Gojko Adzic

SPECYFIKACJA NA PRZYKŁADACH

Poznaj zwinne metody pracy
i dostarczaj właściwe oprogramowanie

Skutecznie zbieraj wymagania!

Helion

Tytuł oryginału: Specification by Example: How Successful Teams Deliver the Right Software

Tłumaczenie: Arkadiusz Romanek

ISBN: 978-83-246-9118-0

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

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

Printed in Poland.

Spis treści

4 Inicjowanie zmian 63

CZĘŚĆ II Wzorce kluczowych procesów

5 Definiowanie zakresu na podstawie celów 93

6 Wspólne specyfikowanie 107

7 Wykorzystanie przykładów ilustrujących 127

8 Udoskonalanie specyfikacji 147

Wprowadzenie

Książka, którą trzymasz w dłoniach (lub którą widzisz na monitorze swojego komputera), jest efektem końcowym cyklu badań nad procesem, w czasie którego zajmujące się wytwarzaniem oprogramowania zespoły na całym świecie definiują, rozwijają i dostarczają *właściwe* produkty — wolne od błędów i opracowywane w krótkich cyklach projektowych. Ta książka zawiera wiedzę zgromadzoną dzięki analizie prawie pięćdziesięciu projektów inżynierii oprogramowania, podczas których zespoły pracowały nad dostarczeniem szerokiego zakresu produktów — począwszy od ogólnodostępnych stron internetowych aż po wewnętrzne systemy typu *back-office*. Zbierając materiały do tej książki, przeanalizowałem działania zespołów rozmaitych typów — były to zarówno małe grupy ludzi pracujących w tym samym biurze, jak i powiązane sieci programistów czasami rozlokowane nawet na różnych kontynentach, osoby posługujące się różnorakimi metodologiami procesów inżynierii: programowaniem ekstremalnym (ang. *Extreme Programming*), Scrumem, Kanbanem czy podobnymi metodami (często połączonymi ze sobą i przybierającymi formy metod zwinnych, tj. *agile* i *lean*). Wszystkie te zespoły łączyło jedno: ścisła współpraca w zakresie opracowywania poprawnych specyfikacji oraz testów, a także końcowy sukces i korzyści, jakie stały się ich udziałem dzięki tej współpracy.

Specyfikacja przez przykłady

Różne zespoły stosują różne terminy dla nazwania tego, co robią ze specyfikacjami i testami, a przecież w każdym przypadku ludzie ci odwołują się do tych samych kluczowych zasad i koncepcji, które moim zdaniem w gruncie rzeczy są zawsze takie same. Wśród używanych przez zespoły terminów i skrótów znajdujemy takie jak:

- Zwinne testowanie akceptacyjne.
- ATDD (ang. *Acceptance Test-Driven Development*), czyli wytwarzanie oprogramowania w oparciu o testy akceptacyjne.
- EDT (ang. *Example-Driven Development*), tj. wytwarzanie oprogramowania w oparciu o przykłady.
- Testowanie przez historyjki użytkownika (ang. *story testing*).
- BDD (ang. *Behavior-Driven Development*), czyli wytwarzanie oprogramowania w oparciu o analizę zachowań użytkowników.
- Specyfikacja przez przykłady (ang. *Specification by Example*).

Już sam fakt, że te same działania mają tak wiele różnych nazw, może nam wiele powiedzieć o ogromnej innowacyjności, z jaką mamy do czynienia w tej dziedzinie. Mnogość stosowanych terminów i definicji świadczy także o tym, że praktyki opisane w tej książce wpływają na zmianę podejścia zespołów do specyfikacji oraz do zagadnienia wytwarzania oprogramowania i jego testowania. Kierując się dążeniem do zachowania spójności przekazu, musiałem

jednak wybrać jeden termin. Zdecydowałem się na *specyfikację przez przykłady* i będę używał tego określenia w całej książce. Wyjaśnię mój wybór w części tego wprowadzenia zatytułowanej „Kilka słów na temat terminologii", gdzie znajdzie się także kilka szczegółów dotyczących reszty terminów zastosowanych dla nazwania elementów opisywanego procesu.

