



# SCRUM

PRAKTYCZNY PRZEWODNIK PO  
NAJPOPULARNIEJSZEJ METODYCE  
AGILE

---

KENNETH S. RUBIN



*Metodyki zwinne — Twój klucz do sukcesu!*

Tytuł oryginału: Essential Scrum: A Practical Guide to the Most Popular Agile Process

Tłumaczenie: Janusz Grabis

ISBN: 978-83-246-8076-4

Authorized translation from the English language edition, entitled: ESSENTIAL SCRUM: A PRACTICAL GUIDE TO THE MOST POPULAR AGILE PROCESS; ISBN 0137043295; by Kenneth S. Rubin; published by Pearson Education, Inc, publishing as Addison Wesley. Copyright © 2013 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Polish language edition published by HELION S.A. Copyright © 2013.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiekolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION  
ul. Kościuszki 1c, 44-100 GLIWICE  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto: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/scruma\\_ebook](http://helion.pl/user/opinie/scruma_ebook)  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Poleć książkę na Facebook.com](#)
- [Kup w wersji papierowej](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

## Pochwały dla książki

„Trenerzy zwinności — będącie zadowoleni z tej książki. Kenny Rubin stworzył dla nas niezbędną źródło wiedzy. Czy macie nad sobą menedżera, który zwyczajnie „nie chwyta, o co chodzi”? Dajcie mu do ręki tę książkę, poproście, żeby przeskoczył bezpośrednio do napisanego specjalnie dla menedżerów rozdziału trzeciego i poczytał o tym, dlaczego Scrum jest mniej ryzykownym podejściem od zarządzania sterowanego planem. Chcacie pomóc zespołom w kolektywnym zrozumieniu Scruma? Pomoże im w tym wizualny język ikon rozrzucony po całej książce. Są to zaledwie dwie cechy spośród wielu, które pomogą Ci trenować zespoły scrumowe — wykorzystaj je dobrze”.

— Lyssa Adkins, trener trenerów zwinności, Agile Coaching Institute. Autorka książki *Coaching Agile Teams*

„Jeden z najlepszych, najbardziej wszechstronnych opisów środowiska Scrum spośród dostępnych na rynku! *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* jest przeznaczone dla każdego — zaczynającego przygodę ze Scrumem lub doświadczonego w tym środowisku — kto jest zainteresowany najistotniejszymi aspektami tego procesu. Kenny w sposób doskonały wyizolował kluczowe elementy środowiska Scrum do przejrzystej treści — ilustrowanej łatwymi do zrozumienia rysunkami. Jako trener Scruma w licznych zespołach regularnie sięgam do tego materiału w poszukiwaniu nowych sposobów pomagania w poznawaniu i praktykowaniu środowiska. Przez lata miałem wielokrotnie okazję naocznie widzieć nieudolne, oparte na złej interpretacji próby wprowadzania Scruma przez wielkie korporacje. Lektura tej książki pomoże Ci powrócić do podstaw i skupić się na tym, co najważniejsze”.

— Joe Balistrieri, menedżer ds. rozwoju procesów, Rockwell Automation

„Kierownictwo IT w korporacjach, które bardzo opornie wprowadza metody zwinności, mogłoby niezwykle skorzystać dzięki wręczeniu egzemplarza tej książki każdemu menedżerowi odpowiedzialnemu za prowadzenie projektu lub wdrożenie. Kenny Rubin w swojej książce przedstawia w sposób pragmatyczny wszelkie możliwe materiały na temat przypadków i procesów biznesowych, których działa IT potrzebują do pomyślnego wprowadzenia Scruma”.

— John F. Bauer III, weteran na polu dostarczania rozwiązań technicznych w dużych korporacyjnych działach IT

„Niniejsza książka bez cienia wątpliwości ukazuje bogate doświadczenie Kenny'ego jako konsultanta, trenera i byłego dyrektora zarządzającego w Scrum Alliance. Po wprowadzeniu do podstaw Scruma książka odpowiada również na pytanie, które stawia sobie cała branża — jaką przyszłość mają menedżerowie projektów? *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* pozwala nam spojrzeć całościowo na środowisko oraz uczy, w jaki sposób osoby na stanowiskach kierowniczych mogą wspierać swoje zespoły scrumowe i angażować się w nie, aby pomyślnie przejść transformację organizacji na metody zwinne”.

— Sameer S. Bendre, CSM, PMP, starszy konsultant, 3i Infotech Inc.

„Jeśli jesteś nowicjuszem w zakresie metod zwinnych i Scruma, ta książka pozwoli Ci na przyspieszony start. Przykłady i opisy są przejrzyste i wyraziste. Będziesz miał okazję przekonać się, że kiedy tylko zadasz sobie jakieś pytanie, za chwilę dojdiesz do tej części książki, która opisuje interesujące Cię zagadnienie”.

— Johannes Brodwall, główny architekt rozwiązań, Steria Norway

„Doskonałe pod względem konstrukcji wyjaśnienia Kenny’ego cechują się przejrzystością rodem z języka Smalltalk — środowiska, w którym pracujemy od lat i które dało początek Scrumowi i programowaniu ekstremalnemu. Niniejsza książka zbiera razem zestaw zasad zinnego zarządzania, które trafiają w sedno i bez wątpienia pozwolą Ci pójść w kierunku bardziej efektywnego użycia zasad zwinności”.

— Rowan Bunning, założyciel Scrum WithStyle

„Obecnie pojawia się wiele książek poświęconych Scrumowi, ale ta rozpatruje temat pod zupełnie nowym kątem — konfrontacji praktyków oprogramowania z rzeczywistością. Kenny używa przykładów z życia i jasnych ilustracji, aby wskazać solidny fundament pomyślnej produkcji z wykorzystaniem metod zwinnych. Czytelnicy zrozumieją wartość budowania jakości oraz uświadomią sobie, że nie ma możliwości zrobienia wszystkiego dobrze za pierwszym razem. Musimy pracować w sposób przyrostowy i zdobywać wiedzę w trakcie. Książka ma w nazwie Scrum, ale w rzeczywistości promuje efektywne praktyki należące do szerszego świata metod zwinnych, aby pomóc odnieść sukces menedżerom i ich zespołom”.

— Lisa Crispin, współautorka *Testing Agile*

„Kenny’emu Rubinowi udało się napisać książkę, którą uważam za lekturę obowiązkową dla każdej osoby związanej z produkcją w Scrumie! Opisuje ona wszystko, co powinieneś wiedzieć na temat Scruma i nie tylko!”

— Scott Duncan, trener metod zwinności i Scruma

„Każdy, kto przeszedł szkolenie scrumowe lub jest członkiem zespołu scrumowego, uzna *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* za doskonałą kontynuację tematu. Książka w sposób szczegółowy opisuje, jak osiągnąć jeszcze większą zwinność poprzez implementację procesów scrumowych i wyjaśnia sposób podziału skomplikowanych projektów na zdolne do przeprowadzenia inicjatywy (lub „sprinty”). Kenny Rubin dostarcza mnóstwa istotnych studiów przypadków opisujących, jakie podejścia sprawdziły się lub nie sprawdziły się w różnorodnych organizacjach. Prosty układ oraz ilustracje w stylu biznesowym pozwalają na szybkie przeglądanie książki i znalezienie interesującej czytelnika treści. *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* będzie niezastąpionym przewodnikiem dla każdej organizacji, która pragnie przejść z tradycyjnego procesu kaskadowego w kierunku metod zwinnych”.

— Julia Frazier, menedżer produktu

„Produkcja oprogramowania nie należy do rzeczy łatwych — szczególnie jeśli nową metodologię pracy należy przyjąć podczas realizacji projektu. Ta książka pozwala uniknąć wielu pułapek, powiększyć zdolność zespołu do generowania wartości biznesowej i odnosić większe sukcesy w środowisku Scrum. Żałuję, że nie miałem takiej książki, kiedy sam zaczynałem swoją przygodę ze Scrumem”.

— Geir Hdemark, menedżer ds. produkcji, Basefarm AS

„Jestem przekonany o tym, że *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* stanie się podręcznikiem dla następnego pokolenia praktyków Scruma. Jest to nie tylko najbardziej wyczerpujące wprowadzenie do Scruma dostępne obecnie na rynku, ale również niezwykle dobrze napisane i przyjazne dla oczu — dzięki zastosowaniu nowego wizualnego języka ikon — opracowanie. Dodatkowo Kenny dzieli się swoimi własnymi ocenami i doświadczeniami będącymi z całą pewnością dobrym przykładem, z którego wszyscy możemy czerpać wiedzę”.

— Ilan Goldstein, menedżer rozwiązań zwinnych, Reed Elsevier

„Scrum jest ujmująco prosty, ale jednocześnie zaskakująco złożony. W książce *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* Kenny Rubin prowadzi nas krok po kroku przez te złożoności, zachowując jednak prosty przekaz. Doświadczenia pochodzące z rzeczywistości połączone z przejrzystymi ilustracjami pozwalają przyjrzeć się Scrumowi w jego prawdziwym wymiarze. Jeżeli jesteś menedżerem lub członkiem zespołu i zaczynasz pracę w Scrumie lub rozważasz wprowadzenie tego środowiska w swojej organizacji, musisz przeczytać tę książkę. Ja z całą pewnością zarekomenduję ją swoim studentom”.

— John Hebley, Hebley & Associates

„W *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* Kenny obdarowuje nas masą dobrych rad oraz wiedzą, udostępniając nam cenny wgląd w praktyczne zastosowanie metod zwinności i Scruma. Niezależnie od tego, czy dopiero zaczynasz przygodę ze zwinnością, czy też szukasz bardziej dojrzałego spojrzenia na ciągłe doskonalenie swojej organizacji, ta książka stanowi nieodzowny podręcznik w Twojej podręcznej bibliotece”.

— David Luzquiños, szef Agile Enablement, trener metod zwinności, Betfair

„Kenny Rubin w sposób pragmatyczny i przejrzysty wprowadza dogłębne spojrzenie na proces adaptacji metod zwinności. Z jednej strony prezentuje formalne (idealne) definicje Scruma, a z drugiej ich praktyczną implementację. W swojej najnowszej książce dzieli się z nami wiedzą zdobytą podczas prowadzenia swoich warsztatów, a także podczas długich lat pracy. Jeśli sam przymierzasz się do przejścia na metody zwinne lub szukasz kursu wprowadzającego, zaopatrz się w egzemplarz tej pozycji”.

— Cuan Mulligan, niezależny trener metod zwinności

„Dziesięć lat po publikacji pierwszych książek poświęconych Scrumowi nadeszła pora, aby połączyć fundamentalne zasady środowiska Scrum z praktycznymi doświadczeniami i podejściami stosowanymi w trakcie tej dekady. Kenny Rubin robi to dokładnie, w sposób satysfakcjonujący, pozbawiony dogmatyzmu. Czytelnik ma szansę zobaczyć pragmatyczne przedstawienie Scruma, a także nauczyć się, kiedy i w jaki sposób zastosować Scrum, aby osiągnąć pozytywne wyniki w biznesie”.

— dr Yves Stalgies, dyrektor działu IT, [www.etracker.com](http://www.etracker.com)

„Adaptacja Scruma wychodzi najlepiej, kiedy wszystkie osoby związane z procesem deweloperskim, nawet tylko peryferyjnie, dobrze rozumieją podstawy środowiska. *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* w sposób idealny opisuje zarówno ogólną perspektywę, jak i szczegóły Scruma, zachowując przystępny styl. Z całą pewnością książka ta stanie się standartowym źródłem odniesienia”.

— Kevin Tureski, dyrektor Devin Tureski Consulting



*Dla mojej żony, Jenine, za całe Twoje wsparcie z miłością.*

*Dla moich synów, Jonah i Ashera, za inspirowanie mnie.*

*Dla mojego ojca, Manny'ego, za przekazanie mi wartości ciężkiej pracy.*

*Dla mojej matki, Joyce, za pokazanie mi, jak wygląda prawdziwa odwaga  
(niech pamięta o niej będzie błogosławieństwem).*



# SPIS TREŚCI

---

<b>Słowo wstępne — Mike Cohn .....</b>	<b>21</b>
<b>Słowo wstępne — Ron Jeffries .....</b>	<b>23</b>
<b>Wstęp .....</b>	<b>25</b>
<b>Rozdział 1 Wprowadzenie .....</b>	<b>33</b>
Czym jest Scrum? .....	34
Początki Scruma .....	35
Dlaczego Scrum? .....	36
Wyniki Genomiki .....	37
Czy Scrum może pomóc Tobie? .....	37
Domena złożona .....	40
Domena skomplikowana .....	40
Domena prostoty .....	40
Domena chaosu .....	41
Nieporządek .....	41
Praca sterowana przerwaniami .....	41
Zakończenie .....	42
<b>Część I POJĘCIA PODSTAWOWE</b>	
<b>Rozdział 2 Środowisko Scrum .....</b>	<b>45</b>
Wprowadzenie .....	45
Role w Scrumie .....	46
Właściciel produktu .....	47
Mistrz młyna .....	47
Zespół deweloperski .....	48
Aktywności i artefakty Scruma .....	48
Rejestr produktu .....	50
Sprinty .....	52
Planowanie sprintu .....	52

Wykonanie sprintu .....	54
Codzienne działania scrumowe .....	55
Koniec pracy .....	56
Przegląd sprintu .....	57
Retrospekcja sprintu .....	58
Zakończenie .....	59
<b>Rozdział 3 Zasady zwinności .....</b>	<b>61</b>
Wprowadzenie .....	61
Zmienna i niepewność .....	64
Wspieraj pomocną zmienność .....	64
Stosuj budowanie w sposób iteracyjny i przyrostowy .....	65
Wykorzystuj zmienność poprzez inspekcję, adaptację i przejrzystość .....	66
Redukuj wszystkie formy niepewności jednocześnie .....	67
Przewidywanie i adaptacja .....	68
Pozostaw wszystkie opcje otwarte .....	69
Pogódź się z tym, że nie wszystko da się przewidzieć z góry .....	69
Preferuj podejście odkrywcze i adaptacyjne .....	71
Wspieraj zmiany w sposób uzasadniony ekonomicznie .....	72
Równoważ pracę, którą można przewidzieć z góry, z pracą adaptacyjną w samą porę .....	74
Wiedza potwierdzona .....	75
Potwierdź bezzwłocznie istotne założenia .....	76
Wykorzystuj różne konkurencyjne ścieżki pozyskiwania wiedzy .....	76
Organizuj przepływ pracy umożliwiający szybkie pozyskiwanie informacji zwrotnej .....	77
Praca cząstkowa .....	78
Stosuj rozsądne ekonomicznie rozmiary zapotrzebowania .....	79
Zrób rozpoznanie inventarza i zarządzaj nim w celu dobrego przepływu .....	80
Skup się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę .....	81
Bierz pod uwagę koszt opóźnień .....	83
Postęp .....	84
Zaadaptuj się w oparciu o napływające informacje i zmodyfikuj plan .....	84
Mierz postęp poprzez ocenę działających rzeczy .....	84
Skup się na dostarczaniu wartości .....	85
Wydajność .....	86
Działaj szybko, ale nie w pośpiechu .....	86
Buduj z zachowaniem jakości .....	86
Stosuj minimalną niezbędną ilość ceremonii .....	87
Zakończenie .....	88
<b>Rozdział 4 Sprinty .....</b>	<b>91</b>
Wprowadzenie .....	91
Ograniczenie czasowe .....	92
Ograniczenie pracy cząstkowej .....	92
Wymuszanie priorytetów .....	93
Demonstrowanie postępów .....	93
Unikanie zbędnego perfekcjonizmu .....	93
Motywowanie domykania .....	94
Poprawianie przewidywalności .....	94

Krótki okres trwania .....	94
Łatwość planowania .....	94
Szybka informacja zwrotna .....	95
Lepsze zyski z inwestycji .....	95
Ograniczanie błędów .....	95
Rozbudzenie podekscytowania .....	95
Regularne punkty kontrolne .....	96
Stały czas trwania .....	97
Zalety taktowania .....	97
Uproszczone planowanie .....	98
Niezmienność celu .....	98
Czym jest cel sprintu? .....	99
Wzajemne zobowiązanie .....	99
Zmiana kontra doprecyzowanie .....	99
Konsekwencje zmiany .....	100
Pragmatyczność .....	101
Zakończenie przed czasem .....	102
Definicja ukończenia .....	103
Czym jest definicja ukończenia? .....	104
Definicja ukończenia może ewoluować .....	106
Definicja ukończenia kontra kryteria akceptacji .....	107
Ukończony kontra rzeczywiście ukończony .....	107
Zakończenie .....	108
<b>Rozdział 5 Wymagania i historyjki użytkownika .....</b>	<b>109</b>
Wprowadzenie .....	109
Wykorzystanie dyskusji .....	111
Stopniowe udoskonalanie .....	112
Czym są historyjki użytkownika?	113
Karta .....	113
Rozmowa .....	114
Potwierdzenie .....	115
Poziom szczegółowości .....	116
Inwestuj w dobre historyjki .....	118
Niezależność .....	118
Negocjalność .....	118
Wartościowość .....	120
Ocenialność .....	121
Dobry (mały) rozmiar .....	121
Testowalność .....	122
Wymagania niefunkcjonalne .....	122
Historyjki pozyskiwania wiedzy .....	123
Zbieranie historyjek .....	124
Warsztaty pisania historyjek .....	125
Mapa historyjek .....	125
Zakończenie .....	127

<b>Rozdział 6 Rejestr produktu .....</b>	<b>129</b>
Wprowadzenie .....	129
Elementy rejestru produktu .....	129
Cechy dobrego rejestru produktu .....	131
Odpowiedni stopień uszczegółowienia .....	131
Emergencja .....	132
Przypisanie ocen .....	132
Przypisanie priorytetów .....	133
Pielegnacja .....	134
Czym jest pielegnacja? .....	134
Kto pielegnuje rejestr? .....	135
Kiedy odbywa się pielegnacja? .....	136
Definicja gotowości .....	138
Zarządzanie przepływem .....	139
Zarządzanie przepływem wersji dystrybucyjnych .....	139
Zarządzanie przepływem sprintu .....	140
Jakie rejesty produkту i w jakiej liczbie? .....	142
Czym jest produkt? .....	142
Duże produkty — hierarchiczne rejesty produktu .....	143
Wiele zespołów — jeden rejestr produktu .....	144
Jeden zespół — wiele produktów .....	145
Zakończenie .....	146
<b>Rozdział 7 Nadawanie ocen i prędkość .....</b>	<b>147</b>
Wprowadzenie .....	147
Co i kiedy oceniamy .....	148
Ocena elementów z rejestru portfela .....	149
Ocena elementów z rejestru produktu .....	149
Ocena zadań .....	150
Koncepcje nadawania ocen elementom rejestru produktu .....	150
Nadawanie ocen jako zespół .....	151
Oceny nie są zobowiązaniem .....	152
Dokładność kontra precyzja .....	153
Ocenianie w sposób względny .....	153
Jednostki służące do oceny .....	155
Punkty historyjkowe .....	155
Idealne dni .....	156
Planowanie pokerowe .....	156
Skala ocen .....	157
Zasady gry .....	158
Zalety .....	160
Czym jest prędkość? .....	160
Obliczanie przedziału prędkości .....	161
Prognozowanie prędkości .....	162
Wpływanie na prędkość .....	163
Nieprawidłowe korzystanie z prędkości .....	164
Zakończenie .....	165

---

<b>Rozdział 8 Dług techniczny .....</b>	<b>167</b>
Wprowadzenie .....	167
Konsekwencje dłużu technicznego .....	169
Nieprzewidywalny punkt przegięcia .....	169
Wzrost czasu potrzebnego do dostarczenia wersji .....	169
Znaczna liczba błędów .....	170
Rosnące koszty produkcji i wsparcia .....	170
Umieranie produktu .....	171
Spadek przewidywalności .....	171
Działanie poniżej optymalnej wydajności .....	172
Powszechna frustracja .....	172
Spadek satysfakcji klientów .....	172
Przyczyny dłużu technicznego .....	172
Presja nieprzekroczenia terminu końcowego .....	173
Próby sztucznego zwiększenia prędkości .....	173
Mit: Rezygnacja z testowania zwiększa prędkość .....	174
Dług narasta na dłużu już istniejącym .....	174
Długiem technicznym trzeba zarządzać .....	176
Zarządzanie przyrostem dłużu technicznego .....	177
Stosowanie dobrych praktyk technicznych .....	177
Solidna definicja ukończenia .....	178
Prawidłowe rozumienie ekonomii dłużu technicznego .....	178
Ujawnianie dłużu technicznego .....	180
Ujawniaj dług techniczny na poziomie biznesowym .....	181
Ujawniaj dług techniczny na poziomie inżynierijnym .....	182
Obsługa dłużu technicznego .....	183
Nie każdy dług techniczny powinien być spłacany .....	184
Zastosuj metodę skauta (obsłuż dług, kiedy go napotkasz) .....	186
Spłacaj dług techniczny przyrostowo .....	186
Spłacaj w pierwszej kolejności dług o najwyższych odsetkach .....	187
Spłacaj dług techniczny podczas wykonywania pracy mającej wartość dla klienta .....	187
Zakończenie .....	189

## Część II ROLE

<b>Rozdział 9 Właściciel produktu .....</b>	<b>193</b>
Wprowadzenie .....	193
Główne obowiązki .....	194
Zarządzanie ekonomicą .....	195
Udział w planowaniu .....	196
Pielęgnacja rejestru produktu .....	197
Definiowanie kryteriów akceptacji i weryfikowanie, iż zostały one spełnione .....	197
Współpraca z zespołem deweloperskim .....	198
Współpraca z interesariuszami .....	199
Cechy i umiejętności .....	199
Umiejętności domenowe .....	199
Umiejętności interpersonalne .....	200

Podejmowanie decyzji .....	201
Odpowiedzialność .....	201
Dzień z życia właściciela produktu .....	202
Kto powinien być właścicielem produktu? .....	204
Wewnętrzne prace deweloperskie .....	205
Komercyjne prace deweloperskie .....	205
Outsourcing .....	207
Tworzenie komponentów .....	208
Właściciel produktu pełniący inne role .....	208
Zespół właścicieli produktu .....	209
Pełnomocnicy właściciela produktu .....	210
Szef właścicieli produktu .....	211
Zakończenie .....	211
<b>Rozdział 10 Mistrz młyna .....</b>	<b>213</b>
Wprowadzenie .....	213
Główne obowiązki .....	213
Trener .....	213
Lider służby .....	214
Autorytet w dziedzinie procesu .....	215
Ochrona przed zakłóceniami .....	215
Usuwanie przeszkód .....	215
Agent zmiany .....	215
Cechy i umiejętności .....	216
Wiedza .....	216
Umiejętność stawiania pytań .....	216
Cierpliwość .....	217
Współpraca .....	217
Proaktywność .....	217
Przezroczystość .....	217
Dzień z życia .....	218
Wypełnianie roli .....	219
Kto powinien być mistrzem młyna? .....	219
Czy mistrz młyna to zajęcie na pełny etat .....	220
Połączenie roli mistrza młyna z innymi rolami .....	220
Zakończenie .....	221
<b>Rozdział 11 Zespół deweloperski .....</b>	<b>223</b>
Wprowadzenie .....	223
Zespoły o specyficznej roli .....	223
Główne obowiązki .....	224
Wykonanie sprintu .....	225
Codzienna inspekcja i adaptacja .....	225
Pielęgnacja rejestru produktu .....	225
Planowanie sprintu .....	225
Inspekcja i adaptacja produktu oraz procesu .....	225

---

Cechy i umiejętności .....	226
Samoorganizacja .....	226
Różnorodny pod względem umiejętności i samowystarczalny .....	228
Umiejętności typu T .....	229
Postawa muszkieterów .....	231
Komunikacja szerokopasmowa .....	232
Przezroczysta komunikacja .....	233
Prawidłowy rozmiar .....	233
Skupienie i poświęcenie .....	234
Praca w podtrzymywальnym tempie .....	236
Długotrwałość .....	237
Zakończenie .....	238
<b>Rozdział 12 Budowa zespołów scrumowych .....</b>	<b>239</b>
Wprowadzenie .....	239
Zespoły budujące cechy kontra zespoły budujące komponenty .....	240
Koordynacja wielu zespołów .....	244
Scrum scrumów .....	244
Pociąg wersji dystrybucyjnych .....	246
Zakończenie .....	249
<b>Rozdział 13 Menedżerowie .....</b>	<b>251</b>
Wprowadzenie .....	251
Kształtowanie zespołów .....	253
Definiowanie granic .....	253
Dostarczanie jasnego motywującego celu .....	254
Formowanie zespołów .....	254
Zmiana kompozycji zespołów .....	255
Upoważnianie zespołów .....	256
Pielegnacja zespołów .....	257
Motywowanie ludzi .....	257
Rozwijanie umiejętności .....	257
Przewodzenie obszarowi funkcjonalności .....	258
Utrzymywanie integralności zespołu .....	258
Przystosowywanie i adaptowanie środowiska .....	259
Promowanie wartości zwinności .....	259
Usuwanie przeszkód organizacyjnych .....	259
Przystosowywanie grup wewnętrznych .....	260
Przystosowywanie partnerów .....	260
Zarządzanie przepływem wytwarzania wartości .....	261
Spoglądanie z perspektywy systemu .....	261
Zarządzanie ekonomią .....	261
Obserwacja pomiarów i raportowanie .....	261
Menedżerowie projektów .....	262
Obowiązki menedżera projektu w zespole scrumowym .....	263
Pozostanie przy roli niezależnego menedżera projektu .....	264
Zakończenie .....	267

**CZĘŚĆ III PLANOWANIE**

<b>Rozdział 14 Zasady planowania w Scrumie .....</b>	<b>273</b>
Wprowadzenie .....	273
Nie zakładaj, że z góry uda Ci się wszystko dobrze zaplanować .....	273
Planowanie z góry powinno być pomocne, ale nie przesadne .....	274
Pozostaw opcje planowania otwartymi do ostatniego rozsądniego momentu .....	275
Skup się bardziej na adaptowaniu i ponownym planowaniu	
niż na trzymaniu się już ustalonego planu .....	275
Prawidłowo zarządzaj inventarzem pracy .....	277
Preferuj mniejsze i częstsze wersje dystrybucyjne .....	278
Planuj szybki proces nauki i wykonywanie zwrotów, kiedy jest to niezbędne .....	280
Zakończenie .....	280
<b>Rozdział 15 Planowanie wielopoziomowe .....</b>	<b>281</b>
Wprowadzenie .....	281
Planowanie portfela .....	283
Planowanie produktu (tworzenie wizji produktu)	
Wizja .....	283
Rejestr produktu wysokiego poziomu .....	283
Mapa drogowa produktu .....	284
Planowanie wersji dystrybucyjnej .....	285
Planowanie sprintu .....	287
Planowanie codzienne .....	287
Zakończenie .....	288
<b>Rozdział 16 Planowanie portfela .....</b>	<b>291</b>
Wprowadzenie .....	291
Czas .....	291
Uczestnicy .....	292
Proces .....	292
Strategie tworzenia harmonogramu .....	293
Optymalizacja pod względem zysków z cyklu życia .....	294
Wyznaczanie kosztu opóźnienia .....	295
Staraj się obliczać dokładnie, ale nie precyzyjnie .....	298
Strategie napływu .....	299
Zastosowanie filtra ekonomicznego .....	299
Równoważ tempo przybywania z tempem ubywania .....	300
Szybko wykorzystuj nadarzające się okazje .....	302
Planuj mniejsze i częstsze wersje dystrybucyjne .....	303
Strategie odpływu .....	304
Skup się na pracy czekającej na realizację, nie na pracownikach czekających na pracę .....	304
Ustal limit pracy częściowej .....	305
Zaczekaj na cały zespół .....	306
Strategie aktywności .....	306
Użycie ekonomii końcowej .....	307
Zakończenie .....	308

<b>Rozdział 17 Planowanie produktu (tworzenie wizji produktu) .....</b>	<b>309</b>
Wprowadzenie .....	309
Czas .....	310
Uczestnicy .....	310
Proces .....	312
Przykład ZODC .....	312
Tworzenie wizji .....	313
Tworzenie rejestru produktu wysokiego poziomu .....	316
Definiowanie mapy drogowej produktu .....	317
Inne aktywności .....	319
Przeprowadzanie wizji w sposób ekonomiczny .....	321
Celuj w realistyczny poziom ufności .....	322
Skup się na krótkim horyzoncie .....	323
Działaj szybko .....	324
Płać za wiedzę potwierdzoną .....	324
Stosuj przyrostowe lub prowizoryczne finansowanie .....	325
Ucz się szybko i wykonuj zwroty (czyli stosuj strategię „szybkiej porażki”) .....	326
Zakończenie .....	327
<b>Rozdział 18 Planowanie wersji dystrybucyjnej (planowanie długoterminowe) .....</b>	<b>329</b>
Wprowadzenie .....	329
Czas .....	330
Uczestnicy .....	331
Proces .....	331
Ograniczenia wersji dystrybucyjnej .....	333
Wszystko sztywne .....	333
Ustalony zakres i czas .....	334
Ustalony zakres .....	335
Ustalony czas .....	335
Zmienna jakość .....	336
Uaktualnianie ograniczeń .....	336
Pielegnacja rejestru produktu .....	337
Definiowanie minimalnego zestawu cech kwalifikującego do dystrybucji .....	338
Tworzenie mapy sprintów (umieszczanie elementów rejestru produktu w slotach) .....	339
Planowanie wersji z ustaloną datą .....	341
Planowanie wersji z ustalonym zakresem .....	345
Obliczanie kosztów .....	347
Komunikacja .....	348
Komunikowanie postępów dla wersji z ustalonym zakresem .....	348
Komunikowanie postępów dla wersji z ustaloną datą .....	350
Zakończenie .....	351

## CZĘŚĆ IV WYKONYWANIE SPRINTÓW

<b>Rozdział 19 Planowanie sprintu .....</b>	<b>355</b>
Wprowadzenie .....	355
Czas .....	355
Uczestnicy .....	356
Proces .....	356
Podejścia do planowania sprintu .....	358
Dwuczęściowe planowanie sprintu .....	358
Jednoczęściowe planowanie sprintu .....	359
Określanie pojemności .....	360
Czym jest pojemność? .....	360
Pojemność w punktach historyjkowych .....	361
Pojemność w roboczogodzinach .....	362
Wybieranie elementów rejestru produktu .....	363
Nabieranie pewności .....	364
Doprecyzywianie celu sprintu .....	365
Ostateczne ustalenie zobowiązania .....	365
Zakończenie .....	366
<b>Rozdział 20 Wykonanie sprintu .....</b>	<b>367</b>
Wprowadzenie .....	367
Czas .....	367
Uczestnicy .....	367
Proces .....	368
Planowanie wykonania sprintu .....	369
Zarządzanie przepływem .....	369
Praca równoległa i działanie w kopcu .....	370
Która pracę rozpocząć? .....	372
Jak zorganizować pracę nad zadaniami? .....	372
Jaką pracę trzeba zrealizować? .....	373
Kto wykonuje pracę? .....	373
Codzienne działania scrumowe .....	374
Realizacja zadań — praktyki techniczne .....	374
Komunikowanie .....	375
Tablica zadań .....	375
Wykres spalania sprintu .....	376
Wykres rozpalania sprintu .....	379
Zakończenie .....	380
<b>Rozdział 21 Przegląd sprintu .....</b>	<b>381</b>
Wprowadzenie .....	381
Uczestnicy .....	382
Przygotowanie .....	383
Wskazanie, kogo zaprosić na przegląd .....	384
Ustalenie harmonogramu .....	384

---

Potwierdzenie, że praca w sprintie została zrealizowana .....	385
Przygotowanie do demonstracji .....	386
Wyznaczenie, kto co robi .....	386
<b>Podejście .....</b>	<b>386</b>
Podsumowanie .....	387
Demonstracja .....	388
Dyskusja .....	388
Adaptacja .....	389
Problemy przeglądu sprintu .....	389
Zatwierdzanie .....	390
Mała frekwencja .....	390
Duże projekty .....	391
Zakończenie .....	391
<b>Rozdział 22 Retrospekcja sprintu .....</b>	<b>393</b>
Wprowadzenie .....	393
Uczestnicy .....	395
Przygotowania .....	396
Zdefiniowanie punktu skupienia retrospekcji .....	396
Wybór ćwiczeń .....	397
Zebranie obiektywnych danych .....	397
Dopasowanie szczegółów retrospekcji .....	398
Podejście .....	398
Przygotowanie odpowiedniej atmosfery .....	400
Ustalenie wspólnego kontekstu .....	400
Identyfikacja spostrzeżeń .....	403
Wskazanie działań .....	405
Zakończenie retrospekcji .....	408
Wprowadzenie zmian w życie .....	408
Problemy retrospekcji sprintu .....	408
Zakończenie .....	411
<b>Rozdział 23 Co dalej? .....</b>	<b>413</b>
Nie ma stanu końcowego .....	413
Odkryj swoją własną ścieżkę .....	414
Stosowanie najlepszych praktyk .....	414
Używanie Scruma do odkrywania ścieżki naprzód .....	415
Naprzód! .....	416
<b>Słowniczek .....</b>	<b>419</b>
<b>Bibliografia .....</b>	<b>443</b>
<b>Skorowidz .....</b>	<b>447</b>



# Słowo wstępne

MIKE COHN

Zjadłem dzisiaj obiad w Burger Kingu. Zauważyłem na ścianie napis „Dom Whoppera” i wyjaśnienie, iż istnieje ponad milion różnych sposobów zamawiania tego dania. Jeżeli różnorodne kombinacje dodatkowych ogórków, pomidorów, sałaty, sera i tak dalej pozwalają na ponad milion kombinacji przyrządzenia hamburgera, muszą istnieć miliardy sposobów implementacji Scruma. I chociaż nie istnieje jedna słuszna metoda, są lepsze i gorsze sposoby jego realizacji.

W książce *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* Kenny Rubin pomaga czytelnikom znaleźć te lepsze metody. Jego książka nie stanowi przepisu — nie ma tutaj stwierdzeń w stylu „Musisz zrobić to”. Zamiast tego autor uczy kluczowych reguł stanowiących podstawę sukcesu Scruma, a następnie daje nam wybór, jak chcemy je zrealizować. Dla przykładu: nie istnieje jedyna słuszna metoda planowania sprintu dla wszystkich zespołów scrumowych. To, co działa w danej firmie lub projekcie, nie sprawdzi się w innych. Dlatego też Kenny daje nam wybór — opisuje ogólny powód planowania sprintów przez zespoły scrumowe, wyjaśnia, co powinno być wynikiem takiego planowania, a następnie podaje kilka alternatywnych sposobów, które sprawdzą się w działaniu. Ostateczna decyzja należy do każdego zespołu — na szczęście mają teraz tę książkę do pomocy.

Niespodziewanym wsparciem, jakie pojawiło się w książce *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile*, jest wprowadzony przez Kenny'ego język wizualny przekazujący wiedzę o Scrumie. Dla mnie osobiście ilustracje okazały się bardzo pomocne podczas czytania tekstu — przypuszczam, że staną się one czymś powszechnym podczas przyszłych dyskusji na temat Scruma.

Świat potrzebował tej książki od dłuższego czasu. Scrum zaczął swoje życie jako prosta koncepcja. Pierwsza książka na ten temat — *Wicked Problems, Righteous Solutions*, napisana w 1990 roku przez Petera DeGrace'a i Leslie'ego Stahla, składała się z zaledwie sześciu stron. Jednak w ciągu ponad dwudziestu lat od tego momentu pojawiały się inne książki, a Scrum rozrastał się. Wprowadzone zostały nowe role, typy spotkań, a artefakty uległy uszczegółowieniu. Z każdym nowym elementem rosło zagrożenie zagubienia sedna Scruma, tej części, która mówi o zespołowym planowaniu rozwiązań, decydowaniu o sposobie wykonania małej części całego rozwiązania, a następnie przeanalizowania tego, co zrobili członkowie zespołu oraz jak dokonali tego wspólnie.

W tej książce Kenny prowadzi nas z powrotem do serca Scruma. W tym miejscu zespoły mogą podjąć decyzje niezbędne do wprowadzenia Scruma na ich własny sposób. Jest to niezbędny przewodnik, pomagający zespołom wybrać jeden spośród miliardów możliwych sposobów implementacji Scruma, który zapewni im powodzenie.

— Mike Cohn

Autor książek *Succeeding with Agile*, *Agile Estimating and Planning* oraz *User Stories Applied*.  
[www.mountaingoatsoftware.com](http://www.mountaingoatsoftware.com).

# Słowo wstępne

RON JEFFRIES

---

Kiedy Kenny poprosił mnie o napisanie słowa wstępnego do *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile*, pomyślałem sobie „To będzie szybka i łatwa rzecz — ta książka musi być krótka i prowadzić bezpośrednio do prostego opisu tego, czym jest Scrum”. Znałem już wcześniej pracę Kenny’ego, więc wiedziałem, że to będzie dobra i krótka lektura. Czego lepszego mogłem się spodziewać?

Wyobraźcie sobie moje zaskoczenie i zachwyt, kiedy przekonałem się, że ta książka opisuje niemal wszystko, co należy wiedzieć o Scrumie w pierwszym dniu lub w pierwszych latach stosowania Scruma. Kenny nie poprzestaje jednak na tym. Zaczyna od fundamentalnych idei, włączając w to zasady zwinności (agile), które stoją u podstaw wszystkich zwinnych metod i szybkiego przeglądu środowiska Scrum. Następnie zaczyna wkraczać coraz głębiej i głębiej. Jest to nadal bardzo dobra i całkiem wyczerpująca lektura.

Kenny opisuje w dużych szczegółach planowanie, przygląda się wymaganiom, historyjkom, rejestrowi, estymacjom, prędkości. Następnie zagłębia się mocniej w zasady i pomaga rozprawić się z wszystkimi poziomami planowania oraz wszystkimi horyzontami czasowymi. Opisuje, jak sprinty są planowane, wykonywane, przeglądane, a następnie ulepszane — cały czas wspiera nas czymś więcej niż tylko podstawy, uwypuklając kluczowe problemy, jakich możesz się spodziewać na swojej drodze.

Ja w odniesieniu do Scruma i metod zwinnych skupiam się na niezbędnych umiejętnościach programisty, aby upewnić się, że zespoły są w stanie dostarczyć w każdym kolejnym sprincie prawdziwe, działające, biznesowe oprogramowanie. Kenny pomaga nam zrozumieć, w jaki sposób dobrze i bezpiecznie korzystać z takich idei jak prędkość i dług techniczny. Oba te zagadnienia mają kluczowe znaczenie — polecam je Twojej uwadze.

Prędkość mówi nam, jak wiele zespół jest w stanie dostarczyć wraz z upływem czasu. Na jej podstawie możemy odczuć, ile udaje nam się zrobić i czy poprawiamy się pod tym względem. Kenny ostrzega nas jednak, że stosowanie prędkości do oceny wydajności ma szkodliwy wpływ na wyniki biznesowe, i wyjaśnia, dlaczego tak się dzieje.

Dług techniczny stał się bardzo ogólnym pojęciem, odnoszącym się niemal do wszystkiego, co może być popsułe w kodzie. Kenny pomaga nam rozdzielić dług techniczny na różne kategorie i zrozumieć, dlaczego powinniśmy zwracać uwagę na te na pozór techniczne detale. Podoba mi się jego wyjaśnienie wpływu nacisków wywieranych na zespół na nieuniknione niszczące skutki dla biznesu, przekreślające szanse na wytworzenie dobrego produktu we właściwym czasie.

Scrum podobnie jak inne metody zwinne bazuje na podejściu odkrywczym z bardzo szybką pętlą zwrotną. Kenny opowiada swoją historię krótkiego okresu używania kart perforowanych, co przypomina mi moje najwcześniejsze doświadczenia z komputerami, mające miejsce na wiele lat wcześniej, zanim Kenny zobaczył na oczy swoją pierwszą kartę perforowaną.

Jako student miałem dostatecznie dużo szczęścia, aby dostać pracę stażysty w kwaterze głównej strategicznego dowództwa lotnictwa w Omaha. W tamtych czasach wszelkie prace informatyczne wykonywane były przy użyciu kart perforowanych. Moje karty były wysyłane kilka pięter w dół do podziemi kwatery dowództwa i wykonywane na komputerze istniejącym na potrzeby wojny, gdyby taka miała nastąpić. Miałem możliwość puszczania swoich programów raz lub dwa razy na dzień.

Po otrzymaniu odpowiedniej przepustki byłem w stanie zjechać na dół nawet w środku nocy, a następnie zagadać sierżanta Whittakera i przekonać go, aby pozwolił mi puścić na komputerze moje własne programy, siedząc przed konsolą — tak, na maszynie, której głównym zadaniem było przeprowadzenie ewentualnego ataku nuklearnego. Na szczęście czerwony przycisk znajdował się w innym pomieszczeniu.

Pracując bezpośrednio na maszynie, byłem w stanie wykonać dziesięciokrotnie więcej pracy w porównaniu z wysłaniem moich kart na dół i czekaniem na powrót wydruków z wynikami. Znacznie szybciej otrzymywałem informację zwrotną, szybciej się uczyłem i dzięki temu moje programy znacznie wcześniej zaczynały wykonywać oczekiwany pracę.

Właśnie na tym polega Scrum. Zamiast czekać miesiącami, a nawet latami, aby przekonać się o faktycznym działaniu programu, w Scrumie można to zrobić co kilka tygodni. Właściciel produktu z naprawdę dobrym zespołem będzie widział kształtowanie się nowych cech w przeciągu dni!

I właśnie o tym jest książka Kenny'ego. Jeśli Scrum jest dla Ciebie nowością, przeczytaj ją od deski do deski, a następnie trzymaj w pobliżu. Jeśli praktykujesz już Scrum od jakiegoś czasu, przejrzyj ją i również trzymaj pod ręką.

Kiedy nadjejdzie taki moment, że zaczynasz się zastanawiać nad tym, co się dzieje z Twoim zespołem, lub będziesz chciał spróbować czegoś nowego, wróć do lektury i zaczniąj szukać. Istnieją spore szanse na to, że znajdziesz coś wartościowego.

— Ron Jeffries

# WSTĘP

---

Ta książka omawia esencję Scruma — rzeczy, które musisz znać, jeśli masz zamiar odnieść sukces, stosując Scrum do wytwarzania innowacyjnych produktów i usług.

## Czym jest Scrum?

Scrum bazuje na małym zbiorze fundamentalnych **wartości, zasad i praktyk** (zwanych wspólnie **środowiskiem Scrum**). Organizacje używające Scruma powinny uwzględniać jego środowisko całościowo, być może nie od razu we wszystkich swoich wydziałach, ale zdecydowanie w pierwszych zespołach, które będą używać Scruma. Całościowa akceptacja zasad środowiska Scrum nie oznacza jednak, że organizacje muszą wdrażać jedną, uniwersalną formułę dla wszystkich. Chodzi raczej o pozostanie w zgodzie z zasadami środowiska Scrum przy jednoczesnym doborze „mieszanki” **podejść** do własnych implementacji Scruma.

*Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* łączy wartości, zasady i praktyki Scruma ze zbiorem wypróbowanych i pewnych podejść zgodnych ze środowiskiem Scrum, ale nie nakazanych przez nie. Niektóre z tych podejść będą odpowiednie dla Twojej sytuacji, inne nie. Każde podejście będzie musiało być zaadaptowane do Twoich konkretnych okoliczności.

## Geneza powstania książki

Będąc trenerem Scruma i ogólnie programowania zwinnego, jestem często pytany o dobrą pozycję książkową opisującą Scrum — taką, która w sposób zrozumiały opisuje środowisko Scrum i przedstawia najbardziej popularne podejścia do jego zastosowania w praktyce. Ponieważ nie byłem w stanie znaleźć pojedynczej książki opisującej tę tematykę w stopniu dostatecznie szczegółowym, aby mogła być ona użyteczna dla współczesnych praktykantów Scruma, rekomendowałem listę pozycji: kilka, które opisują środowisko Scrum, ale są już przestarzałe lub niekompletne, kilka wysoko cenionych książek na temat zwinności, które nie koncentrują się wyłącznie na Scrumie, i kilka skupionych na specyficznych aspektach lub podejściach Scruma, ale pozbawionych całościowego opisu

środowiska Scrum. To bardzo dużo książek jak dla kogoś, kto chciałby mieć wyłącznie jeden podręcznik opisujący esencję Scruma!

Twórcy Scruma (Jeff Sutherland i Ken Schwaber) mają swoją publikację opisującą Scrum pod tytułem „Przewodnik po Scrumie”<sup>1</sup>. Ten krótki dokument (około 15 stron) jest przedstawiany przez autorów jako „podręcznik rozstrzygający zasady Scruma i dokumentujący Scrum” [Schwaber i Sutherland, 2011]. Autorzy przyrównują swój dokument do reguł gry w szachy, „opisując, w jaki sposób poruszają się poszczególne figury, jak wygląda kolejność gry, kiedy dochodzi do wygranej itd.”. Choć dokument ten daje ogólny pogląd na Scruma i jego zasady, nie został on jednak zaprojektowany jako wyczerpujące źródło podstawowej wiedzy na temat Scruma. Rozwijając analogię podaną przez autorów, danie nowemu zespołowi scrumowemu „Przewodnika po Scrumie” i oczekiwanie od niego rozsądnych wyników byłoby jak danie 15-stronicowego opisu zasad gry początkującemu szachiście i oczekивание, że po jego przeczytaniu przeprowadzi mecz szachowy na dobrym poziomie. Ten pojedynczy dokument zwyczajnie nie wystarczy.

Niniejsza książka, *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile*, jest próbą stworzenia tego pojedynczego, brakującego źródła kluczowej wiedzy na temat Scruma. Zawiera ona dogłębną analizę zasad, wartości i praktyk Scruma i w większości przypadków zgadza się z innymi liderami idei zwinności oraz samym „Przewodnikiem po Scrumie”. (Miejsca, w których książka odbiega myślą od ogólnie przyjętych poglądów, zostały przeze mnie oznaczone i odpowiednio opisane). Ta książka opisuje również podejścia zgodne ze środowiskiem Scrum, które były z powodzeniem stosowane przeze mnie i zespoły, które trenowałem. Moją intencją nie jest zastępowanie tą książką innych pozycji opisujących w sposób przekrojowy wybrane praktyki lub podejścia scrumowe. Takie książki uzupełniają i rozszerzają zawartą tutaj wiedzę. Proponuję raczej, abyś traktował *Scrum. Praktyczny przewodnik po najpopularniejszej metodyce Agile* jako punkt startowy na drodze do używania Scruma w celu zadziwienia swoich klientów.

## Dla kogo jest ta książka

Książka ta będzie stanowić odświeżenie, a być może również dalsze wyjaśnienie zagadnień, jakie przedyskutowałem z tysiącami ludzi, którzy brali udział w moich zajęciach zatytulowanych „Praca w zespole scrumowym”, „Certyfikowany mistrz młyna” i „Certyfikowany właściciel produktu” lub byli członkami trenowanych przeze mnie zespołów scrumowych. Natomiast dla jeszcze większej rzeczy ludzi, z którymi nie miałem jeszcze okazji pracować, ta książka będzie pierwszym wprowadzeniem do Scruma i zwinności lub szansą spojrzenia na Scrum w innym świetle i być może po prawieniu własnego sposobu realizacji zasad tego środowiska.

Książka nie została napisana z myślą o jakiejkolwiek konkretnej roli — to nie jest pozycja przeznaczona dla właścicieli produktu, mistrzów młyna czy też członków zespołu deweloperskiego. Przeciwnie, jest ona przeznaczona dla każdego, kto zaangażowany jest w Scrum, poczynając od członków samego zespołu, a kończąc na tych, którzy współpracują z nimi wewnętrz organizacji. Każdej z tych osób ma ona pozwolić na zrozumienie Scruma w oparciu o kluczowy zbiór koncepcji omówionych przy użyciu zrozumiałego słownictwa. Dzieląc się tą fundamentalną wiedzą, mam nadzieję zblizić Twoją organizację do jak najlepszego wykorzystania Scruma i dostarczania wartości biznesowych.

<sup>1</sup> Dokument można pobrać z witryny [www.scrum.org](http://www.scrum.org) w polskiej wersji językowej — przyp. tłum.

Wyobrażam sobie sytuację, w której każdy członek zespołu scrumowego ma ten podręcznik na swoim biurku, otwarty na rozdziale, który odpowiada aktualnie wykonywanej pracy. Widzę również menedżerów wszystkich poziomów organizacji czytających ją w celu zrozumienia, dlaczego Scrum może być efektywnym podejściem do zarządzania pracą oraz do zrozumienia zmian potrzebnych w organizacji, aby można było pomyślnie wprowadzić Scrum do użycia. Również organizacje planujące użyć zinnego podejścia innego niż Scrum znajdą tutaj informacje pomocne w specyficznej dla siebie adaptacji idei zwinności.

## Jak zorganizowana jest ta książka?

Zaczynamy od krótkiego wprowadzenia do Scruma (rozdział 1.), a kończymy omówieniem kierunków, w których możesz podążać dalej (rozdział 23.). Pozostałe rozdziały zostały podzielone na cztery części:

- Część I — Zagadnienia podstawowe (rozdziały 2 – 8): środowisko Scrum, zasady zwinności, sprinty, wymagania i historyjki użytkownika, rejestr produktu, estymacja i prędkość, dług techniczny.
- Część II — Role (rozdziały 9 – 13): właściciel produktu, mistrz młyna, zespół deweloperski, struktury zespołu scrumowego, menedżerowie.
- Część III — Planowanie (rozdziały 14 – 18): zasady planowania w Scrumie, planowanie wielopoziomowe, planowanie portfela, przewidywanie/planowanie produktu, planowanie wersji dystrybucyjnej.
- Część IV — Sprinty (rozdziały 19 – 22): planowanie sprintu, wykonanie sprintu, przegląd sprintu, retrospekcja sprintu.

## Jak korzystać z tej książki

Jak zapewne się domyślasz, napisałem tę książkę z założeniem, iż większość osób będzie ją czytać chronologicznie od początku do końca. Jeśli dopiero zaczynasz swoją przygodę ze Scrumem, właśnie tak powinieneś podejść do tej książki, ponieważ kolejne rozdziały bazują na wiedzy z rozdziałów poprzednich. Jeśli jednak szukasz całościowego opisu środowiska Scrum, przeczytaj rozdział drugi.

Osoby lepiej zaznajomione ze Scrumem mogą wykorzystać tę książkę jako poradnik. Jeśli interesuje Cię retrospekcja sprintu, skocz bezpośrednio do rozdziału 22. Jeśli chciałbyś rozpracować niuanse rejestrów produktu, przejdź do rozdziału 6. Gorąco polecam jednak, aby każdy z czytelników, łącznie z tymi, którzy znają Scruma, przeczytał w całości rozdział 3. Przedstawione tam zasady formułują fundament środowiska Scrum i pozostałą częścią książki. Nie jest to zwykłe powtórzenie wartości i zasad Manifestu Zwinności [Beck i in., 2001], co często ma miejsce w wielu innych opisach Scruma.

## Wizualny język ikon

Z dumą prezentuję w tej książce nowy wizualny język służący do opisu Scruma. Język ten bazuje na słowniku ikon zaprojektowanych w celu uchwycenia kluczowych ról, artefaktów i aktywności w Scrumie. Pozwala na przekazywanie koncepcji i znacznie ułatwia budowanie wspólnego rozumienia Scruma. Jeśli chciałbyś otrzymać kolorowe grafiki języka Scrum (ta książka zawiera wersję dwukolorową), odwiedź stronę [www.innolution.com](http://www.innolution.com). Znajdziesz tam również szereg zasobów oraz dyskusji związanych z książką.

## Zaczynamy

Zatem niezależnie od swojej roli i sytuacji z jakiegoś powodu zdecydowałeś się wziąć do ręki tę książkę i spędzić trochę czasu, poznając Scrum. Być może na kolejnych stronach znajdziesz niezwykle funkcjonalne środowisko, które przystosujesz do swoich potrzeb i dzięki któremu w zdecydowany sposób poprawisz metodologię produkcji i dystrybucji produktów i usług do zachwyconych klientów.

# PODZIĘKOWANIA

---

Powstanie tej książki nie byłoby możliwe bez pomocy wielu osób, włączając w to tysiące studentów uczęszczających na moje zajęcia z technik zwinności oraz sesje treningowe. Wymieniając osoby z nazwiska i imienia, ryzykuję pominięcie kogoś. Do osób, o których zapomniałem wspomnieć, kieruję następujące słowa: wiedziecie, że wszystkie nasze dyskusje oraz wymiana e-maili miały dla mnie nieocenione znaczenie i z całą pewnością wpłynęły na kształt tej książki!

Trzem osobom pragnę podziękować szczególnie, są to: Mike Cohn, Rebecca Traeger i Jeff Schaich. Bez ich zaangażowania — każdego z nich z osobna — ta książka byłaby zaledwie cieniem tego, co macie przed sobą.

Mike Cohn jest moim przyjacielem i kolegą od czasu, kiedy pierwszy raz mieliśmy okazję pracować razem w firmie Genomica, był to rok 2000. Był on niezwykle łaskaw umieścić moją książkę w swojej serii zatytułowanej *Mike Cohn Signature Series*. Dzięki związkowi z Mike’iem i innymi prestiżowymi autorami tej serii „wyglądam dobrze przez pryzmat moich znajomych” — jak powiedzieliby moi rodzice. Mike był moim słuchaczem za każdym razem, kiedy chciałem przedyskutować jakieś idee lub omówić strategię książki. Znajdował dla mnie czas w swoim niezwykle ciasnym planie zajęć, aby przejrzeć kolejny rozdział i podzielić się wnikliwymi uwagami na jego temat. Praca z nim przez wszystkie te lata była bardzo owocnym doświadczeniem, które mam nadzieję będzie kontynuowane w przyszłości.

Rebecca Traeger była moim osobistym redaktorem tej książki. Pracowaliśmy razem od czasów, kiedy pełniłem rolę dyrektora zarządzającego w Scrum Alliance (rok 2007). W tym okresie Rebecca była redaktorem witryny internetowej Scrum Alliance — dzięki tej pracy (a także wielu innym od tego czasu) stała się czołowym redaktorem materiałów związanych z metodami zwinności. Już na samym początku powstawania tej książki skontaktowałem się z Rebeccą i zapytałem, czy chciałaby powrócić do współpracy ze mną, i na moje szczęście zgodziła się. Nikt nie miał prawa zobaczyć jakiegokolwiek rozdziału, zanim nie przeszedł on przez jej ręce. Były momenty, kiedy rumieniłem się na widok jej poprawek, ponieważ nieustannie poprawiała mój sposób wypowiedzi, czyniąc go bardziej zrozumiałym i przystepnym. Jeżeli jakaś sekcja tej książki przypadła Ci do gustu, możesz być pewny, że przyczyniła się do tego Rebecca. Jeśli jest przeciwnie, prawdopodobnie nierozsądnie postanowiłem zignorować jej uwagi w tej części.

Jeff Schaich jest niezwykłym artystą i designerem. Pracowaliśmy nad tak dużą liczbą różnych projektów artystycznych, że nie jestem w stanie przywołać ich wszystkich. Już na etapie formułowania założeń tej książki chciałem stworzyć graficzny słownik pojęć związanych ze Scrumem i metodami zinnymi. Rysunki miały być podstawą do prezentacji w trakcie prowadzonych przeze mnie treningów, ale także częścią ponad 200 ilustracji w tej książce. Wiedziałem, że do zrealizowania tego pomysłu będę potrzebował doskonałego дизайнера. Jeff zgodził się podjąć wyzwanie. Bywały momenty, kiedy niniejsza książka wydawała się być dwoma różnymi projektami — pisaniem treści i tworzeniem koncepcji artystycznych. Nie jestem w stanie powiedzieć, która część zajęła więcej czasu, ale z całą pewnością bez artystycznego wkładu Jeffa książka byłaby o wiele uboższa.

Spotkał mnie niezwykły honor ze strony Mike'a Cohna i Rona Jeffriesa, dwóch guru w środowisku zwinności, którzy napisali słowo wstępne do książki! Każdy z nich na swój unikatowy sposób wykonał wspaniałą pracę, umieszczając książkę w odpowiednim kontekście i otwierając dyskusję na temat esencji Scruma. Od siebie dodam tylko: Mike, przestań się stoować w Burger Kingu; Ron, dzięki, że nie nacisnął czerwonego przycisku.

Chciałbym również podziękować całej masie ludzi, którzy mimo nawału prac znaleźli czas na przejrzenie rozdziałów i odesłanie swoich uwag. Zaczynę od recenzentów, od których otrzymałem rozszerzoną dawkę informacji zwrotnej. To: Joe Balistrieri, Johannes Brodwall, Leyna Cotran, Martine Devos, Scott Duncan, Ilan Goldstein, John Hebley, Geir Hedemark, James Kovacs, Lauri Mackinnon, Robert Maksimchuk i Kevin Tureski.

Dodatkowo pragnę podziękować innym recenzentom, którzy dostarczyli doskonałych spostrzeżeń na temat wybranych rozdziałów, są to: Lyssa Adkins, John Bauer, Sameer Bendre, Susan Briscoe, Paweł Brodzinski, Rowan Bunning, Josh Chappell, Lisa Crispin, Ward Cunningham, Cornelius Engelbrecht, Julia Frazier, Brindusa Gabur, Caroline Gordon, Drew Jemilo, Mike Klimkosky, Tom Langerhorst, Bjarne Larsen, Dean Leffingwell, Mauricelle Rutte, David Luzquiños, Lv Yi, Shay McAulay, Armond Mehrabian, Sheriff Mohamed, Cuan Mulligan, Greg Pease, Roman Pichler, Jacopo Romei, Jens Schauder, Bill Schroeder, Yves Stalgies, Branko Stojaković, Howard Sublett, Julie Sylvain, Kevin Tambascio, Stephen Wolfram i Michael Wollin.

Dziękuję pracownikom wydawnictwa Pearson, którzy byli wspaniałymi parterami podczas tego projektu. Cierpliwie tolerowali moje opóźnienia i zachęcali do dalszej pracy. Szczególne podziękowania należą się Chrisowi Guzikowskiemu, który nadzorował całość od samego początku do końca. Był na miejscu od mojego pierwszego spotkania z przedstawicielami Pearson w pubie w Lexington aż do ukończenia produkcji. Pragnę również podziękować Olivii Basegio za biegłą obsługę logistyczną i Julii Nahil za wspaniałą pracę nadzorczą nad projektem. Dodatkowo podziękowania kieruję do Barbary Wood za wspaniałą pracę przy wygładzaniu skryptu i do Gail Cooker za zebranie wszystkich grafik w jedną spójną i przepiękną całość.

Dziękuję również mojej asystentce, Lindsey Kalicki, której przekazywałam realizację wielu ważnych zadań, dzięki czemu mogłem skupić się na rozwijaniu książki. Mam wielkie szczęście pracować z taką profesjonalistką.

Przede wszystkim pragnę wyrazić podziękowania mojej rodzinie — Jenine, Jonah i Asherowi — za odegranie kluczowej roli. Prosiłem ich o bardzo wiele w trakcie długiego wytężonego wysiłku tworzenia tej książki. Żaden dowód wdzięczności nie jest w stanie zrównoważyć presji, jaką nałożyłem na moją rodzinę, oraz wynagrodzić utracony czas.

Jennie jest moją kochającą bratnią duszą, która trwała przy mnie podczas wszystkich wzlotów i upadków związanych z pisaniem książki. Gdybym chciał wysłuchać wszystkich poświęceń, jakie musiała ponieść, abym mógł pisać, wystarczyłoby ich na powieść dwa razy dłuższą od tej pozycji. Nie dokonałbym tego bez niej!

Co ciekawe, rok po naszym ślubie w 1993 roku opublikowałem moją pierwszą książkę *Succeeding with Objects*. W tym czasie Jenine obiecała, że nigdy więcej nie napisze kolejnej. Na szczęście dla mnie pamiętać o tych wydarzeniach uleciała po piętnastu latach, a patrząc wstecz, przytaczający ciężar pracy nie wydawał się już taki zły. Kiedy zatem zachęciła mnie do napisania niniejszej książki, byłem, mówiąc najdelikatniej, zaskoczony. Nie powiedziała mi jeszcze, że nie mogę napisać książki numer 3, ale przypuszczam, że będzie musiało minąć kolejnych piętnaście lat, zanim wspomnienie o tej książce zatrze się dostatecznie mocno, aby którekolwiek z nas chciało powstania następnej!

Bardzo mocno doceniam również pełne miłości wsparcie moich synów, Jonah i Ashera. Zrezygnowali ze spędzania czasu ze swoim ojcem, dzięki czemu mogłem pisać. Zawsze znajdywali chwilę, aby podyskutować o różnych pomysłach i podzielić się swoimi spostrzeżeniami na temat książki. Wiele z nich uwag, a także sugestii na temat grafik znalazło odzwierciedlenie w książce — dzięki nim jest ona lepsza! Mam nadzieję, że poznali, jaką wartością jest wytrwałość, a także że nawet najbardziej zniechęcająca praca może zostać wykonana, jeśli realizuje się ją krok po kroku bez poddawania się.

I w końcu chcę uhonorować moją mamę, Joyce Rubin (Genesha Ester bat Avrahm), za całe jej wsparcie i miłość, jaką mi dała. Bez jej wpływu powstanie tej książki byłoby niemożliwe. Niestety zmarła, nie doczekawszy publikacji. Jej odejście w styczniu 2012 roku pozostawiło w moim życiu i w życiu jej rodziny pustkę, która nigdy nie zostanie wypełniona. Była bardzo wyjątkową osobą dla wielu, których miała okazję spotkać na swojej drodze. Mamo, brakuje mi Ciebie bardziej, niż jestem w stanie wyrazić to jakimikolwiek słowami.

## O AUTORZE

---

**Kenny Rubin** prowadzi szkolenia i treningi zwinności oraz Scruma, pomagając firmom wytwarzanie produkty w sposób efektywny i ekonomicznie opłacalny. Jako certyfikowany trener Scruma wyszkolił ponad osiemnaście tysięcy ludzi w dziedzinach takich jak zwinność, Scrum, programowanie w języku Smalltalk, zarządzanie projektami zorientowanymi obiektowo, a także zarządzanie transformacją. Wytrenował ponad 200 firm, zaczynając od startupów, po firmy z listy Fortune 10.

Kenny był pierwszym dyrektorem zarządzającym Scrum Alliance, organizacji typu non profit skupiającej się na pomyślnym wdrażaniu Scruma. Kenny jest również współautorem książki z 1995 roku zatytułowanej *Succeeding with Objects: Decision Frameworks for Project Management*. Otrzymał tytuł licencjata w dziedzinie informatyki w Instytucie Techniki stanu Georgia, a także tytuł magistra informatyki na Uniwersytecie Standford.

Kenny wywodzi się ze społeczności specjalistów technologii zorientowanych obiektowo. Swoją karierę rozpoczął w 1995 roku jako programista języka Smalltalk w projekcie sfinansowanym przez NASA, gdzie stworzył pierwszy, poza LISP, system ekspercki wykorzystujący architekturę tablicową. W roku 1988 przyłączył się do nowo powstałej firmy ParcPlace System, będącej odłamem Xerox PARC, której celem było wyprowadzenie technologii zorientowanych obiektowo z laboratoriów badawczych i pokazanie ich światu. Jako konsultant języka Smalltalk współpracujący z wieloma różnymi organizacjami pod koniec lat osiemdziesiątych i na początku dziewięćdziesiątych Kenny jako jeden z pierwszych zaczął wdrażać praktyki zwinne. Pierwszy raz wykorzystał Scrum w roku 2000 do stworzenia oprogramowania z dziedziny bioinformatyki.

Kenny piastował wiele funkcji na ścieżce swojej kariery. Był między innymi odnoszącym sukcesy scrumowym właścicielem produktu, mistrzem młyna oraz członkiem zespołu deweloperskiego. Oprócz tego zajmował również stanowiska kierownicze: dyrektora generalnego, dyrektora operacji, zastępcy prezesa ds. produkcji, zastępcy prezesa ds. zarządzania produktem oraz zastępcy prezesa ds. usług. Nadzorował prace produkcyjne pięciu komercyjnych linii produktów, które wspólnie przyniosły zysk ponad 200 milionów dolarów. Był również bezpośrednio zaangażowany w zebranie ponad 150 milionów dolarów kapitału inwestycyjnego i asystował wejściu dwóch firm na rynek giełdowy NASDAQ.

Wielorakie doświadczenie Kenny'ego daje mu możliwość zrozumienia (i wyjaśniania) Scruma oraz jego implikacji w równym stopniu z perspektywy zespołu deweloperskiego, jak i kierownictwa.

# Rozdział 1

## WPROWADZENIE

---

Dnia 21 czerwca 2000 roku zostałem zatrudniony jako wiceprezes w bioinformatycznej firmie Genomica z siedzibą w Boulder w stanie Colorado. Pamiętam tę datę dokładnie, ponieważ kilka godzin wcześniej, o pierwszej w nocy, urodził się mój syn, Asher.

Jego urodziny zwiastowały dobry początek dnia. Asher urodził się w przewidzianym dniu rozwiązania (w Stanach Zjednoczonych zdarza się to w około 5% przypadków). Zatem my (a właściwie moja żona, Jenine) ukończyliśmy nasz dziewięciomiesięczny „projekt” zgodnie z planem. Mało tego, Asher otrzymał bardzo wysoką ocenę Apgar, co oznaczało, że wynik naszej pracy był zdrowy i dobrej jakości! Nasz największy interesariusz — starszy syn Jonah — był przeszczęśliwy, mając młodszego brata. Na czas, dobrej jakości, z zachwyconymi interesariuszami — to był naprawdę dobry dzień!

Po krótkiej drzemce sprawdziłem swoją pocztę i zobaczyłem, że prezes Genomiki wysłał do mnie pilną wiadomość z prośbą o pojawienie się na posiedzeniu dyrektorów o ósmej rano tego samego dnia. Niechętnie opuściłem szpital i pojechałem na to spotkanie.

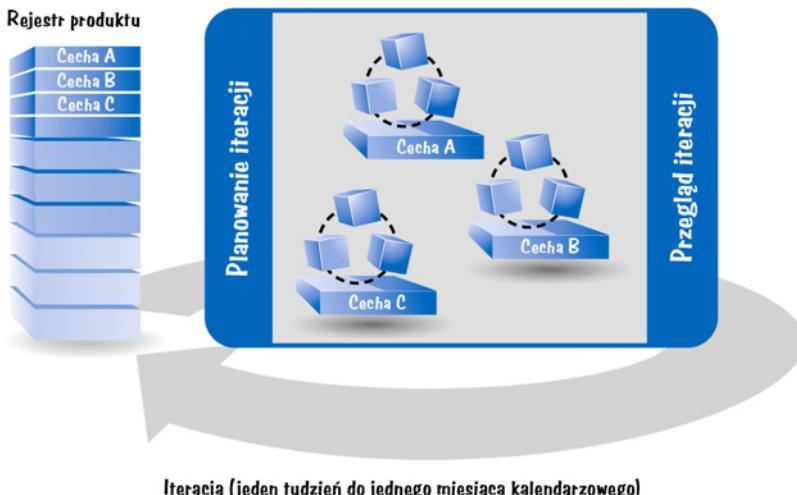
Kiedy przyjechałem na miejsce, powiedziano mi, że dyrektor produkcji został zwolniony po przedniej nocy i od tej pory przejmuję po nim 90-osobowy zespół produkcyjny. Nie byłem zaskoczony. Od kilku miesięcy menedżerowie i dyrektorowie dyskutowali nad problemem niezdolności firmy do dostarczania wartościowych produktów na czas z zachowaniem odpowiedniej jakości, a dyrektor produkcji był centralnym punktem tej dyskusji.

Od tego momentu ja odpowiadałem za nadzór nad wysiłkami zmierzającymi do stopniowej poprawy wyników zespołu zajmującego się wytwarzaniem produktów. Pamiętam, że poraziła mnie ironia pomyślnych narodzin i moich nowych obowiązków.

Ponieważ i tak byłem już bardzo zajęty nadzorem nad sprzedażą i marketingiem, pozwolono mi zatrudnić nowego dyrektora odpowiedzialnego za produkcję i podlegającego bezpośrednio mnie. Osobą, którą wybrałem, był Mike Cohn [Cohn, 2004; Cohn, 2006, Cohn, 2010], a nowym podejściem, jakie zamierzaliśmy wprowadzić, był Scrum.

## Czym jest Scrum?

Scrum jest zwinnym podejściem służącym produkcji innowacyjnych produktów i usług. Rysunek 1.1 pokazuje w uproszczeniu ogólną zasadę produkcji w sposób zwinny.



RYSUNEK 1.1. Przedstawienie procesu zwinnej produkcji

Zwinne podejście do produkcji rozpoczyna się od stworzenia **rejestru produktu** — posortowanej według priorytetów listy cech i innych możliwości niezbędnych do stworzenia działającego produktu. Kierując się rejestrzem produktu, zawsze pracujesz w pierwszej kolejności nad elementami najważniejszymi. Kiedy kończą się zasoby (takie jak czas), każda praca, która nie została wykonana, będzie miała niższy priorytet od tej, która została ukończona.

Sama praca jest wykonywana w krótkich, ograniczonych czasowo **iteracjach**, które trwają przeważnie od jednego tygodnia do jednego miesiąca kalendarzowego. Podczas każdej iteracji samoorganizujący się, **wielofunkcyjny zespół** wykonuje całą niezbędną pracę — projektowanie, budowanie i testowanie — potrzebną do stworzenia kompletniej, działającej cechy, którą można przeznaczyć do produkcji.

Zazwyczaj zakres pracy w rejestrze produktu jest znacznie większy od tego, jaki może być ukończony przez zespół w trakcie jednej krótkiej iteracji. W związku z tym na początku każdej iteracji zespół planuje, jaką najważniejszą część rejestru produktu wykonać w ciągu najbliższej iteracji. Na przykład na rysunku 1.1 zespół zgodził się stworzyć cechy A, B i C.

Na końcu iteracji zespół dokonuje przeglądu ukończonych cech wspólnie z interesariuszami w celu otrzymania od nich ewentualnych uwag. Na podstawie tych uwag właściciel produktu oraz zespół deweloperski mogą zmienić to, co zamierzają zrealizować w kolejnym sprincie, a także sposób realizacji tej pracy. Na przykład: jeżeli interesariusze zobaczą gotową cechę w działaniu i na tej podstawie dojdą do wniosku, że pewna inna cecha, której wcześniej nie rozważali, musi również trafić do produktu, właściciel produktu będzie mógł zwyczajnie stworzyć nową historyjkę reprezentującą tę cechę i wstawić ją w odpowiednim miejscu rejestru produktu, tak aby praca nad nią została podjęta podczas jednej z nadchodzących iteracji.

Pod koniec każdej iteracji zespół powinien posiadać produkt potencjalnie gotowy do wdrożenia (lub przyrost produktu) — czyli taki, który w razie potrzeby może zostać wypuszczony na rynek. Jeżeli wdrażanie po każdym sprincie nie jest w interesie firmy, można przeprowadzać wdrożenia całego zestawu cech będącego wynikiem wielu iteracji.

Po zakończeniu każdej iteracji cały proces powtarzany jest od początku, zaczynając od planowania następnej iteracji.

## Początki Scruma

Bogatą historię Scruma można prześledzić wstecz do opublikowanego w roku 1986 artykułu w „Harvard Business Review” zatytułowanego *The New New Product Development Game* („Nowy sposób wytwarzania nowych produktów” [Takeuchi i Nonaka, 1986]). Artykuł ten opisuje, w jaki sposób firmy takie jak Honda, Canon i Fuji-Xerox osiągnęły wyniki na światowym poziomie, używając skalowalnego podejścia, opartego na zespołowym **wytwarzaniu w trybie wszystko naraz**. Akcentuje również wagę samodzielnych, samoorganizujących się zespołów i nakreśla rolę kadry kierowniczej w procesie produkcyjnym.

Artykuł z 1986 roku miał duży wpływ na połączenie ze sobą wielu koncepcji, które dały początek temu, co dzisiaj nazywamy Scrumem. Scrum nie jest akronimem — jest terminem zaczerpniętym z gry rugby, w której określa sposób ponownego rozpoczęcia gry po przypadkowym naruszeniu zasad lub jeśli piłka opuściła teren gry. Nawet jeśli nie jesteś fanem rugby, prawdopodobnie widziałeś „młyn” złożony z dwóch grup zawodników skierowanych naprzeciw siebie, oplatających się ramionami z głowami skierowanymi w dół i usiłujących przejąć kontrolę nad znajdująca się w centrum piłką.

Takeuchi i Nonaka posłużyli się przenośniami rugby i młyna do opisania pracy nad produktem:

*„Sztafetowe” podejście do wytwarzania produktu... może stać w sprzeczności z celami maksymalizacji szybkości i elastyczności. Zamiast podejścia całościowego lub podejścia w stylu „rugby”, w którym zespół próbuje pokonać pewien dystans w jednym kroku, lepsze z punktu widzenia współczesnych wymogów konkurencyjności wydaje się wielokrotne przekazywanie piłki do tyłu i do przodu.*

W roku 1993 Jeff Sutherland i jego zespół w korporacji Easel stworzyli proces Scrum przeznaczony do wytwarzania oprogramowania komputerowego, łącząc ze sobą koncepcje przedstawione w artykule z roku 1986 z ideami programowania zorientowanego obiektywnego, empirycznej kontroli nad procesem, wytwarzania w sposób iteracyjny i przyrostowy, analizy procesów i wydajności wytwarzania oprogramowania oraz złożonych systemów adaptacyjnych. W roku 1995 Ken Schwaber opublikował pierwszą pracę dotyczącą Scruma na konferencji OOPSLA 1995 [Schwaber, 1995]. Od tego czasu Schwaber i Sutherland, działając wspólnie lub indywidualnie, opublikowali kilka prac poświęconych Scrumowi — między innymi *Agile Software Development with Scrum* [Schwaber i Beedle, 2001], *Agile Project Management with Scrum* [Schwaber, 2004] i „Przewodnik po Scrumie” [Schwaber i Sutherland, 2011].

Chociaż Scrum jest stosowany najczęściej do produkcji oprogramowania komputerowego, jego kluczowe wartości i zasady są używane przy wytwarzaniu innych typów produktów lub do organizowania **przepływu** różnych rodzajów pracy. Przykładowo: miałem okazję współpracować

z instytucjami, które z powodzeniem wykorzystały Scrum do organizacji i zarządzania procesem wytwarzania sprzętu elektronicznego, programów marketingowych i inicjatyw związanych ze sprzedażą.

## Dlaczego Scrum?

Dlaczego zatem podejście zwinne, takie jak Scrum, okazało się dobrym wyborem w firmie Genomica? Po pierwsze, nie ulegało wątpliwości, iż stosowane wcześniej podejście do wytwarzania oprogramowania nie działało. To była zła wiadomość. Dobrą wiadomością było to, że wszyscy zgadzali się z tym faktem.

Genomica operowała w domenie złożonej, w której istniało więcej niewiadomych niż wiadomych. Budowaliśmy produkty, których nikt nigdy wcześniej nie budował. Skupialiśmy się na najnowocześniejszych, nieustannie ewoluujących, najwyższej klasy platformach informatycznych, za pomocą których naukowcy mogli odkryć następną supermolekułę. Potrzebowaliśmy sposobu, który pozwoliłby nam na szybkie analizowanie nowych idei i podejść, a także na szybkie odkrywanie, które rozwiązania są możliwe, a które nie. Mieliśmy strategicznego partnera biznesowego, któremu należało prezentować działające wyniki mniej więcej co kilka tygodni, aby otrzymać użyteczną informację zwrotną, ponieważ nasz produkt musiał współpracować z jego linią sekwenserów DNA. Ta potrzeba szybkiej eksploracji i informacji zwrotnej nie integrowała się zbyt dobrze z naszym istniejącym procesem szczegółowego planowania w przód.

Chcieliśmy również uniknąć konieczności odgórnego projektowania potężnej architektury. Po przednia próba stworzenia następnej generacji kluczowego produktu w Genomice skończyła się ogromnymi kosztami związanymi z niemal rocznym projektowaniem samej architektury pozwalającej na stworzenie ogromnej, zunifikowanej platformy bioinformatycznej. Kiedy w końcu powstała pierwsza rzeczywista naukowa aplikacja bazująca na tej architekturze i mogliśmy sprawdzić decyzje projektowe podjęte wiele miesięcy wcześniej, okazało się, że przejście tabulatorem z jednego okienka do drugiego zajmuje 42 sekundy. Biorąc pod uwagę brak cierpliwości wśród zwykłych użytkowników, spróbuj sobie wyobrazić biologa molekularnego z tytułem doktorskim, który musi czekać 42 sekundy! To była katastrofa. Potrzebowaliśmy innego, bardziej wyważonego podejścia do projektowania, takiego, które wymagałoby niewielkiej dozy projektowania w przód połączonej ze sporą dawką projektowania wstępującego wykonywanego w chwili, kiedy będzie to potrzebne.

Chcieliśmy również, aby nasze zespoły były bardziej wielofunkcyjne. Z historycznego punktu widzenia Genomica działała tak jak większość organizacji. Zespoły deweloperskie przekazywały pracę zespołowi testującym dopiero w chwili ukończenia prac. Pragnęliśmy, aby wszyscy członkowie zespołów synchronizowali swoją pracę częściej — naszym celem było robienie tego codziennie. W przeszłości błędy potęgowały się, ponieważ ważne problemy były analizowane zbyt późno w procesie deweloperskim. Ludzie z różnych obszarów procesu produkcyjnego komunikowali się ze sobą zbyt rzadko.

Z tych i innych powodów uznaliśmy, że Scrum będzie dobrym rozwiązaniem dla Genomiki.

## Wyniki Genomiki

Scrum nie był jeszcze dobrze znany, kiedy zdecydowaliśmy się działać zgodnie z jego zasadami. Pierwsza książka na jego temat pojawiła się dopiero w następnym roku [Schwaber i Beedle, 2001]. Udało nam się jednak zebrać razem dostępne informacje i zacząć działać najlepiej, jak potrafiliśmy — wyniki okazały się zdecydowanie lepsze od naszych dotychczasowych pocynań (patrz tabela 1.1).

**TABELA 1.1.** Wyniki Genomiki w Scrumie

Rodzaj miary	Proces kaskadowy	Scrum
Wysiłek	10x	1x
Prędkość	1x	7x
Satyfakcja klienta	Bardzo mała	Doskonała

Patrząc z punktu widzenia podejmowanego wysiłku, dzięki produkcji w Scrumie potrzebowaliśmy jednej dziesiątej siły roboczej (wyliczonej w osobomiesiącach) w porównaniu ze stosowanym wcześniej **kaskadowym** procesem sterowanym planem, aby wyprodukować taką samą funkcjonalność produktu. Proces produkcyjny w Scrumie przebiegał siedem razy szybciej w porównaniu z pracą w trybie kaskadowym, czyli mówiąc inaczej: praca w Scrumie dostarczała mniej więcej siedem razy więcej wartościowych cech w porównaniu z pracą kaskadową. Jeszcze bardziej zachęcające było to, że mogliśmy dostarczać oprogramowanie naszemu partnerowi w oknie czasowym, które odpowiadało jego wymaganiom związanym z wdrażaniem nowej platformy sprzętowej. To wzmacniło nasze długofalowe partnerstwo biznesowe i w konsekwencji doprowadziło do zwiększenia wartości rynkowej Genomiki.

## Czy Scrum może pomóc Tobie?

Zaobserwowane w Genomice zjawisko budowania nikomu niepotrzebnych cech produktu oraz dostarczania ich zbyt późno i ze zbyt małą jakością nie jest odosobnionym przypadkiem. Genomica, jak wiele innych organizacji, przetrwała dzięki temu, że zachowywała poziom nie gorszy od konkurencji. Takie same problemy obserwowałem, rozpoczynając pracę zawodową jako programista w połowie lat osiemdziesiątych ubiegłego wieku. W przypadku wielu firm ta sytuacja nie uległa poprawie pomimo upływu 30 lat.

Co odpowiedzieliby Twoi sprzedawcy i deweloperzy, gdybyś dzisiaj zapytał ich „Czy jesteście zadowoleni z naszego procesu produkcyjnego?” lub „Czy waszym zdaniem dostarczamy wartościowe produkty na czas, z zachowaniem zasad ekonomii i jakości?”.

Ludzie, których spotykam podczas moich sesji treningowych na całym świecie, odpowiadają częściej na oba pytania „Nie”. Twarzyszą temu chory głosów mówiące: „Liczba nieudanych projektów jest na zbyt wysokim, nieakceptowanym poziomie”, „Wdrożenia opóźniają się”, „Zwrot z inwestycji bardzo często wypada poniżej oczekiwani”, „Jakość oprogramowania jest bardzo niska”, „Produktywność jest na żenującym poziomie”, „Nikt nie ponosi odpowiedzialności za osiągane wyniki”, „Morale wśród pracowników jest niskie”, „Rotacja pracowników jest zbyt duża”. Między wierszami pojawia się również mniej lub bardziej poważne stwierdzenie „Musisz istnieć lepszy sposób”.

Mimo że ludziom nie podoba się ten stan rzeczy, większość z nich wydaje się być obojętna wobec faktu, iż brak zadowolenia jest częścią pracy przy produkcji oprogramowania — chociaż nie musi tak być.

Organizacje, które przyłożyły się do wdrożenia Scruma, doświadczają zupełnie innych realiów (patrz rysunek 1.2).



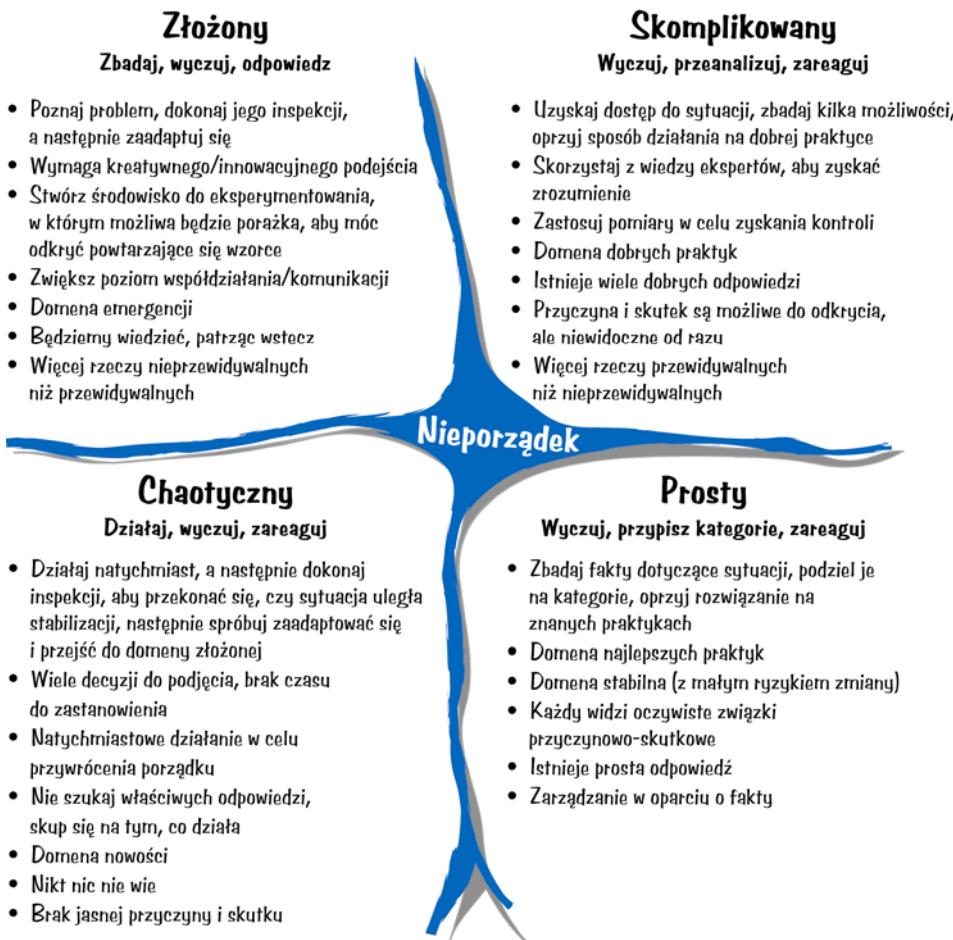
**RYSUNEK 1.2.** Zalety wdrożenia Scruma

Raz za razem pozytywnie zaskakują swoich klientów, dając im to, czego tak naprawdę oczekują, a nie tylko i wyłącznie cechy, które wyspecyfikowali w pierwszym dniu kontraktu, kiedy ich wiedza na temat faktycznych potrzeb była najmniejsza. Organizacje stosujące Scrum widzą również wyższe zyski z inwestycji dzięki dostarczaniu mniejszych, częstszych **wersji dystrybucyjnych**, a bezkompromisowe ujawnianie słabości organizacyjnych i **marnotrawstwa** redukuje koszty.

Skupienie podejścia scrumowego na dostarczaniu działających, zintegrowanych z resztą systemu i przetestowanych cech po każdej iteracji daje wynik w postaci szybkiego wdrażania wyprodukowanego oprogramowania. Konstrukcja Scruma pomaga organizacjom odnieść sukces w skomplikowanych realiach świata, gdzie niezbędna jest szybka adaptacja w oparciu o powiązane ze sobą działania konkurencji, klientów, użytkowników, urzędów regulacyjnych i innych interesariuszy. Scrum to również większa przyjemność z pracy dla wszystkich jego uczestników. Klienci są zachwyceni, a ludzie wykonujący rzeczywistą pracę odczuwają rzeczywistą satysfakcję wynikającą ze wzajemnej współpracy ze sobą, prowadzącej do polepszenia wzajemnych stosunków i większego zaufania pomiędzy członkami zespołu.

Nie zrozum mnie źle. Chociaż Scrum jest doskonałym rozwiązaniem w wielu sytuacjach, nie oznacza to, że może być stosowany uniwersalnie we wszystkich przypadkach. Służące do wnioskowania środowisko **Cynefin** [Snowden i Boone, 2007] pomaga nam zrozumieć sytuację, w której musimy działać i podejmować decyzje w oparciu o bieżące uwarunkowania. Definiuje ono i porównuje cechy pięciu różnych domen: **prostoty, skomplikowania, chaosu, złożoności i nieporządku**,

które mają miejsce, kiedy nie wiesz, w jakiej innej domenie znajdujesz się w danej chwili (rysunek 1.3). Posłużę się środowiskiem Cynefin do omówienia sytuacji, w których Scrum nie jest dobrym rozwiązaniem.



RYSUNEK 1.3. Środowisko Cynefin

Po pierwsze, należy zdać sobie sprawę, iż wiele aspektów produkcji i wsparcia oprogramowania komputerowego nie będzie pasować idealnie do wyłącznie jednej domeny środowiska Cynefin. Produkcja oprogramowania to skomplikowane przedsięwzięcie z nakładającymi się na siebie aspektami i czynnościami — kwalifikującymi się do wszystkich możliwych domen [Pelrine, 2011]. Stąd chociaż większość przypadków produkcji oprogramowania mieści się w domenach skomplikowanych i złożonych, stawianie tezy, iż produkcja oprogramowania jest domeną złożoną, byłoby naiwne, szczególnie jeśli zdefiniujemy ją jako pewne spektrum zawierające w sobie wytwarzanie nowych innowacyjnych produktów, utrzymanie istniejących produktów i działań oraz wsparcie użytkowników.

## Domena złożona

Podczas radzenia sobie z problemami złożonymi rzeczy są bardziej nieprzewidywalne niż przewidywalne. Jeżeli istnieje prawidłowa odpowiedź, poznamy ją, dopiero patrząc wstecz. Jest to domena **emergencji**. Musimy zagłębić się w sytuację, aby poznać problem, a następnie przeprowadzić jego **inspekcję i zaadaptować się** w oparciu o uzyskaną wiedzę. Praca w domenie złożonej wymaga podejścia kreatywnego i innowacyjnego. Rutynowe rozwiązania nie sprawdzają się. Musimy stworzyć środowisko testowe, w którym będziemy mogli przeprowadzać eksperymenty i tą drogą odkryć istotne informacje. W tym środowisku kluczowe znaczenie ma wysoki poziom interakcji i komunikacji. Do tej domeny zaliczają się prace nad nowymi innowacyjnymi produktami, a także wzbgacanie istniejących produktów o innowacyjne cechy.

Scrum nadaje się doskonale do realizacji działań w domenie złożonej. W takich sytuacjach klawiszowe znaczenie mają nasze umiejętności badania (odkrywania), wyczuwanie (inspekcji) i reagowania (adaptacji).

## Domena skomplikowana

Problemy skomplikowane należą do domeny dobrych praktyk realizowanych przez ekspertów. Może istnieć wiele odpowiedzi, ale do ich odkrycia potrzebna jest diagnoza przeprowadzona przez eksperta. Chociaż Scrum zdecydowanie nadaje się do rozwiązywania problemów tej klasy, niekonicznie będzie tutaj najlepszym podejściem. Dla przykładu: do poprawienia wydajności wymagającego dostosowania pewnych parametrów i znalezienia warunków najlepszej wydajności systemu znacznie lepiej będzie zwołać grupę ekspertów i pozwolić im ocenić sytuację, przetestować kilka możliwości, a następnie udzielić odpowiedzi w oparciu o stosowane przez nich najlepsze praktyki. Do tej kategorii zalicza się również wiele z codziennych zadań związanych z utrzymaniem oprogramowania (radzeniem sobie z problemami zgłaszanymi przez użytkowników czy też błędami w oprogramowaniu). W tej domenie sprawdzają się szczególnie różne podejścia taktyczne i kwantytatywne, jak Sześć Sigma, chociaż można je stosować również w przypadku domen prostoty.

## Domena prostoty

W przypadku problemów prostych każdy widzi przyczynę i skutek. Często prawidłowa odpowiedź jest oczywista i nie podlega dyskusji. Jest to domena najlepszych praktyk — istnieją znane rozwiązania problemu. Po ocenie faktów związanych z naszą sytuacją możemy podjąć decyzję odnośnie do zastosowania najlepszego znanego rozwiązania. Do rozwiązywania prostych problemów można zastosować Scrum, chociaż może nie być to najlepsze narzędzie do tego celu. O wiele lepszym podejściem będzie zastosowanie procesu o dobrze zdefiniowanej, powtarzalnej liście kroków, co do których wiadomo, iż rozwiązują dany problem. Przykładowo: jeśli chcemy produkować na okrągło ten sam produkt, dobrze skonstruowany proces budowania będzie lepszym rozwiązaniem niż Scrum. Podobnie będzie w przypadku wdrożenia tego samego komercyjnego produktu z półki (ang. *commerical-off-the-shelf [COSF] product*) w setnym środowisku klienta — wystarczy powtórzyć dobrze udokumentowane kroki instalacji i konfiguracji produktu.

## Domena chaosu

Problemy chaotyczne wymagają szybkiej reakcji. Jesteśmy w kryzysie i musimy działać natychmiast, aby zapobiec dalszym stratom i przywrócić względny porządek. Wyobraźmy sobie, że pewien uniwersytet opublikował artykuł twierdzący, iż nasz produkt ma błędnie działający algorytm i w związku z tym generuje błędne wyniki. W oparciu o takie wyniki nasi klienci podjęli istotne inwestycje biznesowe i teraz pozywają nas o odszkodowania. Nasz czołowy projektant algorytmów siedzi gdzieś w dżungli na Borneo i będzie nieosiągalny przez kolejne dwa tygodnie. Tutaj Scrum nie będzie najlepszym rozwiązaniem. Nie interesuje nas nadanie odpowiednich priorytetów elementom znajdującym się w rejestrze produktu i określenie, co będziemy robić w kolejnej iteracji. Musimy działać natychmiast i zdecydowanie zatrzymać „krwawienie”. W przypadku problemów chaotycznych ktoś musi stanąć na czele i zacząć działać.

