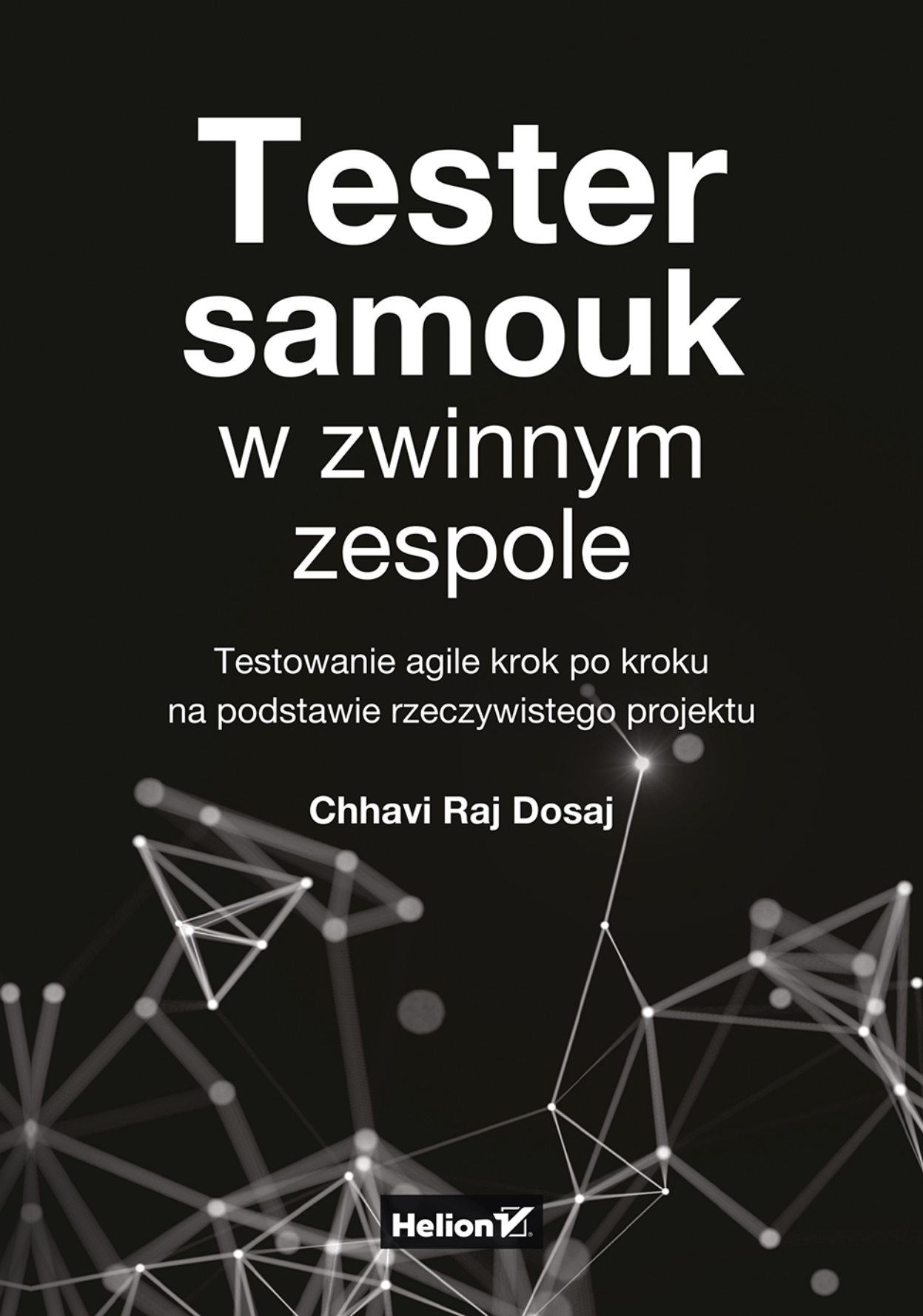
Tester samouk w zwinnym zespole

Testowanie agile krok po kroku
na podstawie rzeczywistego projektu

Chhavi Raj Dosaj



Helion

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

Tłumaczenie: Robert Górczyński

ISBN: 978-83-289-0969-4

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/tesazw
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

Dlaczego potrzebujemy modelu zwinnego podczas tworzenia oprogramowania?

