Tilo Linz

Testowanie w procesie Scrum

Przewodnik po zarządzaniu jakością oprogramowania
w świecie programowania zwinnego

Przekład: Jakub Niedźwiedź

APN Promise, Warszawa 2014

Testowanie w procesie Scrum. Przewodnik po zarządzaniu jakością oprogramowania w świecie programowania zwinnego

Title of German original:
Testen in Scrum-Projekten
ISBN 978-3-89864-799-1

Projekt okładki: Helmut Kraus, www.exclam.de

APN PROMISE SA, biuro: ul. Kryniczna 2, 03-934 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: mspress@promise.pl

ISBN: 978-83-7541-154-6

Przekład: Jakub Niedźwiedź
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Podziękowania

Chociaż to moje nazwisko pojawia się na okładce, książka ta nie byłaby możliwa bez porad i wsparcia wielu moich kolegów.

Chciałbym szczególnie podziękować rozmówcom i autorom studiów przypadków, którzy recenzowali tę książkę; są to dr Stephan Albrecht z firmy Avid, Dierk Engelhardt z firmy imbus, Andrea Heck z firmy Siemens, Eric Hentschel z firmy ImmobilienScout24, Sabine Herrmann z zooplus, Joachim Hofer z firmy imbus oraz Terry Zuo z firmy GE Oil & Gas. Wiele naszych wspólnych rozmów dało mi mnóstwo cennych informacji na temat implementacji technik programowania zwinnego oraz zastosowania procesu Scrum w rzeczywistych warunkach.

Chciałbym też podziękować swoim eksperckim recenzentom za ich cenne uwagi, sugestie i poprawki; są to Oliver Gradl z firmy Siemens (za jego wkład w integrację technik programowania zwinnego i testowanie systemów), dr Stefan Kriebel z BMW oraz Horst Pohlmann (za wiedzę dotyczącą systemów wbudowanych), Stefan Schmitz z iq-stz (za jego rozległą znajomość standardów ISO 9000 i prowadzenia audytów), Uwe Vigenschow z oose Innovative Informatik (za owocne dyskusje na temat testów akceptacyjnych) oraz prof. Mario Winter z Uniwerstytetu w Kolonii (który mimo że był zajęty projektem własnej książki, poświęcił swój czas jako recenzent i zaoferował cenne materiały do rozdziału dotyczącego testów integracyjnych). Podziękowania należą się też innym recenzentom, których nazwisk tutaj nie wymieniłem.

Dziękuję też całemu zespołowi firmy imbus za ich cenne rady i czas poświęcony na wspieranie tego projektu, szczególnie należą do nich Arne Becher, dr Christian Brandes, Thomas Roßner i Carola Wittigschlager. Serdeczne podziękowania kieruję też do Claudii Wissner, bez której wiele ilustracji pozostałoby na etapie szkiców.

Dziękuję też Matthiasowi Rossmanith z wydawnictwa Rocky Nook za jego niestrudzoną pomoc podczas produkcji tej książki i jego cierpliwość, gdy okazało się, że potrzebny jest jeszcze jeden sprint lub dwa.

Na koniec kieruję wielkie podziękowania dla swojej żony, Sabine i naszych córek Lisy i Leny, które musiały radzić sobie beze mnie przez wiele wieczorów i weekendów, gdy pracowałem nad tym tekstem.

Życzę wszystkim ciekawej lektury i wielu sukcesów w zastosowaniu tych technik w swoim własnym środowisku testowym.

Tilo Linz, luty 2014

Spis treści

1 Wprowadzenie

Oprogramowanie jest wszędzie. Właściwie każdy złożony wyrób produkowany obecnie jest sterowany przez oprogramowanie, a wiele usług komercyjnych opiera się na systemach informatycznych. Jakość wykorzystywanego oprogramowania jest więc istotnym czynnikiem dla bycia konkurencyjnym. Im szybciej firma będzie mogła zintegrować nowe oprogramowanie w swoich produktach lub dostarczyć produkty programowe na rynek, tym większą będzie miała okazję do pokonania konkurencji.

Metody programowania zwinnego (*agile*) zaprojektowano w celu ograniczenia czasu dostarczenia produktu na rynek i poprawienia jakości oprogramowania przy jednoczesnym zwiększeniu dopasowania produktów do potrzeb klientów. Nie jest zaskoczeniem, że zastosowanie metodyki zwinnej staje się coraz popularniejsze w dużych projektach międzynarodowych, a także w rozwijaniu produktów w różnego rodzaju wielkich korporacjach. W większości przypadków oznacza to przejście z wypróbowanego modelu V do testowania zwinnego przy użyciu metodyki Scrum.

Jednakże przejście na programowanie zwinne i nauczenie się korzystania z niego w sposób efektywny i wydajny nie zawsze jest łatwe, zwłaszcza gdy zaangażowanych jest wiele zespołów. Każdy członek zespołu, menedżer projektu i wszyscy członkowie zarządzający muszą dokonać znaczących zmian w swoim sposobie pracy. W szczególności skuteczność testowania oprogramowania i zarządzania jakością stanowi istotny czynnik sukcesu przy wprowadzaniu i wykorzystywaniu metodyki zwinnej w zespole, dziale lub całym przedsiębiorstwie oraz będzie wpływać na potencjalną możliwość zastosowania jej w praktyce.