W świecie rzeczywistym…

Szczegóły proponowanej przeze mnie metodyki przedstawiam, odwołując się do licznych studiów przypadków i zarejestrowanych rozmów z członkami grup programistów. Wybrałem to podejście świadomie, z myślą o tym, żeby czytelnicy mieli szansę dowiedzieć się, jak radziły sobie prawdziwe zespoły, pracujące zgodnie z regułami tej metody i czerpiące z niej spore korzyści. Specyfikacja przez przykłady w żadnym razie nie jest ciemniejszą stroną sztuki, mimo że niektóre popularne media przekonują, że jest inaczej.

Prawie wszystko, o czym mówimy w tej książce, pochodzi z rzeczywistego świata inżynierii oprogramowania, dotyczy działających w tym świecie zespołów oraz ich — jak najbardziej realnych — doświadczeń. Tylko nieliczne praktyki zostały przedstawione w formie sugestii niepopartych studium przypadku. Są to zwykle pomysły, które — moim zdaniem — będą w przyszłości stanowić ważny przyczynek do dyskusji. I właśnie jako takie są przeze mnie przedstawiane.

Jestem przekonany, że moje wnioski, a także badania, jakie przeprowadziłem na potrzeby tej książki, nie zostaną uznane za poważne studia naukowe przez sceptyków, którzy twierdzą, iż zwinne praktyki się nie sprawdzają, a branża powinna wrócić do „prawdziwej inżynierii oprogramowania"[1]. Akceptuję to. Zasoby, do jakich miałem dostęp podczas zbierania materiałów na potrzeby tego projektu, można uznać za niewystarczające w porównaniu z tym, czego wymaga się od poważnych badań naukowych. I nawet gdybym dysponował zasobami spełniającymi kryteria uznanych metodologii, to przecież ani nie jestem naukowcem, ani nie zamierzam przedstawiać siebie jako naukowca. Jestem praktykiem.

Kto powinien zapoznać się z tą książką?

Jeśli tak jak ja jesteś praktykiem i zarabiasz na chleb, pracując przy wytwarzaniu oprogramowania lub wspierając pracę nad oprogramowaniem, ta książka ma Ci wiele do zaoferowania. Powstała wszak przede wszystkim z myślą o zespołach, które starały się wdrożyć w swoich środowiskach zwinne praktyki, ale napotkały problemy wpływające niekorzystnie na jakość produktu końcowego, zwiększające nakład pracy i prowadzące do tego, że dzieło rąk programistów nie spełniało oczekiwań klientów. (Tak… wówczas mamy do czynienia

[1] Jeśli chcesz poznać argumenty tych, którzy łudzą się, iż rygorystyczne podejście do inżynierii oprogramowania przynosi korzyści, tak jak gdyby była to jakaś drugorzędna gałąź fizyki, zajrzyj na stronę *http://www.semat.org*. Dobre kontrargumenty zawiera prezentacja Glenna Vanderburga zatytułowana *Software Engineering Doesn't Work!*, z którą można zapoznać się na stronie *http://confreaks.net/videos/282-lsrc2010-real-software-engineering*.

z prawdziwymi kłopotami i w takiej sytuacji zwykła iteracja nie na wiele się zda, ponieważ staje się „objazdem", a nie rozwiązaniem problemu). Specyfikacja przez przykłady, zwinne testowanie akceptacyjne, BDD i wszystkie alternatywne nazwy określają ten sam proces, zorientowany na rozwiązywanie tych problemów. Ta książka pomoże Ci poznać podstawy wspomnianych dobrych praktyk inżynierii oprogramowania. Dowiesz się z niej, jak możesz mieć większy pozytywny wpływ na zespół, niezależnie od tego, czy jesteś testerem, deweloperem, analitykiem, czy właścicielem produktu.