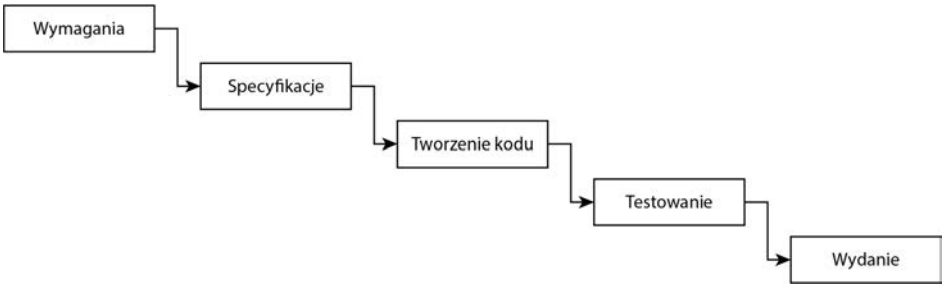
Od 1970 roku oprogramowanie było tworzone z wykorzystaniem tradycyjnego modelu kaskadowego. To oznacza zastosowanie liniowego, sekwencyjnego podejścia projektowego, w którym postęp odbywa się w jednym kierunku, do dołu, jak w przypadku wodospadu. Skąd więc wzięła się potrzeba zastosowania innego podejścia, takiego jak zwinne, podczas tworzenia oprogramowania?

Rozważ przedstawiony tutaj przykład porównujący model kaskadowy tworzenia oprogramowania z podejściem wykorzystującym metody zwinne.

Bank planuje opracowanie nowej aplikacji mobilnej dla klientów, która pozwoli im wykorzystać następujący zestaw funkcjonalności:

- **Funkcja A** — klient może zmienić PIN karty płatniczej.
- **Funkcja B** — klient może zastrzec kartę i poprosić o wydanie nowej.
- **Funkcja C** — klient może sprawdzić historię transakcji przeprowadzonych dla danego rachunku.
- **Funkcja D** — klient może zmienić wysokość dziennych limitów dla operacji przeprowadzanych na rachunku.
- **Funkcja E** — klient może zablokować i odblokować zgubioną kartę.

Bank rozpoczął pracę nad nowym projektem mającym na celu dostarczenie klientom tej nowej aplikacji mobilnej i zastanawia się, które podejście powinno być zastosowane podczas pracy: model kaskadowy czy model programowania zwinnego. Jeżeli bank zdecyduje się na model kaskadowy, wówczas projekt będzie realizowany za pomocą etapów przedstawionych na rysunku 1.1.



Rysunek 1.1. Etapy modelu kaskadowego podczas tworzenia oprogramowania

W modelu kaskadowym projekt powstaje jako seria etapów, jak pokazałem to na rysunku 1.1. Poszczególne etapy są wykonywane po kolei. Jeżeli dany etap nie zostanie ukończony i zatwierdzony, przejście do następnego będzie niemożliwe.

W takim modelu testowanie odbywa się dopiero po zakończeniu wszystkich działań związanych z tworzeniem oprogramowania.

Zapoznaj się teraz z wysokiego poziomu działaniami, które będą podejmowane podczas tworzenia aplikacji mobilnej banku.

Etap wymagań

Na tym etapie analityk biznesowy bądź analityk systemu będzie zbierał informacje dotyczące poszczególnych funkcjonalności (**A+B+C+D+E**) od klientów bądź ich reprezentantów (zespół biznesowy). Wymagania te zostaną zawarte w dokumencie wymagań użytkownika, który zdefiniuje zakres projektu. Jeżeli klient zażąda jakichkolwiek zmian dotyczących uzgodnionego zakresu projektu już po rozpoczęciu pracy nad projektem, konieczne będzie przejście przez długi proces wprowadzenia zmian, który może mieć wpływ na pracę nad całym projektem oraz na czas dostarczenia produktu.

Etap specyfikacji/analizy

Na tym etapie, bazując na wymaganiach użytkownika, analityk biznesowy opracuje dokumenty określające specyfikację projektu, np. dokument wymagań biznesowych (ang. *business requirement document*, BRD), przedstawiający wszystkie funkcjonalne i niefunkcjonalne wymagania dla wszystkich funkcji produktu. Architekt lub projektant rozwiązania zdefiniuje projekt techniczny bądź architekturę dla rozwiązania, co będzie obejmowało wszystkie funkcje wymienione w dokumencie specyfikacji. Bazując na dokumentach specyfikacji i projektowych, zespół testujący może rozpocząć przygotowania do utworzenia przypadków testowych wysokiego poziomu. Tymczasem zespół programistyczny może rozpocząć przygotowania do utworzenia dokumentów specyfikacji komponentu, opierając się na dokumentach specyfikacji projektu, które będą stanowić punkt wyjścia podczas podejmowania wszelkich działań programistycznych.