## Nieporządek

Jesteś w domenie nieporządku, jeśli nie jesteś w stanie stwierdzić, w której z pozostałych domen się znajdują się. Jest to bardzo niebezpieczne miejsce, ponieważ nie wiesz, jak wyjść z danej sytuacji. W takich przypadkach ludzie zazwyczaj działają w oparciu o własną intuicję. W przypadku produkcji oprogramowania wiele osób najlepiej zna i dlatego też preferuje podejście sekwencyjne oparte na przechodzeniu przez kolejne fazy, które sprawdzają się najlepiej w domenach prostoty. Niestety takie podejście bardzo kiepsko nadają się do produkcji oprogramowania, o czym będę pisał w rozdziale 3. Kiedy znajdziesz się w domenie nieporządku, najlepszym sposobem wyjścia z tego stanu będzie rozbicie problemu na poszczególne części i przypisanie każdej z nich jednej z pozostałych domen. Nie próbuj zastosować Scruma w domenie nieporządku, Twoim celem jest wydostanie się z tej domeny.

## Praca sterowana przerwaniami

Scrum nie nadaje się do pracy sterowanej licznymi przerwaniami. Założymy, że prowadzisz firmę zajmującą się wsparciem użytkowników i chcesz zastosować Scrum do organizacji i zarządzania pracą pomocy technicznej. Rejestr produktu jest nieustannie aktualizowany w oparciu o żądania pomocy przychodzące drogą telefoniczną lub przez pocztę elektroniczną. W żadnym punkcie czasu nie posiadasz rejestru przewidującego zakres pracy daleko w przód, a jego treść i kolejność może się zmieniać bardzo często (być może co godzinę lub nawet co kilka minut).

W takiej sytuacji nie będziesz w stanie zaplanować iteracji trwających tydzień lub dłużej, ponieważ nie będziesz wiedział, jaką pracę należało zrealizować w tak odległym momencie. Nawet jeśli myślisz, że wiesz, jaka miałaby to być praca, istnieje bardzo duże prawdopodobieństwo, że w międzyczasie pojawi się żądanie pomocy o bardzo wysokim priorytecie, które będzie musiało być skierowane do realizacji przed zaplanowanymi zadaniami.

W środowisku sterowanym przerwaniami znacznie lepiej poradzisz sobie, rozważając alternatywne podejście zwinne o nazwie **kanban**. Kanban nie jest samodzielnym rozwiązaniem, ale podejściem nałożonym na istniejący proces. Kanban zaleca w szczególności:

- wizualizację przepływu pracy przez system (na przykład kroki podejmowane przez pracowników pomocy technicznej w celu rozwiązania problemu);
- ograniczenie pracy cząstkowej na każdym kroku, aby zapewnić, że nie wykonujesz więcej pracy, niż pozwalają na to Twoje zasoby;
- pomiar i optymalizację przepływu pracy przez system, aby umożliwić ciągłą poprawę wydajności.

Doskonałymi obszarami do zastosowania metody kanban są utrzymanie oprogramowania i pomoc techniczna użytkownikom. Niektóre z organizacji stosujących kanban zwracają uwagę na eliminowanie przez to podejście przeciżeń w pracy (przez dostosowywanie pracy cząstkowej do dostępnych zasobów) oraz redukowanie wahań w przepływie pracy przy jednoczesnym ewolucyjnym podejściu do zmian. Te cechy kwalifikują kanban do zastosowania w domenach złożonych.

Scrum i kanban są zwinnymi podejściami do produkcji oprogramowania, z których każde ma swoje wady i zalety. Zastosowanie jednego z nich powinno być poprzedzone wywnioskowaniem domeny, w której działasz. W niektórych organizacjach obie metodologie mogą być stosowane do rozwiązywania różnych współistniejących potrzeb. Na przykład Scrum może posłużyć do stworzenia nowego produktu, a kanban do sterowanej przerwaniami pomocy technicznej użytkownikom i utrzymania istniejących produktów.

## Zakończenie

Scrum nie jest lekarstwem na wszystko. Może jednak umożliwić Ci przeprowadzenie zmian, które towarzyszą wszystkim złożonym **wysiłkom deweloperskim produktu**. Scrum sprawdził się w Genomice i wielu innych firmach, które zdecydowały się zastosować podejście do wytwarzania oprogramowania lepiej pasujące do ich specyfiki.

Chociaż środowisko Scrum jest proste, błędem byłoby twierdzenie, iż jego zastosowanie w praktyce jest rzeczą trywialną i bezbolesną. Scrum nie odpowiada wyczerpująco na pytania stawiane przez Twój obecny proces, zamiast tego daje zespołom mandat do zadawania takich pytań i znajdowania odpowiedzi na nie. Scrum nie daje gotowych przepisów na wszelkie możliwe bolączki, zamiast tego ujawnia wszelkie nieprawidłowości i marnotrawstwa uniemożliwiające organizacjom osiągnięcie ich prawdziwego potencjału.

Osiągnięcie takiego stanu wiedzy może być bolesne dla niektórych organizacji, jeśli jednak pójdą one dalej tą drogą, nie zważając na początkowy dyskomfort i rozwiązyując problemy wykazane przez Scrum, będą mogły osiągnąć niezwykły postęp pod względem własnego procesu produkcyjnego, wytwarzanych produktów, a także poziomów zadowolenia klientów i pracowników.

Pozostała część książki poświęcona jest na analizę fundamentalnych aspektów Scruma. Zacznę od opisu całego środowiska Scrum, włączając w to role, aktywności, artefakty i **zasady**. Kto wie, jeśli zastosujesz Scrum w odpowiedni sposób i w odniesieniu do odpowiedniej sytuacji, być może również dostarczysz wartość w sposób tak pomyślny, jak moja żona i ja zrobiliśmy to tego brzemionego w skutki dnia w roku 2000.

Część I

## Pojęcia podstawowe

---



## Rozdział 2

# ŚRODOWISKO SCRUM

---

Ten rozdział poświęcony jest przeglądowi środowiska Scrum, a w szczególności jego zwyczajom, włączając w to role, aktywności i artefakty. Każdy z tych tematów będzie omawiany dokładniej w kolejnych rozdziałach z dogłębnym opisem rządzących nim zasad.

### Wprowadzenie

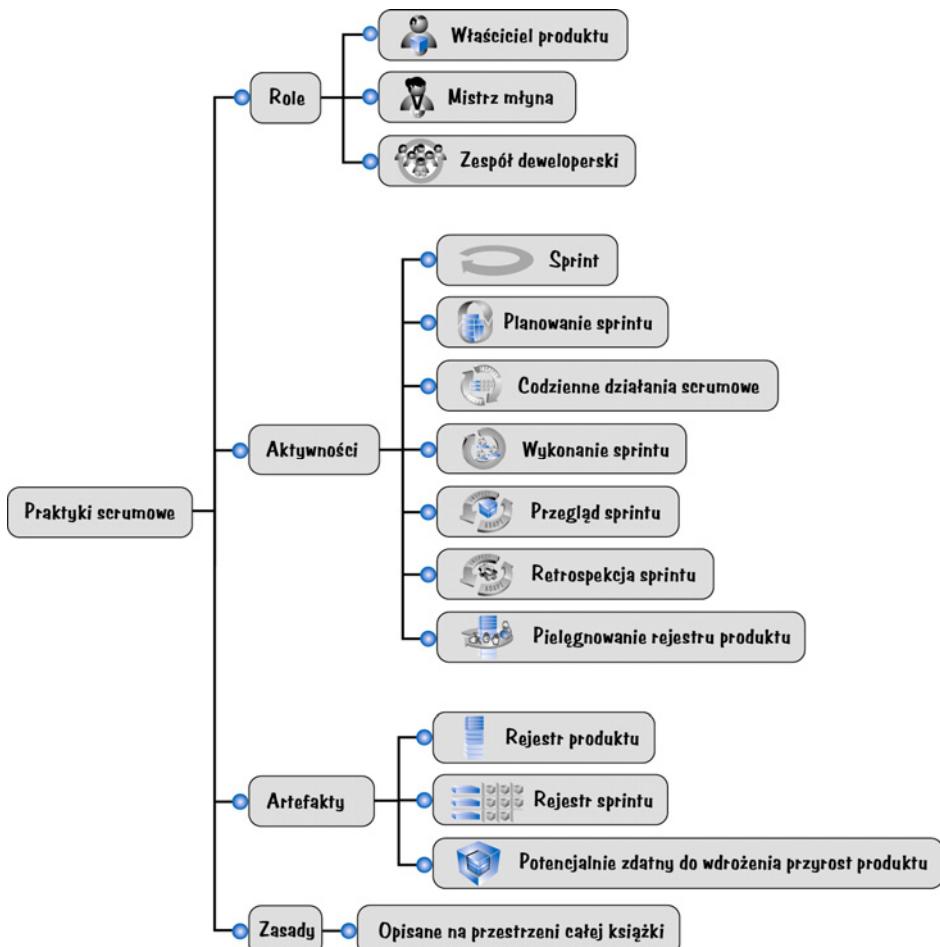
Scrum nie jest ustandaryzowanym procesem, w którym wystarczy wykonać metodycznie określona liczbę kroków, aby ze stu procentową pewnością otrzymać zachwycający klientów, wysokiej jakości produkt na czas i bez przekraczania założonego budżetu. Scrum jest raczej **środowiskiem** służącym do organizacji i zarządzania pracą. Środowisko Scrum to zbiór wartości, zasad i praktyk — podstawa, do której Twoja organizacja doda swoje własne metody realizacji prac inżynierskich, a także własne sposoby postępowania zgodnie ze Scrumem. W wyniku tego połączenia powstanie Twoja unikatowa wersja Scruma.

Aby lepiej zrozumieć koncepcję środowiska, spróbuj wyobrazić sobie Scrum jako fundament i ściany budynku — wartości, zasady i praktyki Scruma są tutaj kluczowymi elementami konstrukcji. Nie wolno ignorować lub całkowicie zmieniać jednej z wartości, zasad lub praktyk, nie ryzykując tym samym zawalenia całej konstrukcji. Możesz natomiast pozwolić sobie na dostosowanie wnętrza konstrukcji Scruma, dodając własne poprawki i cechy przekształcające proces do momentu, kiedy stanie się on użyteczny dla Ciebie.

Scrum jest środowiskiem niesamowicie prostym, skupionym na ludziach — bazuje na wartościach uczciwości, otwartości, szacunku, skupienia, zaufania, mobilizacji i pracy zespołowej. Zasady Scruma zostaną omówione szczegółowo w rozdziale 3., natomiast kolejne rozdziały podkreślą praktyki i podejścia bazujące na tych zasadach i wartościach.

Praktyki Scruma wyrażają się poprzez specyficzne role, aktywności oraz artefakty z towarzyszącymi im zasadami (patrz rysunek 2.1).

Pozostała część tego rozdziału będzie dotyczyć praktyk scrumowych.

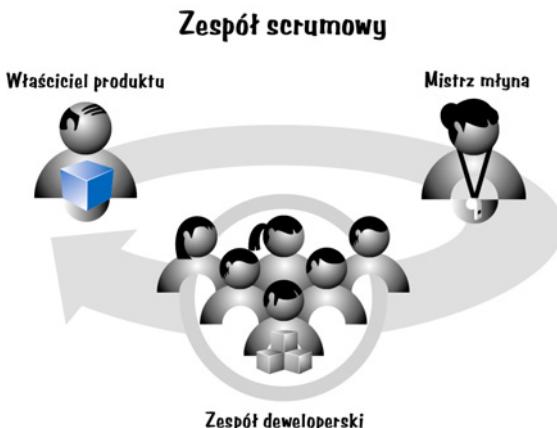


RYSUNEK 2.1. Praktyki scrumowe

## Role w Scrumie

Produkcja w Scrumie opiera się na minimum jednym **zespołe scrumowym**, na który składają się trzy role scrumowe: **właściciela produktu, mistrza młyna i zespołu deweloperskiego** (patrz rysunek 2.2). Stosując Scrum, można wprowadzić również inne role, ale samo środowisko Scrum wymaga jedynie trzech wspomnianych wyżej.

Właściciel produktu jest odpowiedzialny za to, co będzie produkowane i w jakiej kolejności. Mistrz młyna pomaga zespołowi w stworzeniu i przestrzeganiu swojego własnego procesu w kontekście środowiska Scrum. Na zespole deweloperskim spoczywa odpowiedzialność za dostarczenie wyników, o które poprosił właściciel produktu.



RYSUNEK 2.2. Role w Scrumie

Jeżeli jesteś menedżerem, nie przejmuj się brakiem roli „menedżer” na rysunku 2.2. Menedżerowie nadal odgrywają ważną rolę w organizacjach praktykujących Scrum (patrz rozdział 13.). Scrum definiuje jedynie role potrzebne w tym środowisku, nie zważając na wszystkie inne role, które mogą i powinny istnieć wewnętrz organizacji zdecydowanej skorzystać z tego podejścia.

## Właściciel produktu

Właściciel produktu jest uprawnionym centralnym punktem zarządzania. Jest osobiście odpowiedzialny<sup>1</sup> za podejmowanie decyzji, które cechy produktu powinny zostać zbudowane i w jakiej kolejności. Właściciel produktu utrzymuje i komunikuje innym uczestnikom procesu jasną wizję celu, jaki zespół scrumowy chce osiągnąć. Jest on odpowiedzialny za całkowite powodzenie wytwarzanego lub rozwijanego rozwiązania.

Nie ma znaczenia, czy realizowany jest produkt zewnętrzny, czy też wewnętrzna aplikacja — zadaniem właściciela produktu jest zadbanie o to, aby wykonana praca, nawet gdyby dotyczyła ona wyłącznie zadań czysto inżynierskich, miała jak najwyższy poziom. Właściciel produktu, chcąc mieć pewność, że zespół buduje wyłącznie to, na czym mu zależy, aktywnie współpracuje z mistrzem młyna i zespołem deweloperskim — jest dostępny i może szybko odpowiadać na wszelkie pojawiające się pytania. Szczegółowy opis roli właściciela produktu znajduje się w rozdziale 9.

## Mistrz młyna

Mistrz młyna pomaga wszystkim zaangażowanym osobom w zrozumieniu i przestrzeganiu wartości, zasad i praktyk Scruma. Działa jako trener, przewodząc procesowi i pomagając zespołowi scrumowemu oraz pozostałej części organizacji w rozwinięciu ich własnej, wydajnej wersji Scruma. Zadaniem mistrza młyna jest również pomóc organizacji w trudnych momentach związanych ze zmianą procesu zarządzania podczas adaptacji Scruma.

<sup>1</sup> W niniejszej książce dla odróżnienia właściciel produktu jest przedstawiany na rysunkach jako postać męska, natomiast mistrz Scruma jako postać żeńska.

Mistrz młyna jako animator pomaga zespołowi w rozwiązywaniu problemów i wprowadzaniu ulepszeń w jego własnej implementacji Scruma. Jest odpowiedzialny za ochronę zespołu przed wpływami zewnętrznymi — podejmuje kierowniczą rolę przy usuwaniu **przeszkód**, które ograniczają produktywność (w sytuacji, kiedy poszczególni członkowie zespołu nie są w stanie usunąć tych przeszkód samodzielnie). Mistrz młyna nie ma uprawnień kierowniczych nad zespołem, zatem ta rola różni się od roli menedżera zespołu lub kierownika produkcji. Mistrz młyna ma działać jako lider, nie kierownik. Role menedżera operacyjnego i menedżera projektu omówię w rozdziale 13. Więcej na temat mistrza młyna dowiesz się z rozdziału 10.

## Zespół deweloperski

Tradycyjne metody produkcji oprogramowania mówią o różnych typach pracowników, takich jak architekci, programiści, testerzy, administratorzy baz danych, projektanci interfejsu użytkownika itd. Scrum definiuje rolę zespołu deweloperskiego, który nie jest niczym innym jak zbiorem tego typu ludzi odpowiedzialnych za projektowanie, budowanie i testowanie oczekiwaneego produktu.

Zespół deweloperski organizuje się samodzielnie, aby określić najlepszy sposób realizacji celu wyznaczonego przez właściciela produktu. W skład zespołu wchodzi zazwyczaj od pięciu do dwudzięciu osób, które wspólnie muszą posiadać wszystkie niezbędne cechy potrzebne do wyprodukowania dobrej jakości oprogramowania. Oczywiście Scrum może zostać użyty przy pracach produkcyjnych wymagających znacznie większych zespołów. Zamiast jednak tworzyć jeden zespół scrumowy o rozmiarze, powiedzmy, 35 osób, lepiej będzie podzielić go na cztery lub więcej zespołów scrumowych, z których każdy będzie zawierał zespół deweloperski o liczbie członków nie większej niż 9. Więcej na temat zespołu deweloperskiego znajdziesz w rozdziale 11., natomiast o koordynacji wielu zespołów przeczytasz w rozdziale 12.

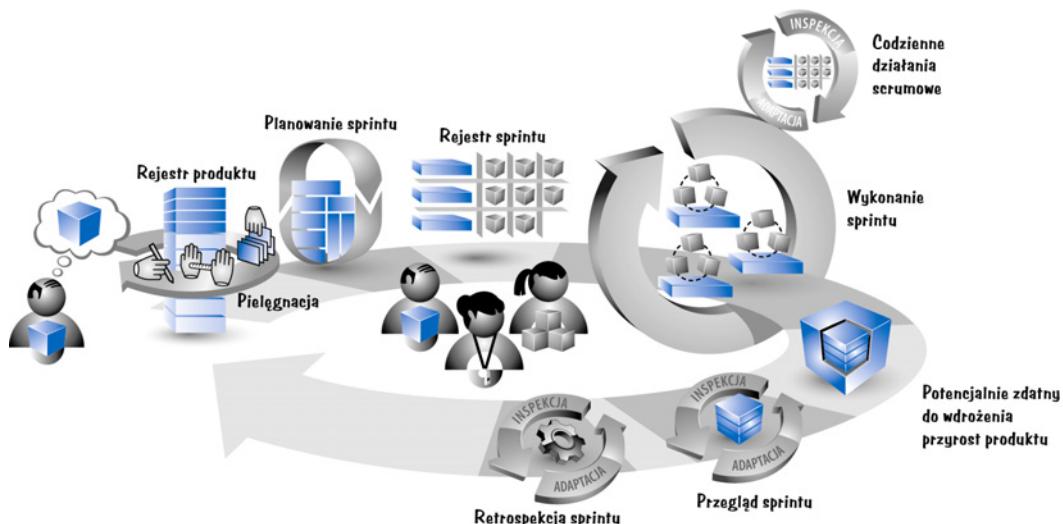
## Aktywności i artefakty Scruma

Rysunek 2.3 ilustruje większość z aktywności i artefaktów Scruma oraz ich wzajemne relacje.

Przeanalizujmy diagram z rysunku 2.3, zaczynając od postaci po lewej stronie, a następnie poruszając się zgodnie z ruchem wskazówek zegara wzdłuż głównej strzałki (reprezentując sprint).

Właściciel produktu ma wizję tego, co chciałby stworzyć (duży sześcian). Ten sześcian może być bardzo duży. Można go podzielić na mniejsze kawałki reprezentujące poszczególne cechy i zebrać w formie listy o ustalonych priorytetach zwanej rejestrtem produktu. Sama operacja dzielenia nazywana jest **pielęgnacją** dziennika produktu.

Sprint rozpoczyna się od fazy planowania sprintu, po której prowadzone są prace deweloperskie (wykonanie sprintu), a kończy się przeglądem i retrospekcją. Sam sprint przedstawiony jest w formie dużej pętli zakończonej strzałką, która wypełnia środek ilustracji. Liczba elementów w rejestrze produktu zazwyczaj jest o wiele większa od tego, co zespół deweloperski jest w stanie wykonać w ciągu jednej krótkiej iteracji. Z tego powodu na początku sprintu zespół deweloperski musi wskazać pewien podzbior elementów z rejestru produktu, który jego zdaniem może zostać zrealizowany podczas sprintu — ta aktywność, nazywana planowaniem sprintu, pokazana została na prawo od bryły przedstawiającej dziennik produktu.



RYSUNEK 2.3. Środowisko Scrum

Mała dygresja. Zmiana wprowadzona w roku 2011 do „Przewodnika po Scrumie” [Schwaber i Sutherland, 2011] spowodowała dyskusję na temat tego, czy wynik planowania sprintu powinien nazywać się **prognozą** (ang. *forecast*), czy **zobowiązaniem** (ang. *commitment*). Zwolennicy słowa *prognoza* woleli takie określenie, ponieważ czuli, że chociaż zespół deweloperski tworzy najlepsze możliwe oceny, na jakie może sobie pozwolić w danej chwili, ich wartości mogą ulec zmianie po uzyskaniu dodatkowej wiedzy w trakcie sprintu. Niektórzy uważali również, że zobowiązanie części zespołu może, w razie konieczności, prowadzić do zaniżania jakości lub rzeczywistych możliwości zespołu w celu zapewnienia pomyślnej realizacji tego, do czego zespół deweloperski zobowiązał się podczas planowania.

Zgadzam się z poglądem, iż zespoły deweloperskie powinny tworzyć prognozę (ocenę) tego, co mogą dostarczyć w trakcie sprintu. Niemniej jednak w przypadku wielu zespołów deweloperskich bardziej korzystne byłoby korzystanie z prognozy do określenia zobowiązania. Zobowiązanie pomaga w wytworzeniu wzajemnego zaufania pomiędzy właścicielem produktu i zespołem deweloperskim, a także we wnętrzu samego zespołu deweloperskiego. Ponadto zobowiązania są narzędziem wspierającym racjonalne planowanie krótkoterminowe i podejmowanie decyzji we wnętrzu organizacji, a w przypadku produkcji kilku produktów jednocześnie ułatwiają również zsynchronizowane planowanie — dany zespół może podejmować decyzje w oparciu o zobowiązania podjęte przez inne zespoły. W tej książce preferuję określenie *zobowiązanie*. Czasem używam również *prognozy*, gdy słowo to bardziej pasuje do kontekstu.

Dla uzyskania pewności odnośnie do zobowiązania podjętego przez zespół jego członkowie tworzą w trakcie planowania sprintu drugi rejestr, zwany rejestrtem sprintu. Rejestr sprintu opisuje poprzez zestaw szczegółowych **zadań**, w jaki sposób zespół planuje zaprojektować, zbudować i przetestować dany podzbiór cech z rejestru produktu w trakcie trwania tego konkretnego sprintu.

Następnie przychodzi czas na wykonanie sprintu, w trakcie którego zespół pracuje nad zadaniami niezbędnymi do wytworzenia określonych cech. Każdego dnia w trakcie wykonania sprintu członkowie zespołu zarządzają przepływem pracy, przeprowadzając synchronizację, inspekcję i adaptację

swoich poczynań — aktywności określane mianem codziennych działań scrumowych. Pod koniec wykonania sprintu zespół ma gotowy, potencjalnie zdany do wdrożenia przyrost produktu, który stanowi część, ale nie całość, wizji właściciela produktu.

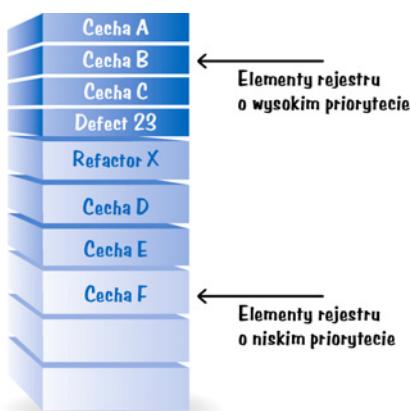
Zespół scrumowy kończy sprint, wykonując dwie aktywności polegające na inspekcji i adaptacji. Podczas pierwszej, zwanej przeglądem sprintu, interesariusze i zespół scrumowy analizują budowany produktu. Podczas drugiej, zwanej retrospekcją sprintu, zespół scrumowy przygląda się procesowi scrumowemu używanemu do wytwarzania produktu. Wynikiem obu tych aktywności może być podjęcie działań adaptacyjnych, które przełożą się na konkretne wpisy w dzienniku produktu lub zmiany w procesie deweloperskim.

W tym momencie cykl sprintu powtarza się, poczynając od ustalenia przez zespół deweloperski następnego najważniejszego podzbioru elementów rejestru produktu, jakie mogą zostać zrealizowane w trakcie sprintu. Po odpowiednio dużej liczbie ukończonych sprintów wizja właściciela produktu zostanie zrealizowana i będzie można wdrożyć gotowe rozwiązanie.

W pozostałej części rozdziału omówię bardziej szczegółowo każdą z tych aktywności i artefaktów.

## Rejestr produktu

W Scrumie wykonujemy najpierw pracę o największej wartości. Właściciel produktu w oparciu o informacje uzyskane od pozostałych członków zespołu deweloperskiego oraz interesariuszy jest ostatecznie odpowiedzialny za określenie i zarządzanie kolejnością tej pracy oraz jasne komunikowanie jej kolejności w formie listy uporządkowanej według priorytetów, zwanej **rejestrem produktu** (patrz rysunek 2.4). W przypadku produkcji nowego produktu rejestr produktu zawiera początkowo cechy niezbędne do zrealizowania wizji właściciela produktu. W przypadku prac nad istniejącym produktem rejestr produktu może zawierać nowe cechy produktu, zmiany cech istniejących, błędy wymagające naprawy, poprawki techniczne itd.



RYSUNEK 2.4. Rejestr produktu

Właściciel produktu współpracuje z wewnętrznymi i zewnętrznymi interesariuszami w celu zebrań i zdefiniowania elementów rejestru produktu. Do jego obowiązków należy zapewnienie odpowiedniej kolejności elementów rejestru (w oparciu o czynniki takie jak wartość, koszt, wiedza i ryzyko)

tak, aby elementy o dużej wartości pojawiły się na szczycie rejestru, a pozostałe wylądowały niżej. Rejestr produktu jest artefaktem, który nieustannie ewoluje. Elementy rejestru mogą być dodawane, usuwane lub modyfikowane przez właściciela produktu wraz ze zmianą uwarunkowań biznesowych, a także w wyniku wzrostu wiedzy (poprzez informacje odkryte w trakcie kolejnych iteracji) na temat produktu wśród członków zespołu scrumowego.

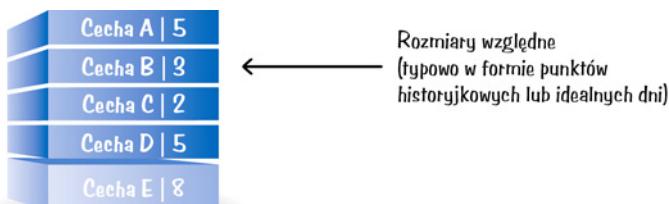
Wszelka działalność związana z tworzeniem i uszczegółowianiem elementów rejestru produktu, ocenianiem ich i nadawaniem priorytetów określana jest mianem pielęgnacji (patrz rysunek 2.5).



**RYSUNEK 2.5.** Pielęgnacja rejestru produktu

Krótką dygresja. W 2011 roku odbyła się kolejna debata nad słownictwem użyтыm w „Przewodniku po Scrumie” [Schwaber i Sutherland, 2011]. Zastanawiano się, czy elementy w rejestrze produktu należy określić mianem *uporządkowanych według priorytetów* (ang. *prioritized*) lub *posortowanych* (ang. *sorted*). Argumentowano, iż uporządkowanie według priorytetów jest zwyczajnie jedną z form sortowania (według niektórych nawet nie najważniejszą). Uporządkowanie elementów w rejestrze zależy od tak wielu czynników, że trudno znaleźć pojedyncze słowo, które w pełni odda całą koncepcję. Być może istnieje teoretyczne uzasadnienie debaty o wyższości jednego z tych określeń nad drugim, ale większość ludzi (ze mną włącznie) podczas dyskusji nad historyjkami z rejestrzu produktu używa obu terminów zamiennie.

Zanim zakończymy układanie według priorytetów, sortowanie lub też inną formę uporządkowania rejestrów, musimy poznać rozmiar każdego elementu w rejestrze (rysunek 2.6).

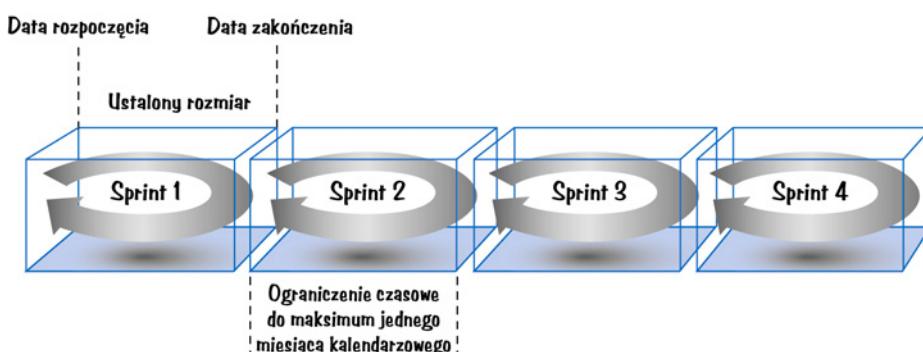


**RYSUNEK 2.6.** Rozmiary elementów w rejestrze produktu

Rozmiar przekłada się na koszt, który musi być znany właścicielom produktu, aby mogli oni odpowiednio ustawić priorytet elementu. Scrum nie określa miary, jakiej należy użyć do „wyceny” elementów produktu. W praktyce wiele zespołów stosuje **szacunek względny**, taki jak **punkty historyjkowe** lub **idealne dni**. Szacunek względny wyraża całkowity rozmiar elementu w sposób, który nie rozpatruje wartości absolutnej, a jedynie rozmiar względny w odniesieniu do innych rozpatrywanych elementów. Dla przykładu: cecha C na rysunku 2.6 ma rozmiar 2, a cecha E rozmiar 8. Możemy stąd wyciągnąć wniosek, iż cecha E jest około cztery razy większa od cechy C. Tego typu miarami przyjrzymy się dokładniej w rozdziale 7.

## Sprinty

Praca w Scrumie odbywa się na zasadzie iteracji lub cykli trwających nie dłużej niż jeden miesiąc kalendarzowy. Każdy taki cykl określany jest mianem **sprintu** (patrz rysunek 2.7). Praca ukończona w każdym sprintie powinna być pewną widoczną wartością dla klienta lub użytkownika.

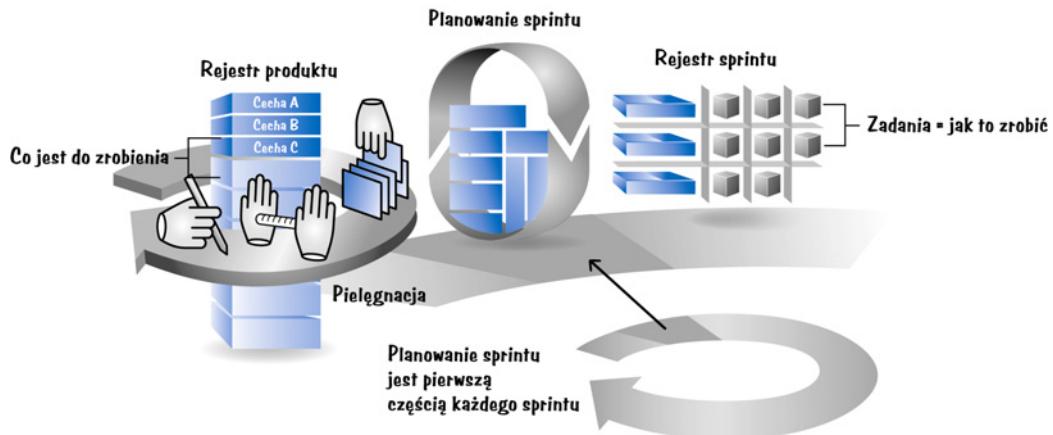


**RYSUNEK 2.7.** Charakterystyka sprintu

Sprinty są zawsze **ograniczone czasowo**, czyli posiadają ustaloną datę rozpoczęcia i zakończenia i powinny trwać jednakowo długo. Nowy sprint następuje bezpośrednio po zakończeniu sprintu poprzedniego. Z zasady nie pozwalamy w trakcie sprintu na zmiany zakresu realizowanej pracy lub składu zespołu, chociaż potrzeby biznesowe mogą uniemożliwić przestrzeganie tych wymogów. Sprinty zostaną omówione szczegółowo w rozdziale 4.

## Planowanie sprintu

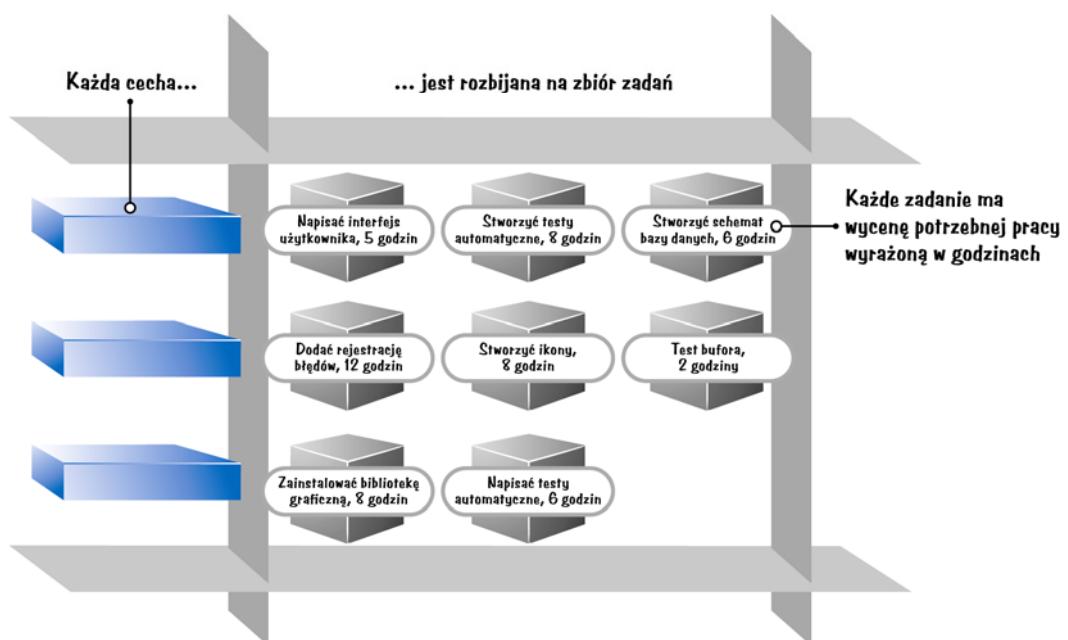
Rejestr produktu powinien przedstawiać sobą wiele tygodni lub miesięcy pracy, czyli znacznie więcej, niż można wyprodukować w ciągu pojedynczego, krótkiego sprintu. Chcąc określić podzbior najważniejszych elementów rejestru produktu przeznaczonych do wykonania w ciągu nadchodzącego sprintu, właściciel produktu, zespół deweloperski oraz mistrz młyna przeprowadzają **planowanie sprintu** (patrz rysunek 2.8).



RYSUNEK 2.8. Planowanie sprintu

Podczas planowania sprintu właściciel produktu i zespół deweloperski zgadzają się na **cel sprintu**, który definiuje to, co chcemy osiągnąć poprzez realizację nadchodzącego sprintu. Używając tego celu, zespół deweloperski przegląda rejestr produktu i wskazuje elementy o wysokim priorytecie, które jest w stanie realistycznie zrealizować w nadchodzącym sprintie, pracując w **podtrzymywalnym tempie** — z prędkością zapewniającą komfort pracy w długim okresie.

Chcąc zyskać pewność, co może być faktycznie wykonane, wiele zespołów dzieli każdą z cech przeznaczonych do realizacji na zbiór zadań. Zbiór tych zadań razem z towarzyszącymi im elementami z rejestru produktu tworzy drugi rejestr, zwany **rejestrem sprintu** (patrz rysunek 2.9).



RYSUNEK 2.9. Rejestr sprintu

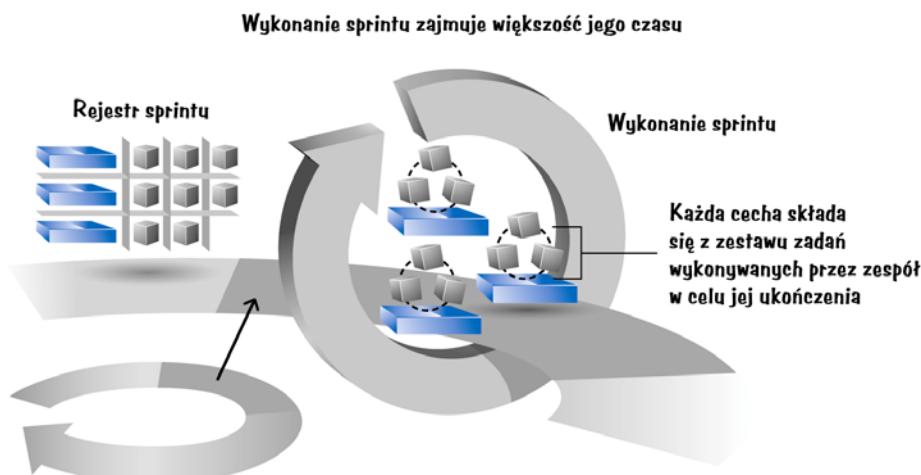
Następnie zespół deweloperski „wycenia” (zazwyczaj w godzinach) wysiłek niezbędny do wykonania każdego z zadań. Dzielenie elementów z rejestru produktu na zadania jest formą projektowania i planowania w samą porę sposobu realizacji cech produktu.

Większość zespołów scrumowych, których sprinty trwają od dwóch tygodni do jednego miesiąca kalendarzowego, stara się przeprowadzić planowanie sprintu w ciągu czterech do sześciu godzin. Zaplanowanie tygodniowego sprintu powinno zająć nie więcej niż dwie godziny (a prawdopodobnie jeszcze krócej). W tym czasie można skorzystać z kilku różnych podejść — stosowane przeze mnie opiera się na prostym cyklu wybrania elementu z rejestru produktu (jeśli to tylko możliwe, następnego o najwyższym priorytecie przypisanym przez właściciela produktu), rozbicia go na zadania i sprawdzenia, czy zmieści się on rozsądnie wewnętrz sprintu (razem z innymi elementami przeznaczonymi na ten sprint). Jeśli się zmieści i pozostałe jeszcze miejsce na wykonanie innej pracy, powtarzam cały cykl do momentu, kiedy zespołowi zabraknie mocy produkcyjnej.

Innym podejściem byłoby wybranie przez właściciela produktu i zespół deweloperski wszystkich przeznaczonych do realizacji elementów z rejestru produktu za jednym razem. Wtedy zespół deweloperski mógłby samodzielnie podzielić je na zadania, aby przekonać się, że mogą one zostać faktycznie zrealizowane. Każde z tych podejść opiszę dokładniej w rozdziale 19.

## Wykonanie sprintu

Kiedy zespół scrumowy zakończy planowanie sprintu i ustali zawartość dla następnej iteracji, zespół deweloperski wspomagany radą i pomocą mistrza młyna wykonuje kolejne zadania aż do ukończenia nowych cech (patrz rysunek 2.10). Ukończenie oznacza, że zespół jest pewny realizacji wszelkiej niezbędnej pracy potrzebnej do wyprodukowania cech o dobrej jakości.



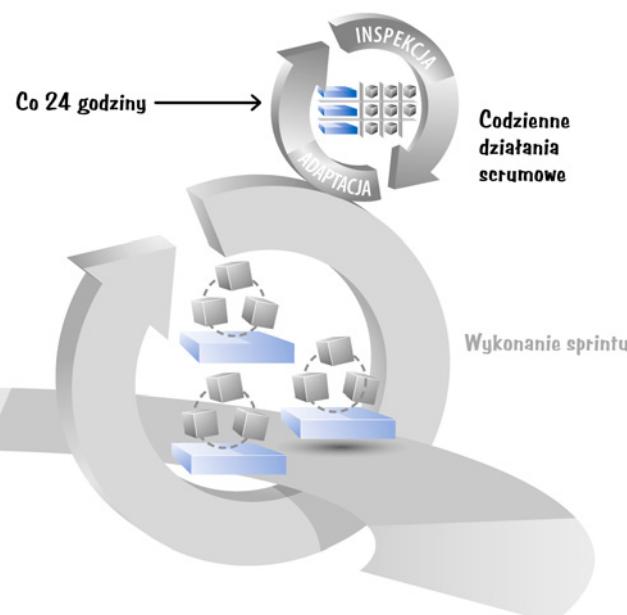
RYSUNEK 2.10. Wykonanie sprintu

Rodzaj wykonywanych zadań zależy oczywiście od realizowanego celu (na przykład czy budujemy oprogramowanie, a jeśli tak, jakiego typu jest to oprogramowanie, czy budujemy sprzęt, czy też jest to praca marketingowa?).

Nikt nie narzuca zespołowi deweloperskiemu kolejności oraz sposobu realizacji zadań z rejestru sprintu. To członkowie zespołu organizują samodzielnie pracę, przydzielając sobie zadania w sposób, który ich zdaniem daje najlepsze szanse realizacji celu sprintu. Więcej na temat wykonania sprintu przeczytasz w rozdziale 20.

## Codzienne działania scrumowe

Każdego dnia sprintu, najlepiej o tej samej porze, zespół deweloperski przeprowadza krótkie (trwające 15 minut lub mniej) **działania scrumowe** (patrz rysunek 2.11). Ta aktywność inspekcyjno-adaptacyjna nazywana jest czasem **codziennym spotkaniem na stojąco** (ang. *daily stand-up*), ponieważ w jego trakcie wszyscy uczestnicy zachowują postawę stojącą, co promuje zwiążłość spotkania.



RYSUNEK 2.11. Codzienne działania scrumowe

Bardzo często codzienne działania scrumowe przeprowadzane są z udziałem mistrza młyna jako animatora zachęcającego kolejno każdego członka zespołu do odpowiedzenia na trzy pytania:

- Co udało ci się osiągnąć od ostatniego spotkania?
- Nad czym zamierzasz pracować do czasu kolejnego spotkania?
- Jakie przeszkody lub utrudnienia stoją na przeszkodzie do osiągnięcia postępu w pracy?

Udzielenie odpowiedzi na powyższe pytania pozwala każdemu na zrozumienie ogólnej sytuacji w sprintie, przekonanie się o postępach w realizacji celu sprintu, a także poznanie wszelkich zamiarów modyfikacji planów na nadchodzący dzień pracy oraz ewentualnych problemów, którym trzeba zaradzić. Codzienne działania scrumowe w sposób nieoceniony pomagają zespołowi deweloperskiemu w zarządzaniu szybkim i zmiennym nurtem pracy w trakcie sprintu.

Celem codziennych działań scrumowych nie jest rozwiązywanie problemów. Jeśli takie istnieją, wiele zespołów decyduje się rozmawiać o nich po spotkaniu scrumowym w małym gronie zainteresowanych osób. Nie jest to również tradycyjne spotkanie w celu zdania statusu zwoływane przez menedżerów projektu, aby poznać aktualny stan projektu. Codzienne działanie scrumowe pozwala raczej zakomunikować stan elementów z rejestru sprintu innym członkom zespołu deweloperskiego — jest to realizowana codziennie aktywność inspekcjno-adaptacyjna i synchronizacyjna, która pozwala samoorganizującemu się zespołowi na lepsze wykonywanie pracy.

W Scrumie stosowane były pojęcia „świn” i „kurczaków” (ale wyszły z użytku) do odróżnienia osób, które powinny brać udział w codziennym spotkaniu scrumowym, od osób będących zwykłymi obserwatorami. Nazwy zwierząt zostały zapożyczone ze starego dowcipu (występującego w kilku różnych wariantach): „Do jajecznicy na bekonne potrzebne jest zaangażowanie kury i poświęcenie świń”. Najwyraźniej intencją użycia tych określeń w Scrumie było rozróżnienie osób zaangażowanych (kurczaków) od osób poświęcających się dla realizacji celu sprintu (świn). Podczas codziennego spotkania scrumowego powinny odzywać się jedynie świnie, kurczaki, jeśli takowe są, powinny uczestniczyć w formie obserwatorów.

W moim przekonaniu wszyscy członkowie zespołu scrumowego są świniami, natomiast pozostałe osoby kurczakami. Nie każdy zgadza się z tym poglądem. Na przykład właściciel produktu nie jest wymagany na codziennym spotkaniu scrumowym, dlatego czasem uważany jest za kurczaka (wynika to z prostej logiki: jak można się „poświęcać”, skoro nie ma wymogu bycia obecnym na spotkaniu?). To rozumowanie wydaje się dla mnie błędne, ponieważ nie potrafił sobie wyobrazić, jak właściciel produktu będący członkiem zespołu scrumowego może wykazywać mniejsze poświęcenie dla wyniku sprintu od zespołu deweloperskiego. Metafora świń i kurczaków załamuje się, jeśli spróbujemy zastosować ją wewnątrz zespołu scrumowego.

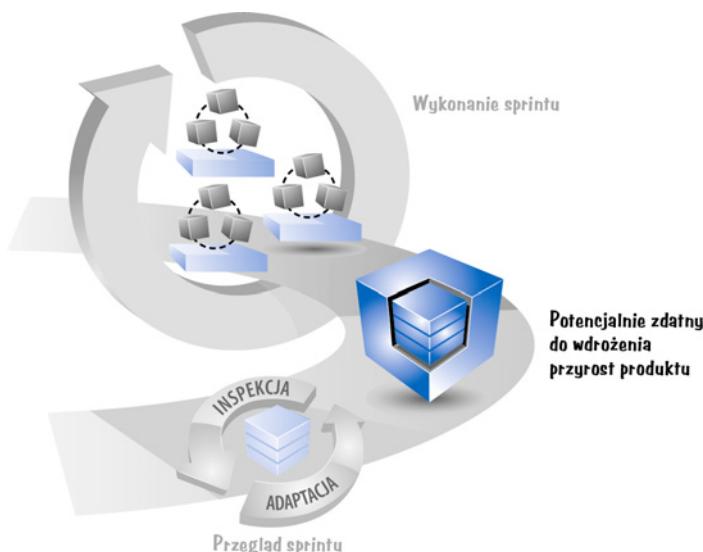
## Koniec pracy

W Scrumie wynik sprintu nazywamy **potencjalnie zdatnym do wdrożenia przyrostem produktu** (patrz rysunek 2.12) — oznacza to, że cokolwiek zespół scrumowy zgodził się zrealizować, zostało zrobione zgodnie z uzgodnioną definicją ukończenia. Ta definicja precyzuje poziom pewności odnośnie do dobrej jakości ukończonej pracy i jej potencjalnej zdatności do wdrożenia. Przykładowo: podczas produkcji oprogramowania minimalną definicją ukończenia jest dostarczenie kompletnego fragmentu funkcjonalności, która została zaprojektowana, zbudowana, zintegrowana i udokumentowana.

Agresywna definicja ukończenia pozwala firmie decydować po każdym sprintie, czy chce wypuścić produkt (lub wdrożyć wersję dystrybucyjną) dla wewnętrznych lub zewnętrznych klientów w oparciu o to, co zostało w nim zbudowane.

Dla pełnej jasności: „potencjalna zdatność do wdrożenia” nie oznacza, że to, co zbudowaliśmy, musi zostać wdrożone. Wdrożenie lub dystrybucja jest decyzją biznesową, często zależną od odpowiedzi na pytanie: „Czy mamy wystarczająco dużo nowych cech lub wystarczająco dużo zrealizowanych zadań klienta, aby usprawiedliwić wdrożenie?” lub „Czy nasi klienci są w stanie przyswoić kolejną zmianę, biorąc pod uwagę fakt, iż daliśmy im wersję dystrybucyjną dwa tygodnie temu?”.

Lepiej jest traktować potencjalną zdatność do wdrożenia jako stan pewności, że to, co zostało zbudowane w sprintie, jest faktycznie gotowe, czyli że wszystkie niezbędne prace (takie jak testowanie lub integracja), które muszą zostać wykonane, zanim będziemy mogli wdrożyć wyniki sprintu (jeśli tego zapragniemy), zostały zrobione.

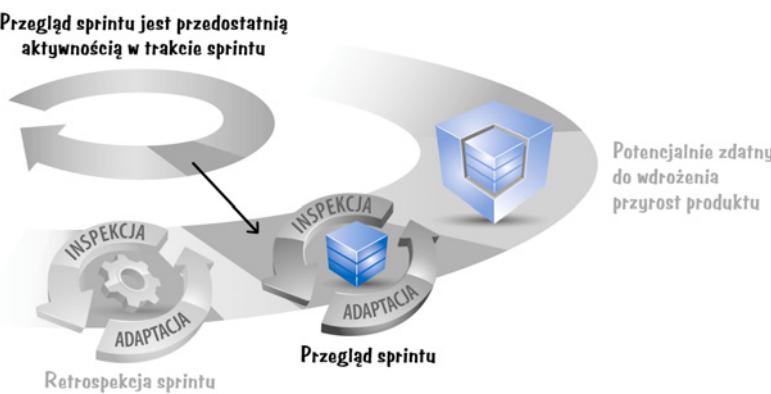


RYSUNEK 2.12. Wyniki sprintu (potencjalnie zdatne do wdrożenia przyrosty produktu)

W praktyce niektóre zespoły mogą modyfikować kryteria gotowości wraz z upływem czasu. Na przykład w początkowych fazach budowy gry posiadanie cech zdatnych do wdrożenia jest nie do końca wykonalne i potrzebne (szczególnie po uwzględnieniu odkrywczego procesu wczesnej fazy produkcji gry). W takich sytuacjach odpowiednią definicją ukończenia może być fragment funkcjonalności dostatecznie sprawny i używalny, aby móc wygenerować informację zwrotną pozwalającą zespołowi na podjęcie decyzji, jaką pracę wykonać w następnej kolejności lub jak ją wykonać. Więcej na temat definicji ukończenia prac znajdziesz w rozdziale 4.

## Przegląd sprintu

Pod koniec sprintu wykonywane są dwie dodatkowe aktywności inspekcyjno-adaptacyjne. Jedna z nich nosi nazwę **przegląd sprintu** (patrz rysunek 2.13).



RYSUNEK 2.13. Przegląd sprintu

Celem tej aktywności jest inspekcja i adaptacja produktu będącego w procesie produkcji. Podczas przeglądu sprintu kluczowe znaczenie ma dyskusja pomiędzy mistrzem młyna, interesariuszami, sponsorami, klientami i zainteresowanymi tematem członkami innych zespołów. Celem dyskusji jest przegląd ukończonych właśnie cech w kontekście całego wysiłku deweloperskiego. Każdy z uczestników spotkania ma wgląd w prezentację i może pomóc w wytyczeniu ścieżki dalszych prac deweloperskich, zapewniając tym samym stworzenie rozwiązania o jak najlepszych war狼orach biznesowych.

Pomyślnie przeprowadzony przegląd cechuje się dwukierunkowym przepływem informacji. Ludzie niebędący członkami zespołu scrumowego dowiadują się o przeprowadzonych pracach deweloperskich i pomagają wyznaczyć ich dalszy kierunek. Zespół scrumowy z kolei ma szansę podnieść wartość biznesową i marketingową produktu na podstawie informacji zwrotnej odnośnie do zbieżności produktu z celami zadowolonych klientów i użytkowników. Tym samym przegląd sprintu stanowi zaplanowaną okazję do inspekcji i adaptacji produktu. W praktyce osoby spoza zespołu scrumowego mogą przeprowadzić przegląd cech wewnętrz sprintu i podzielić się swoją opinią, pomagając w ten sposób zespołowi w osiągnięciu jeszcze lepszego wyniku sprintu. Więcej na temat przeglądu sprintu przeczytasz w rozdziale 21.

## Retrospekcja sprintu

Druga aktywność inspekcyjno-adaptacyjna pod koniec sprintu to **retrospekcja sprintu** (patrz rysunek 2.14). Często ma ona miejsce po przeglądzie sprintu i przed planowaniem kolejnego sprintu.



RYSUNEK 2.14. Retrospekcja sprintu

Podczas gdy przegląd sprintu jest czasem przeznaczonym na inspekcję i adaptację produktu, retrospekcja sprintu stanowi okazję do inspekcji i adaptacji procesu. W trakcie retrospekcji zespół deweloperski, mistrz młyna i właściciel produktu spotykają się w celu omówienia tego, co funkcjonuje lub nie funkcjonuje dobrze w Scrumie i związanych z nim procedurach technicznych. Celem jest nieustająca poprawa procesu, dzięki której dobry zespół scrumowy stanie się zespołem jeszcze lepszym. Pod koniec tego spotkania zespół scrumowy powinien móc zaprezentować listę problemów odkrytych podczas sprintu, a także zestaw działań naprawczych, jakie wdrożone zostaną przez zespół w kolejnym sprincie. Retrospekcję sprintu opisuję szczegółowo w rozdziale 22.

Po zakończeniu retrospekcji sprintu cały cykl jest powtarzany od nowa — zaczynając od sesji planowania kolejnego sprintu, której celem jest ustalenie pracy o najwyższym priorytecie w danym momencie. Praca ta będzie celem, na którym skupi się zespół.

## Zakończenie

W tym rozdziale opisane zostały fundamentalne praktyki Scruma, a także zamieszczono przekrojowy opis ról, aktywności i artefaktów środowiska. Istnieją jeszcze inne praktyki stosowane przez zespoły scrumowe, należą do nich między innymi planowanie na poziomie ogólnym i śledzenie postępu. Im poświęcone zostaną dalsze rozdziały. W następnym rozdziale opiszę fundamentalne zasady, na których bazuje Scrum. To umożliwi głębszą analizę środowiska Scrum w następnych rozdziałach.



# Rozdział 3

## ZASADY ZWINNOŚCI

---

Zanim zagłębimy się w mechanikę Scruma, dobrze będzie zrozumieć podstawowe zasady sterujące tą mechaniką.

Ten rozdział opisuje zasady zwinności, które stanowią podłoż Scruma, i porównuje je z tradycyjnymi zasadami sekwencyjnej i sterowanej planem produkcji oprogramowania. Takie podejście pozwala zrozumieć, w jaki sposób Scrum różni się od tradycyjnych metod produkcji, oraz bardziej szczegółowo przeanalizować praktyki Scruma w kolejnych rozdziałach.

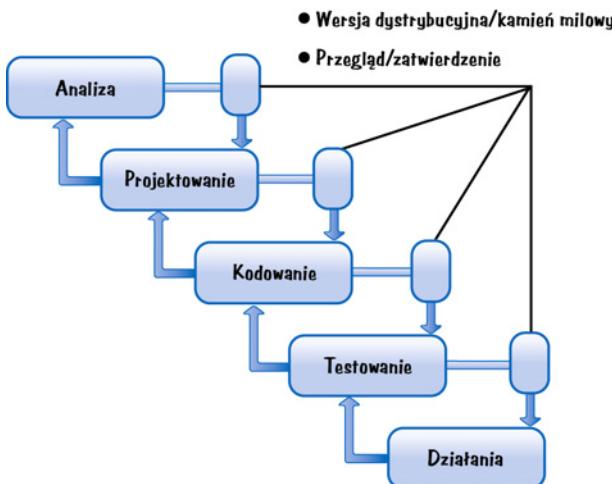
### Wprowadzenie

Uważam za bardzo pouczające przedstawienie reguł stojących u podstaw Scruma przez porównanie ich z przekonaniami, które rządzają tradycyjnym, sekwencyjnym modelem produkcji oprogramowania. W ten sposób łatwiej jest ludziom dostrzec podobieństwa oraz różnice pomiędzy Scrumem a modelem produkcji, który znają i rozumieją.

Celem porównania zasad zwinności z zasadami tradycyjnymi nie jest absolutnie stwierdzenie, iż produkcja metodą sekwencyjną i sterowaną planem jest zła, a metodą Scruma dobra. Oba podejścia należą do arsenalu narzędzi profesjonalnego producenta oprogramowania. Nie ma czegoś takiego jak złe narzędzie, są jedynie nieodpowiednie czasy do jego zastosowania. Jak wspomniałem o tym pokrótko w kontekście środowiska Cynefin (rozdział 1.), zarówno Scrum, jak i produkcja w sposób sekwencyjny, sterowany planem są odpowiednim podejściem dla różnych klas problemów.

Porównując oba podejścia, stosuję czysty lub też tzw. podręcznikowy opis produkcji metodą sekwencyjną, sterowaną planem. W ten sposób obrazując tradycyjny model produkcji, jestem w stanie lepiej wykazać różnice i bardziej przejrzystie zilustrować zasady stojące u podstaw produkcji opartej na Scrumie.

Jedna z form tradycyjnej produkcji sterowanej planem określana jest mianem **wodospadu** (patrz rysunek 3.1). Jest to jednak tylko jeden z przykładów szerszej klasy procesów **sterowanych planem** (znanych również pod nazwą **tradycyjnych, sekwencyjnych, antycypujących, przewidujących lub perspektywicznych procesów produkcyjnych**).



**RYSUNEK 3.1.** Proces wodospadowy

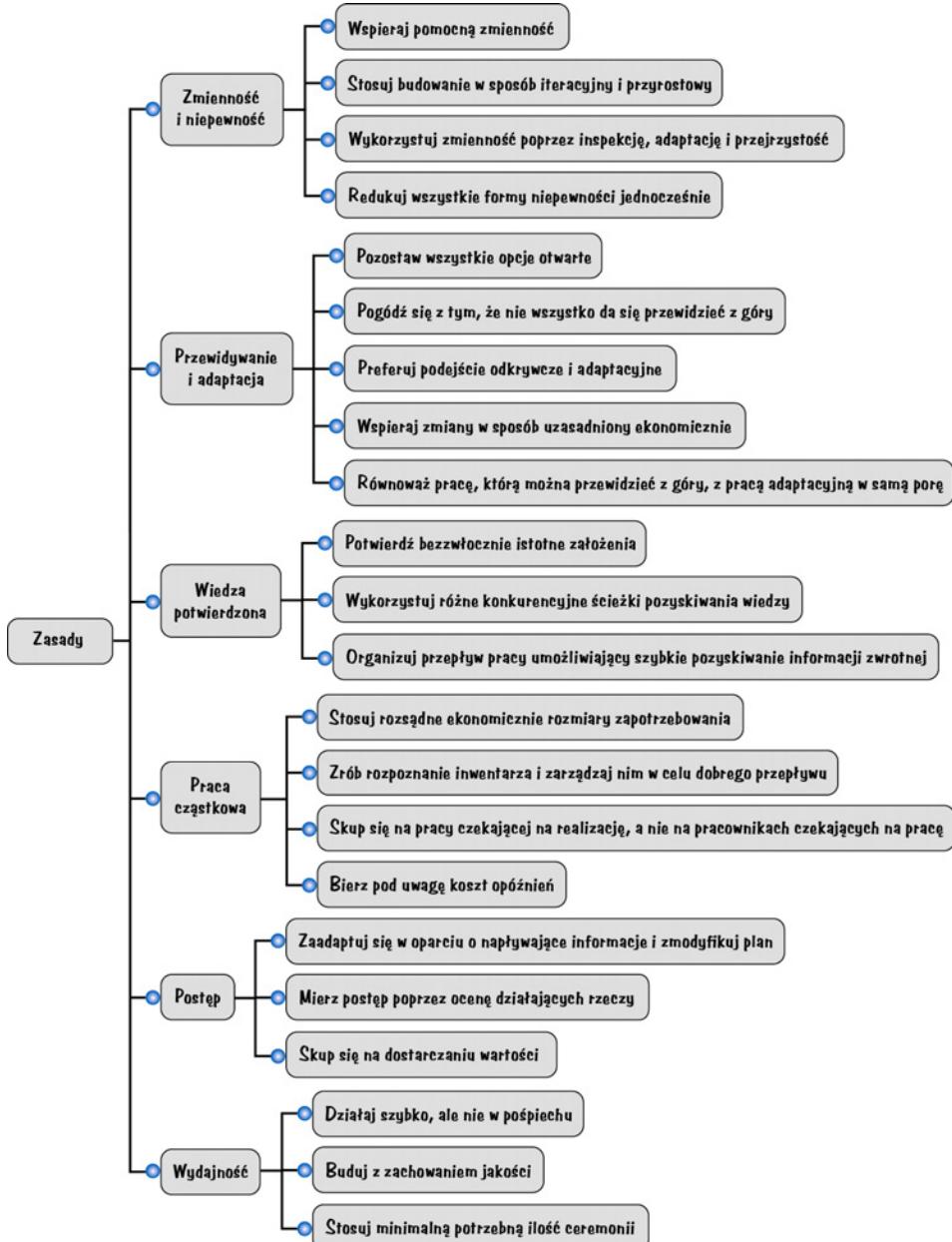
Procesy sterowane planem noszą taką nazwę, ponieważ usiłują z góry przewidzieć wszystkie możliwe cechy, jakie użytkownik chciałby mieć w produkcie końcowym, a także przewidzieć, w jaki sposób należy zbudować te cechy. Przyświeca tutaj idea: im lepsze planowanie, tym lepsze zrozumienie i lepsze wykonanie. Procesy sterowane planem nazywane są często planami sekwencyjnymi, ponieważ uczestnicy wykonują kolejno (sekwencyjnie) pełną analizę potrzeb, następnie kompletny projekt, kodowanie, budowanie i w końcu testowanie.

Produkcja sterowana planem sprawdza się w przypadku problemów dobrze zdefiniowanych, przewidywalnych, o małym ryzyku zajścia znaczących zmian. Niestety większość wysiłków związanych z produkcją oprogramowania ma niewiele wspólnego z przewidywalnością, szczególnie na samym początku. Zatem chociaż proces sterowany planem daje wrażenie podejścia uporządkowanego, pewnego i mierzalnego, to wrażenie może prowadzić do fałszywego poczucia bezpieczeństwa. Rzadko kiedy wytwarzanie oprogramowania przebiega zgodnie z planem.

Dla wielu proces sekwencyjny, sterowany planem, zwyczajnie ma sens: zrozumieć, zaplanować, zakodować, przetestować i wdrożyć — wszystko zgodnie z dobrze zdefiniowanym, przewidzianym planem. Istnieje przekonanie, że to powinno działać. Jeżeli zastosowanie podejścia sterowanego planem nie działa, przyjmuje się powszechnie, iż musielibyśmy zrobić coś źle. Nawet jeśli proces sterowany planem wielokrotnie produkuje niezadowalające wyniki, wiele organizacji nie rezygnuje z jego stosowania w przekonaniu, że jeśli tylko będą go lepiej wykonywać, wyniki poprawią się. Problemem nie jest jednak samo wykonanie, ale podejście bazujące na zbiorze założeń, które nie idą w parze z niepewnością będącą nierozerlączną częścią większości wysiłków deweloperskich.

Scrum dla odmiany bazuje na zupełnie innym zbiorze założeń — pasujących dobrze do problemów z dużą niepewnością, w przypadku których bardzo trudno jest przeprowadzić jakiekolwiek planowanie wysokiego poziomu. Zasady opisywane w tym rozdziale pochodzą z wielu różnych źródeł, włączając w to Manifest Agile [Beck i in., 2001], szczupiej produkcji oprogramowania [Reinertsen, 2009b; Poppendieck i Poppendieck, 2003] i „Przewodnik po Scrumie” [Schwaber i Sutherland, 2011].

Wspomniane zasady dzielą się na kilka kategorii — patrz rysunek 3.2.

**RYSUNEK 3.2.** Podział zasad na kategorie

Zacznę od przedyskutowania zasad wykorzystujących zmienność i niepewność — cechy nieodzworne związanego z procesem produkcyjnym. Następnie przejdę do zasad związanych z równoważeniem przewidywania w przód i adaptacją w samą porę. Później przyjdzie pora na zasady związane z nauką i zarządzaniem pracą cząstkową. Na zakończenie przyjrzę się zasadom dotyczącym postępu i wydajności.

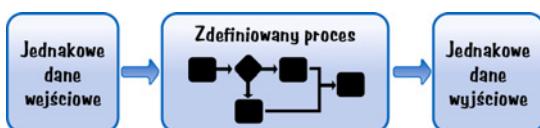
## Zmienna i niepewność

Scrum wykorzystuje **zmiennosć** i **niepewnośc** występujące w produkcji do tworzenia rozwiązań innowacyjnych. Opiszę cztery zasady związane z tym tematem:

- Wspieraj pomocną zmienność.
- Stosuj budowanie w sposób iteracyjny i przyrostowy.
- Zyskuj przewagę nad zmiennością poprzez inspekcję, adaptację i przejrzystość.
- Redukuj wszystkie formy niepewności jednocześnie.

## Wspieraj pomocną zmienność

Procesy sterowane planem traktują produkcję oprogramowania jak zwykłą produkcję przemysłową — unikają zmienności i promują zgodność ze **zdefiniowanym procesem**. Problem polega jednak na tym, że produkcja oprogramowania różni się znacznie od wytwarzania innych typów produktów. W produkcji przemysłowej naszym celem jest wziąć ustalony zbiór wymogów, a następnie wykonać serię dobrze zdefiniowanych kroków, aby wyprodukować za każdym razem produkt o powtarzalnych, identycznych cechach (z pewnym dopuszczalnym marginesem zmienności) — patrz rysunek 3.3.



**RYSUNEK 3.3.** Proces zdefiniowany

Przy produkcji oprogramowania celem jest stworzenie unikatowej *pojedynczej instancji* produktu, a nie jego *wytwarzanie na masową skalę*. To pojedyncza instancja przypomina pewien unikatowy przepis. Nie chcemy stworzyć go dwukrotnie — jeśli tak jest, to zmarnowaliśmy nasze pieniądze. Przeciwne, to, czego chcemy, to unikatowy przepis na nowy produkt. Niezbędny jest pewien stopień zmienności, aby za każdym razem wytworzyć inny produkt. W praktyce każda cecha, którą budujemy w naszym produkcie, różni się od wszystkich pozostałych, więc nawet na tym poziomie potrzebna jest zmienność. Dopiero kiedy mamy ten przepis, możemy przystąpić do masowego wytwarzania produktu — w przypadku oprogramowania ta produkcja będzie się sprowadzać do banalnego kopowania bitów.

Pomimo powyższych stwierdzeń pewne koncepcje masowej produkcji mają zastosowanie przy wytwarzaniu produktów i mogą, a nawet powinny być wykorzystywane. Przykładem są rozpoznanie i zarządzanie **inwentarzem** — zadania kluczowe dla produkcji masowej i równie istotne podczas wytwarzania produktu (będę o nich pisał już wkrótce). Jednak ze względu na samą naturę pracy wytwarzanie produktu i produkcja masowa są działaniami zupełnie odmiennymi i jako takie wymagają zupełnie innych procesów.

## Stosuj budowanie w sposób iteracyjny i przyrostowy

Produkcja w sposób sekwencyjny, sterowany planem zakłada, że uda nam się dobrze wykonać wszystkie rzeczy za pierwszym razem i że większość elementów produktu będzie do siebie pasować w fazie końcowej.

Scrum dla odmiany bazuje na **produkcji w sposób iteracyjny i przyrostowy**. Chociaż oba te określenia są często używane, tak jakby stanowiły jedną koncepcję, w rzeczywistości produkcja w sposób iteracyjny różni się od produkcji w sposób przyrostowy.

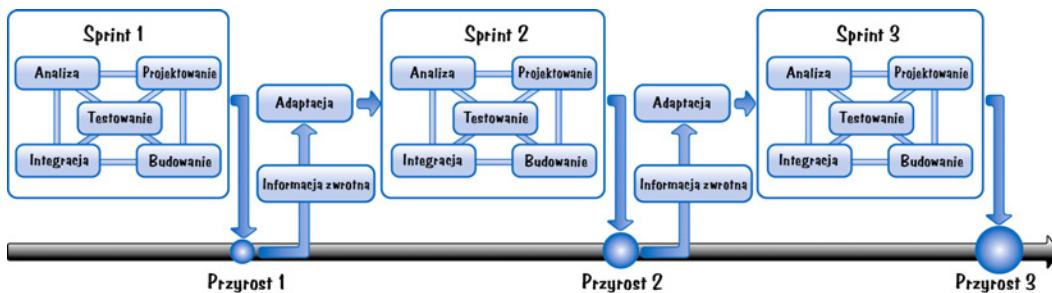
**Budowanie w sposób iteracyjny** przyjmuje, że prawdopodobnie wykonamy pewne elementy błędnie, zanim zrobimy je dobrze, a także że pierwsze wyniki naszej pracy będą niskiej jakości, a dopiero później zostaną ulepszone [Goldberg i Rubin, 1995]. Produkcja w sposób iteracyjny zakłada strategię ponownego wykonania. Stosujemy kilka przebiegów, aby ulepszyć to, co budujemy, i ostatecznie otrzymać dobre rozwiązanie. Możemy zacząć od stworzenia prototypu, aby zyskać pewną wiedzę na temat słabo znanego fragmentu produktu. Potem tworzymy wersję w pewnym stopniu lepszą od poprzedniej, a następnie być może jeszcze taką, która jest już całkiem dobra. W procesie powstawania tej książki na przykład pisałem, a następnie udoskonalałem kilka razy każdy z rozdziałów w miarę otrzymywania recenzji i wzrostu mojej świadomości odnośnie do tego, w jaki sposób chcę przekazać dany temat.

Produkcja w sposób iteracyjny jest doskonałym sposobem udoskonalenia produktu w trakcie prac nad nim. Największą wadą tego podejścia jest trudność w określeniu z góry (zaplanowaniu) ilości potrzebnych iteracji w obliczu niepewności.

**Budowanie w sposób przyrostowy** bazuje na niestarzejającej się zasadzie „Zbuduj fragment, zanim zbudujesz całość”. Unikamy jednego ogromnego zdarzenia pod koniec produkcji, kiedy wszystkie elementy łączone są ze sobą i cały produkt zostaje wdrożony. Zamiast tego dzielimy produkt na mniejsze fragmenty, aby móc zbudować część, uczymy się, w jaki sposób ta część funkcjonuje w swoim docelowym środowisku, dokonujemy adaptacji na podstawie zdobytej wiedzy, a następnie dobudowujemy kolejny fragment. Tworząc tę książkę, pisałem po jednym rozdziale i każdy gotowy wysyałem do oceny. Nie czekałem z wysyłką do momentu zakończenia całej książki. Dzięki temu otrzymywane recenzje pozwalały mi dostosować ton, styl oraz treść w kolejnych rozdziałach.

Budowanie w sposób przyrostowy pozwala nam uzyskać kluczową informację, która pozwala dostosować nasz wysiłek deweloperski oraz nasze podejście do problemu. Największym ryzykiem tego podejścia jest niebezpieczeństwo „przegapienia” obrazu całości poprzez budowanie w kawałkach (widzimy drzewa, ale nie widzimy lasu).

Scrum wykorzystuje zalety zarówno budowania iteracyjnego, jak i przyrostowego, odrzucając jednocześnie niedociągnięcia wynikające ze stosowania tych podejść osobno. Jest to możliwe dzięki użyciu obu idei w serii adaptacyjnych i ograniczonych czasowo iteracji zwanych sprintami (patrz rysunek 3.4).



RYSUNEK 3.4. Scrum używa budowania w sposób iteracyjny i przyrostowy

Podczas każdego sprintu wykonujemy wszystkie aktywności potrzebne do stworzenia działającego przyrostu produktu (jego części, nie całości). Jest to widoczne na rysunku 3.4 w postaci analizy, projektowania, budowania, integracji i testowania wykonywanych w każdym sprincie. To podejście w stylu „wszystko naraz” daje zysk w postaci szybkiego sprawdzania założeń przyjmowanych podczas wytwarzania cech produktu. Na przykład podejmujemy pewne decyzje projektowe, opierając się na nich, piszemy kod, a następnie testujemy zarówno nasz projekt, jak i jego fizyczne wykonanie — wszystko w tym samym sprincie. Wykonując całą potrzebną pracę w jednym sprincie, jesteśmy w stanie szybko poprawić cechy, osiągając korzyści wynikające z produkcji w sposób iteracyjny, bez konieczności jawnego planowania dodatkowych iteracji.

Zły wykorzystaniem koncepcji sprintów jest skupianie się w każdym z nich na jednym typie pracy — na przykład w sprincie pierwszym tylko na analizie, w sprincie drugim tylko na projektowania, w sprincie trzecim tylko na kodowaniu i w sprincie piątym na testowaniu. Takie podejście usiłuje nałożyć na Scrum kaskadowy styl pracy. To źle zrozumiane podejście określam często jako **WaterScrum** i wiem, że funkcjonuje również inne określenie — **Scrummerfall**.

W Scrumie nie pracujemy w danej chwili wyłącznie nad jedną fazą produkcji, pracujemy nad jedną cechą w danej chwili. Dzięki temu pod koniec sprintu mamy gotowy, wartościowy przyrost produktu (pewne, chociaż nie wszystkie cechy produktu). Przyrost zawiera w sobie wszelkie cechy stworzone wcześniej lub jest zintegrowany i przetestowany z nimi — jeśli tak nie jest, nie można powiedzieć, iż przyrost został ukończony. Dla przykładu: przyrost numer 2 na rysunku 3.4 zawiera cechy przyrostu numer 1. Pod koniec sprintu możemy uzyskać recenzję na temat nowo stworzonych cech w kontekście cech już istniejących. Dzięki temu uzyskujemy obraz z szerszej perspektywy.

Otrzymujemy informację zwrotną na temat wyników sprintu, dzięki czemu możemy się dostosować, wybierając inne cechy do produkcji w kolejnym sprincie lub zmieniając proces używany do ich budowy. W niektórych przypadkach może się okazać, że nasz przyrost produktu, chociaż poprawny z punktu widzenia wymogów technicznych, nie jest tak dobry, jak oczekiwano. Jeśli wystąpi taka sytuacja, możemy ponownie pracować nad wybranymi elementami w ramach naszego zobowiązania do nieustającej poprawy jakości i wydajności. W ten sposób pomagamy sobie w przezwyciężeniu braku odgórnnej wiedzy na temat ilości potrzebnych iteracji poprawiających jakość. Scrum nie wymaga od nas wskazywania jawnie liczby iteracji. Ciągła pętla zwrotnej informacji w trakcie produkcji przyrostowej będzie nam pomagać w dobraniu odpowiedniej i ekonomicznie uzasadnionej liczby iteracji.

## Wykorzystuj zmienność poprzez inspekcję, adaptację i przejrzystość

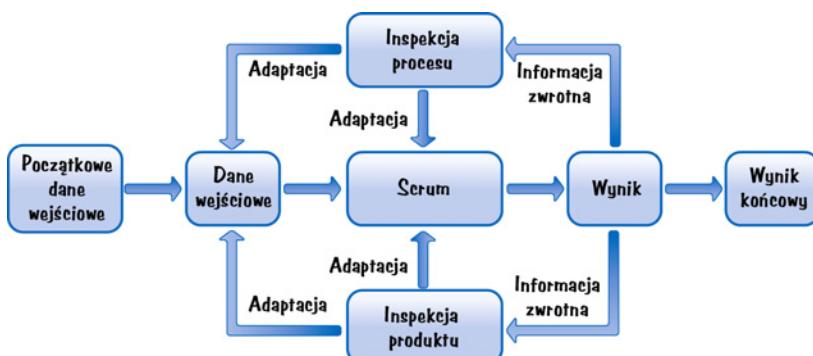
Pod pewnymi względami procesy sterowane planem i Scrum różnią się od siebie fundamentalnie (patrz tabela 3.1 — aspekty w oparciu o [Reinertsen, 2009a]).

Proces sekwencyjny i sterowany planem zakłada brak jakiekolwiek zmienności w wynikach produkcji. Stanowi on szereg dobrze zdefiniowanych kroków i korzysta z bardzo małej ilości informacji zwrotnych w trakcie swojego trwania. Scrum dla odmiany bazuje na założeniu, iż podczas wytwarzania produktu pewien poziom zmienności jest niezbędny do stworzenia czegoś nowego. Podejście to zakłada również, że proces potrzebny do stworzenia produktu jest złożony i dlatego przeciwstawia się próbom definiowania z góry. Ponadto Scrum generuje wczesną i częstą informację zwrotną, aby zapewnić, iż tworzony produkt oraz metoda jego produkcji są właściwe.

**TABELA 3.1.** Porównanie Scruma z procesami sterowanymi planem

Aspekt	Procesy sterowane planem	Scrum
Stopień zdefiniowania procesu	Zbiór dobrze zdefiniowanych kroków do wykonania w sposób sekwencyjny	Proces złożony, niemożliwy do całkowitego zdefiniowania w sposób odgórny
Przypadkowość wyników	Bardzo mała lub wręcz zerowa zmienność	Należy oczekiwać zmienności, ponieważ nie budujemy na okrągło tej samej rzeczy
Ilość wykorzystanych informacji zwrotnych	Bardzo mała i pojawiająca się na końcu	Regularna i wcześnie

W sercu Scruma znajdują się zasady **inspekcji, adaptacji i przejrzystości** (w opracowaniu [Schwaber i Beedle, 2001] określone zbiorczo mianem **empirycznej kontroli nad procesem**). W Scrumie dokonujemy inspekcji i adaptacji nie tylko tego, co budujemy, ale także jak to budujemy (patrz rysunek 3.5).

**RYSUNEK 3.5.** Model procesu scrumowego

Aby zrobić to dobrze, bazujemy na przejrzystości: wszelkie dostępne informacje potrzebne do stworzenia produktu muszą być dostępne dla osób zaangażowanych w jego produkcję. Przejrzystość umożliwia przeprowadzenie inspekcji i w konsekwencji — adaptację. Pozwala ona również wszystkim zainteresowanym na obserwację i zrozumienie tego, co się dzieje. Umożliwia lepszą komunikację i wprowadza zaufanie (zarówno do procesu, jak i pomiędzy członkami zespołu).

## Redukuj wszystkie formy niepewności jednocześnie

Wytwarzanie nowych produktów jest złożonym przedsięwzięciem o dużym stopniu niepewności. Tę niepewność można podzielić na dwie kategorie [Laufer, 1996]:

- **Niepewność końca** (niepewność typu *co*) — niepewność otaczająca cechy produktu końcowego.
- **Niepewność środków** (niepewność typu *jak*) — niepewność otaczająca proces i technologie używane do stworzenia produktu.

W niektórych środowiskach oraz w przypadku niektórych produktów może również występować **niepewność klienta** (niepewność typu *kto*). Na przykład firmy rozpoczynające swoją działalność (włączając w to duże organizacje skupiające się na nowych produktach) mogą jedynie zakładać, kim będą faktyczni klienci kupujący ich produkty. Ta niepewność musi zostać usunięta — w przeciwnym razie może okazać się, że doskonały produkt nie ma rynku zbytu.

Tradycyjny, sekwencyjny proces produkcji skupia się w pierwszej kolejności na usunięciu niepewności końca poprzez pełne zdefiniowanie tego, co będzie budowane, dopiero po tym przystępuje do analizy niepewności środków.

To proste, liniowe podejście do redukcji niepewności wadliwie współgra z produkcją odbywającą się w domenie złożonej, gdzie wszystkie nasze akcje oraz otoczenie, w którym działamy, zależą od siebie nawzajem. Na przykład:

- Decydujemy się zbudować pewną cechę (nasza akcja).
- Pokazujemy następnie tę cechę klientowi, który po jej zobaczeniu zmienia zdanie odnośnie do tego, czego faktycznie potrzebuje, lub uświadamia sobie, że niedostatecznie dobrze opisał nam swoje oczekiwania (nasza akcja spowodowała reakcję otoczenia).
- Zmieniamy projekt w oparciu o informację zwrotną (reakcja otoczenia zmusza nas do podjęcia akcji, której wcześniej nie przewidzieliśmy).

W Scrumie nie nakładamy na siebie obowiązku pełnego rozwiania jednego typu niepewności przed przystąpieniem do rozwiązywania niepewności innego typu. Zamiast tego podchodzimy bardziej holistycznie i skupiamy się na jednoczesnej redukcji wszelkich niepewności (konca, środków, klienta itp.). Oczywiście, w wybranych momentach możemy skupiać się bardziej na jednym typie niepewności bardziej niż na innych. Symultanicznej redukcji różnych typów niepewności sprzyja iteracyjny i przyrostowy proces produkcji sterowany nieustającą inspekcją, adaptacją i przejrzystością. Takie podejście pozwala nam na wytrwałą eksplorację naszego własnego środowiska w celu zidentyfikowania i poznania **nieznanych nieznanych** (nieznanych rzeczy, o których jeszcze nie wiemy) w trakcie, gdy te będą wypływać na światło dzienne.

## Przewidywanie i adaptacja

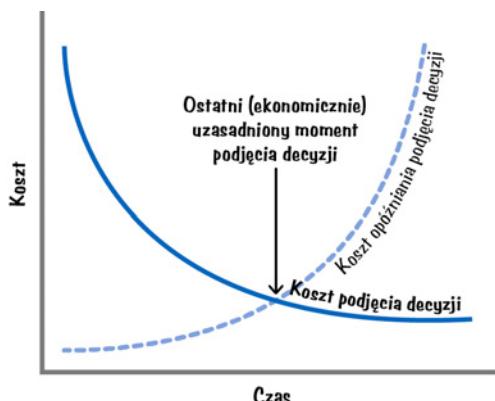
Używając Scruma, musimy nieustannie bilansować chęć przewidywania z potrzebą adaptacji. Opiszę pięć zasad związanych z tym tematem:

- Pozostaw wszystkie opcje otwarte.
- Pogódź się z tym, że nie wszystko da się przewidzieć z góry.
- Preferuj podejście odkrywcze i adaptacyjne.
- Wspieraj zmiany w sposób uzasadniony ekonomicznie.
- Równoważ pracę, którą można przewidzieć z góry, z pracą adaptacyjną w samą porę.

## Pozostaw wszystkie opcje otwarte

Produkcja sekwencyjna, sterowna planem wymaga, aby istotne decyzje w obszarach takich jak wymagania lub projekt zostały podjęte, przejrzone i zatwierdzone w odpowiednich dla siebie fazach. Dodatkowo każdą z tych decyzji należy podjąć przed przejściem do kolejnej fazy, nawet jeśli bazuje ona na niepełnej wiedzy.

Scrum twierdzi, iż nigdy nie powinniśmy podejmować przedwczesnej decyzji tylko i wyłącznie dlatego, że jakaś ogólna wytyczna procesu wskazuje, że właśnie nadeszła pora na jej podjęcie. Przeciwnie, w Scrumie preferujemy strategię trzymania naszych opcji otwartych. Ta zasada określana jest często mianem **ostatecznego momentu podjęcia decyzji** [Poppdeck i Popendeck, 2003] i oznacza, że opóźniamy nasze pełne zaangażowanie i nie podejmujemy ważnych i nieodwracalnych decyzji do ostatniego dopuszczalnego momentu w czasie. Kiedy nadchodzi ten czas? Kiedy koszt niepodjęcia decyzji przewyższy kosz jej podjęcia (patrz rysunek 3.6). Właśnie w tym momencie podejmujemy decyzję.



**RYSUNEK 3.6.** Podejmuj decyzje w ostatnim rozsądny momencie

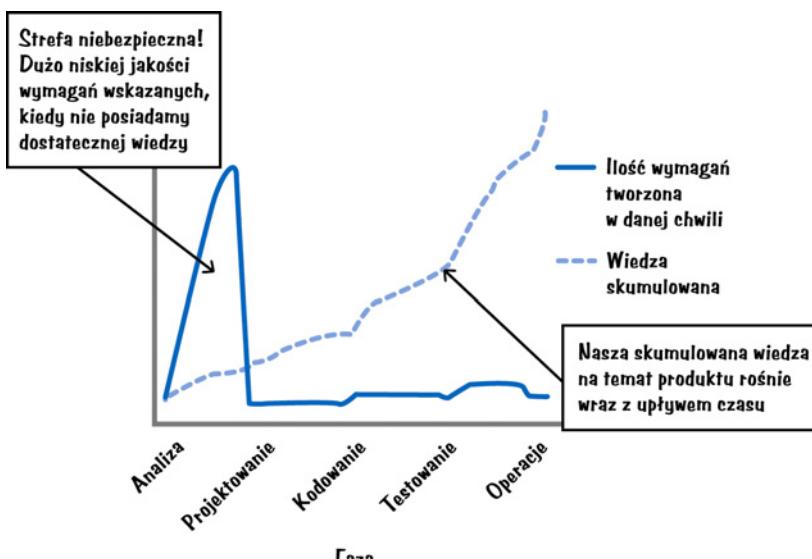
Aby docenić tę zasadę, rozważ następującą sytuację. Pierwszego dnia produkcji mamy najmniej informacji na temat tego, co faktycznie robimy. Każdego kolejnego dnia zyskujemy odrobinę więcej wiedzy. Dlaczego zatem zgodzilibyśmy się podjąć najbardziej istotne i być może nieodwracalne decyzje w ciągu pierwszych kilku dni? Większość z nas wolałaby poczekać do chwili, kiedy będziemy mogli podjąć decyzję bardziej świadomą. Jeśli zdecydujemy się podjąć ważną lub nieodwracalną decyzję i popełnimy błąd, nasze koszty w czasie zaczną rosnąć w postępie wykładniczym (patrz rysunek 3.6). W miarę lepszego rozumienia decyzji koszt jej podjęcia maleje (prawdopodobieństwo podjęcia złej decyzji maleje ze względu na wzrost pewności odnośnie do możliwości technicznych i handlowych). Właśnie dlatego powinniśmy zaczekać z podjęciem decyzji i tym samym zaangażowaniem się do momentu otrzymania dokładniejszych informacji.

## Pogódź się z tym, że nie wszystko da się przewidzieć z góry

Proces sterowany planem zakłada nie tylko znajomość pełnego zakresu wymogów i kompletnego projektu, ale również to, że jesteśmy w stanie „zrobić wszystko dobrze za pierwszym razem”. W rzeczywistości jest jednak mało prawdopodobne, że uda nam się z góry poprawnie zgromadzić wszystkie

wymagania czy też szczegółowe projekty dla tych wymagań. Co gorsza, jeśli te wymagania ulegną zmianom, będziemy musieli zmodyfikować wymogi bazowe i projekty tak, aby pasowały do nowych realiów (więcej na ten temat w rozdziale 5.).

W Scrumie przyjmujemy, że nie jesteśmy w stanie zgromadzić wszystkich wymogów lub projektów na samym początku projektu. Dodatkowo uważamy, że próba przeprowadzenia takich działań byłaby niebezpieczna, ponieważ z dużym prawdopodobieństwem zabrakłoby nam istotnej wiedzy, co doprowadziłoby do stworzenia sporej ilości wymagań o niskiej jakości (patrz rysunek 3.7).



**RYSUNEK 3.7.** Pozyskiwanie wymagań w odniesieniu do wiedzy o produkcie w przypadku procesu sterowanego planem

Wykres na rysunku 3.7 pokazuje, że w przypadku procesu sekwencyjnego, sterowanego planem duża liczba wymogów powstaje bardzo wcześnie, kiedy mamy bardzo małą wiedzę na temat produktu. Takie podejście jest ryzykowne, ponieważ powstaje błędne przekonanie, iż pozbyliśmy się niepewności końca. Jest to również działanie o dużym potencjale marnotrawstwa, który ujawnia się wraz ze wzrostem naszego rozumienia problematyki lub przy okazji zmian (o czym będę pisał już niedługo).

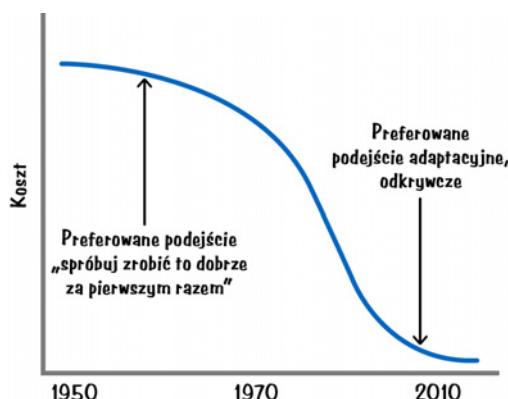
W przypadku Scruma również będziemy musieli przygotować pewne plany z góry, ale tylko tyle, ile będzie nam potrzeba na początek, i z założeniem, że wszelkie szczegóły dotyczące wymagań uzupełnimy, kiedy będziemy wiedzieć więcej na temat budowanego produktu. Nawet jeśli jesteśmy przekonani w 100 procentach, co budujemy i jak należy z góry zaplanować pracę, przekonamy się, że jesteśmy w błędzie, kiedy tylko dostarczymy nasze wczesne przyrosty produktu do środowiska, w którym będą musiały funkcjonować. W tym momencie ujawnią się wszystkie niewygodne dla nas realia dotyczące faktycznych potrzeb i zmuszą nas do podjęcia zmian.

## Preferuj podejście odkrywcze i adaptacyjne

Procesy sekwencyjne, sterowane planem skupiają się na tym (lub wykorzystują to), co jest aktualnie znane, i przewidują to, co jest nieznane. Scrum preferuje bardziej adaptacyjne podejście w stylu „spróbuj-i-polegnij” bazujące na odpowiednim wykorzystaniu eksploracji.

**Eksploracja** odnosi się do czasów, kiedy decydowaliśmy się zdobywać wiedzę poprzez pewną formę aktywności, taką jak budowanie prototypu, udowadnianie w praktyce pewnej koncepcji, przeprowadzenie badań naukowych lub wykonanie eksperymentu. Innymi słowy, kiedy napotykaliśmy niepewność, kupowaliśmy wiedzę poprzez eksplorację.

Koszt eksploracji jest zdecydowanie zależny od naszych narzędzi i technologii. Patrząc wstecz: eksploracja podczas produkcji oprogramowania była zawsze droga, ze względu na co faworyzowano bardziej przewidywalne podejście w stylu „spróbuj zrobić to dobrze za pierwszym razem” (patrz rysunek 3.8).



RYSUNEK 3.8. Historia kosztu eksploracji

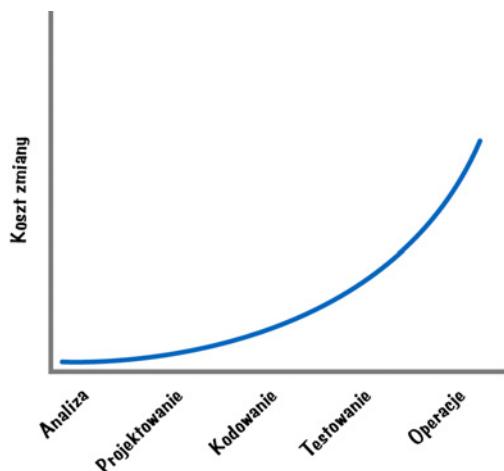
Oto pewien przykład. Na moim pierwszym roku studiów w Georgia Tech (wczesne lata osiemdziesiąte) używałem przez krótki czas kart perforowanych — narzędzi, które podobnie jak maszyna do pisania sprawiało, że nie cierpałem popełniać jakichkolwiek błędów lub wprowadzać prawek. Trudno było stosować w praktyce koncepcję „Spróbujmy szybko tego podejścia i zobaczymy, co się stanie” w sytuacji, kiedy każde potencjalne rozwiązanie wymagało przeprowadzenia mozolnego procesu perforowania kart, zapisania się w kolejce do komputera i odczekania do 24 godzin przed poznaniem wyniku. Nawet pojedyncza literówka powodowała przesunięcie w harmonogramie o jeden dzień. Zastosowanie procesu wodoszadowego, który pozwalał na dokładne przeanalizowanie bieżącej wiedzy i przewidywanie w obliczu niepewności z zamiarem osiągnięcia dobrego rozwiązania, wydawało się ekonomicznie uzasadnione.

Na szczęście narzędzia i technologie uległy polepszeniu, a koszt eksploracji znacznie zmalał. Nie ma już przeszkody ekonomicznej zniechęcającej do eksploracji. W praktyce obecnie często taniej jest budować coś, szybko adaptując się, w oparciu o informacje uzyskane od klienta, niż inwestować w próbę zrobienia wszystkiego dobrze za pierwszym razem. Jest to również dobra rzecz z punktu widzenia rosnącego stopnia skomplikowania kontekstu (otaczających nas technologii), w którym muszą funkcjonować nasze rozwiązania.