Jeszcze kilka lat temu większość ludzi, których spotykałem na konferencjach, nigdy wcześniej nie słyszała o opisanych w tej książce praktykach. Gros tych, z którymi rozmawiam dziś, ma świadomość istnienia wspomnianych praktyk, ale wielu z nich nie udało się wdrożyć ich prawidłowo. Istnieje niewielka baza literatury fachowej traktującej o problemach, jakie napotykają zespoły podczas szeroko pojętego wdrażania zwinnych praktyk, dlatego każdy zniechęcony zespół stwierdza, że jego przypadek musi być wyjątkowy, a sformalizowane koncepcje po prostu nie sprawdzają się w „jego świecie". Ludzie z takich zniechęconych zespołów dziwią się, jak po zaledwie pięciu minutach rozmowy udaje mi się trafnie zidentyfikować ich trzy lub cztery największe problemy. Często są bardzo zdziwieni, słysząc, że wiele innych zespołów napotyka na swojej drodze te same przeszkody.

Jeśli pracujesz w takim zespole, książka, którą właśnie trzymasz w dłoniach, uświadomi Ci przede wszystkim, że nie jesteś sam. Zespoły, z którymi rozmawiałem podczas zbierania materiałów do tej pozycji, wcale nie były zespołami idealnymi. Także one napotkały na swojej drodze mnóstwo problemów! Jednak zamiast rezygnować po zderzeniu z murem przeciwności, ludzie Ci postanowili go „objechać" lub zburzyć. Taka świadomość wspólnoty często pozwala zobaczyć swoje problemy w innym świetle. Mam nadzieję, że po przeczytaniu tej książki będziesz właśnie jedną z tych wyjątkowych osób, które potrafią spojrzeć na sytuację z innej perspektywy.

Jeśli akurat znajdujesz się w trakcie wdrażania specyfikacji przez przykłady, ta książka dostarczy Ci przydatnych wskazówek, dzięki którym dowiesz się, jak poradzić sobie z bieżącymi problemami i czego możesz spodziewać się w przyszłości. Mam nadzieję, że będziesz uczyć się na błędach innych ludzi i niektórych z nich uda Ci się uniknąć.

Ta książka została napisana także z myślą o doświadczonych praktykach, osobach, którym udało się wdrożenie zasad specyfikacji przez przykłady. Gdy zaczynałem rejestrowanie wywiadów z zespołami, spodziewałem się, że nie dowiem się niczego nowego. Myślałem, że „wiem, co jest grane", i szukałem tylko potwierdzenia. Tymczasem ostatecznie musiałem przyznać, że zaskoczyło mnie to, jak wiele różnych pomysłów można zastosować zależnie od kontekstu. Dowiadywałem się o rozwiązaniach, które wcześniej trudno mi było sobie wyobrazić! Wiele się przy tym nauczyłem. Dlatego mam nadzieję, że i Ty, drogi Czytelniku, wiele nauczysz się dzięki lekturze tej książki. Przedstawione w niej praktyki i pomysły powinny zainspirować Cię do wypróbowania alternatywnych rozwiązań problemów i dać Ci kilka podpowiedzi, jak usprawnić pracę zespołu, gdy tylko natkniesz się na historię przypominającą jakiś opisany tu przykład.

Co znajdziesz w środku?