Etap tworzenia kodu

Na tym etapie programiści zajmują się tworzeniem poszczególnych komponentów aplikacji mobilnej, bazując na specyfikacji komponentów, aby w ten sposób zapewnić spełnienie wymagań zarówno funkcjonalnych, jak i niefunkcjonalnych. Po zakończeniu etapu tworzenia kodu źródłowego programiści przeprowadzają testy jednostkowe i testowanie integracji dla tych komponentów aplikacji mobilnej. Tymczasem zespół testujący zajmie się przygotowaniem dokładnych przypadków testowych niskiego poziomu, bazując na opracowanych wcześniej scenariuszach wysokiego poziomu i powiąże je z wymaganiami projektu, aby zapewnić wystarczające pokrycie testami. Ponadto zespół ten upewni się, czy przygotowane są środowisko testowe i dane testowe.

Etap testowania

Na tym etapie zespół programistyczny będzie zajmował się wdrożeniem pełnej aplikacji mobilnej w środowisku testowym, zespół testujący zaś rozpocznie wykonywanie przypadków testowych, co ma na celu przetestowanie całej funkcjonalności. Odbędzie się testowanie systemu i integracji systemu oraz przeprowadzone zostaną testy akceptacyjne użytkowników, aby mieć pewność, że cała funkcjonalność działa zgodnie z oczekiwaniami. Po zakończeniu etapu testowania aplikację mobilną będzie można umieścić w środowisku produkcyjnym, aby stała się dostępna dla użytkowników końcowych.

Teraz w zależności od wielkości zespołu przeprowadzenie każdego etapu prac będzie wymagało określonego czasu:

- działania związane z określaniem wymagań — 1 miesiąc;
- działania związane z analizą i przygotowaniem specyfikacji — 1 miesiąc;
- działania związane z tworzeniem kodu źródłowego — 2 miesiące;
- działania związane z testowaniem i wydaniem produktu — 2 miesiące.

Tak więc po upływie mniej więcej pół roku zespół będzie w stanie dostarczyć aplikację mobilną użytkownikom końcowym. Jeżeli w trakcie prowadzenia prac nad projektem klient będzie chciał wprowadzić jakiekolwiek zmiany, to może opóźnić dostarczenie produktu.

Warto teraz przeanalizować **wady** stosowania modelu kaskadowego podczas realizacji projektu takiego jak utworzenie aplikacji mobilnej.

Wprowadzanie zmian jest trudne

W przypadku projektu programistycznego zespół biznesowy bądź grupa potencjalnych użytkowników działającą wersję aplikacji po raz pierwszy może zobaczyć na etapie testów akceptacyjnych użytkowników, czyli mniej więcej pięć miesięcy po rozpoczęciu pracy nad projektem.

Istnieje duże prawdopodobieństwo, że w tym czasie pewne wymagania mogą ulec zmianie z powodów wewnętrznych lub zewnętrznych. Co się stanie, jeśli użytkownik nie będzie usatysfakcjonowany, kiedy zobaczy działającą aplikację podczas testów akceptacyjnych użytkowników i zasugeruje zmiany? Wprowadzenie zmian na tak późnym etapie pracy będzie wymagało ogromnego wysiłku związanego z przeredagowaniem specyfikacji, utworzeniem nowego kodu źródłowego i przeprowadzeniem testowania, co z kolei o kilka miesięcy opóźni dostarczenie produktu użytkownikowi końcowemu.

Dlatego też ta metodologia praktycznie nie pozostawia żadnego pola manewru w przypadku nieoczekiwanych zmian, pojawiających się podczas wydawania projektu w jego pierwotnej postaci.

Opóźniony lub nieprzewidywalny termin dostarczenia produktu

Jeśli bank skorzysta z modelu kaskadowego, będzie musiał poczekać przynajmniej pół roku na udostępnienie aplikacji klientom końcowym. Załóżmy, że bank chce jak najszybciej otrzymać prostą wersję aplikacji z minimalną funkcjonalnością, aby zadowolić klientów i nie dać się wyprzedzić konkurencji. Niestety w przypadku podejścia programowania sekwencyjnego nie jest to możliwe, ponieważ nie da się dostarczać funkcjonalności etapami. Ponadto jeśli którykolwiek z etapów zostanie opóźniony bądź też pojawią się jakiekolwiek poważne problemy podczas testowania, wówczas data ukończenia aplikacji stanie się nieprzewidywalna.