W Scrumie wykonujemy krok do przodu, tylko jeśli posiadamy dostateczną wiedzę i możemy podjąć świadomą i racjonalną decyzję. Natomiast w przypadku niepewności zamiast próbować przewidywać, używamy niskonakładowej eksploracji, aby kupić odpowiednie informacje, niezbędne do podjęcia odpowiedzialnej decyzji o przeniesieniu naszego rozwiązania do następnego etapu. Informacje, jakie uzyskamy w wyniku podjęcia tej akcji, pozwolą nam stwierdzić, czy niezbędna jest dalsza eksploracja.

## Wspieraj zmiany w sposób uzasadniony ekonomicznie

Jak zdążyliśmy się już przekonać, w sekwencyjnym procesie produkcyjnym zmiana kosztuje o wiele więcej w późnej fazie projektu niż w fazie wczesnej (patrz rysunek 3.9, źródło: [Boehm, 1981]).



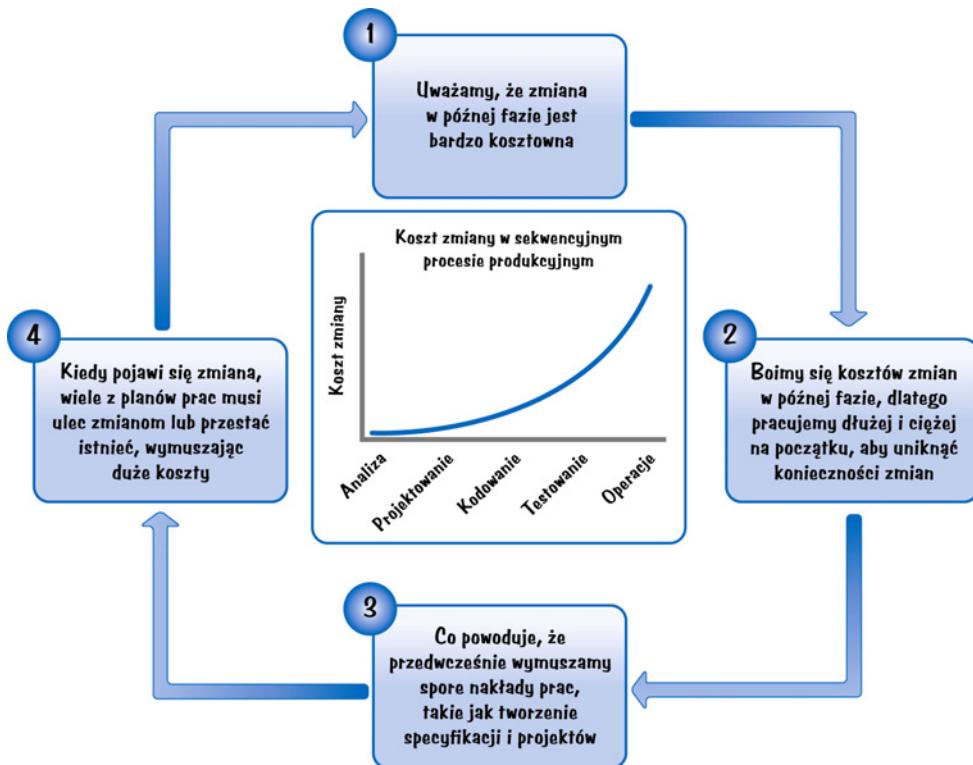
RYSUNEK 3.9. Znaczny wzrost kosztu zmiany w późnej fazie sekwencyjnego procesu produkcji

Przykładowo: zmiana przeprowadzona w trakcie analizy może kosztować parę złotych, ta sama zmiana w trakcie testowania może kosztować tysiące złotych. Skąd to wynika? Jeżeli popełnimy błąd podczas analizy i znajdziemy go w trakcie analizy, jego naprawienie będzie kosztować bardzo mało. Jeżeli ten sam błąd nie zostanie znaleziony aż do momentu wejścia w fazę projektowania, będziemy musieli nie tylko zmienić wymagania, ale również bazujące na nich części naszego projektu. To narastanie siły błędu trwa przez każdą kolejną fazę — rzecz wymagająca niewielkiej korekty w trakcie analizy urasta do ogromnego problemu podczas testowania lub operacji po zakończeniu produkcji.

Chcąc uniknąć zmian w fazie późnej, sekwencyjny proces produkcyjny stara się szczegółowo kontrolować i minimalizować wszelkie żądania zmian wymogów lub projektów poprzez nieustającą poprawę precyzyjności przewidywań odnośnie do tego, co projektowany system powinien robić lub jak powinien to robić.

Niestety przesadna przewidywalność we wczesnych fazach projektu często odnosi skutek przeciwny do zamierzzonego. Nie tylko nie pomaga w wyeliminowaniu zmian, ale stanowi również jeden z czynników prowadzących do opóźnień we wdrożeniach produktu i przekraczania założonego budżetu. Skąd bierze się ten paradoks? Po pierwsze, pragnienie wyeliminowania kosztowej zmiany zmusza nas do nadmiernego inwestowania w każdą fazę — wykonywania pracy ponad to, co należy

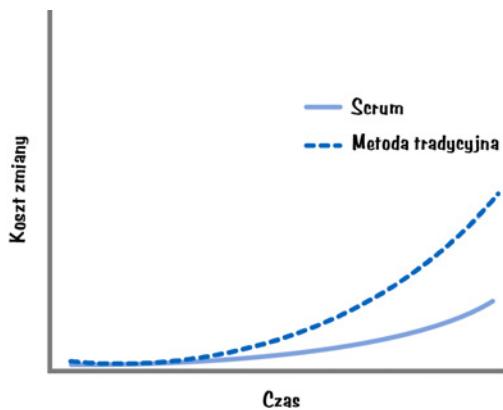
uznać za niezbędnie i praktyczne podejście do problemu. Po drugie, jesteśmy zmuszeni do podejmowania decyzji w oparciu o ważne założenia we wczesnych fazach projektu, zanim jeszcze mamy szansę potwierdzić ich słuszność na podstawie informacji uzyskanych od naszych interesariuszy dzięki już działającym zasobom. W konsekwencji wytwarzamy spory inwentarz pracy oparty na tych założeniach. Później, kiedy udowodnimy nasze założenia (ewentualnie zaprzeczymy im) lub dojdzie, jak to zwykle bywa, do zmiany (na przykład wyewoluują lub pojawią się nowe wymagania), ten inwentarz będzie musiał być poprawiony lub wręcz odrzucony. Wszystko to pasuje do znanego wzorca samospełniającej się przepowiedni (patrz rysunek 3.10).



**RYSUNEK 3.10.** Samospełniająca się przepowiednia

W Scrumie zakładamy, że zmiana to rzecz powszednia. Uważamy, że nie jesteśmy w stanie pozbyć się niepewności nierozerwalnie związanej z procesem wytwarzania produktu poprzez dłuższą i cięższą pracę. W związku z tym musimy być przygotowani na akceptowanie zmian, a kiedy te nastąpią, chcemy, aby ekonomia była dla nas bardziej przyjazna w porównaniu z tą, jaka ma miejsce w tradycyjnym procesie wytwórczym — nawet jeśli zmiana nastąpi w późnej fazie procesu.

Naszym celem jest zatem utrzymać krzywą kosztu zmian na niskim poziomie tak długo, jak tylko jest to możliwe, sprawiając tym samym, aby zmiana nawet w późnej fazie projektu była dla nas czymś całkowicie akceptowalnym z ekonomicznego punktu widzenia — ilustruje to rysunek 3.11.



**RYSUNEK 3.11.** Spłaszczanie krzywej kosztu zmiany

Możemy go osiągnąć poprzez zarządzanie ilością pracy cząstkowej oraz przepływem tej pracy w taki sposób, aby koszt zmiany w Scrumie był mniej związany z czasem, niż ma to miejsce w projektach sekwencyjnych.

Niezależnie od przyjętego podejścia do wytwarzania produktu chcemy, aby spełniona była następująca zależność: mała zmiana w wymaganiach powinna powodować proporcjonalnie małą zmianę w implementacji i co za tym idzie w kosztach (oczywiście spodziewamy się, że większa zmiana powiąga za sobą większy koszt). Inną oczekiwana przez nas właściwością tego związku jest jego prawdziwość bez względu na to, *kiedy* pojawi się żądanie zmiany.

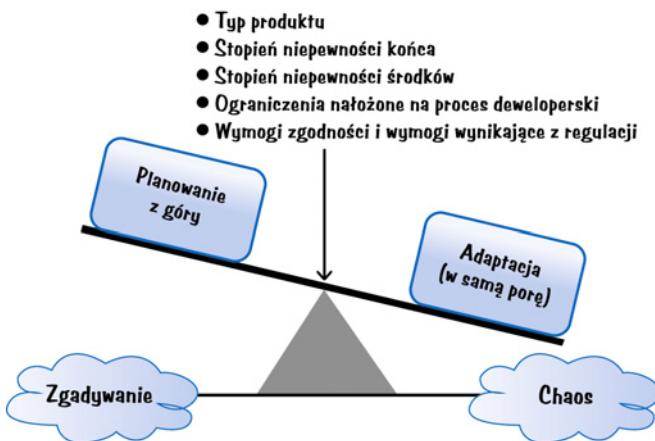
W Scrumie wiele z produktów pośrednich pracy (takich jak szczegółowa specyfikacja wymagań, projekty, studia przypadków) tworzymy w chwili, kiedy są one potrzebne (w samą porę), unikając w ten sposób generowania artefaktów potencjalnie niepotrzebnych. Dzięki temu kiedy nastąpi zmiana, zazwyczaj będziemy mieli o wiele mniej artefaktów lub ograniczających nas decyzji narzuconych przez założenia, które trzeba będzie zmodyfikować lub porzucić. To z kolei pozwoli utrzymać lepszą proporcję kosztu do rozmiaru zażąданej zmiany.

W sekwencyjnym modelu produkcji bardzo szybko rozpoczęte tworzenie artefaktów i presja na podejmowanie przedwczesnych decyzji oznaczają, że koszt zmian wraz z upływem czasu będzie gwałtownie rósł w miarę rozrastania się inventarza. Z tego względu na krzywej modelu tradycyjnego (rysunek 3.11) bardzo wcześnie pojawia się punkt przegięcia (miejscie, w którym linia wykresu zaczyna się gwałtownie wzrosnąć). W przypadku Scruma również pojawia się moment, w którym koszt zmian przestaje być proporcjonalny do rozmiaru zmian (krzywa dla Scruma na rysunku 3.11 ma również swój punkt przegięcia), ale ma on miejsce znacznie później.

## Równoważ pracę, którą można przewidzieć z góry, z pracą adaptacyjną w samą porę

Fundamentalna zasada produkcji sterowanej planem mówi, że przyjęte z góry założenia oraz szczegółowy plan mają absolutnie krytyczne znaczenie i powinny być przygotowane przed przejściem do kolejnych faz działania. W Scrumie uznajemy, że praca związana z planowaniem powinna być pomocna, ale nie może przytłaczać. Zgadzamy się, że niemożliwe jest precyzyjne określenie wymogów i planów z góry. Czy to oznacza, że powinniśmy zrezygnować z prowadzenia analizy

wymagań i planowania na początku? Oczywiście nie! Celem Scruma jest znalezienie równowagi pomiędzy przewidywaniem z góry i adaptacyjną pracą w samą porę (patrz rysunek 3.12, zaadaptowany w oparciu o źródło: [Cohn, 2009]).



**RYSUNEK 3.12.** Balansowanie pracy planowanej z pracą adaptacyjną

Podczas wytwarzania produktu środek ciężkości powinien być ustawiony w sposób uzasadniony ekonomicznie, tak aby zmaksymalizować ilość pracy adaptacyjnej w oparciu o szybką informację zwrotną i zminimalizować ilość pracy planowanej przy jednoczesnym zachowaniu wymogów wynikających z obowiązków zachowania zgodności ze specyfikacjami, regulacjami lub celami narzuconymi przez korporację.

Dokładna metoda osiągnięcia środka ciężkości zależy po części od typu wytwarzanego produktu, stopnia niepewności tego, co chcemy zbudować (niepewność końca) i jak chcemy to coś zbudować (niepewność środków), a także od ograniczeń nałożonych na proces deweloperski. Nadmierna chęć przewidywania zmusiłaby nas do poczynienia wielu założeń w obliczu dużej niepewności. Nadmierna skłonność do adaptacji doprowadziłaby do funkcjonowania w środowisku nieustannych zmian, czyniąc naszą pracę na pozór chaotyczną i nieefektywną. Chcąc tworzyć szybko innowacyjne produkty, musimy działać w środowisku, gdzie adaptacyjność jest równoważona przez zaledwie wystarczającą dawkę planowania przeciwstawiającą zapadnięciu się w chaos. Właśnie takie balansowanie na granicy porządku i chaosu zapewnia Scrum.

## Wiedza potwierdzona

Używając Scruma, organizujemy pracę w taki sposób, aby szybko uzyskiwać **wiedzę potwierdzoną** (termin zaproponowany w opracowaniu [Ries, 2011]). Nabywając wiedzę potwierdzoną, udowadniajmy lub odrzucamy pewne poczynione wcześniej założenie. Opiszę trzy zasady związane z tym tematem:

- Potwierdź bezzwłocznie istotne założenia.
- Wykorzystuj różne konkurencyjne ścieżki pozyskiwania wiedzy.
- Organizuj przepływ pracy umożliwiający szybkie pozyskiwanie informacji zwrotnej.

## Potwierdź bezzwłocznie istotne założenia

**Założenie** to ślepy strzał lub przekonanie, że założona rzecz jest prawdziwa, realna czy też pewna w sytuacji, kiedy nie mamy potwierdzonej wiedzy udowadniającej słuszność twierdzenia. Produkcja sterowana planem w porównaniu ze Scrumem o wiele lepiej toleruje założenia długoterminowe. Używając produkcji sterowanej planem, wytwarzamy dużą ilość projektów i wymagań wstępnych, w których z dużym prawdopodobieństwem występują istotne założenia — takie, które nie zostaną potwierdzone aż do znacznie późniejszej fazy deweloperskiej.

Założenia stanowią poważne **ryzyko** procesu deweloperskiego. W Scrumie staramy się minimalizować liczbę istotnych założeń obecnych w dowolnej chwili produkcji. Nie chcemy również pozwolić na to, aby istotne założenia istniały przez dłuższy czas bez potwierdzenia. Do szybkiej walidacji założeń może zostać użyte połączenie produkcji iteracyjnej i przyrostowej z jednoczesną niskonakładową eksploracją. Dzięki temu nawet jeśli przyjmiemy fundamentalnie złe założenie w Scrumie, będziemy mieli szansę przekonać się o tym w miarę szybko i naprawić błąd. W przypadku produkcji sterowanej planem to samo złe założenie odkryte późno może doprowadzić do znaczącej lub całkowitej porażki wysiłku deweloperskiego.

## Wykorzystuj różne konkurencyjne ścieżki pozyskiwania wiedzy

Podczas sekwencyjnego procesu produkcji również zachodzi etap nauki. Jednak jej najważniejsza forma ma miejsce dopiero po zbudowaniu, zintegrowaniu i przetestowaniu cech produktu, czyli dopiero na sam koniec wysiłku deweloperskiego. Późne zdobycie wiedzy daje ograniczony zysk, ponieważ do zakończenia projektu pozostaje już niewiele czasu lub dodatkowe nakłady, jakie należałyby ponieść, są zbyt wysokie.

W Scrumie rozumiemy, że nieustanna nauka jest kluczem do naszego sukcesu — identyfikujemy i wykorzystujemy pętle informacji zwrotnej do pogłębiania wiedzy. Często pojawiającym się wzorcem postępowania w tym stylu produkcji jest przyjęcie pewnego założenia (lub celów), zbudowanie czegoś (przeprowadzenie aktywności), zdobycie informacji na temat tego, co zbudowaliśmy, i wykorzystanie ich do porównania naszego wyniku z przyjętymi założeniami. Następnym krokiem jest adaptacja produktu, procesu i (lub) naszych przekonań w oparciu o to, czego się nauczyliśmy (patrz rysunek 3.13).



RYSUNEK 3.13. Wzorzec pętli zdobywania wiedzy

Scrum korzysta z kilku zdefiniowanych **pętli zdobywania wiedzy**. Na przykład codzienne działania scrumowe są pętlą realizowaną każdego dnia, a przegląd sprintu jest pętlą wykonywaną co iterację. Te i inne pętle zostaną opisane w kolejnych rozdziałach.

Scrum jest na tyle elastyczny, iż pozwala również na korzystanie z innych pętli zdobywania wiedzy. Przykładem mogą być często używane w Scrumie — chociaż niebędące częścią definicji tego środowiska — pętle zdobywania wiedzy technicznej, takie jak programowanie w parach (informacja zwrotna pojawia się natychmiast) czy programowanie sterowane testami (informacja zwrotna w przeciągu minut).

## Organizuj przepływ pracy umożliwiający szybkie pozyskiwanie informacji zwrotnej

Bycie tolerancyjnym wobec długowiecznych założeń sprawia, że proces sterowany planem staje się tolerancyjny wobec późnego pozyskiwania wiedzy — stąd **szybka pętla informacji zwrotnej** nie jest w centrum uwagi. W Scrumie staramy się uzyskać szybko informację zwrotną, ponieważ ma ona kluczowe znaczenie dla wcześniejszego przerwania ścieżek błędnego postępowania i odgrywa nieocenioną rolę w sprawnym odkrywaniu i wykorzystywaniu krótkotrwałych okazji pojawiających się na horyzoncie.

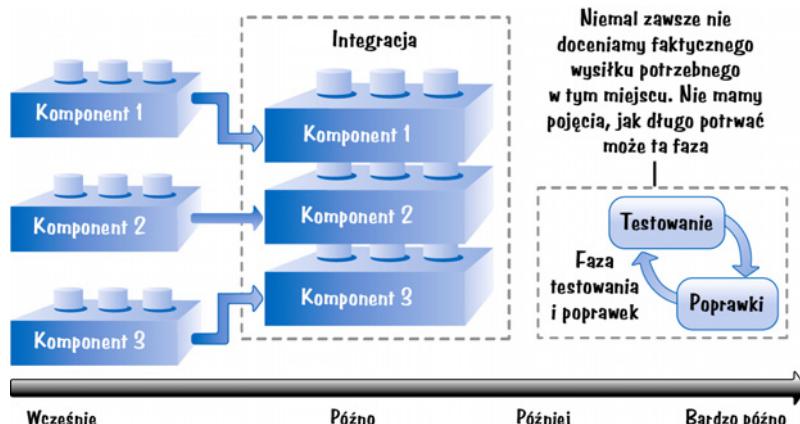
W procesie sterowanym planem wystąpienie każdej akcji jest przewidziane w ścisłe określonym czasie, zgodnie z dokładnie zdefiniowaną sekwencją faz. Takie podejście zakłada, że wcześniejsze aktywności mogą zostać zakończone bez informacji zwrotnych uzyskiwanych w fazach późniejszych. W wyniku tego może nastąpić długa przerwa pomiędzy wykonaniem jakiegoś zadania i otrzymaniem informacji zwrotnej na jego temat (czyli domknięcie pętli zdobywania wiedzy).

Jako przykład użymy **integracji** modułów i testowania. Założmy, że pracujemy równolegle nad trzema komponentami. W którymś momencie te komponenty muszą zostać zintegrowane i przetestowane, dając w wyniku produkt nadający się do sprzedaży. O tym, czy zbudowaliśmy komponenty prawidłowo, dowiemy się, dopiero integrując je. Podjęcie próby integracji da nam kluczowe odpowiedzi odnośnie do naszego wysiłku deweloperskiego.

W procesie sekwencyjnym integracja i testowanie będą odroczone w czasie do zdefiniowanego, odległego momentu, kiedy niemal wszystkie komponenty będą już połączone ze sobą. Niestety założenie, iż możemy budować wiele komponentów równolegle, a później — w fazie integracji — połączyć je płynnie ze sobą w jedną spójną całość, ma małe szanse powodzenia. W rzeczywistości nawet jeśli zaczniemy od dobrego zdefiniowania interfejsów przed zbudowaniem komponentów, istnieje spore prawdopodobieństwo, że coś pojedzie nie tak przy próbie integracji (patrz rysunek 3.14).

Aktywności generujące informację zwrotną, które mają miejsce na długo po zakończeniu procesu deweloperskiego, mają bardzo niewygodne efekty uboczne — na przykład zamieniają integrację w długotrwałą fazę testowania i poprawek, ponieważ komponenty produkowane w zupełnym odseparowaniu od siebie nie integrują się gładko. Można jedynie zgadywać, ile czasu zajmie naprawienie takiego problemu i jakie pociągnie to za sobą koszty.

W Scrumie organizujemy przepływ pracy tak, aby przejść przez pętlę zdobywania wiedzy pokazaną na rysunku 3.13 i jak najszybciej otrzymać informację zwrotną. W ten sposób zbliżamy w czasie prace deweloperskie i aktywności generujące informacje zwrotne. Szybka ocena pracy przynosi dodatkowy zysk ekonomiczny wynikający z przeciwdziałania kumulacjom błędów, które prowadzą do porażek o wykładniczo większej skali.



**RYSUNEK 3.14.** Integracja komponentów

Przyjrzymy się ponownie naszemu przykładowi integracji. Projektując komponenty, przyjęliśmy ważne założenia dotyczące sposobu ich wzajemnej integracji. W oparciu o te założenia szliśmy dalej wyznaczoną ścieżką projektu. Teraz nie możemy powiedzieć, czy była ona dobra lub zła. Jest to jedynie nasze najlepsze przypuszczenie.

Niemniej jednak kiedy zdecydowaliśmy się na daną ścieżkę, podjęliśmy szereg innych decyzji opartych na tym wyborze. Im dłużej czekamy ze sprawdzeniem oryginalnych założeń projektowych, tym więcej pojawia się zależnych od nich decyzji. Jeżeli później przekonamy się (w oparciu o informacje uzyskane w czasie fazy integracji), że oryginalne założenia były błędne, staniemy przed ogromnym bałaganem. Będziemy musieli naprawić szereg złe podjętych decyzji i to po upływie bardzo długiego czasu. Ponieważ wiedza i rozumienie tematu przez ludzi uległy zatarciu, będą oni musieli poświęcić swój czas na ponowne wdrożenie się w pracę, którą zakończyli jakiś czas temu.

Jeżeli weźmiemy jeszcze pod uwagę całkowity koszt ponownego rozpracowania potencjalnie złe podjętych decyzji na niższych poziomach, a także koszt opóźnienia produktu, korzyści ekonomiczne szybkiej informacji zwrotnej staną się bardzo przekonujące. Szybka informacja zwrotna domyka pętlę zdobywania wiedzy, pozwalając nam zaprzestać prac deweloperskich bez szans powodzenia, zanim doprowadzą one do poważnych strat ekonomicznych.

## Praca częstkowa

**Praca częstkowa** odnosi się do pracy, która została rozpoczęta, ale nie jest jeszcze zakończona. Podczas wytwarzania produktu należy umieć ją rozpoznać i odpowiednio nią zarządzać. Opisz cztery zasady zwinności związane z tym tematem:

- Stosuj rozsądne ekonomicznie rozmiary zapotrzebowania.
- Zrób rozpoznanie inwentarza i zarządzaj nim w celu dobrego przepływu.
- Skup się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę.
- Bierz pod uwagę koszt opóźnień.

## Stosuj rozsądne ekonomicznie rozmiary zapotrzebowania

Innym fundamentalnym podejściem stojącym u podstaw sekwencyjnych, sterowanych planem procesów produkcyjnych jest skupianie wszelkich zadań tego samego typu w całość wykonywaną podczas jednej fazy. Podejście to określам mianem **całości przed kolejnym krokiem** — kończymy całą (lub niemal całą) aktywność przed rozpoczęciem następnej. Na przykład w fazie analizy zapisujemy wszystkie potrzebne wymagania. Następnie przenosimy wszystkie wymogi do fazy projektowania. Ponieważ wytworzyliśmy kompletny zbiór wymagań, rozmiar naszego zapotrzebowania w tym przykładzie wynosi 100%.

Podejście w stylu **całości przed kolejnym krokiem** jest po części konsekwencją przekonania, iż ekonomia dużej skali obowiązująca w przemyśle wytwórczym ma zastosowanie również do produkcji oprogramowania. Zasady tej ekonomii mówią, że koszt wytworzenia jednostki spada wraz ze wzrostem liczby (rozmiaru zapotrzebowania) wytwarzanych jednostek. Zatem przyjmuje się, że również w sekwencyjnym procesie produkcyjnym większy rozmiar zapotrzebowania w przypadku produkcji oprogramowania podda się regułom ekonomii skali.

W Scrumie twierdzimy, że chociaż myślenie w kategoriach ekonomii skali jest podstawą w produkcji przemysłowej, zastosowanie tej myśli dogmatycznie w odniesieniu do wytwarzania oprogramowania spowoduje poważne straty ekonomiczne.

Mimo że wydaje się to nieintuicyjne, wytwarzanie mniejszych rozmiarów zapotrzebowania podczas produkcji oprogramowania ma wiele zalet. Rozmiary zapotrzebowania omawia szczegółowo Reinertsen. W tabeli 3.2 przedstawione zostały niektóre z opisanych przez niego korzyści małych rozmiarów zapotrzebowania [Reinertsen, 2009b].

**TABELA 3.2.** Zalety małych rozmiarów zapotrzebowania (źródło: [Reinertsen, 2009b])

Zaleta	Opis
Krótszy czas trwania cyklu	Mniejsze rozmiary zapotrzebowania powodują mniejsze rozmiary pracy czekającej na realizację, co z kolei oznacza krótszy czas oczekiwania na jej wykonanie. Szybciej wykonamy zaplanowane rzeczy.
Redukcja zmienności przepływu	Wyobraź sobie restaurację, w której pojawiają się klienci i po pewnym czasie odchodzą (przepływają oni płynnie przez restaurację). Teraz wyobraź sobie liczbę osób wysiadających z dużego autobusu turystycznego (duży rozmiar zapotrzebowania) i wpływ tego zdarzenia na przepływ pracy w restauracji.
Przespieszona informacja zwrotna	Małe rozmiary zapotrzebowania przespieszają informację zwrotną, zmniejszając potencjalne konsekwencje potencjalnego błędu.
Zmniejszenie ryzyka	Mniejsze rozmiary zapotrzebowania przekładają się na mniejszy inwentarz podlegający zmianom. Mniejsze rozmiary zapotrzebowania mają również mniejsze szanse niepowodzenia (istnieje większe ryzyko porażki w przypadku 10 rzeczy do zrobienia niż w przypadku 5 rzeczy).
Zmniejszone nakłady dodatkowe	Z większymi rozmiarami zapotrzebowania wiążą się większe nakłady związane z zarządzaniem — na przykład utrzymywanie listy 3000 zadań do wykonania wymaga większego wysiłku niż utrzymywanie listy 30 zadań.
Zwiększona motywacja i poczucie pilności	Mniejsze rozmiary zapotrzebowania pozwalają na lepsze skupienie i poczucie odpowiedzialności. O wiele łatwiej jest zrozumieć konsekwencje opóźnień i porażki w przypadku małych rozmiarów zapotrzebowania niż w przypadku rozmiarów dużych.
Redukcja przyrostu kosztów i harmonogramu	Kiedy mylimy się odnośnie do dużych rozmiarów zapotrzebowania, dotyczy do dużych kosztów i rozległego harmonogramu. Kiedy robimy rzeczy na małą skalę, nie popełnimy tak dużego błędu.

Jeżeli małe zapotrzebowania są lepsze od dużych, to czy nie powinniśmy używać wyłącznie rozmiarów jednostkowych, tzn. pracować wyłącznie nad jednym wymaganiem, przechodząc przez wszystkie aktywności do momentu zakończenia prac i gotowości do wdrożenia? Niektórzy określają takie podejście mianem **przepływu jednoelementowego**. W kolejnych rozdziałach pokażę, że rozmiar jednostkowy może być przydatny w niektórych sytuacjach, ale przyjmowanie „jedynki” jako naszego celu może nie być optymalne z punktu widzenia przepływu pracy i ogólnej ekonomii.

## Zrób rozpoznanie inwentarza i zarządzaj nim w celu dobrego przepływu

W tym rozdziale starałem się przypominać Ci, że produkcja wytwarzca i produkcja oprogramowania są odmiennymi zagadnieniami i tak też powinny być traktowane. Jest jednak jedna lekcja, którą przemysł wytwórczy ma już za sobą i którą my również powinniśmy odbyć w przypadku prac deweloperskich, a często nie robimy tego. Ta lekcja dotyczy wysokich kosztów inwentarza, nazywanego również pracą częstekową. Środowisko związane ze szczupłą produkcją oprogramowania już od bardzo dawna ma świadomość wagi pracy częstekowej [Poppendieck i Poppendieck, 2003; Reinertsen, 2009b] — jest to również koncepcja uznawana przez zespoły scrumowe.

Producenci mają faktyczną widzę o swoich inwentarzach i konsekwencjach związanych z ich stanem. Czy mogliby być inaczej? Inwentarz przyrasta szybko w magazynach i czeka na przetworzenie. Jest on widoczny nie tylko fizycznie, ale również poprzez finanse. Jeśli chcesz wiedzieć, ile dokładnie towaru czeka w magazynach lub ile towaru przybyło w ciągu ostatniego miesiąca, zapytaj dyrektora finansowego firmy.

Żaden kompetentny producent nie trzyma ogromnych zasobów magazynowych. Części leżące na półkach i czekające na zamontowanie w urządzeniach amortyzują się, co wpływa ujemnie na wartość księgową firmy. Co zrobić z dużymi partiami części w sytuacji, kiedy projekt urządzenia uległ zmianie? Można spróbować zmienić projekt tak, aby jednak wykorzystywał te części, lub pozbyć się ich, ponieważ nie będą dłużej wykorzystywane. Inne rozwiązanie pozwalające zapobiec stratom na częściach już zamówionych to pozostałe przy starym projekcie (chociaż wskazane byłoby go zmienić) pozwalającym na wykorzystanie części czekających w magazynie. Czy takie działanie warte jest ryzyka stworzenia mniej satysfakcjonującego produktu?

Oczywiście jest, że jeśli jesteśmy w posiadaniu sporego inwentarza i nastąpi jakaś zmiana, dotknie nas jakaś forma poważnej straty. Odpowiedzialni wytwórcy, chcąc zminimalizować straty, zarządzają inwentarzem w sposób ekonomiczny — są w posiadaniu pewnych zasobów, ale oprócz tego stosują zarządzanie inwentarzem w samą porę.

Organizacje zajmujące się produkcją oprogramowania mają o wiele mniejszą świadomość swojej pracy częstekowej. Wynika to po części z faktu, iż w przypadku produkcji oprogramowania mamy do czynienia z zasobami w postaci wiedzy, która w odróżnieniu od części na półkach magazynów fabrycznych jest niewidoczna. Pokłady wiedzy są o wiele mniej natrętne w porównaniu z kodem zajmującym przestrzeń dyskową, teczkami z dokumentami w szafkach czy planszą wiszącą na ścianie.

Dodatkowo inwentarz podczas produkcji oprogramowania jest zazwyczaj niewidoczny w formie finansowej. Jeśli zapytasz dyrektora finansowego firmy softwarowej, jakiego rozmiaru inwentarz ma pod sobą, spotkasz się ze zdziwionym wzrokiem i usłyszysz najpewniej „Nie mam żadnego”. Podczas gdy zespół finansowy śledzi parametry związane z wysiłkiem deweloperskim mającym na celu stworzenie nowego produktu, najprawdopodobniej nie bierze pod uwagę inwentarza dla tego typu przedsięwzięcia.

Tak się jednak składa, że inwentarz ma kluczowe znaczenie podczas wytwarzania produktu, a tradycyjne metody produkcji nie skupiają się na zarządzaniu nim. Jak wspomniałem wcześniej, tradycyjne metody produkcji preferują tworzenie ogromnych rozmiarów zapotrzebowania i ustawiają je na wartości bardzo duże (często rzędu 100%). Ważną konsekwencją posiadania dużego rozmiaru pracy cząstkowej podczas tworzenia produktu jest wpływ tego rozmiaru na opisaną wcześniej (rysunek 3.9) krzywą kosztu zmiany.

Zaczynając pracę nad nowym produktem, potrzebujemy pewnych wymagań, ale nie musimy mieć ich *wszystkich*. Mając zbyt wiele wymogów, najprawdopodobniej doświadczymy pewnych strat inwentarza, kiedy nadejdzie zmiana. Z drugiej strony, jeśli nasz inwentarz wymagań będzie zbyt ubogi, przerwiemy szybki przepływ pracy, co również będzie pewną formą straty. Naszym celem w Scrumie jest znalezienie odpowiedniej równowagi pomiędzy prawidłowym rozmiarem inwentarza a rozmiarem zbyt dużym.

Warto zdać sobie sprawę z faktu, iż wymagania są tylko jedną z form inwentarza w produkcji oprogramowania. Istnieje wiele innych miejsc i sytuacji podczas produkcji, gdzie w grę wchodzi praca cząstkowa. Musimy działać aktywnie, aby móc ją zidentyfikować i zarządzać nimi.

## **Skup się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę**

W Scrumie uważamy, że praca czekająca na realizację jest o wiele większym marnotrawstwem przekładającym się na szkody ekonomiczne niż pracownicy czekający na pracę. **Praca czekająca na realizację** to praca, którą chcemy wykonać (na przykład zbudować jakąś cehę lub przetestować ją), ale nie możemy, ponieważ coś powstrzymuje nas od działania. Być może musimy zaczekać, aż inny zespół wykona swoją pracę, zanim my będziemy mogli wykonać swoją, lub mamy tak dużo pracy, że nie można jej wykonać jednocześnie. W takich przypadkach pewna część pracy czeka na realizację do momentu, kiedy będziemy gotowi ją podjąć. Z drugiej strony **pracownicy czekający na pracę** to ludzie, którzy mają **zdolność** do podjęcia dodatkowej pracy, ponieważ w danej chwili nie są w pełni obciążeni.

Wiele organizacji zajmujących się produkcją oprogramowania skupia się na eliminacji marnotrawstwa wynikającego z czekania na pracę przez pracowników, a nie na eliminacji marnotrawstwa pracy czekającej na realizację. Oto przykład. Zgodnie z tradycyjnymi przekonaniami jeśli zatrudniam Cię jako testera, oczekuję, że będziesz spędzał 100% swojego czasu na testowaniu. Jeżeli nie poświęcasz swojego czasu w 100% na testowanie, w moim przekonaniu pojawią się marnotrawstwo (nie pracujesz, chociaż mógłbyś w tym czasie testować). Aby uniknąć tego problemu, znajduję dla Ciebie więcej zadań związanych z testowaniem — być może przypisując Cię do wielu różnych projektów — i w ten sposób zwiększą Twoje wykorzystanie do 100%.

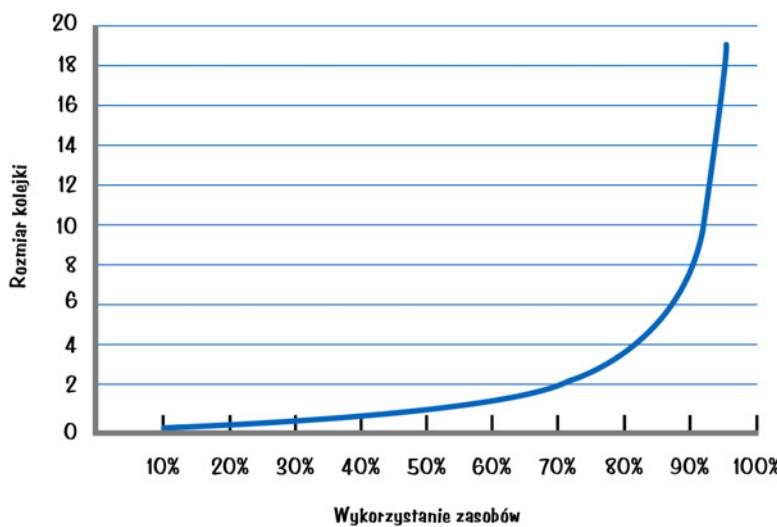
Niestety takie podejście redukuje jedno z marnotrawstw (marnotrawstwo pracowników czekających na pracę), ale jednocześnie zwiększa inną formę marnotrawstwa (marnotrawstwo pracy czekającej na realizację). W większości przypadków koszt pracy czekającej na realizację jest o wiele większy od kosztu pracowników czekających na pracę. Przekonajmy się, dlaczego tak jest.

Do zilustrowania problemu zastosujmy strategię utrzymywania pracowników w 100-procentowym obciążeniu do sztafety 4 razy 100 metrów w trakcie igrzysk olimpijskich. Według strategii 100% wykorzystania pracowników taki wyścig wydaje się absolutnie niewydajny. Płacę sportowcom za to,

żeby biegali, ale oni robią to tylko przez jedną czwartą czasu. W pozostałym czasie tylko stoją. Może kiedy nie biegną z pałeczką, mogliby pobiegać po schodach na widowni lub wziąć udział w innym wyścigu na sąsiednim torze. W ten sposób będą biegać przez 100% ich czasu.

Oczywiście wszyscy doskonale wiemy, że nie wygrywa się złotego medalu w sztafecie, każąc zawodnikom biegać przez cały czas. Wygrywa ten, kto dostarczy pałeczkę na linię mety jako pierwszy. Stąd wniosek: „Obserwuj pałeczkę, a nie biegaczy” [Larman i Vodde, 2009]. Przekładając to na kontekst produkcji oprogramowania, pałeczka leżąca na ziemi odpowiada pracy gotowej do wykonania, ale zablokowanej z powodu braku potrzebnych zasobów. Nie wygrasz wyścigu (nie dostarczysz gotowego produktu), kiedy pałeczka leży na ziemi. (Bardzo lubię porównanie z pałeczką i biegaczami, ponieważ w bardzo przejrzysty sposób pokazuje ono, że powinniśmy obserwować pracę, nie pracowników. Jednak to porównanie jak każde inne ma swoje ograniczenia. W tym przypadku sztafetowe podejście do przekazywania pracy odpowiada jednemu z aspektów tradycyjnego, sekwenncyjnego modelu produkcji, którego chcielibyśmy uniknąć!)

Inny problem to konsekwencje utrzymywania zasobów w stanie 100% wykorzystania. Jeżeli za pożyczczymy wykres z teorii kolejkowania, zauważmy oczywistą szkodę, jak powstaje w wyniku próby osiągnięcia 100% wykorzystania zasobów (patrz rysunek 3.15).



**RYSUNEK 3.15.** Wpływ wykorzystania zasobów na rozmiar kolejki

Każdy posiadacz komputera rozumie ten wykres. Co się stanie, gdy zmusisz swój komputer do pracy z pełną wydajnością (całkowite wykorzystanie procesora i pamięci)? Zacznie się zatycić i każde z wykonywanych zadań zwolni. Innymi słowy, komputer będzie robił wszystko jednocześnie, ale spadnie stopień pracy faktycznie wykonanej. Po osiągnięciu takiego stanu bardzo trudno jest z niego wyjść (najprawdopodobniej trzeba będzie zacząć „zabijać” poszczególne zadania lub zrestartować całą maszynę). Twój komputer będzie pracował znacznie wydajniej przy wykorzystaniu swoich zasobów w 80%. Na rysunku 3.15 rozmiar kolejki odpowiada opóźnieniu, a opóźnienie to pałeczka czekająca na ziemi.

Po wejściu na wysoki poziom wykorzystania zasobów praca czekająca na wykonanie (praca opóźniona) zaczyna przyrastać w sposób wykładniczy, a jej koszt może być bardzo wysoki, często wielokrotnie przekraczający koszt pracowników czekających na pracę (przykład znajdziesz w następnej sekcji poświęconej kosztom opóźnienia). Zatem w Scrumie mamy świadomość, że o wiele lepszym rozwiązaniem od próby wykorzystania pracowników w 100% jest znajdowanie wąskich gardeł w przepływie pracy i skupianie naszych wysiłków na ich eliminowaniu.

## Bierz pod uwagę koszt opóźnienia

**Koszt opóźnienia** to finansowy koszt związany z opóźnieniem prac lub opóźnieniem w osiągnięciu kroku milowego. Rysunek 3.15 pokazuje, że wraz ze wzrostem wykorzystania zasobów rosną również rozmiar kolejki i opóźnienie. Stąd redukując marnotrawstwo pracowników czekających na pracę (przez zwiększenie ich obciążenia pracą), jednocześnie podnosimy marnotrawstwo pracy czekającej na realizację (pracy znajdującej się w kolejce do podjęcia przez pracowników). Stosując koszt opóźnienia, możemy wyliczyć, która forma marnotrawstwa przynosi większe straty ekonomiczne.

Niestety 85% organizacji nie stara się oszacować kosztu opóźnienia [Reinertsen, 2009b]. Jeśli dodamy do tego fakt, iż większość organizacji produkujących oprogramowanie nie zdaje sobie sprawy z tego, że akumuluje pracę (inwentarz) czekającą w kolejkach, łatwo będzie można zrozumieć, dlaczego ich typowym zachowaniem jest skupianie się na eliminacji marnotrawstwa pracowników czekających na pracę.

Oto prosty przykład ilustrujący, dlaczego koszt pracy czekającej na realizację jest zazwyczaj znacznie większy od kosztu pracowników czekających na pracę. Zastanów się nad następującym pytaniem: czy specjalista ds. dokumentacji powinien zostać przypisany do zespołu w pierwszym dniu prac produkcyjnych, czy też pod koniec produkcji? Obie sytuacje zostały porównane w tabeli 3.3 (moglibyśmy również przyjąć inny wariant rozwiązania tego problemu).

**TABELA 3.3.** Przykład wyliczenia kosztu opóźnienia

Parametr	Wartość
Czas trwania ze specjalistą ds. dokumentacji zatrudnionym na pełny etat	12 miesięcy
Czas trwania ze specjalistą ds. dokumentacji zatrudnionym pod koniec (kiedy dotrzemy do fazy „wszystko oprócz dokumentacji”)	14 miesięcy
Koszt czasowy cyklu związany z tworzeniem dokumentacji pod koniec	2 miesiące
Koszt opóźnienia na każdy miesiąc	250 tys. zł
<b>Całkowity koszt opóźnienia</b>	500 tys. zł
Całkowity roczny koszt utrzymania specjalisty ds. dokumentacji	90 tys. zł
Miesięczny koszt utrzymania specjalisty ds. dokumentacji	7,5 tys. zł
Koszt specjalisty ds. dokumentacji w pełnym wymiarze czasu	90 tys. zł
Koszt specjalisty ds. dokumentacji przypisanego pod koniec prac	15 tys. zł
<b>Przyrostowy koszt specjalisty ds. dokumentacji zatrudnionego na pełny etat</b>	75 tys. zł

Załóżmy, że przypisaliśmy specjalistę ds. dokumentacji na pełny etat do pracy nad naszym produktem przez 12 miesięcy, chociaż w rzeczywistości nie będzie on potrzebny przez 100% czasu. Będzie nas to kosztować przyrostowo 75 tysięcy złotych (traktuj tę kwotę jako marnotrawstwo pracownika czekającego na pracę) w porównaniu z kosztami, jakie ponieślibyśmy, zatrudniając go pod koniec prac na dwa miesiące, kiedy nasz produkt dotrze do fazy „wszystko oprócz dokumentacji”.

Jeśli zatrudnimy specjalistę do wykonania całej dokumentacji pod koniec, będziemy potrzebować go na pełny etat tylko przez dwa miesiące, ale tym samym opóźnimy dostarczenie gotowego produktu o te dwa miesiące. To spowoduje powstanie kosztu opóźnienia rzędu 500 tysięcy złotych z zysków cyklu życia produktu (**zyski z cyklu życia produktu** to całkowity potencjał zysków z produktu w trakcie cyklu jego życia — w tym przypadku ten potencjał maleje o 500 tysięcy złotych).

W tym przykładzie koszt pracownika czekającego na pracę wynosi 75 tysięcy złotych, natomiast koszt pracy czekającej na realizację to 500 tysięcy złotych. Skupiając się na optymalizacji czasu pracy specjalisty ds. dokumentacji, w znacznym stopniu pogorszymy całkowity zysk z produktu. Tego typu kompromisy towarzyszą całemu procesowi produkcyjnemu. Koszt opóźnienia jest jedną z najważniejszych zmiennych podczas podejmowania decyzji istotnych z punktu widzenia ekonomii produktu.

## Postęp

Używając Scruma, mierzymy postępy według tego, co udało nam się dostarczyć i zweryfikować, a nie według tego, co aktualnie wykonujemy w oparciu o stworzony wcześniej plan, lub też tego, jak daleko udało nam się dojść w danej fazie produkcji. Opiszę trzy zasady zwinności związane z tym tematem:

- Zaadaptuj się w oparciu o napływające informacje i zmodyfikuj plan.
- Mierz postęp poprzez ocenę działających rzeczy.
- Skup się na dostarczaniu wartości.

### Zaadaptuj się w oparciu o napływające informacje i zmodyfikuj plan

W prawdziwym sekwencyjnym, sterowanym planem procesie plan jest nadzawanikiem czasu i sposobu wykonywania poszczególnych prac. W związku z tym oczekuje się działań, które będą całkowicie zgodne z planem. W Scrumie dla odmiany uważamy, że nieograniczone zaufanie do planu może często zasłonić przed nami fakt, iż on sam może zawierać błędy.

W trakcie scrumowego wysiłku deweloperskiego naszym celem nie jest działanie w zgodzie z planem, pewną narzuconą z góry oceną tego, co może się zdarzyć. Zamiast tego wolimy niezwłocznie zmienić plan i zaadaptować się w oparciu o strumień istotnych informacji ekonomicznych, jaki nieustannie napływa w trakcie wysiłku deweloperskiego.

### Mierz postęp poprzez ocenę działających rzeczy

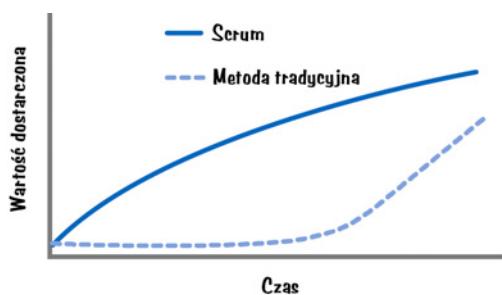
W procesie sekwencyjnym, sterowanym planem postęp jest demonstrowany poprzez zakończenie pewnej fazy i uzyskanie zgody na wejście w fazę następną. Stąd wnioskuje się, że jeśli każda faza rozpoczęła się i zakończyła zgodnie z oczekiwaniemi, proces produkcyjny jest najprawdopodobniej

w dobrej kondycji. Jednak w ostatecznym rozrachunku produkt, który stworzyliśmy zgodnie z wszelkimi wymogami planu, może stanowić o wiele mniejszą wartość dla użytkownika od oczekiwanej. Czy nadal mamy prawo ogłosić sukces, jeśli udało nam się skończyć prace na czas i bez przekraczania budżetu, jeśli nie spełniliśmy oczekiwania klientów?

W Scrumie mierzymy postępy, budując działające, przetestowane zasoby, które przedstawiają sobą wartość i których można użyć do potwierdzenia istotnych założeń projektowych. To pozwala nam uzyskać istotne informacje zwrotne i podjąć decyzję na temat kolejnego kroku. Innymi słowy, w Scrumie nie liczy się ilość pracy rozpoczętej, a jedynie to, jaką pracę mającą wartość dla klienta udało nam się zrealizować.

## Skup się na dostarczaniu wartości

Produkcja metodą sekwencyjną, sterowaną planem skupia się na ścisłym przestrzeganiu planu. Ze względu na swoją strukturę integracja i dostarczanie gotowych cech mają miejsce pod koniec wysiłku deweloperskiego (patrz rysunek 3.16). Przy takim podejściu istnieje ryzyko wyczerpania zasobów (czasu lub pieniędzy), zanim zdążyliśmy dostarczyć wszystkie cechy ważne dla klienta.



RYSUNEK 3.16. Wcześniejste dostarczanie cech o wysokiej wartości

Podobnie w tradycyjnych metodach produkcji uważa się, że wyniki planowania oraz różnorakie dokumenty powstające w trakcie prac deweloperskich same w sobie mają wartość. Jeżeli faktycznie tak jest, ich wartość jest istotna wyłącznie z punktu widzenia procesu, ale nie z punktu widzenia klienta. A jeśli nawet dokumenty te mają znaczenie dla klienta, to tylko w sytuacji, kiedy oczekiwany produkt zostanie dostarczony klientowi. Do tego czasu wspomniane artefakty nie stanowią żadnej bezpośredniej wartości dla klienta.

Scrum z kolei jest formą produkcji skupioną na generowaniu wartości dla klienta. Bazuje na sterowanym priorytetami, przyrostowym modelu produkcji, w którym cechy o największej wartości są budowane w sposób nieustający przez kolejne iteracje. W rezultacie klientom szybciej dostarczane są ważne dla nich cechy.

W Scrumie wartość powstaje przez dostarczanie klientom działających rzeczy, potwierdzanie istotnych założeń lub nabycie kluczowej wiedzy. Uważamy, że pośrednie artefakty nie mają żadnej praktycznej wartości dla klienta i są jedynie „sztuką dla sztuki”, jeśli nie można ich wykorzystać do uzyskania istotnych informacji zwrotnych lub nabycia kluczowej wiedzy.

## Wydajność

Używając Scruma, oczekujemy wydajności o określonych cechach. Opiszę trzy zasady związane z tym tematem:

- Działaj szybko, ale nie w pośpiechu.
- Buduj z zachowaniem jakości.
- Stosuj minimalną potrzebną ilość ceremonii.

### Działaj szybko, ale nie w pośpiechu

Sekwencyjny, sterowany planem proces zakłada, że jeśli będziesz przestrzegał planu i robił wszystko dobrze za pierwszym razem, unikniesz przeróbek pochłaniających koszty i czas. Oczywiście wskażane jest szybkie przechodzenie od jednego kroku do kolejnego, ale nie jest to cel nadzędny.

W Scrumie podstawowy cel to bycie zwinnym, szybkim i łatwo adaptującym się. Brnąć szybko do przodu, będąc szybciej kończyć, szybciej otrzymywać informację zwrotną i wcześniej dostarczać wartość naszym klientom. Nauka i szybkie reagowanie pozwalają nam na wcześniejsze generowanie zysków i (lub) zmniejszenie kosztów.

Nie myl jednak szybkiego poruszania się do przodu z pośpiechem. W Scrumie czas ma kluczowe znaczenie, ale naszym celem nie jest spieszenie się, aby tylko coś skończyć. Takie działanie z całą pewnością naruszyłoby zasadę **podtrzymywalnego tempa** — ludzie powinni mieć możliwość pracy w tempie, które będą w stanie utrzymać przez dłuższy okres. Dodatkowo pośpiech wpłynąłby negatywnie na jakość.

Różnice pomiędzy szybkim poruszaniem się do przodu a pośpiechem najlepiej wyjaśnić w oparciu o przykład. Studuję muay thai (tajski kick boxing). Jak w przypadku każdej sztuki walki skuteczność muay thai zwiększa się wraz z szybkością. Umiejętność zgrabnego wykonywania układów formalnych lub sparingów podnosi satysfakcję z uprawiania tego sportu i osiągane wyniki. Niemniej jednak przyspieszanie ruchów z zamiarem szybszego wykonania zdecydowanie zmniejsza ich efektywność i może doprowadzić do poważnych urazów ciała podczas sparingów. Wykonując muay thai, poruszasz się zwinnie i zdecydowanie, jednocześnie szybko adaptujesz się do sytuacji. Innymi słowy: musisz być szybki, ale nigdy w pośpiechu.

### Buduj z zachowaniem jakości

Podczas produkcji sterowanej planem przyjmuje się założenie, że dzięki dobrej wydajności pracy postępującej sekwencko do przodu osiągniemy produkt wysokiej jakości. Nie możemy jednak faktycznie ocenić tej jakości do momentu przeprowadzenia testów na zintegrowanym produkcie, który jest dostępny w bardzo późnej fazie produkcji. Jeśli testy wykażą braki w jakości, będziemy musieli wejść w kosztowną fazę poprawek i testowania z zamiarem jej podniesienia. Ponieważ nad każdą fazą produktu pracuje inny zespół, często przyjmuje się, że odpowiedzialność za wynikową jakość spoczywa na zespole testującym.

W Scrumie jakość nie jest zagadnieniem sprawdzanym przez zespół testujący na końcu produkcji, ale czymś, co spoczywa na barkach wielofunkcyjnego zespołu scrumowego i jest budowane i weryfikowane w każdym sprincie. Każdy powstający przyrost wartości jest wykańczany z zachowaniem wysokiego stopnia pewności odnośnie do jakości i może potencjalnie trafić do produkcji lub zostać wysłany do klienta (dokładniejszy opis kryteriów zakończenia prac znajdziesz w rozdziale 4.). Stąd nie ma potrzeby przeprowadzania dużych testów w późnej fazie produkcji w celu zweryfikowania jakości produktu.

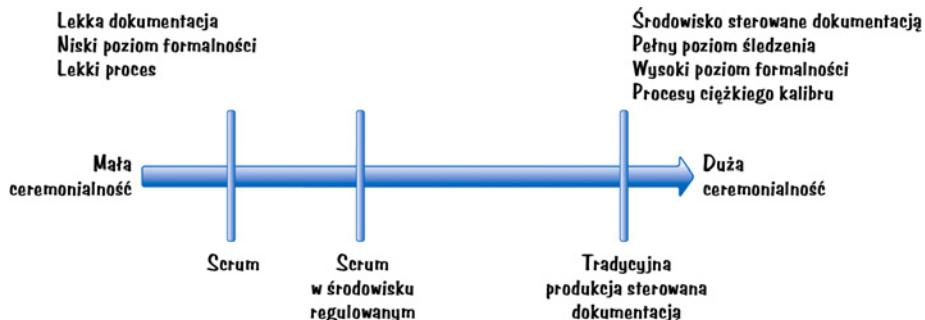
## Stosuj minimalną niezbędną ilość ceremonii

Proces sterowany planem może wiązać się z rozbudowywanymi **ceremoniami**, tworzeniem dokumentacji czy przekształcaniem wszelkich działań w procedury. W Scrumie efektem ubocznym skupiania się na wartości jest kładzenie minimalnego nacisku na ceremonie związane z procesem. Nie chcę przez to powiedzieć, że jakiekolwiek ceremonie są złe. Na przykład „ceremonia” wspólnej wyprawy do pubu w piątkowy wieczór w celu spędzenia wspólnie czasu i zbudowania więzi po pracy byłaby czymś dobrym. Mam raczej na myśli ceremonie będące **niepotrzebnymi formalnościami**. Niektórzy określają je mianem „procedur na potrzeby procedur”. Takie ceremonie niosą ze sobą koszt, ale wnoszą bardzo małą lub wręcz żadną wartość (innymi słowy, stanowią formę marnotrawstwa).

Oto przykładowe ceremonie, które mogą stanowić niepotrzebne formalności:

- Trzydniowy, ciężki proces mający na celu zatwierdzenie i przeniesienie kodu ze środowiska deweloperskiego do środowiska QA, zanim uzyskamy zgodę na rozpoczęcie testowania.
- Wszystkie problemy muszą zostać zapisane w odpowiednim systemie komputerowym, aby można było je śledzić i generować raporty, nawet jeśli wystarczyłoby zagadnąć osobę siedzącą obok i powiedzieć: „Hej, to coś nie działa, czy mógłbyś to naprawić?”, a następnie dopilnować, aby odpowiednia poprawka została wykonana, i kontynuować swoją pracę.
- Piszę dokument, ponieważ teraz zgodnie z procedurą należy stworzyć dokument, chociaż nikt nie jest w stanie powiedzieć, dlaczego ten dokument jest potrzebny lub czy ma jakakolwiek wartość.

W Scrumie naszym celem jest usuwanie niepotrzebnych formalności. Dlatego też ustawiamy bardzo nisko poprzeczkę wymaganych ceremonii — na poziomie minimalnie niezbędnym (niektórzy określają ten poziom jako ledwie satysfakcjonujący) lub wystarczająco dobrym. Oczywiście to, co składa się na poziomie minimalnie niezbędnym lub wystarczająco dobrym, zależy w dużym stopniu od organizacji. Jeśli budujemy nowy portal społecznościowy, nasze potrzeby odbywania ceremonii mogą być niezwykle małe. Z drugiej strony, jeśli naszym zadaniem jest zbudowanie rozrusznika serca, który podlega wielu regulacjom rządowym — a wraz z nimi wielu typom ceremonii — minimalny poziom formalności może być ustawiony wyżej (patrz rysunek 3.17).



**RYSUNEK 3.17.** Skala ceremonialności

Często skupienie Scruma na minimalizowaniu formalności jest błędnie interpretowane jako „negatywne nastawienie wobec dokumentowania”. To nieprawda. Scrum przyjmuje jedynie adaptacyjną, ekonomicznie uzasadnioną podstawę badania, które dokumenty należy tworzyć. Jeżeli piszemy dokument, który będzie podkładką dla kurzu na półce, marnujemy swój czas i pieniądze na rzecz martwą. Niemniej jednak nie wszystkie dokumenty są martwe. Dla przykładu: z całą pewnością napiszemy dokument, jeśli:

- Stanowi on część dostarczanego produktu (może to być instrukcja instalacji, podręcznik użytkownika itd.).
- Naszym celem jest uchwycenie istotnej dyskusji, decyzji lub porozumienia, tak aby w przyszłości móc precyzyjnie odtworzyć to, co zostało przedyskutowane, zdecydowane lub na co się zgodziliśmy.
- Jest to cenny sposób pomocy nowym członkom zespołu w szybkim nabraniu tempa pracy.
- Wymóg stworzenia odpowiedniego dokumentu jest narzucony przez prawo (koszt działalności biznesowej w środowisku regulowanym).

To, czego staramy się unikać, to praca nieprzynosząca krótkoterminowej lub długoterminowej wartości ekonomicznej. W Scrumie uważamy, że najlepiej jest spędzać czas i wydawać pieniądze, dostarczając wartość dla użytkowników.

## Zakończenie

W tym rozdziale skupiłem się na opisie kluczowych zasad zwinności — fundamentalnych przekonań, które wyznaczają sposób produkcji w Scrumie. Porównałem, jak przekonania te różnią się od przekonań będących podstawą podręcznikową, sterowanej planem, sekwencyjnej produkcji (patrz podsumowanie w tabeli 3.4).

Celem tego porównania nie jest przekonanie Cię do tego, że metoda kaskadowa jest zła, a Scrum dobry. Chciałem raczej pokazać, że zasady procesów kaskadowych predysponują te metody do rozwiązywania innych klas problemów od tych, dla których stworzono Scrum. Sam możesz ocenić, jakiego typu zagadnieniami zajmuje się Twoja organizacja, i w związku z tym jakiego procesu powinna używać. Kolejne rozdziały książki będą szczegółowo opisywać wzajemne oddziaływanie przedstawionych zasad wytwarzające niezwykle wydajne środowisko produkcji.

**TABELA 3.4.** Podsumowujące porównanie procesów sterowanych planem z zasadami zwinności

Temat	Zasada procesów sterowanych planem	Zasada metod zwinności
<b>Podobieństwo między tworzeniem oprogramowania a produkcją masową</b>	W obu przypadkach podąża się według zdefiniowanego planu.	Produkcja oprogramowania to nie produkcja masowa, celem produkcji oprogramowania jest stworzenie przepisu na produkt.
<b>Struktura procesu</b>	Produkcja oparta jest na fazach wykonywanych sekwencyjnie.	Produkcja powinna mieć charakter iteracyjny i przyrostowy.
<b>Stopień zmienności procesu i produktu</b>	Usiłowanie wyeliminowania zmienności procesu i produktu.	Wsparcie zmienności poprzez inspekcję, adaptację i przejrzość.
<b>Zarządzanie niepewnością</b>	Wyeliminuj niepewność końca, a następnie niepewność środków.	Redukuj jednocześnie wszystkie rodzaje niepewności.
<b>Podejmowanie decyzji</b>	Podejmij decyzję w odpowiedniej dla niej fazie.	Pozostaw otwarte opcje.
<b>Zrób wszystko dobrze za pierwszym razem</b>	Przyjmuje się założenie z góry, iż posiadamy wszystkie właściwe informacje do stworzenia wymogów i planów.	Nie jesteśmy w stanie zrobić wszystkiego dobrze za pierwszym razem.
<b>Eksploracja kontra eksplotacja</b>	Eksplotuj to, co jest obecnie znane, i przewiduj to, co nieznane.	Preferuj podejście adaptacyjne i odkrywcze.
<b>Zmiana/wypadek</b>	Zmiana wpływa destrukcyjnie na plany, dlatego należy jej unikać.	Preferuj zmiany w sposób uzasadniony ekonomicznie.
<b>Przewidywanie kontra adaptacja</b>	Ten proces stara się przewidywać.	Równoważ przewidywanie z góry z pracą adaptacyjną w samą porę.
<b>Założenia (wiedza niepotwierdzona)</b>	Proces toleruje założenia trwające przez długi czas.	Potwierdzaj jak najszybciej ważne założenia.
<b>Informacja zwrotna</b>	Kluczowa wiedza pojawia się podczas jednej, głównej pętli analizy, projektowania, kodowania i testowania.	Korzystaj z wielu symultanicznych ścieżek zdobywania wiedzy.
<b>Szybka informacja zwrotna</b>	Proces toleruje późne otrzymywanie informacji zwrotnej.	Organizuj pracę tak, aby szybko uzyskiwać informację zwrotną.
<b>Rozmiar zapotrzebowania (jaką ilość pracy należy wykonać przed rozpoczęciem następnej aktywności)</b>	Rozmiary zapotrzebowania są duże, często wynoszą 100% — całość przed kolejnym krokiem. Zakłada się ekonomię skali.	Używaj małych, rozsądnych pod względem ekonomicznym rozmiarów zapotrzebowania.
<b>Inwentarz/praca cząstkowa</b>	Inwentarz nie należy do systemu wartości w tym procesie, więc nie skupia się na nim uwagi.	Rozpoznaj swój inwentarz i zarządzaj nim dla osiągnięcia dobrego przepływu pracy.
<b>Marnotrawstwo pracy kontra marnotrawstwo ludzi</b>	Przypisuj zasoby ludzkie tak, aby każdy pracował na wysokim poziomie wydajności.	Skup się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę.
<b>Koszt opóźnienia</b>	Koszt opóźnienia jest rzadko brany pod uwagę.	Zawsze bierz pod uwagę koszt opóźnienia.

**TABELA 3.4.** Podsumowujące porównanie procesów sterowanych planem z zasadami zwinności — ciąg dalszy

Temat	Zasada procesów sterowanych planem	Zasada metod zwinności
<b>Przestrzeganie planu</b>	Przestrzeganie planu jest uważane za podstawowy środek osiągnięcia dobrego wyniku.	Adaptuj się i modyfikuj plan, zamiast go przestrzegać.
<b>Postęp</b>	Pokazuj postępy, przechodząc do kolejnych faz produkcji.	Mierz postępy, weryfikując działające zasoby.
<b>Cele będące w centrum uwagi</b>	W centrum uwagi jest proces — przestrzegaj procesu.	Wartość jest w centrum uwagi — dostarczaj wartość.
<b>Szybkość</b>	Podążaj według procesu, staraj się wykonywać rzeczy dobrze za pierwszym razem i działań szybko.	Działaj szybko, ale nigdy nie w pośpiechu.
<b>Kiedy otrzymujemy wysoką jakość</b>	Wysoka jakość pojawia się na końcu po intensywnej fazie testowania i poprawek.	Buduj jakość od samego początku.
<b>Formalność (ceremonialność)</b>	Formalności (dobrze zdefiniowane procedury i punkty kontrolne) są ważne dla efektywnej pracy.	Stosuj najmniejszą możliwą ilość ceremonii.

## Rozdział 4

# SPRINTY

---

Scrum organizuje pracę w iteracje lub cykle — o długości nie przekraczającej miesiąca — nazywane sprintami. Ten rozdział opisuje dokładniej, czym są sprinty. Poruszę tutaj kilka kluczowych charakterystyk sprintów: są ograniczone czasowo — mają jednakowy, krótki okres trwania, mają swój cel, który nie powinien być zmieniany po rozpoczęciu, i muszą osiągnąć stan końca zgodnie ze zdefiniowanymi przez zespół kryteriami końca pracy.

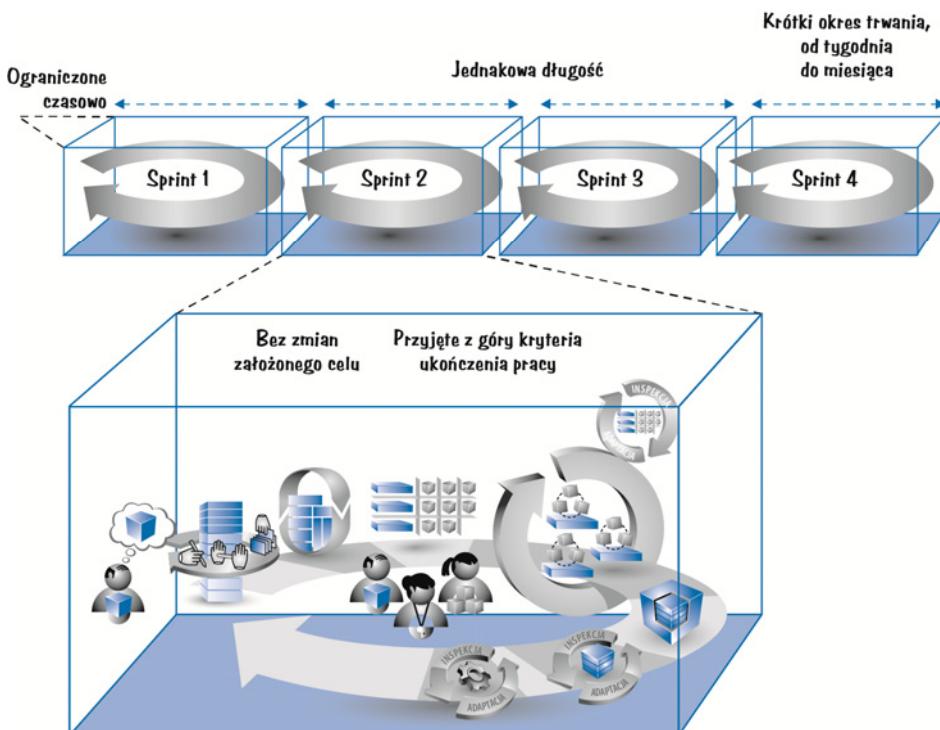
### Wprowadzenie

Sprinty stanowią szkielet środowiska Scrum (patrz rysunek 4.1).

Na rysunku 4.1 sprint jest reprezentowany przez dużą szarą strzałkę rozciągającą się od rejestru produktu poprzez pętlę wykonania sprintu i otaczającą członków zespołu scrumowego. Pozostałe artefakty i aktywności scrumowe pokazane są w punktach odpowiadających chwili ich wystąpienia w czasie trwania sprintu. Wykonanie sprintu jest często mycone ze sprintem jako takim, ale w rzeczywistości jest to tylko jedna z aktywności mających miejsce w trakcie trwania sprintu, razem z planowaniem sprintu, przeglądem sprintu i retrospekcją sprintu.

Wszystkie sprinty są ograniczone czasowo — mają ustalony czas rozpoczęcia i zakończenia. Ponadto sprinty muszą być krótkie — mogą trwać od tygodnia do miesiąca. Ich długość powinna być jednakowa, chociaż pod pewnymi warunkami dopuszcza się wyjątki. W trakcie trwania sprintu nie wolno zmieniać jego celu pod względem zakresu prac lub składu zespołu. Wynikiem sprintu jest potencjalnie zdatny do wdrożenia przyrost produktu spełniający kryteria ukończenia prac ustalone wcześniej przez zespół scrumowy.

Mimo że każda organizacja implementuje Scrum na swój sposób, powyższe kryteria, z kilkoma wyjątkami opisanymi dalej, powinny obowiązywać w każdym sprintie i zespole scrumowym. Przyjrzyjmy się im z bliska, aby zrozumieć, dlaczego mają właśnie taką, a nie inną postać.



**RYSUNEK 4.1.** Sprinty — szkielet środowiska Scrum

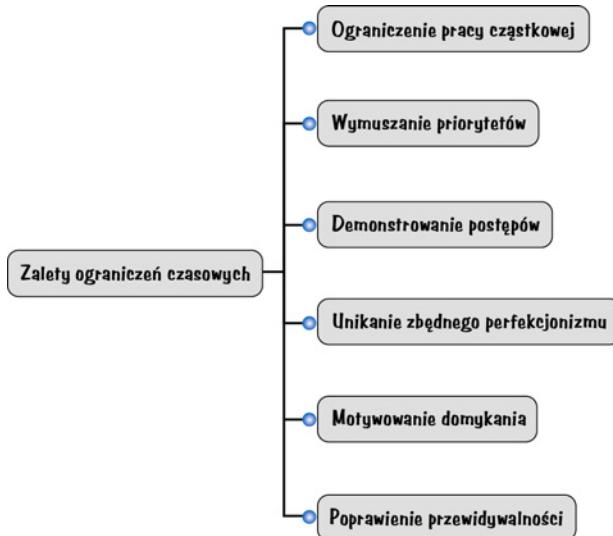
## Ograniczenie czasowe

Sprinty bazują na koncepcji **ograniczenia czasowego** — techniki zarządzania czasem, która pozwala panować nad wydajnością i zakresem pracy. Każdy sprint odbywa się w przedziale czasu mającym ustaloną datę rozpoczęcia i zakończenia, nazywanym pudełkiem czasu (ang. *timebox*). W tym czasie oczekuje się od zespołu pracy w podtrzymywalnym tempie, pozwalającej na ukończenie zadań będących w zgodzie z celem sprintu.

Ograniczenie czasowe jest istotne z kilku powodów (patrz rysunek 4.2).

## Ograniczenie pracy częstekowej

Ograniczenie czasowe jest techniką ograniczenia ilości pracy częstekowej. Praca częstekowa reprezentuje inwentarz pracy rozpoczętej, ale jeszcze nieukończonej. Nieumiejęte zarządzanie tą pracą może mieć poważne konsekwencje finansowe. Sprint (ograniczenie czasowe) zapewnia nałożenie limitu na pracę częstekową, ponieważ zespół zaplanuje wykonanie tylko tych elementów, które uważa za wykonalne w trakcie jego trwania.



RYSUNEK 4.2. Zalety ograniczeń czasowych

## Wymuszanie priorytetów

Ograniczenie czasowe zmusza nas do ustalenia priorytetów i wykonania małej ilości pracy, która ma największe znaczenie. To skupia naszą uwagę na wyprodukowaniu czegoś wartościowego w szybkim tempie.

## Demonstrowanie postępów

Ograniczenie czasowe pomaga nam zademonstrować postępy w pracach poprzez kończenie i weryfikowanie istotnych elementów wytwarzanego produktu do ustalonego dnia (zakończenia sprintu). Tego typu działanie redukuje ryzyko nieefektywnej organizacji pracy poprzez unikanie nierealnych form raportowania postępów w pracy, takich jak przestrzeganie planu. Ograniczenie czasowe pomaga również wykazywać postęp w pracach nad dużymi cechami, których ukończenie wymaga więcej niż jednego zakresu czasu. Każdy sprint daje poczucie pewności, iż osiągnięto mierzalny, wartościowy postęp w pracach, a także pozwala interesariuszom dowiedzieć się, co jeszcze trzeba zrobić, aby dostarczyć gotową cechę produktu.

## Unikanie zbędnego perfekcjonizmu

Ograniczenie czasowe zapobiega zbędнемu perfekcjonizmowi. Każdemu z nas zdarzyło się nieraz spędzić zbyt wiele czasu, próbując zrobić coś w sposób „perfekcyjny” lub doprowadzić pewną cechę do stanu idealnego, w sytuacji, gdy stan dobry był w pełni satysfakcjonujący. Wprowadzając ustaloną datę zakończenia sprintu, czyli moment, kiedy musi zostać dostarczone dobre rozwiązanie, ograniczenie czasowe wymusza zakończenie prac, które teoretycznie można byłoby prowadzić w nieskończoność.

## Motywowanie domykania

Ograniczenie czasowe zachęca do domykania prac. Ze swojego doświadczenia mogę powiedzieć, że praca jest faktycznie wykonywana, kiedy zespoły znają datę zakończenia zadania. Świadomość, że koniec sprintu niesie ze sobą nieprzekraczalny termin rozliczenia, zachęca członków zespołu do włożenia wysiłku w wykonanie pracy na czas. Bez takiej daty znika poczucie pilności zakończenia pracy.

## Poprawianie przewidywalności

Ograniczenie czasowe poprawia przewidywalność. Chociaż nie jesteśmy w stanie przewidzieć precyjnie, jaką ilość pracy uda nam się wykonać w ciągu najbliższego roku, bez wątpienia powinniśmy w miarę dokładnie przewidzieć zadania, które będziemy w stanie zrealizować w ciągu nadchodzącego krótkiego sprintu.

## Krótki okres trwania

Krótkie okresy trwania sprintów przynoszą wiele korzyści (patrz rysunek 4.3).



RYSUNEK 4.3. Zalety krótkich sprintów

## Łatwość planowania

Krótkie sprinty ułatwiają planowanie. O wiele prościej jest zaplanować pracę na okres kilku tygodni niż na przykład na sześć miesięcy. Poza tym planowanie w tak krótkim horyzoncie czasowym wymaga o wiele mniej wysiłku i jest o wiele bardziej zgodne z rzeczywistością w porównaniu z planowaniem długoterminowym.

## Szybka informacja zwrotna

Sprinty o krótkim czasie trwania pozwalają na szybkie uzyskanie informacji zwrotnej. Podczas każdego sprintu tworzymy działające oprogramowanie — możemy dokonać jego inspekcji, a następnie adaptacji tego, co budujemy i jak to budujemy. W ten sposób możemy szybciej wyeliminować ścieżki niekorzystne dla produktu lub samego procesu produkcji, zanim podejmiemy błędą decyzję, która otworzy drogę do kolejnych szkodliwych działań. Szybka informacja zwrotna pozwala nam również na wcześniejsze odkrywanie i wykorzystywanie pojawiających się krótkoterminowych okazji.

## Lepsze zyski z inwestycji

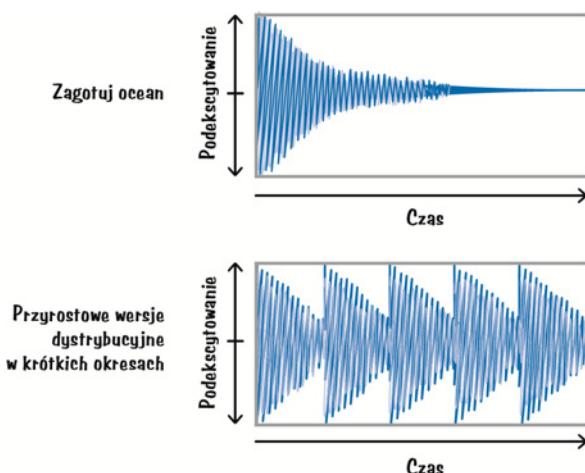
Sprinty o krótkim czasie trwania nie tylko poprawiają ekonomiczność poprzez szybką informację zwrotną, ale również pozwalają na wcześniejsze i bardziej regularne wypuszczanie kolejnych wersji oprogramowania. Dzięki temu mamy okazję zacząć wcześniej generować zysk, poprawiając całkowity zwrot z inwestycji (przykład znajdziesz w rozdziale 14.).

## Ograniczanie błędów

Sprinty o krótkim czasie trwania ograniczają błędy. Jak bardzo możemy się pomylić w ciągu dwóch tygodni? Nawet jeśli całkowicie „położyliśmy” sprint, straciliśmy jedynie dwa tygodnie. Nacisk na sprinty o krótkim czasie trwania jest tak ważny, ponieważ pozwala na częstą koordynację i uzyskiwanie informacji zwrotnych. W ten sposób nawet jeśli jesteśmy w błędzie, jest to na szczęście mały błąd.

## Rozbudzenie podekscytowania

Krótkie sprints mogą pomóc w rozbudzeniu podekscytowania. Cechą natury ludzkiej jest spadek zainteresowania nagrodą i wynikająca z tego eksycytacja wraz z upływem czasu (patrz rysunek 4.4).



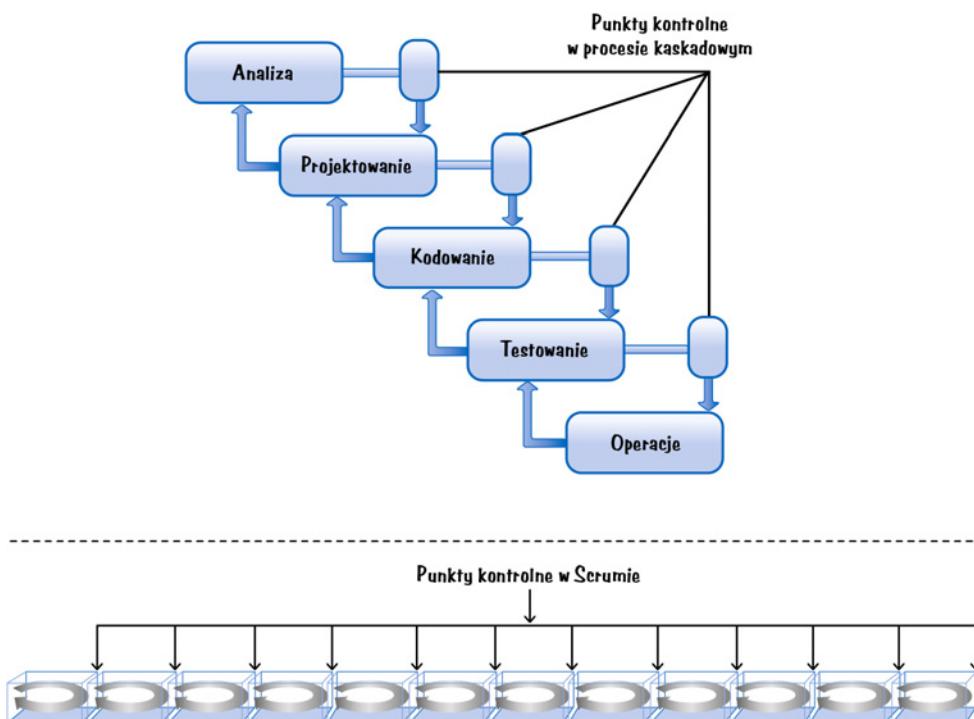
RYSUNEK 4.4. Podekscytowanie w funkcji czasu

Jeżeli pracujemy nad projektem o bardzo długim czasie trwania, nie tylko jest większe prawdopodobieństwo porażki, ale również zwiększone ryzyko utraty entuzjazmu do ponoszenia dalszych wysiłków. (Kiedy pracowałem w IBM, takie zjawiska określaliśmy mianem projektów w stylu „zagotuj ocean”, ponieważ trwały one naprawdę bardzo długo, a ich ukończenie — o ile było możliwe — wymagało wielkiego wysiłku, porównywalnego z próbą zagotowania oceanu). Nie widząc postępów lub też perspektywy końca na horyzoncie, ludzie zaczynają tracić zainteresowanie. Pod koniec są gotowi zapłacić komuś za możliwość przejścia do innego produktu!

Sprinty o krótkim czasie trwania podtrzymują ekscytację uczestników dzięki nieustannemu dostarczaniu działających rzeczy. Zadowolenie płynące z pojawiających się wcześniej i regularnie działających przyrostów produktu rozbudza nasze zainteresowanie i pragnienie kontynuacji pracy nad celem końcowym.

## Regularne punkty kontrolne

Krótkie sprinty dostarczają wielu użytecznych punktów kontrolnych (patrz rysunek 4.5).



**RYSUNEK 4.5.** Porównanie punktów kontrolnych

Jednym z cennych aspektów projektów sekwencyjnych są dobrze zdefiniowane kamienie milowe. Dają one menedżerom narzędzie w postaci zdefiniowanych punktów kontrolnych w trakcie cyklu życia projektu, które zazwyczaj służą do podjęcia decyzji o kontynuacji lub zaprzestaniu finansowania kolejnej fazy projektu. Chociaż potencjalnie użyteczne z punktu widzenia nadzoru, kamienie milowe nie odzwierciedlają prawdziwego stanu wartości dla klienta (o czym pisałem w rozdziale 3.).

Scrum daje menedżerom, interesariuszom, właścicielom produktu i innym uczestnikom o wiele więcej punktów kontrolnych w porównaniu z ilością oferowaną przez projekty sekwencyjne. Istotny punkt kontrolny (przegląd sprintu) pojawia się pod koniec każdego sprintu. Pozwala on każdemu na podjęcie decyzji w oparciu o demonstrowane, działające cechy produktu. Ludzie radzą sobie lepiej w złożonym środowisku, kiedy podczas punktów kontrolnych mają możliwość bardziej interakcyjnej inspekcji i adaptacji.

## Stał czas trwania

Zgodnie z zasadami zespół powinien wybrać stały rozmiar sprintów dla konkretnego projektu i nie zmieniać go, o ile nie istnieje ważny powód ku temu. Ważne powody to między innymi:

- Rozważanie przez Ciebie przejścia ze sprintów 4-tygodniowych na 2-tygodniowe w celu częstszego otrzymywania informacji zwrotnej. Przed podjęciem ostatecznej decyzji chcesz wykonać kilka sprintów 2-tygodniowych na próbę.
- Coroczny okres urlopowy lub okres kończący rok finansowy powodują, że bardziej praktyczne jest wykonywanie 3-tygodniowych sprintów zamiast zwyczajowych 2-tygodniowych.
- Produkt ma zostać wdrożony w ciągu tygodnia, zatem wykonywanie dwutygodniowego sprintu byłoby marnotrawstwem.

Fakt, iż zespół nie jest w stanie wykonać całej pracy w ciągu aktualnie obowiązującej długości sprintu, nie jest przekonującym powodem do zwiększenia długości sprintów. Podobnie niedopuszczalne jest lobbowanie na rzecz przedłużenia sprintu po dotarciu do ostatniego dnia i przekonaniu się, że zabrakło czasu na wykonanie całej pracy. Są to symptomy złego funkcjonowania oraz okazje na poprawienie procesu, ale nie powody usprawiedlwiącące wydłużanie sprintów.

Zatem zgodnie z zasadami: jeśli zespół zgodzi się na wykonywanie sprintów dwutygodniowych, wszystkie sprinty powinny trwać dokładnie tyle czasu. Z praktycznych względów większość zespołów (choć nie wszystkie) definiuje dwa tygodnie jako dziesięć dni roboczych. Jeśli w trakcie sprintu pojawia się dzień wolny od pracy lub jednodniowe szkolenie, liczba dni roboczych sprintu zostaje zmniejszona, ale bez wpływu na fizyczny czas jego trwania.

Stosowanie sprintów o tej samej długości przynosi dodatkowe zalety wynikające z taktowania oraz upraszcza planowanie.

## Zalety taktowania

Sprinty o takiej samej długości zapewniają **takt** — regularny, przewidywalny rytm lub też puls scrumowego wysiłku deweloperskiego. Stabilny, zdrowy takt pozwala zespołowi i organizacji na wzajemne przyswojenie reguł występowania poszczególnych zdarzeń zapewniających szybki, a jednocześnie możliwy do kształtuowania przepływ wartości biznesowej. Z własnego doświadczenia mogę powiedzieć, że regularny takt sprintów pozwala ludziom „wejść w temat”, „być przy piłce”, „wpaść w rutynę”. Uważam, że dzieje się tak, ponieważ regularny takt przekształca mało ciekawe, chociaż niezbędne działania w nawyk niewymagający poświęcania aktywnej uwagi i ten sposób pozwala skupić sił umysłu na pracy przynoszącej radość i wartość dodaną.

Krótki takt sprintowy ma tendencję do niwelowania intensywności wysiłku. W przeciwieństwie do tradycyjnego, sekwencyjnego projektu, podczas którego obserwujemy szybki wzrost intensywności prac w fazach późniejszych, profile intensywności prac w kolejnych sprintach są podobne do siebie. Takt sprintów pozwala zespołom na pracę w podtrzymywalnym tempie — będę o tym pisał w rozdziale 11.

Wykonywanie sprintów z regularnym taktem znacząco obniża narzut pracy koordynacyjnej. Mając sprinty o ustalonej długości, możemy przewidzieć planowanie, przegląd i retrospekcję dla wielu sprintów jednocześnie. Dzięki temu, że wszyscy wiedzą, kiedy będą miały miejsce te aktywności, w znacznym stopniu redukujemy obowiązki niezbędne do zaplanowania dużej ilości sprintów.

Przykładowo: jeżeli w ciągu roku wysiłku deweloperskiego wykonujemy sprinty dwutygodniowe, możemy wysłać powtarzające się przez 26 iteracji zaproszenie na przegląd sprintu do wszystkich uczestników. Wyobraź sobie, ile dodatkowej pracy musielibyśmy wykonać, synchronizując spotkania dla wszystkich interesariuszy (zakładając oczywiście, że znaleźlibyśmy czas odpowiadający wszystkim istotnym interesariuszom, mającym swoje kalendarze wypełnione na wiele tygodni w przód) w sytuacji, kiedy pozwolilibyśmy na sprinty o różnej długości.

I w końcu, jeżeli nad projektem pracuje wiele zespołów o zbliżonym taktie sprintowym, możliwe jest synchronizowanie pracy między nimi (bardziej szczegółowe omówienie tego tematu znajdziesz w rozdziale 12.).

## Uproszczone planowanie

Ujednolicony czas trwania sprintów upraszcza aktywności związane z planowaniem. Kiedy wszystkie sprinty mają taką samą długość (uwzględniając wahania wynikające z dni wolnych od pracy), zespół zyskuje komfort pod względem ilości pracy (**prędkości**), jaką może wykonać podczas typowego sprintu. Prędkość jest zazwyczaj wyznaczana w odniesieniu do sprintów. W przypadku sprintów o zmiennej długości nie możemy powiedzieć, że posiadamy znormalizowaną jednostkę pomiaru. Zdanie „Zespół ma przeciętną prędkość 20 punktów na sprint” byłoby pozbawione sensu.

Wyznaczenie prędkości zespołu pracującego ze zmienną długością sprintów jest możliwe, ale wymaga bardziej skomplikowanych obliczeń. Trzymanie się jednakowej długości sprintów upraszcza wyliczanie prędkości w oparciu o dane historyczne zespołu.

Uproszczeniu ulegają również inne działania matematyczne związane z planowaniem. Na przykład: jeżeli pracujemy nad **dystrybucją z ustaloną datą** i nasze sprinty mają jednakową długość, wyznaczenie liczby sprintów w ramach tej wersji sprawdza się do prostych obliczeń kalendarzowych (znamy datę dzisiejszą, datę dystrybucji i wiemy, że wszystkie sprinty mają taką samą długość). Gdyby sprinty mogły mieć różną długość, wyznaczenie liczby sprintów nie byłoby już takie proste (musielibyśmy przeprowadzić wczesne wyczerpujące planowanie) — wprowadziłoby niepotrzebny narzut pracy, a wyniki nie dawałyby takiej samej pewności jak te wynikające ze stałej długości sprintów.

## Niezmiennosć celu

Istotna zasada Scruma mówi, że po ustaleniu celu sprintu i rozpoczęciu jego wykonania nie wolno wprowadzać zmian, które mogą fizycznie wpływać na jego cel.

## Czym jest cel sprintu?

Każdy sprint można podsumować poprzez jego cel, który opisuje motywizujące biznesowe i wartość sprintu. Zazwyczaj cel sprintu jest wyrażony w jasny sposób i skupia się na pojedynczej rzeczy, na przykład jest to:

- Pomoc w stworzeniu raportu wstępnego.
- Załadowanie i sprawdzenie danych mapy Ameryki Północnej.
- Zademonstrowanie zdolności do wysłania wiadomości tekstowej poprzez zintegrowany stos oprogramowania, sterownika i sprzętu.

Czasami cel sprintu może być wyrażony w formie więcej niż jednego faktu, na przykład „Uruszczyć drukowanie na podstawowym poziomie i dodać obsługę wyszukiwania po dacie”.

Podczas planowania sprintu zespół deweloperski powinien pomagać w ustaleniu i doprecyzowaniu celu, a następnie kierując się tym celem, wskazać elementy z rejestru produktu, które mogą zostać zrealizowane do końca sprintu (więcej na ten temat w rozdziale 19.). Wybrane elementy dziennika produktu posłużą do dalszego uszczegółowienia celu sprintu.

## Wzajemne zobowiązanie

Cel sprintu jest podstawą do wzajemnego zobowiązania pomiędzy zespołem i właścicielem produktu. Zespół zobowiązuje się zrealizować cel przed końcem sprintu, natomiast właściciel produktu zobowiązuje się nie zmieniać celu sprintu w trakcie jego trwania.

To wzajemne zobowiązanie pokazuje wagę sprintów w równoważeniu wymogu adaptacji biznesu do zachodzących zmian, z potrzebą skoncentrowania się zespołu na zadaniu i kreatywnego wykorzystania swojego talentu do stworzenia wartości podczas ustalonej, krótkiej iteracji. Definiując cel sprintu i przestrzegając go, zespół scrumowy jest w stanie skupić się na dobrze zdefiniowanym, wartościowym zadaniu.

## Zmiana kontra doprecyzowanie

Chociaż nie wolno fizycznie zmieniać celu sprintu, dopuszcza się jego *doprecyzowanie*. Pozwolić sobie rozróżnić te dwa pojęcia.

Czym cechuje się zmiana? Zmiana to każda modyfikacja zakresu pracy lub zasobów mogąca doprowadzić do ekonomicznie istotnego marnotrawstwa, szkodliwej przerwy w przepływie pracy lub znaczącego zwiększenia ilości prac w sprincie. Typowe zmiany to dodanie lub usunięcie ze sprintu elementu rejestru produktu lub istotna zmiana zakresu elementu z rejestru produktu, który jest już w sprincie. Oto przykład ilustrujący zmianę:

*Właściciel produktu: „Kiedy mówiłem, że musimy mieć możliwość przeszukiwania policyjnej bazy danych pod kątem młodocianych przestępco, nie miałem na myśli tylko wyszukiwania po imieniu i nazwisku. Chciałem również mieć możliwość wyszukiwania w oparciu o zdjęcia tatuaży podejrzanych!”.*

Dodanie możliwości przeszukiwania w oparciu o zdjęcie wymaga najprawdopodobniej zdecydowanie więcej wysiłku i niemal na pewno wpłynie na zdolność zespołu do dotrzymania zobowiązania dostarczenia funkcjonalności wyszukiwania po imieniu i nazwisku. W takiej sytuacji właściciel produktu powinien rozważyć stworzenie nowego elementu (historyjki) obejmującego wyszukiwanie w oparciu o zdjęcia i dodanie go do rejestru produktu z zamiarem skierowania do realizacji w kolejnym sprincie.