W pierwszej części książki przedstawię podstawy koncepcji specyfikacji przez przykłady. Zamiast przekonywać Cię do tego, byś przestrzegał zasad opisanych w tej książce, pokażę Ci — w sposób charakterystyczny dla metodyki specyfikacji przez przykłady — konkretne korzyści, jakie zespoły pracujące nad wytwarzaniem oprogramowania mogą uzyskać dzięki zastosowaniu prezentowanych tu reguł. Jeśli zastanawiasz się nad zakupem tej książki, najpierw przejrzyj rozdział 1. i odpowiedz sobie na pytanie, czy którejś z opisanych tam korzyści można spodziewać się także podczas realizacji Twojego projektu. W rozdziale 2. przedstawię najważniejsze modele procesów i kluczowe artefakty specyfikacji przez przykłady. W rozdziale 3. omówię szczegółowo koncepcję żyjącej dokumentacji. W rozdziale 4. przedstawię typowe punkty wyjścia dla zainicjowania zmian procesu i kultury pracy zespołu oraz wskażę, na co należy zwrócić uwagę podczas implementacji metodyki.

Jednym z moich celów jako autora tej książki jest stworzenie spójnego słownika dla wzorców, idei i artefaktów, który będzie mógł być stosowany podczas wdrażania zasad specyfikacji przez przykłady. W branży inżynierii oprogramowania używa się tuzina różnych terminów dla nazwania samej praktyki i dwa razy więcej dla nazwania poszczególnych elementów kluczowych metodyki. Ludzie nazywają ten sam element: plikami właściwości, przypadkami testowymi, plikami BDD, testami akceptacyjnymi itp. Z tego powodu w rozdziale 2. przedstawię najlepsze (moim zdaniem) terminy stosowane dla określenia wszystkich kluczowych elementów omawianej metodyki. Nawet jeśli jesteś doświadczonym praktykiem, proponuję, żebyś zapoznał się z tym rozdziałem i odpowiedział sobie na pytanie, czy tak samo rozumiemy kluczowe terminy, wyrażenia i wzorce, o których mówimy w tej książce.

W drugiej części przedstawię najistotniejsze praktyki, z których zespoły opisane w studiach przypadków korzystały, by wdrożyć zasady specyfikacji przez przykłady. Zespoły działające w różnych środowiskach podejmowały różnorakie decyzje — czasem diametralnie różne, a czasami wręcz sprzeczne — aby jednak ostatecznie uzyskać takie same efekty. Opis tych praktyk uzupełnię dodatkowo charakterystyką kontekstów, w których działały zespoły podczas wdrażania reguł specyfikacji przez przykłady. Siedem rozdziałów drugiej części podzieliłem mniej więcej zgodnie z podziałem na obszary procesów.

W branży takiej jak inżynieria oprogramowania nie istnieją uniwersalne, idealne rozwiązania, ale na pewno można mówić o dobrych pomysłach, które da się zastosować ze świetnym skutkiem w różnych kontekstach. W drugiej części tej książki znajdziesz ikonki przedstawiające kciuk uniesiony w górę lub skierowany w dół. Takie grafiki będą pojawiać się przy nagłówkach z opisem działań, które badane zespoły uznały za użyteczne, lub problemów, z którymi często musiały się mierzyć. Potraktuj te wskazania jak propozycje wypróbowania przedstawionego przeze mnie rozwiązania lub ostrzeżenie przed sytuacją, jakiej warto unikać. Nie są to w żadnym razie gotowe recepty, które należy koniecznie zastosować. Ikony przedstawiające strzałki podkreślają najistotniejsze koncepcje związane z każdym z prezentowanych działań.

Wytwarzanie oprogramowania nie jest sztuką statyczną, ponieważ stale zmieniają się zarówno zespoły, jak i środowiska, w których pracują ludzie. Procesy rozwoju produktu muszą nadążać za tymi zmianami. W części trzeciej tej książki przedstawię studia przypad-

ków opisujące „podróże" kilku wybranych zespołów. Omówię procesy, ograniczenia i konteksty, analizując ewolucję projektów. Przedstawione w tej części historie pozwolą Ci lepiej przygotować się do czekających Cię zmian lub zachęcą do wykonania następnego kroku, pomogą w poszukiwaniach inspiracji, a także zainspirują do odkrywania nowych metod działań i rozwiązań.