Większość literatury na temat metodyki zwinnej (w tym wiele książek podanych w bibliografii na końcu tej książki) napisano z punktu widzenia programistów i inżynierów oprogramowania. Kładą one główny nacisk na techniki programistyczne i zwinne zarządzanie projektami – testowanie jest zwykle jedynie wspominane przy omawianiu testów jednostkowych i związanych z nimi narzędzi. Innymi słowy skupiają się na testowaniu z punktu widzenia programisty. Jednakże

same testy jednostkowe nie są wystarczające i szersze ujęcie testowania jest istotne dla sukcesu procesów programowania zwinnego.

Celem tej książki jest wypełnienie tej luki i opisanie procesu programowania zwinnego z punktu widzenia testowania i zarządzania jakością. Przedstawia ona, jak działa testowanie zwinne i pokazuje, gdzie tradycyjne techniki testowania są nadal konieczne w środowisku zwinnym oraz jak je wkomponować w podejście zwinne.

1.1 Grupa docelowa

Zrozumienie, jak działa testowanie w projektach zwinnych

Z jednej strony książka ta jest przeznaczona dla początkujących, którzy dopiero zaczynają pracę z metodyką zwinną, mają w planach pracę nad projektami zwinnymi lub zamierzają wprowadzić (lub właśnie wprowadzają) Scrum do swojego projektu lub zespołu.

- Zapewniamy wskazówki i porady, w jaki sposób menedżerowie zarządzający produktem, menedżerowie projektów, menedżerowie testów i menedżerowie jakości mogą pomóc w wykorzystaniu pełni potencjału metodyki zwinnej.
- Profesjonalni (certyfikowani) testerzy i eksperci od zarządzania jakością dowiedzą się, jak wydajnie pracować w zespołach zwinnych i optymalnie wykorzystywać swoje doświadczenie. Można się też dowiedzieć, jak dostosować swój styl pracy tak, aby dopasować go do środowiska zwinnego.

Wzbogacanie swojej wiedzy na temat (zautomatyzowanego) testowania i zwinnych technik zarządzania jakością

Z drugiej strony kierujemy tę książkę również do czytelników, którzy mają już doświadczenie w pracy w zespołach zwinnych i chcą poszerzyć swoją wiedzę na temat testowania i technik zarządzania jakością w celu zwiększenia wydajności i poprawienia jakości produktów.

- Menedżerowie produktów, mistrzowie Scrum (Scrum Master), pracownicy zarządzania jakością i członkowie zespołów zarządzających dowiedzą się, jak działa systematyczne, zautomatyzowane testowanie i jakie jest znaczenie zapewnienia stałych, niezawodnych i wyczerpujących informacji zwrotnych na temat jakości tworzonego oprogramowania.
- Programiści, testerzy i inni członkowie zespołu zwinnego dowiedzą się, jak stosować wysoce zautomatyzowane metody testowania do przeprowadzania testów jednostkowych, integracyjnych i systemowych.

Tekst tej książki zawiera wiele praktycznych przykładów i pytań testowych, co sprawia, że dobrze nadaje się ona na podręcznik lub samouczek.

1.2 Zawartość książki

Rozdział 2

Rozdział 2 zapewnia krótkie omówienie popularnych obecnie metodyk Scrum i Kanban oraz omówienie najważniejszych metod zarządzania zwinnego dla menedżerów zamierzających wdrażać metody zwinne w swoich departamentach. Metody te są porównywane z metodami tradycyjnymi, żeby pokazać zmiany wprowadzane przez metody zwinne. Czytelnicy, którzy są już zaznajomieni z metodyką Scrum i Kanban, mogą pominąć ten rozdział.

Rozdział 3

Rozdział 3 opisuje proste narzędzia planowania i sterowania wykorzystywane przez Scrum zamiast tradycyjnych narzędzi planowania projektów. Warto pamiętać, że praca w sposób zwinny (*agile*) nie oznacza pracy bez wyznaczonego celu! Rozdział 3 jest też przeznaczony głównie dla czytelników, którzy przechodzą na programowanie zwinne. W każdym razie wyjaśnienia i wskazówki dotyczące znaczenia narzędzi planowania dla konstruktywnego zarządzania jakością i zapobiegania błędom będą przydatne również dla zaawansowanych czytelników.

Rozdział 4

Rozdział 4 obejmuje testowanie jednostkowe i techniki programowania opartego na testach. Omówione zostanie między innymi, czego można oczekiwać od testów jednostkowych i jak je można zautomatyzować. Ten rozdział omawia podstawy technik i narzędzi testowania jednostkowego oraz pomoże testerom systemowym, specjalistom od testowania i członkom zespołów projektowych mającym niewielkie doświadczenie w testowaniu jednostkowym w bliższej i skuteczniejszej współpracy z programistami i testerami jednostkowymi. Zawiera też sporo przydatnych wskazówek, które pomogą zaawansowanym programistom i testerom w poprawieniu swoich technik testowania. Wyjaśnimy też oparte na testach podejście do programowania i jego znaczenie w projektach zwinnych.

Rozdział 5

Rozdział 5 omawia testowanie integracyjne i pojęcie ciągłego testowania integracyjnego. Testowanie integracyjne obejmuje przypadki testów, które są pomijane nawet przez najdokładniejszych programistów przy korzystaniu z testów jednostkowych. Ten rozdział obejmuje wszystkie podstawy integracji oprogramowania i testów integracyjnych oraz wprowadza techniki ciągłej integracji wyjaśniając, jak korzystać z nich w projektach.