Czym cechuje się doprecyzowanie? Doprecyzowanie to dodatkowe szczegóły dostarczone zespołowi w trakcie sprintu, pomagające mu w osiągnięciu celu sprintu. Na początku sprintu elementy rejestru produktu mogą nie być dostatecznie uszczegółowione (będzie o tym mowa w rozdziale 5.). Stąd zupełnie zrozumiałe jest, że zespół będzie zadawał pytania wyjaśniające w trakcie sprintu, a właściciel produktu będzie na nie odpowiadał. Doprecyzowanie ilustruje poniższy przykład:

*Zespół deweloperski: „Kiedy mówiłeś, że znalezione rekordy z bazy danych młodocianych przestępów powinny zostać wyświetcone w formie listy, miałeś na myśli jakiś konkretny porządek tej listy?”.*

*Właściciel produktu: „Tak, posortujcie ją alfabetycznie po nazwisku”.*

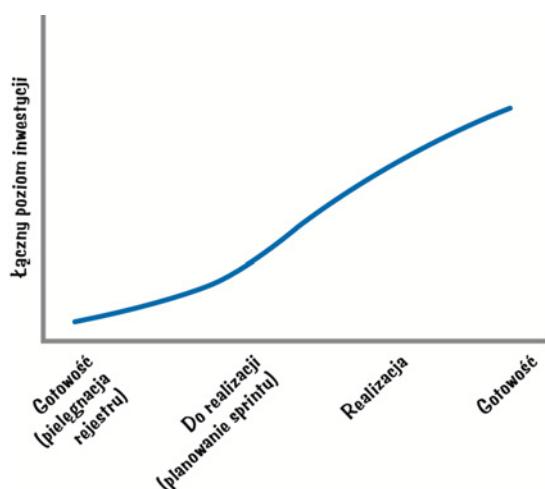
*Zespół deweloperski: „Dobrze, tak zrobimy”.*

W ten sposób właściciel produktu może i powinien dostarczać wyjaśnień w trakcie sprintu.

## Konsekwencje zmiany

Może wydawać się, że zasada zabraniająca zmiany celu sprintu stoi w sprzeczności z fundamentalną zasadą Scruma mówiącą, że powinniśmy być otwarci na zmiany. Jesteśmy otwarci na zmiany, ale chcemy to robić w sposób zrównoważony i mający sens ekonomiczny.

Konsekwencje ekonomiczne zmiany rosną razem z poziomem naszego zaangażowania w pracę wynikającą ze zmian (patrz rysunek 4.6).



RYSUNEK 4.6. Łączny poziom inwestycji w kolejnych fazach

Inwestujemy w elementy rejestru produktu, aby przygotować je do realizacji w sprincie. Kiedy jednak sprint już wystartuje, nasz poziom inwestycji w te elementy wzrasta (ponieważ poświęciliśmy nasz czas w trakcie planowania sprintu na dyskusje i rozpisanie zadań). Jeśli chcemy wprowadzić zmianę po planowaniu sprintu, nie tylko narażamy naszą inwestycję w planowanie, ale również wprowadzamy dodatkowe koszty wynikające z konieczności ponownego rozplanowania zmian w trakcie sprintu.

Dodatkowo kiedy już rozpoczęliśmy wykonanie sprintu, nasza inwestycja w pracę zwiększa się jeszcze bardziej w miarę przechodzenia elementów rejestru produktu przez kolejne fazy od przygotowania (praca czekająca na wykonanie), przez realizację (wykonanie pracy) do gotowości (zakończenie pracy).

Załóżmy, że chcemy wymienić cechę X będącą częścią aktualnego zobowiązania sprintowego na cechę Y, która nie została wzięta do sprintu. Nawet jeśli nie zaczeliśmy jeszcze pracować nad cechą X, ponosimy stratę związaną z planowaniem. Dodatkowo cecha X może być powiązana z innymi cechami w sprincie, zatem dotycząca jej zmiana może wypływać również na inne elementy sprintu i odbić się na całym celu sprintu.

Jeżeli prace nad cechą X zostały już rozpoczęte, to oprócz wspomnianych już strat możemy wprowadzić inne formy marnotrawstwa. Na przykład trzeba będzie porzucić wyniki zrealizowanych już częściowo prac nad cechą X bez możliwości skorzystania z nich w przyszłości (nie zgadzamy się na umieszczenie w potencjalnie gotowym do wdrożenia przyroście produktu pracy ukończonej tylko w części).

Oczywiście, jeśli cecha X jest już gotowa<sup>1</sup>, możemy potencjalnie stracić niemal pełną wartość poniesionych na nią inwestycji. Całe to marnotrawstwo kumuluje się!

Bezpośredni wpływ marnotrawstwa na ekonomię przedsięwzięcia to jedna rzecz, ale mogą również wystąpić straty pośrednie wynikające z potencjalnego pogorszenia motywacji i zaufania zespołu w obliczu zmiany. Kiedy właściciel produktu zobowiązuje się nie zmieniać celu, a następnie łamie to zobowiązanie, zespół w naturalny sposób traci motywację, co z całą pewnością wpłynie na chęć pilnej pracy i ukończenia pozostałych elementów rejestru produktu znajdujących się w sprincie. Złamanie zobowiązania może narazić na szwank zaufanie w zespole scrumowym — zespół deweloperski przestanie ufać w wolę właściciela produktu do przestrzegania przyjętych przez niego zobowiązań.

## Pragmatyczność

Zakaz wprowadzania zmian w sprincie jest tylko zasadą, nie przepisem prawa. Zespół scrumowy musi być pragmatyczny.

Co zrobić, jeśli warunki biznesowe ulegną takim modyfikacjom, że zmiana celu sprintowego będzie nieunikniona? Założmy, że w trakcie trwania sprintu nasz konkurent wprowadził na rynek swój nowy produkt. Po jego analizie dochodzimy do wniosku, że musimy zmienić cel ustalony dla naszego bieżącego sprintu, ponieważ to, co teraz robimy, nagle straciło sens ekonomiczny w obliczu osiągnięcia konkurencji. Czy powinniśmy ślepo przestrzegać zasady zabraniającej zmiany celu sprintu i jego zawartości? Prawdopodobnie nie.

<sup>1</sup> Zapewne chodzi tutaj o sytuację, w której dana cecha została wykonana, ale nie przeszła jeszcze przez końcowe fazy produkcji (testy automatyczne, weryfikacja manualna) i nie może zostać zintegrowana z resztą produktu — *przyp. tłum.*

Jeśli jeden z systemów produkcyjnych w firmie ulegnie poważnej awarii, a jedynymi osobami zdolnymi do jego naprawy są ludzie w naszym zespole, to czy powinniśmy przerwać bieżący sprint, aby dokonać naprawy? Czy też odpowiemy, że będziemy w stanie naprawić ten system na samym początku następnego sprintu? Prawdopodobnie nie.

W ostatecznym rozrachunku bycie pragmatycznym przebija zasadę zakazu zmiany celu. Musiemy działać w sposób uzasadniony ekonomicznie. Każdy członek zespołu scrumowego powinien to rozumieć. Jeśli zmienimy zawartość bieżącego sprintu, odnotujemy negatywne skutki ekonomiczne, które opisałem wcześniej. Jeśli jednak konsekwencje ekonomiczne zmiany będą o wiele mniejsze od skutków jej odroczenia, przeprowadzenie zmiany można uznać za mądrą decyzję biznesową. Nie należy modyfikować sprintu, jeśli wprowadzenie zmiany i powstrzymanie się od niej nie niosą ze sobą konsekwencji finansowych.

Co do motywacji i zaufania zespołu — z własnego doświadczenia mogę powiedzieć, że kiedy właściciel produktu prowadzi z zespołem szczerą, skupioną na ekonomii dyskusję dotyczącą konieczności zmiany, większość potrafi zrozumieć konieczność podjęcia takich działań, a motywacja i zaufanie zostają podtrzymane.

## Zakończenie przed czasem

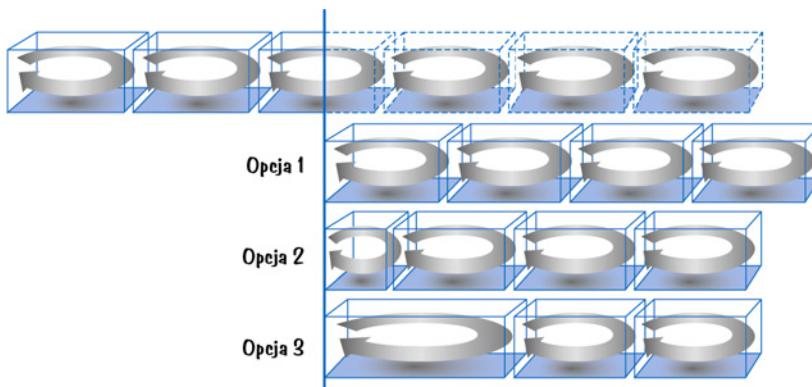
Jeśli cel sprintu stanie się zupełnie nieaktualny, zespół scrumowy powinien uznać kontynuację bieżącego sprintu za pozbawioną sensu i zaproponować właścicielowi produktu jego przedwczesne zakończenie. Zakończenie sprintu przed czasem kończy wszelką pracę, a zespół scrumowy organizuje spotkanie w celu przeprowadzenia retrospekcji. Następnie odbywa się planowanie kolejnego sprintu z nowym celem i innym zestawem elementów rejestru produktu.

Sprint przerwany jest w obliczu ważnego wydarzenia o znaczeniu ekonomicznym. Może to być działanie konkurencji, które fizycznie zmienia sposób finansowania sprintu lub całego produktu.

Chociaż właściciel produktu ma prawo przerwać każdy sprint, z własnego doświadczenia wiem, że opcja ta jest bardzo rzadko wykorzystywana. Często zespół scrumowy podejmuje działania o wiele mniejszego kalibru, aby dostosować się do nowych warunków. Pamiętaj, że sprinty są krótkie i statystycznie rzecz biorąc, zespół będzie gdzieś w środku sprintu, kiedy wystąpi zdarzenie wymuszające zmianę. W takiej sytuacji do zakończenia sprintu pozostałe mniej więcej tydzień czasu i z ekonomicznego punktu widzenia przerwanie sprintu może być mniej opłacalne od pozostania na kursie do zakończenia. W wielu przypadkach można podjąć mniej dramatyczne decyzje, na przykład aby znaleźć czas na naprawienie krytycznego problemu produkcyjnego, wystarczy porzucić pracę nad jedną z cech, zamiast przerывать cały sprint.

Należy zdać sobie sprawę, iż przerwanie sprintu przed czasem ma nie tylko negatywny wpływ na morale, ale jest również poważnym zakłóceniem w szybkim i elastycznym procesie powstawania nowych cech, negującym wiele z zalet sprintów o jednakowej długości, o których mówiłem wcześniej. Przerwanie sprintu powinno być środkiem ostatecznym.

Jeżeli sprint zostanie przerwany, zespół scrumowy będzie musiał określić długość kolejnego sprintu (patrz rysunek 4.7).



**RYSUNEK 4.7.** Wybór długości następnego sprintu po przerwaniu sprintu

Istnieją trzy sensowne możliwości:

1. Pozostać przy oryginalnej długości sprintu. To rozwiązanie pozwala na zachowanie sprintów o jednakowej długości w całym procesie deweloperskim (oczywiście z wyjątkiem sprintu, który został przerwany). Jeżeli w produkcji bierze udział więcej zespołów, zachowanie oryginalnej długości sprintów spowoduje rozsynchonizowanie jedynie zespołu scrumowego, który przerwał swój sprint.
2. Skrócić kolejny sprint tak, aby jego data zakończenia pokryła się z datą zakończenia przerwanego sprintu. Na przykład: jeśli zespół scrumowy przerwał dwutygodniowy sprint pod koniec pierwszego tygodnia, kolejny sprint będzie miał tylko tydzień i w ten sposób zsynchronizuje zespół z jego oryginalnym taktem sprintowym.
3. Przedłużyć kolejny sprint ponad rozmiar normalnego sprintu, tak aby objął swoim zasięgiem czas sprintu przerwanego i czas pełnego sprintu. W naszym poprzednim przykładzie ponowne zsynchronizowanie z oryginalnym taktem sprintowym nastąpi po utworzeniu sprintu trzytygodniowego.

Jeżeli w procesie produkcyjnym bierze udział więcej niż jeden zespół, preferowane jest rozwiązanie nr 2 lub 3. W celu trafnego wyboru jednej z powyższych opcji będziesz musiał rozważyć swój konkretny kontekst.

## Definicja ukończenia

W rozdziale 2. pisałem o tym, że wynik każdego sprintu powinien być potencjalnie nadającym się do wdrożenia przyrostem produktu. Wspomniałem, że „potencjalnie nadający się do wdrożenia” nie oznacza wcale, że to, co zostało zbudowane, zostanie faktycznie wdrożone. Wdrożenie jest decyzją biznesową, często podejmowaną z zupełnie odmiennym taktem. W niektórych organizacjach wdrażanie co sprint może mieć rację bytu.

Potencjalną zdolność do wdrożenia lepiej jest traktować jako stan pewności, że to, co zbudowaliśmy w sprintie, zostało faktycznie ukończone, czyli nie pozostała do wykonania żadna istotna praca (taka jak testowanie, integracja itp.), którą należałoby wykonać przed wdrożeniem wyników sprintu, gdybyśmy chcieli podjąć decyzję biznesową o wdrożeniu. Ustalenie, czy to, co zostało wyprodukowane, nadaje się do potencjalnego wdrożenia, wymaga posiadania przez zespół scrumowy dobrze zdefiniowanej i uzgodnionej definicji ukończenia.

## Czym jest definicja ukończenia?

Koncepcyjnie **definicja ukończenia** to lista kontrolna różnych typów pracy, jakie zespół powinien zakończyć z powodzeniem, zanim będzie mógł zadeklarować, że wyniki pracy nadają się potencjalnie do wdrożenia (patrz tabela 4.1).

**TABELA 4.1.** Przykładowa definicja listy kontrolnej definicji ukończenia

Definicja ukończenia	
<input type="checkbox"/>	Weryfikacja projektu
<input type="checkbox"/>	Kodowanie
<input type="checkbox"/>	Kod został zrefaktoryzowany
<input type="checkbox"/>	Kod posiada format zgodny ze standardem
<input type="checkbox"/>	Umieszczono komentarze w kodzie
<input type="checkbox"/>	Kod został umieszczony w repozytorium
<input type="checkbox"/>	Kod został przejrzany
<input type="checkbox"/>	Aktualizacja dokumentacji
<input type="checkbox"/>	Testy
<input type="checkbox"/>	Wykonano testy jednostkowe
<input type="checkbox"/>	Wykonano testy integracyjne
<input type="checkbox"/>	Wykonano testy regresyjne
<input type="checkbox"/>	Wykonano testy na wszystkich obsługiwanych platformach
<input type="checkbox"/>	Przetestowano pod względem różnych wersji językowych
<input type="checkbox"/>	Zerowy stan znanych problemów
<input type="checkbox"/>	Testy akceptacyjne
<input type="checkbox"/>	Rozwiązanie umieszczone na serwerach produkcyjnych

Faktyczna zawartość listy będzie zależeć od wielu zmiennych:

- Natury budowanego produktu.
- Technologii używanych do budowy produktu.
- Organizacji, która buduje produkt.
- Aktualnych przeszkód ograniczających możliwe rozwiązania.

W większości przypadków absolutne minimum definicji ukończenia powinno wyrażać się przez dostarczenie kompletnego fragmentu funkcjonalności — takiego, który został zaprojektowany, zbudowany, zintegrowany, przetestowany, udokumentowany i zweryfikowany pod względem dostarczania faktycznej wartości dla klienta. Skonstruowanie użytecznej listy kontrolnej wymaga podzielenia tych głównych zadań na bardziej szczegółowe. Co oznacza na przykład przetestowanie? Czy chodzi o puszczenie testów jednostkowych, sprawdzenie całego systemu, weryfikację na różnych platformach, czy też testowanie wersji językowych programu? Prawdopodobnie do tej listy mógłbyś dołożyć jeszcze inne testy, specyficzne dla Twojego produktu. Czy wszystkie wymienione typy testowania zostały włączone do definicji ukończenia?

Pamiętaj, że jeśli nie przeprowadzasz istotnych testów w każdym sprintie (na przykład testów wydajnościowych), będziesz musiał wykonać je w innym czasie. Czy utworzysz na końcu sprint poświęcony wyłączenie testowaniu wydajności? Jeśli zrobisz w ten sposób w sytuacji, kiedy testowanie wydajności jest istotnym elementem „ukończenia”, to tak naprawdę nie będziesz posiadał potencjalnie nadających się do wdrożenia przyrostów produktu co sprint. Co gorsza, jeżeli pod koniec okaże się jeszcze, że wyniki tych testów nie są zadowalające, staniesz w obliczu poważnego problemu w końcowej fazie produkcji i będziesz musiał poświęcić sporo czasu i pieniędzy na jego naprawienie — więcej w porównaniu z kosztem wykonywania testów wydajnościowych wcześniej.

Czasami testowanie może wymagać więcej czasu, niż zajmuje sam sprint. Jeżeli wynika to z ogromnej liczby testów manualnych, jakie trzeba przeprowadzać, zespół musi zacząć je automatyzować, tak aby można było dokończyć testowanie w ramach sprintu. Jeżeli czas trwania testów wynika z ich natury, trzeba zaakceptować fakt, iż testowanie rozpocznie się w jednym sprintie, a zakończy w kolejnych. Mogę podać przykład trenowanej przeze mnie organizacji, która budowała urządzenie składające się ze sprzętu, sterowników i oprogramowania. Jednym z ich standardowych testów był trwający 1500 godzin test wytrzymałościowy — w tym czasie urządzenie pracowało non stop w celu sprawdzenia, czy nie zawiedzie. Takiego testu nie można przeprowadzić w trakcie dwutygodniowego sprintu, zatem zespół scrumowy musiał zmodyfikować swoje kryteria ukończenia tak, aby móc ogłosić zakończenie prac w sprintie mimo trwającego jeszcze testu 1500 godzin.

Często jestem pytany: „Co zrobić, jeśli ostatniego dnia sprintu ujawniony zostanie poważny błąd? Czy w takiej sytuacji można uznać element rejestru produktu za ukończony?” Nie, nie można! A ponieważ zgodnie z zasadą nie przedłużamy sprintów ponad zaplanowane ograniczenie czasowe, nie przedłużymy go również o dzień lub dwa w tym przypadku, żeby móc naprawić błąd w trwającym jeszcze sprintie. Zamiast tego w zaplanowanym dniu zakończenia sprintu niekompletny element rejestru produktu zostaje wyjęty z bieżącego sprintu i wstawiony do rejestru produktu w miejscu odpowiednim ze względu na jego priorytet wobec pozostałych elementów rejestru. To pozwoli na jego potencjalne dokończenie w jednym z przyszłych sprintów.

Zespoły scrumowe muszą posiadać solidną definicję ukończenia prac, taką, która pozwala na osiągnięcie wysokiego stopnia pewności, że to, co zostało zbudowane, ma odpowiednią jakość i może zostać wdrożone. Każe odstępstwo od tej zasady pozbawia organizację biznesowych możliwości wdrażania produktów w stosownym do tego czasie i powoduje narastanie długiego technicznego (o czym będzie mowa w rozdziale 8.).

## Definicja ukończenia może ewoluować

Definicję ukończenia możesz traktować jako definicję stanu prac pod koniec sprintu. Dla wielu bardzo wydajnych zespołów celem końcowym pracy jest umożliwienie jej potencjalnego wdrożenia i cel ten pozostaje względnie stały w trakcie całego procesu produkcyjnego.

Na przykład: kiedy byłem właścicielem produktu podczas przeprojektowywania witryny internetowej Scrum Alliance w 2007 roku, wykonywaliśmy sprints jednetygodniowe. Końcowy stan naszej definicji ukończenia można było podsumować stwierdzeniem „uruchomienie na serwerach produkcyjnych”. Zespół i ja uznaliśmy, że był to najbardziej rozsądny stan, jaki chcieliśmy osiągnąć w każdym sprincie. Zdefiniowaliśmy ten stan na początku prac deweloperskich i pozostał on niezmieniony przez cały czas mojego szefowania jako właściciela produktu.

Wiele zespołów zaczyna jednak od definicji ukończenia, która nie pozwala na stwierdzenie, iż wszystkie cechy zostały wykonane w stopniu pozwalającym na ich publikację lub dystrybucję. Dla niektórych pewne rzeczywiste przeszkody mogą uniemożliwić osiągnięcie takiego stanu na początku procesu produkcyjnego, mimo że jest to cel, który chcą ostatecznie osiągnąć. Te zespoły (z konieczności) będą zaczynać od uboższego stanu końcowego sprintu i stopniowo modyfikować swoje kryteria ukończenia w miarę usuwania przeszkód przez organizację.

Miałem okazję odwiedzić organizację zajmującą się budowaniem systemów informatycznych dla klinik medycznych. Ich produkty instalowane są w szpitalach i magazynują różnorodne dane medyczne (w niektórych przypadkach bezpośrednio z urządzeń wykonujących testy diagnostyczne). Zespół deweloperski pracujący nad danym testem klinicznym posiadał umiejętność zainstalowania oprogramowania w laboratorium i przekonania się, że działa ono dobrze ze sprzętem — rzecz, która powinna być przetestowana, zanim produkt zostanie wdrożony. Ponieważ jednak deweloperzy nie mieli stałego dostępu do laboratorium, na początku zespół nie włączył testów klinicznych do swoich kryteriów ukończenia pracy. Zamiast tego wyznaczył sprints testów klinicznych pod koniec prac nad każdą wersją dystrybucyjną.

Podczas dyskusji dowiedziałem się, że zarówno ludzie z marketingu, jak i członkowie zespołu scrumowego nie cierpieli tych testów klinicznych przed wypuszczeniem wersji. Nikt nie był w stanie przewidzieć, ile sprintów zajmie usunięcie wszystkich błędów, a produkt nie mógł zostać wypuszczony, zanim nie zostały usunięte wszystkie błędy. Kiedy zastanawialiśmy się nad możliwymi rozwiązaniami tego problemu, w rozmowy włączył się dyrektor techniczny, zadając pytanie: „Czy gdybyście mieli dostęp do laboratorium klinicznego, bylibyście w stanie wykonywać testy na miejscu w każdym sprincie?”.

Członkowie zespołu przedyskutowali jego pytanie i odpowiedzieli: „Tak, ale to oznacza, że będziemy realizować mniej cech w każdym sprincie”. Dyrektor zgodził się usunąć przeszkodę, udostępniając zespołowi laboratorium w lokalnej klinice uniwersyteckiej. Właściciel produktu stwierdził, że wyprodukowanie mniejszej liczby cech w każdym sprincie jest rozsądnym kompromisem w zamian za wiedzę, że cechy, które zostały zbudowane, przeszły również testy kliniczne. W tym momencie zespół mógł zmodyfikować swoją definicję ukończenia, osiągając faktyczny stan „przyrostu nadającego się potencjalnie do dystrybucji”, co dało wszystkim zainteresowanym osobom większe poczucie pewności odnośnie do pracy realizowanej w każdym sprincie.

Zespół może napotykać przeszkody, których nie da się usunąć „z marszu”. Wiadomo wtedy, że definicja ukończenia będzie musiała ewoluować w trakcie procesu produkcyjnego. Dobrym przykładem takiej sytuacji jest wytwarzanie produktu, na który składa się sprzęt i oprogramowanie.

Widziałem wiele przypadków zastosowania Scruma do takiej produkcji i często słyszałem od ludzi pracujących nad oprogramowaniem: „Sprzęt jest zawsze spóźniony!”. W sytuacjach takich jak ta zespół budujący oprogramowanie bez sprzętu, na którym to oprogramowanie ma działać, nie może twierdzić, że wyniki każdego sprintu nadają się potencjalnie do dystrybucji. Może co najwyżej stwierdzić, że spełnił kryteria gotowości „dla emulatora”, ponieważ w pierwszych sprintach testy wykonuje się w oparciu o emulator prawdziwego urządzenia. Później, kiedy dostępne będzie rzeczywiste urządzenie, definicja ukończenia zostanie zmodyfikowana i zacznie odzwierciedlać stan potencjalnej gotowości do wdrożenia lub przynajmniej coś bardziej zbliżonego do tego stanu.

## Definicja ukończenia kontra kryteria akceptacji

Definicja ukończenia dotyczy przyrostu produktu wytwarzanego w czasie sprintu. Na przyrost produktu składa się kilka z elementów rejestru produktu, zatem każdy taki element musi zostać wykonany zgodnie z zadaniami przewidzianymi na liście kontrolnej definicji ukończenia.

W rozdziale 5. piszę o tym, że każdy element z rejestru produktu umieszczony w sprintie powinien posiadać swoje **warunki zadowolenia** (kryteria akceptacji specyficzne dla tego elementu) zdefiniowane przez właściciela produktu. **Kryteria akceptacji** będą ostatecznie weryfikowane przez **testy akceptacyjne**, które właściciel produktu wykona, aby sprawdzić, czy gotowa funkcjonalność działa zgodnie z oczekiwaniemi. Na przykład: jeśli element rejestru produktu brzmi „Pozwolić klientowi na zakup przy użyciu karty kredytowej”, warunkiem zadowolenia może być „Działa z użyciem kart AmEx, Visa i MasterCard”. Zatem każdy element z rejestru produktu będzie posiadał swoje indywidualne kryteria akceptacji. Te kryteria specyficzne dla elementu są dodatkiem, nie zastępstwem, do kryteriów ukończenia wyznaczonych przez listę kontrolną definicji ukończenia, które obowiązują wszystkie elementy rejestru produktu.

Element rejestru produktu można uznać za ukończony, jedynie jeśli spełnione zostały zarówno kryteria akceptacji specyficzne dla tego elementu (na przykład „działa ze wszystkimi kartami kredytowymi”), jak i wszystkie punkty definicji ukończenia na poziomie sprintu (na przykład „uruchomiono na serwerze produkcyjnym”).

Jeżeli określanie elementów rejestru produktu, które spełniły swoje kryteria akceptacji, mianem *ukończonych* jest mylące, nazywaj je elementami *skompletowanymi* lub *zaakceptowanymi*.

## Ukończony kontra rzeczywiście ukończony

Niektóre zespoły przyjęły koncepcję „ukończenia” i „rzeczywistego ukończenia”. Okazuje się, że „rzeczywiście ukończone” jest bardziej ukończone od jedynie „ukończonego”. Zespoły nie powinny potrzebować tych dwóch różnych koncepcji, ale muszę się przyznać, że sam stosuję oba pojęcia w odniesieniu do mojego syna i jego pracy domowej. Miałem zwyczaj pytać go, czy już skończył (ukończył) swoją pracę domową, a on odpowiadał, że tak. Potem podczas spotkania z jego wychowawczynią zapytałem: „Czy kiedy mój syn zwraca pracę, jest ona zrobiona?”. W odpowiedzi usyszałem „Nie za bardzo!”.

Po bardziej wnikliwej dyskusji z moim synem zrozumiałem, że jego rozumienie ukończenia oznaczało „Zrobiłem tyle pracy, ile byłem w stanie wykonać!”. Od tego momentu zacząłem używać określenia *rzeczywistego ukończenia*, które zgodnie z naszym obopólnym porozumieniem oznaczało „ukończone w takim stopniu, aby nauczyciel nabrął przekonania, że wykonałeś swoją pracę”.

Zespoły, które nie są przyzwyczajone do szybkiego dokończania prac, częściej stosują kryterium rzeczywistego ukończenia jako podporę. Dla nich istnieje wyraźne rozróżnienie pomiędzy ukończeniem (wykonaniem takiej ilości pracy, jaką były w stanie zrealizować) a rzeczywistym ukończeniem (wykonaniem takiej ilości pracy, aby klienci nabrali przekonania, że zadanie zostało faktycznie zrealizowane). Zespoły, które rozumieją, że mogą uznać pracę za ukończone tylko wtedy, kiedy wykonana została cała praca zaspokajająca potrzeby klienta, nie potrzebują dwóch stanów. Dla nich praca ukończona oznacza pracę rzeczywiści ukończoną.

## Zakończenie

W tym rozdziale podkreśliłem kluczową rolę sprintów w środowisku scrumowym. Sprinty stanowią szkielet Scruma, z którym związana jest większość aktywności i artefaktów. Sprinty są krótkie, ograniczone czasowo i trwają jednakowo długo. Zazwyczaj zdefiniowane są poprzez cel sprintu, który może być modyfikowany tylko w przypadku ważnych przesłanek ekonomicznych. Wynikiem sprintów powinny być potencjalnie nadające się do wdrożenia przyrosty produktu wykonane według uzgodnionej wcześniej definicji ukończenia. W następnym rozdziale skupię się na danych wejściowych sprintu — wymaganiach — i ich najczęstszej formie — historyjkach użytkownika.

## Rozdział 5

# WYMAGANIA I HISTORYJKI UŻYTKOWNIKA

---

W tym rozdziale omówię różnice w sposobie traktowania wymagań przez projekt scrumowy w porównaniu z projektem tradycyjnym. Przekazanie tej wiedzy pozwoli mi na opisanie w dalszej części roli historyjek użytkownika jako platformy do przedstawiania elementów o wartości biznesowej. Skupię się na omówieniu, czym są historyjki użytkownika, w jaki sposób mogą one reprezentować wartość biznesową na różnych poziomach abstrakcji, a także po czym poznać dobre historyjki użytkownika. Dalej opiszę, jak w projekcie scrumowym poradzić sobie z wymaganiami niefunkcjonalnymi i pracą mającą na celu pozyskanie wiedzy. Na zakończenie przedstawię dokładniej dwie techniki tworzenia historyjek użytkownika.

## Wprowadzenie

Scrum i produkcja metodą sekwencyjną traktują wymagania w zupełnie odmienny sposób. W przypadku sekwencyjnego procesu produkcyjnego wymagania nie podlegają negocjacjom, od samego początku mają charakter szczegółowy i tworzone są jako odrębny byt. W Scrumie szczegóły wymagań są negocjowane poprzez dyskusje odbywające się nieustannie w trakcie procesu produkcyjnego. Powstają *w samą porę i w stopniu wystarczającym* dla zespołów do rozpoczęcia wytwarzania funkcjonalności realizującej żądany wymóg.

W sekwencyjnym procesie produkcyjnym wymagania traktowane są podobnie jak w przypadku produkcji masowej. Są niepodlegającymi dyskusji, obowiązkowymi specyfikacjami, z którymi produkt musi zachować zgodność. Wymagania tworzone są na samym początku i dostarczane zespołom deweloperskim w formie bardzo szczegółowego dokumentu. Od tego momentu obowiązkiem zespołów deweloperskich jest stworzenie produktu, który będzie spełniał te szczegółowe wymagania.

Kiedy zajdzie potrzeba zmiany oryginalnego planu, jest ona realizowana poprzez formalny proces. Celem jest zachowanie za wszelką cenę zgodności ze specyfikacjami, w związku z tym wszelkie odejście od nich są niepożądane. Przyczyną takiego stanu jest potencjalna konieczność modyfikacji lub porzucenia istotnej części pracy częstekowej — w formie szczegółowo udokumentowanych wymagań (lub zadań zrealizowanych na ich podstawie).

Scrum dla odmiany postrzega wymagania jako istotny stopień swobody, którym możemy manipulować w celu osiągnięcia naszych celów biznesowych. Na przykład kiedy zaczyna brakować pieniędzy lub czasu, możemy porzucić wymagania o niskim priorytecie. Jeśli podczas produkcji pojawiła się nowa informacja zmieniająca współczynnik kosztów do zysków dla danego wymogu na zupełnie dla nas niekorzystny, nic nie stoi na przeszkodzie, abyśmy wykluczyli ten wymóg z produktu. Natomiast gdy na horyzoncie pojawi się wymóg o większej wartości, mamy możliwość dodania go do produktu, być może w miejsce wymogu o niższej wartości.

Zapewne wszyscy mieliśmy kiedyś okazję przejść przez tworzenie dokumentu zawierającego „kompletne” wymagania na początku prac deweloperskich tylko po to, aby później przekonać się, że jakiś ważny wymóg został w nim pominięty. Kiedy odkryliśmy ten brakujący wymóg, rozmowa na jego temat przebiegała mniej więcej tak:

*Klient: „Teraz, kiedy zobaczyłem listę cech, zdałem sobie sprawę, że potrzebuję jeszcze jednej cechy, której nie ma w dokumencie z wymaganiami”.*

*Deweloperzy: „Skoro ta cecha jest potrzebna, dlaczego nie wyspecyfikował jej pan wcześniej?”.*

*Klient: „Nie zdawałem sobie sprawy, że potrzebuję tej cechy, dopóki cały produkt nie został złożony w całość”.*

*Deweloperzy: „Gdyby poświęcił pan więcej czasu i energii na przeanalizowanie wymagań wcześniej, znalazłby pan tę cechę wtedy, a nie teraz”.*

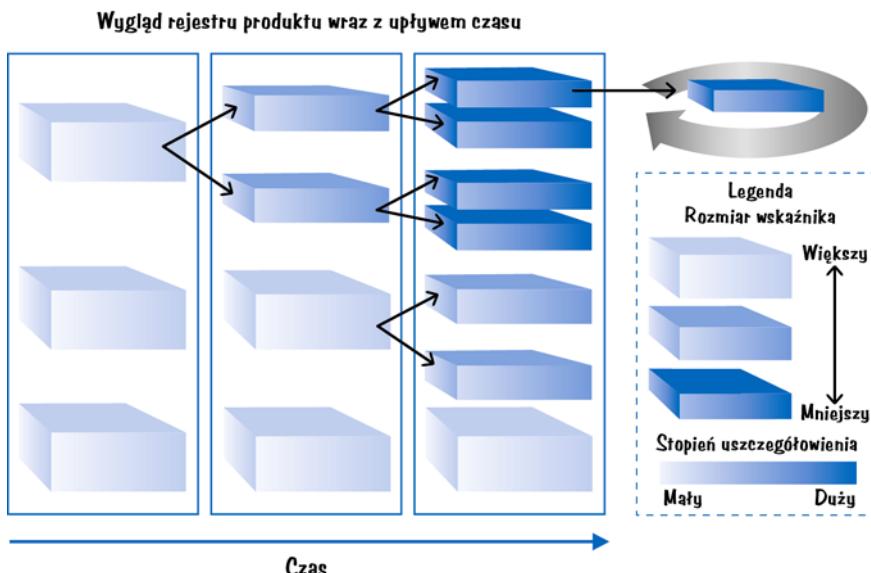
Nie ulega wątpliwości, że podczas tworzenia innowacyjnych produktów nie da się z góry stworzyć wyczerpującej listy wymogów lub projektów przez wydłużenie czasu prac lub wzmożenie wysiłków. Pewne wymogi lub projekty zostaną ujawnione w toku prac deweloperskich — żadna ilość gruntownych przemyśleń na początku projektu nie zdoła zapobiec takiej sytuacji.

W związku z tym w Scrumie nie inwestujemy dużych ilości czasu i pieniędzy wtworzenie szczegółowych wymagań na samym początku projektu. Spodziewając się zmian w miarę upływu czasu i wzrostu naszej wiedzy na temat budowanego rozwiązania, unikamy nadmiernego inwestowania w wymagania, z których być może zrezygnujemy w przyszłości. Zamiast obszernego inwentarza szczegółowych wymagań tworzymy na początku jedynie wskaźniki wymagań nazywane **historyjkami rejestru produktu**<sup>1</sup>. Każda historyjka rejestru produktu reprezentuje sobą oczekiwana wartość biznesową (patrz rysunek 5.1).

Początkowo historyjki rejestru produktu mają duży rozmiar (obejmują duże obszary wartości biznesowej) i są bardzo ubogie w szczegóły. Wraz z upływem czasu historyjki są wielokrotnie dyskutowane przez interesariuszy, właściciela produktu oraz zespół deweloperski i rozbijane na kolekcje mniejszych, bardziej szczegółowych elementów rejestru produktu. W końcu staną się wystarczająco małe i bogate w szczegóły, aby można było wziąć je do sprintu, gdzie zostaną zaprojektowane, zbudowane i przetestowane. Jednak nawet w trakcie sprintu — w toku dyskusji zespołu deweloperskiego z właścicielem produktu — ujawniane będą dalsze szczegóły.

---

<sup>1</sup> Historyjki rejestru produktu to inaczej elementy rejestru produktu — przyp. tłum.



**RYSUNEK 5.1.** Scrum stosuje wskaźniki do docelowych wymagań

Rejestr produktu nie jest niczym innym jak zestawieniem bieżącej kolekcji historyjek wraz z towarzyszącymi im szczegółami — będę o tym pisał szerzej w rozdziale 6.

Chociaż Scrum nie narzuca standardowej formy elementów w rejestrze produktu, wiele zespołów stosuje historyjki użytkownika. Ty nie musisz. Niektóre zespoły wolą przypadki użycia, inne decydują się na stosowanie swoich własnych form.

W tej książce do reprezentacji elementów rejestru produktu używam przeważnie historyjek użytkownika. Czym są historyjki użytkownika, opowiem dokładniej w dalszej części tego rozdziału. Nawet jeśli ostatecznie zdecydujesz się na inne rozwiązanie, omówienie tematu historyjek użytkownika pozwoli Ci lepiej zrozumieć, czego powinieneś oczekwać od innych reprezentacji elementów rejestru produktu.

## Wykorzystanie dyskusji

Wymagania — jako środek przekazu — pozwalają osiągnąć wspólną świadomość tego, co powinno zostać zbudowane. Umożliwiają osobom, które rozumieją, co ma powstać, przekazanie w jasny sposób swoich wizji ludziom, którzy będą musieli wprowadzić je w życie.

Sekwencyjny proces produkcyjny bazuje mocno na spisanych wymaganiach, które mimo swojej imponującej formy mogą w łatwy sposób zostać źle zrozumiane. Przypominam sobie rozmowę z dyrektorem odpowiedzialnym za zarządzanie produktami w odwiedzanej przeze mnie firmie. Zapytałem tę osobę, odpowiedzialną za nadzór nad biznesowymi analizami firmy, w jaki sposób radzą sobie z wymaganiem. Dyrektor odpowiedział mi przykładem: „Pierwszego stycznia moj zespół dostarcza zespołom inżynieryjnym dokument zawierający wymagania, potem spotykamy się 31 grudnia i sprawdzamy, na czym stoimy”.

Zapytałem go, kto z jego zespołu jest dostępny przez cały rok, aby odpowiadać na pytania i wyjaśniać wymagania programistom. Odpowiedź brzmiała „Nikt. Cały czas, jaki mój zespół miał dla tego projektu, został zainwestowany w pracę nad stworzeniem dokumentu z wymaganiami. Teraz analitycy są zajęci spisywaniem wymagań dla innych projektów. Nie martw się, napisaliśmy dobry dokument. Jeżeli programiści lub testerzy będą mieli jakieś pytania, znajdą odpowiedzi, studując dokładnie specyfikację wymagań”.

Wydawało mi się mało prawdopodobne, aby w jego 150-stronicowym dokumencie, opisującym szczegółowo przypadki użycia nowego elektronicznego systemu dokumentacji medycznej, nie występły jakiekolwiek niejasności. Język angielski zwyczajnie nie jest aż tak precyzyjny, a nawet jeśli byłby, osoby spisujące wymagania nie są w stanie zapisać wszystkiego dostatecznie precyzyjnie.

Sposobem znacznie lepiej zapewniającym zbudowanie oczekiwanych cech jest przeprowadzanie regularnych dyskusji pomiędzy osobami, które wiedzą, jakie cechy powinny powstać, a osobami, które je projektują, implementują i testują.

W Scrumie rozmowy są mile widziane — stanowią kluczowy mechanizm omawiania i komunikowania wymogów. Komunikacja ustna jest szybka i zapewnia błyskawiczną informację zwrotną, dzięki czemu łatwiej i taniej osiąga się wzajemne zrozumienie. Dodatkowo rozmowy umożliwiają komunikację dwukierunkową, która może zrodzić pomysły rozwiązymania bieżących problemów oraz nowe idee — coś, co ma małe szanse zaistnienia w wyniku lektury dokumentu.

Rozmowa jest jednak tylko narzędziem. Nie zastępuje wszystkich dokumentów. W Scrumie to rejestr produktu jest „ żywym dokumentem” dostępnym przez okres całego procesu produkcyjnego. Osoby, które nadal potrzebują lub muszą posiadać dokument specyfikujący wymagania, mogą go stworzyć w dowolnym momencie, zbierając razem wszystkie elementy rejestru produktu wraz z towarzyszącymi im szczegółami i umieszczać je w dokumencie o dowolnie wybranym układzie.

## Stopniowe udoskonalanie

W sekwencyjnym procesie produkcyjnym wszystkie wymagania muszą jednocześnie posiadać ten sam poziom uszczegółowienia. Oznacza to przede wszystkim, że zatwierdzony dokument musi specyfikować każdy możliwy wymóg, tak aby zespoły wykonujące projekty, implementację i testowanie mogły zrozumieć, co trzeba zrobić dla zachowania zgodności ze specyfikacją. Ani jeden detale nie powinien zostać pominięty.

Wymuszanie posiadania jednakowego stopnia uszczegółowienia wszystkich wymogów w tym samym czasie niesie ze sobą kilka niedogodności:

- Musimy przewidzieć wszystkie możliwe detale w bardzo wczesnej fazie produkcji, kiedy posiadamy najmniejszą możliwą wiedzę.
- Traktujemy wszystkie wymagania jednakowo niezależnie od ich priorytetu, co zmusza nas do poświęcania już teraz cennych zasobów na tworzenie szczegółowych wymagań, których być może nigdy nie zrealizujemy.
- Tworzymy duży inwentarz wymagań, którego koszt modyfikacji lub porzucenia w przypadku zmiany będzie bardzo drogi.
- Zmniejszamy szanse wyjaśniania i doprecyzowania wymagań poprzez rozmowy, ponieważ wymagania są już „kompletne”.

Zgodnie z ilustracją na rysunku 5.1 w Scrumie nie wszystkie wymagania muszą posiadać ten sam stopień uszczegółowienia w danej chwili. Wymagania, nad którymi będziemy pracować wcześniej, będą mniejsze i bardziej sprecyzowane od tych, które jeszcze przez jakiś czas będą czekać na realizację. Stosujemy strategię **stopniowego udoskonalania** w samą porę do dzielenia dużych, ogólnych wymagań na mniejsze, lepiej opisane elementy.

## Czym są historyjki użytkownika?

**Historyjki użytkownika** to wygodna forma wyrażania oczekiwanej wartości biznesowej, którą można zastosować w przypadku wielu elementów rejestru produktu. Historyjki użytkownika są zapisywane w taki sposób, aby mogły być zrozumiane zarówno przez osoby z biznesowej strony przedsięwzięcia, jak i przez inżynierów. Mają prostą strukturę i stanowią dobrą platformę do prowadzenia konwersacji. Ponadto można je tworzyć z różnym poziomem detali, a następnie stopniowo uszczegółowiać.

Chociaż historyjki można elegancko przystosować do swoich potrzeb, nie uważam ich za jedyną możliwą reprezentację elementów rejestru produktu. Są one zwyczajnie prostym podejściem, które współgra doskonale z fundamentalnymi zasadami zwinności i naszymi potrzebami posiadania wydajnych i skutecznych wskaźników spodziewanej funkcjonalności. Osobiście używam historyjki jako centralnego wskaźnika, do którego dołączam wszelkie informacje mające w mojej opinii związek z wymaganiem lub pomocne w jego uszczegółowieniu. Jeśli odkryję, że historyjka została na siłę dopasowana do pewnej sytuacji (np. posłużyła do reprezentacji określonego błędu), zmieniam reprezentację na inną. Na przykład pewnego razu zobaczyłem, że zespół pisze historyjkę użytkownika zatytułowaną „Jako klient chciałbym, aby system nie powodował uszkodzenia bazy danych”. Myśle, że wszyscy zgodzimy się, iż historyjka użytkownika nie jest najlepszą metodą przedstawienia tego problemu. O wiele lepiej będzie on reprezentowany jako rekord w systemie śledzenia błędów.

Czym zatem są historyjki użytkownika? Ron Jeffries oferuje prostą, a jednocześnie skuteczną metodę myślenia o historyjkach użytkownika [Jeffries, 2001]. Opisuje je jako trzy C, od angielskich słów *card* (karta), *conversation* (rozmowa) i *confirmation* (potwierdzenie).

### Karta

Idea karty jest bardzo prosta. Na początku ludzie mieli zwyczaj (w niektórych przypadkach nadal tak czynią) zapisywać historyjki użytkownika na kartach indeksowych o wymiarach około 7,5 cm × 12,5 cm lub na karteczkach samoprzylepnych (patrz rysunek 5.2).

Wzorzec, według którego zapisywano historyjki użytkownika (pokazany na rysunku 5.2), zawierał klasę użytkowników (rolę użytkownika), zamiar, jaki ta klasa użytkowników pragnie osiągnąć (cel), a także dlaczego użytkownicy chcą osiągnąć dany cel (zysk) [Cohn, 2004]. Część „dzięki czemu” historyjki użytkownika jest opcjonalna, ale powinniśmy umieścić ją w każdej historyjce, chyba że jej cel jest całkowicie zrozumiałą dla wszystkich. Przykład historyjki bazującej na powyższym wzorcu pokazany został po prawej stronie rysunku 5.2.



**RYSUNEK 5.2.** Wzorzec i przykład historyjki użytkownika

Karta nie ma na celu uchwycenia wszystkich informacji składających się na dany wymóg. Celowo używamy małych kart, aby promować zwięzość. Karta powinna zawierać kilka zdań wyrażających sedno lub intencję danego wymogu. Służy jedynie jako wskaźnik dla bardziej szczegółowych dyskusji, jakie będą mieć miejsce pomiędzy interesariuszami, właścicielem produktu i zespołem deweloperskim.

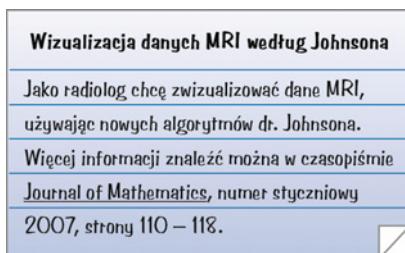
## Rozmowa

Szczegóły wymagań są ujawniane i komunikowane podczas rozmowy pomiędzy zespołem deweloperskim, właścicielem produktu i interesariuszami. Historia użytkownika to zaledwie obietnica przeprowadzenia tej rozmowy.

Pisz „tej rozmowy”, chociaż w rzeczywistości rozmowa zazwyczaj nie jest wydarzeniem jednorazowym — ma charakter nieustającego dialogu. Jedna rozmowa może mieć miejsce w chwili spisania historyjki, kolejna podczas jej uszczegóławiania, następna podczas głosowania, jeszcze inna podczas planowania sprintu (kiedy zespół deweloperski zabierze się za rozpisywanie zadań). W końcu pojawi się cała seria konwersacji, kiedy historyjka będzie projektowana, budowana i testowana w trakcie sprintu.

Jedną z zalet historyjek użytkowników jest to, że przesuwają one punkt ciężkości z pisania na konwersacje. Konwersacje z kolei umożliwiają bogatszą w formie wymianę informacji i współpracę — cechy gwarantujące prawidłowe wyrażenie wymagań i ich zrozumienie przez wszystkich.

Chociaż rozmowy mają na ogół formę ustną, często mogą być i są uzupełniane przez dokumenty. Wynikiem rozmów mogą być szkice interfejsu użytkownika lub spisane szczegóły reguł biznesowych. Oto przykład. Pewnego razu odwiedziłem firmę zajmującą się produkcją oprogramowania służącego do diagnostyki obrazowej. Jedna z ich historyjek została przedstawiona na rysunku 5.3.



**RYSUNEK 5.3.** Historyjka użytkownika z dołączonymi dodatkowymi informacjami

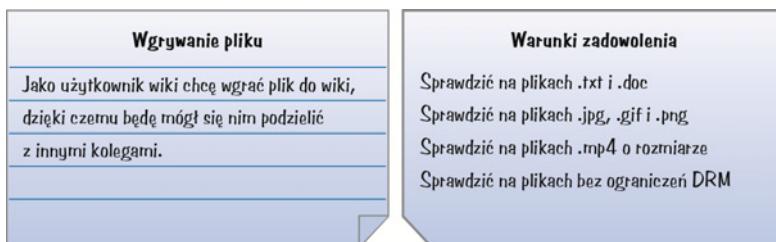
Zauważ, że historyjka użytkownika odsyła do lektury całego artykułu i rozmowy.

Zatem nie generujemy „ton” dokumentów, ale stosujemy historyjki użytkownika i powiązane z nimi karty. Innymi słowy, historyjki użytkownika są dobrym punktem startowym do wydobycia sedna poszukiwanego rozwiązania, a także przypomnieniem o konieczności przeprowadzenia bardziej szczegółowej dyskusji na temat wymogów, kiedy nadziejdie odpowiednia pora. W świetle powyższych stwierdzeń należy dodać, że historyjki użytkownika mogą i powinny być uzupełniane o wszelkie informacje w formie pisemnej, które są w stanie lepiej wyjaśnić, na czym polega spodziewane rozwiązanie.

## Potwierdzenie

Historyjka użytkownika zawiera również informacje potwierdzające w formie warunków zadowolenia. Są to kryteria akceptacji, które klarują spodziewane zachowanie. Służą one zespołowi deweloperskiemu do lepszego zrozumienia, co należy zbudować i przetestować, a także właścielowi produktu do potwierdzenia, że implementacja historyjki użytkownika jest dla niego satysfakcjonująca.

Przednia część karty zawiera kilka wierszy opisujących historyjkę, natomiast tylna część może specyfikować warunki zadowolenia (patrz rysunek 5.4).



**RYSUNEK 5.4.** Warunki zadowolenia historyjki użytkownika

Warunki te mogą być wyrażone jako kryteria akceptacji wysokiego rzędu. Nie będą to jednak jedynie testy wykonywane podczas prac deweloperskich nad historyjką. W praktyce oprócz kilku testów akceptacyjnych związanych z historyjką zespół będzie dysponował znacznie (być może dziesięciokrotnie lub stukrotnie) większą liczbą testów na niższym, bardziej technicznym poziomie, o których właściciel produktu naprawdopodobnie nie ma nawet pojęcia.

Testy akceptacyjne historyjki istnieją z kilku powodów. Po pierwsze, z perspektywy właściciela produktu stanowią istotny sposób uchwycenia i zakomunikowania sposobu sprawdzenia, czy historyjka została zaimplementowana poprawnie.

Mogą być również pomocne podczas tworzenia początkowych historyjek i udoskonalania ich w miarę pojawiania się kolejnych szczegółów. Takie podejście nazywane jest czasem **specyfikacją przez przykład lub produkcją sterowaną testami akceptacyjnymi**. Idea wydaje się być w miarę intuicyjna. Dyskusje o historyjkach skupią się często na definiowaniu konkretnych przykładów lub oczekiwanych zachowań. Na przykład rozmowa w przypadku historyjki „Wgrywanie pliku” z rysunku 5.4 mogłaby przebiegać następująco:

*Początkowo ograniczmy rozmiar wgrywanych plików do 1 GB. Upewnijmy się, że potrafimy poprawnie załadować plik tekstowy i graficzny. Z powodów prawnych nie możemy pozwolić, aby do wiki wgrane zostały jakiekolwiek pliki z dołączoną informacją DRM<sup>2</sup>.*

<sup>2</sup> DRM — ang. *digital rights management*, zobacz [http://pl.wikipedia.org/wiki/Digital\\_rights\\_management](http://pl.wikipedia.org/wiki/Digital_rights_management) — przyp. tłum.

Jeżeli używamy narzędzia takiego jak Fit lub FitNesse, możemy skorzystać z konwencji polegającej na przedstawieniu powyższych testów w formie tabeli, takiej jak tabela 5.1, która pokazuje różne przykładowe rozmiary pliku oraz fakt, czy są one poprawne lub też nie.

**TABELA 5.1.** Przykład testu automatycznego

Rozmiar	Poprawność
0	Tak
1 073 741 824	Tak
7 073 741 825	Nie

Budowanie tego typu przykładów pomaga w procesie tworzenia i precyzowania historyjek, a także zapewnia posiadanie (automatycznych) testów akceptacyjnych dla każdej z nich.

## Poziom szczegółowości

Historyjki użytkownika są doskonałym środkiem do przeprowadzania elementów o wartości dla klienta lub użytkownika poprzez scrumowy proces wytwarzania wartości. Jeżeli jednak posiadamy tylko jeden rozmiar historyjki (rozmiar pozwalający na swobodne umieszczenie w sprincie o krótkim czasie trwania), trudno będzie przeprowadzić planowanie na wysokim poziomie, a następnie skorzystać z mechanizmu stopniowego udoskonalania.

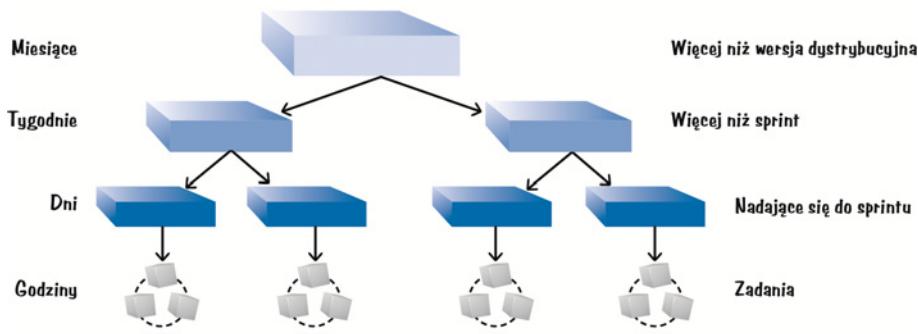
Historyjki używane na poziomie sprintów są zbyt małe i zbyt liczne, aby można było wykorzystać je do planowania na wyższym poziomie czy też na poziomie planowania wersji dystrybucyjnej. Na tych poziomach potrzebujemy elementów mniej szczegółowych i bardziej abstrakcyjnych. W przeciwnym razie utknimy w morzu przeważnie nieistotnych detali. Wyobraź sobie, że masz 500 bardzo małych historyjek i zostałeś poproszony o dostarczenie opisu proponowanego produktu kadrze menedżerskiej w celu zapewnienia środków finansowych na dalszą działalność. Spróbuj przypisać priorytety wśród tych 500 historyjek, aby zdefiniować, co znajdzie się w następnej wersji dystrybucyjnej.

Ponadto jeśli istnieje tylko jeden (mały) rozmiar historyjki, będziemy zobowiązani do zdefiniowania wszystkich wymagań z bardzo dużym poziomem szczegółów na długo, zanim powinniśmy to zrobić. Posiadanie jedynie małych historyjek przeciwodziła zaletom stopniowego udoskonalania w samą porę i w stopniu wystarczającym.

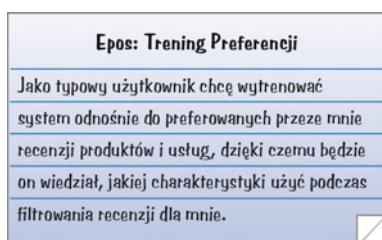
Na szczęście historyjki użytkownika mogą służyć do opisywania potrzeb klientów i użytkowników na różnych poziomach abstrakcji (patrz rysunek 5.5).

Rysunek 5.5 przedstawia historyjki na różnych poziomach abstrakcji. Największe historyjki mają rozmiar kilku miesięcy pracy i mogą się rozciągać na jedną lub nawet kilka wersji dystrybucyjnych. Wiele osób określa je mianem **eposów**, czyniąc w ten sposób aluzję, jakoby były one historyjkami o rozmiarze porównywalnym z takimi dziełami jak *Władca Pierścieni* czy *Wojna i pokój*. Eposy pomagają w całościowym dostrzeżeniu oczekiwanej funkcjonalności (patrz rysunek 5.6).

Nigdy nie umieścilibyśmy w sprincie eposu z zamarem produkcji, ponieważ jest on o wiele za duży i pozbawiony szczegółów. Eposy nadają się za to doskonale jako wskaźniki do dużych zbiorów bardziej szczegółowych historyjek, które powstaną w odpowiednim czasie w przyszłości. Użycie eposów omówię podczas omawiania tematu planowania produktu w rozdziale 17.



RYSUNEK 5.5. Hierarchia abstrakcji historyjek użytkownika

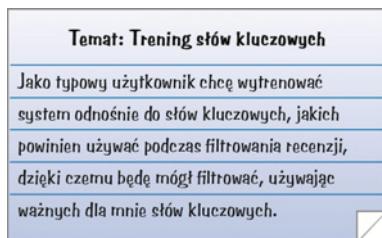


RYSUNEK 5.6. Przykładowy epos

Historyjki następnego rozmiaru, pokazane na rysunku 5.5, mają często wartość kilku tygodni pracy i w związku z tym są również zbyt duże, aby zmieścić się w pojedynczym sprincie. Niektóre zespoły określają je mianem **cech**.

Najmniejszą formą historyjek użytkownika jest ta, którą ja zazwyczaj określam mianem **historyjki**. W celu uniknięcia nieporozumienia z eposami, cechami lub innymi większymi elementami, które również są „historyjkami”, niektórzy określają je mianem **historyjek nadających się do sprintu** lub **historyjek nadających się do implementacji**, wskazując w ten sposób, że ich rozmiar jest rzędu dni, a zatem dostatecznie mały, aby zmieściły się w sprintie i zostały zaimplementowane. Przykład historyjki nadającej się do implementacji pokazuje rysunek 5.2.

Niektóre zespoły stosują dodatkowo pojęcie **tematu** do opisania zbioru powiązanych ze sobą historyjek. Tematy zapewniają wygodny sposób wskazania, iż grupa historyjek ma coś wspólnego ze sobą — na przykład należą do tego samego obszaru funkcjonalności. Na rysunku 5.7 przedstawiony został temat grupujący historyjki mówiące o tym, w jaki sposób przeprowadzić trening słów kluczowych.



RYSUNEK 5.7. Przykład tematu

Często traktuję temat jako kartę podsumowującą plik notek spiętych gumką w celu wskazania, że są one podobne do siebie w obszarze, który uważamy za istotny.

Zadania znajdują się na poziomie poniżej sprintów i zazwyczaj realizowane są przez pojedyncze osoby lub osoby pracujące w parach. Realizacja zadania wymaga zazwyczaj kilku godzin. Kiedy dochodzimy do poziomu zadań, specyfikujemy, *jak* należy coś zbudować, w przeciwieństwie do tego *co* należy zbudować (reprezentowanego przez eposy, cechy i historyjki). Zadania nie są historyjkami, zatem pisząc historyjki, powinniśmy unikać spisywania szczególnów specyficznych dla zadań.

Należy pamiętać, że określenia takie jak *epos*, *cecha*, *historyjka* i *zadanie* służą jedynie wygodzie i nie stanowią żadnego uniwersalnego podejścia. Nie ma znaczenia, jakich etykietek użyjesz, o ile będziesz stosował je konsekwentnie. Istotne jest to, aby zrozumieć, że historyjki mogą funkcjonować na różnych poziomach, a ich wykorzystanie w taki sposób wspiera nasze wysiłki w planowaniu na różnych poziomach abstrakcji i progresywne dzielenie dużych elementów na mniejsze w miarę postępu prac.

## Inwestuj w dobre historyjki

Skąd wiemy, że napisane przez nas historyjki są dobre? Bill Wake określił sześć kryteriów (składających się na skrót INVEST) pozwalających ocenić, czy nasze historyjki spełniają swój cel, czy też należy je dopracować [Wake, 2003].

Kryteria INVEST to: niezależność (ang. *Independent*), negocjowalność (ang. *Negotiable*), wartościowość (ang. *Valuable*), ocenialność (ang. *Estimatable*), dobry (mały) rozmiar (ang. *Sized correctly*) i testowalność (ang. *Testable*). Kiedy połączymy informacje wynikające z zastosowania się do każdego z tych kryteriów, otrzymamy jasny obraz sytuacji, obrazujący, czy potrzebne są dodatkowe zmiany w celu utworzenia historyjki. Prześledźmy każde z powyższych kryteriów.

### Niezależność

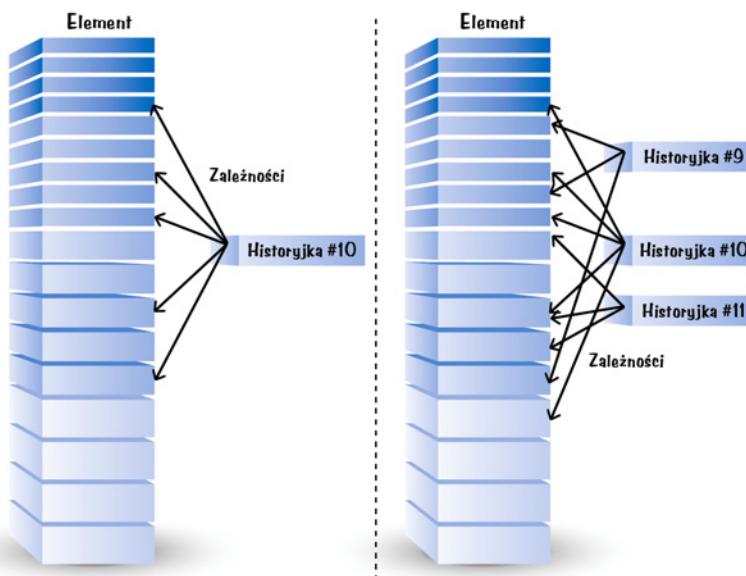
Na ile jest to możliwe, historyjki użytkownika powinny być *niezależne* lub przynajmniej luźno powiązane ze sobą. Historyjki cechujące się dużym stopniem zależności od innych są trudne do wycenienia, nadania priorytetów i zaplanowania. Weźmy za przykład lewą stronę rysunku 5.8 — historyjka #10 zależy od wielu innych elementów.

Zanim będziemy mogli rozpocząć pracę nad historyjką #10, musimy zbudować wszystkie historyjki, od których ona sama zależy. W tym pojedynczym przypadku może nie wyglądać to tak źle, ale wyobraź sobie, że masz całe mnóstwo różnych historyjek powiązanych ze sobą tak, jak przedstawia to prawa strona rysunku 5.8. Próba nadania priorytetów wszystkim tym historyjkom, a także zdecydowanie, które z nich wziąć do sprintu, byłaby co najmniej trudne.

Przy zastosowaniu kryterium *niezależności* celem nie jest całkowita eliminacja powiązań, ale takie pisanie historyjek, aby zminimalizować liczbę zależności.

### Negocjowalność

Szczegóły historyjki powinny być *negocjowalne*. Historyjki nie są spisany kontraktem w formie przyjętej z góry specyfikacji wymagań. Są raczej wskaźnikiem do rozmów, w trakcie których negocjowane będą ich szczegóły.



RYSUNEK 5.8. Historyjki mocno od siebie zależne

Dobre historyjki w jasny sposób oddają sedno oczekiwanej funkcjonalności biznesowej i dodatkowo wyjaśniają, czemu dana funkcjonalność jest pożądana. Jednocześnie pozostawiają przestrzeń właścielowi produktu, interesariuszom i zespołowi do wynegocjowania szczegółów.

Ta negocjalność pomaga wszystkim zaangażowanym osobom uniknąć wpadnięcia w pułapkę mentalności wskazywania winnego — „my kontra oni”, która często ma miejsce podczas stosowania przyjętych z góry specyfikacji wymagań. Przy negocjalnych historyjkach deweloperzy nie mogą powiedzieć: „Hej, jeśli chciałeś to mieć, powinieneś był zapisać to w specyfikacji”, ponieważ szczegóły będą negocjowane właśnie z nimi. Z drugiej strony, osoby odpowiedzialne za stronę biznesową nie mogą zarzucić: „Hej, najwyraźniej nie zrozumieliście specyfikacji wymagań, bo zdబو-waliście nie to, co trzeba”, ponieważ oni będą często uczestniczyć w dialogu z deweloperami, aby upewnić się, że istnieje wzajemne zrozumienie. Tworzenie negocjalnych historyjek zapobiega problemom wynikającym ze stosowania przyjętych z góry specyfikacji wymagań przez jasne zakomunikowanie, iż potrzebny jest dialog.

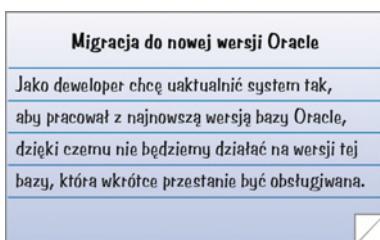
Często spotykany przykład złamania reguły negocjalności ma miejsce, kiedy właściciel produktu mówi zespołowi deweloperskiemu, *jak* należy zaimplementować historyjkę. Historyjki powinny mówić co i dlaczego, nie jak. Negocjowanie *jak* powoduje zmniejszanie szans zespołu na bycie innowacyjnym. Powstające w ten sposób **marnotrawstwo innowacyjności** może mieć druzgocące skutki ekonomiczne.

Istnieją jednak przypadki, kiedy to, *jak* coś zostanie zbudowane, ma istotne znaczenie dla właściciela produktu. Przykładem mogą być regulacje prawne narzucające zaprogramowanie cechy w określony sposób lub też ograniczenia biznesowe nakazujące użycie określonej technologii. W takich sytuacjach historyjki będą odrobinę mniej negocjalne ze względu na wymagalność pewnego aspektu „jak”. Taka sytuacja jest zrozumiała — nie wszystkie historyjki są w pełni negocjalne, ale większość z nich tak.

## Wartościowość

Historyjki powinny być *wartościowe* dla klienta, użytkownika lub obu. Klienci (lub kandydaci na klientów) wybierają i płacą za produkt. Użytkownicy faktycznie korzystają z produktu. Jeżeli historyjka nie ma wartości dla żadnego z nich, nie ma dla niej miejsca w rejestrze produktu. Nie wyobrażam sobie powiedzenia „Historyjka #10 nie ma wartości, ale mimo to zbudujmy ją”. Nigdy nie zrobilibyśmy czegoś takiego. Przepisalibyśmy ją tak, aby nabrała wartości dla klienta lub użytkownika, lub pozbylibyśmy się jej.

Co z historyjkami, które mają wartość dla deweloperów, ale wartość ta nie jest oczywista dla klientów lub użytkowników? Czy można posiadać **historyjki inżynieryjne**, jak ta pokazana na rysunku 5.9?

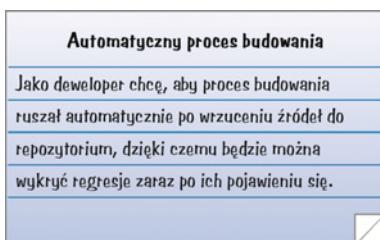


**RYSUNEK 5.9.** Przykład historyjki inżynieryjnej

Podstawowym problemem z historyjkami inżynieryjnymi jest to, że właściciel produktu może nie dostrzegać w nich żadnej wartości, co bardzo utrudnia lub wręcz uniemożliwia nadanie im priorytetów odpowiednio wysokich w porównaniu z historyjkami o wartości biznesowej. Zaistniecie historyjki inżynieryjnej wymaga zrozumienia przez właściciela produktu, dlaczego ponosi jej koszty i w związku z tym, jaką wartość dostarczy ona na końcu.

W przypadku historyjki „Migracja do nowej wersji Oracle” właściciel produktu może początkowo nie rozumieć, dlaczego zmiana bazy ma swoją wartość. Kiedy jednak zespół wyjaśni ryzyko związane z kontynuacją prac deweloperskich na nieobsługiwanej wersji bazy danych, właściciel produktu może zdecydować, iż migracja ma wystarczająco dużą wartość, aby odsunąć w czasie prace nad nowymi cechami produktu i dokonać migracji. Doceniając wartość historyjki inżynieryjnej, właściciel produktu może potraktować ją tak jak każdą inną historyjkę o wartości biznesowej i podjąć świadomy kompromis. Efektem będzie włączenie tej historyjki do rejestru produktu.

W praktyce jednak większość historyjek inżynieryjnych (jak ta z rysunku 5.10) nie powinna być umieszczana w rejestrze produktu.



**RYSUNEK 5.10.** Niepożądana historyjka inżynieryjna

Zamiast tego powinny one być traktowane jako zadania realizowane w ramach innych historyjek dostarczających rzeczywistą wartość biznesową. Jeżeli zespół deweloperski posiada solidną definicję ukończenia, nie ma konieczności tworzenia historyjek inżynierijnych, ponieważ związana z nimi praca niejako wynika z definicji ukończenia.

Sednem kryterium *wartościowości* jest wymóg posiadania wartości przez wszystkie historyjki w rejestrze produktu (opłacalności inwestowania w nie). Wymóg ten musi być spełniony z punktu widzenia właściciela produktu, który reprezentuje interesy klientów i użytkowników. Nie wszystkie historyjki są niezależne, nie wszystkie są również w pełni negocjalne, ale wszystkie muszą posiadać wartość.

## Ocenialność

Historyjki powinny dać się *oceniać* przez zespół deweloperski, który będzie je projektował, budował i testował. Ocena historyjki określa jej rozmiar, czyli niezbędny wysiłek deweloperski, oraz koszty (większe historyjki wymagają większego wysiłku, a zatem większych kosztówtworzenia w porównaniu z mniejszymi historyjkami).

Znajomość rozmiaru historyjki daje zespołowi scrumowemu informację, która pozwala podjąć działanie. Właściciel produktu na przykład musi znać rozmiar historyjki, aby nadać jej priorytet w rejestrze produktu. Natomiast zespół scrumowy na podstawie rozmiaru historyjki decyduje, czy wymaga ona dalszego uszczegółowiania lub też może podzielić ją na mniejsze elementy. Jeżeli planujemy wkrótce podjąć pracę nad większą historyjką, będzie trzeba podzielić ją na historyjki o mniejszym rozmiarze.

Jeżeli zespół deweloperski nie jest w stanie ocenić historyjki, oznacza to, że jest ona zbyt duża lub niejasna. Inna możliwość to brak w zespole wiedzy pozwalającej na określenie jej rozmiaru. Jeżeli jest zbyt duża, zespół wspólnie z właścicielem produktu muszą podzielić ją na historyjki o rozmiarach pozwalających na lepsze zarządzanie nimi. W przypadku braku wiedzy w zespole trzeba będzie podjąć pewne działania eksploracyjne, aby zdobyć niezbędne informacje (będę o tym pisał już niedługo).

## Dobry (mały) rozmiar

Historyjki powinny mieć *odpowiedni rozmiar*, pasujący do czasu, w którym chcemy rozpocząć nad nimi pracę. Historyjki przeznaczone do sprintów powinny być *małe*. Wykonując kilkutygodniowy sprint, chcemy mieć kilkanaście historyjek o rozmiarze kilku dni każda. W przypadku dwutygodniowego sprintu nie chcemy historyjki o rozmiarze dwóch tygodni, ponieważ istnieje zbyt duże ryzyko, że nie damy rady jej ukończyć.

Zatem w ostatecznym rozrachunku potrzebujemy historyjek małych, chociaż duży rozmiar historyjki niekoniecznie oznacza, że jest ona zła. Historyjka ma odpowiedni rozmiar tak długo, jak długo nie chcemy jej brać do sprintu. Dzielenie już teraz eposu na mniejsze historyjki może okazać się kompletnym marnotrawstwem naszego czasu. Oczywiście jeśli mamy epos, który chcemy wziąć do następnego sprintu, jego rozmiar jest zbyt duży i czeka nas praca nad jego podziałem. Przed zastosowaniem się do tego kryterium musisz jednak rozważyć, *kiedy* zamierzysz podjąć pracę nad daną historyjką.

## Testowalność

Historyjki powinny być *testowalne* w sposób dający dwa możliwe wyniki — albo przejdą zвязane z nimi testy, albo nie. Testowalność oznacza posiadanie dobrych kryteriów akceptacji (związkanych z warunkami zadowolenia) dla historyjki. Mówimy tutaj o aspekcie „potwierdzania” historyjki użytkownika, o którym wspomniałem wcześniej.

Jak moglibyśmy stwierdzić pod koniec sprintu, że historyjka została wykonana, nie posiadając kryterium testowalności? Testy akceptacyjne niosą ze sobą istotne detale dotyczące historyjki i w związku z tym są potrzebne, zanim zespół będzie w stanie ocenić jej rozmiar.

Nie zawsze będzie istnieć możliwość lub konieczność przetestowania historyjki. Przykładem mogą być eposy, które najprawdopodobniej nie mają związków ze sobą testów lub ich nie potrzebują (eposy nie są historyjkami przeznaczonymi bezpośrednio do budowy).

Czasem może pojawić się historyjka, którą właściciel produktu uważa za cenną, chociaż nie istnieje żadna praktyczna metoda przetestowania jej. Z historyjkami tego typu wiążą się na ogół wymagania niefunkcjonalne, na przykład „Jako użytkownik chcę, aby system pracował przez 99,999% czasu”. Chociaż kryteria akceptacji wydają się jasne, nie ma zestawu testów, które można byłoby przeprowadzić po uruchomieniu systemu, aby udowodnić, iż uzyskany został taki poziom aktywnej pracy systemu. Niemniej jednak sam wymóg ma swój sens i będzie miał wpływ na prace projektowe.

## Wymagania niefunkcjonalne

**Wymagania niefunkcjonalne** reprezentują wymogi na poziomie systemu. Osobiście często zapisuję wymagania niefunkcjonalne w formie historyjek użytkownika (patrz lewa strona rysunku 5.11), chociaż nie czuję się do tego zobligowany, szczególnie jeśli wymagania te wydają się być nie na miejscu lub należałyby je zapisać w innej formie (patrz prawa strona rysunku 5.11).

Wersje językowe	Obsługa przeglądarek
Jako użytkownik chcę, aby interfejs użytkownika był w języku angielskim, w języku rumuńskim i w języku złożonym, dzięki czemu będzie istnieć statystyczne prawdopodobieństwo, że interfejs obsługuje wszystkie 70 wymaganych języków.	System musi obsługiwać następujące przeglądarki: IE8, IE9, Firefox 6, Firefox 7, Safari 5 i Chrome 15.

RYSUNEK 5.11. Wymagania niefunkcjonalne

Wymogi systemowe, niefunkcjonalne mają duże znaczenie, ponieważ wpływają na projekt i testowanie większości lub wszystkich historyjek w rejestrze produktu. Przykładem może być niefunkcjonalny wymóg „Obsługi przeglądarek” (prawa strona rysunku 5.11) — obecny w niemal każdym projekcie webowym. Kiedy zespół produkuje cechy przeznaczone dla sieci WWW, musi się upewnić, że będą one działać we wszystkich wyspecyfikowanych przeglądarkach.

Zespół musi również zdecydować, kiedy przeprowadzić testy we wszystkich przeglądarkach. Każdy wymóg niefunkcjonalny jest podstawowym celem do umieszczenia w zespołowej definicji ukończenia prac. Jeżeli zespół umieści niefunkcjonalny wymóg „Obsługi przeglądarek” w definicji

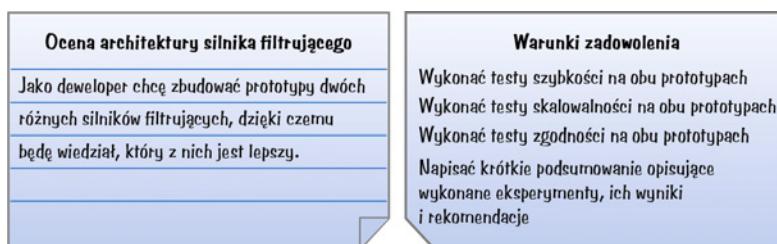
ukończenia prac, każda nowa cecha utworzona w trakcie sprintu będzie musiała przejść testy we wszystkich wymienionych przeglądarkach. W przypadku, gdy nowa cecha nie działa w którejś z przeglądarek, historyka jest nieukończona.

Osobiście sugeruję, aby zespoły umieszczały w swojej definicji ukończenia jak najwięcej wymogów niefunkcjonalnych. Czekanie z testowaniem tych wymogów do późnej fazy produkcji odraca możliwość uzyskania informacji zwrotnych dotyczących absolutnie krytycznej charakterystyki działania systemu.

## Historyjki pozyskiwania wiedzy

Czasami zachodzi potrzeba utworzenia historyjki w rejestrze produktu, której celem jest uzyskanie pewnych informacji. Być może nie posiadamy dostatecznie dużo wiedzy odnośnie do produktu lub procesu jego budowania, aby móc brnąć naprzód. Wtedy, tak jak opisywałem to w rozdziale 3., musimy dokonać eksploracji. Taka eksploracja może nosić różne nazwy: *prototypowanie, udowadnianie koncepcji, eksperymentowanie, studiowanie* itd. Są to wszystko formy działalności mające na celu „zakup” informacji.

Ja często wykorzystuję historyjki jako wskaźniki do prac eksploracyjnych (patrz rysunek 5.12).



RYSUNEK 5.12. Historyjka pozyskiwania wiedzy

W tym przykładzie zespół chce ocenić dwie możliwe architektury nowego silnika filtrującego. Pada propozycja, aby wykonać prototypy obu architektur, a następnie na obu przeprowadzić testy szybkości, skalowalności i zgodności. Wynikiem tego działania będzie krótka informacja opisująca przeprowadzone eksperymenty, otrzymane wyniki oraz rekomendacja zespołu odnośnie do zalecanego postępowania.

Ta historyjka pozyskiwania wiedzy wygląda jak historyjka inżynierijna, a tak jak wspomniałem wcześniej, wartość biznesowa dowolnej historyjki inżynierijnej musi zostać potwierdzona przez właściciela produktu. Ponieważ właściciele produktu myślą w kategoriach ekonomicznych, musi istnieć ekonomiczne uzasadnienie wykonania tego typu pracy. Zazwyczaj istnieje przekonujący argument (z punktu widzenia inżynierów) do wykonania historyjki pozyskiwania wiedzy, ponieważ zespół stoi w martwym punkcie i nie może ruszyć dalej do momentu poznania informacji będących wynikiem tej historyjki. Pytanie, na które musi sobie odpowiedzieć zespół scrumowy, brzmi: Czy wartość pozyskanych informacji przewyższa koszt ich uzyskania?

Oto jak zespół może podejść do znalezienia odpowiedzi na to pytanie. Po pierwsze, musimy poznać koszt wspomnianego prototypowania. Żaden dobry właściciel produktu nie zezwoli na nieograniczoną eksplorację. Zespół może nie być w stanie odpowiedzieć na pewne pytania do momentu

wybrania konkretnej architektury, ale musi umieć odpowiedzieć, ile wysiłku chce włożyć w zakupienie wiedzy niezbędnej do wyboru architektury. Prosimy zatem, aby zespół wycenił historyjkę pozyskiwania wiedzy.

Załóżmy, że z rozmiaru historyjki wynika konieczność pracy nad nią całego zespołu przez jeden sprint. Wiemy, kto należy do zespołu i jak długo trwa sprint, zatem wiemy, ile będzie kosztować pozyskanie wiedzy (niech będzie to 10 tysięcy złotych). Teraz musimy poznać wartość informacji.

Jeden ze sposobów oceny wartości wygląda następująco. Wyobraź sobie, że rzucam monetą. Jeżeli wypadnie reszka, wykonujemy architekturę A, jeżeli orzeł — architekturę B. Teraz proszę zespół, aby ocenił koszt błędnej decyzji. Na przykład: jeśli rzuciłem monetą, wyszła reszka, zaczynamy budować cechy biznesowe architektury A i okazuje się, że architektura A jest złym podejściem, to jaki będzie koszt naprawienia tej decyzji i zbudowania wszystkiego na bazie architektury B? Przyjmijmy, że zespół ocenił koszt takiej operacji na 500 tysięcy złotych.

Teraz posiadamy dostatecznie dużo informacji, aby podjąć ekonomicznie uzasadnioną decyzję. Czy jesteśmy w stanie wydać 10 tysięcy złotych, aby kupić informacje, których wartość oczekiwana to 250 tysięcy (istnieje 50% szans, że nasz losowy wybór byłby prawidłowy)? Oczywiście, że taka decyzja wydaje się sensowna. Właściciel produktu może bez trudu usprawiedliwić obecność historyjki pozyskiwania wiedzy w rejestrze produktu.

Dla ostatecznego potwierdzenia słuszności użycia rachunku ekonomicznego do potwierdzenia wartości historyjek pozyskiwania wiedzy spróbujmy zmienić nieco nasze liczby. Co by się stało, gdyby odpowiedź zespołu na pytanie o koszt podjęcia błędnej decyzji brzmiała 15 tysięcy? Wtedy decyzja o wykonaniu historyjki z prototypowaniem byłaby błędna. Po co wydawać 10 tysięcy, aby kupić informacje o oczekiwanej wartości 7,5 tysiąca złotych? Znacznie prościej będzie rzucić monetą (lub przyjąć rozsądne założenie), a jeśli okaże się, że popełniliśmy błąd, wykonać całą pracę przy użyciu drugiej architektury. Biorąc pod uwagę dzisiejsze, nieustannie rozwijające się technologie, ten scenariusz nie jest aż tak naciągany, jak mogłoby się wydawać. Jest to przykład strategii nazywanej przez niektórych **szybką porażką** (spróbuj czegoś, zdobądź informację zwrotną i dokonaj niezwłocznej inspekcji oraz adaptacji).

## Zbieranie historyjek

W jaki sposób powstają historyjki użytkownika? Tradycyjne metody gromadzenia wymagań polegają na pytaniu użytkowników o ich potrzeby. Ja osobiście nigdy nie odnoszę szczególnych osiągnięć na tym polu. Z moich doświadczeń wynika, że użytkownicy są znacznie lepszymi krytykami niż autorami.

Zatem kiedy spytasz użytkownika „Czego potrzebujesz?”, ten może nie być w stanie odpowiedzieć na Twoje pytanie, a nawet jeśli dostaniesz odpowiedź i zbudujesz dokładnie to, o co on poprosił, w następnym podejściu usłyszysz „Tak, to jest dokładnie to, o co prosiłem, ale teraz kiedy to widzę, chcę czegoś innego”. Przypuszczam, że każdy z nas przeżył kiedyś podobną sytuację.

Lepszym podejściem jest zaangażowanie użytkowników jako członków zespołu, który wskazuje, co należy zbudować, a następnie analizuje wykonaną rzecz. Chcąc promować zaangażowanie na takim poziomie, wiele organizacji woli przeprowadzać warsztaty pisania historyjek użytkownika jako podstawowy środek pozyskiwania przynajmniej początkowego zbioru historyjek. Inne firmy z kolei tworzą mapę historyjek, co pozwala im zbudować lepsze zrozumienie kontekstu użytkownika. Opiszę po-krótko każdą z tych technik.

## Warsztaty pisania historyjek

Celem **warsztatów pisania historyjek** jest przeprowadzenie burzy mózgów odnośnie do oczekiwanej wartości biznesowej i stworzenie wydmuszek historyjek wskazujących, czym powinna być nowa usługa lub produkt.

W warsztatach biorą na ogół udział właściciel produktu, mistrz młyna i zespół deweloperski, a także wewnętrzni i zewnętrzni interesariusze. Większość warsztatów trwa mniej więcej od kilku godzin do kilku dni. Rzadko zdarzało mi się słyszeć o dłuższych warsztatach — nie uważam również, aby powinny one trwać dłużej. Celem nie jest stworzenie kompletnej listy historyjek (w sposób zbliżony do kompletnej specyfikacji wymagań w sekwencyjnym procesie produkcyjnym). Warsztaty są zazwyczaj skupione na konkretnym problemie. Ja na przykład często przeprowadzam warsztaty z początkowym planowaniem wersji dystrybucyjnej, aby stworzyć listę historyjek — kandydatów do następnej wersji dystrybucyjnej (więcej na ten temat znajdziesz w rozdziale 18.).

Jeżeli są to pierwsze warsztaty, zaczynam zwykle od analizy ról użytkowników. Celem jest ustalenie wszystkich możliwych ról użytkowników, jakich będzie można użyć do obsadzenia części **rola użytkownika** w opisie historyjki („Jako <rola użytkownika> chcę...”). Niewykluczone, że nasi użytkownicy zostali już dobrze zdefiniowani w wyniku innych działań podjętych przez ludzi z działów marketingu i analiz.

Możemy również mieć **persony**, będące prototypowymi postaciami reprezentującymi cechy określonych ról. Dla przykładu: „Laura” razem z towarzyszącym jej opisem może być personą odpowiadającą roli siedmio- do dziewięcioletniej dziewczynki grającej w gry wideo przeznaczone dla dziewcząt. Po zdefiniowaniu Laury moglibyśmy zacząć pisać historyjki z Laurą w roli użytkownika, zamiast stosować jakiś bardzo abstrakcyjny opis w stylu „Młodej graczki płci żeńskiej”. Oto przykład: „Jako Laura chcę móc wybierać spośród wielu różnych sukienek, dzięki czemu będę mogła zmienić awatar zgodnie z moimi upodobaniami”.

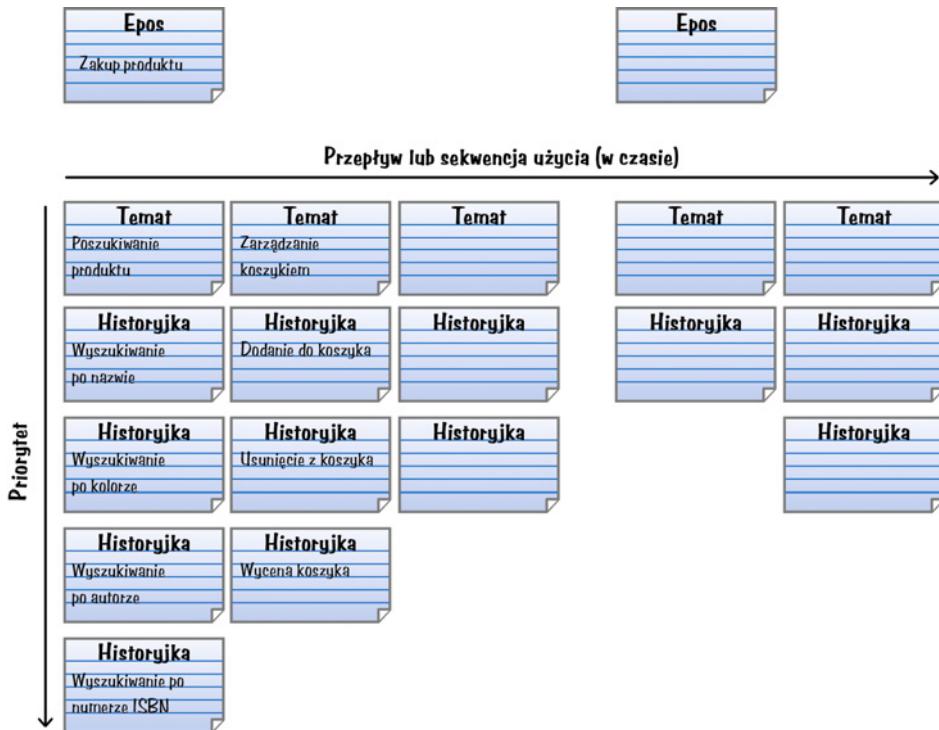
Podczas warsztatów nie ma żadnej standardowej metody tworzenia historyjek. Niektóre zespoły wolą poruszać się z góry na dół, inne od dołu do góry. Podejście z góry na dół wymaga rozpoczęcia od dużej historyjki (takiej jak epos), a następnie skoncentrowania swoich wysiłków na pisaniu mniejszych historyjek związanych z tym eposem.

Inna możliwość to zacząć od dołu i przejść bezpośrednio do burzy mózgów nad historyjkami związanymi z następną wersją dystrybucyjną istniejącego systemu. Nie można powiedzieć, że któreś z tych podejść jest lepsze od drugiego. Stosuj to, które w Twojej opinii daje lepsze rezultaty, lub oba jednocześnie, biorąc z każdego to, co uważasz za najcenniejsze.

## Mapa historyjek

Mapa historyjek to technika spopularyzowana przez Jeffa Pattona [Patton, 2009], która podchodzi do tworzenia zestawu historyjek z punktu widzenia użytkownika. Podstawowa idea polega na przekształceniu akcji użytkownika na wysokim poziomie w pewien przepływ działań, który następnie można zamienić na zbiór szczegółowych zadań (patrz rysunek 5.13).

Do opisania hierarchii w mapie historii Patton używa określeń takich jak *aktywność*, *zadanie* i *podzadanie*. Dla zachowania zgodności z terminologią, którą wprowadziłem wcześniej, będę się posługiwał *eosem*, *tematem* i *historyjką nadającą się do sprintu*.



**RYSUNEK 5.13.** Mapa historyjek

Na najwyższym poziomie znajdują się eposy, które reprezentują obszerne aktywności o miejscowości dla użytkownika wartości ekonomicznej — na przykład epos „Zakup produktu”.

Następnie zastanawiamy się nad sekwencją typowych zadań (reprezentowanych przez tematy), które odpowiadają temu eposowi. Tematy układamy na osi czasu w taki sposób, aby tematy pojawiające się wcześniej były umiejscowione na lewo od tych, które mają miejsce później. Na przykład temat „Szukanie produktu” pojawi się po lewej stronie tematu „Zarządzanie koszykiem”.

Każdy temat jest rozkładany na zbiór historyjek nadających się do implementacji ułożonych pionowo według priorytetów (a dokładniej mówiąc według stopnia pożąданia, ponieważ najprawdopodobniej nie zostały one jeszcze wycenione i nie możemy mówić o priorytetach, nie znając kosztów). Nie wszystkie historyjki w ramach tematu muszą znaleźć się w tej samej wersji dystrybucyjnej. Na przykład historyjka „Wyszukiwanie po kolorze” nie musi trafić do pierwszej wersji, podczas gdy „Wyszukiwanie po nazwie” powinno trafić do tej wersji.

Mapa historyjek łączy ze sobą koncepcję projektowania z punktu widzenia użytkownika z ideą dekompozycji. Dobre mapy historyjek pokazują przepływ aktywności widzianych oczami użytkownika i dostarczają kontekstu pomagającego w zrozumieniu poszczególnych historyjek, a także ich związku z większymi jednostkami wartości dla użytkownika.

Nawet jeśli nie zamierzasz podejść do tworzenia mapy historyjek w sposób formalny, muszę powiedzieć, że osobiście uważam ideę przepływu akcji za bardzo pomocną podczas moich warsztatów pisania historyjek użytkownika. Pozwala ona skupić się na pisaniu historyjek w taki sposób, aby dostarczać użytkownikowi wartość w postaci kompletnego zbioru działań.

Rozumienie przepływu działań ułatwia nam wykrycie ewentualnych istotnych historyjek należących do tego przepływu, które pominęliśmy.

Różnicą pomiędzy tradycyjnymi warsztatami pisania historyjek a tworzeniem mapy historyjek jest to, że podczas warsztatów skupiamy się bardziej na tworzeniu historyjek, a mniej na przypisywaniu im priorytetów (pionowej pozycji historyjek nadających się do implementacji w mapie historyjek). Zatem tworzenie mapy historyjek możemy uznać za uzupełnienie warsztatów — wizualną technikę nadawania priorytetów historyjkom. Mapy historyjek, w odróżnieniu od tradycyjnej liniowej (jednowymiarowej) reprezentacji, stanowią dwuwymiarowy widok rejestru produktu.

## Zakończenie

W tym rozdziale omówiłem różnicę pomiędzy sposobem traktowania wymagań przez Scrum i tradycyjne, sekwencyjne metody produkcji. W przypadku scrumowego wysiłku deweloperskiego tworzymy wskaźniki do wymagań nazywane elementami rejestru produktu. Te elementy są często wyrażone w formie historyjek użytkownika i przechodzą przez cały proces scrumowy ze szczególną uwagą położoną na prowadzenie dyskusji jako środka pozwalającego odkryć szczegóły wymagań. Stosujemy również strategię stopniowego uszczegółowiania dużych, bardziej ogólnych historyjek, tworząc historyjki mniejsze, bogatsze w detale. Robimy to w myśl idei działania w samą porę.

Następnie wprowadziłem oficjalnie pojęcie historyjek użytkownika, opisując je w kontekście trzech słów kluczowych: karty, rozmowy i potwierdzenia. Dalej opisałem, w jaki sposób historyjki użytkownika mogą posłużyć do przedstawienia wartości biznesowej na różnych poziomach abstrakcji. Wyjaśniłem, w jaki sposób kryteria INVEST pomagają w stwierdzeniu, czy nasze historyjki są dobrze zdefiniowane. Później przedstawiłem sposoby radzenia sobie z wymaganiami niefunkcjonalnymi, a także aktywnościami mającymi na celu pozyskanie wiedzy. Pod koniec rozdziału omówiłem sposoby generowania historyjek użytkownika, skupiając się na dwóch technikach: warsztatach pisania historyjek i tworzeniu mapy historyjek. Kolejny rozdział poświęcony będzie rejestrowi produktu.