Odroczenie testowania aż do ukończenia etapu tworzenia aplikacji

Skoro w modelu kaskadowym zespół testujący wstrzymuje się z rozpoczęciem testowania dynamicznego aż do chwili ukończenia całego produktu, etap testowania w tym podejściu rozpocznie się bardzo późno. W przypadku naszej przykładowej aplikacji mobilnej nastąpi to niemalże cztery miesiące po rozpoczęciu pracy nad projektem. To powoduje zwiększenie ryzyka technicznego — jeżeli podczas testów zostanie znaleziony jakikolwiek poważny błąd, wymagający wprowadzenia większych zmian w kodzie źródłowym, skutkiem może być opóźnienie daty dostarczenia gotowego produktu.

Brak zaangażowania ze strony klienta

Jak można się przekonać, poza etapem zbierania wymagań klient lub jego przedstawiciele nie są zaangażowani w pracę na różnych etapach projektu, a tym samym nie mają możliwości, by dostarczać uwagi na jego temat. W zmieniającym się obecnie środowisku rynku opracowanie produktu bez aktywnego udziału klienta w tym procesie będzie zadaniem trudnym i ambitnym.

Brak pola dla innowacji

Model kaskadowy nie jest wystarczająco elastyczny, aby pozwalał na wprowadzanie zmian w projekcie. Z tego powodu implementowanie nowych idei w produkcie okazuje się trudne.

Jednak istnieją pewne zalety wynikające z używania modelu kaskadowego.

Prosta i przejrzysta struktura

Struktura jest intuicyjna i prosta do przyswojenia. Wszystkie etapy są wykonywane w kolejności, a wynik każdego z nich to określone produkty robocze. Tego rodzaju metodologiczne podejście zapewnia nowym członkom zespołu prostą strukturę, którą łatwo jest stosować i zrozumieć.

Wyraźny przepływ informacji

W modelu sekwencyjnego tworzenia oprogramowania duży nacisk kładzie się na dokumentację. To oznacza opracowywanie obszernych dokumentów wspierających wykonywanie poszczególnych zadań w projekcie. Przekazywanie informacji między różnymi etapami pracy i pomiędzy zespołami okazuje się łatwe, zwłaszcza jeśli zespoły nie znajdują się w tym samym położeniu. Jednak z drugiej strony ten nacisk na obszerną dokumentację oznacza większe obciążenie dla zespołu pracującego nad projektem.

Jak widać, wady modelu kaskadowego przewyższają jego zalety. Dlatego też programiści zaczęli szukać alternatyw. Zobacz teraz, jak wspomniany projekt oprogramowania mógłby zostać opracowany z wykorzystaniem **zwinnego modelu tworzenia oprogramowania** (ang. *agile software development*), czyli modelu, w którym używa się podejścia iteracyjnego i przyrostowego. Jednak wcześniej musisz zrozumieć stosowane w nim reguły.

W **programowaniu iteracyjnym** praca nad projektem odbywa się za pomocą serii kolejnych kroków udoskonalania produktu. Zespół programistyczny rozpoczyna od opracowania pierwszej wersji systemu, która może być niekompletna bądź niedopracowana w pewnych obszarach. Następnie zespół iteracyjnie udoskonala te obszary do chwili, gdy produkt będzie można uznać za satysfakcjonujący. Z każdą iteracją produkt jest usprawniany dzięki dopracowywaniu w nim kolejnych szczegółów.

Na przykład w trakcie pierwszej iteracji zespół przygotuje funkcjonalność transakcji zapewniającą obsługę jedynie dziesięciu ostatnich transakcji. W trakcie następnej iteracji zespół dopracuje tę funkcjonalność, aby produkt oferował możliwość wyświetlania wielu stron wcześniejszych transakcji. Wreszcie w trakcie trzeciej iteracji pojawi się funkcjonalność filtrowania transakcji.