Rozdział 6

Rozdział 6 omawia testowanie systemowe i pojęcie testowania non-stop. W oparciu o podstawy testowania systemowego rozdział ten wyjaśnia, jak korzystać z technik zwinnych do przeprowadzania testów ręcznych. Wyjaśnia też, jak automatyzować testy systemowe i umieszczać je w procesie ciągłej integracji. Ten rozdział jest przeznaczony nie tylko dla testerów systemowych i innych specjalistów od testowania, ale również dla programistów, którzy chcą lepiej zrozumieć techniki

testowania zwinnego, które znajdują się poza ich zwykłymi kompetencjami programistycznymi.

Rozdział 7

Rozdział 7 porównuje tradycyjne i zwykłe sposoby zarządzania jakością oraz wyjaśnia konstruktywne i profilaktyczne praktyki zarządzania jakością, które są wbudowane w Scrum. Ten rozdział zawiera wskazówki, jak przeprowadzać zwinne zarządzanie jakością i wyjaśnia, jak wszyscy specjaliści zarządzania jakością mogą wykorzystać swoją wiedzę w projektach zwinnych i przyczynić się do ich sukcesu.

Rozdział 8

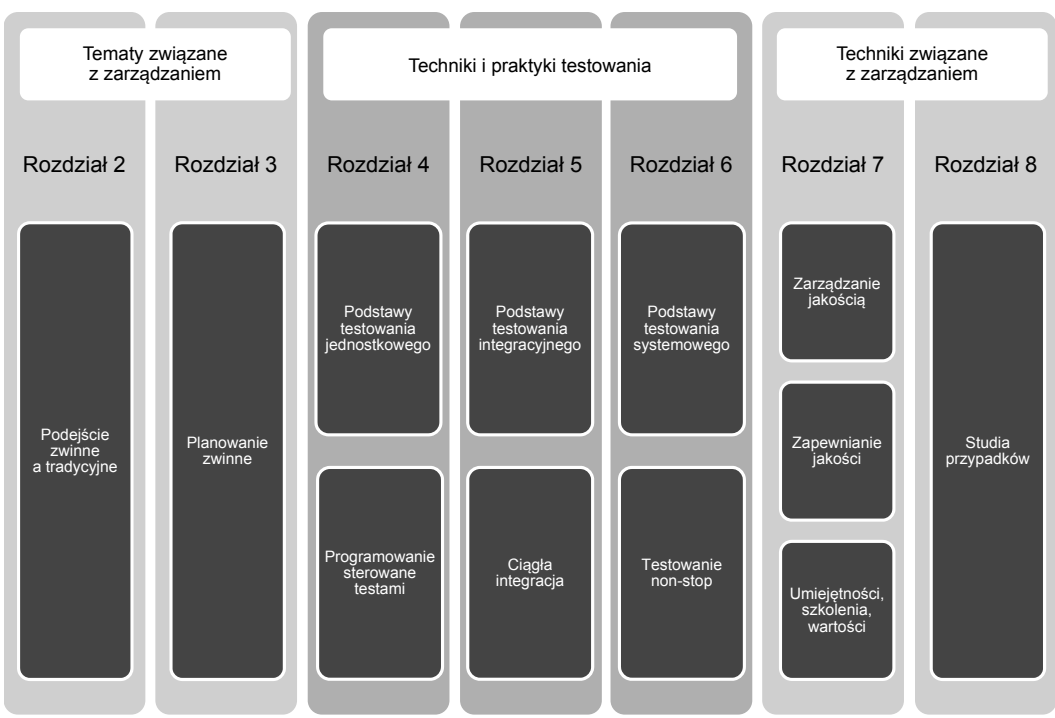
Rozdział 8 przedstawia sześć przemysłowych studiów przypadku z dziedziny programowania. Odzwierciedlają one nabyte doświadczenie i wnioski zebrane przez respondentów podczas wprowadzania i stosowania technik zwinnych.

Omówienie rozdziałów

Rozdziały 2, 3, 7 i 8 omawiają zagadnienia związane z procesami oraz zarządzaniem i są przeznaczone dla czytelników, którzy są bardziej zainteresowani zarządczymi aspektami tej tematyki. Rozdziały 4, 5 i 6 omawiają (zautomatyzowane) testowanie zwinne na różnych poziomach i są przeznaczone dla bardziej technicznie nastawionych czytelników. Ponieważ testowanie zwinne nie jest tym samym, co testowanie jednostkowe, wyjaśniamy też szczegółowo cele i różnice pomiędzy testowaniem jednostkowym, integracyjnym i systemowym.

Rys. 1–1
Struktura książki Studium przypadku, podsumowanie i ćwiczenia

Rysunek 1-1 przedstawia wizualny przegląd struktury tej książki:



Według najpopularniejszej literatury na ten temat wiele zagadnień, technik i praktyk programowania zwinnego jest prostych w implementacji. Zagadnienia, porady i wskazówki omówione w kolejnych rozdziałach również mogą na pierwszy rzut oka wydawać się proste, ale najważniejsze kwestie staną się jasne dopiero w trakcie implementacji. Żeby pomóc w zrozumieniu wyzwań z tym związanych, tekst zawiera:

- Fikcyjne studium przypadku ilustrujące omawiane metodyki i techniki.
- Testy i ćwiczenia pomagające zapamiętać treści omawiane w każdym rozdziale i przeanalizować swoją własną sytuację i zachowanie w ramach projektu.