# Rozdział 6

## REJESTR PRODUKTU

---

W tym rozdziale omówię istotną rolę, jaką rejestr produktu odgrywa w projekcie deweloperskim koordynowanym przez Scrum. Zacznę od opisania najczęściej spotykanych w rejestrze elementów. Następnie wskażę cztery cechy dobrego rejestru produktu, a także znaczenie pielęgnacji rejestru dla zachowania tych cech. Później wyjaśnię, dlaczego rejestr produktu jest kluczem do szybkiego i elastycznego zarządzania przepływem pracy zarówno na poziomie wersji dystrybucyjnej, jak i na poziomie sprintu. Rozdział zakończę omówieniem tego, ile i jakie rejesty produkту powinniśmy posiadać.

### Wprowadzenie

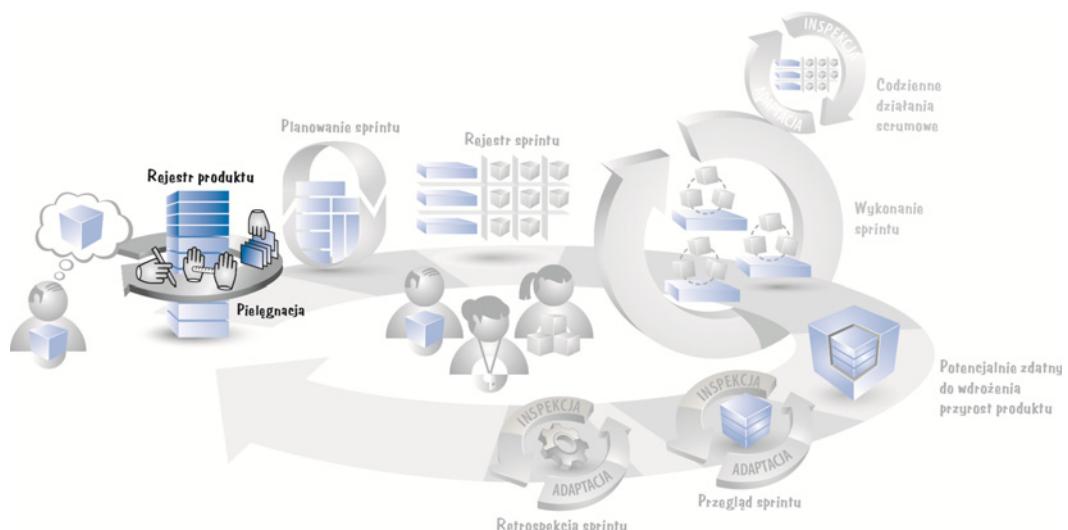
Rejestr produktu to nic innego jak ułożona zgodnie z priorytetami lista funkcjonalności oczekiwanej w produkcie. Jest to centralny ośrodek dostarczający wszystkim wiedzy i świadomości odnośnie do tego, co należy zbudować i w jakiej kolejności. Rejestr produktu jest niezwykle widocznym artefaktem tkwiącym w samym środku środowiska Scrum i dostępnym dla wszystkich uczestników projektu (patrz rysunek 6.1).

Rejestr produktu jest niezbędny, jeśli budujemy, rozbudowujemy bądź utrzymujemy produkt lub system.

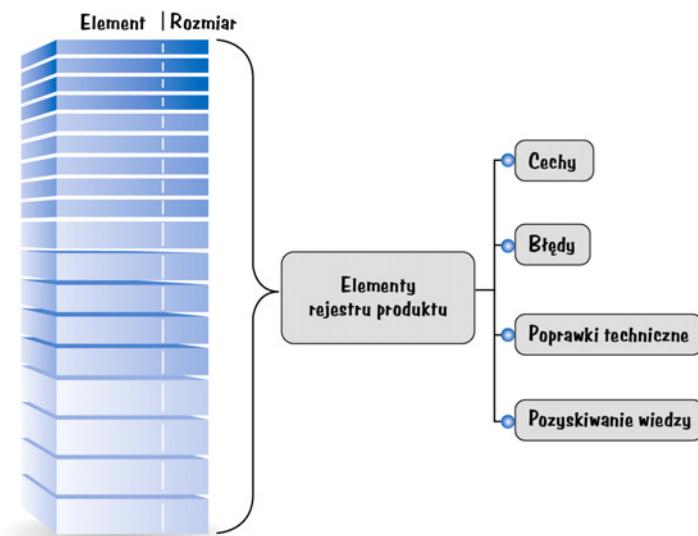
### Elementy rejestrów produktu

Rejestr produktu składa się z elementów rejestrów, które nazywam elementami rejestrów produktu lub w skrócie elementami (patrz rysunek 6.2).

Większość elementów rejestrów produktu to cechy funkcjonalności, które kiedyś będą miały namacalną wartość dla użytkownika lub klienta. Cechy są zazwyczaj wyrażane poprzez historyjki użytkownika (choć sam Scrum nie definiuje konkretnej formy elementów). Przykładami cech produktów mogą być rzeczy zupełnie nowe (ekran logowania dla nowej witryny internetowej) lub zmiany cech już istniejących (bardziej przyjazny dla użytkownika ekran logowania do witryny).



**RYSUNEK 6.1.** Rejestr produktu tkwiący w samym centrum środowiska Scrum



**RYSUNEK 6.2.** Elementy rejestru produktu

Inne elementy rejestru produktu to błędy wymagające naprawy, poprawki techniczne, prace mające na celu pozyskanie wiedzy, a także inne zadania, które właściciel produktu uważa za wartościowe. Przykłady różnych elementów rejestru produktu znajdziesz w tabeli 6.1.

**TABELA 6.1.** Przykładowe elementy rejestru produktu

Typ elementu	Przykład
Cecha	Jako przedstawiciel biura obsługi klienta chcę stworzyć wpis dla problemu zgłoszonego przez klienta, dzięki czemu będę mógł zarejestrować żądanie pomocy przez klienta i zarządzać nim.
Zmiana	Jako przedstawiciel biura obsługi klienta chcę, aby wyniki wyszukiwania były domyślnie sortowane według nazwiska, a nie numeru zgłoszenia, dzięki temu łatwiej będzie znaleźć żądanie pomocy.
Błąd	Naprawić błąd #256 z bazy danych błędów, dzięki czemu program nie będzie „wylatywać”, kiedy klient użyje znaków specjalnych w wyszukiwarce.
Poprawka techniczna	Przejść na najnowszą wersję bazy danych Oracle.
Pozyskiwanie wiedzy	Stworzyć prototypy lub udowodnić koncepcje dwóch architektur, a następnie przeprowadzić trzy testy, aby przekonać się, która z nich jest lepszym rozwiązańiem dla naszego produktu.

## Cechy dobrego rejestru produktu

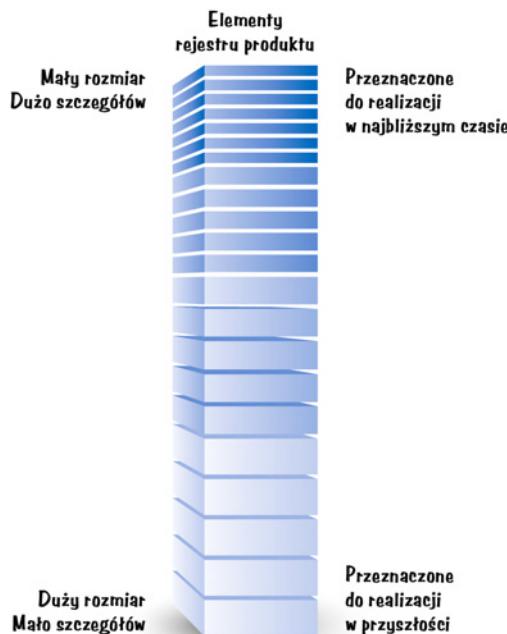
Dobre rejestyry produktu mają podobne charakterystyki. Roman Pichler [Pichler, 2010] i Mike Cohn stworzyli skrót **DEEP** będący podsumowaniem ważnych cech dobrych rejestrów produktu. Są to kolejno: odpowiedni stopień uszczegółowienia (ang. *Detailed appropriately*), emergencja (ang. *Emergent*), przypisanie ocen (ang. *Estimated*) i przypisanie priorytetów (ang. *Prioritized*). Tak jak kryteria INVEST służą do oceny jakości historyjki użytkownika, tak kryteria DEEP nadają się do wskazania, czy rejestr produktu posiada dobrą strukturę.

## Odpowiedni stopień uszczegółowienia

Nie wszystkie elementy w rejestrze produktu będą na tym samym poziomie uszczegółowienia w danej chwili (patrz rysunek 6.3).

Elementy rejestrów produktu, nad którymi planujemy pracować wkrótce, powinny znajdować się na szczytach rejestrów, mieć mały rozmiar i być bogate w detale. Dzięki temu będzie można je wziąć do jednego z nadchodzących sprintów. Elementy, nad którymi nie będziemy pracować jeszcze przez dłuższy czas, powinny być bliżej końca rejestrów, mieć większy rozmiar i mniej szczegółów.

Zbliżając się do pracy nad większymi elementami, takimi jak eposy, będziemy dzielić je na zbiorę mniejszych historyjek nadających się do sprintu. Takie działanie powinno odbywać się w samą porę. Jeżeli zbyt wcześnie dokonamy uszczegółowienia, być może poświęcimy sporą ilość czasu na wypracowanie szczegółów, a mimo to nigdy nie zaimplementujemy całej historii. Jeżeli będziemy czekać zbyt długo, wyhamujemy strumień elementów wchodzących do sprintów i tym samym spowolnimy pracę zespołu. Musimy znaleźć równowagę pomiędzy pracą w samą porę i pracą w stopniu wystarczającym.



**RYSUNEK 6.3.** Elementy rejestrów produktu mają różne rozmiary

## Emergenčia

Rejestr produktu nie jest kompletny lub zamrożony tak długo, jak długo trwa proces deweloperski lub utrzymanie produktu. Podlega on ciągłym aktualizacjom w oparciu o strumień napływających nieustannie cennych ekonomicznie informacji. Na przykład klienci mogą zmieniać swoje zdanie odnośnie do własnych potrzeb, konkurencja może podejmować odważne, nieprzewidywalne posunięcia, na horyzoncie mogą pojawiać się niespodziewane trudności techniczne. Rejestr produktu jest zaprojektowany w taki sposób, aby adaptować się do tych zdarzeń.

W rezultacie rejestr produktu podlega nieustajcej emergencji. Właściciel produktu, biorąc po uwagę nowe informacje, musi dokonywać zmian w rejestrze produktu i na nowo przypisywać priorytety w miarę pojawiania się nowych elementów lub redefiniowania elementów już istniejących.

## Przypisanie ocen

Każdy element rejestrów produktu ma przypisaną do siebie ocenę rozmiaru odpowiadającą wysiłkowi potrzebnemu do jego realizacji (patrz rysunek 6.4).

Właściciel produktu używa tych ocen jako jednego z czynników pomagających mu w ustaleniu priorytetów elementów (i co za tym idzie, ich pozycji) w rejestrze produktu. Duże elementy o wysokim priorytecie (umieszczone blisko szczytu rejestrów) są informacją dla właściciela produktu, że trzeba je doprecyzować, zanim będą mogły osiągnąć poziom nadającym się do sprintu.



**RYSUNEK 6.4.** Elementy rejestru produktu mają przypisane oceny

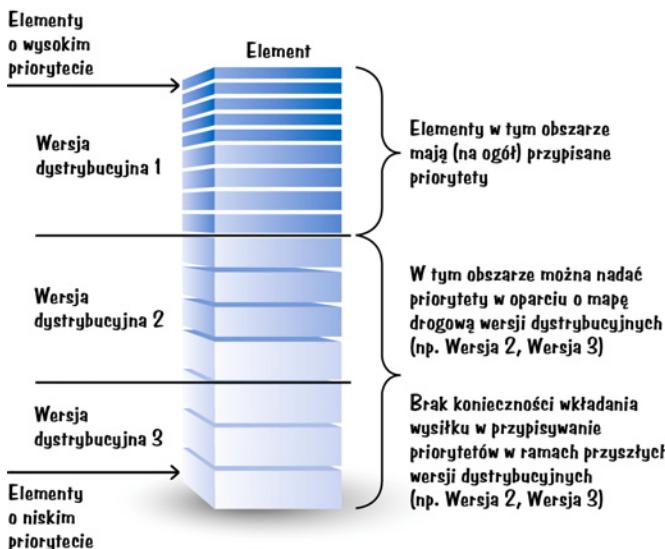
Większość elementów rejestru produktu jest oceniana w punktach historyjkowych lub idealnych dniach (będę o tym pisał szerzej w rozdziale 7.). Oceny muszą być w miarę dokładne, ale bez zbytniej przesady. Elementy znajdujące się u szczytu rejestru będą mniejsze i bogatsze w szczegóły, a zatem będą miały również dokładniejsze oceny. Przypisanie numerycznych ocen do dużych elementów (takich jak eposy) u dołu rejestru może okazać się niemożliwe — w takich przypadkach zespoły mogą zdecydować się w ogóle nie przypisywać im ocen lub użyć miar stosowanych dla ubrań (L, XL, XXL itd.). Wartości numeryczne będą przypisywane w miarę rozbijania większych elementów na zbiory elementów mniejszych.

## Przypisanie priorytetów

Chociaż rejestr produktu jest listą elementów uporządkowaną według priorytetów, rzadko zdarza się, aby *wszystkie* zawarte w nim elementy miały przypisany priorytet (patrz rysunek 6.5).

Użytecznym rozwiązaniem jest przypisanie priorytetów elementom w krótkim horyzoncie czasowym — przeznaczonym do realizacji w ciągu kilku nadchodzących sprintów. Być może powinniśmy przypisać priorytety do miejsca w rejestrze, które uznajemy za możliwe do zrealizowania w pierwszej wersji dystrybucyjnej. Pójście dalej z bardzo przybliżonymi priorytetami nie jest warte poświęcania naszego czasu.

Wystarczy na przykład, że zadeklarujemy (w oparciu o naszą mapę drogową) dany element jako przeznaczony na wersję dystrybucyjną numer 2 lub 3. Będąc jednak we wczesnej fazie wersji numer 1, poświęcenie czasu na ustalanie priorytetów elementów, nad którymi być może będziemy pracować kiedyś przy wersji 2 lub 3, nie jest dobrą inwestycją. Może się okazać, że nigdy faktycznie nie zrealizujemy wersji numer 2 i 3 lub nasze wyobrażenie tych wersji ulegnie zupełnej zmianie podczas prac nad wersją numer 1. Widać więc, że czas poświęcony na ustalanie priorytetów dla tak odległej przyszłości z dużą dozą pewności będzie marnotrawstwem.



**RYSUNEK 6.5.** Elementy rejestru produktu są ułożone według priorytetów

Oczywiście w miarę pojawiania się nowych elementów w rejestrze produktu właściciel produktu będzie odpowiedzialny za umieszczenie ich w odpowiednim miejscu, biorąc pod uwagę elementy już istniejące.

## Pielęgnacja

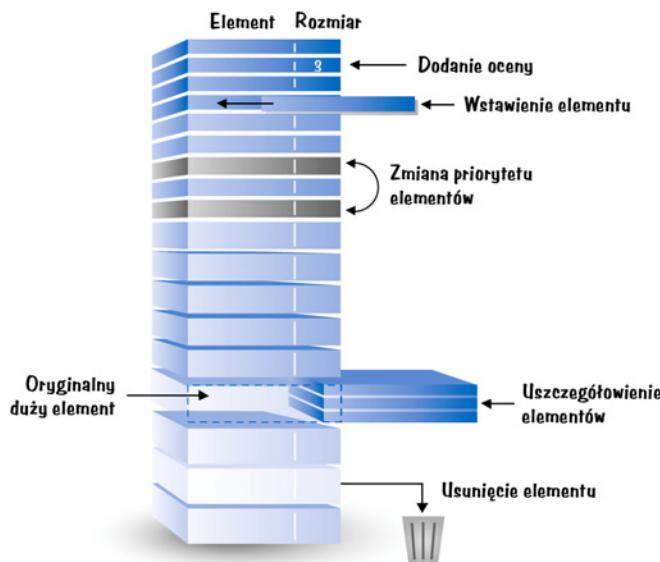
Jeśli chcemy posiadać dobry, zgodny z kryteriami DEEP rejestr produktu, musimy nim aktywnie zarządzać i administrować — zajęcia te nazywane są potocznie pielęgnowaniem rejestrów produktu.

### Czym jest pielęgnacja?

Pielęgnacja odnosi się do trzech głównych aktywności: tworzenia i ulepszenia (dodawania szczegółów), oceniania i nadawania priorytetów elementom rejestrów produktu.

Rysunek 6.6 przedstawia niektóre z zadań pielęgnacyjnych oraz ich wpływ na strukturę rejestrów produktu.

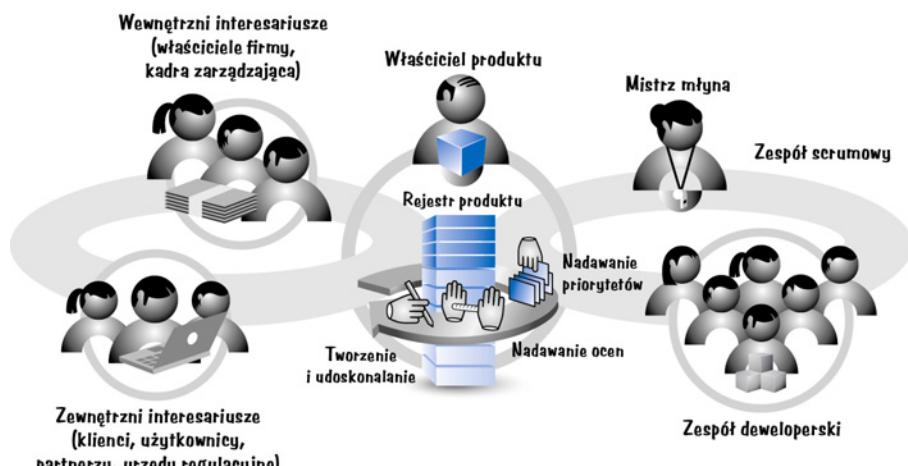
Wszystkie elementy rejestrów muszą zostać wcześniej czy później ocenione, aby można było określić ich miejsce w rejestrze i zdecydować, czy uzasadnione jest dalsze rozbijanie na elementy mniejsze. Pojawianie się istotnych informacji powoduje wstawienie w odpowiednie miejsca rejestrów nowych elementów. Jeśli następuje zmiana priorytetów, potrzebna jest zmiana porządku w rejestrze, a w miarę zbliżania się do implementacji dużego elementu chcemy rozbić go na zbiór mniejszych elementów. Możemy również zdecydować, że dany element rejestrów nie jest potrzebny, i usunąć go.



**RYSUNEK 6.6.** Pielęgnacja powoduje zmiany kształtu rejestru produktu

## Kto pielęgnuje rejestr?

Pielęgnacja rejestru produktu jest nieustającym zespołowym wysiłkiem, któremu przewodzi właściciel produktu, z istotnym udziałem wewnętrznych i zewnętrznych interesariuszy, a także mistrza młyna i zespołu deweloperskiego (patrz rysunek 6.7).



**RYSUNEK 6.7.** Pielęgnacja jest wysiłkiem zbiorowym

W ostatecznym rozrachunku jest tylko jedna osoba o decydującym głosie w zakresie pielęgnacji — właściciel produktu. Niemniej jednak dobrzy właściciele produktu rozumieją, że zespołowe działania w zakresie pielęgnacji rozwijają dialog pomiędzy wszystkimi uczestnikami, a także służą

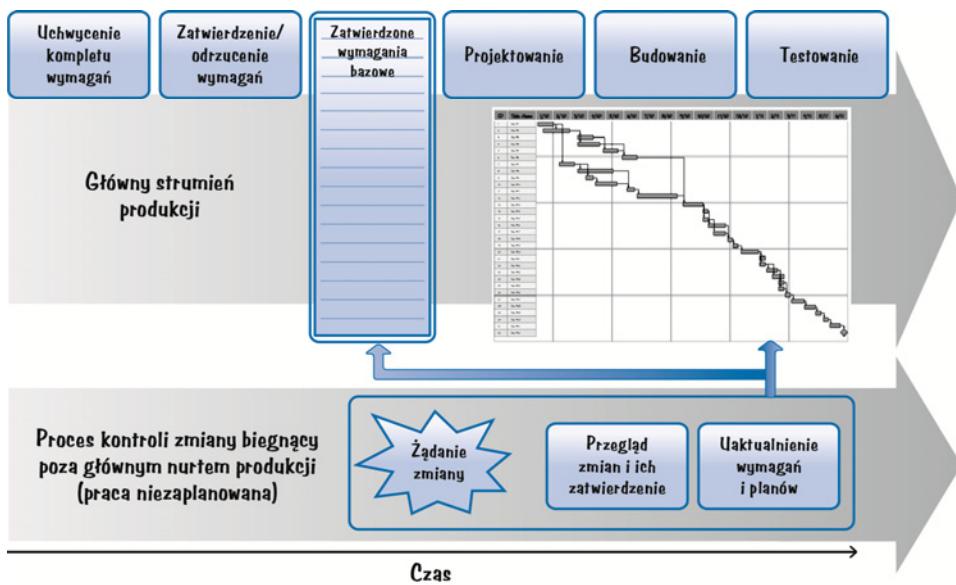
rozwojowi kolektywnej inteligencji i dalekowzroczności grupy ludzi o różnych osobowościach, a co za tym idzie — ujawniają istotne informacje, które w przeciwnym wypadku mogłyby być z łatwością pominięte. Dobrzy właściciele produktu wiedzą również, że poprzez angażowanie różnych członków zespołu w pielęgnację rejestru zapewniają lepsze, obopólne rozumienie jego zawartości, co z kolei redukuje problemy komunikacyjne i działania w błędnym przekonaniu o własnej słuszności. Tego typu kolektywny wysiłek pomaga również w pokonaniu odwiecznej bariery pomiędzy ludźmi z działów biznesowych i inżynierami zajmującymi się produkcją.

Interesariusze, w zależności od specyfiki organizacji i typu projektu, powinni zarezerwować wystarczającą ilość czasu na pielęgnację rejestru. Przyjmuje się ogólną zasadę, według której zespół deweloperski powinien poświęcić 10% swojego czasu w każdym sprincie na asystowanie właścielowi produktu w działańach związanych z pielęgnacją. W tym czasie członkowie zespołu pomagają w tworzeniu nowych elementów rejestru, przeglądaniu istniejących, a także dzieleniu elementów większych na mniejsze. Zespół ocenia również rozmiar elementów i pomaga właścielowi produktu w nadaniu im odpowiednich priorytetów, biorąc pod uwagę ograniczenia techniczne i dostępne zasoby.

## Kiedy odbywa się pielęgnacja?

Środowisko Scrum zaznacza jedynie, że pielęgnacja powinna mieć miejsce. Nie specyfikuje, *kiedy* powinna nastąpić. Zatem kiedy faktycznie pielęgnujemy rejestr produktu?

W sekwencyjnym procesie produkcyjnym próbujemy uchwycić kompletny i szczegółowy opis wymagań na samym początku prac, stąd po zatwierdzeniu wymagań pielęgnacja jest praktycznie niepotrzebna. W wielu organizacjach te wymagania bazowe mogą być zmieniane wyłącznie poprzez specjalnie do tego celu przeznaczony proces kontroli zmian, który biegnie niezależnie od głównego procesu wytwórczego (patrz rysunek 6.8).

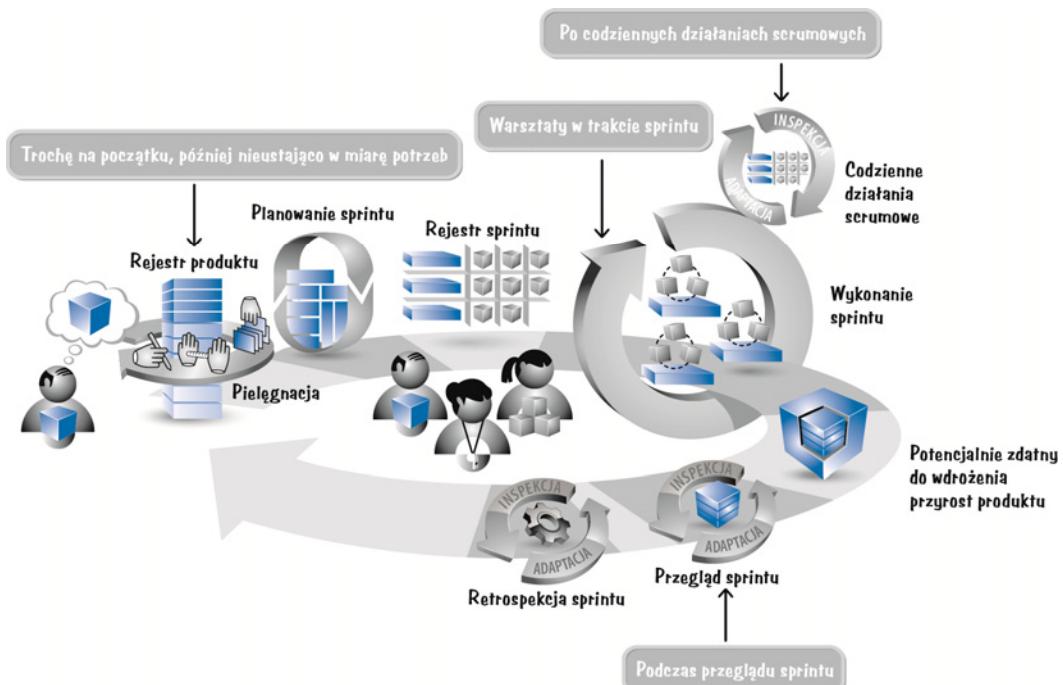


**RYSUNEK 6.8.** Pielęgnacja w procesach sekwencyjnych, bieżąca w oderwaniu od głównego nurtu

Stąd pielęgnacja podczas sekwencyjnego procesu produkcyjnego jest zjawiskiem wyjątkowym, niezaplanowanym, biegnącym poza głównym nurtem produkcyjnym. Jest stosowana wyłącznie, jeśli nie ma innego wyjścia i powoduje przerwanie płynnego procesu dostarczania wartości biznesowej.

W Scrumie zakładamy zmienność otaczającego nas środowiska i dlatego musimy być przygotowani na nieustanną inspekcję i adaptację. Oczekujemy, że rejestr nie pozostanie w stanie zablokowanym od samego początku, z jedyną możliwością jego zmiany poprzez jakiś drugoplanowy proces przeznaczony do obsługi wyjątkowych, niespodziewanych zdarzeń, ale będzie nieustannie ewoluował. W związku z tym musimy zrobić wszystko, aby pielęgnacja stała się kluczową, nieodłączną częścią naszego sposobu zarządzania pracą.

Rysunek 6.9 ilustruje różne momenty, w których można dokonywać pielęgnacji.



**RYSUNEK 6.9.** Kiedy ma miejsce pielęgnacja

Początkowa pielęgnacja jest częścią aktywności planowania wersji dystrybucyjnej (więcej szczegółów na ten temat znajdziesz w rozdziale 18.). W trakcie produkcji właściciel produktu spotyka się z interesariuszami z częstotliwością, która pozwala na nieprzerwaną pielęgnację.

W przypadku zespołu deweloperskiego właściciel produktu może wyznaczyć spotkanie mające miejsce raz w tygodniu lub raz w ciągu wykonania sprintu, w trakcie którego przeprowadzane są warsztaty pielęgnacji rejestru. Dzięki temu zapewnione jest regularne przeprowadzanie pielęgnacji, a zespół może upomnieć się o ten czas w trakcie planowania sprintu. Redukcji ulega również maratonistyczne planowanie spotkań z marszu (na przykład sprawdzania, kiedy ludzie mają wolny czas, szukania wolnej sali konferencyjnej itd.).

Czasami zespoły zamiast blokować z góry określony przedział czasu, wolą rozłożyć pielęgnację na okres całego sprintu. Poświęcają odrobinę czasu po codziennych działaniach scrumowych, aby przeprowadzić pielęgnację w sposób przyrostowy. Pielęgnacja nie wymaga obecności wszystkich członków zespołu. Na przykład właściciel produktu może poprosić o pomoc w podzieleniu dużej historyjki. Członkowie zespołu posiadający odpowiednią wiedzę i zainteresowani tematem zostają po spotkaniu i udzielają niezbędnego wsparcia. Innym razem asystować mogą zupełnie inni członkowie zespołu.

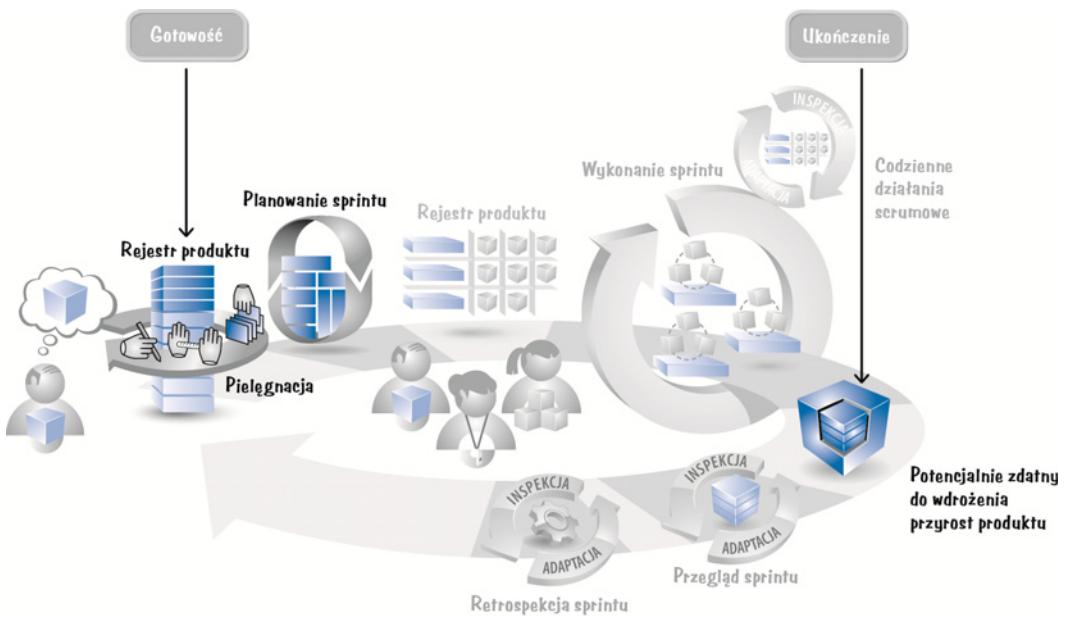
Nawet jeśli zespół ma zaplanowane warsztaty lub poświęca trochę czasu każdego dnia, aby spojrzeć na rejestr, większość zespołów uznaje za naturalne przeprowadzenie pielęgnacji w ramach przeglądu sprintu. W miarę uzyskiwania coraz lepszego rozumienia aktualnego stanu produktu i jego przyszłości przez wszystkie zaangażowane osoby tworzone są nowe elementy rejestru, a istniejącym zmienia się priorytet lub się je usuwa, gdy przestają być potrzebne.

To, kiedy pielęgnacja ma miejsce, nie jest tak ważne, jak upewnienie się, iż stanowi integralną część scrumowego procesu deweloperskiego i w ten sposób zapewnia sprawne i dostosowane do potrzeb dostarczanie wartości biznesowych.

## Definicja gotowości

Celem pielęgnacji rejestru produktu jest zapewnienie na jego szczycie elementów gotowych do umieszczenia w sprincie, tak aby zespół mógł z dużą dozą pewności zobowiązać się i zrealizować je w ciągu sprintu.

Niektóre zespoły scrumowe formalizują ten cel, ustalając **definicję gotowości**. Definicję gotowości i definicję ukończenia (patrz rozdział 4.) należy traktować jako dwa stany elementów rejestru produktu w cyklu sprintu (patrz rysunek 6.10).



**RYSUNEK 6.10.** Definicja gotowości

Obie definicje — gotowości i ukończenia — są listami kontrolnymi pracy, które muszą zostać skompletowane, zanim będzie można uznać, że element rejestru produktu znajduje się w danym stanie. Tabela 6.2 pokazuje przykład listy kontrolnej definicji gotowości dla elementu rejestru produktu.

**TABELA 6.2.** Przykład listy kontrolnej definicji gotowości

Definicja gotowości	
<input type="checkbox"/>	Wartość biznesowa jest wyartykułowana w jasny sposób.
<input type="checkbox"/>	Zespół deweloperski rozumie szczegółowo do tego stopnia, że jest w stanie podjąć świadomą decyzję o tym, czy da radę ukończyć ten element rejestru produktu.
<input type="checkbox"/>	Wszystkie zależności zostały zidentyfikowane i wiadomo, że nie ma zależności zewnętrznych, które uniemożliwiłyby ukończenie elementu rejestru produktu.
<input type="checkbox"/>	Zespół ma odpowiedni skład umożliwiający ukończenie elementu rejestru produktu.
<input type="checkbox"/>	Element ma przypisaną ocenę i jest ona na tyle mała, że pozwala na swobodne ukończenie go w pojedynczym sprintie.
<input type="checkbox"/>	Kryteria akceptacji są zrozumiałe i mogą zostać przetestowane.
<input type="checkbox"/>	Kryteria wydajnościowe, jeśli takie istnieją, są zdefiniowane i mogą zostać przetestowane.
<input type="checkbox"/>	Zespół scrumowy wie, w jaki sposób zademonstrować element rejestru produktu podczas przeglądu sprintu.

Dobrze skonstruowana definicja gotowości zdecydowanie poprawia szanse zespołu scrumowego na pomyślne osiągnięcie założonego celu sprintu.

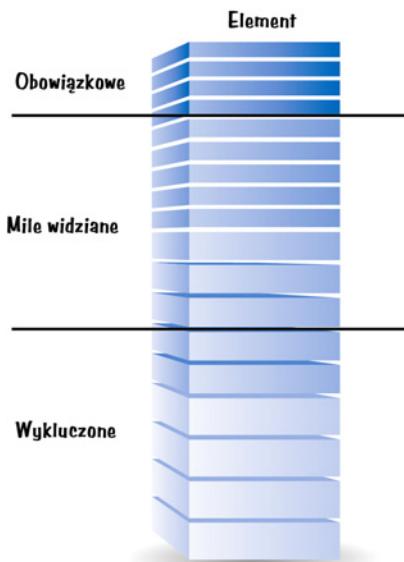
## Zarządzanie przepływem

Rejestr produktu jest kluczowym narzędziem umożliwiającym zespołowi scrumowemu osiągnięcie szybkiego i elastycznego potoku dostarczania wartości biznesowej w obliczu niepewności. Niepewności nie da się wyeliminować z procesu produkcyjnego. Musimy założyć, że będziemy nieustannie „bombardowani” ważnymi ekonomicznie informacjami, i tak zorganizować pracę i zarządzać nią (czyli rejestrem produktu), aby móc przetwarzać te informacje na bieżąco w sposób wydajny pod względem kosztów i zachowania ciągłości pracy. Przeanalizujmy rolę rejestru produktu we wsparciu dobrego przepływu wersji dystrybucyjnych i sprintów.

## Zarządzanie przepływem wersji dystrybucyjnych

Rejestr produktu musi być pielęgnowany w sposób pozwalający na nieustanne planowanie wersji dystrybucyjnych (cech trafiających do danej wersji). Rysunek 6.5 pokazuje, że wersja dystrybucyjna może zostać przedstawiona w formie linii dzielącej rejestr produktu. Wszystkie elementy rejestru ponad tą linią są przewidziane do umieszczenia w danej wersji, a elementy poniżej nie.

Ja odkryłem, że użytecznym rozwiązaniem jest podzielenie rejestru produktu przy użyciu dwóch linii na każdą wersję, tak jak pokazuje to rysunek 6.11.



**RYSUNEK 6.11.** Widok rejestru produktu z punktu widzenia wersji dystrybucyjnej

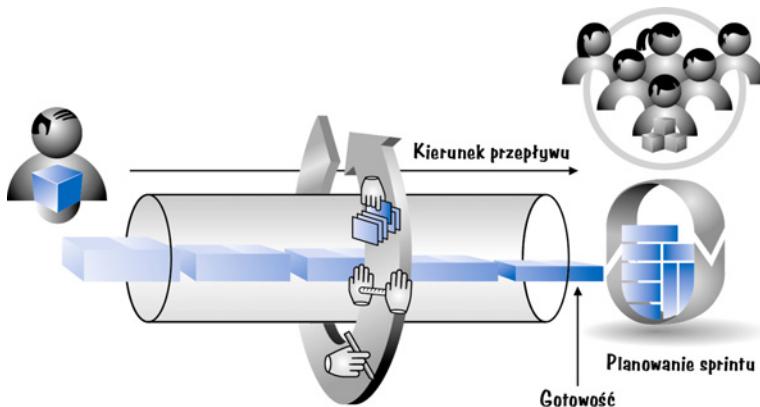
Te dwie linie dzielą rejestr na trzy obszary: *obowiązkowe* (ang. *must have*), *mile widziane* (ang. *nice to have*) i *wykluczone* (ang. *won't have*). **Cechy obowiązkowe** reprezentują elementy, które musimy mieć w nadchodzącej wersji — w przeciwnym razie nie będziemy posiadać dobrej wersji z punktu widzenia użytkownika. **Cechy mile widziane** to elementy, które przeznaczyliśmy na następną wersję i chcielibyśmy je do niej dołączyć. Jeżeli jednak zabraknie nam czasu lub zasobów, możemy porzucić cechy mile widziane i nadal będziemy w stanie wypuścić cenny produkt. **Cechy wykluczone** to elementy, których postanowiliśmy nie dodawać do bieżącej wersji dystrybucyjnej. Druga linia — oddzielająca elementy wykluczone od mile widzianych — odpowiada linii wersji dystrybucyjnej 1 z rysunku 6.5.

Utrzymywanie rejestru produktu w takim układzie pomoże nam w lepszym planowaniu wersji dystrybucyjnych, o czym będę pisał w rozdziale 18.

## Zarządzanie przepływem sprintu

Pielegnacja rejestru produktu ma kluczowe znaczenie dla sprawnego zaplanowania sprintu, a co za tym idzie dla potoku cech trafiających do tego sprintu. Jeżeli rejestr produktu został odpowiednio uszczegółowiony, elementy na jego szczytce są przejrzyste i wiadomo, w jaki sposób można je przetestować.

Podczas pielegnacji mającej na celu uzyskanie dobrego przepływu sprintu pomocne jest traktowanie rejestru produktu jako strumienia wymagań, który wpływa do sprintów celem zaprojektowania, zbudowania i przetestowania przez zespół (patrz rysunek 6.12).



**RYSUNEK 6.12.** Rejestr produktu jako strumień wymagań

Widzimy na tym rysunku, że większe, mniej rozumiane wymagania zostają wstawione do strumienia po lewej stronie. W miarę poruszania się w nim — w kierunku wylotu — czyli zbliżania się do czasu, kiedy trafią do realizacji, są stopniowo uszczegóławiane poprzez działania pielęgnacyjne. Z prawej strony, przy wylotie, znajduje się zespół deweloperski. W chwili, kiedy element wydostanie się z potoku, musi być gotowy — wyposażony w dostateczną liczbę szczegółów, co pozwoli na zrozumienie go przez zespół i dostarczenie w komfortowy sposób podczas sprintu.

Jeżeli istnieje dysproporcja pomiędzy ilością elementów wpływającymi i wypływającymi, pojawia się problem. Gdy wypływ wypielęgnowanych, uszczegółowionych, gotowych do implementacji elementów jest zbyt wolny, wkrótce strumień wyschnie i zespół nie będzie w stanie zaplanować oraz wykonać kolejnego sprintu (poważne naruszenie lub marnotrawstwo przepływu w Scrumie). Z drugiej strony, wpuszczenie do strumienia zbyt dużej ilości elementów do podzielenia utworzy duży inwentarz szczegółowych wymagań, który być może trzeba będzie ponownie przeanalizować lub porzucić, kiedy dowiemy się czegoś więcej o celu naszej działalności (istotne źródło marnotrawstwa). Dlatego idealnym rozwiązaniem jest posiadanie inwentarza wystarczającego do zachowania równego przepływu, ale na tyle małego, aby nie generować marnotrawstwa.

Jednym z rozwiązań stosowanych przez zespoły scrumowe jest posiadanie odpowiedniego inwentarza wypielęgnowanych, gotowych do implementacji elementów w rejestrze produktu. Heurystyką, która wydaje się działać dla wielu zespołów, jest posiadanie w rejestrze gotowych do wykonania elementów o wartości mniej więcej dwóch do trzech sprintów. Zatem dla przykładu: jeśli w normalnych warunkach zespół potrafi wykonać 5 elementów rejestru w ciągu sprintu, pielęgnacja sprowadza się do posiadania przez cały czas od 10 do 15 gotowych do implementacji elementów<sup>1</sup> w rejestrze produktu. Ten dodatkowy inwentarz daje gwarancję, że potok nie wyschnie, a także zapewnia zespołowi pewną swobodę w wybieraniu elementów w innym porządku, jeśli zachodzi taka potrzeba (na przykład ze względu na ilość deweloperów dostępnych w sprintie lub też inne ograniczenia sprintowe). Bardziej szczegółowo omówienie tego tematu znajdziesz w rozdziale 19.

<sup>1</sup> Przy założeniu, że wszystkie elementy mają mniej więcej ten sam rozmiar wyrażony w punktach historyjkowych lub idealnych dniach — *przyp. tłum.*

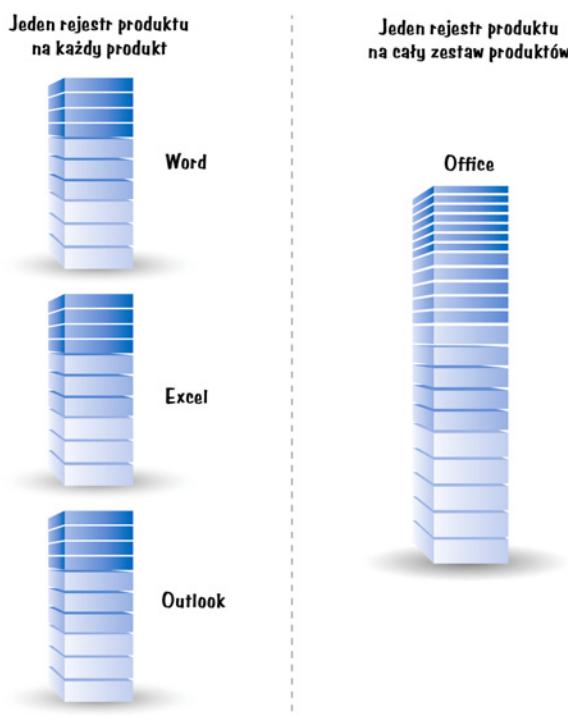
## Jakie rejestyry produktu i w jakiej liczbie?

Podejmując decyzję o tym, jakie rejestyry produktu utworzyć i w jakiej liczbie, zaczynam od prostej zasady: jeden produkt, jeden rejestr produktu, co oznacza, że każdy produkt powinien posiadać swój pojedynczy rejestr produktu pozwalający na przeprowadzenie wszelkich możliwych działań związanych z jego opisywaniem i nadawaniem priorytetów pracy przeznaczonej do wykonania.

Istnieją jednak przypadki, kiedy chcąc uzyskać praktyczną, nadającą się do pracy strukturę rejestrów, musimy podejść ostrożnie do wykonania tej reguły. Czasami na przykład trudno jasno określić, czym właściwie jest produkt. Niektóre z produktów są bardzo duże, a pracujące nad nimi zespoły nie mogą działać w sposób wymienny. W innym przypadku możemy mieć wiele produktów, a tylko jeden zespół. Prześledźmy każdy z tych szczególnych przypadków, aby przekonać się, jak wpływają one na naszą zasadę pojedynczego rejestrów produktu.

### Czym jest produkt?

Problem z zasadą jednego rejestrów produktu na jeden produkt polega na tym, że czasem trudno stwierdzić, co definiuje projekt. Czy Microsoft Word jest produktem? Czy też jednym z interfejsów większego produktu — Microsoft Office? Jeżeli sprzedajemy jedynie zestaw produktów, to czy wystarczy nam pojedynczy rejestr produktu dla całego zestawu, czy też powinniśmy utworzyć po jednym rejestrze dla każdej aplikacji w tym zestawie (patrz rysunek 6.13)?



**RYSUNEK 6.13.** Rejestr produktu jest związany z produktem

Kiedy pracowałem w IBM-ie, odpowiedzią na pytanie „Czym jest produkt” było „Cokolwiek, co ma swój własny unikatowy identyfikator (PID)”. Piękno tej odpowiedzi tkwiło w jej prostocie. IBM sprzedawał produkty z katalogu, zatem jeśli potrafisz wskazać coś poprzez PID, sprzedawcy byli w stanie umieścić to w formularzu zamówienia i stąd można było powiedzieć, że jest to „produkt”. Chociaż odpowiedź w stylu IBM może wydawać się przesadnie prosta, proponuję, abyśmy przyjęli ją za nasz punkt wyjściowy. Produkt jest czymś wartościowym, za co klient gotów jest zapłacić, a co my jesteśmy gotowi zapakować i sprzedać.

Zastosowanie tej reguły przestaje być takie proste, jeśli sformułowaliśmy **zespoły budujące komponenty**, których zadaniem jest, jak sama nazwa wskazuje, budowanie części składowych większego produktu, przeznaczonego do zakupu przez klienta (o komponentach będzie mowa w rozdziale 12.). Oto przykład. Kiedy kupiłem swój odbiornik GPS do nawigacji, nie kupiłem wyłącznie algorytmu wyznaczania trasy. Kupiłem urządzenie przenośne, którego zadaniem jest podanie wizualnych i słuchowych wskazówek prowadzących mnie do celu zakręt po zakręcie. Komponent „wyznaczania trasy” był zaledwie jednym z wielu połączonych razem w celu stworzenia urządzenia, które klient taki jak ja chciałby kupić.

Gdyby twórca GPS-ów utworzył zespół, którego zadaniem byłoby stworzenie komponentu wyznaczania trasy, to czy temu komponentowi powinien towarzyszyć rejestr produktu? Czy powinien raczej istnieć tylko jeden rejestr produktu dla całego urządzenia GPS, z wplecionymi do środka cechami wyznaczania trasy?

To jeszcze nie koniec możliwości — wyobraźmy sobie, że ten sam komponent wyznaczania trasy może zostać umieszczony w różnych urządzeniach GPS, z których każde ma swój własny identyfikator produktu. Czy bylibyśmy skłonni stworzyć oddzielny rejestr produktu dla komponentu, wiedząc, że może on zostać wykorzystany przez różne urządzenia (produkty)?

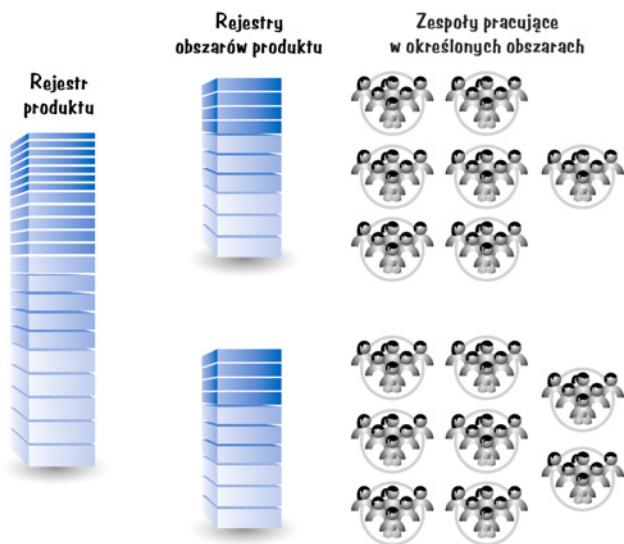
Jak widać, kiedy zaczniemy zadawać tego typu pytania, szybko dojdziemy do bardzo skomplikowanych przypadków. Chcąc wydostać się z tego kryzysu, powinniśmy pamiętać, że naszym celem jest minimalizacja zespołów budujących komponenty i co za tym idzie minimalizacja konieczności posiadania przeznaczonych dla nich rejestrów produktów. Myśl o tym, co tworzysz, jak o rzeczy, która jest pakowana, dostarczana i zwiększa wartość dla klienta, a następnie dostosuj swój rejestr produktu w oparciu o ten tok myślenia.

## Duże produkty — hierarchiczne rejesty produktu

Jeżeli jest to tylko możliwe, preferuję jeden rejestr produktu, nawet w przypadku dużych produktów, takich jak Microsoft Office. Musimy jednak podejść do tego w sposób praktyczny. W przypadku dużych przedsięwzięć, mających na celu stworzenie produktu takiego jak telefon komórkowy, możemy mieć dziesiątki, a nawet setki zespołów, których praca musi zostać zintegrowana, aby utworzyć produkt nadający się do sprzedaży. Umieszczenie elementów rejestrów produktu ze wszystkich zespołów w jednym wspólnie zarządzanym rejestrze jest nieporęczne (i niekonieczne).

Zaczniemy od tego, że nie wszystkie z tych zespołów pracują w powiązanych ze sobą obszarach. Możemy mieć na przykład siedem zespołów zajmujących się odtwarzaczem audio-video dla telefonu i kolejne osiem zespołów pracujących nad przeglądarką. Każda z tych rzeczy przynosi namacalną wartość dla klienta, a praca w każdym z tych obszarów może mieć szczegółowo zorganizowaną i uporządkowaną pod względem priorytetów w pewnym oderwaniu od innych obszarów.

Przy takiej charakterystyce problemu większość zespołów stara się poradzić sobie z dużymi produktami poprzez tworzenie hierarchicznych rejestrów produktów (patrz rysunek 6.14).



**RYSUNEK 6.14.** Hierarchiczne rejesty produkłów

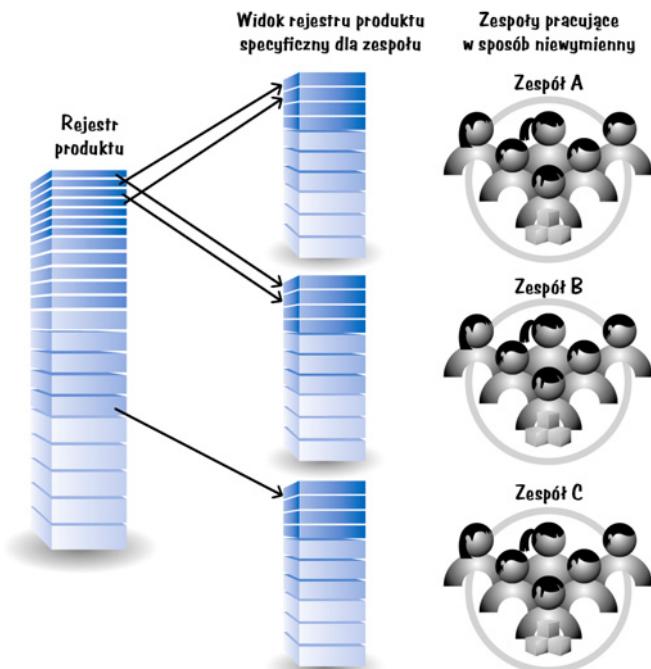
Na szczytce hierarchii nadal mamy jeden rejestr, który opisuje i nadaje priorytety wszystkim cechom produktu o dużej skali (być może eposom). Na tym poziomie jest tylko jeden właściciel produktu (o czym będę pisał w rozdziale 9.). Następnie każda z cech powiązanych ma swój własny rejestr produktu. Zatem odtwarzacz audio-video ma rejestr, który zawiera wszystkie elementy dla siedmiu zespołów pracujących w tym obszarze. Elementy na poziomie specyficznego obszaru będą mniejszej skali (rozmiaru cechy lub historyjki) w porównaniu z elementami w rejestrze całego produktu. W rozdziale 12. będę omawiał koncepcję ciągu wersji dystrybucyjnych bazującego na trzy-poziomowym modelu rejestrów przedsięwzięcia: rejestrze portfela (zawierającego eposy), rejestrze programu (zawierającego cechy) i na końcu rejestrach zespołów (zawierających historyjki użytkownika nadające się do sprintu).

## Wiele zespołów — jeden rejestr produktu

Zasada jednego rejestru na jeden produkt pozwala wszystkim zespołom pracującym nad produktem współdzielić jego rejestr. Przypisanie wszystkich zespołów do pojedynczego rejestrów umożliwia optymalizację kosztów w skali całego produktu. Jest to możliwe, ponieważ umieszczamy wszystkie cechy w jednym rejestrze i sprawiamy, że muszą one rywalizować o priorytet między sobą. W ten sposób zapewniamy identyfikację cech o najwyższym priorytecie w skali całego produktu i jako pierwsze kierujemy je do realizacji.

Jeżeli wszystkie nasze zespoły mogą pracować w sposób wymienny, czyli każdy z nich może podjąć realizację dowolnego elementu w naszym współdzielonym rejestrze, jesteśmy w stanie urzeczywistnić zysk z nadawania priorytetów dostępny dzięki stosowaniu pojedynczego rejestru produktu. A co w sytuacji, kiedy nasze zespoły jednak nie są wymienne? Na przykład zespół pracujący nad silnikiem układu tekstu w Wordzie nie może zostać przypisany do pracy nad silnikiem obliczeń w Excelu? Otrzymujemy rozwiązanie mniej idealne w pewnych sytuacjach — nie każdy zespół może pracować nad dowolnym elementem w rejestrze produktu.

Żeby móc pracować w tych realiach, musimy wiedzieć, które elementy z rejestru produktu może wykonać każdy z zespołów. Koncepcyjnie zatem potrzebujemy rejestrów specyficznych dla zespołów. W praktyce jednak nie tworzymy rejestrów produktów na poziomie zespołów. Zamiast tego mamy do dyspozycji widoki współdzielonego rejestru przeznaczone dla każdego zespołu (patrz rysunek 6.15).



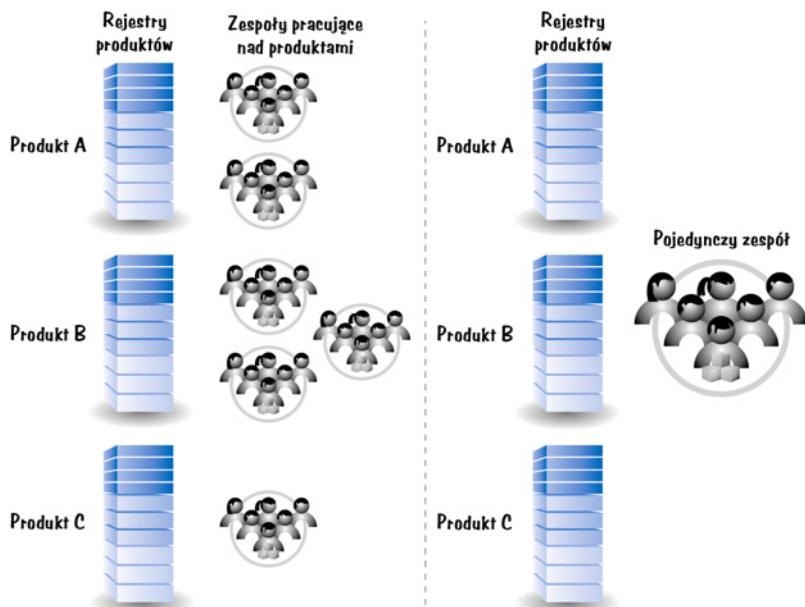
**RYSUNEK 6.15.** Widoki rejestru produktu specyficzne dla zespołów

Zgodnie z rysunkiem 6.15 istnieje tylko jeden rejestr produktu, ale jest on skonstruowany w taki sposób, że zespoły mogą widzieć i wybierać jedynie cechy, które odpowiadają ich umiejętnościom.

Zwrót również uwagi, że na rysunku 6.15 element o najwyższym priorytecie w rejestrze dla zespołu C pochodzi od elementu, który znajduje się bardzo nisko pod względem priorytetów w rejestrze produktu. Gdyby zespoły mogły działać w sposób wymienny, rejestr zespołu C odpowiadałby elementom o znacznie wyższym priorytecie w rejestrze produktu. Ten brak elastyczności jest powodem, dla którego wiele organizacji stara się uczynić kod wspólną własnością i wprowadzić wymienne zespoły — co pozwala skorzystać z zalet posiadania deweloperów mogących pracować nad różnymi obszarami produktu.

## Jeden zespół — wiele produktów

Jeżeli organizacja posiada wiele produktów, będzie miała również wiele rejestrów produktów. Najlepszą metodą obsługi wielu rejestrów jest przypisanie jednego lub kilku zespołów do pracy wyłącznie nad danym rejestrem (patrz lewa strona rysunku 6.16).



**RYSUNEK 6.16.** Scenariusze wielu rejestrów produktów

W niektórych przypadkach jednak znajdzie się zespół, który będzie pracował na kilku rejestrach jednocześnie (patrz prawa strona rysunku 6.16). Naszym celem, o czym będę pisał w rozdziale 11., powinna być minimalizacja pracy, jaką zespoły lub ich członkowie wykonują na różnych projektach w tym samym czasie. Pierwszym i najlepszym rozwiązaniem tego problemu jest skierowanie zespołu do pracy wyłącznie nad jednym projektem. W każdym sprintie zespół wykonuje jedynie elementy z jednego rejestru produktu.

Jeżeli utrudnienia organizacyjne zmuszają nas do posiadania jednego zespołu dla wielu projektów jednocześnie, możemy rozważyć połączenie elementów wszystkich trzech produktów w jeden rejestr produktu. To wymagałoby od właścicieli trzech produktów wzajemnej współpracy w ustalaniu priorytetów pomiędzy ich produktami.

Jeżeli zdecydujemy się pozostać przy trzech oddzielnych rejestrach produktu, w każdym sprintie ktoś (najprawdopodobniej właściciel produktu dla naszego zespołu) będzie musiał stworzyć posortowany według priorytetów zestaw elementów z tych trzech rejestrów (być może w oparciu o z góry założone limity czasu, jakie zespół będzie poświęcał na dany produkt w sprintie) i przedstawić go zespołowi celem analizy i podjęcia zobowiązania na nadchodzący sprint.

## Zakończenie

W tym rozdziale omawiałem kluczową rolę rejestru produktu w osiąganiu szybkiego i dostosowanego do potrzeb przepływu dostarczania wartości w obliczu niepewności. Zwróciłem uwagę na szereg problemów strukturalnych i procesowych związanych z rejestrami produktów. Chodzi między innymi o rodzaje elementów umieszczanych w rejestrach, a także sposoby pielęgnacji rejestrów pozwalające nadać im odpowiednią charakterystykę. Na koniec zająłem się zagadnieniem ilości i typów rejestrów, jakie powinniśmy posiadać. W kolejnym rozdziale skupimy się na sposobie oceniania elementów rejestrów, a także na wykorzystaniu tych ocen do pomiaru prędkości zespołu.

## Rozdział 7

# NADAWANIE OCEN I PRĘDKOŚĆ

---

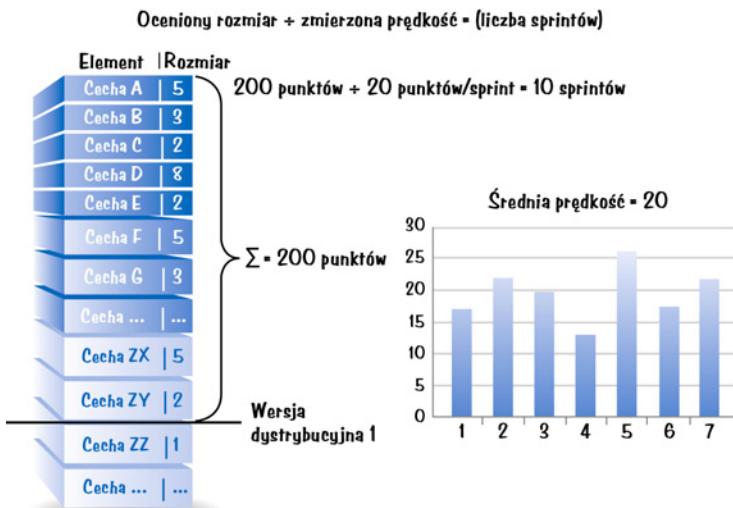
W tym rozdziale opiszę koncepcje nadawania ocen i prędkości. Zaczynę od wskazania istotnych ról odgrywanych przez oceny i prędkość w planowaniu zwinnym. Następnie opiszę różne elementy, które podlegają ocenie, a także wskażę, kiedy i jak należy im nadawać oceny. Większość tego rozdziału poświęcona jest sposobom oceniania elementów rejestru, włączając w to wybór jednostki pomiaru oraz planowanie metodą pokerową. Dalej przejdziemy do omówienia koncepcji prędkości, a także do wskazania kluczowego znaczenia zakresu prędkości podczas planowania. Opiszę, w jaki sposób nowe zespoły mogą przewidywać swoją prędkość w oparciu o dane historyczne. Na koniec zajmiemy się sposobami wpływania na prędkość i metodami jej pomiaru.

## Wprowadzenie

Planując i zarządzając wytwarzaniem produktu, musimy umieć odpowiedzieć na istotne pytania, takie jak „Ile cech uda się wykonać?”, „Kiedy ukończymy pracę?” i „Ile to będzie kosztować?”. Żeby odpowiedzieć na te pytania, używając Scruma, musimy ocenić rozmiar tego, co budujemy, a także prędkość lub tempo, z jakim jesteśmy w stanie realizować pracę. Mając te informacje, możemy ocenić z pewnym prawdopodobieństwem czas trwania produkcji (i związane z tym koszty), dzieląc przyjęty rozmiar zbioru cech przez prędkość zespołu (patrz rysunek 7.1).

Ile czasu zajmie nam stworzenie cech dla wersji dystrybucyjnej numer 1, jeżeli nasz rejestr produktu wygląda jak ten przedstawiony na rysunku 7.1? Aby odpowiedzieć na to pytanie, musimy najpierw wycenić rozmiar wersji dystrybucyjnej numer 1. Możemy to zrobić, dodając oceny rozmiaru poszczególnych elementów rejestru przeznaczonych na tę wersję. (W naszym przykładzie ta suma wynosi 200 punktów).

Kiedy znamy już przybliżony rozmiar wersji dystrybucyjnej, przechodzimy do prędkości zespołu, czyli ilości pracy realizowanej typowo przez zespół w jednym sprincie. Prędkość można wyznaczyć w prosty sposób. Pod koniec każdego sprintu sumujemy ze sobą rozmiary wszystkich elementów, które udało się wykonać w danym sprintie. Jeżeli element nie został wykonany zgodnie z kryteriami ukończenia, nie jest uwzględniany w prędkości. Otrzymana suma ukończonych elementów rejestru produktu jest prędkością zespołu w tym sprintie. Wykres na rysunku 7.1 pokazuje dane dotyczące prędkości zespołu z siedmiu sprintów. Zauważ, że średnia prędkość to 20.



**RYSUNEK 7.1.** Związek pomiędzy rozmiarem, prędkością i czasem trwania

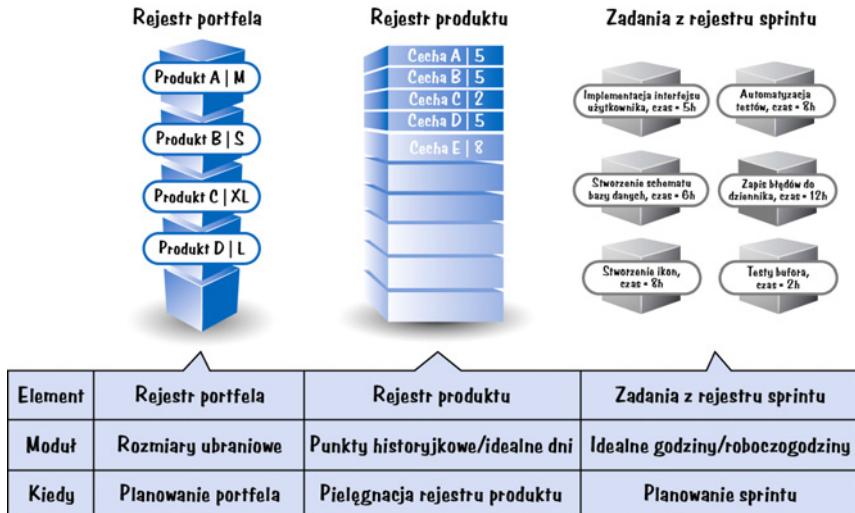
Teraz, kiedy oceniliśmy już rozmiar i zmierzyliśmy prędkość, możemy wyliczyć czas trwania produkcji. W tym celu dzielimy całkowity rozmiar przez prędkość. Jeżeli rozmiar wersji dystrybucyjnej 1 to 200 punktów, a zespół może średnio zrealizować 20 punktów pracy w ciągu sprintu, ukończenie wersji dystrybucyjnej numer 1 powinno zająć zespołowi 10 sprintów (bardziej szczegółowy opis planowania wersji dystrybucyjnej znajduje się w rozdziale 18.). W dalszej części tego rozdziału wyjaśnimy, dlaczego użycie przedziału prędkości daje bardziej dokładne wyniki w porównaniu ze stosowaniem prędkości średniej (tutaj w celach ilustracyjnych będziemy używać wartości średniej).

Chociaż podstawowy związek pomiędzy rozmiarem, prędkością i czasem trwania pozostaje bez zmian, pewne szczegóły mogą podlegać wahaniom w zależności od tego, w którym momencie wysiłku deweloperskiego znajdujesz się, co próbujesz zmierzyć i w jaki sposób zamierzasz wykorzystać dane. Przyjrzymy się bliżej ocenie rozmiaru i prędkości, aby przekonać się, jak te czynniki ulegają zmianie w odpowiedzi na to, co usiłujesz zrobić i kiedy.

## Co i kiedy oceniamy

Na rysunku 7.1 oceny elementów rejestru produktu potrzebne do wyliczenia czasu produkcji wersji dystrybucyjnej wyrażone zostały w punktach historyjkowych. Jednak w trakcie całego cyklu wytwarzania produktu musimy dokonywać ocen na różnych poziomach szczegółowości, w związku z tym będziemy używać różnych jednostek (patrz rysunek 7.2).

Większość organizacji używa ocen do planowania na trzech różnych poziomach szczegółów. Są to poziomy rejestru portfela, rejestru produktu i rejestru sprintu. Przyjrzymy się pokrótce każdemu z nich.



**RYSUNEK 7.2.** Co i kiedy oceniamy

## Ocena elementów z rejestru portfela

Chociaż rejestr portfela nie jest formalnie częścią Scruma, wiele organizacji posiada go i przechowuje w nim ułożoną według priorytetów listę wszystkich produktów (lub projektów), jakie powinny zostać zbudowane. Nadanie dobrych priorytetów w tym rejestrze wymaga znajomości przybliżonego kosztu każdego elementu. Zgodnie z tym, co napisałem w rozdziale 5., zazwyczaj w chwili, kiedy potrzebne będzie wskazanie tych kosztów, nie będziemy posiadać kompletnej listy dostatecznie szczegółowych wymagań i w związku z tym nie będziemy mogli użyć standardowej techniki oceny każdego indywidualnego wymogu, a następnie zsumowania ich razem w celu otrzymania przybliżonego kosztu całkowitego.

Zamiast tego wiele organizacji, przypisując oceny w rejestrze portfela, stosuje rozmiary względne, takie jakie spotkać można na ubraniach (na przykład mały, średni, duży, bardzo duży itd.). Zastosowanie rozmiarów ubraniowych podczas planowania portfela omówię w rozdziale 16.

## Ocena elementów z rejestru produktu

Po zatwierdzeniu produktu lub projektu zaczynamy dodawać więcej szczegółowych informacji do rejestru produktu. Tutaj jednak oceny przypisujemy w inny sposób. Po nadaniu wyższego priorytetu elementom rejestru lub uszczegółowieniu ich w procesie pielęgnacji wiele zespołów woli oocnić je, używając wartości numerycznych — są to najczęściej punkty historyjkowe lub idealne dni. Oba podejścia zostaną opisane w dalszej części rozdziału.

Przypisywanie ocen elementom rejestru produktu jest częścią całego procesu pielęgnacji tego rejestru. Momenty, w których przeprowadzana jest pielęgnacja, ilustruje rysunek 6.9. Zazwyczaj nadawanie ocen elementom rejestru produktu ma miejsce podczas „spotkań estymacyjnych”, z których pierwsze zbiega się w czasie z planowaniem wersji dystrybucyjnej. Właściciel produktu może również zwoływać dodatkowe spotkania estymacyjne w trakcie sprintu, jeżeli zaistnieje potrzeba nadania ocen nowym elementom rejestru.

Nie wszyscy uczestnicy Scruma uważają, że nadawanie ocen elementom rejestru produktu jest niezbędne. Niektóre zespoły na podstawie własnych doświadczeń dochodzą do takiej perfekcji, iż są w stanie tworzyć za każdym razem odpowiednio małe elementy rejestru produktu o bardzo zbliżonych rozmiarach. Tacy uczestnicy Scruma uznają, że przypisywanie ocen takim samym, małym elementom jest marnotrawstwem czasu — w ich wypadku wystarczy policzyć elementy rejestru produktu. Zespoły takie nadal używają koncepcji prędkości, ale jest ona mierzona liczbą elementów rejestru wykonywanych w ciągu sprintu, a nie sumą rozmiarów tych elementów.

Rozumiem argument „braku potrzeby nadawania ocen”, ale mimo to nadal wolę oceniać elementy rejestru produktu z kilku powodów:

- Zgodnie z tym, co napisałem w rozdziale 5., nie wszystkie elementy rejestru produktu będą miały taki sam rozmiar w tym samym czasie. Oprócz kolekcji małych elementów o podobnych rozmiarach w rejestrze będą również elementy większe.
- Nabranie przez zespół umiejętności rozbijania elementów na mniejsze o zbliżonym rozmiarze wymaga czasu.
- Zespoły mogą być zmuszone do dzielenia historyjek według nienaturalnej punktacji, aby osiągnąć ich zbliżony rozmiar.
- I w końcu rzecz najważniejsza — jedną z podstawowych zalet nadawania ocen jest wiedza uzyskiwana w trakcie rozmów o rozmiarach elementów. Nic tak nie pobudza dyskusji jak poproszenie ludzi, aby przypisali czemuś ocenę, co niemal natychmiast spowoduje ujawnienie wszelkich różnic zdań i przyjętych założeń. Jeżeli mielibyśmy odpuścić sobie nadawanie ocen, musielibyśmy znaleźć inną równie skuteczną metodę promowania zdrowej dyskusji.

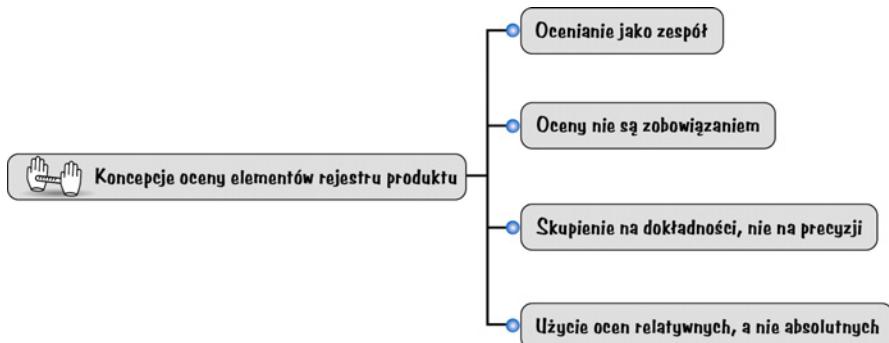
## Ocena zadań

Na bardziej szczegółowym poziomie mamy zadania znajdujące się w rejestrze sprintu. Większość zespołów decyduje się nadać oceny swoim zadaniom podczas planowania sprintu, dzięki czemu mogą upewnić się, że zobowiązanie, które mają zamiar podjąć, ma racjonalne podstawy (więcej szczegółów na ten temat znajdziesz w rozdziale 19.).

Rozmiary zadań wyrażane są w **idealnych godzinach** (określanych również jako roboczogodziny). Na rysunku 7.2 zespół ocenia, że zaprojektowanie interfejsu użytkownika zajmie pięć roboczogodzin. To nie znaczy, że będzie to pięć godzin zegarowych. Zaimplementowanie interfejsu użytkownika może zająć jednej osobie kilka dni, natomiast kilku osobom mniej niż dzień. Ocena mówi jedynie, jak duży wysiłek zespołu powinien wystarczyć do zrealizowania danego zadania. Do omówienia nadawania ocen zadaniom powróćmy w rozdziale 19., kiedy będę opisywał szczegółowo planowanie sprintu.

## Koncepcje nadawania ocen elementom rejestru produktu

Chociaż wszystkie trzy poziomy szczególowości są istotne, pozostała część tego rozdziału skupia się na ocenianiu elementów rejestru produktu. Zespoły scrumowe korzystają z kilku istotnych koncepcji podczas nadawania ocen elementom rejestru produktu (patrz rysunek 7.3).



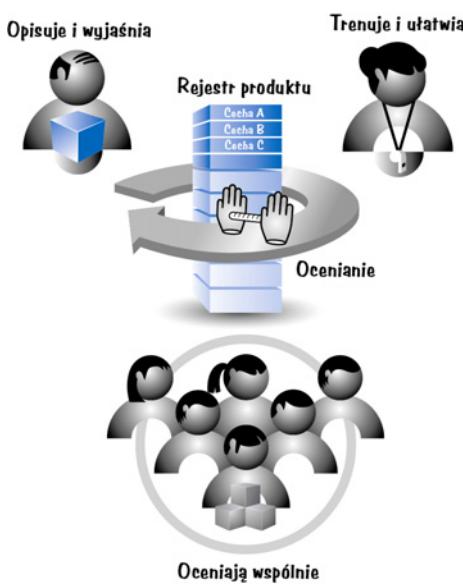
**RYSUNEK 7.3.** Koncepcje oceny elementów rejestru produktu

Przeanalizujmy je kolejno.

### Nadawanie ocen jako zespół

W wielu firmach o tradycyjnym modelu organizacji początkowe oceny nadawane są przez menedżera projektu, architekta systemu lub lidera zespołu deweloperskiego. Pozostali członkowie zespołu mają potencjalną szansę przejrzenia tych ocen i wydania własnej opinii na ich temat w późniejszym czasie. W Scrumie przestrzegamy prostej zasady: oceny nadają osoby, które będą wspólnie wykonywać rzeczywistą pracę.

Dla pełnej jasności, kiedy piszę „ludzie wykonujący rzeczywistą pracę”, mam na myśli zespół deweloperski, który faktycznie zaprojektuje, zbuduje i przetestuje elementy rejestru produktu. Oceny nie dostarczają właściciela produktu i mistrza młyna. Są one obecni podczas głosowania nad elementami rejestru, ale sami nie podają ocen elementów (patrz rysunek 7.4).



**RYSUNEK 7.4.** W nadawaniu ocen bierze udział cały zespół scrumowy

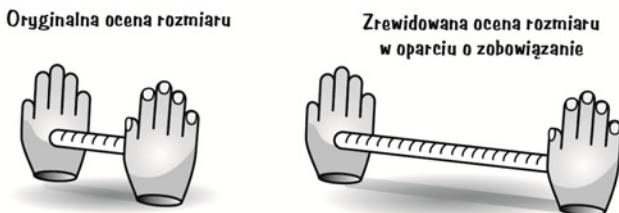
Rolą właściciela produktu jest opisywanie elementów rejestru produktu i odpowiadanie na pytania wyjaśniające zadawane przez zespół deweloperski. Właściciel produktu nie powinien kierować lub „prowadzić” zespołu w stronę oczekiwanej przez niego oceny. Rolą mistrza młyna jest trenowanie i ułatwianie aktywności nadawania ocen.

Celem dla zespołu deweloperskiego jest dostarczenie oceny każdego elementu rejestru produktu ze wspólnej perspektywy wszystkich jego członków. Ponieważ każdy, zależnie od swojej wiedzy eksperckiej, postrzega historyjkę z innego punktu widzenia, duże znaczenie ma obecność wszystkich członków zespołu w trakcie głosowania.

## Oceny nie są zobowiązaniem

Oceny nie są zobowiązaniem i ważne jest, abyśmy nie traktowali ich w taki sposób. To zdanie odnosi się zwykle do menedżerów. „Co masz na myśli, mówiąc, że nie prosimy zespołu, aby zobowiązywał się do swoich ocen? W jaki sposób osiągniemy precyzyjne oceny, jeśli zespół nie będzie tego robił?”

Kiedy ten temat zostanie podniesiony w trakcie moich zajęć, przeprowadzam prostą demonstrację wizualną, aby przekazać mój punkt widzenia. Biorę do ręki karteczkę samoprzylepną i mówię „Wyobraźmy sobie, że proszę was o nadanie oceny tej historyjce, a wy mówicie mi, że jest duża”. Następnie używam rąk do zilustrowania rozmiaru historyjki, tak jak pokazuje to lewa strona rysunku 7.5.



**RYSUNEK 7.5.** Wynik przyjęcia zobowiązań wobec ocen

Następnie dodaję coś takiego „Ach, zapomniałem dodać, że wasza przyszłoroczna premia zależy w całości od poprawności waszych ocen. Macie teraz jeszcze szansę do zrewidowania swojej oceny”. W tym momencie zaczynam rozsuwać dłonie, wskazując coraz większe oceny (prawa strona rysunku 7.5). Na ogół rzucam wtedy: „Powiedziecie mi, kiedy przestać, mam ograniczony zasięg ramion. Nie jestem koszykarzem!”.

Mój cel jest oczywisty. Jeżeli proszę ludzi o ocenę historyjki, oczekuję oceny zgodnej z rzeczywistością. Jeżeli potem dodam jeszcze, że premie zależą od poprawności oceny, wszyscy, ze mną włącznie, podadzą oceny znacznie większe od tych uważanych oryginalnie za poprawne.

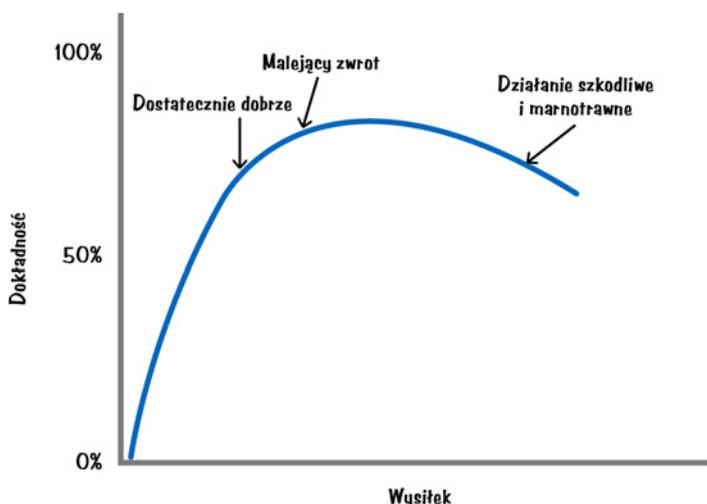
Oceny powinny być realnymi odpowiednikami wielkości rzeczy. Nie chcemy, aby były sztucznie zawyżane pod wpływem zewnętrznych nacisków. Takie podejście zniekształca jedynie harmonogramy i prowadzi do wzajemnych przepychanek pomiędzy członkami zespołu deweloperskiego, którzy zawyżają oceny, a zarządem, który je redukuje. Po zakończeniu całego procesu nie rozumiemy ostatecznych liczb, ponieważ przez cały czas manipulowały nimi różne osoby.

## Dokładność kontra precyzja

Nasze oceny powinny być dokładne, ale bez przesadnej precyzji. Chyba każdy z nas miał okazję zetknąć się z produktami, w przypadku których oceny były podawane na absurdalnym wręcz poziomie **dokładności**. Na przykład realizacja zadania miała zająć 10,275 roboczogodziny lub projekt miał kosztować 123 856,87 złotego.

Tworzenie tego typu błędnych, nazbyt precyzyjnych ocen jest marnotrawstwem. Po pierwsze: pojawia się marnotrawstwo wynikające z wysiłku włożonego w wytworzenie takich ocen — jego rozmiar może być znaczny. Po drugie: marnotrawstwo powstaje w wyniku oszukiwania samych siebie myśleniem, że rozumiemy coś, czego tak naprawdę nie rozumiemy, i na tej podstawie podejmowania ważnych, błędnych i kosztownych biznesowo decyzji.

Powinniśmy inwestować tyle wysiłku, ile wystarczy, aby uzyskać dobre, w przybliżeniu prawne oceny (patrz rysunek 7.6).



RYSUNEK 7.6. Porównanie wysiłku włożonego w planowanie z otrzymaną dokładnością

Podczas oceniania zawsze dojedziemy do punktu malejącego zwrotu. Przekroczenie tego punktu powoduje, że każda kolejna jednostka wysiłku zainwestowanego w ocenianie nie powoduje proporcjonalnego wzrostu **dokładności** tej oceny. Za tym punktem zaczynamy zwyczajnie marnować nasz czas i prawdopodobnie wpływać negatywnie na dokładność ocen przez analizowanie nadmiernej ilości danych o małej wartości.

## Ocenianie w sposób względny

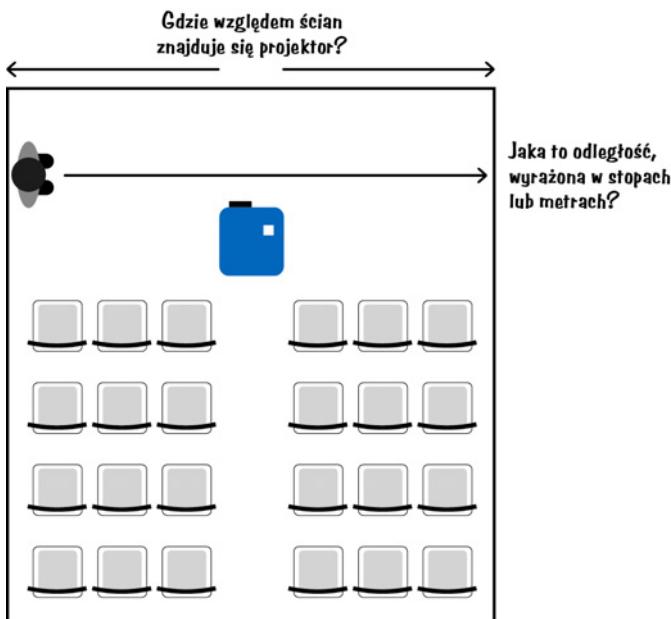
Elementy rejestru produktu powinniśmy oceniać, używając ocen względnych, a nie absolutnych. Porównujemy elementy, aby określić, jak duży jest każdy z nich w odniesieniu do innych (rysunek 7.7).

Jak pokazuje rysunek 7.7, łatwo jest oceniać rozmiar szklanki w odniesieniu do innych stojących w pobliżu, ale o wiele trudniej będzie uczynić to samo w stosunku do faktycznej (absolutnej) ilości płynu w każdej z nich.



**RYSUNEK 7.7.** Ocenianie w sposób wzajemny

Moje osobiste obserwacje przekonały mnie, że ludzie znacznie lepiej oceniają w sposób wzajemny niż w sposób absolutny. Oto przykład, jakiego używam podczas swoich zajęć w celu udowodnienia tej tezy (patrz rysunek 7.8).



**RYSUNEK 7.8.** Porównanie oceniania w sposób wzajemny i absolutny

Rozpoczynam od podejścia do jednego z boków sali wykładowej i odwrócenia się twarzą w kierunku ściany po przeciwej stronie. Proszę wszystkich w sali o zapisanie w rzeczywistych jednostkach (metrach, stopach) odległości, jaka ich zdaniem dzieli mnie od ściany po przeciwej stronie. (Osobom, które patrzą na sufit i liczą płyty podwieszanego sufitu, mówię, żeby przestały oszukiwać!)

W wielu salach wykładowych gdzieś pośrodku znaleźć można projektor zamocowany do sufitu. W dalszej kolejności proszę zatem wszystkich, aby zapisały drugą ocenę pozycji projektora względem odległej ściany i mnie.

Niemal zawsze dostaję takie same wyniki. W typowej sali wykładowej o szerokości 10 metrów odpowiedź na pytanie „Jak daleko znajduję się od ściany po drugiej stronie w jednostkach absolutnych?” daje na ogół 27 różnych wyników. Kiedy zapytam „Gdzie w stosunku do mnie i przeciwległej ściany znajduje się projektor?”, 29 z 30 osób mówi „mniej więcej pośrodku”. Trzydziesta osoba na ogół droczy się ze mną, podając wynik w stylu „5/11 całej odległości”.

Nie jest to absolutnie rygorystyczny eksperyment naukowy, ale większość osób wydaje się natychmiast zgodzić z ideą, że znacznie lepiej oceniają w sposób względny niż absolutny. Dla porównania: czasami projektor znajduje się w jednej trzeciej lub dwóch trzecich szerokości sali ode mnie i wtedy wyniki niemal zawsze są takie same: większość osób zapisuje taką samą względową odległość ode mnie.

Podsumowując, jeżeli zamierzamy prosić ludzi o podanie swoich ocen, powinniśmy wybrać technikę, w której ludzie są dobrzy (ocenianie w sposób względny), a nie technikę, w której są kiepscy (ocenianie w sposób absolutny).

## Jednostki służące do oceny

Chociaż nie ma ustandaryzowanej jednostki oceny rozmiaru elementów rejestru produktu, dwoma najczęściej używanymi są punkty historyjkowe i idealne dni. Wybór jednej lub drugiej jednostki jest arbitralny. W mojej ocenie 70% organizacji, z którymi współpracowałem, używa punktów historyjkowych, pozostałe 30% używa idealnych dni. Przyjrzyjmy się każdej z tych jednostek.

### Punkty historyjkowe

Punkty historyjkowe mierzą wielkość zakresu elementu rejestru produktu. Oczekujemy wpływu kilku czynników na punkty, takich jak złożoność i rozmiar fizyczny. Duży rozmiar niekoniecznie oznacza, że coś jest fizycznie duże. Historyjka może reprezentować stworzenie złożonego algorytmu biznesowego. Wynik końcowy nie będzie wielki, ale wysiłek potrzebny do jego stworzenia może taki być. Z drugiej strony, historyjka może być całkiem spora fizycznie, ale nieskomplikowana. Założymy, że musimy zmodyfikować każdą komórkę w arkuszu zawierającym 60 tysięcy wierszy. Żadna z pojedynczych operacji modyfikacji nie jest trudna, ale nie ma możliwości ich zautomatyzowania. Ile takiej pracy jesteśmy w stanie zrealizować w sprincie? Chociaż historyjka nie jest skomplikowana, będzie miała duży rozmiar.

Punkty historyjkowe łączą czynniki takie jak złożoność i rozmiar fizyczny w jedną pojedynczą miarę rozmiaru. Celem jest osiągnięcie zdolności porównywania historyjek i możliwości stwierdzenia w stylu „Jeżeli historyjka utworzenia rekordu ma rozmiar 2, to historyjka wyszukiwania rekordu ma rozmiar 8”, co oznacza, że historyjka z wyszukiwaniem jest mniej więcej cztery razy większa od historyjki z tworzeniem rekordów.

W przykładzie z początku rozdziału przyjęte podejście polegało na określeniu rozmiarów elementów rejestru produktu, a następnie określeniu czasu trwania jako sumy rozmiarów podzielonej przez średnią prędkość zespołu. Ponieważ miary rozmiaru takie jak punkty historyjkowe służą ostatecznie do wyliczenia czasu (trwania), muszą one odzwierciedlać wysiłek związany z historyką, postrzegany z punktu widzenia zespołu deweloperskiego.

## Idealne dni

Alternatywnym podejściem do oceny elementów rejestru produktu jest użycie idealnych dni. Idealne dni są znaną jednostką — reprezentują osobodni potrzebne do ukończenia historyjki. Idealny czas różni się od czasu zegarowego. Idealny czas trwania meczu amerykańskiego futbolu to cztery kwarty, z których każda ma 15 minut (zatem czas rozgrywania meczu to idealna godzina). W rzeczywistości rozegranie gry zajmuje od trzech do trzech i pół godziny.

Wspomniałem wcześniej, że żadna z tych jednostek — idealnych dni lub punktów historyjkowych — nie ma istotnej przewagi nad drugą. Istnieje jednak poważny czynnik działający na niekorzystność idealnych dni — ryzyko błędnej interpretacji.

Na przykład mamy teraz wczesne wtorkowe popołudnie, a ja pokazuję Ci element rejestru produktu i pytam: „Jak duży jest ten element?”. Ty odpowiadasz: „Dwa dni”. Wtedy ja mówię: „Dobrze, w takim razie skończysz do wczesnego popołudnia w czwartek”. Ty odpowiadasz: „Nie, kończę jeszcze dwudniowe zadanie dzisiaj i jutro [środa]. Potrzebuję całego dnia, żeby się zorganizować, zatem tym elementem będę mógł się prawdopodobnie zająć w czwartek. Ponieważ jednak nie mam całych wolnych dni na poświęcenie uwagi temu elementowi, myślę, że będę gotowy gdzieś w okolicach poniedziałku”. Na co ja odpowiadam: „Nie rozumiem, powiedziałeś mi, że to historyjka na dwa dni, w związku z czym powinieneś mieć ją gotową w czwartek”. Wtedy Ty odpowiadasz: „Mówię o dwóch dniach idealnych, nie dniach kalendarzowych. Proszę, żebyś nie przekładał moich idealnych dni bezpośrednio na kalendarz, to nie działa w taki sposób”.

W przypadku 30% organizacji, z którymi miałem styczność i które z powodzeniem używały idealnych dni, ich komentarz do powyższej konwersacji byłby następujący: „Tak, ale my nie mamy tego problemu z błędą interpretacją. Mówimy ludziom dwa dni, a oni wiedzą, że nie chodzi o dwa dni kalendarzowe”.

Jeżeli w Twojej organizacji istnieje małe ryzyko złej interpretacji, idealny czas będzie dobrze funkcjonował. Jeśli jednak uważasz, że ludzie będą błędnie interpretować idealny czas, lepiej będzie zastosować punkty historyjkowe.

Są jeszcze inne różnice pomiędzy punktami historyjkowymi i idealnym czasem, ale błędna interpretacja jest jednym z największych problemów. Pewna studentka w trakcie moich zajęć podsumowała swoje preferencje odnośnie do obu możliwości, mówiąc swoim kolegom „Słuchajcie, używaliśmy idealnych dni przez ostatnie 15 lat mojej pracy tutaj i to podejście nigdy nie działało. Powiem szczerze, wolałabym spróbować czegoś innego”.

## Planowanie pokerowe

**Planowanie pokerowe** jest techniką określania rozmiaru elementów rejestru produktu, ujętą po raz pierwszy przez Jamesa Grenninga [Greening, 2002], a następnie spopularyzowaną przez Mike'a Cohna [Cohn, 2006]. Planowanie pokerowe bazuje na kilku istotnych koncepcjach (patrz rysunek 7.9).



**RYSUNEK 7.9.** Koncepcje planowania pokerowego

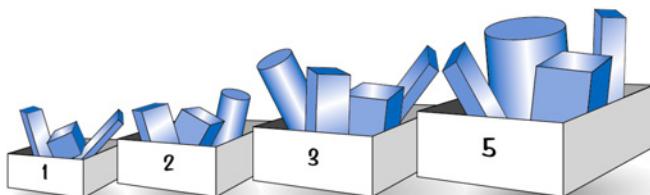
Planowanie pokerowe jest techniką oceny potrzebnego wysiłku opartą na porozumieniu (konsensusie). Osoby posiadające odpowiednią wiedzę — eksperci — wyznaczone do pracy nad elementem rejestru angażują się w intensywną dyskusję, aby ujawnić wszelkie założenia, zrozumieć istotę problemu, a następnie wycenić rozmiar elementu. Planowanie pokerowe generuje względne oceny rozmiaru przez prawidłowe grupowanie lub też zbieranie elementów o podobnych rozmiarach. Zespół korzysta z historii ocen elementów rejestru produktu w celu ułatwienia sobie oceny elementów nowych.