Podczas **programowania przyrostowego** oprogramowanie jest tworzone i dostarczane we fragmentach. Każdy fragment przedstawia pełny podzbiór funkcjonalności. Ten przyrost może być mały bądź ogromny, począwszy od np. opracowania ekranu logowania do

systemu (mały przyrost), a skończywszy na opracowaniu wysoce elastycznej funkcjonalności zarządzania danymi (ogromny przyrost).

Na przykład zespół będzie dokładnie analizował pełny kod źródłowy i testował funkcjonalność historii transakcji. Ta funkcjonalność zostanie udostępniona klientowi, kolejne zaś będą dostarczane stopniowo.

Zwinny model tworzenia oprogramowania jest zarówno **iteracyjny, jak i przyrostowy**. Iteracyjny — ponieważ rozwiązanie opracowane w trakcie jednej iteracji może zostać usprawnione podczas kolejnych. Przyrostowy — ponieważ w trakcie pracy nad projektem są dostarczane produkty powstałe w poszczególnych iteracjach. W zwinnym modelu programowania, podobnie jak w przypadku modelu sekwencyjnego, są stosowane te same etapy analizy wymagań, tworzenia kodu źródłowego i testowania, przy czym praca odbywa się nad niewielkimi fragmentami produktów roboczych, jak się wkrótce przekonasz.

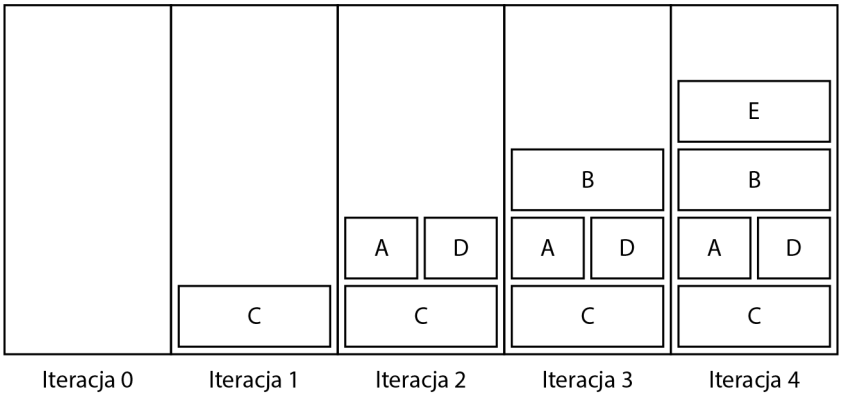
- Zespół dzieli funkcjonalność na wiele różnych historyjek użytkownika, które ogólnie rzecz biorąc, służą do wyjaśnienia danej funkcjonalności, ale w sposób zapisany z perspektywy użytkownika końcowego.
- Następnie te historyjki użytkownika wraz z nadanym priorytetem są dodawane do tzw. backlogu produktu, aby zespół wiedział, co jeszcze pozostało do zrobienia.
- Cały projekt zostaje podzielony na małe iteracje, których czas trwania może wynosić mniej więcej od 2 do 4 tygodni.
- W trakcie każdej iteracji zespół wybiera z backlogu historyjki użytkownika w oparciu o ich priorytet. Następnie kod źródłowy na podstawie tych historyjek może być niezależnie opracowany, przetestowany i dostarczony.
- Przez cały okres pracy nad projektem z zespołem będzie współpracował reprezentant klienta. W projekcie zwinnym jest on nazywany właścicielem produktu. Zadanie reprezentanta to wyjaśnienie zespołowi wymagań i funkcjonalności, nadawanie im priorytetu, a także współpraca z zespołem programistycznym w celu dopracowania wymagań.

Zespół przystępuje do pracy nad iteracją początkową, która może być dłuższa niż iteracja zwykła ze względu na przygotowanie wstępnej konfiguracji i konieczność podjęcia ogólnych działań związanych z planowaniem:

- określenie zakresu projektu;
- utworzenie początkowej architektury systemu;
- podział funkcjonalności na historyjki użytkownika;
- nadanie tym historyjkom priorytetu i dodanie ich do backlogu;
- przygotowanie środowisk programistycznego i testowego;
- zdobycie narzędzi niezbędnych do pracy nad projektem.

Po zakończeniu iteracji początkowej zespół może przejść do stałej długości iteracji, w trakcie których z backlogu będą wybierane historyjki użytkownika na bazie przypisanego im priorytetu. Podczas tych iteracji zespół będzie analizował, opracowywał i testował kod źródłowy powstały na podstawie historyjek użytkownika, korzystając przy tym z pomocy właściciela produktu. Po zakończeniu pracy nad wszystkimi historyjkami użytkownika dla danej funkcjonalności nowo opracowana funkcja może być dostarczona użytkownikowi.