1.3 Studium przypadku

Fikcyjne studium przypadku, do którego odwołuje się ta książka, korzysta z następującego scenariusza. Firma eHome Tools jest twórcą systemów automatyki domowej opartych na następujących elementach:

Studium przypadku: eHome Tools

- **Elementy wykonawcze:** Lampy i inne urządzenia elektryczne są przyłączane do systemu przy użyciu przełączników elektrycznych. Każdy element wykonawczy jest połączony przewodem lub bezprzewodowo z magistralą komunikacyjną, która może służyć do sterowania nim zdalnie.
- **Czujniki:** Czujniki temperatury, wiatru i wilgotności oraz proste czujniki styku (na przykład wskazujące na otwarte okno) również mogą być przyłączone do magistrali.
- **Magistrala:** Instrukcje przełączające i sygnały o stanie dla elementów wykonawczych i czujników są przesyłane przez magistralę do centralnego sterownika w formie „telegramów".
- **Sterownik:** Centralny sterownik wysyła sygnały przełączające do elementów wykonawczych (np. „włącz światło w kuchni") i odbiera sygnały o stanie z czujników (np. „temperatura w kuchni 20 stopni") i elementów wykonawczych (np. „światło w kuchni jest włączone"). Sterownik jest albo sterowany zdarzeniami (tzn. reaguje na nadchodzące sygnały, albo działa w oparciu o zaplanowane polecenia (np. „o 8 wieczorem zasuń żaluzje w kuchni").
- **Interfejs użytkownika:** Sterownik ma interfejs użytkownika, który wizualizuje aktualny stan wszystkich elementów elektronicznego domu i umożliwia mieszkańcom wysyłanie poleceń (np. „wyłącz światło w kuchni") dla systemów elektronicznego domu.

Firma eHome Tools ma wielu konkurentów i aby pozostać konkurencyjną zamierza opracować nowe oprogramowanie sterownika. Coraz większa liczba klientów prosi o oprogramowanie, którym można

sterować za pośrednictwem smartfonów i innych urządzeń mobilnych, od początku jest więc jasne, że projekt ten musi być ukończony szybko, jeśli ma się udać. Ważne jest, aby nowy system był łatwy do rozszerzania i otwarty na urządzenia firm trzecich. Jeśli nowy system będzie w stanie kierować urządzeniami innych producentów, to w odczuciu zarządu firma może zdobyć klientów, którzy chcą rozszerzać swoje istniejące systemy. Aby osiągnąć ten cel, nowy system musi obsługiwać możliwie szeroki zakres istniejącego sprzętu innych firm i musi być w stanie przystosowywać się do obsługi nowych urządzeń, jak tylko będą się one pojawiać na rynku.

W obliczu tych wyzwań podjęta zostaje decyzja o opracowaniu nowego systemu przy użyciu metodyki zwinnej i wypuszczaniu zaktualizowanej wersji oprogramowania sterownika (z obsługą nowych urządzeń i protokołów) co miesiąc.

1.4 Strona WWW

Przykłady kodu zawarte w tekście są dostępne do pobrania ze strony WWW tej książki [URL: SWT-knowledge][1]. Można z nich swobodnie korzystać do tworzenia swoich własnych przypadków testowych.

Pytania i ćwiczenia są również dostępne w sieci i można je komentować. Chętnie omówimy komentarze i sugestie czytelników.

Pomimo podjęcia wszelkich wysiłków ze strony autora i wydawcy, możliwe, że tekst zawiera jakieś błędy. Wszelkie niezbędne poprawki będą publikowane online.

1 Spis odnośników internetowych i innych źródeł bibliograficznych znajduje się w Dodatku B (przyp. red.).

2 Podejście zwinne a tradycyjne

Ten rozdział przedstawia platformę zarządzania zwinnym projektem Scrum oraz popularną metodykę zarządzania projektami Kanban. Kanban został rozwinięty na bazie teorii szczupłej produkcji i wykazuje pewne podobieństwa do Scrum. Porównamy obie te metody z tradycyjnymi modelami procesów. Osobom zaangażowanym w zarządzanie przejściem do metod zwinnych na poziomie projektu lub całego działu rozdział ten da ogólne wyobrażenie zmian potrzebnych na poziomie przedsiębiorstwa, działu i zespołu projektowego. Osoby zaznajomione z działaniem Scrum i/lub Kanban mogą pominąć ten rozdział.

2.1 Scrum

Scrum jest platformą zwinnego[1] zarządzania projektami wprowadzoną po raz pierwszy w 1999 roku przez Kena Schwabera w artykule *Scrum: A Pattern Language for Hyperproductive Software Development* [Beedle i in. 99]. Wydana w 2002 roku książka *Agile Software Development with Scrum* [Schwaber/Beedle 02] pomogła metodyce Scrum dotrzeć do szerszej publiczności.

Scrum nie określa, jakie konkretne techniki powinny być stosowane przez twórców oprogramowania (np. refaktoring), natomiast pozostawia takie decyzje samemu zespołowi programistów. Zwykle prowadzi to do wykorzystania technik programowania ekstremalnego[2] (XP),

1 Termin zwinne (agile) jest stosowany do opisywania lekkich procesów rozwijania oprogramowania, które znacznie różnią się od tradycyjnych, „ciężkich" metod. Termin ten został ukuty na konferencji programistycznej w Utah w 2001, gdzie powstał szkic manifestu programowania zwinnego (*Agile Manifesto*) (zobacz [URL: Agile Manifesto]).

2 Programowanie ekstremalne (Extreme Programming – XP) jest luźnym zbiorem wartości, zasad i praktyk programowania zwinnego wprowadzonym przez Kenta Becka w roku 1999. Większość procesów programowania zwinnego opiera się na elementach XP. Aktualna wersja XP jest wyjaśniona przez Becka w [Beck/Andres 04].

które są wprowadzane podczas przechodzenia na metodykę Scrum. Ponadto Scrum nie dyktuje, jakim rodzajom testów ma być poddawany zwinny projekt.