## Skala ocen

Aby móc przeprowadzić planowanie pokerowe, zespół musi zdecydować, jakiej skali lub sekwencji liczb użyć do przypisywania ocen. Naszym celem jest osiągnięcie odpowiedniego poziomu dokładności, ale bez zbytniej szczegółowości, dlatego nie będziemy stosować wszelkich możliwych liczb. Zamiast tego będziemy preferować pewną skalę rozmiarów z większą ilością liczb po stronie dolnej granicy zakresu oraz z liczbami bardziej rozproszonymi po górnej stronie zakresu.

Najczęściej stosowaną jest zaproponowana przez Mike'a Cohna skala oparta na zmodyfikowanym ciągu Fibonacciego: 1, 2, 3, 5, 8, 13, 20, 40 i 100. Inna skala możliwa do zastosowania bazuje na potęgowaniu: 1, 2, 4, 8, 16, 32, ....

Stosując tego typu skalę, grupujemy lub też wrzucamy do jednego worka pod względem rozmiaru elementy rejestru produktu i przypisujemy im ten sam rozmiar na skali. Aby zilustrować tę koncepcję, założmy, że pracujemy na poczcie i musimy pogrupować paczki o zbliżonym rozmiarze, wrzucając je do tego samego pojemnika (patrz rysunek 7.10).



**RYSUNEK 7.10.** Planowanie pokerowe z wykorzystaniem pojemników

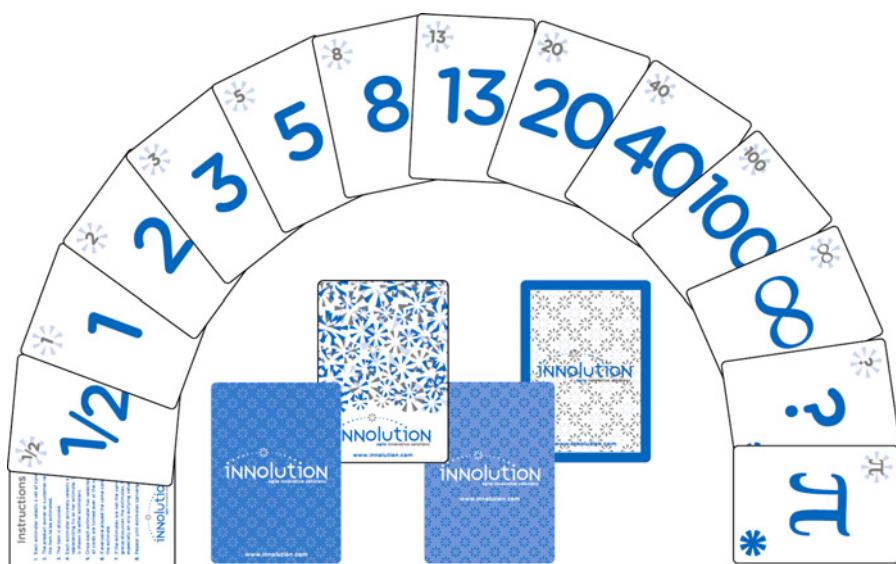
Kiedy otrzymamy paczkę, decydujemy, w którym worku należy ją umieścić. Trzeba przyjąć, że nie wszystkie przesyłki w tym samym pojemniku będą miały identyczny kształt, wymiary lub wagę. W związku z tym będziemy musieli przeanalizować paczki znajdujące się już w pojemnikach i w ten sposób znaleźć najlepsze miejsce dla paczki, którą właśnie oceniamy. Po znalezieniu pojemnika, do którego paczka nadaje się najlepiej, przechodzimy do kolejnej przesyłki. Oczywiście im więcej paczek umieścimy w pojemnikach, tym łatwiej będzie ocenić i rozmiścić paczki, które dopiero nadjejdą, ponieważ będziemy mieli więcej punktów odniesienia.

W celu uniknięcia przesadnej dokładności nie mamy pojemnika o rozmiarze „4” (zakładając, że korzystamy ze skali opartej na ciągu Fibonacciego). Zatem kiedy dostaniemy paczkę, która wydaje się większa niż 2, ale mniejsza niż 8, musimy umieścić ją w pojemniku „3” lub „5”.

## Zasady gry

W planowaniu pokerowym udział bierze cały zespół. W trakcie sesji właściciel produktu prezentuje, opisuje i wyjaśnia szczegóły elementów rejestru produktu. Mistrz młyna trenuje zespół, aby pomóc mu w lepszym wykonaniu planowania pokerowego. Jego rolą jest również nieustanne wyszukiwanie i angażowanie w spór ludzi, którzy poprzez mowę swojego ciała lub powstrzymywanie się od dyskusji wydają się posiadać odmienną opinię. Oceny są wypracowywane wspólnie przez zespół deweloperski.

Każdy członek zespołu deweloperskiego otrzymuje karty do planowania pokerowego (patrz rysunek 7.11).



**RYSUNEK 7.11.** Karty do planowania pokerowego Innolution

Powszechnie przyjęty sposób rozumienia tych kart przedstawiony został w tabeli 7.1.

**TABELA 7.1.** Powszechnie przyjęta interpretacja kart do planowania pokerowego

Karta	Znaczenie
0	Niewidoczna na rysunku 7.11, ale obecna w niektórych talich celem wskazania, że element został już wykonany lub jest tak mały, że nie warto nadawać mu jakiegokolwiek numeru.
1/2	Używana do wyceny bardzo małych elementów.
1, 2, 3	Używane do wyceny małych elementów.
5, 8, 13	Używane do wyceny elementów średniego rozmiaru. Dla wielu zespołów element o rozmiarze 13 jest największym, jaki gotowe są przyjąć do realizacji w sprincie. Elementy większe zostałyby podzielone na zbiór mniejszych.
20, 40	Używane do wyceny dużych elementów (na przykład cech lub tematów).
100	Bardzo duża cecha lub epoś.
$\infty$ (nieskończoność)	Używana do wskazania, iż element rejestru jest tak duży, że nie ma sensu przypisywać mu jakiekolwiek liczby.
? (znak zapytania)	Używana do wskazania, iż członek zespołu nie rozumie elementu i prosi właściciela produktu o podanie dodatkowych wyjaśnień. Niektórzy członkowie zespołu używają również znaku zapytania jako sposobu na uratowanie się przed koniecznością nadania oceny bieżącemu elementowi — zazwyczaj wynika to ze słabego związku z zespołem i braku pojęcia, jak należy ocenić dany element. Choć dopuszczalne jest powstrzymanie się od nadania własnej oceny, niedopuszczalny jest brak uczestnictwa w procesie! Zatem to, że czujemy się niekomfortowo, oddając głos, nie stanowi dostatecznego usprawiedliwienia do powstrzymania się od udziału w dyskusji lub rezygnacji z obowiązku udzielania pomocy zespołowi w znalezieniu konsensusu.
$\pi$ (pi)	W tym kontekście $\pi$ nie oznacza wartości 3,1415926! Karta pi jest używana, kiedy członek zespołu chce powiedzieć „Jestem zmęczony i głodny, mam ochotę na kawałek ciasta!”. Inne talie do planowania pokerowego zamiast karty pi zawierają kartę z rysunkiem filiżanki kawy. Niezależnie od formy karta ta podkreśla istotną rzecz. Członkowie zespołu mogą angażować się w intensywną dyskusję jedynie przez ograniczoną ilość czasu (być może godzinę lub dwie). Po tym czasie jest im potrzebna przerwa — w przeciwnym razie entuzjazm do prowadzenia dalszej dyskusji zamieni się w wysiłek mający na celu jak najszybsze przebrnięcie przez przypisywanie ocen bez większego zwracania uwagi na ich prawidłowość, czy też sam proces pozyskiwania wiedzy, jaki niesie ze sobą taka dyskusja. Jeżeli uczestnicy głosują kartą pi, zespół musi zrobić przerwę.

Reguły planowania pokerowego wyglądają następująco:

1. Właściciel produktu wybiera element rejestru produktu do oceny i czyta jego treść zespołowi.
2. Zespół deweloperski dyskutuje nad elementem i zadaje pytania wyjaśniające właścielowi produktu, który na nie odpowiada.
3. Każda osoba głosująca wybiera dyskretnie kartę reprezentującą jej ocenę.
4. Kiedy wszyscy oceniający ustalą dyskretnie swoją ocenę, karty zostają odkryte jednocześnie, ujawniając w tym samym czasie wszystkie oceny.

5. Jeżeli każdy wybrał taką samą kartę, wtedy liczba będąca wynikiem konsensusu staje się oceną elementu rejestru produktu.
6. Jeżeli oceny nie są jednakowe, członkowie zespołu angażują się w dyskusję, aby ujawnić wszelkiego typu założenia i różnice w pojmowaniu treści elementu. Zazwyczaj zaczynamy od poproszenia osób, które głosowały najniżej i najwyższej, aby wyjaśnili przyczyny podania takich, a nie innych ocen.
7. Po zakończonej dyskusji wracamy do kroku numer 3 i powtarzamy czynności od tego momentu aż do osiągnięcia porozumienia.

Przy planowaniu pokerowym nie uśredniamy wartości głosów i nie uznajemy wartości niebędących na skali (kartach). Naszym celem jest, aby zespół deweloperski nie godził się na kompromis, lecz osiągał konsensus co do całkowitego rozmiaru historyjki (wysiłku z nią związanego) ze swojego własnego punktu widzenia. Zwykle konsensus taki jest możliwy do osiągnięcia po dwóch, trzech rundach głosowania, między którymi dyskusja prowadzona przez członków zespołu pomaga osiągnąć wzajemne rozumienie historyjki.

## Zalety

Planowanie pokerowe zbliża do siebie grupę ludzi o różnych osobowościach, którzy będą razem pracować, i pozwala im osiągnąć porozumienie w sprawie oceny elementu. Wypracowana wspólnie ocena jest często znacznie lepsza od jakiekolwiek innej podanej przez indywidualnych członków zespołu.

Jak wspomniałem wcześniej, niektórzy członkowie społeczności metod zwinnych uważają, że ocenianie elementów rejestru produktu nie jest warte zachodu. Jednak z drugiej strony, dyskusje prowadzone podczas planowania pokerowego elementów mają bardzo dużą wartość. Z mojego własnego doświadczenia mogę powiedzieć, że kiedy poprosisz ludzi o przypisanie oceny elementowi, pobudzisz ich tym samym do myślenia o szczegółach, a także ujawnisz wszelkie ukryte założenia.

Najcenniejszą wartością związaną z planowaniem pokerowym jest dyskusja i lepsze zrozumienie elementu rejestru produktu przez zespół deweloperski. Mam nadzieję, że zespoły te przypisują również oceny elementom, ale bardziej interesuje mnie, aby zdobywały one wiedzę o elementach. Jeżeli tak się stanie, otrzymają odpowiednio wysoki zwrot z włożonego przez siebie wysiłku.

## Czym jest prędkość?

**Prędkość** to ilość pracy wykonanej<sup>1</sup> w każdym sprincie. Jest ona mierzona poprzez zsumowanie rozmiarów elementów rejestru produktu, które zostały zrobione przed końcem sprintu. Element rejestru jest zrobiony (ukończony) lub nie. Właściciel produktu nie otrzymuje żadnej wartości z elementów, które nie zostały ukończone, zatem prędkość nie uwzględnia elementów wykonanych częściowo.

<sup>1</sup> Zgodnie z kryteriami ukończenia — przyp. tłum.

Prędkość mierzy wynik wyjściowy (rozmiar tego, co zostało dostarczone), a nie wartość wyjściową (wartość, tego, co zostało dostarczone). Przyjmujemy założenie, że jeżeli właściciel produktu wyraził zgodę na realizację elementu przez zespół, element ten musiał stanowić dla niego pewną wartość. Nie oznacza to jednak wcale, że ukończenie elementu o rozmiarze 8 przynosi większą wartość w porównaniu z wykonaniem elementu o rozmiarze 3. Być może element o rozmiarze 3 ma dużą wartość i dlatego pracujemy nad nim wcześniej (ze względu na wysoką wartość i mały koszttworzenia), a element o rozmiarze 8 odkładamy na później (ponieważ ma niższą wartość i większy koszttworzenia).

Prędkość służy dwóm istotnym celom. Po pierwsze: ma ona kluczowe znaczenie podczas planowania scrumowego. W przypadku planowania na poziomie wersji dystrybucyjnej (rysunek 7.1) decydujemy o rozmiarze wersji, używając średniej prędkości zespołu do obliczenia liczby sprintów niezbędnych do ukończenia tej wersji. Dodatkowo podczas planowania sprintu prędkość zespołu pomaga w oszacowaniu zdolności produkcyjnej i wynikającej stąd ilości pracy, jaką zespół jest w stanie zrealizować w nadchodzącym sprincie (patrz rozdział 19.).

Po drugie: prędkość jest parametrem diagnostycznym, który może zostać wykorzystany przez zespół do oceny i poprawy procesu scrumowego i w konsekwencji wartości dostarczanej klientom. Obserwując swoją własną prędkość, zespół jest w stanie lepiej zrozumieć, jak poszczególne zmiany w procesie wpływają na dostarczaną, mierzalną wartość dla klienta.

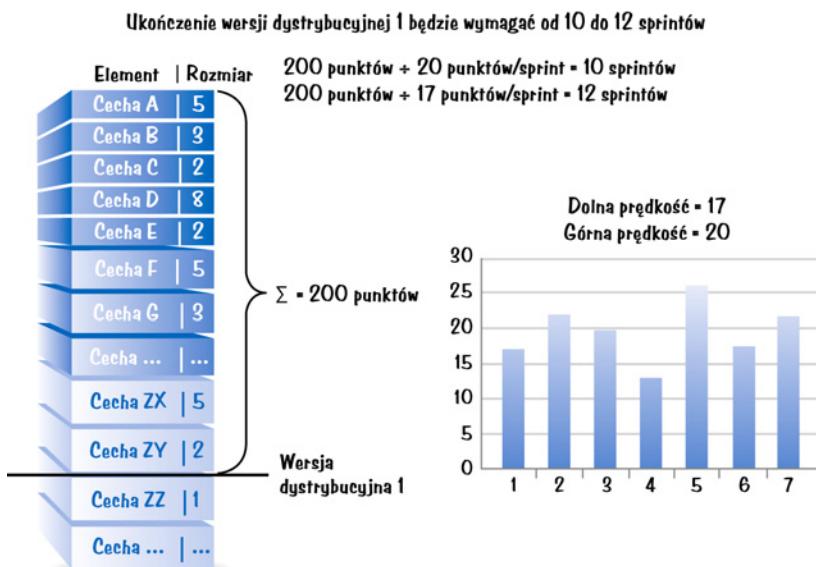
## Obliczanie przedziału prędkości

W przypadku planowania prędkość ma największą wartość, kiedy wyrażona jest w formie przedziału. Na przykład: „Zespół może typowo zrealizować od 25 do 30 punktów historyjkowych w każdym sprincie”. Użycie przedziału pozwala nam na osiągnięcie odpowiedniej dokładności, ale bez przesadnej precyzji.

Mając do dyspozycji przedział prędkości, dokładniej będziemy odpowiadać na pytania typu „Kiedy to zostanie zrobione?”, „Ile elementów jesteśmy w stanie ukończyć?” lub „Ile to wszystko będzie kosztować?”. Ponieważ większość z tych pytań jest zadawana we wczesnej fazie produkcji, kiedy mamy najmniej informacji na temat tego, co produkujemy, udzielenie precyzyjnych odpowiedzi na nie jest niemożliwe. Używając przedziału, jesteśmy w stanie wyrazić naszą niepewność (patrz rysunek 7.12).

W tym przykładzie (modyfikacja rysunku 7.1) zamiast deklarowania (a właściwie zgadywania przez nas) dokładnego numeru sprintu, po którym wszystkie elementy z wersji dystrybucyjnej zostaną ukończone, dostarczamy odpowiedzi na pytanie w postaci pewnego przedziału. Wyliczenie tego przedziału wymaga znajomości dwóch prędkości naszego zespołu. Jeżeli podzielimy rozmiar wersji przez szybszą prędkość zespołu, otrzymamy najmniejszą liczbę potrzebnych sprintów. Natomiast dzieląc rozmiar wersji przez wolniejszą prędkość zespołu, otrzymamy największą liczbę potrzebnych sprintów.

Dwie wartości prędkości (w naszym przykładzie 17 i 20) możemy uzyskać, stosując prostą matematykę (w postaci niskiej i wysokiej średniej, 90% przedziałów ufności itp.) na historycznych danych prędkości naszego zespołu. Więcej szczegółów na temat przeprowadzania tego typu obliczeń w celu odpowiedzenia na pytania na temat czasu ukończenia wersji, ilości sprintów i zakresu prac przekażę w rozdziale 18.



**RYSUNEK 7.12.** Wyliczanie i zastosowanie przedziału prędkości

## Prognozowanie prędkości

W poprzednich przykładach założyłem, że zespół posiadał dane historyczne prędkości, których mogliśmy użyć do przewidzenia przyszłej prędkości. Jedną z niewątpliwych zalet długowiecznych zespołów jest to, że posiadają one takie użyteczne dane historyczne (więcej zalet długowiecznych zespołów znajdziesz w rozdziale 11.). Jak jednak poradzimy sobie w sytuacji zupełnie nowego zespołu, którego członkowie nie pracowali ze sobą do tej pory i w związku z tym nie istnieją dane historyczne dotyczące prędkości? Będziemy musieli dokonać prognozy.

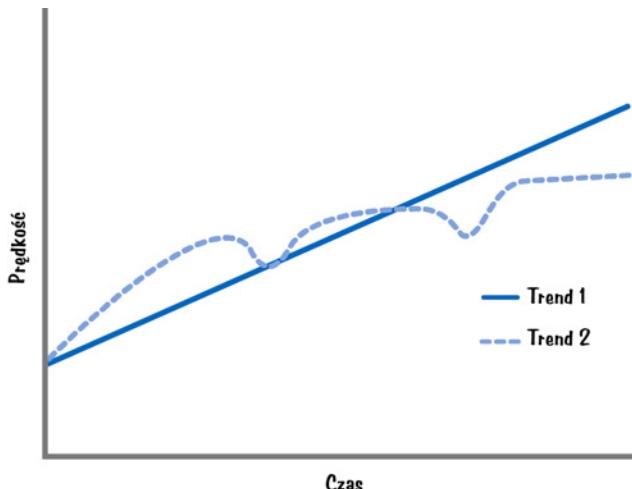
Jedną z często stosowanych metod prognozowania prędkości jest poproszenie zespołu o przeprowadzenie planowania sprintu i ustalenie zbioru elementów rejestru produktu, jakie zobowiązże się wykonać w trakcie pojedynczego sprintu. Jeżeli zobowiązanie wydaje się być racjonalne, możemy zsumować rozmiary wybranych elementów rejestru produktu i otrzymać prognozę prędkości zespołu.

Ponieważ to co nas naprawdę interesuje to *przedział prędkości*, możemy poprosić zespół o zaplanowanie dwóch sprintów, a następnie użyć jednej z wartości jako wyższej prędkości, a drugiej jako niższej (istnieje duże prawdopodobieństwo, że te dwie wartości będą różne). Inna możliwość to przyjęcie pewnych intuicyjnych poprawek pojedynczej prędkości podanej przez zespół w oparciu o dane historyczne innych zespołów i w ten sposób wygenerowanie szacunkowego przedziału prędkości.

Po zakończeniu sprintu przez zespół będziemy mieć w ręku faktyczny pomiar prędkości — wtedy powinniśmy odrzucić wartość szacunkową i użyć wartości rzeczywistej. W miarę generowania przez zespół danych historycznych swojej prawdziwej prędkości powinniśmy zacząć wyliczać przedział prędkości, stosując średnie lub inne metody statystyczne (więcej przykładów znajdziesz w opracowaniu [Cohn, 2009]).

## Wpływanie na prędkość

Czy uważasz, że prędkość zespołu powinna rosnąć w miarę upływu czasu? Pewien przedstawiciel kadry zarządzającej powiedział mi kiedyś: „W zeszłym roku prędkość mojego zespołu wynosiła średnio 30 punktów na sprint, w tym roku oczekuję wyniku średnio 35 punktów na sprint”. Ten menedżer uważa, że prędkość zespołu powinna odpowiadać trendowi numer 1 na rysunku 7.13.



**RYSUNEK 7.13.** Prędkość zespołu w miarę upływu czasu

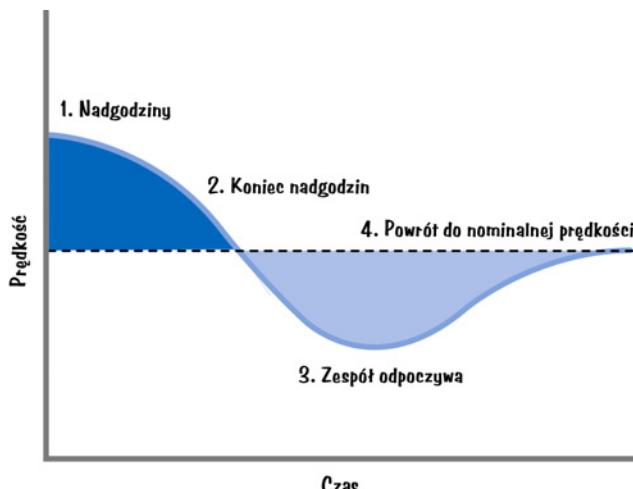
W jego przekonaniu jeśli zespół dokonuje nieustannej inspekcji i adaptacji (przez cały czas poprawia swoją wydajność), jego prędkość powinna również nieustannie rosnąć.

Ja uważam, że zespół może spodziewać się wzrostu prędkości, jeśli próbuje w sposób agresywny poprawić swoje działania i skupia się na dostarczaniu cech zgodnie z dobrą definicją ukończenia i przy niskim poziomie dłużu technicznego (patrz rozdział 8.). Mówiąc o wzroście, mam na myśli osiągnięcie pewnego pułapu, na którym prędkość zespołu najprawdopodobniej będzie stała (coś na wzór trendu numer 2 z rysunku 7.13).

Osiągnięcie wyrównanego poziomu nie oznacza wcale, że zespół nie ma potencjału do dalszego wzrostu. Zespół scrumowy i menedżerowie mogą skorzystać z szeregu metod, aby pomóc prędkości w dojściu do kolejnego poziomu. Pozytywny wpływ na prędkość może mieć na przykład wprowadzenie nowych narzędzi lub zwiększenie ilości szkoleń. Menedżerowie mogą również zmienić skład zespołu w nadzieję, że zmiana ta doprowadzi do poprawy jego całosciowej prędkości. Oczywiście należy zachować tutaj dużą ostrożność, ponieważ przesuwanie ludzi między zespołami bez jakiejkolwiek strategii może doprowadzić do utraty prędkości.

Chociaż wprowadzanie nowych narzędzi, szkolenie lub zmiana składu zespołu mogą odbić się pozytywnie na prędkości, działania te zazwyczaj powodują spadek prędkości w czasie, gdy zespół absorbuje i przetwarza zaistniałą zmianę (patrz rysunek 7.13, trend numer 2). Po tym spadku można spodziewać się wzrostu do nowego poziomu, który będzie utrzymany tak długo, aż pojawi się kolejna zmiana umożliwiająca przejście do jeszcze wyższego poziomu.

Istnieje jeszcze jeden oczywisty sposób, który można wykorzystać do poprawy prędkości — wydłużenie czasu pracy. Praca w godzinach nadliczbowych może początkowo spowodować wzrost prędkości (patrz „Nadgodziny” na rysunku 7.14).



**RYSUNEK 7.14.** Wpływ nadgodzin na prędkość (na podstawie rysunku z [Cook, 2008])

Po tym wzroście niemal na pewno nastąpi znaczny spadek prędkości połączony z utratą jakości produkcji. Nawet kiedy nadgodziny zostaną zlikwidowane, zespół będzie potrzebował pewnego czasu na odpoczynek, zanim będzie mógł powrócić do swojej nominalnej prędkości. Widziałem przypadki, w których dno wykresu (obszar zmniejszonej prędkości) w czasie powrotu zespołu do normy było większe od grzbietu (obszaru zwiększonej prędkości) w trakcie nadgodzin.

Stąd wniosek, iż utrzymywanie nadgodzin przez długi czas może dać krótkoterminowy zysk, ale w dłuższej perspektywie czasu będzie on niwelowany.

## Nieprawidłowe korzystanie z prędkości

Prędkość jest narzędziem służącym do planowania oraz parametrem diagnostycznym zespołu. Nie powinna ona być wykorzystywana jako środek pomiaru wydajności i oceny produktywności zespołu. Wykorzystanie prędkości w ten sposób może prowadzić do niebezpiecznych i marnotrawnych zachowań.

Oto przykład. Założymy, że zdecydowaliśmy się wręczyć największą premię zespołowi o największej szybkości. Na pierwszy rzut oka ta idea wydaje się rozsądna — spodziewamy się w końcu, że zespół o największej prędkości wykonuje najwięcej pracy w ciągu każdego sprintu, czy nie tak? Dlaczego nie nagrodzić takiego działania?

Jeżeli porównuję zespoły, które nie oceniają swoich elementów rejestru produktu przy użyciu jednakowej skali (co jest bardzo prawdopodobne), zestawienie ich prędkości ze sobą nie ma sensu. Powiedzmy, że zespół A przypisuje wartość 5 pewnemu elementowi, a zespół B ten sam element ocenia na 50. Zespół A wcale nie chce, aby oceniał jego prędkość w odniesieniu do zespołu B,

którego prędkość będzie dziesięć razy większa, nawet jeśli oba realizują fizycznie mniej więcej taką samą ilość pracy w każdym sprincie.

Kiedy problem zostanie zauważony przez członków zespołu A, zaczną oni „majstrować” przy systemie w taki sposób, aby ich prędkość była odpowiednio duża. Najprostszym sposobem na to będzie zmiana skali używanej przez zespół do oceniania elementów rejestru produktu. Zatem teraz zespół A ocenia ten sam element rejestru (wcześniej wart 5) na 500. Takie zjawisko nazywam **inflacją punktową** — nie ma ono żadnego celu z wyjątkiem dostosowania zachowania zespołu do źle rozumianego systemu pomiarowego. Nie rób tego.

Nawet przy takich samych jednostkach oceny elementów rejestru produktu w zespołach ustalenie przeze mnie systemu nagród bazującego na większych liczbach spowoduje, że otrzymam dokładnie to, o co prosiłem — większe liczby (inflację punktową).

Jeszcze gorsze od inflacji punktowej jest ignorowanie jakości celem zwiększenia liczby elementów ukończonych i tym samym osiągnięcie większej prędkości. Takie postępowanie powoduje stałe zwiększanie poziomu dłużu technicznego.

Podsumowując, powinniśmy traktować prędkość jako czynnik pomagający w prawidłowym przeprowadzeniu planowania, a także pozwalający na wewnętrzny rozwój zespołu. Kazde inne wykorzystanie stanie się najprawdopodobniej źródłem nadużyć.

## Zakończenie

W tym rozdziale omówiłem ocenianie rozmiarów, mierzenie prędkości oraz sposoby wyliczania czasu trwania produkcji. Pokazałem, w jaki sposób nadawanie ocen działa w przypadku elementów na poziomie rejestru portfela, rejestru produktu i konkretnych zadań. Następnie skupiłem się na elementach rejestru produktu, wskazując istotne koncepcje związane z ich oceną — w szczególności punkty historyczne i idealne dni. Później opisałem technikę nazywaną planowaniem pokerowym, używaną powszechnie do oceny elementów rejestru produktu.

Od tematu nadawania ocen przeszedłem do pojęcia prędkości i metod jej prawidłowego wykorzystania. Podkreśliłem fakt, iż prędkość ma największą wartość, kiedy wyrażona jest w formie przedziału, a nie pojedynczej wartości. Opisałem pokrótkę metody przewidywania prędkości dla nowego zespołu. Na koniec zająłem się najczęstszymi przypadkami błędnego wykorzystania prędkości. W następnym rozdziale skupię się na pojęciu dłużu technicznego i sposobach radzenia sobie z nim w Scrumie.



# Rozdział 8

## DŁUG TECHNICZNY

---

W tym rozdziale omówię koncepcję dłużu technicznego. Zacznę od definicji, która obejmuje dług natywny, dług nieunikniony i celowy dług techniczny. Następnie prześledzimy pewne często spotykane przyczyny powstawania dłużu technicznego, a także konsekwencje gromadzenia go w dużej ilości. Dalej opiszę trzy aktywności związane z dłużem technicznym: zarządzanie przyrostem dłużu technicznego, ujawnianie dłużu i jego obsługę. Podkreślę szczególnie sposób realizacji tych aktywności w Scrumie.

### Wprowadzenie

Pierwszą osobą, która opisała koncepcję **długu technicznego**, był Ward Cunningham [Cunningham, 1992]. Jego definicja wyglądała następująco:

*Wdrożenie pierwszej wersji kodu przypomina zadłużanie się. Mała ilość dłużu przyspiesza proces deweloperski tak długo, jak długo jest on spłacany odpowiednio szybko przez refaktoryzację... Ryzyko pojawia się, kiedy dług nie jest spłacany. Każda minuta spędzona na niezbyt poprawnym kodzie jest liczona jak odsetki od dłużu. Ilość dłużu wytworzona przez nieskonsolidowane implementacje może zablokować prace całych organizacji...*

Cunningham używał metafory dłużu technicznego, aby wyjaśnić swojemu zespołowi biznesowemu, dlaczego dobrą rzeczą jest tworzenie oprogramowania w sposób odpowiednio szybki, zapewniający informację zwrotną. Mówiąc o tym, podkreślał jednak dwa kluczowe czynniki: zespół i organizacja muszą być czujne pod względem spłacania dłużu w miarę wzrostu świadomości odnośnie do domeny prowadzonego przedsięwzięcia, a projektowanie i implementacja systemu muszą ewoluować, aby wspomagać ten wzrost świadomości.

Od momentu wprowadzenia określenia dłużu technicznego we wczesnych latach dziewięćdziesiątych przemysł softwarowy pozwolił sobie na pewne odstępstwa od definicji Cunninghama. Obecnie dług techniczny odnosi się zarówno do skrótów, które podejmujemy świadomie, jak również do wielu złych rzeczy, które dotykają systemów softwarowych. Są to między innymi:

- niedopasowany (błędny) projekt — projekt, który kiedyś miał sens, ale stracił go pod wpływem istotnych zmian w biznesie lub stosowanych przez nas obecnie technologii;
- błędy — znane problemy w oprogramowaniu, którym nie poświęciliśmy jeszcze czasu w celu ich rozwiązania;
- niewystarczające pokrycie testami — obszary, o których wiemy, iż powinny być dogłębniej testowane, ale mimo to nie są;
- nadmierne testowanie manualne — testowanie wykonywane ręcznie, które powinno zostać zastąpione testami automatycznymi;
- kiepskie zarządzanie integracją i wdrażaniem — wykonywanie tych czynności w sposób czasochłonny i podatny na błędy;
- brak doświadczenia na danej platformie — na przykład nasze główne aplikacje są napisane w języku COBOL, ale nie ma już w firmie wielu doświadczonych programistów tego języka;
- wiele innych przyczyn, ponieważ obecnie *dług techniczny* jest traktowany jako określenie wielu różnorodnych problemów.

Celem Cunninghama nie było odnoszenie dłużu technicznego do członka zespołu, niedojrzałości biznesu lub też niedociągnięć procesowych prowadzących do niedbałych projektów, kiepskich praktyk inżynieryjnych i braku testowania. Tego typu dług może zostać wyeliminowany przez odpowiednie szkolenia, dobre zrozumienie sposobów stosowania umiejętności technicznych i podejmowanie odpowiedzialnych decyzji biznesowych. Ze względu na mechanizm powstawania tego dłużu — często na zasadzie przypadkowości i w wyniku braku odpowiedzialności — nazywam go **natywnym długiem technicznym**. Ma on również inne określenia: dług lekkomyślny (ang. *reckless debt* [Fowler, 2009]), dług mimowolny (ang. *unintentional debt* [McConnell, 2007]) i bałagan (ang. *mess* [Martin, 2008]).

Oprócz tego istnieje również **nieunikniony dług techniczny**, którego zazwyczaj nie można przewidzieć lub też zapobiec mu. Na przykład nasze pojmowanie tego, co jest dobrym projektem, powstaje w wyniku prowadzenia prac projektowych, a następnie budowania na tej podstawie cech mających wartość dla użytkownika. Nie jesteśmy w stanie w sposób nieomylny z góry przewidzieć, jak powinien ewoluować nasz produkt i jego projekt. W związku z tym podejmowane na samym początku decyzje projektowe i implementacyjne będą musiały zostać zmienione w miarę domykania istotnych pętli zdobywania wiedzy i pozyskiwania wiedzy potwierdzonej. Potrzebne zmiany w dotkniętych obszarach są nieuniknionym długiem technicznym.

Oto kolejny przykład: powiedzmy, że wykupiliśmy licencję na komponent innego producenta i użyliśmy go w naszym produkcie. Wraz z upływem czasu interfejsy tego komponentu zmieniają się. Nasz produkt, który kiedyś funkcjonował zupełnie dobrze z kupionym komponentem, zaczyna akumulować dług techniczny nie z naszej winy. Chociaż taki dług można przewidzieć (założenie, że dostawca komponentu zmieni jego interfejsy, ma jak najbardziej racjonalne podstawy), nie można mu zapobiec, ponieważ nie jesteśmy w stanie przewidzieć, w jakim kierunku programiści tego komponentu poprowadzą go w przyszłości.

Ostatni typ dłużu to **strategiczny dług techniczny**. Ten typ dłużu jest narzędziem, które może pomóc firmom wskazać ważne decyzje ekonomiczne, często związane z presją czasu, i skorzystać na ich podjęciu. Na przykład organizacja może celowo podjąć strategiczną decyzję pójścia na skróty w trakcie produkcji tylko po to, aby osiągnąć istotny cel krótkoterminowy, taki jak wprowadzenie

na rynek produktu w krytycznym momencie czasu. Z kolei firmy o ograniczonym kapitale, stojące w obliczu utraty wszelkich środków finansowych przed zakończeniem projektu, mogą zdecydować się na początkową sprzedaż produktu z długiem technicznym (wynikającym z braku środków na dalsze prace deweloperskie). Takie podejście pozwoli na generowanie zysków, kontynuację prac deweloperskich i uniknięcie (nieuchronnego w innej sytuacji) upadku przed wypuszczeniem czegokolwiek na rynek.

Niezależnie od sposobu nabycia dług techniczny jest bardzo funkcjonalną metaforą, ponieważ podnosi świadomość i zwraca uwagę na bardzo istotny problem. Pojęcie dłużu trafia doskonale do ludzi biznesu, którzy przeważnie znają się doskonale na długach finansowych. Kiedy słyszą hasło *dług techniczny*, bardzo szybko przyswajają obrazowe porównania, z których najistotniejsze mówi, że podobnie jak w przypadku dłużu finansowego dług techniczny wymaga spłaty odsetek, wyrażonych w formie dodatkowej pracy deweloperskiej, jaką trzeba będzie podjąć w przyszłości. Możemy zdecydować się na dalszą spłatę odsetek (przez znajdywanie obejść problemów) lub spłacić zadłużenie zasadnicze (na przykład przez **refaktoryzację** kodu, czyli doprowadzenie go do bardziej przejrzystszej i łatwiejszej do modyfikowania postaci).

## Konsekwencje dłużu technicznego

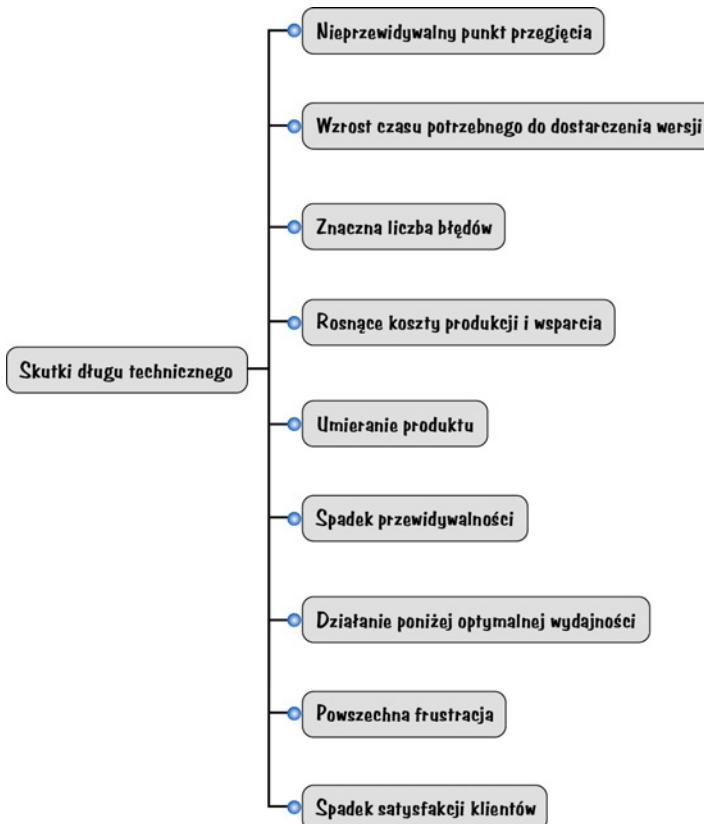
Wraz ze wzrostem poziomu dłużu technicznego rośnie dotkliwość związanych z nim konsekwencji. Przyjrzyjmy się kilku z istotnych skutków wysokiego poziomu dłużu technicznego (podsumowanych na rysunku 8.1).

### Nieprzewidywalny punkt przegięcia

Istotnym atrybutem dłużu technicznego jest to, że przyrasta w sposób nieprzewidywalny i nielinowy. Każda mała porcja dłużu technicznego po dodaniu do całego zbioru dłużu już istniejącego może uczynić znacznie więcej szkody, niż należałoby się spodziewać na podstawie rozmiaru wyłącznie nowego dłużu. W pewnym momencie dług techniczny osiąga „masę krytyczną”, a produkt przekracza punkt przegięcia, stając się niemożliwym do zarządzania lub wręcz chaotycznym. W punkcie przegięcia nawet małe zmiany w produkcie stają się okazją do powstania wielkiej niepewności. Ta nieliniowa charakterystyka stanowi poważne ryzyko biznesowe. Nie jesteśmy w stanie powiedzieć, w którym momencie dołożenie małego kamyczka do stosu spowoduje lawinę, ale kiedy tak się w końcu stanie, konsekwencje będą znacznie wzmacnione.

### Wzrost czasu potrzebnego do dostarczenia wersji

Przyjęcie dłużu technicznego oznacza zaciągnięcie dzisiaj pożyczki kosztem czasu potrzebnego do wykonania przyszłej pracy. Im większy dług mamy dzisiaj, tym mniejszą prędkość będziemy mieć jutro. Wraz ze spadkiem prędkości coraz więcej czasu zajmuje dostarczanie nowych cech oraz po prawek produktu dla klientów. Zatem obecność dłużu technicznego powoduje wzrost, a nie spadek odstępów czasu pomiędzy kolejnymi wersjami dystrybucyjnymi. W środowisku nieustannej konkurencji na rynku dług techniczny wpływa niekorzystnie na powodzenie naszego biznesu.



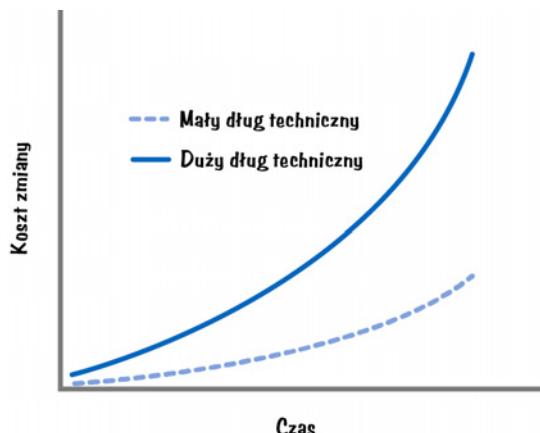
**RYSUNEK 8.1.** Skutki dłużego technicznego

## Znacząca liczba błędów

Produkty obarczone znaczną ilością dłużego technicznego stają się skomplikowane, co utrudnia rozwiązywanie ich w sposób poprawny. Narastająca liczba błędów może być źródłem krytycznych awarii produktu występujących z alarmującą częstotliwością. Te awarie w istotny sposób zakłócają normalny przepływ pracy deweloperskiej dostarczającej nową wartość dodaną. Narzut pracy wynikający z konieczności zarządzania dużą liczbą błędów zabiera lwią część czasu przeznaczonego na wytwarzanie nowych cech produktu. W pewnym momencie zaczynamy tonąć, ale jesteśmy tak zacięci próbą przedostania się przez obszar błędów, że przestajemy dostrzegać możliwość wydostania się z bałaganu, w którym się znaleźliśmy.

## Rosnące koszty produkcji i wsparcia

Wraz ze wzrostem dłużego technicznego zaczynają rosnąć koszty prac deweloperskich i wsparcia użytkowników. To, co kiedyś było rzeczą prostą i tanią w realizacji, teraz staje się skomplikowane i drogie. W obecności rosnącego poziomu dłużego technicznego nawet małe zmiany stają się bardzo kosztowne (patrz rysunek 8.2).



**RYSUNEK 8.2.** Krzywa kosztów zmiany zniekształcona przez dług techniczny

Kiedy krzywa dłużu technicznego na rysunku 8.2 zaczyna wzrosnąć się ostro do góry, osiągamy masę krytyczną dłużu technicznego i dochodzimy do punktu przegięcia.

Ponadto rosnące koszty mogą wpływać na decyzje o wykonaniu nowej cechy lub naprawieniu błędu. Cecha, którą można by zbudować (lub błęd, który można by naprawić) niskim kosztem w warunkach niskiego dłużu technicznego, może okazać się zbyt droga przy wysokim poziomie dłużu technicznego. W wyniku rosnących kosztów nasze produkty stają się mniej adaptywne do ewoluującego środowiska, w którym muszą funkcjonować.

## Umieranie produktu

Zaprzestanie dodawania nowych cech produktu lub naprawiania błędów pozwalających na rewitalizację naszego starzejącego się produktu sprawia, że staje się on coraz mniej atrakcyjny dla aktualnych i potencjalnych klientów. Efektem tego jest stopniowe starzenie się produktu i dryfowanie do stanu, w którym przestaje on być opcją opłacalną dla większości użytkowników. Ci, którzy trzymają się produktu, robią to tylko do pewnego momentu, czekając jedynie na okazję do przełączenia się na inne rozwiązanie.

## Spadek przewidywalności

Próba jakiegokolwiek przewidywania w przypadku produktu obarczonego dużym poziomem dłużu technicznego jest niemal niemożliwa. Przykładem jest nadawanie błędnych ocen nawet przez najbardziej doświadczonych członków zespołu. Stwierdzenie, jak długo może potrwać realizacja czegokolwiek w produkcie pełnym dłużu technicznego, jest zwyczajnie związane ze zbyt dużą dozą niepewności. Konsekwencją tego jest poważne osłabienie naszych możliwości zobowiązania się do wykonania pracy i faktycznego osiągania założonego celu. Strona biznesowa przestaje ufać w to, co deweloperzy mają do powiedzenia, a klienci przestają wierzyć w wypowiedzi marketingu i sprzedaważy!

## Działanie poniżej optymalnej wydajności

Z przykrością należy stwierdzić, że wraz ze wzrostem dłużu technicznego ludzie redukują swoje oczekiwania wobec możliwej wydajności produkcji i tym samym zmniejszają swoje oczekiwania wobec tego, co da się zrobić. Ten spadek oczekiwania zaczyna się rozprzestrzeniać poprzez łańcuch dostarczania wartości, skutkując ogólnym zmniejszeniem wydajności w całej organizacji.

## Powszechna frustracja

Niefortunną konsekwencją dużego dłużu technicznego jest frustracja wśród osób znajdujących się w łańcuchu dostarczania wartości. Kumulacja wszystkich tych małych, ale uprzykrzających życie skrótów sprawia, że praca nad produktem staje się bardzo męcząca. W końcu ginie wszelka radość z pracy deweloperskiej, a w jej miejsce pojawia się codzienna męczarnia zwalczania problemów, z którymi nikt nie chce mieć do czynienia (nawet jeśli powinien). Ludzie się wypalają. Świadomi sytuacji członkowie zespołu deweloperskiego odchodzą w poszukiwaniu innych, bardziej satysfakcyjnych zajęć. Ponieważ to właśnie oni nadają się najlepiej do rozwiązywania problemu z dłużem technicznym, ich odejście jeszcze bardziej pogarsza sprawę dla tych, którzy zostają na miejscu. Spadek morale zaczyna przybierać jeszcze bardziej na sile.

Dług techniczny wysysa radość pracy nie tylko z inżynierów — taki sam wpływ ma na ludzi ze strony biznesowej. Jak długo jesteśmy w stanie podejmować zobowiązania biznesowe, których nie da się osiągnąć? I co z naszymi biednymi klientami, którzy usiłują rozwijać swój interes w oparciu o nasz przesiąknięty dłużem produkt? Oni również bardzo szybko zmęczą się nieustannymi awariami systemu i naszą niemożnością wypełniania jakichkolwiek składanych przez nas obietnic. Zaufanie, jakie kiedyś istniało w łańcuchu dostarczania wartości, zostaje zastąpione frustracją i wzajemnymi urazami.

## Spadek satysfakcji klientów

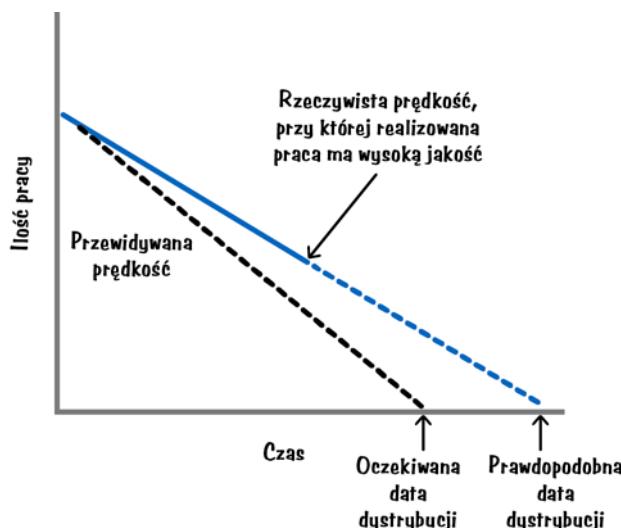
Wzrost frustracji wśród klientów spowoduje spadek ich satysfakcji. Zatem zasięg zniszczeń spowodowanych przez dług techniczny nie będzie ograniczony jedynie do zespołu deweloperskiego lub do samego działu produkcyjnego firmy. Co gorsza, konsekwencje dłużu technicznego mogą poważnie dotknąć naszych klientów i ich sposobu postrzegania nas.

## Przyczyny dłużu technicznego

Przypomnijmy sobie, iż dług techniczny występuje w trzech formach, z których każda ma inne przyczyny. Nieunikniony dług techniczny pojawia się niezależnie od przyjętych przez nas środków zaradczych. Źródłem natywnego dłużu technicznego może być członek zespołu, organizacja działań lub (i) niedojrzałość procesu. Dług strategiczny z kolei jest czymś, co godzimy się przyjąć w sytuacji, kiedy koszt dłużu jest nieporównywalnie mniejszy od potencjalnych zysków.

## **Presja nieprzekroczenia terminu końcowego**

Zarówno dług strategiczny, jak i techniczny są często generowane w następstwie biznesowej presji nieprzekraczania ważnego, zbliżającego się terminu końcowego (patrz rysunek 8.3, w oparciu o [Mar, 2006]).



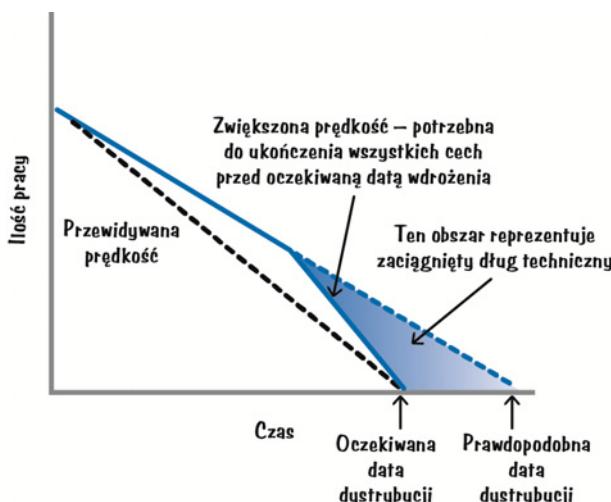
**RYSUNEK 8.3.** Presja osiągnięcia terminu końcowego powoduje powstawanie dłużu technicznego

Oś pionowa na wykresie z rysunku 8.3 reprezentuje ilość pracy, jaką chcemy zrealizować przed wyznaczoną datą dystrybucji (pokazaną na osi poziomej). Linia pomiędzy ilością pracy a oczekiwanej datą dystrybucji oznacza stałą przewidywaną prędkość, z jaką praca musi być realizowana, aby zmieścić się w czasie. Pracując z założoną prędkością, chcemy ukończyć cechy przy zachowaniu wysokiej jakości i jednocześnie pracować w sposób komfortowy pod względem czasu i minimalizując przyrost dłużu technicznego.

Po rozpoczęciu pracy okazuje się, że faktyczna prędkość potrzebna do wytwarzania wyników o odpowiednio wysokiej jakości jest mniejsza od zakładanej. Jeżeli będziemy kontynuować produkcję z rzeczywistą prędkością, nie zmieścimy się w czasie przed oczekianą datą dystrybucji i najprawdopodobniej skończymy w prawdopodobnym dniu dystrybucji.

## Próby sztucznego zwiększenia prędkości

W tym momencie musimy podjąć decyzję biznesową. Czy chcemy zmniejszyć zawartość wersji, aby zmieścić się w czasie, czy też dodamy więcej czasu do terminarza, aby umożliwić wypuszczenie niezmienionego zakresu funkcjonalności w prawdopodobnym dniu dystrybucji? Niestety, w wielu przypadkach ludzie odpowiedzialni za stronę biznesową odrzucają obie opcje i decydują, iż zespół musi wykonać wszystkie przewidziane cechy przed oczekiwany dniem wdrożenia. W takiej sytuacji zespół realizujący prace dostaje polecenie zwiększenia swojej prędkości (patrz rysunek 8.4).



**RYSUNEK 8.4.** Akumulowanie dlużu technicznego w celu zmieszczenia się w czasie przed nieracjonalną datą wdrożenia z ustalonym zakresem

Zespół pracujący ze zwiększoną prędkością będzie musiał świadomie zgadzać się na generowanie dlużu technicznego (czyli mówiąc inaczej, rezygnować z jakości, aby realizować zadania z szybkością pozwalającą na zmieszczenie się w czasie przed zaplanowaną datą wdrożenia). Być może projekt nie będzie taki dobry, jak mógłby być, lub pewne rodzaje testowania (na przykład sprawdzanie obciążenia aplikacji) zostaną odłożone na później. W wyniku zaciągnięcia dlużu technicznego, tak jak przedstawia to trójkątny obszar na rysunku 8.4. Ten region reprezentuje całą pracę, którą powinniśmy byli zrobić, ale nie mieliśmy na nią czasu.

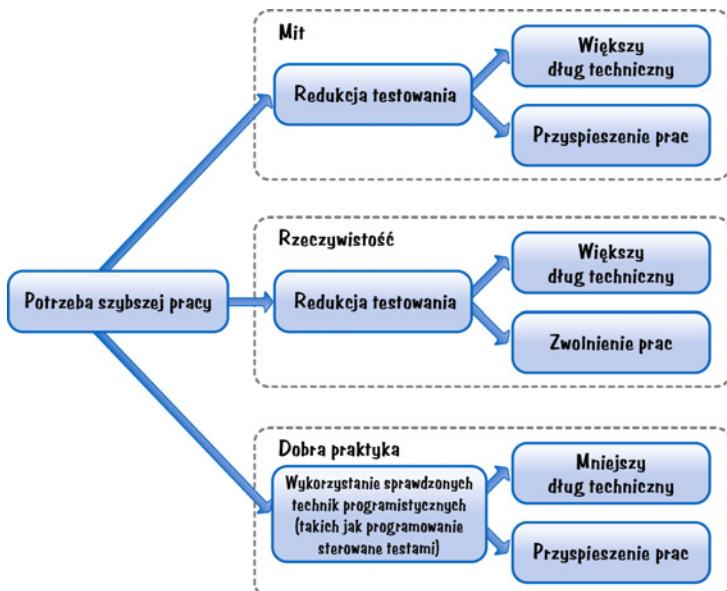
### Mit: Rezygnacja z testowania zwiększa prędkość

Według obiegowego mitu testowanie wprowadza nadmiarowy nakład pracy — redukując go, możemy zwiększyć prędkość (patrz rysunek 8.5).

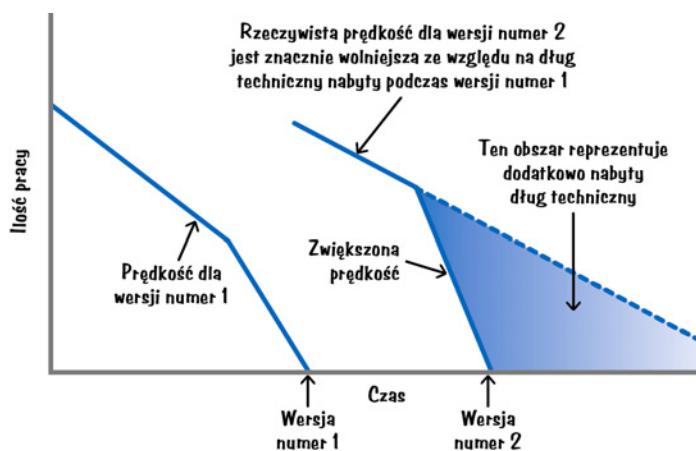
W rzeczywistości redukcja zadań związanych z testowaniem zwiększy poziom dlużu technicznego, jak również sprawi, że będziemy działać wolniej, ponieważ pewne problemy będą następować w sposób niezauważalny, a kiedy w końcu zostaną odkryte, ich naprawienie będzie o wiele bardziej czasochłonne. Doświadczone zespoły dostarczają wyników o dobrej jakości w szybszym czasie i przy mniejszym poziomie dlużu technicznego, kiedy testowanie jest nieodłączną częścią procesu deweloperskiego. Tego typu zespoły stosują dobre techniki pracy, takie jak **programowanie sterowane testami** polegające na pisaniu przez programistę małego automatycznego testu jednostkowego przed napisaniem małego kawałka kodu, który sprawi, że ten test przejdzie [Crispin i Gregory, 2009].

### Dług narasta na dlużu już istniejącym

Nowy dług techniczny narasta szybko na dlużu już istniejącym, a wraz z jego narastaniem zaczynają się pojawiać szkodliwe skutki ekonomiczne. Przykład konsekwencji budowania wersji dystrybucyjnej numer 2 na szczytce dlużu technicznego wersji dystrybucyjnej numer 1 pokazuje rysunek 8.6.



**RYSUNEK 8.5.** Mit, rzeczywistość oraz dobra praktyka wpływu testowania na prędkość



**RYSUNEK 8.6.** Wraz ze wzrostem dłużu technicznego maleje prędkość

Na rysunku tym widać, że prędkość podczas realizacji wersji numer 2 jest mniejsza od tej z wersji numer 1. Oczywiście wydaje się, że przy takiej prędkości ponownie nie trafimy w planowaną datę wdrożenia, a mimo to kadra zarządzająca ponownie oczekuje od zespołu realizacji wszystkich cech w ustalonym czasie. Wynikiem jest akumulacja jeszcze większej ilości dłużu technicznego.

Jeżeli taki scenariusz będzie się powtarzał, przedżej czy później linia prędkości zacznie biec równolegle do osi czasu. Będzie to taki stan, w którym ogrom dłużu technicznego sprawia, że nasza faktyczna prędkość jest równa zero. Otrzymujemy tym samym produkt, w którym boimy się dokonać jakichkolwiek zmian, ponieważ malutka zmiana w jednym miejscu może doprowadzić do popsu 18 innych miejsc, które na pierwszy rzut oka wydają się być zupełnie niezwiązany

ze sobą obszarami produktu. Co gorsza, nie ma możliwości, abyśmy byli w stanie przewidzieć, że problemy mogą pojawić się właśnie w tych 18 miejscach. Oczywiście nie dysponujemy żadnym znaczącym zestawem testów, który pomogłby nam wskazać popsułe miejsca, ale czym tu się marzyć — nasi klienci na pewno wskażą te miejsca!

Kiedy znajdziemy się w sytuacji wysokiego poziomu dłużu technicznego, każda decyzja, jaką podejmiemy, będzie zła. Możemy wtedy:

- nie robić nic, wtedy problem jeszcze bardziej przybierze na sile;
- zwiększać inwestycje w redukcję dłużu technicznego, co będzie się wiązać z pochłanianiem coraz większej ilości naszych cennych zasobów deweloperskich;
- ogłosić upadłość techniczną, zaprzestać spłaty istniejącego dłużu technicznego, zastępując produkt nim obarczony nowym produktem (co wiąże się z poniesieniem pełnego kosztu i ryzykiem wytworzenia takiego produktu).

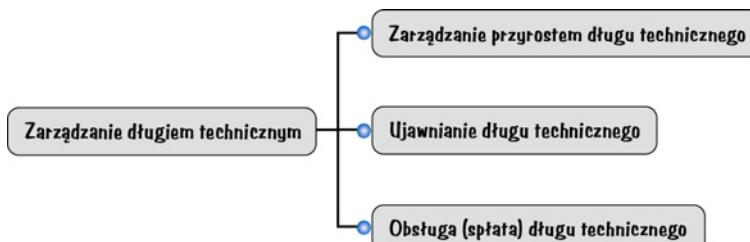
Gdy ma się na horyzoncie tego typu wybory, kluczowego znaczenia nabiera prawidłowe zarządzanie dłużem technicznym, zanim ten wydostanie się spod kontroli.

## Dłużem technicznym trzeba zarządzać

Dług techniczny, podobnie jak dług finansowy, musi być zarządzany. Trzeba sobie uzmysłowić, że nie istnieje produkt pozbawiony dłużu technicznego, dlatego nie proponuję, abyś próbował osiągnąć stan zerowego dłużu. Zakładając nawet istnienie możliwości osiągnięcia takiego stanu, prawdopodobnie byłoby to nieuzasadnione z ekonomicznego punktu widzenia. Zamiast tego powinieneś utrzymywać poziom dłużu technicznego na dostatecznie niskim poziomie, tak aby nie wpływał on znacząco do przyszłą produkcję.

Zarządzanie dłużem technicznym wymaga wyważonej dyskusji pomiędzy członkami zespołu deweloperskiego i osobami zaangażowanymi w stronę biznesową. Jest to między innymi jedna z przyczyn istnienia w każdym zespole właściciela produktu. Obecność właściciela produktu w zespole scrumowym pozwala na przeprowadzenie dyskusji uwzględniającej w równym stopniu techniczne, jak i biznesowe punkty widzenia, co umożliwia osiągnięcie uzasadnionego ekonomicznie kompromisu. Bardzo istotne jest zatem (o czym będę pisał w rozdziale 9.), abyśmy wybrali na właściciela produktu osobę z odpowiednią wiedzą biznesową, umożliwiającą jej aktywny udział w prowadzonych dyskusjach.

Zarządzanie dłużem technicznym sprowadza się do trzech zasadniczych aktywności (patrz rysunek 8.7). Każdej z nich poświęć oddzielną sekcję.



**RYSUNEK 8.7.** Aktywności zarządzania dłużem technicznym

## Zarządzanie przyrostem dłużu technicznego

Absolutnie krytycznym aspektem zarządzania dłużem technicznym jest kierowanie procesem jego przyrastania. Jak pisałem wcześniej, istnieje pewna granica dłużu, jaki jesteśmy w stanie przyjąć, zanim osiągniemy masę krytyczną. Szukając analogii, można powiedzieć, że nieustanne gromadzenie dłużu technicznego jest odpowiednikiem ciągłego zadłużania się na hipotece własnego domu lub mieszkania. W pewnym momencie musimy powiedzieć „Dość!”, ponieważ konsekwencje mogą być tragiczne.

Po pierwsze, musimy przestać dodawać dług natywny do naszych produktów (skończyć z lekkomyślnością i tworzeniem bałaganu). Musimy również uświadomić sobie, że istnieje pewna dopuszczalna granica strategicznego lub nieuniknionego dłużu technicznego, jakiej nie wolno nam przekroczyć, jeśli nie chcemy dotrzeć do punktu przegięcia. Omówię sposoby radzenia sobie z tymi dwoma typami dłużu. Nie będę wnikał w zarządzanie przyrostem dłużu nieuniknionego, ponieważ z samej jego natury wynika brak możliwości przeciwdziałania mu (możemy jednak ujawnić go, a następnie obsłużyć).

### ***Stosowanie dobrych praktyk technicznych***

Pierwszym sposobem radzenia sobie z przyrastaniem dłużu technicznego jest zaprzestanie dodawania natywnego dłużu technicznego do naszych produktów. Doskonałym punktem startowym na drodze w tym kierunku jest wykorzystanie dobrych praktyk technicznych. Choć Scrum nie definiuje formalnie **praktyk technicznych**, każdy spotkany przeze mnie pomyślnie działający zespół scrumowy stosuje praktyki w postaci prostego projektu, programowania sterowanego testami, **ciągłej integracji**, testowania automatycznego, refaktoryzacji itd. (omówienie tego tematu znajdziesz w rozdziale 20.). Rozumienie tych praktyk i ich aktywne wykorzystanie pomaga zespołowi w zaprzestaniu dodawania różnych form natywnego dłużu technicznego do swoich produktów.

Istotnym narzędziem pozwalającym spłacić nabity już dług techniczny jest refaktoryzacja kodu. Polega ona na zmianie struktury istniejącego kodu, ale z zachowaniem istniejącego działania [Fowler i in., 1999]. Innymi słowy, czyścimy wnętrzności, ale z punktu widzenia klienta produkt nadal działa tak samo. Refaktoryzacja ma na celu zredukowanie złożoności programu, co ułatwia jego utrzymywanie i rozszerzanie. Wynikiem refaktoryzacji ma być ułatwienie bieżącej pracy (odpowiednik redukcji odsetek od zadłużenia).

Cunningham [2011] wyjaśnia zalety refaktoryzacji w oparciu o przykład:

*... klient jest gotowy zapłacić za nową cechę, ale cecha ta jest niemożliwa do umiejscowienia w aktualnym kodzie. Zreorganizuj kod, dzięki czemu cecha będzie możliwa do umieszczenia w kodzie. Teraz jej implementacja powinna być łatwa. Można to nazwać refaktoryzacją w samą porę. Działanie takie wyjaśniłbym kadrze zarządzającej w następujący sposób: pragniemy, aby w naszym oprogramowaniu było miejsce na każde nowe żądanie. Czasem jednak nie mamy miejsca na nową cechę i w związku z tym musimy je najpierw przygotować, a dopiero potem zaimplementować daną cechę...*

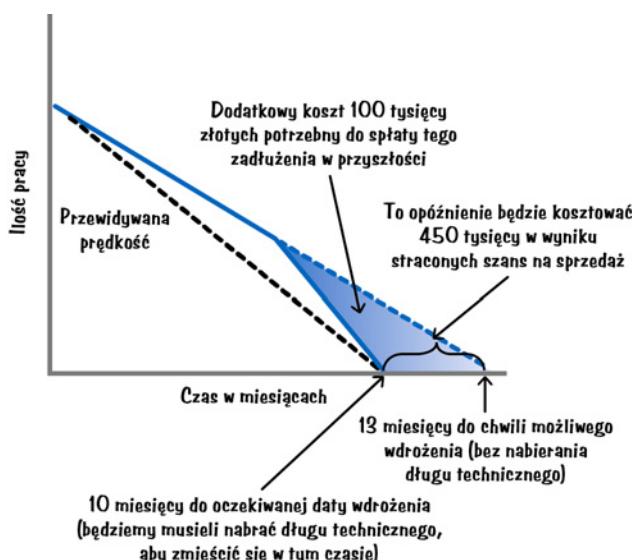
## Solidna definicja ukończenia

Praca, którą powinniśmy wykonać podczas tworzenia nowej cechy, a którą odłożyliśmy na później, jest istotnym źródłem dłużu technicznego. Stosując Scrum, chcemy mieć solidną definicję ukończenia (patrz rozdział 4.), która pomoże zespołowi kończyć pracę w każdym sprintie z niskim lub wręcz zerowym poziomem dłużu.

Im nasza lista kontrolna definicji ukończenia będzie bardziej szczegółowa pod względem technicznym, tym mniejsze będą szanse nabierania przez nas dłużu technicznego. W wielu przypadkach — o czym pisałem w rozdziale 2. — koszt spłaty dłużu technicznego, który przedostaje się do produktu przez słabą definicję ukończenia, jest o wiele większy od kosztów radzenia sobie z nim w trakcie sprintu. Działanie bez solidnej definicji ukończenia jest zaproszeniem do akumulowania dłużu technicznego.

## Prawidłowe rozumienie ekonomii dłużu technicznego

Chcąc używać dłużu technicznego w sposób strategiczny i przynoszący nam zysk, musimy dobrze zrozumieć jego wpływ na ekonomiczność podejmowanych przez nas decyzji. Z przykrością trzeba stwierdzić, że większość organizacji nie rozumie skutków dłużu technicznego w dostatecznie dobrym stopniu, aby móc prawidłowo ocenić ekonomiczny efekt jego przyjęcia. Zilustrujmy to na przykładzie (patrz rysunek 8.8).



**RYSUNEK 8.8.** Przykład analizy ekonomiczności dłużu technicznego

W tym przykładzie przyjmiemy następujące założenia:

- każdy miesiąc prac deweloperskich kosztuje 100 tysięcy złotych;
- ilość obowiązkowych cech jest niemożliwa do zrealizowania w sposób rozsądny przed oczekwaną datą wdrożenia (10 miesięcy od teraz);
- porzucenie jakichkolwiek cech jest niemożliwe.

Rozważmy dwa alternatywne scenariusze. W pierwszym opóźniamy datę wdrożenia o trzy miesiące, dzięki czemu jesteśmy w stanie rozsądnie i profesjonalnie ukończyć prace nad obowiązkowymi cechami produktu przy minimalnym poziomie dłużu po 13 miesiącach. Całkowity koszt prac deweloperskich wyniesie 1,3 miliona złotych. Rozmawiając z działami marketingu i sprzedaży, prognozujemy również, iż trzymiesięczne opóźnienie spowoduje stratę rzędu 450 tysięcy złotych, wynikającą z potencjalnej możliwości sprzedaży produktu w tym czasie.

W drugim przypadku przyśpieszamy prace deweloperskie bez wprowadzenia skrótów w celu zmieszczenia się z wdrożeniem w okresie 10 miesięcy. Aby prawidłowo ocenić ekonomię takiej opcji, musimy znać koszt przyjęcia dłużu technicznego.

Tutaj sytuacja komplikuje się. Wyobraź sobie, że zadajemy zespołowi deweloperskiemu następujące pytanie: „Zatem jeśli dzisiaj będziecie musieli zdecydować się na pewne cięcia przy projekcie i implementacji, aby zrealizować obowiązkowe cechy przed oryginalnie założoną datą wdrożenia, ile będzie kosztowało spłacenie nabranego dłużu po wypuszczeniu pierwszej wersji?”.

Powiedzmy, że zespół przedyskutował to pytanie i w jego opinii wyczyszczenie systemu z problemów zajmie cztery miesiące dodatkowej pracy. To oznacza, że zespół będzie potrzebował dodatkowego miesiąca ponad te trzy, które „zaoszczędził”, przycinając prace na początku. Czyli ostatecznie zespół wyda dodatkowe 100 tysięcy złotych na prace deweloperskie (razem 1,4 miliona złotych zamiast 1,3 miliona w przypadku pierwszej opcji). To dodatkowe 100 tysięcy, których organizacja nie musiałaby wydawać, gdyby poświęciła czas na wykonanie pracy tak jak trzeba za pierwszym razem bez umieszczania dłużu technicznego w produkcie.

Patrząc powierzchownie, decyzja wydaje się oczywista. Czy powinniśmy zaciągać dłuż techniczny rzędu 100 tysięcy złotych, aby wygenerować dodatkowy zysk 450 tysięcy złotych? Oczywiście, kto by tego nie chciał? To może być prawidłowa decyzja, jeśli uważamy, że rozważyliśmy wszystkie ważne czynniki związane z dłużem technicznym (lub przynajmniej większość z nich).

Poniżej znajdują się jednak dwa z wielu możliwych czynników, których w ogóle nie wzięliśmy pod uwagę:

- Co z kosztem opóźnienia wynikającym z konieczności spłaty dłużu technicznego? 100 tysięcy złotych pokryje prace zespołu związane z redukcją dłużu w przyszłości. A co z kosztem czasu potrzebnym do redukcji zadłużenia? Czas spędzony na spłacie zadłużenia wprowadza koszt opóźnienia prac nad innym produktem lub kolejną wersją tego samego produktu. Czy znamy koszt tego opóźnienia? Jeśli zespół spędza miesiąc na odrobieniu dłużu, wypuszczenie innego produktu będzie najprawdopodobniej opóźnione o jeden miesiąc. Ten koszt straconych szans ma rzeczywistą ekonomiczną wartość, którą należy wziąć pod uwagę.
- Większość organizacji nie radzi sobie dobrze ze spłatą swojego własnego dłużu technicznego. Kiedy robi się gorąco, zarząd firmy woli, aby prace skupiły się na nowych cechach, a nie na poprawianiu rzeczy już istniejących. Lądujemy zatem w rzeczywistości, w której być może nigdy nie spłacimy żadnej części zaciągniętego dłużu, co oznacza, że będziemy spłacać odsetki od tego dłużu przez cały okres życia naszego systemu. To również trzeba rozważyć.

Tabela 8.1 przedstawia liczbowe podsumowanie powyższego przykładu.

**TABELA 8.1.** Przykład porównania ekonomii unikania dłużu technicznego z ekonomicznością akumulowania dłużu

	Unikanie dłużu	Przyjmowanie dłużu
Miesięczny koszt prac deweloperskich	100 000 zł	100 000 zł
Całkowita liczba miesięcy prac deweloperskich	13	10
Całkowity koszt prac deweloperskich	1,3 mln zł	1,0 mln zł
Opóźnienie w miesiącach (w celu wypuszczenia produktu)	3	0
Koszt opóźnienia na każdy miesiąc	150 000 zł	150 000 zł
Całkowity koszt opóźnienia	450 000 zł	0
Liczba miesięcy potrzebna na obsłużenie dłużu	0	4
Koszt obsłużenia dłużu	0 zł	400 000 zł
Całkowity koszt w zyskach z cyklu życia produktu	1,75 mln zł	1,4 mln zł
Koszt opóźnienia wynikający z narastającej ilości czasu potrzebnej do spłaty dłużu	0 zł	X
Dożywotnie odsetki od dłużu technicznego	0 zł	Y
Inne koszty związane z dłużem technicznym	0 zł	Z
Rzeczywisty koszt w zyskach z cyklu życia produktu	1,75 mln zł	1,4 mln zł + X + Y + Z

Widać wyraźnie, że „macki” dłużu technicznego są dłużie i wpływają na szereg aspektów całosciowej kalkulacji ekonomicznej. Unikanie rozważania przynajmniej najistotniejszych czynników w tym zakresie sprawi, że nie będziemy w stanie poprawnie ocenić skutków ekonomicznych przyjęcia dłużu technicznego.

Oczywiście, jeżeli wyliczenia ekonomiczne na korzyść przyjęcia dłużu technicznego są bezsporne i przekonujące — na przykład powstrzymanie się od przyjęcia dłużu technicznego i szybkiego wypuszczenia produktu na rynek ze wszystkimi obowiązkowymi cechami spowoduje, że nasz biznes upadnie lub ominie nas lwna część zysków wynikająca z bycia pierwszym na rynku — nie musimy poświęcać dodatkowego czasu na rozważanie mniej istotnych czynników, ponieważ już wiemy, że przyjęcie dłużu technicznego jest ekonomicznie uzasadnione.

Częściej decyzja nie jest jednak tak oczywista. Wybór przyjęcia dłużu technicznego wymaga zazwyczaj dogłębnej analizy pozwalającej na dostrzeżenie, która z opcji jest lepsza. Mniejsze ryzyko błędu leży po stronie nieprzyjmowania dłużu technicznego. Z mojego doświadczenia wynika, iż większość organizacji mocno nie docenia prawdziwego kosztu dłużu technicznego, a także nie jest tak rygorystyczna pod względem jego spłaty, jak początkowo zakładała.

## Ujawnianie dłużu technicznego

Jednym z kluczowych zysków metafory dłużu technicznego jest możliwość wykorzystania jej do osiągnięcia obopólnie rozumianego kontekstu przez zespół deweloperski oraz ludzi zarządzających przedsięwzięciem podczas niezbędnych rozmów. Prowadzenie takich rozmów wymaga widzenia przez obie strony dłużu technicznego produktu w zrozumiały dla siebie sposób.

## Ujawniaj dług techniczny na poziomie biznesowym

Problemem wielu organizacji jest to, że poziom dłużu technicznego produktu jest dobrze widoczny dla zespołu deweloperskiego, ale nie dla kadry zarządzającej. Wystarczy zapytać dowolnego inżyniera znającego dany produkt o to, gdzie skoncentrowana jest największa ilość dłużu technicznego, a z pewnością uzyskamy szybką odpowiedź. Zadajmy to samo pytanie menedżerowi wyższego rzędu, a najprawdopodobniej przekonamy się, że nie ma on odpowiedniej wiedzy na temat wielkości dłużu, a także tego, jakiego typu jest to dług.

Zupełnie inaczej sprawa ma się z dłużem finansowym. Menedżer zapytany o deficyt finansowy organizacji będzie w stanie odpowiedzieć bardzo precyzyjnie.

Stąd bardzo duże znaczenie ma uświadamianie menedżerom i dyrektorom poziomu dłużu technicznego produktu. Gdybym był w stanie określić dług techniczny w sposób ilościowy — istnieje w miarę aktualna praca naukowa opisująca, jak można to zrobić [SIE, 2011] — rozważyłbym wprowadzenie wierszy z krótkoterminowym i długoterminowym dłużem technicznym do dokumentów bilansowych firmy zaraz obok dłużu finansowego (patrz tabela 8.2).

**TABELA 8.2.** Dług techniczny wykazany w bilansie finansowym firmy

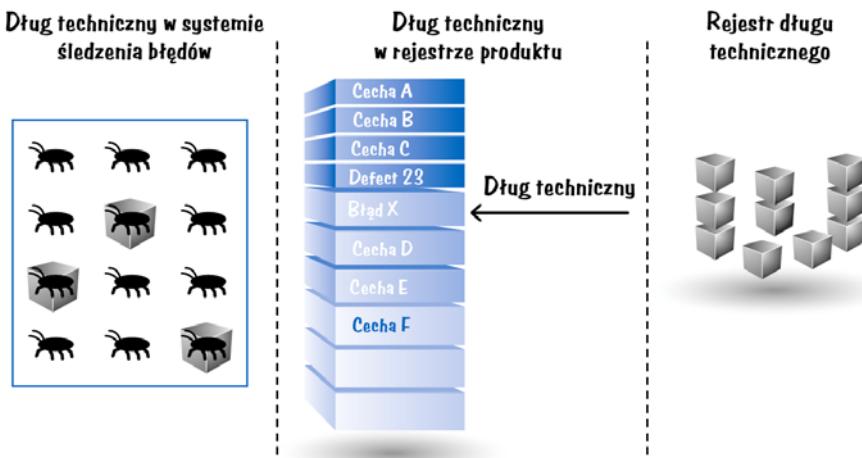
Aktywa	Zobowiązania
Gotówka	600 000 zł
Należności	450 000 zł
	Zobowiązania wekslowe
	75 000 zł
	<b>Krótkoterminowy dług techniczny</b>
Narzędzia i sprzęt	250 000 zł
	Zobowiązania długoterminowe
	300 000 zł
	<b>Długoterminowy dług techniczny</b>
...	...
	...

Nie przypominam sobie żadnej organizacji, która faktycznie wykazywałaby w swoim bilansie krótkoterminowy i długoterminowy dług techniczny (chociaż uważam, że jest to dobry pomysł). Powyższy przykład ma jedynie na celu zilustrowanie potrzeby znalezienia przez każdą organizację sposobu na komunikowanie skali dłużu technicznego w sposób zrozumiały dla osób z kadry zarządzającej. W przeciwnym razie strona biznesowa nie będzie miała odpowiedniego wglądu w stan produktu, aby podejmować świadome ekonomicznie decyzje.

Pewnym sposobem ujawniania konsekwencji dłużu technicznego przez organizacje jest śledzenie prędkości. Rysunek 8.6 pokazuje wpływ wzrostu dłużu technicznego na spadek prędkości. Ten spadek może zostać opisany w sposób finansowy. Założymy na przykład, że dysponujemy zespołem scrumowym, którego koszt działania to 20 tysięcy złotych na sprint, a historia prędkości to 20 punktów na sprint. Znając te wartości, wyliczamy wartość jednego punktu w zespole na tysiąc złotych. Jeżeli przyrost dłużu technicznego powoduje spadek prędkości zespołu o 10 punktów na sprint, wtedy koszt każdego punktu wzrasta do dwóch tysięcy. Jeżeli sumarycznie zespół ma do ukończenia mniej więcej 200 punktów pracy, a jego prędkość spada o połowę, wykonanie pracy kosztującej do tej pory 200 tysięcy będzie teraz kosztować 400 tysięcy. Zatem używając prędkości, widzimy wyraźnie odsetki wypłacane z tytułu dłużu technicznego.

## Ujawniaj dług techniczny na poziomie inżynierijnym

Inżynierowie posiadają zazwyczaj ukrytą wiedzę na temat tego, gdzie w produkcji tkwi najbardziej skandaliczna część dłużu technicznego. Niemniej jednak wiedza ta nie jest dostatecznie widoczna, aby można było ją przeanalizować, przedyskutować i podjąć odpowiednie działania. Rysunek 8.9 ilustruje trzy sposoby ujawniania dłużu technicznego na poziomie inżynierijnym.



**RYSUNEK 8.9.** Sposoby ujawniania dłużu technicznego na poziomie inżynierijnym

Po pierwsze, dług techniczny może być odnotowywany jak błędy w istniejącym systemie śledzenia błędów (lewa strona rysunku 8.9). Zaletą takiego rozwiązania jest umieszczanie dłużu w znanym miejscu przy użyciu znanych narzędzi i technik. Jeżeli informacja o dłużu jest zapisana razem z informacją o błędzie, ważne jest oznaczenie dłużu w taki sposób, aby można było znaleźć go w prosty sposób, ponieważ zespół może zechcieć obsługiwać dług w inny sposób, niż obsługuje błędy (o czym będę pisał już wkrótce).

Innym sposobem ujawniania dłużu technicznego jest tworzenie reprezentujących go elementów w rejestrze produktu (środkowa część rysunku 8.9). W ten sposób istotny dług techniczny będzie równie widoczny co nowe cechy w rejestrze. Zespoły stosują to podejście zazwyczaj, kiedy koszt obsługi dłużu jest wysoki i potrzebne jest zaangażowanie właściciela produktu w podejmowanie decyzji o podziale pracy pomiędzy likwidacją dłużu a dodawaniem nowych cech produktu.

Trzecim sposobem uwidaczniania dłużu technicznego jest stworzenie specjalnego rejestru dłużu technicznego, w którym poszczególne obszary zadłużenia reprezentowane są przez indywidualne elementy (prawa strona rysunku 8.9). Za każdym razem, kiedy nowy dług zostanie odkryty lub wprowadzony do produktu, członek zespołu może stworzyć nowy element w rejestrze dłużu technicznego. Przez ujawnianie elementów dłużu w rejestrze zespół deweloperski nie tylko widzi, na jakim poziomie znajduje się cały dług, ale może również wskazać, kiedy chce się zająć jego stopniową redukcją.

Dla zespołów pracujących w jednym miejscu najprostszym podejściem do wizualizacji dłużu technicznego jest wieszanie karteczek samoprzylepnych, z których każda odpowiada pewnemu elementowi dłużu, na przygotowanej do tego celu tablicy reprezentującej rejestr dłużu. Zazwyczaj tablica taka powieszona byłaby gdzieś w pobliżu rejestru sprintu, dzięki czemu w trakcie planowania sprintu zespół ma wgląd w dług i może rozpatrzyć podjęcie jego naprawy w nadchodzący sprincie (o czym będę pisał w następnej sekcji).

Większość zespołów traktuje dług techniczny w sposób bezceremonialny, przyklejając kolejne karteczki wprost na murze. Niektórzy decydują się natomiast na bardziej dokładną pielęgnację rejestru dłużu poprzez poświęcanie odrobiny czasu na uporządkowanie jego elementów, tak aby uzyskać przybliżoną ocenę wysiłku potrzebnego do zlikwidowania problemów przedstawionych na poszczególnych karteczkach.

## Obsługa dłużu technicznego

Ostatnią aktywnością zarządzania dłużem technicznym jest jego obsługa lub też spłata. Przy obsłudze dłużu uważam za pomocne posługiwanie się trzema następującymi kategoriami dłużu:

- **odkryty dług techniczny** — dług, o którego istnieniu zespół deweloperski nie miał pojęcia do momentu odkrycia go w toku normalnych prac deweloperskich nad produktem. Przykładem może być odkrycie podczas dodawania nowej cechy produktu obejścia wbudowanego w kod przez kogoś, kto całe lata temu odszedł z firmy;
- **znany dług techniczny** — dług znany zespołowi deweloperskiemu, odkryty jedną z technik opisanych wcześniej;
- **celowy dług techniczny** — znany dług techniczny przeznaczony do obsługi przez zespół deweloperski.

Bazując na powyższych kategoriach, stosuję następujący algorytm obsługi dłużu technicznego:

1. Sprawdzić, czy znany dług techniczny powinien być obsłużony (niedługo będę pisał o tym, że nie każdy dług powinien podlegać obsłudze). Jeżeli powinien być obsłużony, przejdź do kroku numer 2.
2. Jeżeli pracujesz nad kodem i odkryłeś nowy dług techniczny, zlikwiduj go. Jeżeli poziom odkrytego dłużu technicznego przekracza pewien dopuszczalny poziom, wyczyść go do tego poziomu. Pozostałą (nieobsłużoną) część odkrytego dłużu przekształć w znany dług techniczny (na przykład tworząc wpis w rejestrze dłużu technicznego).
3. Postaraj się przekształcić pewną część znanego dłużu technicznego w celowy dług techniczny z zamiarem obsłużenia go w nadchodzącym sprincie. Preferuj obsługę znanego dłużu technicznego o najwyższej wartości, będącej w zgodzie z wartościową pracą dla klienta.

Na powyższym algorytmie obsługi dłużu bazują podejścia pokazane na rysunku 8.10.



**RYSUNEK 8.10.** Podejścia do obsługi dłużu technicznego

Opiszę każde z nich, a także wskażę ich zastosowanie w Scrumie.

## Nie każdy dług techniczny powinien być spłacany

Czasami dług techniczny nie powinien być spłacany. Jest to obszar, w którym analogia do dłużu finansowego staje się lekko naciągana. Oczekuje się, że cały dług finansowy zostanie kiedyś spłacony — chociaż wiemy, że nie zawsze tak się dzieje!

Istnieje cały szereg scenariuszy, w przypadku których dług techniczny nie powinien być spłacany. Omówię trzy z nich: produkt zbliżający się do końca swojego cyklu życia, prototyp przeznaczony do porzucenia i produkt zbudowany dla krótkiego cyklu życia.

## Produkt zbliżający się do końca swojego cyklu życia

Jeżeli produkt skumulował dużą ilość dłużu technicznego i zbliża się do końca swojego cyklu życia, wszelkie poważne inwestycje w spłatę dłużu byłyby finansową nieodpowiedzialnością. Jeśli produkt ma małą wartość, powinniśmy raczej przenieść go w stan spoczynku (razem z dłużem) i poświęcić nasze zasoby produktom o wyższej wartości. W przypadku bardzo cennego produktu z dużą ilością dłużu technicznego zbliżającego się do kresu swojego cyklu życia być może sensowniejsze od spłaty dłużu w starym produkcie będzie podjęcie wysokiego ryzyka i kosztu tworzenia nowego produktu.

## Prototyp przeznaczony do porzucenia

Czasami najbardziej ekonomicznym rozwiązaniem jest zgoda na celowe kumulowanie dłużu technicznego ze świadomie podjętym planem niespłacania tego dłużu. Częstym przykładem takiej sytuacji jest produkcja prototypu w celu pozyskania wiedzy, a następnie porzucenie go [Goldbert i Rubin, 1995]. Wartością otrzymaną z takiego prototypu nie jest kod, ale pozyskana przez nas

wiedza potwierdzona [Ries, 2011]. Ponieważ prototyp nie powstaje w celu wdrożenia lub dystrybucji na rynek, najprawdopodobniej obarczony jest mniejszą lub większą ilością dłużu technicznego. Jest to prototyp przeznaczony do porzucenia, zatem nie ma powodu, aby zajmować się spłatą za ciągniętego w nim dłużu. Oczywiście jeśli stworzymy prototyp do wyrzucenia, a następnie zdecydujemy się zachować go i w sposób ewolucyjny przekształcić w produkt, będziemy musieli zacząć od stanu przesiąkniętego dłużem technicznym.

## Produkt zbudowany dla krótkiego cyklu życia

Jeżeli zbudujemy produkt dla bardzo krótkiego cyklu życia, związane z nim wyliczenia ekonomiczne mogą wykazać, że nie warto spłacać dłużu technicznego. Zilustruję tę sytuację interesującym przykładem, jaki napotkałem w późnych latach osiemdziesiątych ubiegłego wieku. Pracowałem w tym czasie dla firmy ParcPlace Systems, lidera na młodym rynku środowisk zorientowanych obiektywnie. Jednocześnie pomagałem kilku znaczącym bankom z Wall Street wdrożyć środowisko deweloperskie oparte na języku Smalltalk. W jednym konkretnym przypadku przyszło mi szkolić zespół, aby pomóc jego członkom w lepszym zrozumieniu technologii zorientowanych obiektywnie i dzięki temu zwiększyć ich produktywność w Smalltalku. Zespół ten wyprodukował właśnie jeden z pierwszych systemów do handlowania instrumentami pochodnymi. Moją pierwszą czynnością po przybyciu na miejsce było poproszenie dyrektora grupy o przejrzenie projektu i implementacji dopiero co ukończonego produktu, który jeszcze nie został wdrożony, chociaż miało to nastąpić wkrótce.

Po dniu spędzionym na analizie architektury i kodu spotkałem się z owym dyrektorem i powiedziałem mu, że ten system pretenduje do miana najbardziej paskudnego pod względem implementacji w Smalltalku spośród wszystkich, jakie widziałem w swoim życiu. Wskazałem liczne problemy w implementacji, które powinny być natychmiast naprawione — w przeciwnym razie ich system (i biznes) czeka ciężki żywot.

W tym momencie dyrektor odpowiedział mi, cytując: „Synu, jeśli wydasz choćby centa na naprawę tego systemu, osobiście wyprowadzę cię na zewnątrz i zastrzelę”. Ta odpowiedź, mówiąc najlagodniej, wprawiła mnie w osłupienie. Odpowiedziałem: „Musisz mi pan zaufać. Ten system został kiepsko zaprojektowany, a jego implementacja wygląda okropnie — w dłuższej perspektywie czasu będziecie mieć z nim ogromne problemy”. On zripostował: „Nie rozumiesz mojego biznesu, tam gdzie działam, wychodząc na rynek z nowym produktem finansowym, osiągamy lwią część zysków w ciągu pierwszych trzech miesięcy. Mniej więcej tyle czasu zajmuje mojej konkurencji wejście na rynek z ich wersją tego samego produktu. Kiedy tak się stanie, najlepszym rozwiązaniem dla mnie jest opuszczenie rynku i stworzenie nowego produktu. Potrzebuję tego systemu jedynie przez trzy miesiące, więc jeśli o mnie chodzi, możesz użyć nawet gumy do żucia i drutu, byle działał. Nie opóźniaj jedynie mojego zwrotu z inwestycji i nie dawaj szansy moim konkurentom na pokonanie mnie w drodze na rynek. Włączamy go tak, jak jest”.

I dokładnie tak uczynili. W ciągu pierwszej godziny działania systemu używający go handlowcy wygenerowali 14 milionów zysku. Osobiście uważałem, że włączając system w tak kruchym stanie, firma podejmuje duże ryzyko, ale patrząc z perspektywy zysków, byłem w błędzie.

Zazwyczaj organizacje nie budują produktów z oczekiwany cyklem życia rzędu trzech miesięcy. Jesteśmy zainteresowani wytwarzaniem czegoś, co będzie na rynku przez dłuższy czas.

## Zastosuj metodę skauta (obsłuż dług, kiedy go napotkasz)

**Skauci** stosują następującą **zasadę**: „Pozostaw zawsze obóz w stanie czystszy niż ten, który zostałeś”. Jeżeli znajdziesz śmieci na ziemi, sprzątnij je, nie zastanawiając się, kto mógł je tam pozostawić. Twoim zadaniem jest intencjonalne poprawienie otoczenia dla następnej grupy obozowiczów. Bob Martin (i inni) w bardzo przystępny sposób wyjaśnił, dlaczego ta reguła ma zastosowanie podczas prac deweloperskich i obsługi dłużu technicznego [Martin, 2008].

Stosując się do tej zasady, kiedy dotycamy naszego produktu, staramy się zawsze chociaż odrobinę poprawić, a nie pogorszyć jego projektu i implementację. Kiedy członek zespołu deweloperskiego pracuje nad danym obszarem produktu i zauważa problem (odkrywa dług techniczny), stara się go naprawić. Nie robi tego wyłącznie ze względu na swoje dobro, chociaż w praktyce tak jest, ale również ze względu na dobro całego zespołu deweloperskiego i organizacji.

Przedstawiony wcześniej algorytm mówił, że staramy się naprawić odkryty dług techniczny do pewnego rozsądniego poziomu. Nie mówimy źle, że zespół powinien naprawić cały odkryty dług techniczny w momencie jego ujawnienia. Taka naprawa może wymagać znaczącego wysiłku, a zespół jest w trakcie sprintu, w którym ma do wykonania inną pracę. Podejmując próbę całościowego rozwiązania problemu dłużu, może nie być w stanie zrealizować założonego celu sprintu.

Aby poradzić sobie z tym problemem, zespół może wygospodarować pewną procentową ilość czasu na zwalczanie odkrytego dłużu technicznego w chwili jego ujawnienia. Jednym ze sposobów ustalania takiego budżetu czasowego jest podnoszenie ocen indywidualnych elementów rejestru produktu pozwalające na dodatkową obsługę odkrytego dłużu technicznego, który zazwyczaj tkwi w kodzie. Ewentualnie zespół może zabudżetować procentową ilość całego swojego czasu dostępnego w sprintie na spłacanie dłużu technicznego. Przykłady, które widziałem w przeszłości, mieściły się w przedziale od 5% do 33% pojemności sprintu. Przymierzając się do tego podejścia, powinieneś dokonać wyboru procentowej ilości czasu w oparciu o swoje specyficzne okoliczności.