Na rysunku 1.2 pokazałem, jak poszczególne funkcjonalności będą opracowywane w przypadku zastosowania zwinnego modelu tworzenia oprogramowania.



Rysunek 1.2. Iteracje w zwinnym modelu tworzenia oprogramowania

Takie podejście zapewnia wiele korzyści podczas tworzenia oprogramowania w porównaniu z modelem kaskadowym.

Wczesny lub nieprzewidywalny termin dostarczania produktów

Zwinny model tworzenia oprogramowania pozwala na jak najwcześniejsze wydanie produktu o kluczowej funkcjonalności (ang. *minimal viable product*, MVP). Jeżeli bank chce udostępnić aplikację oferującą jedynie funkcjonalność „historia transakcji", takie rozwiązanie jest możliwe jedynie w przypadku użycia zwinnego modelu tworzenia oprogramowania. Jeżeli podczas opracowywania takiej funkcjonalności zespół poświęci trzy tygodnie na początkową iterację, a później dwie kolejne (o długości dwóch tygodni) na dopracowanie historyjek użytkownika, wówczas przed upłynięciem siedmiu tygodni aplikacja będzie gotowa do udostępnienia użytkownikowi końcowemu. Porównaj ten czas do 26 tygodni (pół roku) w przypadku zastosowania modelu kaskadowego.

Kolejne funkcjonalności będą mogły być udostępniane natychmiast po ich opracowaniu na podstawie etapów konfiguracji i planowania przeprowadzonych w trakcie iteracji początkowej.

Wczesne i częste otrzymywanie opinii od użytkowników

W modelu kaskadowym reprezentant klienta ma dostęp do pojedynczego ukończonego produktu dopiero pod koniec procesu tworzenia oprogramowania. Jeżeli na tym etapie będzie miał jakiekolwiek uwagi, wówczas z reguły jest już zbyt późno, by zespół programistyczny mógł efektywnie wprowadzić zmiany w produkcie.

Z kolei w zwinnym modelu tworzenia oprogramowania projekt posiada krótkie iteracje. Na końcu każdej z nich klientowi jest prezentowany działający produkt, choć o ograniczonej funkcjonalności. Dzięki temu zespół pracujący nad projektem może wcześnie i nieustannie otrzymywać od użytkownika uwagi dotyczące danego produktu.

Przekazywane wcześnie i często uwagi pomagają zespołowi skoncentrować się na funkcjonalności o największej wartości biznesowej, która powinna być jak najwcześniej dostarczona klientowi.

Wysokie prawdopodobieństwo, że produkt spełni oczekiwania klienta

Reprezentant klienta (właściciel produktu) współpracuje z zespołem podczas analizowania wymagań, planowania i testowania produktu — od początku do końca pracy nad projektem — i regularnie dostarcza uwagi na jego temat. Ponadto w trakcie każdej iteracji zespół prezentuje działającą wersję oprogramowania właścicielowi produktu oraz szerszemu kręgowi odbiorców. Dlatego też istnieje ogromne prawdopodobieństwo, że tworzone w taki sposób oprogramowanie będzie spełniało wymagania klienta.

Możliwość częstego wprowadzania zmian

Skoro w trakcie poszczególnych iteracji zespół koncentruje się na dostarczeniu uzgodnionych wcześniej funkcjonalności produktu, nieustannie pojawia się możliwość ich udoskonalania i zmiany priorytetów w backlogu produktu. Nowe bądź zmienione elementy backlogu mogą być zaplanowane dla następnej iteracji, co zapewni możliwość wprowadzenia zmian w produkcie w ciągu zaledwie kilku tygodni.

Jeżeli podczas pracy nad projektem bank uzna, że konieczne jest wprowadzenie pewnych zmian dla danej funkcjonalności, bardzo łatwo można je wprowadzić do backlogu i uniknąć przechodzenia przez długi proces żądania zmiany w produkcie.

Koncentracja na użytkowniku

Wymagania w zwinnym modelu tworzenia oprogramowania, czyli historyjki użytkownika zdefiniowane właśnie z jego perspektywy, pozwalają zespołowi zrozumieć wartość biznesową wnoszoną przez poszczególne historyjki. To z kolei pomaga w opracowaniu zadań koniecznych do wykonania na etapach programowania i testowania. Dokładne omówienie historyjek użytkownika znajdziesz w rozdziale 5.