W ostatnim rozdziale podsumuję najważniejsze wnioski, do których doszedłem dzięki analizie licznych studiów przypadków zebranych na potrzeby tej książki.

Coś więcej niż podstawy

Gdyby odwołać się do tradycyjnego modelu doskonalenia *Shu-ha-ri*[2], ta książka znajduje się na poziomie *ha*. Na tym etapie uczeń łamie obowiązujące zasady i dowiaduje się, że istnieje wiele różnych sprawdzających się modeli. Mój model i moje doświadczenie przedstawiłem w *Bridging the Communication Gap*. Natomiast w książce, którą właśnie trzymasz w dłoniach, bardzo staram się zachować obiektywne spojrzenie i uniknąć ulegania wpływom moich dotychczasowych osobistych doświadczeń. O projektach, nad którymi pracowałem osobiście, mówię tylko wtedy, gdy naprawdę zależy mi na przekazaniu ważnych informacji, a żaden z ankietowanych zespołów nigdy nie zetknął się z podobną sytuacją. W tym sensie specyfikacja przez przykłady zaczyna się tam, gdzie skończyłem *Bridging the Communication Gap*.

Podstawowe zasady opisuję krótko w rozdziale 2. Nawet jeśli nigdy wcześniej nie słyszałeś o żadnej z przedstawionych tam koncepcji, lektura tego rozdziału powinna zapewnić Ci dość danych, żebyś mógł zrozumieć resztę książki. Staram się przy tym nie zagłębiać za bardzo w podstawy. Szczegółową analizę fundamentów teorii specyfikacji przez przykłady znajdziesz w *Bridging the Communication Gap* i nie jest moim celem kopiowanie tych informacji.

Jeśli chcesz poszerzyć swoją wiedzę w zakresie podstaw, zajrzyj na stronę *http://specificationbyexample.com*, zarejestruj posiadany egzemplarz tej książki, a otrzymasz za darmo plik PDF z *Bridging the Communication Gap*.

Nie sądzę, że kiedykolwiek powstanie kontynuacja tej pozycji — to znaczy książka mojego autorstwa na poziomie *ri* — ponieważ moim zdaniem poziom ten jest w ogóle nieosiągalny dla książek. Z drugiej strony wierzę, że ta książka pomoże Ci przejść na wyższy poziom. Gdy zaczniesz rozumieć, że wybór konkretnego narzędzia nie ma znaczenia, będzie to znaczyło, że znalazłeś się na poziomie *ri*.

[2] *Shu-ha-ri* to model doskonalenia związany z nauką aikido. Shu-ha-ri znaczy mniej więcej tyle co: *podporządkuj się – zerwij – odejdź*. Na pierwszym poziomie (*shu — podporządkuj się*) uczeń zdobywa wiedzę, pilnie naśladując przedstawiony mu model. Na drugim poziomie (*ha — zerwij*) uczeń dowiaduje się, że istnieją inne modele i rozwiązania. Na trzecim poziomie (*ri — odejdź*) uczeń przekracza granicę, wychodząc poza działania polegające na bezrefleksyjnym powtarzaniu znanych rozwiązań.

Ta książka nie zawiera kodu źródłowego i nie wyjaśnia działania żadnych narzędzi

Nie znajdziesz w tej książce kodu źródłowego ani instrukcji opisujących, jak użyć konkretnego narzędzia. Czuję się zobligowany do uprzedzenia o tym już teraz, ponieważ nawet w trakcie trwania procesu wydawniczego byłem zmuszony do składania wyjaśnień w tej kwestii (zwykle chodziło o odpowiedź na pytanie: „Co właściwie masz na myśli? Książka o programowaniu bez kodu źródłowego? Jak to możliwe?").