Zwinne praktyki w zarządzaniu

Platforma Scrum opisuje zestaw praktyk zarządzania projektami (programistycznymi), które radykalnie zmieniają wykorzystywane procesy i zastępują tradycyjne podejście planowania deterministycznego przez adaptacyjny, empiryczny system sterowania projektem [Schwaber/Beedle 02]. Głównym celem wykorzystania Scrum jest umożliwienie zespołowi projektowemu szybkiego, prostego i właściwego reagowania zamiast tracenia czasu i energii na tworzenie, implementowanie i poprawianie nieaktualnych planów.

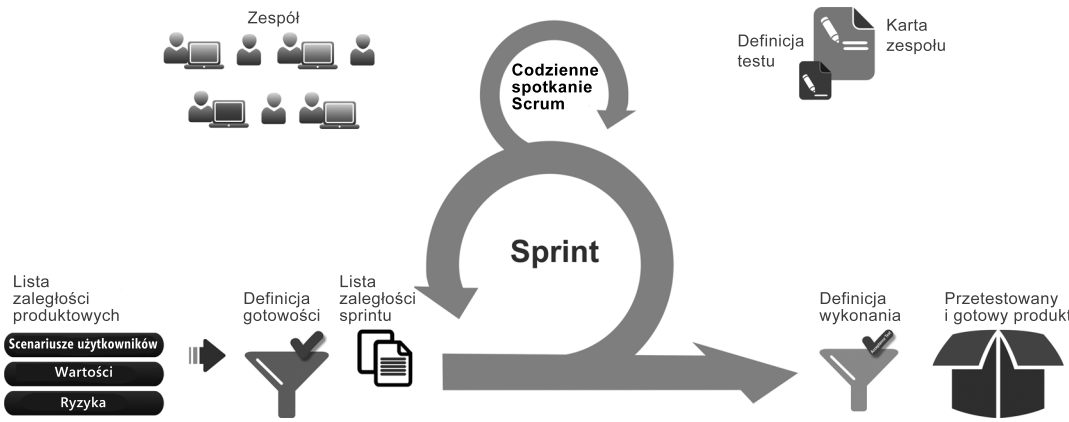
Podstawowymi instrumentami zarządzania projektami wykorzystywanymi przez Scrum są:

- **Sprint:** Scrum dzieli projekt na krótkie iteracje o ustalonej długości zwane sprintami[3]. Chodzi o to, że sprint trwający trzy lub cztery tygodnie można planować i zarządzać nim prościej i skuteczniej niż tradycyjnymi cyklami wypuszczania oprogramowania trwającymi rok lub więcej (zobacz [Schwaber/Beedle 02, str. 52]).
- **Przyrost produktowy:** Produkt rośnie z każdą iteracją, więc każdy sprint daje w wyniku działający i potencjalnie nadający się do wypuszczenia/sprzedaży produkt (zwany przyrostem).
- **Zaległość produktowa:** Jeśli planowanie jest ograniczone do jednowymiarowej listy celów, projekt staje się znacznie mniej skomplikowany, a dokładnie o to chodzi w metodyce Scrum. Właściciel produktu (zobacz poniżej) utrzymuje listę tzw. zaległości produktowych, która według [Pichler 10] zawiera wszystkie znane wymagania i wyniki, które mają zostać zaimplementowane lub osiągnięte, aby udanie zrealizować projekt. Obejmuje to wymagania funkcjonalne i niefunkcjonalne, a także specyfikacje interfejsów. Zaległość produktowa może też zawierać elementy robocze, takie jak struktura środowiska testowego i programistycznego oraz debugowania. Elementy wymagań produktowych na tej liście mają nadane priorytety hierarchizujące je względem siebie, ale poza tym nie są zakładane żadne współzależności lub założenia czasowe. Lista zaległości produktowych nie jest ustalona raz na zawsze i stale się zmienia wraz z kończeniem każdego sprintu. Utrzymywanie zaległości na bieżąco jest zwane czyszczeniem zaległości. Właściciel produktu odpowiada za dodawanie nowych wymagań, a jeśli to konieczne,

3 Metafora „sprintu" odnosi się nie do wysokich prędkości, ale do krótkich dystansów.

rozbijanie ich na mniejsze jednostki tak, aby cały zespół dobrze rozumiał nowe wymaganie. Wymagania podlegają ciągłej analizie i przydzielaniu priorytetów na nowo. Przestarzałe wymagania są usuwane. Mówiąc zwięźle, Scrum upraszcza planowanie i czyni je bardziej niezawodnym, ponieważ unikamy wszystkich czynników źle wpływających na niezawodność planowania konwencjonalnego.