Koncentracja na wartości biznesowej

Zwinny model tworzenia oprogramowania pozwala klientowi bądź jego reprezentantowi na określanie w trakcie projektu priorytetów nadawanych poszczególnym funkcjonalnościom. Zespół jest elastyczny, rozumie to, co jest najważniejsze dla klienta, a także może pracować nad dostarczeniem klientowi najpierw tej funkcjonalności, która ma dla niego największą wartość biznesową.

Jeżeli podczas pracy nad projektem bank zaobserwuje ogromne zapotrzebowanie na pewną funkcjonalność, może zwiększyć priorytet prezentującej ją historyjki użytkownika. Dzięki temu w trakcie następnej iteracji zespół będzie pracował nad potrzebną funkcjonalnością.

Poprawa jakości

Dzięki podzieleniu projektu na mniejsze fragmenty, łatwiejsze w zarządzaniu, pracujący nad nim zespół może skoncentrować się na zapewniających wysoką jakość zadaniach w zakresie tworzenia kodu źródłowego, testowania i współpracy. To z kolei pozwala na częste tworzenie kolejnych wersji produktu, przeprowadzanie testowania i analizowania podczas każdej iteracji. Takie podejście prowadzi do poprawy jakości produktu, ponieważ usterki są szybko znajdowane i usuwane. Natomiast wszelkie rozbieżności dotyczące oczekiwań zostaną wykryte już na wczesnych etapach pracy nad projektem.

Przejrzystość

W projektach agile wszyscy członkowie zespołu, włączając w to reprezentantów klientów, są zaangażowani we wszystkie zadania związane z tworzeniem projektu. Informacje na temat postępu pracy są przekazywane każdemu członkowi podczas codziennych spotkań. Tym samym zachowana zostaje pełna przejrzystość postępów pracy nad projektem.

Na podstawie zamieszczonych tu informacji powinno stać się jasne, że w obecnym środowisku biznesowym, w którym coraz więcej projektów podlega zmianom ze względu na wymagania prawne, działalność konkurencji i nieustanny rozwój technologii, zwinny

model tworzenia oprogramowania będzie lepszym podejściem. Co więcej, jeśli czas udostępnienia produktu na rynku ma znaczenie dla danego projektu, wówczas cel można osiągnąć jedynie, stosując podejście agile.

W następnych rozdziałach dokładniej przedstawię podejście zwinnego modelu tworzenia oprogramowania i wyjaśnię, czym się różni od tradycyjnego podejścia, polegającego na sekwencyjnym wykonywaniu kolejnych etapów.

Twój program działa? Podziękuj testerowi!

W dzisiejszym środowisku biznesowym, w którym wymagania co do aplikacji wciąż się zmieniają, o sukcesie decyduje zwinny model tworzenia oprogramowania! Co więcej, jeśli czas udostępnienia produktu ma znaczenie dla danego projektu, cel można osiągnąć jedynie dzięki podejściu agile. Tester oprogramowania w zwinnym zespole jest bardzo ważną osobą, jednak musi dobrze poznać zasady pracy w tym modelu.

Ta książka pomoże Ci przygotować się do pracy na stanowisku testera w zwinnym zespole programistów. Znajdziesz tu szczegółowe informacje na temat kolejnych etapów testowania, dzięki czemu zrozumiesz, jak w projektach agile te działania są planowane, przeprowadzane i monitorowane. Dowiesz się też, jak tę wiedzę stosować podczas wykonywania codziennych zadań testera w projekcie agile i na czym polegają różne procesy prowadzone przez zwinne zespoły. Co ważne, poszczególne koncepcje i zagadnienia odnoszą się do rzeczywistych projektów — a to najlepszy sposób, by dobrze się przygotować do podjęcia pracy w zawodzie testera agile!

W książce między innymi:

- na czym polega zwinny model tworzenia i testowania oprogramowania
- struktura zwinnego zespołu i zarządzanie wymaganiami w projektach agile
- różne metody testowania stosowane w projektach zwinnych
- zarządzanie zagrożeniami dla jakości produktu w projektach agile
- automatyzacja i różne narzędzia używane przez zwinne zespoły

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 978-83-289-0969-4

9 788328 909694

Cena: 59,00 zł