Zasady i praktyki specyfikacji przez przykłady wpływają przede wszystkim na sposób komunikacji międzyludzkiej w zespołach pracujących nad dostarczaniem oprogramowania, a także na współpracę zespołów z użytkownikami biznesowymi i interesariuszami (ang. *stakeholders*). Jestem pewien, że wielu sprzedawców narzędzi informatycznych będzie próbować sprzedać Ci gotowe rozwiązania. Z kolei wielu menedżerów chętnie nawet zapłaciłoby za to, aby ktoś sprawił, żeby ich problem po prostu zniknął. Niestety mamy do czynienia przede wszystkim z problemem ludzkim, a nie technicznym.

Bill Gates mawiał: „Pierwszą zasadą dotyczącą wszelkich technologii wykorzystywanych w działalności gospodarczej jest to, że zautomatyzowanie skutecznego działania potęguje efektywność. Druga zasada mówi, że automatyzacja zastosowana do nieefektywnego działania zwiększa poziom nieskuteczności". Wiele zespołów, którym nie udało się poprawne wdrożenie specyfikacji przez przykłady, powiększyło swoją nieefektywność poprzez automatyzację dotychczasowych (niepoprawnych) praktyk. Zamiast więc skupiać się na konkretnym narzędziu, chcę zaproponować rozwiązanie problemów, które są prawdziwą przyczyną niepowodzeń. Jeśli tylko uda Ci się zapanować nad komunikacją i ustalić zasady współpracy, przyjdzie czas na wybranie odpowiednio dopasowanego narzędzia. Jeśli po przeczytaniu tej książki będziesz chciał poszerzyć swoją wiedzę na temat narzędzi, które wspierają wdrożenie specyfikacji przez przykłady, odwiedź stronę *http://specificationbyexample.com* i przejrzyj znajdujące się tam zasoby.

Kilka słów na temat terminologii

Jeśli jest to Twój pierwszy kontakt ze specyfikacją przez przykłady, ATDD, ze zwinnymi testami akceptacyjnymi, BDD czy z którymkolwiek z terminów, których ludzie używają na określenie zbioru praktyk będącego podstawowym tematem tej książki, oznacza to, że udało Ci się uniknąć wielu lat zamieszania spowodowanego powielaniem i kreowaniem mylących nazw. Powinieneś się z tego ucieszyć i możesz pominąć tę część wprowadzenia. Jeśli jednak zetknąłeś się kiedyś z którymkolwiek z wymienionych terminów, określenia używane przeze mnie w tej książce mogą Cię zaskoczyć. Za chwilę wyjaśnię, dlaczego wybrałem te, a nie inne terminy, i dlaczego uważam, że Ty także powinieneś zacząć z nich korzystać.

Podczas prac nad tą książką borykałem się z problemami, które nie są obce wszystkim praktykom piszącym na temat zautomatyzowanych specyfikacji. Jeśli terminologia ma mieć sens, musi być spójna. Często rozumiemy to dopiero wtedy, gdy sami zabieramy się do opisania jakiegoś zagadnienia. Ponieważ moja książka jest produktem bazującym na cyklu wywiadów, a wiele osób, z którymi rozmawiałem, używało różnych terminów do nazwania tej samej rzeczy, uzyskanie spójności terminologicznej okazało się zadaniem niezwykle trudnym.

Zdałem sobie sprawę, że specjaliści realizujący reguły specyfikacji przez przykłady w praktyce — na czele ze mną — sami są sobie winni, bo to przecież oni używają zdradliwych terminów technicznych, wprowadzających w błąd nie tylko ich samych, ale i wszystkich tych, którzy próbowali wdrożyć owe praktyki. Wtedy zdecydowałem, że jednym z celów tej książki będzie opracowanie spójnej terminologii, którą będą mogli posługiwać się wszyscy członkowie społeczności wytwórców oprogramowania. Jeśli zależy nam na większym zaangażowaniu w prace projektowe użytkowników biznesowych — co jest przecież jednym z naszych głównych celów — musimy zacząć używać właściwych terminów do nazwania właściwych rzeczy. Czas przestać wprowadzać zamieszanie!