- **Zaległość sprintu:** Oczywiście w rzeczywistości sprawy nie są tak całkiem proste. Złożony projekt oprogramowania komputerowego nie może być zarządzany jedynie przy użyciu długiej listy zhierarchizowanych wymagań, a zespół Scrum musi ustalić, którzy członkowie zespołu mają do wykonania które zadania i kiedy. Członkowie zespołu i właściciel produktu podejmują te decyzje krok po kroku dla każdego kolejnego sprintu. Na początku sprintu zespół sięga po zadania ze szczytu listy zaległości produktowych (te o najwyższym priorytecie) i wykorzystuje je do utworzenia ściślejszej listy zaległości sprintu. W tym momencie istotne jest, aby wymagania na nowej liście zaległości były dobrze rozumiane i precyzyjnie określone. Wymagania, które nie spełniają tych kryteriów, nie są jeszcze gotowe na to, aby je włączyć do sprintu. W tym momencie podejmowane są decyzje dotyczące współzależności pomiędzy wymaganiami, potrzebnych zadań i zasobów oraz ram czasowych całego sprintu. Wszystkie zadania, które zespół uzna za konieczne do udanego zakończenia sprintu, są teraz wprowadzane do listy zaległości sprintu.
- **Definicja wykonania:** W celu zapewnienia, że wynikiem sprintu będzie działająca wersja produktu (potencjalnie gotowy produkt), zespół nakreśla kryteria, które umożliwiają mu ocenę, czy praca nad danym przyrostem produktu jest ukończona [URL: Scrum Guide]. Użyte kryteria są określane przez zespół jako „definicja wykonania" i mogą być sporządzone w postaci globalnej listy kontrolnej dla całego sprintu lub w odniesieniu do poszczególnych zadań w ramach tego sprintu. Dyskusja nad tymi kryteriami realizacji jest istotna dla jasnego zdefiniowania wymagań i zadań dla zespołu, choć odnoszą się one tylko do bieżącego sprintu. Horyzont planowania jest krótkoterminowy a zakres prac do wykonania jest niewielki w porównaniu z całym projektem. To podejście oznacza, że zespół jest skoncentrowany na bieżących zadaniach, a zaległości sprintu nie mogą być zmieniane podczas danego sprintu (zobacz [Pichler 10]). Ten typ planowania daje duże szanse na zakończenie projektu z powodzeniem.



Rys. 2–1
Podstawowe zasady procesu Scrum

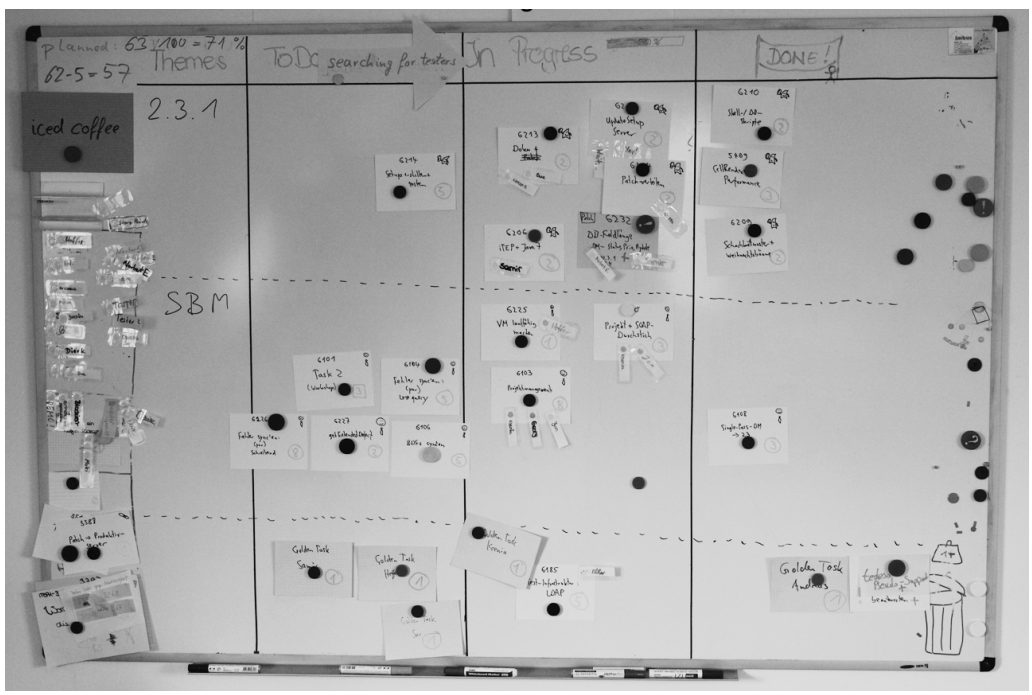
- **Ramy czasowe:** Wymaganiem względem każdego sprintu jest dostarczenie potencjalnie gotowego produktu[4]. To z kolei oznacza, że zaległości sprintu powinny zawierać tylko te zadania, które przyczyniają się do stworzenia gotowego produktu. Wszelkie elementy produktu, których nie można ukończyć podczas sprintu, należy odłożyć, a w tym celu na początku sprintu zespół musi wybrać te funkcje, które realistycznie da się ukończyć. W razie wątpliwości elementy możliwe do zrealizowania mają pierwszeństwo nad elementami funkcjonalnymi – zgodnie z tzw. zasadą ram czasowych. Zespół musi oszacować ilość pracy związanej z ukończeniem wszystkich funkcji, które zamierza dołączyć do danego sprintu. Funkcje, które wymagają zbyt dużo pracy, są odkładane, dzielone na mniejsze zadania lub zakres ich docelowej funkcjonalności jest ograniczany. Podobnie jak w przypadku tradycyjnych procesów planowania zespoły Scrum muszą podjąć wysiłek związany z szacowaniem, choć dwa istotne czynniki zapewniają, że szacunki oparte na Scrum są dokładniejsze od tradycyjnych:
  - Zespół musi jedynie brać pod uwagę pracę związaną z aktualnym sprintem. Analizowane zadania mają ograniczony rozmiar, a dzięki wcześniej zakończonym sprintom zespół jest zwykle dobrze przygotowany.
  - Szacunków dokonuje się przy pomocy pokera planistycznego będącego kolejną techniką XP (zobacz podrozdział 3.5). Szacunki dokonywane przez poszczególnych członków zespołu mogą się znacznie różnić, ale po uśrednieniu są dość dokładne.