Część odkrytego dłużu technicznego, która nie została obsłużona w chwili napotkania, powinna zostać zakwalifikowana jako znany dług techniczny i ujawniona (udokumentowana) przy użyciu wybranej przez zespół techniki wizualizacji dłużu technicznego.

## Spłacaj dług techniczny przyrostowo

W niektórych produktach poziom skumulowanego dłużu technicznego może być całkiem wysoki. Zespoły pracujące nad takimi produktami często w ramach obsługi dłużu technicznego muszą dokonywać spłaty w formie raty balonowej<sup>1</sup>. Znacznie lepiej byłoby dla nich, gdyby zamiast wielkich płatności dokonywały wielu regularnych, przyrostowych spłat dłużu technicznego. Mniejsze, bardziej regularne płatności przypominają miesięczne spłaty kredytu hipotecznego. W ten sposób można likwidować dług częstokrotnie każdego miesiąca, unikając wielkich płatności pod koniec pożyczki.

Niepokoi mnie, kiedy zespoły zaczynają rozważać „sprinty spłaty dłużu technicznego” lub „sprinty refaktoryzacji”. Są to sprinty, których jedynym celem jest wykonanie pracy redukującej dług techniczny i dla mnie wydają się być one płatnościami balonowymi. Obecność tego typu sprintów świadczy o tym, że pozwolono bez nadzoru i prób redukcji rosnąć długowi technicznemu. Teraz problem urósł do takiego rozmiaru, że zamiast wytwarzania w następnym sprintie cech mających

<sup>1</sup> Patrz [http://pl.wikipedia.org/wiki/Rata\\_balonowa](http://pl.wikipedia.org/wiki/Rata_balonowa) – przyp. tłum.

wartość dla klienta, zespół poświęci się całkowicie rozwiązywaniu problemów, którym powinien był poświęcać odrobinę czasu w każdym sprincie. Kiedy poziom dłużu jest bardzo duży i nikt nie zwraca na niego uwagi, czasami skupienie wysiłków całego zespołu na jego spłacie może być pomocne w uświadczaniu wszystkim skali problemu. Jednak należy unikać tego typu sprintów, kiedy tylko jest to możliwe — spłata dłużu powinna przebiegać w sposób przyrostowy.

Przy takim podejściu bierzemy pewną ilość znanego dłużu technicznego i zamieniamy go w celowy dług techniczny przeznaczony do likwidacji w trakcie następnego sprintu. Decyzję o tym, ile celowego dłużu technicznego przyjąć do każdego sprintu, podejmuje zespół scrumowy w trakcie planowania sprintu.

## Spłacaj w pierwszej kolejności dług o najwyższych odsetkach

Chociaż bardzo wygodne jest nadać wszystkim skrótom i niedociągnięciom jedną wspólną etykietkę dłużu technicznego, trzeba zdać sobie sprawę z faktu, iż nie wszystkie typy dłużu technicznego mają jednakową wagę. Przykładem istotnego dłużu technicznego jest często modyfikowany moduł, od którego zależy wiele innych modułów, wymagający refaktoryzacji ze względu na rosnący szybko stopień jego skomplikowania i związaną z tym trudność we wprowadzaniu zmian. Na okrągło płacimy odsetki od tego dłużu, a ich skala rośnie wraz z ilością modyfikacji samego modułu.

Z drugiej strony, moglibyśmy mieć dług techniczny (w postaci znanego problemu projektowego lub implementacyjnego) w mało używanej części produktu, która jest bardzo rzadko modyfikowana. W trakcie regularnej pracy płacimy bardzo małe lub wręcz zerowe odsetki od tego dłużu. Nie jest to dług, który wymaga wiele uwagi, z wyjątkiem pewnego ryzyka, iż ta część produktu może zawieść, co miałyby istotne negatywne konsekwencje.

Zatem obsługując dług techniczny, powinniśmy w pierwszej kolejności celować w dług o najwyższych odsetkach i redukować go. Każda rozsądnie myśląca osoba zajmująca się biznesem zrobiłaby to samo z dłużem finansowym. Na przykład: o ile nie ma ważnego powodu, aby zachować się inaczej, w pierwszej kolejności spłacilibyśmy kredyt z oprocentowaniem 18%, a dopiero później kredyt z oprocentowaniem 6%.

Niektóre organizacje kumulują tak duże ilości dłużu technicznego, iż dotycza ich pewna forma paraliżu spowodowana brakiem pomysłu na rozpoczęcie jego spłaty. Mogą być one świadome istnienia dłużu o wysokim oprocentowaniu, ale jego rozmiar zniechęca do podejmowania działań. Chcąc powstrzymać nieustanny napływ dłużu, organizacje takie mogą zacząć od spłaty w małej ilości, przyzwyczajając się w ten sposób do procesu spłaty dłużu w ratach. Popieram wszelkiego typu działania prowadzące do zmian kulturowych, które wstrząsną organizacją dostatecznie mocno, aby ta rozpoczęła zarządzanie swoim dłużem. Jeżeli będziemy spłacać dług, realizując cechy mające wartość dla klienta — o czym będę pisał za chwilę — możemy stopniowo skupiać się na małych porcjach dłużu, który warto spłacić.

## Spłacaj dług techniczny podczas wykonywania pracy mającej wartość dla klienta

Doskonałym sposobem spłaty znanego dłużu technicznego metodą przyrostową, z uwagą poświęconą na obsługę dłużu o najwyższych odsetkach i jednoczesnym podejściem skupionym na dostarczaniu wartości, jest dokonywanie płatności podczas wykonywania pracy cennej z punktu widzenia klienta. Kiedy tylko jest to możliwe, staraj się unikać rezerwowania całego sprintu na redukcję dłużu

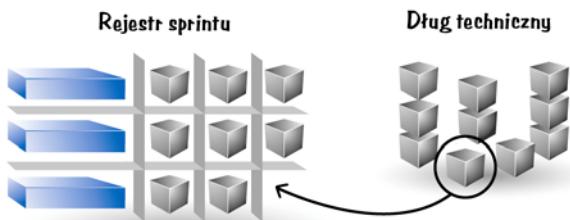
lub też tworzenia rejestru, którego elementy opisują zadania związane wyłącznie z likwidowaniem dłużu. Zamiast stosować tego typu działania, powinniśmy obsługiwać znany dług techniczny jednocześnie z wytwarzaniem cech mających wartość dla klienta umieszczonych w rejestrze produktu.

Załóżmy, że przy każdym elemencie rejestru produktu reprezentującym wartość dla klienta, nad którym pracujemy, wykonujemy również kilka innych rzeczy. Po pierwsze, zobowiązujemy się wykonać pracę na wysokim poziomie jakości, dzięki czemu nie dodamy nowego natywnego dłużu technicznego podczas tworzenia nowej cechy. Po drugie, stosujemy zasadę skauta i do pewnego rozsądniego poziomu usuwamy odkryty dług techniczny, jaki napotkaliśmy w obszarze, w którym realizujemy nową cechę. Po trzecie (kluczowy składnik tego podejścia), spłacamy pewną część celowego dłużu technicznego w obszarze będącym terenem naszej nadchodzącej pracy.

Takie podejście ma kilka zalet:

- przepłata pracę nad redukcją dłużu z pracą nad cechami dla klienta, którą właściciel produktu może odpowiednio uporządkować według priorytetów;
- uświadamia wszystkim członkom zespołu deweloperskiego, iż redukcja dłużu technicznego jest ich wspólną odpowiedzialnością, a nie czymś, co można odłożyć lub oddelować celem wyczyszczenia do innej osoby lub zespołu;
- rozbudowuje umiejętności przeciwdziałania dłużowi technicznemu i usuwania go, ponieważ każdy ma okazję praktykować te umiejętności na okrągło;
- pomaga nam zidentyfikować obszary dłużu o najwyższych odsetkach, na których powinniśmy skupić nasze działania związane ze spłatą dłużu, a w najgorszym przypadku dowiadujemy się, że kod (lub inny artefakt procesu deweloperskiego), którym się zajmujemy, jest nadal ważny, ponieważ używamy go do stworzenia nowej cechy;
- pozwala na uniknięcie marnotrawstwa spłaty dłużu technicznego w obszarach, gdzie nie jest to faktycznie potrzebne.

Wcześniej wspomniałem o podejściu stosowanym przez kilka znanych mi zespołów scrumowych, którego celem było ułatwienie jednoczesnej realizacji zadań redukcji dłużu z wykonywaniem elementów z rejestru produktu (patrz rysunek 8.11).



**RYSUNEK 8.11.** Technika służąca do zarządzania dłużem technicznym w Scrumie

W tym podejściu podczas planowania sprintu elementy znanego dłużu technicznego wprowadzane są do rejestru dłużu technicznego umieszczonego na ścianie obok rejestru sprintu (lub do systemu komputerowego przeznaczonego do tego samego celu).

Podczas planowania sprintu, gdy członkowie zespołu pracują razem z właścicielem produktu nad wyborem z rejestru produktu elementów wartościowych dla klienta przeznaczonych do realizacji w kolejnym sprincie, biorą oni pod uwagę karteczki na tablicy z długiem technicznym i sprawdzają, czy planowana przez nich praca nad nowymi cechami produktu nie przetnie się w naturalny sposób z obszarem dłużu technicznego opisanego na danej karteczkę. Jeśli tak jest, ktoś odrywa tę karteczkę z tablicy dłużu i umieszcza ją w rejestrze sprintu jako element przeznaczony do realizacji w tym sprincie. Podczas realizacji prac niezbędnych do ukończenia elementów z rejestru produktu członkowie zespołu zajmują się również zadaniami związanymi z redukcją dłużu wciągniętymi do sprintu.

Takie podejście jest bardzo prostym i eleganckim sposobem ułożenia prac nad obsługą dłużu technicznego z wytwarzaniem wartości dla klienta.

## Zakończenie

W tym rozdziale omawiałem koncepcję dłużu technicznego, który przyrasta w wyniku stosowania skrótów kosztem przyszłych nakładów pracy. Rozróżniłem trzy rodzaje dłużu: natywny, nieunikniony i strategiczny. Wyjaśniłem konsekwencje złego zarządzania dłużem technicznym. Następnie zająłem się trzema aktywnościami związanymi z kontrolą dłużu technicznego: zarządzaniem przyrostem, ujawnianiem i obsługą dłużu.

Ten rozdział kończy pierwszą część książki. W kolejnym rozdziale przejdę do omawiania różnych ról obecnych w Scrumie, zaczynając od właściciela produktu.



Część II

ROLE

---



## Rozdział 9

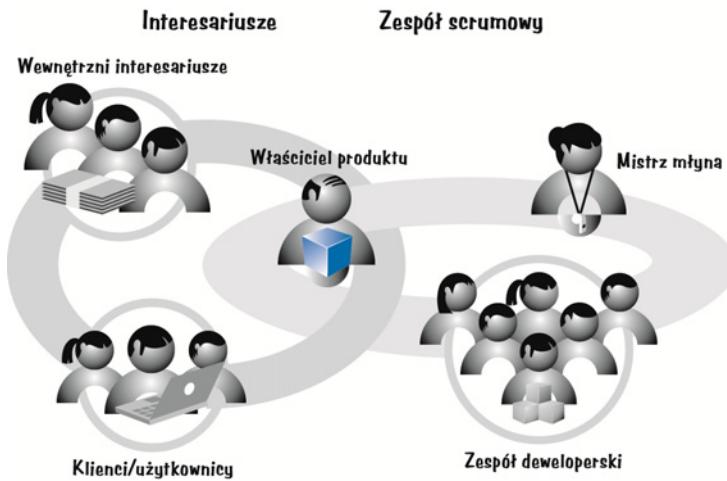
# WŁAŚCICIEL PRODUKTU

W tym rozdziale rozwinię opis roli właściciela produktu. Zaczynę od wyjaśnienia celu istnienia tej roli w odniesieniu do pozostałych ról scrumowych. Następnie opiszę szczegółowo cechy i obowiązki właściciela produktu. Dalej przyjdzie pora na przedstawienie „dnia z życia” właściciela produktu na przestrzeni kilku tygodni. Wskażę, kim powinien być właściciel produktu dla różnych typów prac deweloperskich. Rozdział zakończę, opisując sposoby łączenia roli właściciela produktu z innymi rolami oraz sposób przeskalowania takiego podejścia na zespół właścicieli produktu.

## Wprowadzenie

Właściciel produktu jest scentralizowanym punktem dowodzenia produktem. Jest to jedna z trzech współpracujących ze sobą ról, jakie tworzą wspólnie zespół scrumowy (pozostałe dwie to mistrz młyna i zespół deweloperski).

Właściciel produktu musi patrzeć jednocześnie przynajmniej w dwóch kierunkach (patrz rysunek 9.1).



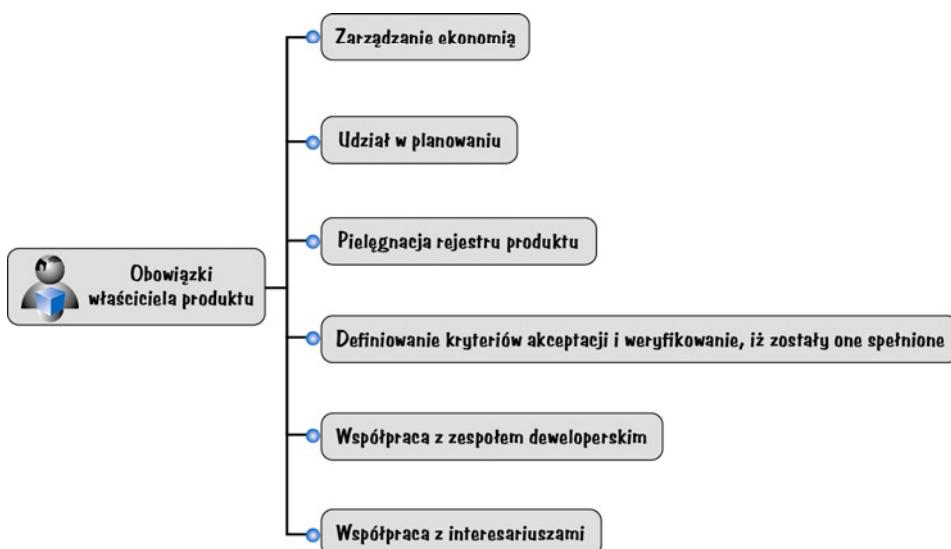
**RYSUNEK 9.1.** Właściciel produktu patrzy w dwóch kierunkach jednocześnie

Z jednej strony właściciel produktu musi dostatecznie dobrze rozumieć potrzeby i priorytety interesariuszy organizacji, jej klientów i użytkowników, aby być ich przedstawicielem. Pod tym względem właściciel produktu działa jako menedżer produktu pilnujący, aby powstało prawidłowe rozwiązanie.

Z drugiej strony właściciel produktu musi komunikować zespołowi deweloperskiemu, co należy zbudować oraz w jakiej kolejności. Jego zadaniem jest również zapewnienie odpowiednich kryteriów akceptacji cech, a także wykonanie w późniejszej fazie testów udowadniających, iż kryteria te zostały spełnione. Właściciel produktu nie zajmuje się pisaniem szczegółowych testów, ale musi zadbać o to, aby zostały stworzone testy wysokiego rzędu, dzięki którym zespół będzie mógł zrozumieć, co jest niezbędne, aby właściciel produktu mógł uznać cechę za wykonaną. Z tej perspektywy właściciel produktu jest po części analitykiem biznesowym, a po części testerem.

## Główne obowiązki

Główne obowiązki właściciela produktu ilustruje rysunek 9.2.

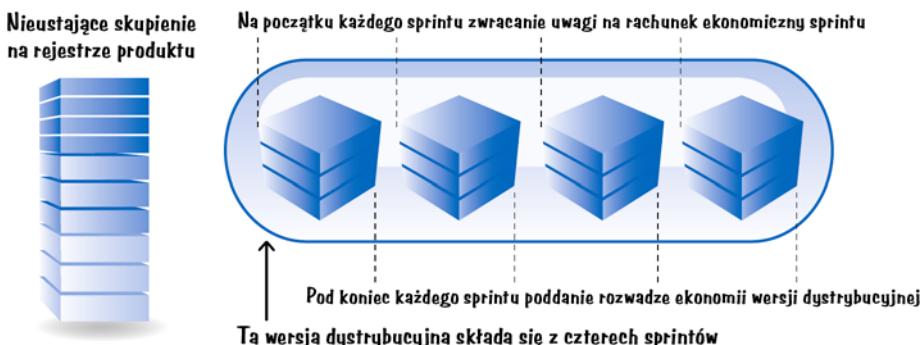


**RYSUNEK 9.2.** Główne obowiązki właściciela produktu

Na pierwszy rzut oka widać, że jest to rola na pełny etat cechująca się istotnymi obowiązkami. Czytając listę zadań, można wręcz pomyśleć, że jedna osoba jest niewystarczająca do sprawowania wszystkich wymienionych funkcji lub też niezdolna do posiadania wszystkich atrybutów niezbędnych do osiągnięcia sukcesu w tej roli. W większości przypadków jedna osoba może i powinna wy pełniać rolę właściciela produktu, chociaż z praktycznego punktu widzenia dopuszczalne jest wprowadzenie pod pewnymi warunkami pełnomocników. Oba podejścia zostaną opisane w dalszej części rozdziału.

## Zarządzanie ekonamią

Właściciel produktu ma za zadanie zapewnić podejmowanie właściwych ekonomicznie decyzji na poziomach wersji dystrybucyjnej, sprintu i rejestru produktu (patrz rysunek 9.3).



**RYSUNEK 9.3.** Zarządzanie ekonamią przez właściciela produktu

### Ekonoma na poziomie wersji dystrybucyjnej

Na poziomie wersji dystrybucyjnej właściciel produktu odbiera strumień ważnych informacji napływających w trakcie prac deweloperskich i dokonuje nieustannie kompromisów pod względem zakresu prac, czasu wykonania, budżetu i jakości. Kompromisy podjęte na początku mogą stracić ważność w obliczu nowych informacji napływających w trakcie prac nad wersją.

Na przykład: co zrobić, jeśli na kilka tygodni przed końcem sześciomiesięcznego projektu z ustaloną datą dystrybucji odkrywamy możliwość zwiększenia zysku o 50%, jeśli poświęcimy mu dodatkowy tydzień (wprowadzimy 4-procentowy poślizg w czasie) i dodamy świeżo zidentyfikowaną cechę do wersji? Czy powinniśmy poświęcić ten tydzień i dodatkowe środki w celu uzyskania dodatkowego zysku? Podjęcie takiej decyzji nadzoruje właściciel produktu. W wielu przypadkach może on zadecydować jednostronnie, a innym razem jedynie zarekomendować propozycję, prosząc jednocześnie o opinię (lub zatwierdzenie) innych przed faktycznym wykonaniem postanowienia.

Pod koniec każdego sprintu właściciel produktu nadzoruje podjęcie decyzji o przedłużeniu lub wstrzymaniu finansowania kolejnego sprintu. Jeżeli osiągnięto duży postęp w kierunku założonego celu wersji lub istnieje inne ekonomiczne uzasadnienie finansowania następnego sprintu, zostaną przyznane środki na następną iterację. W przypadku kiepskiego postępu lub uwarunkowań ekonomicznych niesprzyjających dalszemu wydawaniu pieniędzy wysiłek deweloperski może zostać przerwany.

Usatysfakcjonowany właściciel produktu może również przekazać informację o zaprzestaniu finansowania dalszych prac deweloperskich pod koniec sprintu, jeśli uzna, że produkt nadaje się do dystrybucji i dalsze wydawanie pieniędzy na produkcję nie ma sensu. Założmy, że planowaliśmy wersję dystrybucyjną na dziesięć sprintów. Po sprintie numer 7 właściciel produktu przegląda elementy pozostające w rejestrze produktu i dochodzi do wniosku, że koszt ichtworzenia przekracza potencjalny zysk z ich posiadania. W jego opinii bardziej uzasadnione ekonomicznie może być wcześniejsze wypuszczenie produktu na rynek zamiast kontynuacji oryginalnego 10-sprintowego planu. Ta elastyczność pozwalająca na wcześniejsze wdrożenie jest możliwa dzięki nadaniu elementom priorytetów w taki sposób, aby najważniejsze z nich znalazły się na szczycie, a także dzięki wykonywaniu przez zespół pracy w każdym sprintie zgodnie z solidną definicją ukończenia.

Oczywiście właściciel produktu może również dojść do wniosku, iż należy zaprzestać finansowania pod koniec sprintu ze względu na zmianę klimatu ekonomicznego. Na przykład możemy tworzyć produkt przeznaczony dla specyficznego kraju, a w tym samym czasie pewne ciało nadzorcze w tym kraju postanowi zmienić przepisy, sprawiając, że nasza aplikacja nie będzie miała szansy przynieść dochodów lub wręcz jej sprzedaż stanie się nielegalna. W takich przypadkach właściciel produktu może doprowadzić do anulowania wysiłku deweloperskiego, nawet jeśli prace postępują płynnie do przodu.

### **Ekonoma na poziomie sprintów**

Poza ekonomią na poziomie wersji dystrybucyjnej właściciel produktu zarządza ekonomią na poziomie sprintów, zapewniając, aby każdy z nich przynosił dobry zwrot z inwestycji (ROI). Dobrzy właściciele produktu traktują pieniądze organizacji tak, jakby były one ich własnymi pieniędzmi. W większości przypadków właściciel produktu wie, jaki koszt wiąże się z kolejnym sprintem (czas jego trwania oraz skład zespołu w następnym sprincie są znane). Mając tę wiedzę, właściciel produktu powinien zapytać samego siebie podczas planowania sprintu: „Czy byłbym gotowy wypisać czek obciążający moje własne konto bankowe na kwotę odpowiadającą kosztom tego sprintu, aby otrzymać w zamian cechy, które zmierzamy w nim zrealizować?”. Jeśli odpowiedź brzmi „nie”, dobry właściciel produktu nie powinien wydać pieniędzy organizacji na ten sprint.

### **Ekonoma na poziomie rejestru produktu**

Jak wspominałem w rozdziale 6., właściciel produktu odpowiada za posortowanie elementów rejestru produktu według priorytetów. Kiedy zmieniają się uwarunkowania ekonomiczne, również priorytety w rejestrze najprawdopodobniej ulegną przetasowaniu.

Oto przykład. Powiedzmy, że na początku wersji dystrybucyjnej właściciel produktu zakłada sporą wartość pewnej cechy dla szerokiego grona użytkowników docelowych, a zespół deweloperski jest przekonany, że jej stworzenie wymaga niewielkiego wysiłku. Jednak po kilku sprintach zespół przekonuje się, że implementacja cechy będzie wymagać sporych nakładów pracy i ma ona wartość jedynie dla ułamka docelowych użytkowników. Ze względu na dramatyczną zmianę wspólnika kosztu do zysku tej cechy właściciel produktu powinien zmienić priorytety w rejestrze tak, aby odzwierciedlały one tę nową wiedzę — być może przez usunięcie elementów rejestru produktu związanych z tą cechą.

### **Udział w planowaniu**

Właściciel produktu jest kluczowym uczestnikiem aktywności planowania portfela, produktu, wersji dystrybucyjnej i sprintów. Podczas planowania portfela (patrz rozdział 16.) właściciel produktu współpracuje z wewnętrzny interesariuszami (może to być komitet zatwierdzający lub rada nadzorcza) w celu umieszczenia produktu w prawidłowym miejscu rejestru portfela i określenia, kiedy należy zacząć i zakończyć związane z nim prace deweloperskie. W trakcie planowania produktu (patrz rozdział 17.) właściciel produktu współpracuje z interesariuszami w celu stworzenia wizji produktu. Planując wersję dystrybucyjną (patrz rozdział 18.), właściciel produktu wspólnie z interesariuszami i zespołem definiuje zawartość kolejnej wersji. Podczas planowania sprintu (patrz

rozdział 19.) właściciel produktu współpracuje z zespołem deweloperskim, aby określić cel sprintu. Dostarcza również cennych informacji pozwalających zespołowi deweloperskiemu wskazać zbiór elementów rejestru produktu, które będzie można realistycznie dostarczyć przed końcem sprintu.

## Pielęgnacja rejestru produktu

Właściciel produktu sprawuje pieczę nad pielęgnacją rejestru produktu, która obejmuje tworzenie i uszczegóławianie elementów rejestru, nadawanie im ocen oraz priorytetów (patrz rozdział 6.). Nie wykonuje on wszystkich tych prac osobiście. Na przykład nie wszystkie elementy rejestru produktu pisane są przez właściciela produktu — mogą je dodawać również inne osoby. Właściciel produktu nie ocenia również elementów rejestru (robi to zespół deweloperski), ale jest obecny w trakcie głosowania i udziela odpowiedzi na pytania oraz dostarcza dodatkowych wyjaśnień. To on jest jednak ostatecznie odpowiedzialny za przeprowadzanie aktywności związanych z pielęgnacją, które promują płynny przyrost wartości dla klienta.

## Definiowanie kryteriów akceptacji i weryfikowanie, iż zostały one spełnione

Właściciel produktu jest odpowiedzialny za definiowanie kryteriów akceptacji dla każdego elementu rejestru produktu. Są to warunki konieczne, które muszą być spełnione, aby właściciel produktu mógł z satysfakcją przyznać, że wszelkie funkcjonalne i niefunkcjonalne wymagania zostały zrealizowane. Właściciel produktu w oparciu o kryteria akceptacji może również pisać testy akceptacyjne lub werbować w tym celu ekspertów z danej dziedziny lub samych członków zespołu deweloperskiego. Niezależnie od przyjętego podejścia właściciel produktu powinien upewnić się, iż kryteria akceptacji (a często również specyficzne testy akceptacyjne) zostały stworzone, zanim element trafi pod obrady w trakcie planowania sprintu. Bez nich zespół nie będzie w pełni rozumiał danego elementu i nie będzie mógł przyjąć go do realizacji w sprincie. Z tego względu wiele zespołów scrumowych dołącza istnienie jasnych kryteriów akceptacji do swojej listy kontrolnej definicji gotowości (patrz rozdział 6.).

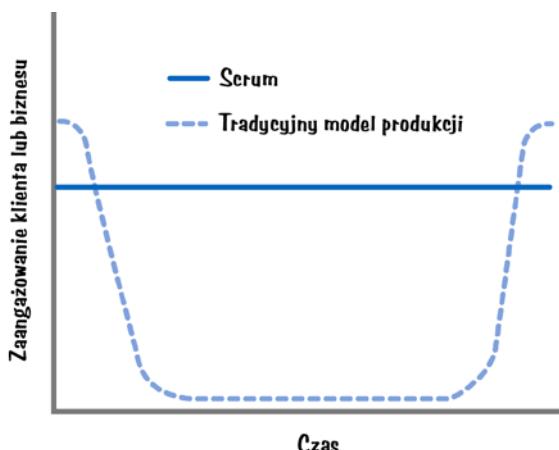
Właściciel produktu odpowiada za ostateczne sprawdzenie, czy spełnione zostały kryteria akceptacji. W tym celu może on przeprowadzić odpowiednie testy osobiście lub poprosić użytkowników posiadających odpowiednią wiedzę ekspercką o sprawdzenie, czy dany element rejestru produktu spełnia warunki satysfakcji. Zespół może pomóc w stworzeniu odpowiedniego środowiska testowego umożliwiającego właścielowi produktu lub ekspertom posiadającym odpowiednią wiedzę przeprowadzenie testów, ale ostateczny werdykt odnośnie do spełnienia oczekiwani wydawany jest przez właściciela produktu.

Ważne jest, aby właściciel produktu zweryfikował kryteria akceptacji jeszcze w trakcie wykonania sprintu i nie odwlekał tej czynności do przeglądu sprintu. Wykonując testy już w momencie ukończenia cech, właściciel produktu może zidentyfikować pomyłki oraz źle zinterpretowane założenia, które zespół zdola jeszcze poprawić przed przeglądem sprintu. Wykonanie testów akceptacyjnych przed przeglądem sprintu jest istotne również z tego względu, iż zespół deweloperski może zaprezentować jedynie ukończone cechy. W związku z tym deweloperzy muszą wiedzieć, które cechy spełniły kryteria akceptacji przed przeglądem.

## Współpraca z zespołem deweloperskim

Właściciel produktu musi ściśle współpracować z zespołem deweloperskim. Rola właściciela produktu polega na zaangażowaniu i poświęceniu dzień po dniu. Wiele organizacji wdrażających Scrum nie radzi sobie z wypracowaniem odpowiedniego poziomu współpracy pomiędzy właścicielem produktu i zespołem deweloperskim, co opóźnia przepływ kluczowych informacji, a także redukuje ich wartość w chwili, kiedy faktycznie zostaną przekazane.

Ta porażka w zaangażowaniu może również ujawnić się, kiedy ludzie nieoswojeni jeszcze z rolą właściciela produktu przyjmują założenie, iż ich poziom zaangażowania w przypadku Scruma powinien odpowiadać poziomowi zaangażowania, jaki miał miejsce w trakcie produkcji starą metodą sekwencyjną. Rysunek 9.4 przedstawia typowy poziom zaangażowania ze strony klienta lub biznesu w tradycyjnym, sekwencyjnym modelu produkcji oraz oczekiwany poziom zaangażowania właściciela produktu w Scrumie.



**RYSUNEK 9.4.** Porównanie zaangażowania klienta lub biznesu w miarę upływu czasu

W tradycyjnym, sekwencyjnym modelu produkcji wykres poziomu zaangażowania przypomina literę U (lub też ma kształt wanny). Na początku klienci są mocno zaangażowani w definiowanie wymagań. Kiedy produkcja wkroczy w bardziej techniczne fazy (projektowanie, implementacja i pewne rodzaje testowania), klienci „nie są dłużej potrzebni.” Oznacza to, że ich poziom zaangażowania przez większą część wysiłku deweloperskiego jest znikomy lub wręcz zerowy. W praktyce wkraczają oni na scenę dopiero pod sam koniec produkcji, kiedy przychodzi pora na przeprowadzenie testów akceptacyjnych tego, co zostało zbudowane. Na ogół klienci odkrywają wtedy, że to, co powstało, nie odpowiada temu, czego oczekiwali. Zazwyczaj jest już dla nich zbyt późno, aby wprowadzać jakiekolwiek zmiany lub koszt wprowadzenia tych zmian jest zbyt duży — przynajmniej w tej wersji dystrybucyjnej. Klienci przybyli z nadziejęą zachwycenia się — wychodzą zaskoczeni, sfrustrowani i zawiedzeni. Wtedy też na sile przybiera wytykanie palcami winnych. Klienci twierdzą: „Gdybyście przeczytali dokładniej naszą specyfikację wymagań, zbudowalibyście dokładnie to, czego potrzebowaliśmy”, natomiast zespół deweloperski ripostuje: „Gdyby dokument został napisany bardziej przejrzyście, zbudowalibyśmy coś zupełnie innego, a tak zbudowaliśmy to, o co prosiliście”.

W Scrumie budujemy jedną cechę na cykl, a nie jedną fazę na cykl. Oznacza to, że w trakcie pojedynczego sprintu wykonujemy wszystkie czynności potrzebne do stworzenia danej cechy (projektowanie, implementacja, integracja i testowanie). Dlatego zaangażowanie właściciela produktu w tym czasie ma kluczowe znaczenie. Dzięki tak bliskiej interakcji w trakcie krótkich, ograniczonych czasowo iteracji istnieje o wiele mniejsze prawdopodobieństwo zgubienia porozumienia pomiędzy właścicielem produktu i zespołem deweloperskim. Przy okazji w Scrumie unika się pokazywania palcem winnych!

## Współpraca z interesariuszami

Właściciel produktu jest pojedynczym głosem reprezentującym całą społeczność interesariuszy, zarówno wewnętrznych, jak i zewnętrznych. **Interesariuszami wewnętrznymi** mogą być właściciele systemów biznesowych, zarząd, menedżerowie programów, marketing i sprzedaży. **Interesariusze zewnętrzni** to między innymi klienci, użytkownicy, partnerzy, urzędy regulujące i inni. Właściciel produktu musi blisko współpracować ze wszystkimi interesariuszami, uzyskując od nich potrzebne informacje, a następnie syntezując je w spójną wizję będącą podstawą do sterowania produkcją.

Jeżeli właściciel produktu pozwoli się przytłoczyć i zepchnąć do zbytnich szczegółów, będzie mu trudno współpracować na odpowiednim poziomie zarówno z zespołem deweloperskim, jak i interesariuszami. W pewnych okolicznościach nawet pracy może być zbyt duży, aby w sposób racjonalny mogła ją wykonać jedna osoba. Wtedy właściciel produktu może zwerbować asystentów, aby pomogli mu w wypełnianiu obowiązków narzuconych przez rolę. Problemem tym zajmę się później, przy okazji omawiania koncepcji zespołu właścicieli produktu.

## Cechy i umiejętności

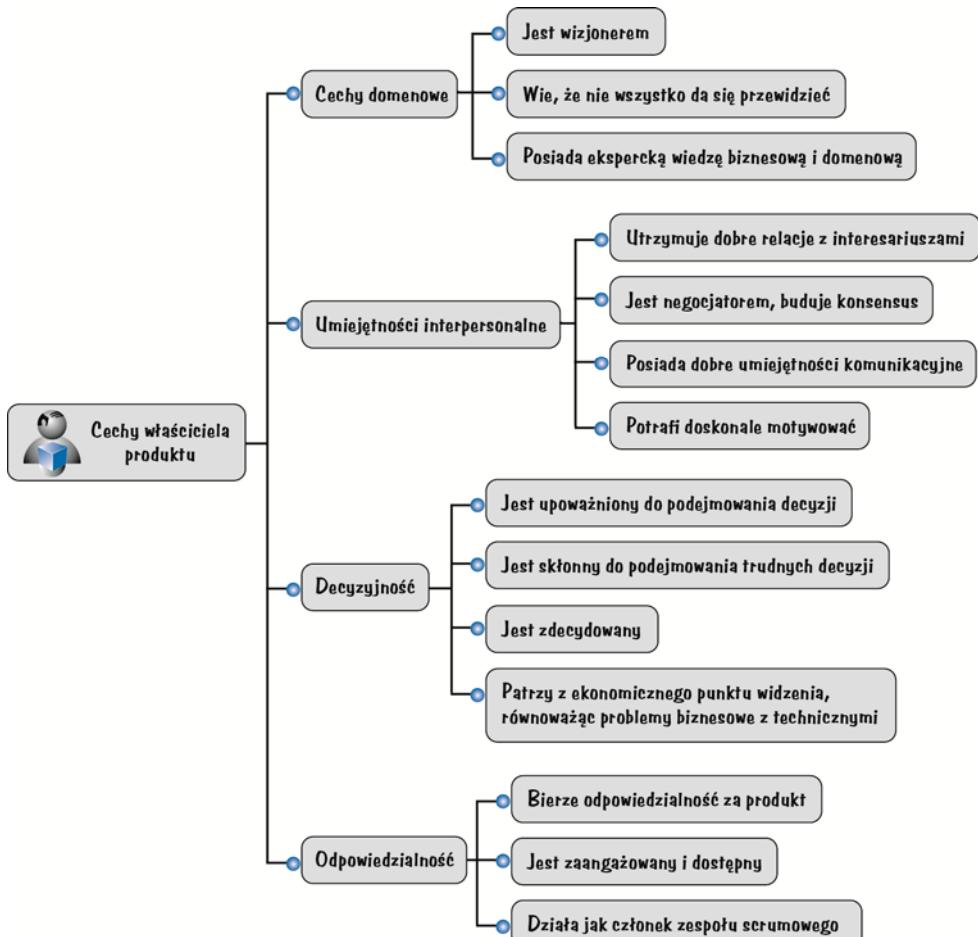
Istotne cechy roli właściciela produktu przedstawia rysunek 9.5.

Chociaż istnieje wiele cech, które usiłuję odnaleźć w dobrym właścicielu produktu, można je zgrupować w cztery kategorie: umiejętności domenowe, zdolności interpersonalne, decyzyjność i odpowiedzialność.

### Umiejętności domenowe

Właściciel produktu syntezuje wizję produktu, a także prowadzi zespół w kierunku osiągnięcia tej wizji. Posiadanie wizji nie oznacza, że każdy jej detal jest znany lub też ścieżka do jej osiągnięcia jest zupełnie jasna. Dobry właściciel produktu wie, że nie wszystko można przewidzieć z góry i jest gotów do adaptacji w obliczu zmian.

Chcąc skutecznie tworzyć i realizować wizję, właściciel produktu musi posiadać odpowiednią wiedzę z zakresu biznesu i domeny produktu. Trudno jest osiągać dobre efekty jako właściciel produktu, wiedząc niewiele o produkcie i jego otoczeniu. Jak zabrać się do ustalania priorytetów pomiędzy konkurencyjnymi ze sobą cechami, jeśli nie zna się danej tematyki?



**RYSUNEK 9.5.** Cechy właściciela produktu

## Umiejętności interpersonalne

Właściciel produktu musi być również „głosem klienta”, co wymaga od niego utrzymywania dobrych stosunków z interesariuszami. Zazwyczaj właściciel produktu ma do czynienia z kilkoma interesariuszami o sprzecznych potrzebach, w związku z czym powinien być dobrym negocjatorem, budującym porozumienie.

Właściciel produktu jest łącznikiem pomiędzy środowiskiem interesariuszy i pozostałą częścią zespołu scrumowego. Przebywanie na tej pozycji wymaga od niego dobrych umiejętności komunikowania się z obiema grupami ludzi i przekazywania każdej z nich informacji przy użyciu odpowiedniej terminologii. Osoba o zdolnościach komunikacyjnych powinna również: zabierać głos w sytuacjach, kiedy wypowiedź może burzyć obecną równowagę, być pewna swoich poglądów, mieć wiedzę w danej tematyce, przekazywać informacje w sposób prosty, zwięzły i łatwy do zrozumienia, być wiarygodna.

Właściciel produktu jest również wielkim motywatorem. Kiedy sprawy się komplikują, właściciel produktu może przypomnieć ludziom, dlaczego inwestują swój wysiłek, i pomóc im utrzymać entuzjastyczne podejście przez ponowne przedstawienie wizji biznesowej.

## Podejmowanie decyzji

Właściciel produktu powinien być również uprawniony do podejmowania decyzji. Częstym utrudnieniem w organizacjach wprowadzających Scrum jest to, że osoba wybrana do pełnienia roli właściciela produktu nie posiada upoważnienia do podejmowania jakichkolwiek ważnych decyzji. Taka osoba nie jest właścicielem produktu.

Właściciel produktu powinien wykazywać chęć do podejmowania trudnych decyzji — zazwyczaj wymagających kompromisów co do zakresu prac, terminów i kosztów. Decyzje te należy często podejmować pod presją czasu i nie wycofywać się z nich bez naprawdę istotnego powodu. Innymi słowy, właściciel produktu powinien być zdecydowanym decydentem.

Podejmując decyzje, właściciel produktu musi zachowywać odpowiednią równowagę pomiędzy potrzebami biznesowymi i możliwościami inżynierijnymi. Chociaż cały zespół ponosi odpowiedzialność za przyrastanie dłużu technicznego ponad akceptowalny poziom, częstym czynnikiem wpływającym na ten przyrost są decyzje podejmowane przez właściciela produktu bez uwzględnienia ich wpływu na system.

## Odpowiedzialność

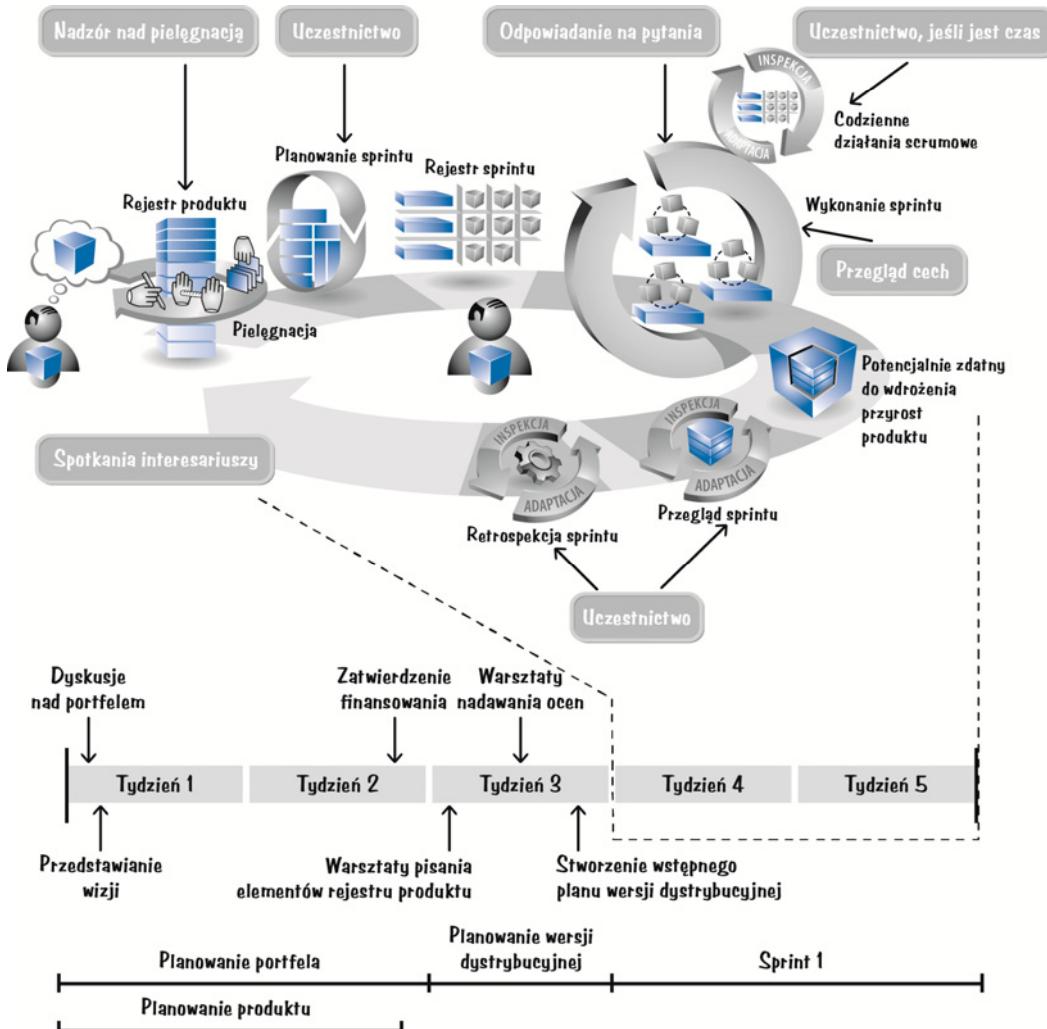
Właściciel produktu jest odpowiedzialny za dostarczenie dobrych wyników biznesowych. Ta odpowiedzialność nie zwalnia pozostałych członków zespołu scrumowego z ich własnego obowiązku uczestnictwa w procesie generowania dobrego zwrotu z inwestycji. Niemniej jednak to właściciel produktu jest na świeczniku pod względem odpowiedzialności za ekonomiczne wykorzystanie zasobów i to on ponosi konsekwencje porażki. Tak czy owak to właśnie właściciel produktu ma wiele możliwości zmiany rejestru produktu w trakcie prac, dostosowania priorytetów lub nawet rezygnacji z prac deweloperskich.

Właściciel produktu musi być zaangażowany i dostępny zarówno dla interesariuszy, jak i dla pozostałej części zespołu scrumowego. Bycie właścicielem produktu to praca na cały etat — próba wykonywania jej jako dodatkowego zajęcia to przepis na porażkę.

I w końcu właściciel produktu jest członkiem zespołu scrumowego. Zdaje sobie sprawę, że dobrych wyników nie da się osiągnąć bez wspólnego wysiłku wszystkich jego członków i w związku z tym traktuje zespół deweloperski i mistrza młyna z szacunkiem, zakładając, że wszyscy razem są partnerami mającymi na celu dostarczanie oczekiwanych od nich wyników. Członkowie zespołu scrumowego powinni mieć nastawienie muszkieterów (tę koncepcję opiszę dalej w rozdziale 11.). Nie ma podejścia w stylu „my kontra oni”. Właściciel produktu, mistrz młyna i zespół deweloperski są jednostką pracującą z zamiarem osiągnięcia tego samego celu.

## Dzień z życia właściciela produktu

Aby lepiej docenić zakres obowiązków właściciela produktu, przyjrzyjmy się, jak wygląda „dzień z jego życia” na przestrzeni wysiłku deweloperskiego (patrz rysunek 9.6).



**RYSUNEK 9.6.** Dzień z życia właściciela produktu

Podczas tygodni numer 1 i 2 właściciel produktu jest zaangażowany zarówno w planowanie portfela (patrz rozdział 16.), jak i w planowanie produktu (patrz rozdział 17.). W ramach planowania portfela właściciel produktu może współpracować z menedżerem portfela lub radą nadzorczą w celu omówienia oczekiwów mogących mieć wpływ na planowanie nowego produktu. Wynikiem tych dyskusji są założenia do **planowania produktu**, w trakcie którego właściciel produktu wspólnie z właściwymi interesariuszami i innymi osobami tworzy wizję nowego produktu.

Po zakończonym planowaniu produktu jego propozycja zostaje przesłana na spotkanie planowania portfela, gdzie przechodzi przez **filtr ekonomiczny** organizacji, mający określić, czy znajdują się środki na realizację produktu oraz kiedy będzie można rozpoczęć prace deweloperskie. Rysunek 9.6 pokazuje, że wszystkie te działania mają miejsce bezpośrednio po zakończeniu planowania produktu. W wielu organizacjach występuje zazwyczaj pewne opóźnienie pomiędzy przedstawieniem wizji produktu a jej przeglądem, zatwierdzeniem środków i dniem zielonego światła przez komitet zatwierdzający lub radę nadzorczą.

W trzecim tygodniu właściciel produktu jest zaangażowany we wstępne planowanie wersji dystrybucyjnej (patrz rozdział 18.). Są to na ogół warsztaty tworzenia elementów rejestru produktu, czyli pisania historyjek (szczegóły na ten temat znajdziesz w rozdziale 5.) — w których udział biorą również wewnętrzni interesariusze, członkowie zespołu deweloperskiego, a czasami także interesariusze zewnętrzni — mające na celu stworzenie rejestru produktu wysokiego rzędu, nadającego się do wykorzystania podczas planowania wersji dystrybucyjnej. W spotkaniu powinni brać udział członkowie zespołu deweloperskiego, ponieważ na tym etapie projekt ma już przyznane środki na realizację. Jeżeli właściwy zespół deweloperski nie jest jeszcze uformowany, w jego miejsce może zostać zaangażowany zespół zastępczy.

Po zakończeniu warsztatów pisania elementów rejestru produktu właściciel produktu uczestniczy w warsztatach oceny rozmiaru (zazwyczaj jest to seria spotkań trwających dzień lub dwa), w trakcie których członkowie zespołu deweloperskiego (właściwego lub zastępczego, jeśli właściwy nie został jeszcze utworzony) przypisują rozmiary elementom wysokiego rzędu.

Dalej właściciel produktu inicjuje wewnętrzna sesję planowania wersji dystrybucyjnej (planowanie długoterminowe). Ponieważ pewna liczba elementów rejestru produktu została już oceniona, celem tej aktywności jest ustalenie priorytetów w rejestrze i zbilansowanie wszelkich ograniczeń wynikających z zakresu i planu prac oraz budżetu (patrz rozdział 18.). Głównymi współuczestnikami tego spotkania są interesariusze, chociaż w pewnym momencie potrzebne będzie zaangażowanie niektórych lub nawet wszystkich członków zespołu deweloperskiego w celu zidentyfikowania ograniczeń technicznych i ich wpływu na uporządkowanie elementów w rejestrze.

Celem jest przeprowadzenie planowania wersji w stopniu wystarczającym do uzyskania dostatecznie przejrzystego obrazu całej wersji dystrybucyjnej, a także do odpowiedzenia na wstępne pytania biznesowe, takie jak co będzie dostarczone i kiedy. W większości przypadków ta aktywność powinna zająć nie więcej niż jeden lub dwa dni. Planowanie wersji dystrybucyjnej jest zajęciem ciągłym (o czym będę pisał w rozdziale 18.), zatem nie powinniśmy przesadzać z inwestowaniem wysiłku w szczegóły na tym etapie. Będziemy uaktualniać plan wersji dystrybucyjnej wraz z pojawianiem się nowych, lepszych informacji.

Po planowaniu wersji zespół scrumowy wykonuje pierwszy sprint (rysunek 9.6 pokazuje dwutygodniowy sprint trwający w tygodniach 5. i 6.). Na początku sprintu właściciel produktu nadzoruje aktywność planowania sprintu (patrz rozdział 19.). Podczas wykonania sprintu (patrz rozdział 20.) właściciel produktu stara się być obecny w trakcie codziennych działań scrumowych. Nie zawsze jest to możliwe, ale takie zachowanie należy traktować jako dobrą praktykę. Podczas codziennych spotkań scrumowych właściciel produktu słucha, chcąc się zorientować w przebiegu sprintu i ewentualnych możliwościach udzielenia pomocy zespołowi deweloperskiemu. Być może któryś z członków wspomni, że nie do końca rozumie szczegóły danego elementu rejestru produktu i potrzebuje dodatkowych wyjaśnień, zanim będzie mógł kontynuować swoje zadanie. Jeśli wyjaśnienie jest krótkie, właściciel produktu może przedstawić je w trakcie spotkania. Jeżeli na odpowiedź trzeba więcej niż kilku sekund, odpowiedź właściciela produktu powinna brzmieć: „Z chęcią zostanę po spotkaniu i omówimy ten temat”.

Właściciel produktu musi być również dostępny (zazwyczaj każdego dnia), aby udzielać odpowiedzi na pytania i testować cechy, kiedy tylko będą one możliwe do przeglądu. Jeżeli właściciel produktu wie, że nie może wykonywać tych obowiązków każdego dnia, musi oddelegować je do odpowiedniej osoby, aby nie doprowadzić do sytuacji, w której zespół zabrnie w ślepy zaulek bez możliwości kontynuacji prac. Do tej idei powrócimy w dalszej części tego rozdziału.

Jednocześnie podczas wykonania sprintu właściciel produktu spotyka się zarówno z wewnętrznyimi, jak i zewnętrznymi interesariuszami, aby upewnić się co do prawidłowości priorytetów na nadchodzący sprint, a także w celu pozyskania cennych informacji od użytkowników, które mogą mieć wpływ na cechy wybrane do realizacji w następnej iteracji.

Właściciel produktu dokonuje również częstej pielęgnacji rejestru produktu, która obejmuje pisanie nowych elementów rejestru produktu i precyzowanie elementów już istniejących, a następnie nadawanie im ocen we współpracy z zespołem oraz priorytetów we współpracy z interesariuszami i zespołem.

Pod koniec sprintu właściciel produktu bierze udział w dwóch aktywnościach inspekcji i adaptacji: przeglądzie sprintu (patrz rozdział 12.) i retrospekcji sprintu (rozdział 22.). Po ich zakończeniu cały cykl zostaje powtórzony, a właściciel produktu zaczyna od udziału w kolejnym planowaniu sprintu.

## Kto powinien być właścicielem produktu?

Większość organizacji niestosujących Scruma prawdopodobnie nie posiada w swoich zasobach kościoła w roli „właściciela produktu”. Kto zatem w organizacji nadaje się do pełnienia takiej roli?

Jak wspomniałem we wcześniejszej części tego rozdziału, właściciel produktu musi patrzyć jednocześnie w dwóch kierunkach: w stronę wewnętrznych i zewnętrznych interesariuszy, a także w stronę zespołu deweloperskiego. Stąd rola właściciela produktu stanowi swego rodzaju połączenie władzy i odpowiedzialności, które historycznie przynależały do kilku tradycyjnych ról. Najbardziej całościowy opis roli właściciela produktu składa się z elementów ról menedżera produktu, specjalisty ds. marketingu produktu, menedżera projektu (omawianego w rozdziale 13.), analityka biznesowego i inżyniera jakości przeprowadzającego testy akceptacyjne.

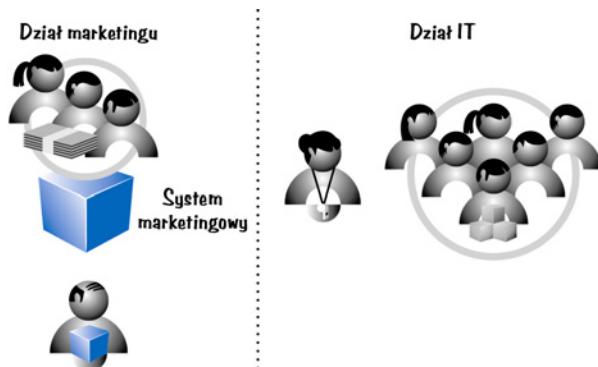
Dokładny wybór osoby na stanowisko właściciela produktu zależy od typu prac deweloperskich oraz specyficznych uwarunkowań organizacji. Tabela 9.1 przedstawia dobre kandydatury na właściciela produktu w zależności od typu prac deweloperskich.

**TABELA 9.1.** Właściciele produktu w zależności od typu wytwarzanego produktu

Typ prac deweloperskich	Kandydat na właściciela produktu
Wewnętrzne prace deweloperskie	Przedstawiciel/klient z obszaru biznesowego będącego beneficjentem tworzonego rozwiązania.
Komercyjne prace deweloperskie	Wewnętrzny przedstawiciel faktycznych klientów i użytkowników (zazwyczaj menedżer produktu, specjalista ds. marketingu produktu, menedżer projektu).
Outsourcing	Przedstawiciel/klient firmy płacącej za rozwiązanie i będącej jej beneficjentem.
Zespół budujący komponenty (architektoniczne prace deweloperskie)	Zazwyczaj inżynier, który jest w stanie najlepiej uporządkować rejestr elementów technicznych pod względem priorytetów.

## Wewnętrzne prace deweloperskie

Właścicielem produktu w przypadku wewnętrznych prac deweloperskich powinna być uprawniona osoba wywodząca się z grupy będącej beneficjentem tych prac. Na przykład: jeśli wewnętrzny dział IT tworzy system dla działu marketingu, właścicielem produktu powinna być uprawniona osoba z zespołu marketingu (patrz rysunek 9.7).



**RYSUNEK 9.7.** Przykład właściciela produktu dla wewnętrznych prac deweloperskich

Niektóre organizacje (zazwyczaj te, które nie zrozumiały jeszcze wagi zatrudniania osoby o wiedzy biznesowej zaangażowanej w codzienną pracę właściciela produktu) mogą poprosić osobę z działu IT o wykonywanie codziennych obowiązków właściciela produktu. Problemy wynikające z tego podejścia opiszę przy okazji omawiania koncepcji zespołu właścicieli produktu, w dalszej części tego rozdziału.

## Komercyjne prace deweloperskie

W przypadku komercyjnego wysiłku deweloperskiego — na przykład firmy budującej produkt przeznaczony do sprzedaży klientom zewnętrznym — właściciel produktu powinien być pracownikiem tej organizacji działającym jako głos rzeczywistych klientów. Bardzo często taka osoba należy w firmie do kadry zarządzającej produktem lub jego marketingiem (patrz rysunek 9.8).



**RYSUNEK 9.8.** Przykład właściciela produktu podczas komercyjnych prac deweloperskich

Uczestnicy Scruma wielokrotnie prowadzili gorące dyskusje na temat tego, czy rola właściciela produktu nie jest tak naprawdę scrumową (lub zwinną) odmianą roli menedżera produktu. Niektórzy uważają, że obie te role są ze sobą tożsame. Inni twierdzą, że rola właściciela produktu jest szersza od roli menedżera produktu. Oczywiście znaleźli się również tacy, w których przekonaniu to rola menedżera produktu jest szersza. Moje poglądy na ten temat wyglądają następująco.

Obszary zarządzania produktem i jego marketingiem są dosyć rozległe. Pragmatic Marketing Inc. — dobrze znana i szanowana firma działająca na polu zarządzania/marketingu — stworzyła wysokiej klasy środowisko definiujące role i obowiązki mające zastosowanie w zespołach zajmujących się zarządzaniem i marketingiem produktów (patrz rysunek 9.9).



**RYSUNEK 9.9.** Środowisko firmy Pragmatic Marketing

W opinii firmy Pragmatic Marketing, aby ogarnąć te aktywności, potrzebne są różne role, włączając w to mistrzów strategii, inżynierijnych menedżerów produktów i marketingowych menedżerów produktów. Większość osób zgodzi się z opinią, że jeśli organizacja o charakterze komercyjnym wymaga realizacji wszystkich tych zadań dla większego produktu, najprawdopodobniej potrzebny będzie zespół ludzi.

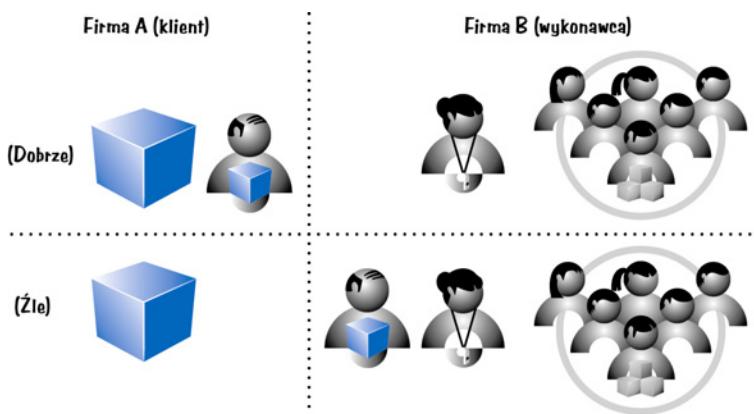
Czy oczekuje się, że wszystkie te zadania będzie realizował właściciel produktu? Zwolennicy traktowania roli właściciela produktu jako podzbioru tradycyjnej roli menedżera produktu argumentują, iż właściciel produktu jest w rzeczywistości zaledwie „inżynierijnym menedżerem produktu” i w związku z tym powinien skupiać się wyłącznie na malej liczbie zadań zaznaczonych na rysunku 9.9 linią przerywaną. Przekonują oni, że skoro właściciel produktu musi być dostępny dla zespołu dzień po dniu, nie będzie on miał czasu, aby skupić się na innych zadaniach.

Właściciel produktu z całą pewnością jest odpowiedzialny za wykonywanie zadań wewnętrz przerywanej linii, ale ja uważam, że na tym nie kończą się jego obowiązki. W moim przekonaniu *rola* właściciela produktu powinna realizować tyle zadań z rysunku 9.9, ile dla się wykonywać, patrząc z praktycznego punktu widzenia. Zakres obowiązków powinien zależeć od typu organizacji, specyfiki produktu i umiejętności osoby wybranej na właściciela produktu. Na przykład organizacja wytwarzająca prostą aplikację do konwersji jednostek przeznaczoną do sprzedaży w sklepie internetowym z aplikacjami nie będzie wymagać tylu czynności co organizacja produkująca kolejną główną wersję dystrybucyjną oprogramowania z zakresu analityki biznesowej. Dlatego definiowanie w sposób uniwersalny dziedziny obowiązków właściciela produktu w odniesieniu do środowiska firmy Pragmatic Marketing jest niepraktyczne.

Niedługo wspomnę o tym, że w niektórych przypadkach zakres obowiązków właściciela produktu może przekraczać możliwości realnego sprawowania ich przez jedną osobę. Wtedy wskazane jest stworzenie zespołu właścicieli produktów, w skład którego wejdą osoby skupiające się na strategii i marketingu. Jednak nawet w takiej sytuacji nadal istnieć będzie tylko jeden człowiek sprawujący rolę właściciela produktu w zespole scrumowym.

## Outsourcing

W przypadku wysiłku deweloperskiego opartego na outsourcingu — na przykład firma A podpisuje kontrakt z firmą B na stworzenie systemu komputerowego — właścicielem produktu powinien być przedstawiciel firmy A. Firma B może wyznaczyć osobę w swojej firmie do bliskiego kontaktu z właścicielem produktu, ale sam właściciel produktu powinien pochodzić z firmy, która płaci za tworzone rozwiązanie i będzie jego beneficjentem (patrz rysunek 9.10).

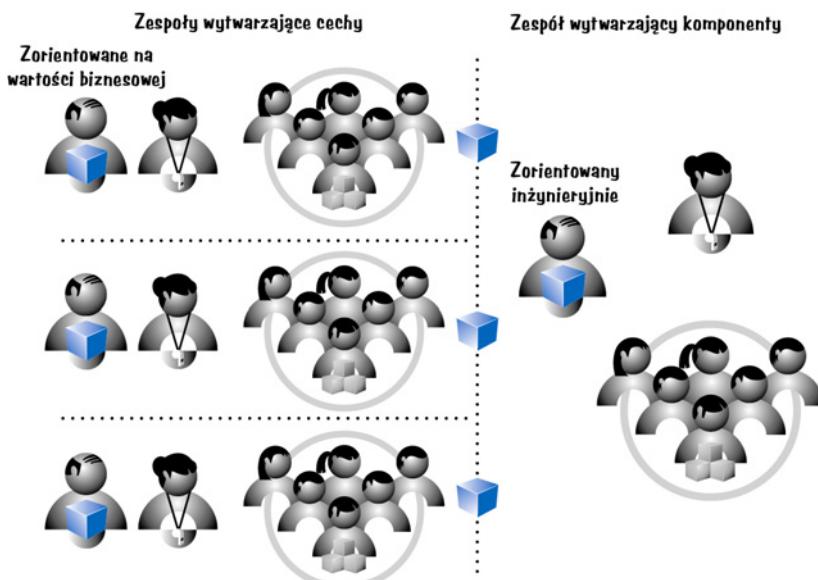


RYSUNEK 9.10. Przykład właściciela produktu w zespole wykonującym outsourcing

Rola właściciela produktu komplikuje się, jeżeli firma A i firma B zawrą tradycyjną formę kontraktu o ustalonej cenie prac deweloperskich. W takim przypadku firma B niemal z całą pewnością będzie czuła, że to ona powinna wykonywać większość obowiązków właściciela produktu, ponieważ to ona przyjęła ryzyko kontraktu o ustalonej wartości. W praktyce to jednak firma A, jako faktyczny klient, powinna pełnić rolę właściciela produktu. Bardziej odpowiednią formą kontraktu w tej sytuacji byłoby wylizingowanie przez firmę A zespołu deweloperskiego o wysokiej wydajności wraz z mistrzem młyna od firmy B i dołączenie do tego zestawu właściciela produktu z firmy A.

## Tworzenie komponentów

W ostatnim już przypadku pewne organizacje mogą użytkować zespoły budujące komponenty (patrz rozdział 12.), które zajmują się wytwarzaniem części rozwiązań dla klienta, ale nie kompletnymi systemami. Tego typu zespoły produkują moduły lub zasoby wykorzystywane następnie przez inne zespoły do złożenia w całość rozwiązań mających wartość dla klienta. Ponieważ takie zespoły skupiają się na poziomie technicznym komponentów, ich właściciele produktu to zazwyczaj osoby o umiejętnościach inżynierijnych umożliwiających definiowanie i nadawanie priorytetów technicznym elementom rejestru produktu (patrz rysunek 9.11).



**RYSUNEK 9.11.** Przykład właściciela produktu w zespole wytwarzającym komponenty

Na rysunku 9.11 przedstawione są trzy zespoły budujące cechy biznesowe mające wartość dla użytkowników końcowych. Każdy taki zespół posiada swojego właściciela produktu skupionego na cechach swojego zespołu. Każdy z zespołów korzysta również z owoców pracy zespołu wytwarzającego komponenty, używając ich rozwiązań jako niezbędnego środka do ukończenia własnych cech. Zespół wytwarzający komponenty potrzebuje własnego właściciela produktu potrafiącego nadać priorytety pracy deweloperskiej i nadzorować jej przebieg. Stąd właściciel produktu w tym zespole musi być lepiej zorientowany technicznie od właścicieli produktów w zespołach tworzących cechy dla użytkownika.

## Właściciel produktu pełniący inne role

Jeżeli pozwala na to czas, dana osoba może realizować zadania właściciela produktu dla więcej niż jednego zespołu scrumowego (patrz rysunek 9.12).



**RYSUNEK 9.12.** Ta sama osoba pełniąca rolę właściciela produktu w więcej niż jednym zespole scrumowym

Zazwyczaj dobrze jest, kiedy osoba taka sprawuje obowiązki właściciela produktu dla zespołów związanych z tym samym wysiłkiem deweloperskim, ponieważ ich praca najprawdopodobniej będzie ze sobą mocno powiązana.

Chociaż możliwy jest przypadek osoby pracującej jednocześnie jako właściciel produktu i członek zespołu deweloperskiego, ogólnie uważa się za zły pomysł, aby ta sama osoba była jednocześnie właścicielem produktu i mistrzem młyna w tym samym zespole. Te dwie role są dla siebie przeciwnią — przypisanie ich jednej osobie tworzy konflikt interesów, którego powinniśmy unikać.

## Zespół właścicieli produktu

Każdy zespół scrumowy musi mieć wyznaczoną jedną osobę pełniąącą rolę właściciela produktu — upoważnioną do wypełniania związanych z nią obowiązków w danym zespole i ponoszenia odpowiedzialności za podejmowane decyzje.

Czy powinniśmy pozwalać na wykonywanie zadań właściciela produktu przez zespół ludzi? Jeżeli przez zespół rozumiemy grupę ludzi, pośród których rozprosiona jest władza do podejmowania decyzji oraz ponoszenia odpowiedzialności za nie, wtedy odpowiedź brzmi: zdecydowanie nie. Aby Scrum był poprawnie zastosowany, ta osoba musi być *jedynym* właścicielem produktu, podejmującym decyzje i stanowiącym głos społeczności interesariuszy przemawiający do zespołu scrumowego.

Mając w pamięci te słowa, niektóre organizacje formują twór zwany „zespołem właścicieli produktu”, ponieważ uważają, że właściciel produktu nie jest w stanie realizować swojej pracy bez posługiwania się grupą ludzi posiadającą odpowiednią wiedzę i zdolności menedżerskie. W innych firmach ilość obowiązków narzuconych właścicielowi produktu może być większa, niż jest w stanie wykonać jedna osoba pracująca na pełny etat. Wtedy właściciel produktu musi scedować pewne obowiązki na innych ludzi. Uformowanie w każdym z tych przypadków zespołu właścicieli produktu jest dopuszczalne, o ile tylko znajdzie się w nim jedna osoba będąca ostatecznym decydentem, a sam zespół właścicieli nie spadnie do rangi komitetu opiniującego, w którym podjęcie jakiejkolwiek decyzji będzie wymagało zatwierdzenia przez większą liczbę osób.

Podejmując decyzję o utworzeniu zespołu właścicieli produktu, zachowaj szczególną ostrożność. Właściciele produktu bez odpowiednich umiejętności pozwalających powierzyć im przywództwo nad centralnym punktem dowodzenia produktem nie potrzebują komitetu doradczego — potrzebują innej roli. Podobnie właściciele produktu zbyt zajęci, aby pełnić przypisane im funkcje, nie będą potrzebowali zespołu. Być może problemem jest to, że organizacja rozpoczęła zbyt wiele różnych projektów deweloperskich jednocześnie lub istnieje zbyt mało właścicieli produktów, aby sprostać obsłudze projektów będących w realizacji.

Inna możliwość to zbyt duży rozmiar budowanego przez nas produktu. Być może należy podzielić go na zbiór mniejszych części i zwiększyć częstotliwość wdrażania wersji. Przy mniejszych rozmiarach jedna osoba będzie w stanie bez problemu wypełnić rolę właściciela produktu. Jeśli źle skonstruowaliśmy nasze zespoły (patrz rozdział 12.) lub kiepsko zaprojektowaliśmy struktury naszych rejestrów produktów, pojedynczy właściciel produktu może mieć problem z realizacją swojej pracy. Upewnij się, czy stworzone przez Ciebie zespoły właścicieli produktu są naprawdę potrzebne i czy nie maskują one zwyczajnie innego problemu. Jeśli tego nie zrobisz, sytuacja ulegnie dalszemu pogorszeniu, co narazi na porażkę cały Twój wysiłek.

## **Pełnomocnicy właściciela produktu**

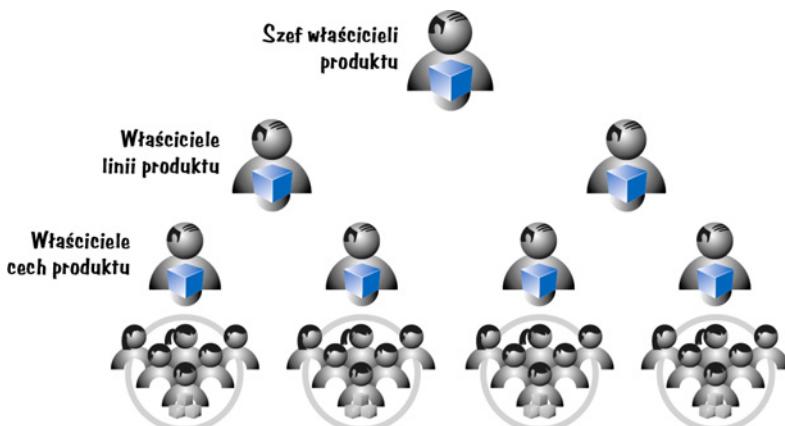
Jak wspomniałem wcześniej, niektóre firmy wykonujące wewnętrzne prace deweloperskie ze względu na brak czasu wśród pracowników z działów biznesowych wyznaczają na właściciela produktu osobę z działu IT (na przykład analityka biznesowego lub menedżera produkcji). Wszyscy wiedzą jednak, że człowiek z działu IT nie posiada dostatecznej władzy, aby podejmować jakiekolwiek ważne decyzje finansowe (czyli wykonywać jedną z kluczowych ról właściciela produktu) i w związku z tym dokonany wybór jest nieoptymalny. Znacznie lepszym rozwiązaniem jest zmiana organizacji pracy tak, aby zarezerwować czas dla pracownika z działu biznesowego i pozwolić mu na wykonywanie rzeczywistej pracy właściciela produktu, natomiast osobę z działu IT ustanowić jego pełnomocnikiem mogącym współdziałać z zespołem w pewnych okolicznościach.

**Pełnomocnik właściciela produktu** to osoba zaangażowana przez właściciela produktu do działania na jego rzecz w pewnych sytuacjach. Każdy członek zespołu scrumowego wie, że pełnomocnik nie jest faktycznym właścicielem produktu, ale wie również, że właściciel produktu dał mu władzę do podejmowania przynajmniej niektórych decyzji taktycznych. Taka sytuacja ma często miejsce, kiedy właściciel produktu bierze udział w licznych spotkaniach z klientami i użytkownikami, aby mieć pewność, że trzyma rękę na pulsie swojego rynku zbytu. Nie może on wtedy zapewnić swojej codziennej obecności w pobliżu zespołu. Chcąc rozwiązać ten problem, właściciel produktu prosi o wsparcie pełnomocnika w kontaktach z zespołem deweloperskim i omawianie z nimi elementów rejestru produktu, gdy zajdzie taka potrzeba.

Funkcjonowanie takiego rozwiązania wymaga, aby właściciel produktu faktycznie udzielił swojemu pełnomocnikowi prawa do podejmowania decyzji i nie podważał ich w sposób, który zburzyłby wiarygodność pełnomocnika w zespole. Pamiętaj, nawet jeśli właściciel produktu wyznaczy innych do asystowania mu, nie może oddelegować ostatecznej odpowiedzialności za wykonanie pracy — to on jest na świeczniku pod tym względem.

## Szef właścicieli produktu

Zespół właścicieli produktu jest często tworzony w przypadku bardzo dużych produktów. Wcześniej napisałem, że jedna osoba może być właścicielem produktu dla dwóch zespołów scrumowych, ale co w przypadku większej liczby zespołów? Przykładem może być trenowana przeze mnie organizacja zatrudniająca około 2500 ludzi. Przy przeciętnym rozmiarze zespołu równym dziesięciu osobom cała organizacja dysponowała ponad 250 zespołami skupionymi na wspólnym zadaniu. Jedna osoba nie może być właścicielem produktu dla 250 zespołów. W praktyce jedna osoba może codziennie współpracować maksymalnie z kilkoma zespołami. W przypadku takim jak ten rola właściciela produktu musi dać się skalować w sposób hierarchiczny, tak jak pokazuje to rysunek 9.13.



**RYSUNEK 9.13.** Hierarchiczna rola właściciela produktu

W ostatecznym rozrachunku *faktycznym* właścicielem całego produktu na rysunku 9.13 jest **szef właścicieli produktu**. Ma on pod sobą zespół właścicieli produktu, których zadaniem jest zapewnienie prawidłowego wypełniania tej roli na kolejnych niższych szczeblach hierarchii. Jeżeli zdecydujesz się na takie rozwiązanie, zadbaj o to, aby każdy właściciel produktu współpracujący z zespołem miał dostateczną władzę do podejmowania większości decyzji na swoim poziomie i nie musiał przekazywać ich w góre hierarchii w celu zatwierdzenia.

## Zakończenie

W tym rozdziale rozwiniętem pojęcie właściciela produktu. Podkreśliłem, że jest on centralnym punktem zarządzania produktem oraz opisałem istotne obowiązki oraz cechy związane z byciem właścicielem produktu. Następnie wskazałem działania podejmowane przez właściciela produktu podczas różnych aktywności scrumowych w trakcie projektu. Dalej przedstawiłem charakterystyki osób nadających się do roli właściciela produktu w różnych typach prac deweloperskich. Wspomniałem o możliwości realizowania przez jedną osobę zadań właściciela produktu w więcej niż jednym zespole scrumowym, a także jeśli zajdzie taka potrzeba, pracowania jednocześnie jako właściciel produktu i członek zespołu deweloperskiego. Rozdział zakończyłem omówieniem idei zespołu właścicieli produktu ze szczególnym uwzględnieniem pośredników i szefa właścicieli produktu. W kolejnym rozdziale zajmiemy się rolą mistrza młyna.



# Rozdział 10

## MISTRZ MŁYNA

---

W tym rozdziale opiszę rolę mistrza młyna. Zaczynę od wskazania celu istnienia tej roli w odniesieniu do innych ról w Scrumie. Następnie opiszę główne obowiązki i pożądane cechy charakteru mistrza młyna. Dalej zilustruję „dzień z życia” mistrza młyna, co doprowadzi do rozwązań na temat tego, czy rola mistrza młyna wymaga zaangażowania na pełny etat. Rozdział zakończę, charakteryzując osoby pełniące zazwyczaj funkcję mistrza młyna.

### Wprowadzenie

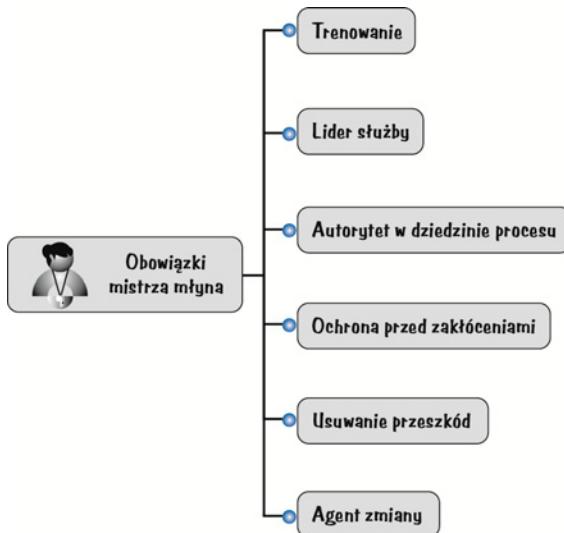
Mistrz młyna jest jedną z trzech ról budujących każdy zespół scrumowy (pozostałe dwie to właściciel produktu i zespół deweloperski). Podczas gdy właściciel produktu jest skupiony na tym, aby zbudować dobrze produkt, zadaniem mistrza młyna jest pomóc wszystkim w zrozumieniu i stosowaniu wartości, reguł i praktyk Scruma. Mistrz młyna jest trenerem zarówno dla zespołu deweloperskiego, jak i właściciela produktu — przewodzi w procesie, pomagając zespołowi scrumowemu, a także całej firmie w wypracowaniu ich własnej, wysoko wydajnej odmiany Scruma.

### Główne obowiązki

Główne obowiązki mistrza młyna przedstawia rysunek 10.1.

### Trener

Mistrz młyna jest w zespole scrumowym trenerem metod zwinnych — zarówno dla właściciela produktu, jak i dla zespołu deweloperskiego (Adkins 2000 zawiera wyczerpujące opracowanie na temat trenera metod zwinnych). Trenując obie strony, mistrz młyna może usuwać bariery pomiędzy tymi rolami i umożliwiać właścielowi produktu bezpośrednie sterowanie produkcją.



**RYSUNEK 10.1.** Główne obowiązki mistrza młyna

Analogicznie do trenera zespołów sportowych mistrz młyna obserwuje sposób wykorzystania Scruma przez zespół i stara się jak może pomóc mu w osiągnięciu kolejnego, wyższego poziomu wydajności. Kiedy pojawią się problemy, które zespół może i powinien rozwiązać samodzielnie, podejście mistrza młyna, tak jak każdego innego dobrego trenera, powinno brzmieć: „Nie jestem tutaj po to, żeby rozwiązywać problemy za was, ale po to, żeby pomóc wam rozwiązać problemy z wami samymi”. Jeżeli problemem jest przeszkoda, której zespół nie jest w stanie ominąć samodzielnie, mistrz młyna przejmuje ją i stara się znaleźć rozwiązanie.

Mistrz młyna trenuje nowego właściciela produktu poprzez pomaganie mu w zrozumieniu i wykonywaniu obowiązków wynikających z roli. Po wytrenowaniu właściciela mistrz młyna służy mu pomocą podczas przeprowadzania pewnych aktywności, takich jak pielęgnacja rejestru produktu. Ponadto zachowując analogię do zespołów sportowych — relacje pomiędzy mistrzem młyną i właścicielem produktu powinny przypominać podstawową relację pomiędzy trenerem drużyną sportową a jej właścicielem: pomaganie właścicielowi w maksymalizacji zysków przy użyciu Scruma, zarządzanie sytuacjami wyjątkowymi, zapewnianie, aby właściciel dostarczał zespołowi to, czego mu potrzeba, a także wsłuchiwanie się w jego skargi oraz żądania zmian, a następnie przekształcanie ich w możliwe do realizacji poprawki w procesie działania zespołu.

## Lider służby

Mistrz młyna jest często opisywany jako **mistrz службы** w zespole scrumowym. Nawet działając jako trener zespołu, mistrz młyna jest w pierwszej kolejności osobą usługującą zespołowi scrumowemu, dbającą o to, aby jego najistotniejsze potrzeby były spełnione. Mistrz służby nigdy nie zapyta: „Co możesz dla mnie dzisiaj zrobić?”. Zamiast tego zapyta: „Co ja mogę dzisiaj zrobić, żeby pomóc Tobie i zespołowi w osiągnięciu lepszej wydajności?“.

## Autorytet w dziedzinie procesu

Mistrz młyna jest autorytatem zespołu scrumowego w dziedzinie procesu. W ramach tego obowiązku mistrz młyna jest upoważniony do zapewnienia, aby zespół scrumowy wykonywał zadania i przestrzegał wartości, zasad oraz praktyki zgodnie z przyjętym przez niego specyficznym podejściem do Scruma. Mistrz młyna nieustannie pomaga zespołowi na każdym kroku w poprawianiu tego procesu i maksymalizowaniu dostarczanej przez niego wartości biznesowej.

Autorytet w tym przypadku nie jest tożsamy z autorytetem, jaki posiadałby menedżer działu lub projektu. Przykładowo: mistrz młyna nie zatrudnia i nie zwalnia pracowników, a także nie może dyktować zespołowi, jakie zadania powinien wykonywać lub też jak ma je wykonywać. Mistrz młyna nie odpowiada za wykonanie pracy. Jego rola to pomaganie zespołowi w zdefiniowaniu i przestrzeganiu jego własnego procesu, który ma zapewnić realizowanie pracy.

## Ochrona przed zakłóceniami

Mistrz młyna chroni zespół deweloperski przed zakłóceniami z zewnątrz, dzięki czemu może on pozostać skupionym w każdym sprincie na dostarczaniu wartości biznesowej. Zakłócenie może mieć wiele źródeł — od menedżerów, którzy chcą przydzielić członkom zespołu inne zadania w środku sprintu, po problemy napływające z innych zespołów. Niezależnie od źródła zakłócenia zadaniem mistrza młyna jest przechwycić problem (odsyłając zapytanie, kierując sprawę do kadry zarządzającej lub negocjując spory) tak, aby zespół mógł w spokoju dostarczyć wartość ze sprintu.

## Usuwanie przeszkód

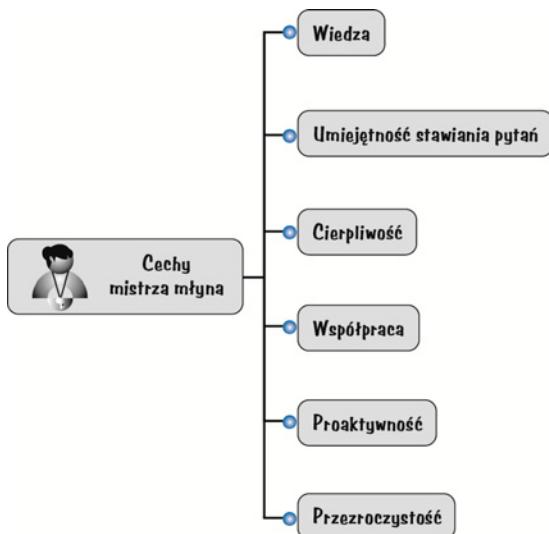
Mistrz młyna bierze również odpowiedzialność za usuwanie przeszkód, które ograniczają produktywność zespołu (w sytuacji, kiedy sam zespół nie jest w stanie w rozsądny sposób ich usunąć). Przykładem może być obserwowany przeze mnie zespół scrumowy, który bardzo często nie był w stanie osiągnąć założonego celu sprintu. Przeszkodą w pracy były niestabilne serwery produkcyjne używane podczas testów (w ramach realizacji definicji ukończenia). Zespół nie posiadał kontroli nad tymi serwerami — zarządzanie nimi należało do obowiązków dyrektora operacyjnego. Ponieważ zespół nie był w stanie samodzielnie usunąć przeszkody, zadanie poprawienia stabilności pracy tych serwerów przejął mistrz młyna, który nawiązał współpracę z dyrektorem operacji. Razem byli oni w stanie podjąć jakieś faktyczne działania w celu usunięcia problemu niestabilności.

## Agent zmiany

Mistrz młyna musi umieć poradzić sobie z czymś więcej niż tylko ze złe działającymi serwerami lub innymi podobnymi usterkami. Dobry mistrz młyna musi pomagać w zmianie mentalności. Scrum może mieć bardzo zakłócające działanie w odniesieniu do istniejącego stanu równowagi — zmiany potrzebne do osiągnięcia sukcesu w Scrumie mogą być trudne. Mistrz młyna pomaga w zrozumieniu innym potrzeby zmiany, wpływu Scruma na otoczenie poza zespołem scrumowym, a także na ogólne zyski, jakie można osiągnąć za pomocą Scruma. To również mistrz młyna dba o to, aby odpowiednie zmiany zachodziły rzeczywiście na wszystkich poziomach organizacji, umożliwiając osiągnięcie nie tylko sukcesu krótkoterminowego, ale również długoterminowego zysku z posiadania Scruma. W dużych organizacjach mistrz młyna może będzie musiał łączyć ze sobą zupełnie przeciwnostawne obozy, aby wytworzyć impuls do zmiany.

## Cechy i umiejętności

Istotne cechy i umiejętności mistrza młyna przedstawia rysunek 10.2.



**RYSUNEK 10.2.** Cechy mistrza młyna

### Wiedza

Chcąc być skutecznym trenerem procesu, mistrz młyna musi posiadać rozległą wiedzę na temat Scruma. Mistrz młyna powinien również rozumieć problemy techniczne, z jakimi boryka się ze sobą, oraz technologie wykorzystywane w produkcji rozwiązań. Nie musi być to wiedza na poziomie лидera technicznego lub lidera produkcji — rozsądny poziom wiedzy technicznej to już duży atut. Mistrz młyna nie musi również być ekspertem w dziedzinie biznesu (od tego jest właściciel produktu), ale pewien poziom praktycznej wiedzy w tym temacie będzie bardzo pomocny.

### Umiejętność stawiania pytań

Mistrzowie młyna stosują swoje umiejętności trenerskie w połączeniu z wiedzą na temat procesu, technologii i biznesu do zadawania kluczowych pytań. Angażują się w wewnętrzne śledztwa, które zmuszają innych do zatrzymania się i stwierdzenia: „Hm, nigdy o tym nie myślałem, ale teraz, kiedy o to pytasz, wydaje mi się, że istnieje inne rozwiązanie”. Wielcy mistrzowie młyna niemal nigdy nie odpowiadają na pytanie w sposób bezpośredni, zamiast tego odpowiadają zawsze kolejnym pytaniem. Nie chodzi tutaj o czystą retorykę czy też poiertywanie przeciwnika, ale o przemyślane, mające sens pytanie pozwalające zgłębić problem i przez to pomóc ludziom w zrozumieniu, że oni sami mają w sobie potencjał do znalezienia odpowiedzi na własne pytania (jest to odmiana egzaminowania w stylu Sokratesa).

## Cierpliwość

Ponieważ mistrzowie młyna preferują powstrzymywanie się od udzielania odpowiedzi, muszą wykazywać się cierpliwością, dając zespołowi czas na ich samodzielne znalezienie. Czasami trudno jest mi pełnić rolę mistrza młyna, ponieważ widzę problem, jaki ma zespół, i „wiem”, jaka powinna być odpowiedź, lub przynajmniej *myślę*, że ją znam! Arogancją z mojej strony (lub ze strony dowolnego innego mistrza młyna) byłoby przekonanie o wyższości własnego intelektu nad zbiorową siłą umysłu całego zespołu. Czasem muszę zwyczajnie ugrzyźć się w język i wykazać cierpliwość, pozwalając zespołowi na wypracowanie rozwiązania, zadając od czasu do czasu pytania, które pomogą mu pójść w dobrym kierunku.

## Współpraca

Mistrz młyna musi posiadać doskonałe umiejętności pracy w grupie, aby móc współpracować z właścicielem produktu, zespołem deweloperskim i innymi uczestnikami, również takimi, którzy nie są bezpośrednio zaangażowani w Scrum. Ponadto jako trener procesu mistrz młyna szukaawsze okazji do wsparcia członków zespołu scrumowego w osiągnięciu stopnia współpracy godnego pozazdrożenia. Wsparcie tego wysiłku jest możliwe przez wykazywanie własnych umiejętności pracy w zespole.

## Proaktywność

Mistrz młyna powinien być bardzo proaktywny wobec zespołu. Często spotykana analogią jest przedstawianie mistrza młyna jako psa pasterskiego chroniącego stado przed wilkami, które mogą chcieć je zaatakować. W naszym kontekście wilkami byłyby przeszkody organizacyjne lub ludzie ze swoimi różnymi celami. Mistrz młyna jest biegły w zapewnianiu ochrony zespołowi w ramach szerszego kontekstu podejmowania ekonomicznie uzasadnionych decyzji biznesowych. Działając z odpowiednim stopniem zrozumienia zarówno potrzeby ochrony zespołu, jak i potrzeb biznesu, mistrz młyna pomaga zespołowi scrumowemu w osiągnięciu właściwej równowagi.

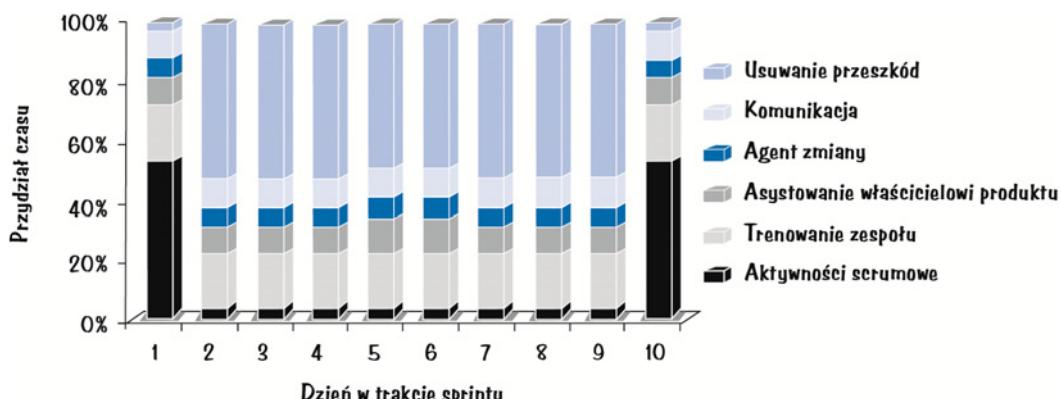
Mistrz młyna pomaga również członkom zespołu oddalającym się od stada. Kiedy sprawy zaczynają się komplikować, ludzie bardzo łatwo przechodzą do starych metod niespełniających zasad zwinności. W takim przypadku rolą mistrza młyna jest pomóc błąkającym się członkom zespołu w pokonaniu napotkanych trudności i ponownym wpojeniu zasad wykorzystania Scruma.

## Przezroczystość

Mistrz młyna stosuje przezroczystość we wszystkich formach komunikacji. Przy pracy w zespole nie może być miejsca na „ukryte negocjacje”. To, co słyszysz od mistrza młyna, jest tym, co dostaniesz. Właśnie tego oczekują ludzie od lidera służby. Mistrz młyna promuje również przezroczystą komunikację poza zespołem scrumowym. Bez niczym nieskrępowanego dostępu do informacji trudno jest organizacji dokonywać inspekcji i adaptacji w celu osiągnięcia rezultatów biznesowych oczekiwanych z wdrożenia Scruma.

## Dzień z życia

Jak dokładnie wygląda życie mistrza młyna podczas sprintu? Rysunek 10.3 pokazuje (z przybliżoną precyją), ile czasu mistrz młyna poświęca każdej z aktywności sprintowych w nowo utworzonym zespole. Wartości procentowe byłyby inne w przypadku mistrza młyna pracującego od kilku lat z zespołem o wysokiej wydajności.



RYSUNEK 10.3. Dzień z życia mistrza młyna

Jak widać na rysunku 10.3., każdego dnia mistrz młyna organizuje i przeprowadza działania scrumowe, włączając w to planowanie sprintu, wykonanie, przegląd i retrospekcję sprintu, a także codzienne spotkania scrumowe. Działania te to zarówno przygotowanie poszczególnych aktywności, nadzorowanie wykonania, a także pomoc pozostałej części zespołu scrumowego w pracy zapewniającej wysoki poziom osiąganych wyników.

Ponadto mistrz młyna każdego dnia trenuje członków zespołu w lepszym wykorzystaniu Scruma oraz ich własnych technik produkcji. Może on również przeprowadzać szkolenia odświeżające, mające na celu na przykład przypomnienie nowemu zespołowi zasad planowania pokerowego podczas głosowania nad elementami rejestru produktu. Pewną część dnia mistrz młyna poświęca na komunikację (na przykład uaktualnienie wykresu spalania lub rozpalania, dyskusję z osobami spoza zespołu scrumowego).

W trakcie sprintu mistrz młyna razem z właścicielem produktu zajmują się pielęgnacją rejestru produktu (na przykład tworzeniem elementów rejestru produktu i nadawaniem im priorytetów). Mistrz młyna współpracuje również z właścicielem w celu osiągnięcia poprawnych ekonomicznie kompromisów odnośnie do istotnych zmiennych, takich jak cechy, daty, budżet czy jakość.

Mistrz młyna spędza również czas na działaniu jako agent zmiany, pomagając organizacji lepiej wykorzystać Scruma w całym obszarze związanym z wytwarzaniem wartości (poczynając od sprzedaży, przez marketing, HR, współpracę z podwykonawcami itd.).

