja	81
Tworzenie oprogramowania sterowane testami akceptacyjnymi	81
Ciągłe dostarczanie i ciągłe wdrażanie	82
Wydania kanarkowe	84
Podsumowanie	85

6. Automatyzacja i architektura testów	87
Założenia dotyczące do architektury testów	87
Poznanie architektury testowanego systemu	88
Powrót do podstaw: czym jest system komputerowy?	88
Czym jest test automatyczny?	89
Rzeczywiste systemy komputerowe	90
Alternatywy i założenia w architekturze warstwowej	93
Związki między zakresem a testem	94
Omówienie warstw	95
Alternatywne zakresy testowania	97
Rzeczywista architektura	110
Architektura planowana kontra architektura rzeczywista	110
Typowe warianty	111
Łączenie testów	112
Podsumowanie czynników	115
Co poza architekturą warstwową?	118
Podsumowanie: dokonywanie własnych wyborów	122
7. Izolacja i środowiska testowe	123
Stan	123
Problemy z izolacją i ich rozwiązywanie	125
Problem 1 – testy manualne i test automatyczny wykonywane w różnym czasie	125
Problem 2 – testy manualne i automatyczne wykonywane jednocześnie	126
Problem 3 – kolejność ma znaczenie	126
Problem 4 – testy automatyczne uruchamiane jednocześnie	127
Techniki izolacji	127
Korzystanie z oddzielnych kont	128
Osobne bazy danych dla testów manualnych i automatyzacji testów	128
Oddzielne środowisko dla każdego członka zespołu	128
Resetowanie środowiska przed każdym cyklem testowania	130
Tworzenie niepowtarzalnych danych dla każdego testu	134
Každy test czyści wszystko, co utworzył	138
Współdzielone dane tylko do odczytu	139
Podsumowanie	141
8. Szersza perspektywa	143
Relacje między architekturą oprogramowania i strukturą biznesu	143
Prawo Conwaya	143
Zespoły pionowe kontra zespoły poziome	144
Zależności między architekturą oprogramowania i strukturą organizacyjną z automatyzacją testów	145
Dedykowany zespół automatyzacji	146
Deweloperzy automatyzacji w zespołach poziomych	146
Deweloperzy automatyzacji w zespołach pionowych	147
Elastyczna struktura organizacyjna	147
Ekspert ds. automatyzacji	148
Podsumowanie	148

CZĘŚĆ II. „JAK”	151
9. Przygotowanie do samouczka	153
Wymagania i założenia wstępne	153
Stosowanie procesu do istniejących systemów automatyzacji testów	154
Omówienie procesu	155
„Z dołu do góry” albo „z góry do dołu”	155
Proces	156
Poznanie testowanego systemu	157
Omówienie projektu MVCForum	157
Przygotowanie środowiska pod samouczek	160
Instalowanie Visual Studio w edycji Community	161
Pobieranie i instalowanie przeglądarki Chrome	161
Pobieranie i instalowanie bazy danych SQL Server Express	161
Pobieranie i budowanie aplikacji	162
Instalacja dodatku ReSharper (krok opcjonalny)	165
Korzystanie z narzędzia Git z poziomu Visual Studio	167
Przełączanie pomiędzy gałęziami	168
Podsumowanie	171
10. Projektowanie pierwszego przypadku testowego	173
Wybieranie pierwszego testu do zautomatyzowania	173
Wybieranie pierwszego przypadku testowego dla aplikacji MVCForum	177
Naukowa metoda projektowania przypadku testowego	178
Projektowanie kroków testu	178
Myślenie w kontekście obiektów i jednostek	181
Wzorzec obiektu strony	183
Podsumowanie	188
11. Kodowanie pierwszego testu	189
Tworzenie projektu	189
Modyfikowanie nazw klas, plików i metod testowych	192
Pisanie pseudokodu	194
Uwagi odnośnie do pseudokodu	195
Uzupełnianie kodu w celu jego skompilowania	197
Deklarowanie klasy LoggedInUser	199
Deklarowanie właściwości MVCForum	199
Deklarowanie metody RegisterNewUserAndLogin	201
Deklarowanie pozostałych klas i metod	201
Omówienie kodu modelu	205
Podsumowanie	206
12. Uzupełnianie pierwszego testu	207
Uruchamianie testu w celu znalezienia pierwszej metody do zaimplementowania	208
Dodawanie Selenium do projektu	209
Uruchamianie IISExpress	211
Implementowanie konstruktora MVCForumClient	211
Implementowanie metody RegisterNewUserAndLogin	213
Prośzenie dewelopera o dodanie unikalnego identyfikatora automatyzacji	217