Doskonale rozumiemy potrzebę zachowania spójności, gdy tworzymy specyfikacje. Wiemy, że musimy zachować powtarzalność terminów i unikać wprowadzania pojęć niejednoznacznych. Tyle że jakoś zapominamy o tym, gdy zaczynamy mówić o procesach. Na przykład gdy mówimy o *ciągłej integracji* w kontekście specyfikacji przez przykłady, tak naprawdę wcale nie chodzi nam o wykonywanie testów integracyjnych. Dlaczego więc mielibyśmy używać tego terminu, a następnie wyjaśniać, czym różnią się testy akceptacyjne od testów integracyjnych? Zanim sam nie zacząłem używać terminu *warsztat specyfikacji* do nazwania spotkania, w którym uczestniczą osoby odpowiedzialne za wspólne opracowanie testów akceptacyjnych, miałem spore trudności, aby przekonać użytkowników biznesowych do udziału w takich zgromadzeniach. Zwykła zmiana nazwy sprawiła, że problem przestał istnieć. Dzięki użyciu lepszego glosariusza terminologicznego można uniknąć zupełnie bezsensownych dysput, sprawiając, że wszyscy dobrze się rozumieją.

Dlaczego specyfikacja przez przykłady?

W pierwszej kolejności chciałbym wyjaśnić, dlaczego wybrałem termin „specyfikacja przez przykłady" dla nazwania zbioru praktyk. Dlaczego nie zwinne testowanie akceptacyjne, BDD czy ATDD?

Podczas konferencji „Domain Driven Design eXchange", która odbyła się w 2010 roku w Londynie[3], Eric Evans stwierdził, że termin *zwinny* (ang. *agile*) stracił już wszelkie znaczenie, bo obecnie wszystko można nazwać zwinnym. Niestety Eric ma rację. Widziałem o wiele za dużo zespołów, próbujących wdrożyć proces, który nie miał szans zakończenia się powodzeniem, choć wszystkim wydawało się, że etykietka *agile* w jakiś magiczny sposób sprawi, że okaże się skuteczniejszy, niż był w rzeczywistości. Przecież mamy do swojej dyspozycji ogromną bibliotekę dostępnego piśmiennictwa omawiającego prawidłowe wdrażanie zasad programowania ekstremalnego, Scruma czy innych mniej popularnych procesów z bogatego wachlarza praktyk zwinnych.

[3] *http://skillsmatter.com/event/design-architecture/ddd-exchange-2010*.

Aby odejść od tych wprowadzających zamieszanie niejasności oraz dyskusji na temat tego, czy zwinne praktyki się sprawdzają, czy nie (i wyjaśnić, co to właściwie znaczy *być agile*), w tej książce staram się ze wszystkich sił unikać terminu *zwinny*. Używam go tylko wtedy, gdy nawiązuję do zespołów, które rozpoczęły realizację dobrze zdefiniowanych procesów zbudowanych na zasadach określonych w Manifeście Agile. Tak więc w sytuacji, gdy nie mogę umieścić wtrętu ze słowem „agile" w co drugim moim zdaniu, użycie terminu „zwinne testowanie akceptacyjne" jest po prostu nie do przyjęcia.

Opisane w tej książce praktyki nie stanowią kompletnej metodyki tworzenia oprogramowania. Są za to uzupełnieniem innych metod — zarówno tych bazujących na iteracji, jak i na przepływach — zapewniając dyscyplinę specyfikacji i testów, poprawiając komunikację między interesariuszami i członkami zespołów deweloperskich, ograniczając liczbę zbędnych przeróbek i ułatwiając wprowadzanie zmian. Nie chcę zatem używać żadnego terminu zawierającego element „Driven Development" (a zwłaszcza *Behavior-Driven Development* — BDD). Nie chodzi o to, że mam coś przeciwko BDD. Wręcz przeciwnie! Kocham BDD i uważam, że większość z tego, co znajdziesz w tej książce, w rzeczywistości stanowi kluczowy element BDD. Ale BDD również jest ofiarą problemu z nazewnictwem.