4 „Przyrostowe dostarczanie ‘gotowych’ produktów zapewnia, że potencjalnie użyteczna wersja działającego produktu jest zawsze dostępna” [URL: Scrum Guide].

Zasada ram czasowych nie dotyczy jedynie sprintów, ale też innych sytuacji, które wymagają realizacji konkretnego zadania od zespołu. Na przykład może być używana do zapewniania, aby zebrania rozpoczynały się punktualnie i trwały tylko przez ustalony okres czasu.

- **Przeźroczystość:** Przeźroczystość jest jednym z najpotężniejszych narzędzi Scrum. Zaległości sprintu są prowadzone publicznie na tablicy[5], co sprawia, że zawartość i postępy bieżącego sprintu są łatwo zrozumiałe dla zespołu, zarządu i innych zainteresowanych stron. Tablica zawiera wymagania i związane z nimi zadania ułożone w rzędach, natomiast kolumny reprezentują aktualny stan każdego zadania (do wykonania, w trakcie realizacji lub ukończone). Stan sprintu jest aktualizowany codziennie podczas spotkania Scrum, a poszczególne karty zadań są przestawiane na tablicy zgodnie z aktualnymi postępami każdego zadania. Rysunek 2-2 przedstawia tablicę zespołu Test*Bench* (zobacz studium przypadku 8.2).

Rys. 2–2
Przydzielanie ról w ramach zespołu Tablica zespołu TestBench



Przeźroczystość zapewniana przez aktualizowanie tablicy podczas codziennego spotkania Scrum oznacza, że każdy członek zespołu wie, co się dzieje wokół niego (co pomaga zapobiegać wchodzeniu

5 Ruchoma ścianka działowa jest dobrą alternatywą. Jeśli zespół korzysta z narzędzi informatycznych, to lista zaległości sprintu powinna być wyświetlana na czołowej pozycji na dużym monitorze w sali zespołu.

sobie w drogę), a każdy jest świadomy postępów zadań poszczególnych osób, co automatycznie wywiera nacisk na sukces wewnątrz zespołu. Zespół jest zachęcany do omawiania i rozwiązywania problemów, które się pojawiają[6]. Gdy trudności są jawnie widoczne, dużo łatwiej jest zaoferować lub przyjąć pomoc. Ponadto codzienne omawianie zadań i wyników zapewnia stały napływ informacji o elementach zrealizowanych przez poszczególnych członków zespołu i zespół jako całość.

W zgodzie z tymi technikami zarządzania projektami znaczenie zespołu i przydziału ról jest inaczej definiowane w środowisku Scrum. Model Scrum zakłada jedynie trzy główne role w ramach zespołu (za [Schwaber/Beedle 02, rozdz. 3]):

- **Mistrz Scrum** jest nowym typem roli zarządzającej. Jest osobą odpowiedzialną za zapewnianie, aby praktyki Scrum były implementowane i realizowane. Gdyby praktyki te stawały się zbyt przeciążone lub utrudnione, zadaniem mistrza Scrum jest zatrzymanie procesu lub znalezienie alternatywnego rozwiązania. Mistrz Scrum nie ma realnej władzy, ale działa raczej jak trener. Przeszkody związane z procesem mogą być często rozwiązywane przez wyjaśnianie zespołowi szczegółów procesu, przypominanie o odpowiednich procedurach lub przeprowadzanie warsztatów. Inne przeszkody (na przykład niedziałające środowisko budowania) czasami wymagają zaangażowania dodatkowych zasobów (takich jak szybszy serwer lub bardziej odpowiednie narzędzie) i konieczne może być przydzielenie lepiej wykwalifikowanych członków zespołu do procesu budowania. Mistrz Scrum nie powinien nigdy przekazywać nierozwiązanych problemów z powrotem do zespołu. Jeśli dzieje się tak regularnie, rozwiązaniem jest zwykle znalezienie nowego mistrza Scrum.
- **Właściciel produktu** odpowiada za aktualność listy zaległości produktowych i stanowi interfejs pomiędzy klientem a zespołem[7]. Właściciel produktu decyduje, które funkcje mają być implementowane – innymi słowy on lub ona odpowiada za określenie natury projektu. W praktyce rola właściciela produktu może być spełniana przez aktualnego menedżera projektu, lidera zespołu lub szefa projektu[8]. Właściciel projektu nie jest przywódcą zespołu

6 Nie mogą być ukrywane.

7 Jeśli dany projekt obejmuje specyfikacje klienckie, właściciel produktu może być członkiem własnego zespołu klienta.

8 Osoby o różnym doświadczeniu podchodzą do roli właściciela produktu w różny sposób i będą wykorzystywać zaległości produktowe zgodnie

i nie nadzoruje całego procesu Scrum, co jest obowiązkiem mistrza Scrum.

- **Zespół programistów**[9] zwykle składa się z trzech do dziewięciu członków, którzy wykonują zadania konieczne do ukończenia produktu sprint po sprincie. „Zespół (programistów) Scrum powinien składać się z osób mających wszystkie umiejętności potrzebne do osiągnięcia celów sprintu. Scrum wystrzega się pionowo zorganizowanych zespołów analityków, projektantów, programistów i inżynierów zarządzania jakością. Zespół Scrum samodzielnie organizuje się tak, aby każdy przyczyniał się do końcowego wyniku. Każdy członek zespołu wykorzystuje swoje doświadczenie do rozwiązywania wszystkich problemów. Synergia wynikająca z tego, że tester pomaga projektantowi skonstruować kod, poprawia jakość kodu i zwiększa wydajność. Niezależnie od składu zespołu, ważne jest realizowanie całej analizy, projektowania, kodowania, testowania i dokumentacji" ([Schwaber/Beedle 02]).