Implementowanie metod ustawiających dla właściwości	219
Usuwanie duplikacji z metod ustawiających właściwości	222
Napotykanie błędu izolacji	224
Implementowanie metody CreateDiscussion i analizowanie niepowodzenia	228
Kończenie testu	231
Podsumowanie	232
13. Badanie niepowodzeń	233
Integrowanie z najnowszą wersją aplikacji MVCForum	233
Usprawnianie raportowania błędów	235
Unikanie debugowania	238
Badanie głównej przyczyny	239
Rozwiązywanie problemu	239
Więcej problemów...	241
Rejestrowanie oraz inne formy zbierania dowodów	247
Przechwytywanie ekranu	247
Rejestrowanie	247
Rejestrowanie zagnieżdżone	248
Rejestrowanie wizualne	249
Dodatkowe opcje rejestrowania i diagnozowania	250
Dodawanie zagnieżdżonego rejestratora wizualnego do testów aplikacji MVCForum	252
Badanie trudniejszych niepowodzeń	254
Niepowodzenia, które zdarzają się tylko na jednej maszynie	254
Badanie testów wpływających na inne testy	258
Badanie testów migoczących	258
Podsumowanie	265
14. Dodawanie kolejnych testów	267
Pisanie kolejnych testów	267
Planowanie kolejnych testów	269
Dodawanie testu: dyskusje mogą być filtrowane według kategorii	270
Podsumowanie procesu dodawania drugiego testu	283
Wprowadzanie dodatkowych usprawnień	283
Tworzenie bardziej zrozumiałych identyfikatorów	283
Organizowanie kodu w foldery	284
Wyodrębnianie klasy bazowej dla testów	284
Obsługa wielu użytkowników i przeglądarek	285
Wskazówki w zakresie korzystania z plików konfiguracyjnych testów	285
Obsługiwanie wielu przeglądarek	287
Dodatkowe możliwości usprawniania	289
Automatyczne ponowne tworzenie bazy danych	289
Oczyszczanie	290
Poprawienie wydajności	292
Dodawanie kolejnych testów	292
Testy sterowane danymi	292
Podsumowanie	294
15. Ciągła integracja	295
Czy to naprawdę konieczne?	296
Tworzenie procesu kompilacji testów	296