Znaczenie terminu BDD podlega ciągłym zmianom. Dan North, największy autorytet w kwestii metodyki BDD oraz człowiek, który jako jeden z niewielu ma prawo decydować, czym jest i czym nie jest BDD, stwierdził podczas konferencji „Agile Specifications, BDD and Testing Exchange" w 2009 roku, że BDD jest *metodyką*[4]. (A dokładniej powiedział, że BDD jest: „zwinną metodyką drugiej generacji, typu *outside-in*, *pull-based*, *multiple-stakeholder*, *multiple-scale*, *high-automation*"). Aby uniknąć nieporozumień i niejasności w kwestii różnicy pomiędzy definicją BDD zaproponowaną przez Dana Northa a tym, co ja sam uważam za BDD, rezygnuję ze stosowania także tego terminu. Ta książka mówi o precyzyjnie zdefiniowanym zestawie praktyk, które można stosować w ramach różnych metod, w tym również metodyki takiej jak BDD (jeśli przyjąć, że BDD naprawdę jest metodyką).

Chciałbym również uniknąć nadużywania terminu *test*. Wielu menedżerów i użytkowników biznesowych niestety uważa testy za aktywność uzupełniającą o czysto technicznej naturze, a nie coś, w czym chcieliby brać udział. Przecież mają od tego wyspecjalizowanych testerów! Specyfikacja przez przykłady wymaga aktywnego uczestnictwa interesariuszy i członków zespołu dostarczającego oprogramowanie, włącznie z deweloperami, testerami i analitykami. Jeśli nie możemy nadużywać terminu *test*, trudno mówić o testowaniu historyjek, zwinnym testowaniu akceptacyjnym i posługiwać się innymi, podobnymi określeniami.

W ten oto sposób doszliśmy do terminu specyfikacja przez przykłady, który staje się najsensowniejszym wyborem, ponieważ niesie najmniejszy bagaż negatywnych konotacji.

[4] *http://skillsmatter.com/podcast/java-jee/how-to-sell-bdd-to-the-business*.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

Ś

T

U

W

X

Z

Ź

Ż

SPECYFIKACJA NA PRZYKŁADACH

Poznaj zwinne metody pracy i dostarczaj właściwe oprogramowanie

Dokładne poznanie wymagań klienta to klucz do w pełni wydajnej aplikacji. Jest niezbędne, by sprostać oczekiwaniom jej przyszłych użytkowników. Metoda SBE (ang. *specification by example*) zachęca do zwinnego (*Agile*) podejścia do tego tematu, dzięki czemu zebranie wymagań będzie przebiegało zdecydowanie sprawniej.

Ta książka rozwieje wszystkie Twoje wątpliwości! Poznasz kluczowe wzorce procesu oraz nauczysz się wprowadzać w nich zmiany. Podejście SBE wymaga zmiany kultury pracy zespołu. Nie jest to zadanie łatwe, dlatego znajdziesz tu najlepsze praktyki stosowane w tej sytuacji. Ostatnie rozdziały książki zostały poświęcone omówieniu przykładów z życia wziętych, a dotyczących najczęściej spotykanych problemów. To szczególnie cenne informacje, które pozwolą Ci wybrać najlepsze sposoby uniknięcia typowych błędów. Książka ta jest obowiązkową lekturą dla wszystkich twórców oprogramowania!

Dzięki tej książce:

- poznasz zalety SBE
- dowiesz się, dlaczego wspólne specyfikowanie jest tak istotne
- nauczysz się definiować cel z uwzględnieniem wzorców
- zmienisz kulturę pracy Twojego zespołu
- skutecznie wprowadzisz SBE w Twojej organizacji

Poznaj zalety SBE!

ISBN: 978-83-246-9118-0
9 788324 691180
Cena: 59,00 zł