W związku z tym zespoły powinny być elastyczne[10]. Jest to trudne do osiągnięcia i zwykle oznacza, że zamiast budowania zespołu, w którym każdy ma porównywalne umiejętności, wszyscy członkowie zespołu chętnie pracują nad wszystkimi bieżącymi zadaniami korzystając ze swoich konkretnych umiejętności. Scrum skupia się na optymalizowaniu możliwości i wydajności zespołu jako całości.

Zespoły wielofunkcjonalne

Fakt, że Scrum zachęca do korzystania z elastycznych zespołów i płaskich hierarchii w zespole jest często opacznie rozumiany. Wielofunkcjonalność oznacza, że członkowie zespołu pracują na różnych podstawach funkcjonalnych. Na przykład architekt systemowy i programista tworzą wspólnie architekturę, a architekt pomaga programiście w fazie kodowania, ucząc się przy okazji, że teoretyczne projekty systemów mogą być dość trudne do praktycznego zaimplementowania. Analogicznie tester może pomagać programiście w definiowaniu odpowiednich testów jednostkowych, być może ucząc się przy tym, jak automatyzować testowanie w ramach procesu. Niemniej jednak każdy członek zespołu ma swoje własne, specyficzne umiejętności i wykorzystuje je do pracy nad bieżącymi zadaniami. Chodzi o to, że nikt nie może stwierdzić na przykład, że „jestem testerem i nie robię

z własnymi umiejętnościami i preferencjami. Wszyscy członkowie zespołu powinni być tego świadomi, zwłaszcza podczas pierwszej implementacji Scrum.

9 „Zespół Scrum składa się z właściciela produktu, zespołu programistów i mistrza Scrum". [URL: Scrum Guide]. [Schwaber/Beedle 02] nie rozróżnia pomiędzy zespołem a zespołem programistów, dlatego termin ten jest użyty powyżej w nawiasie.

10 W XP [Beck/Andres 04] to pojęcie znane jest jako zasada jednego zespołu, natomiast Scrum [Schwaber/Beedle 02] nazywa to zespołem wielofunkcjonalnym.

nic innego". Studia przypadków 8.1 i 8.4 w szczególności ilustrują niektóre przeszkody, które można napotkać podczas przechodzenia na metodykę Scrum.

Studium przypadku 2-1

Projekt sterownika eHome 2-1: przygotowanie projektu

Decyzja o implementacji projektu zawiera budżet obejmujący trzech programistów, jednego testera, właściciela produktu i mistrza Scrum.

Różne są koncepcje dotyczące tego, kto powinien należeć do zespołu. Niektóre głosy są za wybraniem najlepszych pracowników, inni są za najbardziej doświadczonymi, a jeszcze inni woleliby widzieć w zespole najnowszych pracowników z nowymi pomysłami. Członkowie istniejącego zespołu też mają różniące się opinie na temat wprowadzenia metodyki Scrum. Podczas gdy niektórzy sądzą, że powinno się ją wprowadzić już dawno, to inni uważają, że trudno tradycjonalistom będzie przejść na nowy sposób działania. Niektórzy uważają, że już pracują w sposób zwinny, natomiast bardziej technicznie nastawieni członkowie zespołu sądzą, że niemożliwe będzie zagwarantowanie ciągłej funkcjonalności dokładnie zdefiniowanych protokołów magistrali i predefiniowanych standardów bez użycia tradycyjnych specyfikacji i dokumentacji.

Ostatecznie konieczność zapełnienia ról wewnątrz zespołu pracownikami mającymi odpowiednią wiedzę uprościła znacznie decyzję. Co najmniej jeden z programistów musi wiedzieć wszystko na temat protokołów magistrali i urządzeń sprzętowych (dotyczy to zarówno własnych produktów firmy, jak i produktów konkurencji), a wiedza ta jest już dostępna w firmie. Kreatywny programista WWW powinien odpowiadać za interfejs użytkownika, więc to stanowisko wymaga zaangażowania nowego pracownika. Obecny szef zespołu programistycznego przejmuje rolę właściciela produktu.

Ponieważ zespół nie ma wcześniejszego doświadczenia z metodyką Scrum, wynajęty zostanie niezależny konsultant jako mistrz Scrum i będzie odpowiedzialny za nauczenie zespołu wszystkiego odnoście technik i praktyk Scrum oraz za kontakty z zarządem i zapewnienie, że zespół nie wróci do swoich dawnych praktyk.

Projekt dostaje zielone światło na okres 12 miesięcy, a zespół marketingu zażyczył sobie, aby w tym okresie próbnym zawarte były cztery wersje produktu. Innymi słowy, pierwsza wersja musi być gotowa za trzy miesiące!

Zespoły funkcyjne

Duże, skomplikowane projekty muszą być dzielone pomiędzy wiele zespołów Scrum. Niektóre projekty są dzielone zgodnie z podziałami architektury systemu, co daje w efekcie zespoły zorientowane na komponenty, natomiast inni korzystają z zespołów funkcyjnych, które pracują nad grupami wymagań w wielu sprintach obejmujących komponenty dla różnych systemów – podejście to jest znane jako globalna własność kodu. Studium przypadku 8.5, „Scrum w środowisku technologii medycznych" jest przykładem tego podejścia.

W teorii zaletą tego podejścia jest to, że zespół funkcyjny pracuje nad wszystkimi bieżącymi wymaganiami i dlatego ma bardziej