Planowanie procesu kompilacji testów	297
Tworzenie procesu automatycznego wdrażania	299
Dodawanie testów do kompilacji	301
Zmiana procesu tworzenia oprogramowania i kultury	303
Dążenie do „Świętego Graala”	304
Co jest potrzebne do zmiany kultury?	304
Określanie punktu wyjścia	306
Skracanie czasu wykonywania testów	313
Ulepszanie izolacji	314
Realizowanie wymagań wstępnych za pośrednictwem API	314
Równoległe wykonywanie i wirtualizacja	314
Uruchamianie wyłącznie testów poprawności w ramach ciągłej integracji	315
Dzielenie potoku CI na etapy	315
Pisanie głównie testów integracyjnych i jednostkowych	316
Uruchamianie testów wyłącznie dla konkretnych komponentów	316
Optymalizowanie wydajności testów	317
Pokrywanie szerszej macierzy	318
Podsumowanie	319
16. Tworzenie oprogramowania sterowane testami akceptacyjnymi (ATDD) . . .	321
Omówienie metodyki ATDD	321
Bycie bardziej zwinnym	322
Dług techniczny	322
Co sprawia, że zespół jest zwinny?	322
Unikanie długu technicznego	323
Proces	324
Tworzenie historyjki użytkownika	325
Pisanie testów automatycznych	328
Dostarczanie aplikacji i zbieranie opinii na jej temat	335
Używanie testów akceptacyjnych jako dokumentacji	336
Wiązanie kroków zamiast testów	336
Kompromis między możliwością ponownego użycia, poziomem szczegółów i czytelnością	338
Wprowadzanie metodyki ATDD do istniejącego projektu	339
Rozpoczynanie bez testów automatycznych	339
Retrospektywna implementacja automatyzacji	340
Rozpoczynanie od naprawy błędów	340
Zwiększanie pokrycia regresji	341
Podsumowanie	342
17. Test jednostkowe i tworzenie oprogramowania sterowane testami (TDD)	343
Przyswajanie testów jednostkowych i TDD	343
Sposoby pisania testów jednostkowych	344
Mechanizm biblioteki testów jednostkowych	345
Sposób pisania testu jednostkowego	347
Testy jednostkowe i operacje wejścia/wyjścia	350
Mechanizm działania TDD	355
Czerwone-zielone-refaktoryzacja	355
Dlaczego najpierw powinniśmy pisać testy?	358

Prawdziwe wyzwania w testowaniu jednostkowym i TDD	359
Główne wyzwania związane z testowaniem jednostkowym	359
Główne wyzwania związane z podejściem TDD	360
Bardziej szczegółowe wyzwania	360
Opanowywanie czystego kodu i zasad SOLID	364
Opanowywanie umiejętności refaktoryzowania	365
Największe wyzwanie: co testować?	366
Używanie metodyki TDD w celach, do jakich była projektowana	367
Podejście „z zewnątrz do środka” kontra podejście „od środka na zewnątrz”	367
Podsumowanie	368
18. Inne rodzaje testów automatycznych	369
Testy wydajności	369
Mierzenie wydajności w środowisku produkcyjnym	369
Czego nie robić?	371
Definiowanie oczekiwanego rezultatu	372
Ponowne wykorzystywanie kodu pomiędzy testami funkcjonalnymi i testami wydajności	373
Badanie wąskich gardeł w wydajności	373
Wydajność postrzegana a wydajność rzeczywista	373
Testy obciążeniowe	373
Jak działają testy obciążeniowe	374
Definiowanie oczekiwanego rezultatu	375
Łączenie testów wydajności z testami obciążeniowymi	377
Uruchamianie testów w środowisku produkcyjnym	377
Testowanie wdrożenia	377
Testowanie stanu zdrowia środowiska produkcyjnego	378
Które testy uruchamiać?	378
Oczyszczanie danych testu	379
Testowanie wizualne	379
Przepływ pracy testowania wizualnego	380
Testowanie wizualne i testowanie w wielu przeglądarkach/na wielu platformach	381
Testy instalacji	381
Podejścia dla testów instalacji	381
Testowanie instalacji za pośrednictwem interfejsu użytkownika lub instalacji dyskretnej	383
Testowanie programu deinstalacyjnego	383
Testy aktualizacji	383
Podejścia dla testów aktualizacji	384
Testowanie algorytmów statystycznych, niedeterministycznych i sztucznej inteligencji	385
Sposoby testowania algorytmów statystycznych	386
Testowanie aplikacji, które wykorzystują liczby losowe	388
Testowanie aplikacji analityki biznesowej	389
Podsumowanie	390
19. Co dalej?	391
Popelniaj błędy	391
Słuchaj, konsultuj się i zasięgaj porad	391
Poznaj i dostosuj się do celów swojego biznesu	392
Poznaj swoje narzędzia	392

Doskonalenie umiejętności programistycznych	393
Doskonalenie umiejętności w zakresie zapewniania jakości	394
Poszerzaj swoje horyzonty	394
Dzielenie się wiedzą	395
Dziel się własnymi narzędziami	395
Bawmy się dobrze!	396
Dodatek A. Rzeczywiste przykłady	397
Przykład 1 – system monitorowania wodomierzy	397
Symulowanie serwera komunikacji	399
Praca z usługą Google Maps	400
Przykład 2 – system do handlu na rynku Forex	402
Rozwiązanie	403
Niestabilność powodowana przez CRM	403
Izolowanie środowisk	404
Testowanie aplikacji mobilnej z użyciem abstrakcyjnego zakresu testowania	405
Przykład 3 – zarządzanie sklepem detalicznym	405
Opis architektury	405
Wdrożenie minimalne	407
Struktura organizacyjna	408
Rozwiązania automatyzacji testów	409
Symulator daty i godziny	410
Testy dla trzech warstw	411
Testy kompleksowe	411
Dodatek B. Mechanizm oczyszczania	413
Wywołania zwrotne i delegaty	413
Budowanie mechanizmu oczyszczania	415
Problem	415
Proste rozwiązanie	416
Ponowne wykorzystywanie mechanizmu oczyszczania	418
Obsługiwanie zależności pomiędzy akcjami oczyszczającymi	419
Obsługiwanie wyjątków w akcjach oczyszczających	421
Podsumowanie	422
Dodatek C. Projekt „Test Automation Essentials”	423
Kontekst	423
Struktura projektu	424
Uwaga dotycząca testów jednostkowych i komentarzy XML	425
Pakiety NuGet	425
Funkcje i narzędzia	425
TestAutomationEssentials.Common	425
TestAutomationEssentials.MSTest	428
TestAutomationEssentials.CodedUI	429
TestAutomationEssentials.Selenium	429
Pomoc w tworzeniu projektu i przenoszenie na inne języki	432
Dodatek D. Wskazówki i praktyki zwiększające produktywność programisty	433
Preferuj korzystanie z klawiatury	433
Poka-Yoke	435

Unikaj wartości Null	436
Unikaj przechwytywania wyjątków	437
Wybieranie najbardziej odpowiedniego lokalizatora	440
Trwale zakodowane ciągi znaków w automatyzacji testów: za i przeciw	443
Indeks	447

Arnon Axelrod jest ekspertem w dziedzinie automatyzacji testów i pracuje jako starszy konsultant, architekt, szkoleniowiec oraz lider zespołu automatyzacji w Sift Group. W wieku 10 lat Arnon zaczął programować swój komputer ZX Spectrum i od tamtej pory nigdy nie zarzucił pasji do programowania.

Po uzyskaniu w 1999 roku tytułu inżyniera z matematyki i informatyki na Uniwersytecie Ben-Gurion, Arnon rozpoczął pracę w korporacji Microsoft na stanowisku inżyniera trendów oprogramowania, gdzie po raz pierwszy zetknął się z zagadnieniem automatyzacji testów. Od tamtej pory przebiegł w kilku nowoczesnych firmach, głównie jako architekt oprogramowania, aż po dotychczas, w którym ponownie odkrył świat automatyzacji testów z zupełnie nowej perspektywy. W 2010 roku, po kilku latach korzystania z trendów Agile, Arnon – pracując wówczas w firmie Retata (zakupionej później przez NCR Corporation) – miał szcście sprawnie zaktualizować automatyzację testów z dotychczasowej techniki testów z napisanym oprogramowaniem sterowanym przez architektów (ATDD), jest niebawem do trykającego i efektywnego dostarczania wyników jakości oprogramowania. Podczas pracy w NCR Arnon zaprojektował infrastrukturę do automatyzacji testów, która była stosowana przez ponad 100 deweloperów i umożliwiała wykopanie ponad 400 błędów błędnych w mniej niż 20 minut.

W 2013 roku Arnon dołączył do Sift Group, gdzie pragnie oświecić, a jego misją jest popularyzowanie swojej wiedzy wśród jak największej liczby firm i osób indywidualnych, aby pomóc im efektywniej tworzyć wydajną jakość oprogramowania poprzez właściwe wykorzystywanie automatyzacji testów.