Każdego dnia mistrz młyna poświęca zmienną ilość czasu na usuwanie przeszkód. Czasami osoba pełniąca tę rolę decyduje się wygospodarować na to działanie stały przedział czasu każdego dnia. Oczywiście przeszkody mogą pojawić się w dowolnym momencie, a ich zasięg może być bardzo duży i krytyczny z punktu widzenia czasu. Poradzenie sobie z tą sytuacją wymaga od mistrza młyna dokonania modyfikacji w jego planie dnia.

Wiele zespołów i organizacji wdrażających Scrum już na początku napotyka wiele przeskódek i stara się skupiać na tych bardziej oczywistych i łatwych do rozwiązania. Nie oznacza to wcale, że wszystkie przeskody zostaną pokonane w łatwy sposób. W praktyce zlikwidowanie kolejnego poziomu przeskódek jest o wiele trudniejsze i czasochłonne. Usuwanie przeskódek jest wielką niewiadomą każdego dnia pracy mistrza młyna, która może zupełnie zmienić przydział czasu pokazany na rysunku 10.3.

## Wypełnianie roli

Rozważając rolę mistrza młyna, musimy zdecydować, kto będzie się najlepiej nadawał do jej wypełniania, czy jest to rola wymagająca zaangażowania na pełny etat, a także czy można połączyć ją z innymi rolami scrumowymi lub spoza Scruma. Przeanalizujmy każdą z tych rzeczy oddzielnie.

### Kto powinien być mistrzem młyna?

Organizacje wprowadzające Scrum nie będą dysponować ludźmi, których rolę można by już określić mianem mistrza młyna. Gdzie zatem znajdziemy ludzi do tej roli? Osobiście widziałem wspomnianych mistrzów młyna wywodzących się z wielu różnych istniejących ról. Niektórzy byli poprzednio menedżerami projektów lub produktów (choćżeż menedżerowie produktów znacznie częściej ewoluują do roli właścicieli produktów). Inni mistrzowie młyna wywodzili się spośród deweloperów, testerów i innych osób o zapleczu inżynieryjnym. Jeśli tylko osoba posiada wspomniane przeze mnie wcześniej cechy i jest gotowa podjąć obowiązki roli, może zostać skutecznym mistrzem młyna.

Niektóre organizacje uważają, że mistrzem młyna powinien zostać lider techniczny lub lider zespołu deweloperskiego. Ludzie ci faktycznie mogą stać się doskonałymi mistrzami młyna, ale również dobrze mogą *nie* stanowić najlepszego wyboru do tej roli. Nie bez powodu obdarza się ludzi przywództwem technicznym — są oni bardzo dobrzy w tym, co robią. Rola mistrza młyna nie wymaga eksplotowania pełnego potencjału tak wyasmieritych umiejętności technicznych. Za każdym razem, kiedy liderzy techniczni wykonują pracę mistrza młyna, silną rzeczą wkładającą mniej wysiłku w przewodzenie zagadnieniom technicznym. Stąd przekształcanie ich w mistrzów młyna może mieć negatywny wpływ na techniczną stronę produkcji. Do pytania, czy członek zespołu deweloperskiego może być jednocześnie mistrzem młyna, odniosę się w dalszej części tego rozdziału.

Również menedżerowie działów lub też zespołów ludzi mogą odnieść sukces jako mistrzowie młyna, jeśli tylko posiadają umiejętności potrzebne do wypełniania tej roli. Najlepiej byłoby, gdyby tego typu menedżerowie nie zajmowali się dalej zarządzaniem ludźmi, a przynajmniej nie w zespołach, dla których będą pełnić rolę mistrza młyna. Ponieważ mistrz młyna nie posiada uprawnień menedżera, członkowie zespołu mogą mieć trudności w stwierdzeniu, czy w danym przypadku rozmawiają ze swoim mistrzem młyna, czy też ze swoim menedżerem. Wolę unikać takich sytuacji i nie mieć zespołów, których członkowie podlegają swojemu mistrzowi młyna. W niektórych organizacjach okazuje się to jednak nieuniknione, dlatego musimy się uczyć radzić sobie najlepiej jak potrafimy z ewentualnymi konfliktami interesów.

## Czy mistrz młyna to zajęcie na pełny etat

Każdy zespół scrumowy ma mistrza młyna, ale czy jest to rola na pełny etat? Prawdopodobnie nie. Zespół scrumowy, który pracuje razem od dłuższego czasu i osiąga wysoką wydajność dzięki temu podejściu, nie musi wymagać takiej ilości trenowania jak świeży zespół scrumowy złożony z ludzi, którzy wcześniej nie pracowali ze sobą.

Chociaż mistrz młyna może poświęcać zespołowi coraz mniej czasu każdego dnia w miarę jego dorastania, jego rola nadal pozostaje kluczowym czynnikiem pozwalającym osiągnąć sukces zespołowi scrumowemu w organizacji. Zazwyczaj wraz ze spadkiem zapotrzebowania zespołu na mistrza młyna rośnie jego zaangażowanie w usuwanie przeszkód organizacyjnych, a także w bycie agentem zmian w całym łańcuchu wytwarzania wartości.

W większości przypadków rola mistrza młyna wymaga zaangażowania znaczającej ilości czasu. We wspomnianych wcześniej przypadkach, kiedy nie jest wymagane zaangażowanie na pełny etat, można pomyśleć o połączeniu ze sobą pewnych ról.

## Połączenie roli mistrza młyna z innymi rolami

Jeżeli pozwala na to czas, jedna osoba może być zarówno utalentowanym mistrzem młyna, jak i członkiem zespołu deweloperskiego. Niewykluczone jest jednak, że takie połączenie stanie się źródłem konfliktu interesów. Na przykład: jak powinna zachować się taka osoba w obliczu ważnych zobowiązków mistrza młyna (jak usuwanie przeszkód) i jednocześnie krytycznych zadań deweloperskich? Ponieważ obie rzeczy są równie ważne, jakikolwiek wybór będzie miał negatywny wpływ na wydajność zespołu scrumowego. Sprawę komplikuje jeszcze bardziej fakt, iż przeszkody mogą pojawiać się w sposób zupełnie nieprzewidywalny i mogą wymagać poświęcenia im dużej ilości czasu. W takiej sytuacji bardzo trudne będzie przewidzenie, ile czasu mistrz młyna pracujący jako deweloper będzie miał na faktyczną realizację zadań z rejestru sprintu.

Istnieje jednak inne podejście, które często okazuje się lepsze. Jeżeli mistrz młyna faktycznie posiada wolne zasoby czasowe, może pełnić tę rolę dla więcej niż jednego zespołu scrumowego (patrz rysunek 10.4).



**RYSUNEK 10.4.** Ta sama osoba jako mistrz młyna dla więcej niż jednego zespołu

Bycie mistrzem młyna wymaga nabycia cennych, niezbyt często spotykanych umiejętności. Wolę, kiedy osoba posiadająca takie umiejętności wykorzystuje je w wielu zespołach, zamiast spędzić czas na realizacji obowiązków niebędących domeną mistrza młyna. Jest to jednak tylko moja osobista opinia. Ogólnie mówiąc, nie ma tutaj jednego słusznego rozwiązania, chociaż w specyficznych warunkach działania danej organizacji wybranie jednego z tych podejść może wydawać się bardziej rozsądne.

Jak wspomniałem w rozdziale 9., jedną z niechętnie stosowanych kombinacji jest połączenie ról mistrza młyna i właściciela produktu. Mistrz młyna jest trenerem zespołu scrumowego, co oznacza, że jest również trenerem Scruma dla właściciela produktu. Trudno jest być swoim własnym trenerem. Poza tym właściciel produktu posiada rzeczywistą władzę i może stawiać oczekiwania zespołowi. Z drugiej strony, mistrz młyna często pełni rolę negocjatora pomiędzy żądaniami właściciela produktu a potrzebami i możliwościami zespołu deweloperskiego. Istnienie właściciela produktu i mistrza młyna w jednej osobie wprowadza niepotrzebne zamieszanie w sytuacji, w której bardzo łatwo można go uniknąć.

## Zakończenie

W tym rozdziale opisałem rolę mistrza młyna. Podkreśliłem obowiązki mistrza młyna jako trenera, lidera służby, autorytetu pod względem procesu, osoby chroniącej zespół przed zakłóceniami, a także usuwającej przeszkody i będącej agentem zmian. Następnie omówiłem konieczność posiadania odpowiedniej wiedzy przez mistrza młyna, umiejętności zadawania kluczowych pytań, cierpliwego oczekiwania na rozwiązanie swoich problemów przez zespół, współpracy ze wszystkimi, ochrony przed nadmiernymi zakłóceniami z zewnątrz, a także komunikowania w sposób widoczny i przejrzysty. Następnie opisałem podział obowiązków mistrza młyna na przestrzeni sprintu, aby umożliwić lepsze zrozumienie krytycznego znaczenia tej roli. Na koniec wskazałem, kto w organizacji powinien być mistrzem młyna, niezależnie od tego, czy rola ta powinna być zajęciem na pełny etat, a także w jaki sposób można połączyć rolę mistrza młyna z innymi rolami w Scrumie.

W następnym rozdziale zajmę się rolą, jaką odgrywa w Scrumie zespół deweloperski.



## Rozdział 11

# ZESPÓŁ DEWELOPERSKI

---

W tym rozdziale przedstawię rolę zespołu deweloperskiego. Zacznę od przedstawienia pięciu klu-  
czowych obowiązków dla tej roli, a następnie przejdę do dziesięciu cech, jakie powinien posiadać  
taki zespół.

## Wprowadzenie

Tradycyjny model tworzenia oprogramowania definiuje kilka różnych profesji, między innymi ar-  
chitekta, programisty, testera, administratora bazy danych, projektanta interfejsu użytkownika itd.  
W Scrumie mamy do czynienia z zespołem deweloperskim, czyli zbiorem ludzi, którzy posiadają  
umiejętności odpowiadające tym profesjom. Zespół deweloperski jest jedną z trzech ról w każdym  
zespołe scrumowym. Jego członkowie wspólnie posiadają niezbędne umiejętności potrzebne do  
dostarczania wartości biznesowej oczekiwanej przez właściciela produktu.

Określenie *zespół deweloperski* może wydawać się nie najlepszą etykietką dla ludzi, wśród których  
znajdują się nie tylko deweloperzy. Próbowano również innych nazw, na przykład *zespół dostar-  
czający* (ang. *delivery team*), *zespół projektująco-budujący-testujący* (ang. *design-build-test team*)  
i zwyczajnie *zespół*. Nie wydaje się, aby którakolwiek z tych nazw była bardziej jednoznaczna od *ze-  
spolu deweloperskiego*. Społeczność scrumowa przystała na stosowanie terminu *zespół deweloperski*  
i takiego określenia używam również w książce.

## Zespoły o specyficznej roli

Wiele organizacji jest przyzwyczajonych do celowego podziału różnych typów profesji na wyspe-  
cializowane zespoły. Stąd może istnieć jeden zespół projektantów, jeden deweloperów i jeszcze  
jeden testerów. Dany zespół po ukończeniu pracy przekazuje ją innemu zespołowi, co pozwala na  
ich niemal niezależne funkcjonowanie.

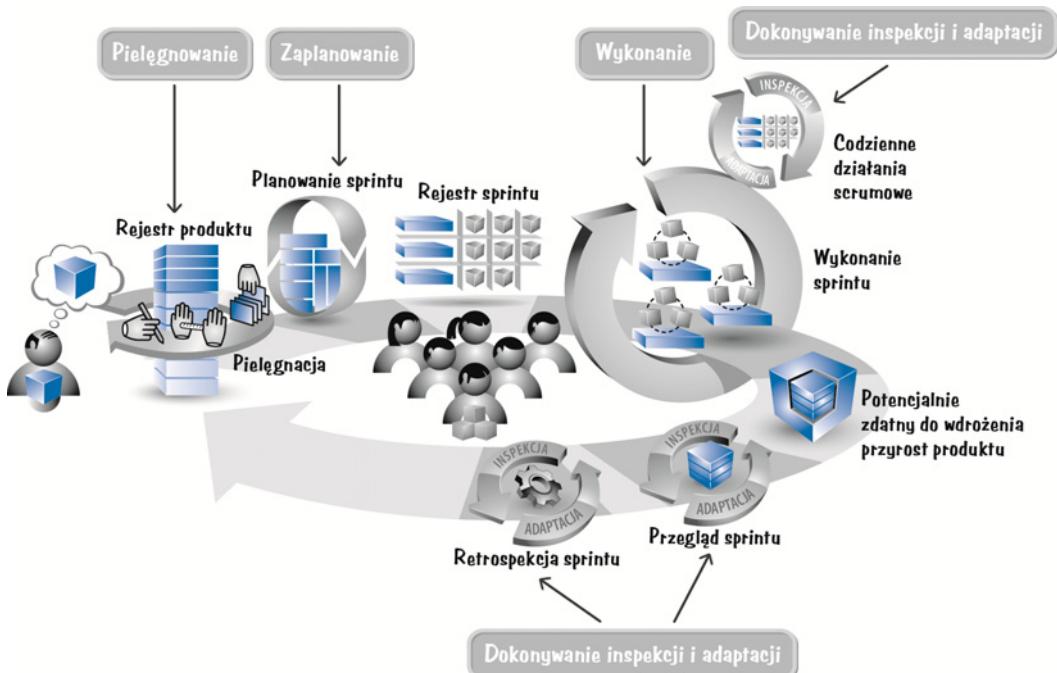
W Scrumie zespół deweloperski musi wykonać całą niezbędną pracę, aby w każdym sprintie wytworzyć jeden lub kilka pionowych wycinków działającej funkcjonalności produktu. Wśród tych prac może znaleźć się projektowanie, programowanie, integracja i testowanie funkcjonalności. Dlatego też potrzebujemy zespołu posiadającego umiejętności wykonywania wszystkich tych zadań.

Niektóre organizacje realizujące Scrum próbują utrzymywać niezależny zespół kontroli jakości. Przyznaję, że w niektórych przypadkach posiadanie niezależnego zespołu skupionego wyłączenie na testowaniu może być niezbędne. Przykładem mogą być regulacje urzędowe, nakazujące utworzenie samodzielnego zespołu do przeprowadzania pewnego typu testów. W większości sytuacji nie ma jednak takiej potrzeby. Testowanie powinno być wplecone w pracę podczas każdego sprintu, czyli powinien wykonywać je zespół deweloperski.

O ile jest to tylko możliwe, powinieneś zawsze tworzyć zespoły wielofunkcyjne. Dzielenie pracy pomiędzy różnymi wyspecjalizowanymi zespołami jest podejrzane i stanowi poważną przeszkodę w pomyślnym wykorzystaniu Scruma. Upewnij się, że masz istotny powód (inny niż przyzwyczajenie) utrzymywania wyspecjalizowanych zespołów.

## Główne obowiązki

Rysunek 11.1 przedstawia aktywności scrumowe opisane kluczowymi obowiązkami zespołu deweloperskiego.



**RYSUNEK 11.1.** Obowiązki zespołu deweloperskiego w odniesieniu do aktywności scrumowych

Opisz kolejno każdy z nich.

## Wykonanie sprintu

W trakcie wykonania sprintu członkowie zespołu deweloperskiego realizują kreatywną pracę projektowania, budowania, integrowania i testowania elementów rejestru produktu. Wynikiem tej pracy jest potencjalnie nadający się do wdrożenia fragment funkcjonalności. Aby osiągnąć ten cel, zespół organizuje się samodzielnie i wspólnie decyduje, jak zaplanować pracę, zarządzać i wykonać ją, a także jak komunikować swoje postępy (patrz rozdział 20.). Zespół deweloperski poświęca większość swojego czasu na wykonanie sprintu.

## Codzienna inspekcja i adaptacja

Od każdego członka zespołu deweloperskiego oczekuje się uczestnictwa w codziennych działańach scrumowych, podczas których cały zespół dokonuje inspekcji swojego postępu w kierunku celu sprintu i adaptuje plan na następny dzień pracy. Jeżeli ktoś jest nieobecny, zespół może nie dostrzec całościowego obrazu sytuacji i z tego powodu nie dać rady osiągnąć celu sprintu.

## Pielęgnacja rejestru produktu

W każdym sprincie należy poświęcić trochę czasu na przygotowania do następnego sprintu. Przeważająca część tego czasu poświęcana jest na pielęgnację rejestru produktu, czyli tworzenie, precyzowanie, nadawanie ocen i ustalanie priorytetów dla elementów rejestru produktu (szczegóły na ten temat znajdziesz w rozdziale 6.). Zespół deweloperski powinien zarezerwować do 10% swojego dostępnego czasu w każdym sprincie na asystowanie właścicielowi produktu w tych działaniach.

## Planowanie sprintu

Zespół deweloperski zaczyna każdy sprint od jego zaplanowania. Pracując wspólnie z właścicielem produktu pod przewodnictwem mistrza młyna, zespół deweloperski ustala cel następnego sprintu. Następnie zespół deweloperski wskazuje podzbior elementów rejestru produktu o wysokim priorytecie, które należy zbudować, aby osiągnąć założony cel (patrz rozdział 19.). W przypadku dwutygodniowego sprintu jego zaplanowanie zajmuje mniej więcej pół dnia. Czterotygodniowy sprint może wymagać poświęcenia całego dnia na planowanie.

Zwróć uwagę, że planowanie odbywa się w sposób iteracyjny. Zamiast skupiać się na bardzo dużym, niepewnym i zbyt szczegółowym planie na początku wysiłku deweloperskiego, zespół przeprowadza serię mniejszych, pewniejszych i bardziej szczegółowych planowań w samą porę na początku każdego sprintu.

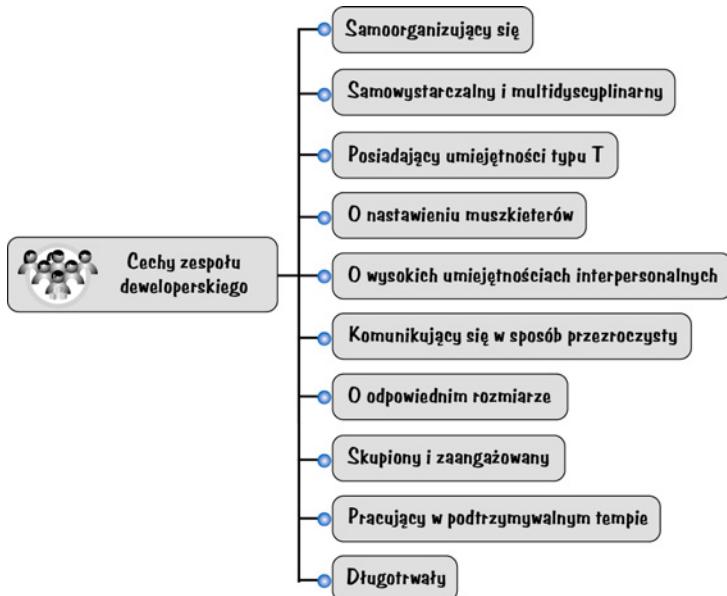
## Inspekcja i adaptacja produktu oraz procesu

Pod koniec każdego sprintu zespół deweloperski bierze udział w dwóch aktywnościach inspekcyjno-adaptacyjnych: przeglądzie sprintu i retrospekcji sprintu. Podczas przeglądu sprintu zespół deweloperski, właściciel produktu, mistrz młyna, interesariusze, sponsorzy, klienci i zainteresowani członkowie innych zespołów przeglądają dopiero co ukończone w bieżącym sprincie cechy i omawiają najlepszą ścieżkę dalszego postępowania (patrz rozdział 21.). Retrospekcja sprintu poświęcona

jest na przedyskutowanie przez zespół scrumowy procesu oraz swoich praktyk technicznych i poprawienie wykorzystania własnej implementacji Scruma do dostarczania wartości biznesowej (patrz rozdział 22.).

## Cechy i umiejętności

Rysunek 11.2 przedstawia istotne cechy i umiejętności zespołu deweloperskiego.



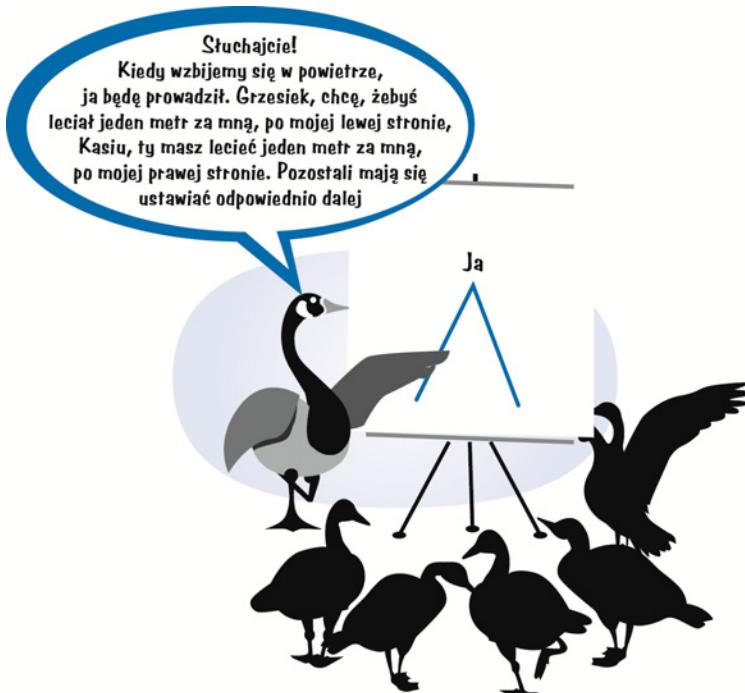
**RYSUNEK 11.2.** Cechy zespołu deweloperskiego

### Samoorganizacja

Członkowie zespołu organizują się samodzielnie w celu jak najlepszej realizacji celu sprintu. Nie ma menedżera projektu lub też innego menedżera, który mówiłby zespołowi, jak ma wykonywać swoją pracę (mistrz młyna z kolei nie powinien nigdy próbować działań menedżerskich). **Samoorganizacja** jest samodzielnie powstającą od dołu właściwością systemu — nie ma tutaj zewnętrznej, dominującej siły narzucającej tradycyjny system przekazywania poleceń i nadzorowania z góry w dół.

Pozwolę sobie zilustrować to przykładem. Niedaleko mniejsza, w którym mieszkam w Colorado, jest mały staw. Zimą przylatuje tam i zamieskuje stado gęsi kanadyjskich. Zatem każdego roku mamy tam kilkaset gęsi, które razem tworzą wielki bałagan, ale przyjemnie jest na nie popatrzeć. Dodam jeszcze, że posiadam dwa psy o imionach Letti i Toast. Zazwyczaj pozostają one w ogrodzie za siatką. Czasem pozwalam im pobiegać poza ogrodzeniem — kiedy zobaczą gęsi przy stawie, biegą jak najszybciej, aby je przywitać. Nie sądzę, aby chciały wyrządzić jakąkolwiek krzywdę gęsiom, ale kiedy te zobaczą nadbiegające psy, decydują się oddać im czasowo staw we władanie i masowo odlatują.

Czy zastanawiałeś się kiedyś, skąd ptaki wiedzą, że po poderwaniu do lotu powinny utworzyć charakterystyczny kształt litery V (klucz żurawi)? Czy sądzisz, że nad moim stawem jest „ptak menedżer” z tablicą, który organizuje spotkania, aby poinstruować wszystkie ptaki, jak utworzyć klucz (patrz rysunek 11.3)?



**RYSUNEK 11.3.** Tworzenie formacji klucza nie jest wynikiem planowania z góry na dół

Mieszkam nad tym stawem od wielu lat i nie przypominam sobie, aby takie spotkanie kiedykolwiek miało miejsce (chociaż wiele lat temu mój syn Jonah stwierdził: „Tato, nie wiedziałeś nigdy tego spotkania, ponieważ gęsi spotykają się nocą!”. Hmm. Może coś w tym jest).

Prawda jest jednak taka, wbrew podejrzeniom mojego syna co do przebiegłości ptaków, że gęsi tworzą klucz poprzez samoorganizację i właściwości emergencji **złożonych systemów adaptacyjnych** z dołu do góry. W systemach takich wiele jednostek współdziała ze sobą na różne sposoby strzeżone poprzez proste, zlokalizowane reguły, w otoczeniu nieustannej informacji zwrotnej (patrz rysunek 11.4).

Tego typu systemy wykazują ciekawe cechy — między innymi są bardzo trwałe i potrafią budować rzeczy niesamowicie oryginalne.

Zespół deweloperski, podobnie do klucza ptaków, nie posiada hierarchicznej, biegnącej z góry na dół struktury wydawania poleceń i nadzorowania, która wskazywałaby zespołowi, jak ma wykonywać swoją pracę. Zamiast tego zespół ludzi o przekrojowych umiejętnościach organizuje się samodzielnie w sposób pozwalający jak najlepiej wykonać powierzoną pracę. W zespole powstaje coś na wzór klucza żurawi.



**RYSUNEK 11.4.** Tworzenie formacji klucza: proste zasady i częsta informacja zwrotna

Menedżerowie odgrywają jednak pewną witalną rolę w Scrumie. Tworzą (i podtrzymują) środowisko dla samoorganizującego się zespołu. Więcej informacji na temat roli menedżerów znajduje się w rozdziale 13.

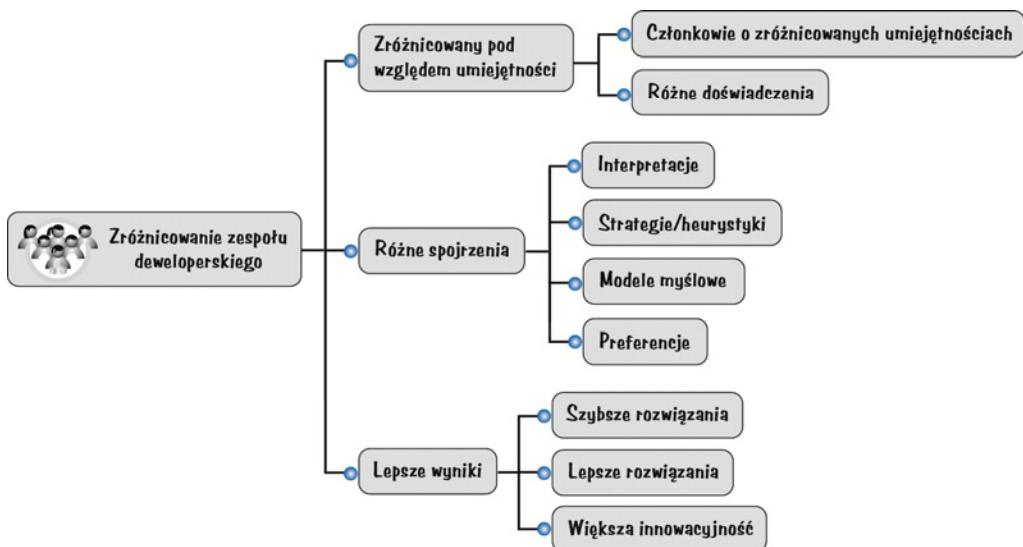
## Różnorodny pod względem umiejętności i samowystarczalny

Członkowie zespołu deweloperskiego powinni być zróżnicowani pod względem pełnionych funkcji. Wspólnie powinni posiadać niezbędny i wystarczający zbiór umiejętności do wykonania przydzielonej pracy. Dobrze uformowany zespół może pobrać element z rejestru produktu i wyprodukować cechę o dobrej jakości spełniającą definicję ukończenia ustanowioną przez zespół scrumowy.

Zespoły składające się z ludzi o tych samych umiejętnościach (tradycyjne zespoły silosowe) mogą wykonać co najwyżej część pracy. W związku z tym zespoły takie muszą przekazywać wyniki swojej pracy innym zespołom silosowym. Na przykład zespół deweloperów przekazuje zbudowany kod zespołowi testerów, a zespół projektujący interfejs użytkownika przekazuje projekty formatek zespołowi zajmującemu się logiką biznesową. Przekazywanie pracy pomiędzy zespołami stanowi doskonałą okazję do źle zrozumianej komunikacji i kosztownych pomyłek. Tworzenie zespołów o zróżnicowanych umiejętnościach nie stoi w sprzeczności z posiadaniem kilku członków wykwalifikowanych w tej samej dziedzinie, na przykład programistów Javy lub C++ czy też testerów.

Zespoły o umiejętnościach przekrojowych potrafią spojrzeć na problem z wielu perspektyw, co daje lepszy wynik końcowy (patrz rysunek 11.5).

Wielofunkcyjny zespół ma członków o różnym doświadczeniu. Każdy z nich wnosi ze sobą zestaw narzędzi poznanowych służących do rozwiązywania problemów. Narzędziami tymi mogą być odmienne sposoby interpretacji (tych samych danych), różne strategie (lub heurystyki) rozwiązywania problemów, różne modele myślenia na temat działania rzeczy, a także różnorodne preferencje podejść i rozwiązań. Tego typu różnorodność prowadzi zazwyczaj do powstawania lepszych wyników, czyli szybszych rozwiązań o wysokiej jakości i lepszej innowacyjności, co przekłada się na większą wartość ekonomiczną [Page, 2007].



**RYSUNEK 11.5.** Zróżnicowanie zespołu

Powinniśmy również dbać o różnorodność poprzez łączenie ze sobą w jednym zespole ludzi na poziomie juniorów i seniorów. Zbyt wielu ludzi na poziomie seniorów może powodować niepotrzebne zawirowania, tak jak obsadzenie wielu kucharzy w jednej kuchni. Z kolei zbyt duża liczba juniorów w zespole może skutkować brakiem dostatecznych umiejętności do wykonania zleconej pracy. Dobra mieszanka stanowi obietnicę zdrowego środowiska promującego kolektywne zdobywanie wiedzy.

## Umiejętności typu T

Elastyczne zespoły deweloperskie składają się z członków o umiejętnościach typu T (patrz rysunek 11.6).



**RYSUNEK 11.6.** Umiejętności typu T

**Umiejętności typu T** oznaczają, że członek zespołu (niech będzie to Sue) posiada dogłębne umiejętności w preferowanym przez siebie obszarze funkcjonalności lub specjalizacji. Na przykład Sue jest doskonałym projektantem z zakresu doświadczeń użytkownika (UX) — to jest jej specjalizacja i obszar, w którym najczęściej pracuje. Sue może jednak pracować również poza swoim klu-czowym obszarem specjalizacji, wykonując testy lub dokumentację. Nie jest tak dobrym testerem i specjalistą ds. dokumentacji jak ci, którzy specjalizują się w tych obszarach, ale może pomóc, jeśli właśnie tam zespół doświadcza wąskiego gardła i musi zebrać się w sobie, aby wykonać pracę.

Wydaje się niemożliwe, aby każda osoba w zespole mogła pracować nad każdym zadaniem. Jest to wzniosły cel do osiągnięcia. Na przykład w domenach o bardzo silnej specjalizacji, takich jak programowanie gier komputerowych, gdzie zespół wymaga posiadania artysty, animatora, inżyniera dźwięku, programisty sztucznej inteligencji i testera, trudno będzie przyjąć, że każdy może wykonywać dowolną pracę. Gdybym to ja znalazła się w takim zespole, mógłbym pracować nad sztuczną inteligencją i wykonywać pewne testy, ale nie zająłbym się tworzeniem projektu artystycznego (a Ty nie chciałbyś, żebybym się nim zajął!). Móglbym natomiast pomóc designerowi z pracą poza-artystyczną, taką jak użycie Photoshopa do konwersji formatu plików, lub w stworzeniu skryptów realizujących zadania na wielu plikach.

Menedżerowie powinni skupić się na formowaniu zespołów posiadających najlepszy zestaw umiejętności typu T, jaki tylko da się uzyskać w oparciu o dostępne zasoby ludzkie. Może się jednak okazać, że uzyskanie na samym początku zespołu o oczekiwanych umiejętnościach będzie niemożliwe i w związku z tym niezbędny będzie jego ewolucyjny rozwój w miarę wzrostu potrzeb wynikających z trwającego wysiłku deweloperskiego. Stąd też kluczowe znaczenie ma posiadanie środowiska pozwalającego ludziom na nieustanne uczenie się i rozwijanie swojego zestawu umiejętności, niezależnie od tego, czy będzie to wiedza domenowa, umiejętności techniczne albo konsepcyjne, czy też inne zdolności. Kadra menedżerska powinna wspierać członków zespołu, oferując im czas na naukę i eksperymentowanie (patrz rozdział 13.).

Czy dopuszczalne jest posiadanie „czystych” specjalistów w zespole? Weźmy pod uwagę nasz poprzedni przykład Sue i założmy, że jest ona świetnym projektantem UX, ale potrafi robić tylko tę jedną rzecz, a ponieważ mamy tak niewielu projektantów UX, nie chcemy, aby zajmowała się ona czymkolwiek innym oprócz swojej dziedziny specjalizacji. Potrzebujemy jej umiejętności w zespole, ale ilość przypadającej na nią pracy w zespole zapewni zaledwie 10-procentowe wykorzystanie jej czasu. W takim przypadku oczywistym wydaje się, aby podzielić czas Sue między różne zespoły.

Musimy jednak podejść do tego w sposób praktyczny. Sue byłaby zbyt „rozbita”, dzieląc swój czas na 10-procentowe fragmenty w wielu zespołach jednocześnie. Wkrótce stałaby się wąskim gardłem w przepływie pracy (patrz sekcja „Skupienie i zaangażowanie” w dalszej części tego rozdziału). Przypomnij sobie z rozdziału 3., że naszym celem nie powinno być utrzymywanie ludzi takich jak Sue na 100-procentowym poziomie wykorzystania. Bardziej powinniśmy się martwić o pracę czekającą na realizację (pałeczkę leżącą na ziemi), która pojawią się, gdy zbyt mocno zależymy od osoby pracującej na dużym poziomie zaangażowania. Możemy zatem przypisać Sue jako specjalistkę do rozsądnej liczby produktów, ale nie aż tak dużej, aby była ona przyczyną upadania pałeczek.

Ponieważ naszym celem jest osiągnięcie dobrego przepływu pracy dzięki członkom zespołu o dobrych umiejętnościach typu T, możemy ewentualnie zachęcić Sue do pomocy innym osobom w nabyciu rozsądnego poziomu wiedzy na temat projektowania UX, dzięki czemu nie będziemy musieli dłużej tak mocno zależeć od specjalistów.

Podsumowując, naszym celem jest uformowanie zespołu z osób posiadających umiejętności w kluczowych obszarach i dodatkowo pokrywających się ze sobą pod względem własnych umiejętności, dzięki czemu zapewniamy pewną elastyczność całemu zespołowi. Aby można było osiągnąć ten cel, wielu członków zespołu musi posiadać umiejętności typu T, chociaż nadal możemy potrzebować specjalistów w tym zestawie.

## Postawa muszkieterów

Członkowie zespołu deweloperskiego (a także całego zespołu scrumowego!) muszą mieć takie samo nastawienie jak słynni powieściowi trzej muszkieterowie — „Wszyscy za jednego, jeden za wszystkich”. Ta **postawa muszkieterów** uwydatnia fakt, iż członkowie zespołu ponoszą wspólną odpowiedzialność za wykonanie pracy. Wygrywają jako zespół lub przegrywają jako zespół.

W dobrze funkcjonującym zespole scrumowym nigdy nie spodziałybym się usłyszeć: „Ja zrobiłem swoją część, ale ty nie zrobiłeś swojej i dlatego poniesliśmy porażkę”. Takie stwierdzenie całkowicie przeczy przekonaniu, iż wszyscy członkowie zespołu płyną tą samą łódką (patrz rysunek 11.7).



**RYSUNEK 11.7.** Członkowie zespołu muszą działać tak, jakby wszyscy płynęli jedną łódką

Członkowie zespołu muszą rozumieć, że pracują razem w celu osiągnięcia podjętego wspólnie zobowiązania, ponieważ jeśli przegrają, będzie to problem ich wszystkich na końcu sprintu. Istnienie członków zespołu o nastawieniu muszkieterów ma kluczowe znaczenie dla osiągnięcia wspólnego sukcesu.

Istnienie członków zespołu z umiejętnościami typu T wspiera to podejście i pozwala na praktyczną realizację dzięki temu, że ludzie mogą pracować nad więcej niż jednym typem zadania. W takich zespołach nie spodziewam się usłyszeć „To nie jest moja działka” od kogoś, kto może wykonać dane zadanie.

Ponieważ jednak nie zawsze jest możliwe, aby dana osoba realizowała dowolne zadanie, można spodziewać się stwierdzenia: „Nie jestem w stanie wykonać tego zadania”. W takiej sytuacji zespół może zdecydować, aby osoba bez danej umiejętności podjęła naukę u osoby, która tę umiejętność posiada, i dzięki temu w przyszłości mogła pochwalić się szerszymi zdolnościami.

Nawet jeśli brak umiejętności uniemożliwia osobom pracowanie w sposób wielofunkcyjny, zespół nadal ma możliwość organizowania swojej pracy tak, aby zapewnić dobry przepływ w sprincie bez przeciążania zbytnio którykolwiek z członków. Na przykład wstrzymywanie testowania aż do końca sprintu, tak aby mógł je przejąć „tester”, gwarantuje niemal pewną porażkę. Więcej na temat sposobów zarządzania przepływem pracy w trakcie wykonywania sprintu znajdziesz w rozdziale 20.

Zatem podejście muszkieterów gwarantuje, że nikt nie jest sam w swojej podróży. Każdy członek zespołu ma obowiązek zadania o to, aby wszyscy byli w pełni zaangażowani przez cały czas. Oznacza to najczęściej konieczność aktywnej komunikacji z innymi i angażowanie się w aktywności wykraczające poza własne specjalizacje — działania dywersyfikujące dyskusję. Na przykład: jeśli członek zespołu specjalizujący się w testowaniu widzi problem w projekcie cechy stworzonym przez zespół, ma obowiązek zakomunikować swoją wątpliwość. Niedopuszczalne jest powiedzenie sobie „To nie należy do moich obowiązków, oni wiedzą lepiej niż ja, co trzeba zrobić”.

## Komunikacja szerokopasmowa

Członkowie zespołu deweloperskiego powinni komunikować się ze sobą, a także z właścicielem produktu i mistrzem młyna w sposób „szerokopasmowy”. Oznacza to częstą i szybką wymianę wartościowych informacji w sposób wydajny i mało kosztowny.

Komunikacja szerokopasmowa zwiększa częstotliwość i jakość współdzielonych informacji. Dzięki temu zespół scrumowy ma więcej okazji do inspekcji i adaptacji i w rezultacie podejmuje lepsze decyzje w krótszym czasie. Ponieważ ekonomiczna wartość informacji ma ścisły związek z czasem, przyśpieszenie tempa wymiany informacji pozwala zespołowi na maksymalizację ich wartości. Możliwość szybkiej eksploracji pojawiących się okazji i rozpoznawanie sytuacji wprowadzających marnotrawstwo pozwala zespołowi unikać alokowania nadmiernych zasobów w ścieżki nieprowadzące do sukcesu.

Istnieje wiele sposobów prowadzenia szerokopasmowej komunikacji przez zespół. Manifest Zwinności [Beck i in., 2001] mówi, że preferowanym podejściem jest komunikacja twarzą w twarz. Członkowie zespołu oddalenie fizyczne od swojego zespołu lub używający głównie nieinteraktywnych środków komunikacji (takich jak dokumenty) są w gorszej sytuacji od ludzi zlokalizowanych w tym samym biurze i mogących komunikować się ze sobą bezpośrednio w czasie rzeczywistym.

Kiedy jest tylko możliwe, wolę, aby członkowie mojego zespołu byli zlokalizowani w jednym miejscu. Jednak organizacje ze względów biznesowych tworzą zespoły rozproszone, dlatego kolokacja nieawsze jest możliwa lub opłacalna. Pracowałem wielokrotnie z zespołami, którym udało się skorzystać z dobrodziejstw szerokopasmowej komunikacji, stąd wniosek, iż możliwość porozumiewania się twarzą w twarz nie jest jedynym środkiem do osiągnięcia tego celu, chociaż warto zacząć właśnie w ten sposób, jeśli tylko pozwalały na to warunki.

W poszerzaniu możliwości komunikacyjnych rozproszonych zespołów pomocna jest nowa technologia. Miałem okazję pracować w organizacjach, których członkowie byli rozrzucone po całym świecie. Dzięki zastosowaniu wysokiej klasy urządzeń do telekonferencji mogłem brać udział w dyskusjach, które dawały wrażenie, jakby wszyscy siedzieli w jednym pokoju. Czy było to aż tak dobre jak kolokacja? Nie. Z drugiej strony jednak, technologia przebyła długą drogę, jeśli chodzi o poszerzanie przepustowości komunikacji pomiędzy członkami zespołu.

Posiadanie zespołów składających się z członków o różnorodnych umiejętnościach jest kluczowym krokiem na drodze do osiągnięcia szerokopasmowej komunikacji. Zespoły takie mają lepiej zoptymalizowane kanały komunikacji dzięki dostępowi do ludzi potrzebnych do wykonania zadanej pracy. Ponadto zdywersyfikowane zespoły stosują o wiele mniej formalności podczas przekazywania informacji innym zespołom (zazwyczaj w formie spisywanych dokumentów). Ponieważ wszyscy pracują w tym samym zespole, spada częstotliwość przekazywania informacji oraz konieczność stosowania formalnego protokołu, co poprawia prędkość komunikacji.

Powinniśmy również redukować czas poświęcany ceremoniom, które nakazują członkom zespołu przeprowadzanie procesów przynoszących znikomą lub wręcz zerową wartość. Oto przykład: jeśli członkowie zespołu muszą przejść przez trzy poziomy przekierowania, zanim będą mogli porozmawiać z faktycznym klientem lub użytkownikiem, ceremonia „rozmawiania z klientem” jest prawdopodobnie poważną przeszkodą do osiągnięcia szerokopasmowej komunikacji. Konieczność tworzenia dokumentów o małej wartości oraz zdobywania licznych i potencjalnie niepotrzebnych pozwoleń ogranicza przepustowość komunikacji. Musimy wynajdywać i eliminować tego typu przeszkody, poprawiając tym samym wydajność komunikacyjną zespołu.

Na przepustowość komunikacji ma wpływ również mały rozmiar zespołu. Liczba kanałów komunikacyjnych wewnętrz zespołu nie rośnie liniowo wraz ze wzrostem liczby członków zespołu, ale podnosi się w tempie potęgowym liczby członków, zgodnie z formułą  $N(N-1)/2$ . Zatem jeśli w zespole jest 5 osób, istnieje 10 kanałów komunikacji. Przy 10 osobach w zespole liczba kanałów komunikacji rośnie do 45. Większa liczba ludzi oznacza większy narzut komunikacyjny i zmniejszoną przepustowość.

## Przezroczysta komunikacja

Oprócz wysokiej przepustowości (czyli szybkości i minimalnego narzutu) komunikację w zespole powinna również cechować przezroczystość. Taka forma komunikacji pozwala zrozumieć, co się dzieje, nie kryje żadnych niespodzianek i pomaga budować zaufanie pomiędzy członkami zespołu. Zawsze czułem, że zespoły powinny komunikować się zgodnie z **zasadą najmniejszego zaskoczenia**. Mówiąc najprościej, ludzie powinni informować się w sposób gwarantujący jak najmniejsze zaskoczenie wśród innych. Przypominam sobie pewien trenowany przeze mnie zespół scrumowy, w którym jeden z członków nieustannie unikał informowania o swoich osiągnięciach i planach dalszej pracy. Ludzie byli nieustannie zaskakiwani, dowiadując się później, że jego sposób przekazywania informacji był celowo niezrozumiały i wprowadzający w błąd. Takie zachowanie powodowało brak zaufania członków zespołu do tej osoby, a to z kolei ograniczało możliwość samoorganizacji zespołu i sprostania celom sprintu.

## Prawidłowy rozmiar

W Scrumie preferowane są zespoły o małych rozmiarach. Ogólna zasada mówi, że najlepszy rozmiar zespołu zawiera się w przedziale od pięciu do dziewięciu osób. Istnieją badania, które potwierdzają tezę o tym, że największą wydajność osiągają małe zespoły [Putnam, 1996; Putnam i Myers, 1998]. Na podstawie mojego 25-letniego doświadczenia mogę powiedzieć, że zespoły najbardziej sprawnie dostarczające wartość biznesową mają rozmiar od pięciu do siedmiu osób.

Mike Cohn oferuje listę powodów przemawiających za małym rozmiarem zespołów, wśród nich są następujące:

- mniejsza jest skala tzw. próżniactwa społecznego — sytuacji, gdy ludzie wkładają mniej wysiłku, myśląc, że inni nadrobią ich zaległości;
- w mniejszym zespole istnieje większa szansa na pojawienie się konstruktywnej interakcji;
- mniej czasu poświęca się na koordynację działań;
- nikt nie jest w stanie wtopić się w tło;
- mniejsze zespoły są bardziej satysfakcyjne dla ich członków;
- istnieje mniejsza szansa na pojawienie się szkodliwego wpływu nadmiernej specjalizacji.

Możliwe jest istnienie zespołu zbyt małego. Przykładem mogą być zespoły, w których brakuje osób niezbędnych do realizowania przydzielonej pracy lub do wykonywania obowiązków w sposób wydajny.

Fakt, iż Scrum preferuje zespoły o małych rozmiarach, nie oznacza wcale, że nie możemy użyć go przy większych projektach deweloperskich. Scrum jest bardzo często stosowany do wytwarzania produktów wymagających pracy więcej niż dziewięciu osób. Zamiast jednak tworzyć jeden duży zespół scrumowy składający się, powiedzmy, z 36 członków zespołu deweloperskiego, zorganizowalibyśmy cztery lub więcej zespołów scrumowych, z których każdy miałby do dziewięciu członków zespołu deweloperskiego.

Projekt scrumowy jest skalowalny nie dzięki większemu zespołowi deweloperskiemu, ale dzięki możliwości tworzenia wielu zespołów scrumowych. Zespoły takie w miarę potrzeby koordynują się na szereg różnych sposobów. Jednym z takich sposobów jest **scrum scrumów**, czyli zebranie przedstawicieli wszystkich zespołów scrumowych w celu przeprowadzenia aktywności odpowiadających codziennym działaniom scrumowym (więcej szczegółów na ten temat znajdziesz w rozdziale 12.).

## Skupienie i poświęcenie

Członkowie zespołu powinni być skupieni na celu sprintu i poświęcić jego realizacji. Skupienie oznacza, że każdy członek zespołu jest zaangażowany, skoncentrowany i poświęca swoją uwagę wspólnemu celowi sprintu. Poświęcenie oznacza, że niezależnie od tego, czy idzie dobrze, czy źle, członek pragnie z jednakowym zapałem osiągnąć wspólny cel.

Jednej osobie o wiele łatwiej jest skupiać się i poświęcać, jeżeli pracuje wyłącznie nad jednym produktem. Prosząc taką osobę o pracę nad kilkoma produktami jednocześnie, zmuszamy ją do dzielenia swojego czasu pomiędzy różne projekty i w konsekwencji zredukowania uwagi i poświęcenia wkładanego w każdy z nich.

Zapytaj dowolnego pracownika pracującego nad kilkoma produktami o jego stopień skupienia i zaangażowania, a usłyszysz zapewne coś takiego: „Mam tyle rzeczy do zrobienia, że w każdym produkcie staram się zwyczajnie zrobić wszystko najlepiej jak potrafię i zaraz przeskakuję do następnego produktu. Nigdy nie mam poczucia, że mam wystarczająco dużo czasu, aby skupić się dostatecznie dobrze na którymkolwiek z nich i zrobić to, co mam do zrobienia, naprawdę dobrze. Jeśli kiedyś wystąpi sytuacja awaryjna w kilku produktach jednocześnie, nie będę zwyczajnie w stanie pomóc wszystkim jednocześnie”.

Członkowi zespołu trudno jest wykonywać pracę na dobrym poziomie jakości, kiedy nieustannie przeskakuje z produktu na produkt. Jeszcze trudniej jest poświęcić się rzeczywiście kilku produktom w tym samym czasie. Zamiast siedzieć w jednej łódce ze swoimi członkami zespołu, wielozadaniowy pracownik przeskakuje z łodzi do łodzi. Jeżeli w kilku łodziach jednocześnie pojawi się wyciek, w jaki sposób pracownik ten będzie miał wybrać, której załodze należy pomóc? Jeżeli ta osoba nie siedzi w łodzi, pomagając wylewać wodę, nie jest ona zaangażowana w cel zespołu, a co najwyżej *bierze udział* w jego pracach. Właściwym podejściem dla osoby biorącej udział w pracach innego zespołu jest danie jasno do zrozumienia pozostałym członkom, że w sytuacji kryzysowej jej pomoc może być niemożliwa.

Istnieje spory zasób badań potwierdzających powszechnie znany pogląd, iż praca nad kilkoma produktami (lub projektami) jednocześnie lub też w kilku zespołach jednocześnie redukuje produktywność. Przykładowe dane tego typu przedstawia wykres na rysunku 11.8 [Wheelwright i Clark, 1992].



RYSUNEK 11.8. Koszt pracy symultanicznej

Dane te wskazują, że nikt nie osiąga produktywności na poziomie 100% — już samo bycie dobrym pracownikiem korporacji wnosi pewien narzut. Produktywność wydaje się być lepsza w przypadku zaangażowania w dwa projekty, a nie w jeden — stąd wniosek, że posiadanie drugiego projektu, na który można byłoby się przełączyć, pozwala pracownikowi na osiągnięcie lepszej produktywności.

Dane te pokazują, że praca nad trzema lub więcej projektami jednocześnie jest złym wyborem ekonomicznym, ponieważ więcej czasu zajmuje koordynowanie, zapamiętywanie i śledzenie informacji niż sama praca przynosząca konkretną wartość. Zatem przy ilu produktach czy projektach (czyli często również w różnych zespołach) można pracować jednocześnie? Prawdopodobnie nie więcej niż dwóch. Ja osobiście jestem zdecydowanym zwolennikiem pracy nad jedną rzeczą — w obecnym funkcjonującym środowisku przesiąkniętym informacją przekazywaną przez pocztę elektroniczną, komunikatory internetowe, Twitter, Facebook i inne technologie bycie dobrym pracownikiem korporacji jest najczęściej równoważne z byciem w jednym projekcie!

Dobrze. A co w przypadku specjalistów, którzy będą musieli być umiejscowieni w kilku produktach? Wcześniej użyłem przykładu Sue (projektantki UX), zużywającej w macierzystym zespole 10% czasu (z pozostałym czasem podzielonym na pracę w innych zespołach). Mimo naszych chęci utrzymywania jej w jednym lub dwóch produktach może zajść potrzeba przydzielenia jej do pięciu

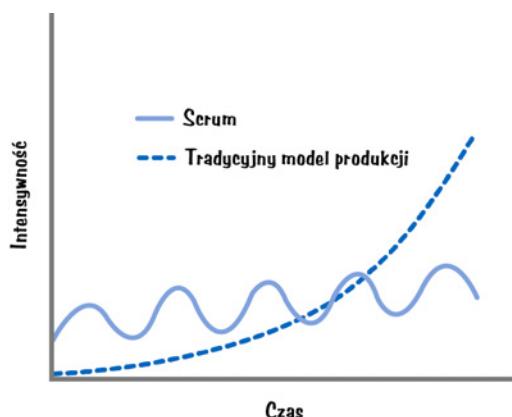
produktów jednocześnie, co wtedy? Praktyczną wskazówką w takiej sytuacji jest poproszenie samego specjalisty o wskazanie, ilu produktom jest się w stanie poświęcić jednocześnie z pełnym zaangażowaniem. Jeżeli powie, że nie jest w stanie poświęcić się większej ilości pracy, nie przypisujmy go do następnego produktu lub zespołu. Jeżeli taka decyzja nie satysfakcjonuje nas z biznesowego punktu widzenia (założymy na przykład, że Sue czuje się komfortowo, pracując wyłącznie nad jednym produktem), wtedy być może powinniśmy poszukać innego rozwiązania tego problemu.

Oto kilka możliwości. Zmniejsz liczbę konkurencyjnych projektów. Jest to zazwyczaj słuszne rozwiązanie, ponieważ wiele organizacji decyduje się na prowadzenie zbyt wielu projektów jednocześnie (rozważania na ten temat znajdziesz w rozdziale 16.). Możesz zatrudnić więcej specjalistów, aby bardziej równomiernie rozłożyć potrzebny wysiłek. Trzecie rozwiązanie to pomóc innym ludziom w poszerzeniu ich umiejętności o tę, na którą jest zapotrzebowanie. Czwarta możliwość to pewna kombinacja wszystkich trzech poprzednich. Ostateczny wniosek jest taki: zmuszanie ludzi do pracy nad zbyt dużą liczbą projektów lub w zbyt dużej liczbie zespołów spowoduje zmniejszenie ich skupienia oraz poświęcenia, narażając na niebezpieczeństwo wyniki biznesowe.

## Praca w podtrzymywalnym tempie

Jedną z nadrzędnych zasad Scruma jest konieczność pracy zespołów w tempie podtrzymywalnym. (Koniec z marszami śmierci!) Dzięki temu są one w stanie dostarczać światowej klasy produkty, zachowując jednocześnie zdrowe i przyjazne środowisko pracy.

Stosując sekwencyjny proces produkcyjny, odkładamy w czasie istotne zadania, takie jak integracja i testowanie, do niemal samego końca prac, kiedy pojawia się natłok problemów wymagających szybkiego rozwiązania ze względu na zbliżającą się datę wypuszczenia. Powoduje to nagły wzrost intensywności prac w późnych fazach produkcji (patrz rysunek 11.9).



RYSUNEK 11.9. Tempo podtrzymywalne ukazane na przestrzeni czasu

Ten niezwykle intensywny czas jest symbolizowany przez superbohaterów zarywających kolejne noce i weekendy, aby „dopiąć” wersję dystrybucyjną. Niektórzy ludzie uwielbiają ten rodzaj pracy, płynący z niej rozgłos oraz oczekiwanie na nagrodę za ponadprzeciętny wysiłek. Dla pozostałych ten rodzaj pracy kojarzy się z wszechogarniającym stresem. Jako organizacja powinniśmy zadawać sobie pytanie: „Dlaczego musimy pracować po nocach i przez weekendy i co zrobić, aby to zmienić?”.

Porównaj ten rodzaj wysiłku z typowym profilem intensywności prac w Scrumie, gdzie niestannie programujemy, testujemy i integrujemy działające cechy w każdym sprincie. W kolejnych iteracjach członkowie zespołu powinni wykorzystywać wysokie umiejętności techniczne, takie jak refaktoryzacja kodu, ciągła integracja i automatyczne testowanie, aby móc zapewnić regularne dostarczanie wartości bez przepracowywania się.

Zatem w każdym sprincie możemy spodziewać się pewnego wzrostu intensywności prac na końcu sprintu, kiedy staramy się upewnić, że wszelkie zadania wynikające z naszej definicji ukończenia zostały zrealizowane. Intensywność pracy w danym sprincie powinna jednak do złudzenia przypominać tę, jaka miała miejsce w poprzednim sprincie — jest to efekt pracy zespołu w podtrzymywальnym tempie.

Sumarycznie otrzymujemy wynik w postaci równoważenia ilości wykonanej pracy. Nie jest ona realizowana w formie ogromnych porcji lub też nagłych zrywów, szczególnie pod koniec, kiedy ma najbardziej niszczący efekt. Równoważenie oznacza, że zespół będzie rzadziej pracował w nadgodzinach i przez to zmniejszy szansę wypalenia się.

## Długotrwałość

Wydajne użycie Scruma wymaga istnienia zespołów, nie grup. **Zespół** jest stworzony z różnorodnych, współpracujących ze sobą osób posiadających wielorakie umiejętności, którym przedstawiono pewną wizję i poproszono o zrealizowanie jej wspólnym wysiłkiem. **Grupa** jest zbiorem osób, którym nadano pewną wspólną nazwę. Ludzi tych nie łączy nic poza ową nazwą i nie można oczekwać od nich, że będą wydajnie realizować obowiązki, jakie wskazałem dla roli zespołu deweloperskiego.

Z zasady zespoły powinny być długotrwałe. Ja utrzymuję swoje zespoły tak długo, jak tylko pozwalały na to uwarunkowania ekonomiczne, a trzeba powiedzieć, że ekonomia sprzyja zespołom długotrwałym. Badania przeprowadzone przez Katza pokazały, że zespoły długotrwałe są bardziej produktywne od zespołów nowo utworzonych [Katz, 1982]. Ponadto badania przeprowadzone przez Staatsa dowodzą, iż zaznajomiony zespół (członkowie zespołu mają wcześniejsze doświadczenia ze wspólnej pracy) może wpływać pozytywnie na wydajność i jakość generowanych przez siebie wyników [Staats, 2011]. Lepsza produktywność, wydajność i jakość prowadzą do lepszych wyników biznesowych.

Jeżeli zaczniemy od grupy ludzi, którzy nigdy nie pracowali ze sobą, musimy poświęcić czas i pieniądze na spojenie ich w prawdziwy zespół. Większość grup musi przejść przez pewne fazy, takie jak formowanie, sztormowanie, normowanie i ostatecznie wydajne pracowanie, aby móc stać się dobrze funkcjonującym zespołem [Tuckman, 1965]. Dobrze funkcjonujący zespół to prawdziwa wartość biznesowa. Jego członkowie potrafią ze sobą współpracować i mają do siebie zaufanie. Ponadto zespół ma zgromadzone istotne dane historyczne, takie jak prędkość oraz wydawane wcześniej oceny elementów rejestru (patrz rozdział 7.). Jeżeli rozmontujemy istniejący zespół lub znaczęco zmienimy jego skład, związane z nim dane historyczne nie będą się dłużej nadawać do bezpośredniego wykorzystania.

Bardzo często widzę organizacje, które nie potrafią docenić posiadanej wartości w formie zespołów. Wiele z nich wypracowało mechanizmy i procesy przesuwania osób w celu dynamicznego formowania „zespołów” (tak naprawdę grup). Moim zdaniem praktyki takie mijają się z krytycznym aspektem Scruma — wartością wynikającą z zespołu. Zespół jest *jednostką monetarną metod zwinnych*. Świadczy o tym jedna z kluczowych wartości Manifestu Zwinności — *Ludzie i współdziałanie*. Mówiąc inaczej, zespół to cenna wartość.

Przesuwanie ludzi pomiędzy zespołami niszczy ich integralność. Śmiem wątpić, czy nowojorska specjalna jednostka policyjna SWAT jest przebudowywana z jakąkolwiek częstotliwością. Jej członkowie nauczyli się pracować ze sobą i w trudnych sytuacjach mogą liczyć na wzajemne wsparcie. Wyciąganie z tego zespołu ludzi i umieszczenie w nim innych najprawdopodobniej zniszczyłoby to zaufanie, spójność i wydajność operacyjną (odpowiednik prędkości w naszym przypadku, a w przypadku zespołu SWAT bezpieczeństwo).

Wiele organizacji zyskałoby znacznie więcej, stosując politykę utrzymywania razem przynajmniej swoich kluczowych zespołów tak długo, jak jest to możliwe, i przesuwania ich z jednego produktu do kolejnego. Niemal zawsze bardziej ekonomiczne jest przesuwanie dobrze uformowanych zespołów niż przesuwanie ludzi.

Nie twierdzę, że możesz i powinieneś zawsze utrzymywać swoje zespoły przez długie okresy. Może się na przykład okazać, że zespół nie zgrał się w oczekiwany przez nas sposób lub w jego działaniu widać wyraźne wady. W takich sytuacjach rozwiązywanie zespołu stanowi mniejsze zło dla całego procesu i jest bardziej uzasadnione ekonomicznie.

Przypominam sobie również przypadek trenowanej przeze mnie organizacji, w której z pełną świadomością rozbiliśmy wysoko wydajny zespół scrumowy w ramach strategii „podziału i rozszerzenia” mającej na celu poszerzenie adaptacji Scruma wewnątrz organizacji. Nie podzieliliśmy zespołu dlatego, że skończył przydzieloną mu pracę i nadszedł czas przepisania ludzi do nowych zespołów w celu realizacji kolejnych projektów. Podzieliliśmy go, ponieważ uznaliśmy, że cenniejsze od zachowania tego zespołu będzie uformowanie sześciu nowych zespołów scrumowych, do których trafi przynajmniej jedna osoba mająca doświadczenie w Scrumie.

Zespoły stanowią zasób firmy. Każdy z nich jest pewną jednostką pojemności, której powinniśmy użyć do ustalenia limitu pracy cząstkowej dla liczby i rodzajów projektów, jakie chcemy realizować w sposób symultaniczny. Do tematu tego wróćmy w rozdziale 16.

## Zakończenie

W tym rozdziale opisałem rolę zespołu deweloperskiego. Podkreśliłem jego odpowiedzialność za zmianę elementów rejestru produktu w potencjalnie zdolne do wdrożenia przyrosty produktu. Omówiłem obowiązki zespołu w trakcie każdego sprintu. Następnie wskazałem dziesięć cech, jakich oczekujemy od naszych zespołów. Powinny to być w szczególności zespoły, które potrafią się samodzielnie zorganizować i są zróżnicowane pod względem pełnionych funkcji, a także posiadają umiejętności pozwalające realizować zlecone prace. Biorąc pod uwagę zadania, jakie musi realizować zespół, chcemy, aby posiadał on dobrą kombinację umiejętności typu T. To pozwoli mu na prawidłowe działanie w roju. Jeżeli umiejętności członków nie są jeszcze dostatecznie szerokie, chcemy, aby mieli oni chęć je poszerzać.

Chcemy również członków zespołu z nastawieniem muszkieterów — „wszyscy za jednego, jeden za wszystkich”. Zespoły powinny być tworzone w taki sposób, aby można było promować i realizować szerokopasmową komunikację. Preferujemy zespoły małe. Mając na uwadze skupienie i zaangażowanie, wolimy, aby członkowie zespołu pracowali nad jednym lub najwyżej nad dwoma produktami jednocześnie. Patrząc w szerszej perspektywie czasu, wybieramy członków zespołu, którzy mogą pracować razem przez długi czas w długotrwałych zespołach.

W następnym rozdziale skupię się na różnorodnych strukturach zespołów scrumowych, jakie będziesz mógł wykorzystać do skalowania w górę swojej implementacji Scruma.

## Rozdział 12

# BUDOWA ZESPOŁÓW SCRUMOWYCH

---

Zespoły scrumowe są kluczowym zasobem organizacji wykorzystującej Scrum. Ich konstrukcja oraz wzajemne relacje mogą wpływać istotnie na pomyślne wykorzystanie modelu scrumowego w organizacji. W tym rozdziale opiszę różne metody organizacji zespołów scrumowych. Zacznę od omówienia różnicy pomiędzy zespołem budującym cechy i zespołem budującym komponenty. Następnie skupię się na problemie koordynowania różnych współpracujących ze sobą zespołów scrumowych.

## Wprowadzenie

Jeżeli dysponujesz jednym, małym produktem, nie musisz specjalnie przejmować się treścią tego rozdziału. Stwórz jeden wielofunkcyjny zespół deweloperski o cechach, jakie opisałem w rozdziale 11., i zadbaj o prawidłowe obsadzenie ról mistrza młyna i właściciela produktu. Patrząc z punktu widzenia zespołu scrumowego, będziesz gotowy do rozpoczęcia pracy!

Wyobraźmy sobie jednak, że Twój pojedynczy, wielofunkcyjny zespół scrumowy stanie się wysoko wydajną maszyną dostarczającą wartość biznesową, dzięki czemu Twoja firma znaczenie rosnąć. Być może już stanowisz dużą organizację i po wyprodukowaniu pierwszego produktu przy użyciu Scruma chcesz rozprzestrzenić użycie tego procesu. W obu przypadkach bardzo szybko znajdziesz się w potrzebie koordynowania pracy wielu zespołów scrumowych, których wspólny wysiłek jest niezbędny do dostarczania coraz większej ilości wartości biznesowej.

W jaki sposób powinieneś zorganizować te zespoły, aby osiągnęły wysoką wydajność, a ich praca była skoordynowana? Odpowiedź na to pytanie, rozważając, czy powinieneś tworzyć zespoły budujące cechy, czy też zespoły budujące komponenty, a także jak podejść do koordynacji działalności wielu zespołów.

## Zespoły budujące cechy kontra zespoły budujące komponenty

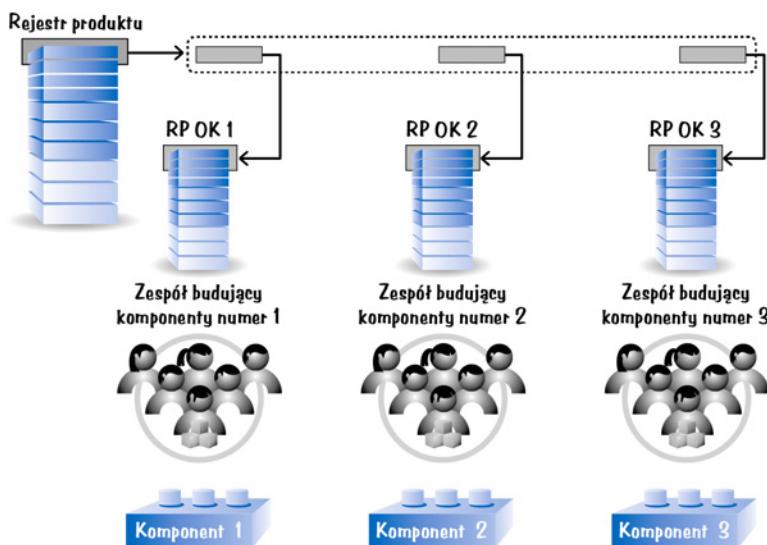
**Zespół budujący cechy** to wielofunkcyjny oraz wielokomponentowy zespół, który może pobrać cechy dla użytkownika końcowego z rejestru produktu i zrealizować je. Z kolei zespół budujący komponenty skupia się na tworzeniu komponentów lub też podsystemów, które mogą posłużyć jedynie jako części do cech przeznaczonych dla użytkownika.

W rozdziale 6. opisywałem, w jaki sposób producent urządzeń GPS może stworzyć zespół zajmujący się budową komponentu zawierającego skomplikowany kod do wyznaczania trasy w oparciu o położenie urządzenia i wyznaczony punkt docelowy. Za każdym razem, kiedy nadziejdie żądanie rozszerzenia cech urządzenia GPS związanych z wyznaczaniem trasy, szczegóły związane z samym algorytmem zostaną przypisane właśnie zespołowi zajmującemu się komponentem wyznaczania trasy.

Zespoły budujące komponenty są czasem określane mianem zespołów zasobowych lub podsystemowych. Takim zespołem mogą być również ludzie wywodzący się ze społeczności praktyków posiadających wiedzę w danym obszarze specjalizacji (patrz rysunek 13.4). W tego typu zespołach wszyscy członkowie zazwyczaj podlegają temu samemu menedżerowi i działają w formie współdzionego, scentralizowanego zasobu przeznaczonego dla innych zespołów. Przykładem może być scentralizowany dział UX projektujący interfejsy użytkownika z przeznaczeniem dla innych zespołów.

W Scrumie preferowane są zespoły budujące cechy. Niestety wiele organizacji preferuje zespoły budujące komponenty, często ze względu na przekonanie, iż zespół ekspertów cieszący się zaufaniem pod względem jakości i bezpieczeństwa zmian wprowadzanych w danym obszarze powinien być właścicielem tego obszaru kodu. Funkcjonuje pogląd, iż zespół, który nie miał styczności z danym kodem, mógłby nieświadomie wprowadzić do niego błędy. Stąd rodzi się chęć posiadania zespołu budującego komponenty i odpowiedzialnego za modyfikowanie wybranego obszaru kodu na rzecz innych zespołów.

Załóżmy, że rozwijamy produkt, którego cechy regularnie przecinają trzy obszary komponentów (patrz rysunek 12.1).

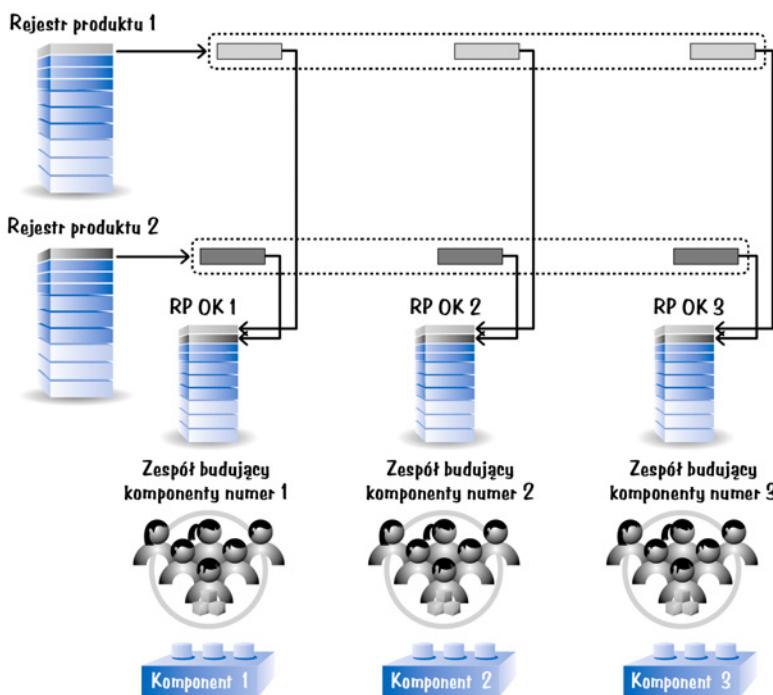


**RYSUNEK 12.1.** Jeden produkt i kilka zespołów budujących komponenty

W tym przypadku nie ma zespołu budującego cechy, który pracowałby nad całą treścią rejestru produktu. Zamiast tego cecha wybrana ze szczytu rejestru produktu jest dzielona na kawałki (zaznaczone przerywaną linią na rysunku 12.1) związanego z odpowiednimi komponentami. Podział ten jest realizowany wspólnie przez członków zespołów scrumowych budujących komponenty lub być może przez architekta systemu.

Następnie fragmenty cechy są umieszczane w odpowiednich rejestrach produktu poszczególnych zespołów budujących komponenty (na przykład pierwszy kawałek trafia do rejestru produktu związanego z obszarem numer 1 — „RP OK 1” na rysunku 12.1). Każdy zespół budujący cechy przeprowadza następnie działania scrumowe w odniesieniu do rejestru produktu związanego z ich własnym obszarem działalności i wytwarza gotowy fragment cechy końcowej, ale nie samą cechę. Następnie zespoły budujące komponenty integrują zbudowane przez siebie fragmenty w jedną całość — końcową postać cechy dla użytkownika — używając techniki podobnej do scruma scrumów, którą opiszę już niedługo.

Jeżeli żądania dotyczą podziału wyłącznie w obszarze jednego produktu, takie podejście będzie zapewne działać. Niemniej jednak organizacje tworzą na ogół zespoły wokół komponentów, które zamierają wykorzystywać w wielu produktach. Rysunek 12.2 pokazuje możliwy przepływ pracy w sytuacji, gdyby do tych samych zespołów budujących komponenty napłynęły żądania podziału z dwóch produktów.



RYSUNEK 12.2. Dwa produkty i kilka zespołów budujących cechy

Każdy rejestr produktu na poziomie cech użytkownika zawiera elementy, które mogą się rozciągać na różne obszary komponentów. Zatem na rysunku 12.2 są teraz dwa produkty, które wymagają pracy od zespołów budujących komponenty w ich obszarach specjalizacji.

Wyobraź sobie, że jesteś właścicielem produktu w jednym z tych zespołów budujących komponenty. Musisz nadać priorytety żądaniom rozszerzania funkcjonalności pochodząącym z dwóch produktów i jednocześnie koordynować swoją pracę z innymi zespołami działającymi na tym samym poziomie, aby mieć pewność, że odpowiednie fragmenty kodu zostaną zintegrowane we właściwym czasie.

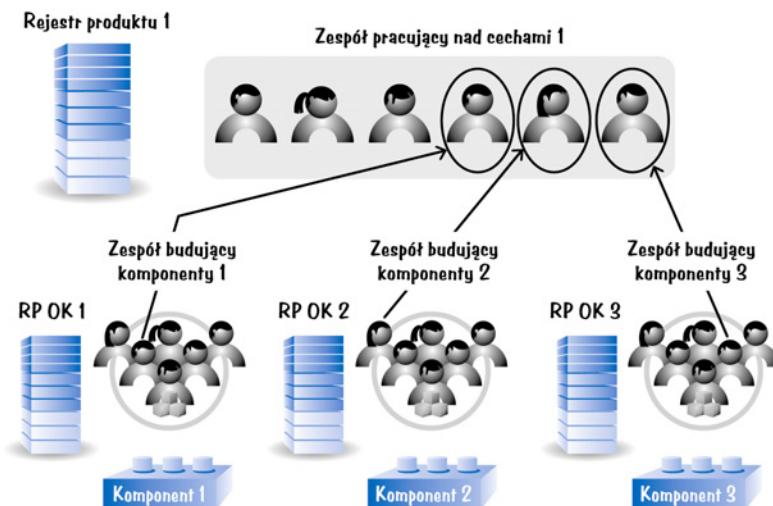
Przy dwóch produktach ten problem można jeszcze opanować logistycznie. Ale co zrobić, kiedy firma pracuje nad 10 lub 15 produktami jednocześnie i każdy z nich powoduje wpadanie zadań do rejestrów produktu zespołów komponentowych? W takiej sytuacji ustalenie prawidłowego porządku prac nad poszczególnymi elementami w ramach poszczególnych komponentów i jednoczesna koordynacja oraz integracja z pozostałymi zespołami budującymi komponenty staje się niemożliwa.

Z doświadczenia wiem, że większość organizacji posiadających zespoły budujące komponenty zaczyna zauważać problem, kiedy wszystko zaczyna się sypać (pałeczka upada na ziemię, powodując przerwanie płynnego potoku dostarczania wartości). Zazwyczaj scenariusz ten rozgrywa się następująco. Wyższy rangą menedżer pyta właściciela produktu zajmującego się cechami: „Dlaczego ta cecha dla użytkownika jest jeszcze niegotowa?”. Właściciel produktu odpowiada: „Jeden z zespołów budujących komponenty nie wykonał jeszcze prac, które mu zleliśmy. Ponieważ nie wykonał swojej pracy, cała cecha jest ciągle nieukończona”. Ów menedżer może wtedy dopytać: „Dlaczego zespół nie skończył jeszcze kawałka, który daliście mu do zrobienia?”. Możliwa odpowiedź to: „Pytałem pracowników i usłyszałem, że mieli na stole 15 innych żądań zmian w obszarze swojego komponentu i uznali, że z technicznego punktu widzenia lepiej będzie, jeśli zaczną od żądań zgłoszonych z innych produktów. Nadal jednak obiecuję, że zajmą się naszą rzeczą — prawdopodobnie w następnym sprincie”.

Jest to fatalny sposób prowadzenia interesów. Nigdy nie będziemy pewni kiedy (lub czy w ogóle) będziemy w stanie dostarczyć jakąkolwiek cechę, ponieważ odpowiedzialność za jej dostarczenie została rozproszona pomiędzy minimum dwa zespoły budujące komponenty, z których każdy może mieć zupełnie odmienne priorytety. Wykorzystanie w ten sposób zespołów budujących komponenty zwiększa wielokrotnie prawdopodobieństwo, że cecha nie zostanie ukończona ze względu na liczne możliwości porażki (zespoły budujące komponenty), a nie jedną taką możliwość (porażka zespołu budującego cechy).

Czy problem ten można jakość rozwiązać? Doskonałym podejściem byłoby stworzenie wielofunkcyjnych zespołów posiadających wystarczające umiejętności do tworzenia różnorodnych cech końcowych dla klienta bez konieczności przekazywania jakąkolwiek części prac zespołom budującym komponenty. Ale jak pogodzić to z podstawowym powodem, dla którego większość organizacji tworzy zespoły komponentowe — posiadaniem zaufanego zespołu do pracy w obszarze danego komponentu? Czy zespoły budujące cechy nie doprowadzą do chaosu podczas rozwijania i utrzymywania komponentów wielokrotnego użytku, wprowadzając do nich duże ilości długiego technicznego? Nie stanie się tak, jeśli dobrze uformowane przez nas zespoły budujące cechy z czasem zaczną współdzielić własność kodu i same staną się zaufanymi zespołami zdolnymi do jego rozwijania.

Takie przejściowe podejście mające na celu osiągnięcie modelu wielu zespołów budujących różnorodne cechy z jednoczesnym posiadaniem całego ich kodu prowadzi do zorganizowania zespołów w sposób przedstawiony na rysunku 12.3.



**RYSUNEK 12.3.** Połączenie zespołu budującego cechy z zespołami budującymi komponenty

W takim podejściu definicja zespołu budującego cechy została zmodyfikowana. Od teraz istnieje pojedynczy zespół budujący cechy, który może pobrać z rejestru produktu dowolną cechę dla użytkownika. Na zespole tym spoczywa całkowita odpowiedzialność za wykonanie pracy i zarządzanie logistiką niezbędną do dostarczenia gotowej cechy.

Na scenie pozostają również zespoły komponentowe. Ich zadaniem jest pomoc w utrzymaniu wewnętrznej integralności komponentów. Mają one nadal swoje rejestyry produktu wypełnione głównie zadaniami technicznymi, które muszą zostać zrealizowane w obszarze komponentów (być może jest to praca związana ze spłatą dłużu technicznego).

Ponadto zgodnie z ilustracją na rysunku 12.3 osoba z zespołu budującego komponenty może zostać przypisana do zespołu budującego cechy. Osoba taka ma podwójną odpowiedzialność siewcy i zbieracza [Goldberg i Rubin, 1995].

Członek zespołu komponentowego w roli siewcy ma za zadanie „rozsiewać” wiedzę na temat komponentu w zespołach budujących cechy i dzięki temu promować udział tych zespołów w odpowiedzialności za kod tego komponentu. Rola zbieracza polega z kolei na zbieraniu przez członka zespołu komponentowego zmian, jakich potrzebują w komponencie zespoły budujące cechy, oraz omawianiu ich ze swoimi kolegami z zespołów komponentowych, którzy również mogą być zbieraczami w tym samym komponencie. Dzięki tym dyskusjom członkowie zespołu komponentowego mogą skoordynować zmiany w obszarach komponentu niezbędne do zaspokojenia żądań spływających z różnych zespołów budujących cechy. Dodatkowo osoby realizujące zmiany w komponencie mają szansę wykonać je w sposób spójny i zapewnić lepszą integralność koncepcyjną modyfikowanych obszarów. Ponieważ każdy z członków zespołu komponentowego ma świadomość zbieranych żądań zmian w obrębie komponentu, powstaje okazja do poinformowania się nawzajem o nadążających się okazjach wielokrotnego wykorzystania tego samego kodu.

Podobnie jak w przypadku czystego zespołu komponentowego to podejście może również nie zadziałać w większej skali, ale z innych powodów — takich, którym jesteśmy w stanie zaradzić. Dla przykładu: kiedy przedstawiłem to podejście ludziom z pewnej dużej firmy, usłyszałem odpowiedź: „Nasze cechy mogą rozciągać się na 50 różnych systemów [komponentów]. Nie możemy przesunąć

50 osób do jednego zespołu budującego cechy". Chociaż cecha faktycznie może wymagać 50 komponentów, rzadko zdarza się, aby 50 komponentów wymagało bezpośredniej interakcji ze sobą. W związku z tym nie potrzebujemy 50 osób w zespole — zamiast tego możemy stworzyć kilka „zespółów budujących cechy” wokół mniejszych zbiorów komponentów, które rzeczywiście mocno współpracują ze sobą (przykłady znajdziesz w rozdziale 13. na rysunkach 13.5 i 13.6), a następnie skupić się na koordynacji wysiłków tych zespołów przy użyciu technik przeznaczonych dla wielu zespołów, jakie opiszę w dalszej części tego rozdziału.

Inna możliwość porażki podejścia z rysunku 12.3 może mieć miejsce, jeśli organizacja pracuje nad 40 różnymi produktami i ma tylko czterech pracowników w obszarze komponentów. Nie ma sensu przypisywać tych osób do dziesięciu różnych zespołów budujących cechy w tym samym czasie. Problem można jednak rozwiązać, redukując liczbę jednocześnie rozwijanych produktów (patrz rozdział 6.), trenując (lub zatrudniając) więcej ludzi posiadających wiedzę ekspercką w obszarach komponentów, a także — najchętniej — lepiej promując współdzieloną odpowiedzialność za kod (jako wizję długoterminową).

Z własnego doświadczenia mogę powiedzieć, że nie istnieje jedno uniwersalne rozwiązanie problemu zespołów budujących cechy kontra zespoły budujące komponenty. Większość dużych organizacji odnoszących sukcesy w Scrumie skłania się w kierunku modelu mieszanego, składającego się głównie z zespołów budujących cechy i pojedynczych zespołów budujących komponenty — w sytuacjach, kiedy posiadanie scentralizowanych zespołów budujących komponenty ma sens ekonomiczny. Z przykrością stwierdzam, że wiele organizacji preferuje model odwrotny — większość zespołów budujących komponenty i nieliczne zespoły budujące cechy. Organizacje takie płacą ogromną cenę w postaci opóźnień wynikających z częstych zatorów w przepływie pracy.

## Koordynacja wielu zespołów

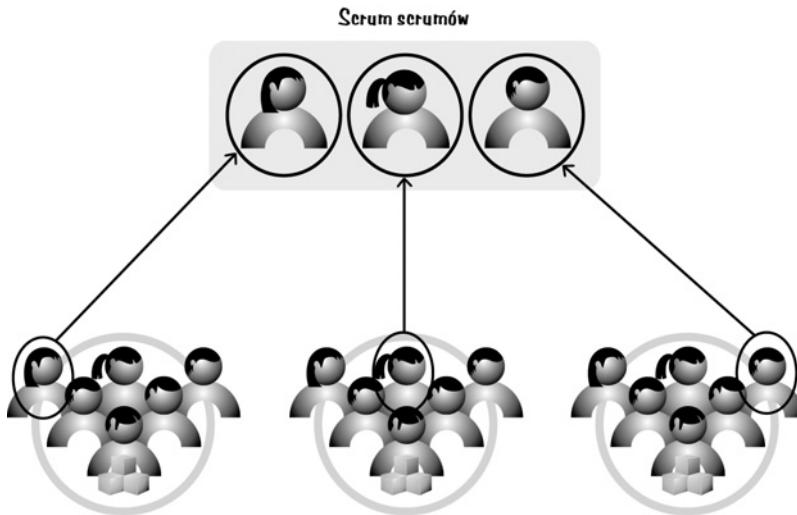
Scrum jest skalowalny nie dzięki rosnącym rozmiarom zespołów, ale dzięki posiadaniu wielu zespołów scrumowych o odpowiednim rozmiarze. Jednak tam, gdzie występuje więcej niż jeden zespół scrumowy, pojawia się problem ich koordynacji. Dwoma technikami zarządzania wieloma zespołami scrumowymi są scrum scrumów oraz bardziej wszechstronna forma koordynacji wielu zespołów, określana mianem pociągu wersji dystrybucyjnych.

### Scrum scrumów

W rozdziale 2. wspomniałem, że każdego dnia podczas wykonywania sprintu zespół wykonuje działania scrumowe. W działaniach tych biorą udział wyłącznie członkowie zespołu deweloperskiego.

Często spotykanym sposobem koordynacji wielu zespołów jest scrum scrumów (patrz rysunek 12.4).

Praktyka ta pozwala wielu zespołom na koordynowanie swojej pracy wewnętrzzespołowej. Zespół wykonujący scrum scrumów jest zbudowany z członków będących przedstawicielami różnych zespołów deweloperskich. O tym, kogo oddelegować do takiego zespołu, decyduje każdy z zespołów osobno — wskazując zazwyczaj osobę, która potrafi najlepiej przedstawić wewnętrzne problemy z zależnościami. Mimo że osobiście wolę stałą reprezentację zespołów, osoba wybrana do reprezentacji zespołu w trakcie scruma scrumów może się zmieniać w zależności od tego, kto najlepiej nadaje się do mówienia o problemach zespołu w danym momencie.



RYSUNEK 12.4. Scrum scrumów

Niektóre zespoły wysyłają na spotkanie zarówno członka zespołu deweloperskiego, jak i mistrza młyna (który może reprezentować kilka zespołów jednocześnie). Wszyscy wspólnie dbają o to, aby całkowita liczba uczestników nie była zbyt duża. Czasem wskazane jest powołanie odrębnego mistrza młyna na tym poziomie. Jeżeli taka rola jest przewidziana, zazwyczaj obsadza ją któryś z mistrzów młyna pochodzących z indywidualnych zespołów lub też osoba niezwiązana z żadnym zespołem uczestniczącym w scrumie scrumów.

Istnieje kilka wariantów przeprowadzania scruma scrumów, o wyborze odpowiedniego decydują sami uczestnicy spotkania. Scrum scrumów zazwyczaj jest przeprowadzany kilka razy w tygodniu według potrzeb. Uczestnicy odpowiadają na podobne pytania, jakie padają w trakcie codziennych działań scrumowych:

- Co mój zespół zrobił od ostatniego spotkania i jaki może mieć to wpływ na inne zespoły?
- Co mój zespół będzie robić do następnego spotkania i jaki może mieć to wpływ na inne zespoły?
- Z jakimi problemami boryka się mój zespół i jaka pomoc innych zespołów mogłaby się okazać pomocna w ich rozwiązyaniu?

Niektóre zespoły decydują się ograniczyć czasowo scrum scrumów do czasu nie dłuższego niż 15 minut (podobnie jak robią to indywidualne zespoły scrumowe podczas swoich codziennych spotkań). Preferowane jest rozwiązywanie problemów po zakończeniu spotkania, tak aby potrzebna była obecność jedynie osób niezbędnych do znalezienia rozwiązania.

Inna możliwość to rozszerzenie scruma scrumów powyżej 15-minutowego przedziału czasu. Uczestnicy mogą zacząć od 15-minutowej sesji odpowiadania na trzy powyższe pytania, a samo spotkanie trwa dłużej, dając wszystkim możliwość udziału w rozwiązywaniu zgłoszonych problemów.

Teoretycznie scrum scrumów może być skalowany na wiele poziomów w górę. Założymy, że pewien produkt jest tworzony przez dużą liczbę zespołów. Zazwyczaj zespoły te są grupowane

w klastry związane z poszczególnymi cechami. Pracę nad konkretną cechą można koordynować w klastrze za pomocą tradycyjnego scruma scrumów. Wskazane jest jednak posiadanie wyższego poziomu scruma scrumów, nazywanego scrumem scruma scrumów (ang. *scrum of scrum of scrums*) — dla uproszczenia „scrumem na poziomie programu” — pomagającego w koordynacji pracy pomiędzy klastrami. Chociaż takie rozwiązanie ma szansę zadziałać, istnieją inne techniki zarządzania dużą liczbą zespołów. Jedną z nich jest pociąg wersji dystrybucyjnych, którym zajmiemy się w następnej kolejności.

## Pociąg wersji dystrybucyjnych

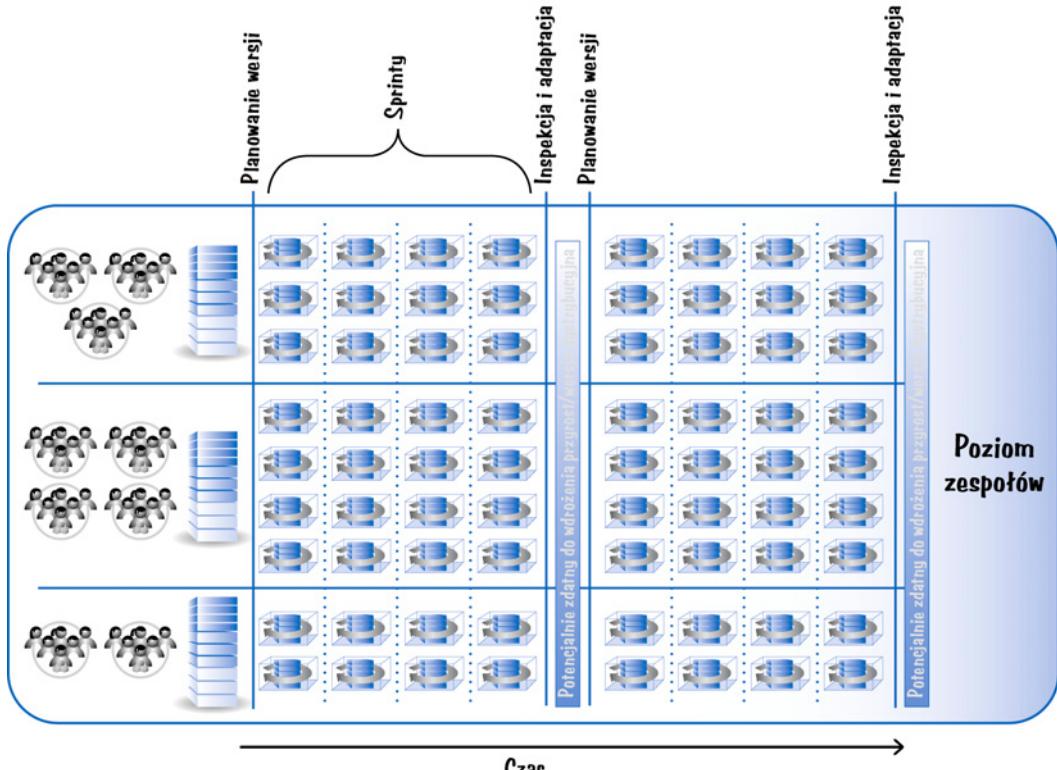
**Pociąg wersji dystrybucyjnych** jest podejściem układającym wizję, planowanie i wzajemne zależności pomiędzy zespołami w jedną całość poprzez **synchronizację** wielu zespołów bazującą na wspólnym takcie. Pociąg wersji dystrybucyjnych skupia się na szybkim i elastycznym przepływie na poziomie całego produktu.

Metafora odnosząca się do pociągu ma za zadanie wskazać, iż istnieje pewien rozkład jazdy, według którego poszczególne cechy będą „opuszczać stację”. Wszystkie zespoły biorące udział w wytwarzaniu produktu muszą dostarczyć swój bagaż do pociągu w ścisłe określonym czasie. Tak jak w każdym kraju z punktualnie funkcjonującą komunikacją kolejową pociąg wersji dystrybucyjnych zawsze odjeżdża w porę i nie czeka na nikogo. Z drugiej strony, jeśli zespół spóźni się na pociąg, nie musi się obawiać, ponieważ będzie następny, odjeżdżający o znanym czasie w przyszłości.

Leffingwell definiuje zasady pociągu wersji dystrybucyjnych w następujący sposób [Leffingwell, 2011]:

- ustalone z góry, częste i zamrożone daty spotkań planujących oraz wdrożeń wersji (lub potencjalnie zdatnych do wdrożenia przyrostów) — daty są ustalone, jakość jest ustalona, zakres jest zmienny;
- zespoły stosują jednakowe długości iteracji;
- ustalane są pośrednie, globalne oraz celowe kamienie milowe;
- ciągła integracja obowiązuje w całym rozwiązaniu lub systemie, a także na poziomie cech i komponentów;
- przyrosty nadające się do wdrożenia są dostępne do wglądu dla klienta oraz działu jakości w regularnych (zazwyczaj 60- lub 120-dniowych) odstępach czasu;
- dla zredukowania dłużu technicznego oraz w celu przeprowadzenia testów i weryfikacji wersji dystrybucyjnej stosowane są (lub mogą być) sesje uszczelniania systemu;
- aby zespoły mogły budować w oparciu o istniejące już konstrukcje oraz sprawdzone komponenty strukturalne, wcześniej muszą być tworzone odpowiednie interfejsy, narzędzia deweloperskie, programy instalacyjne i licencyjne, środowiska interakcji z użytkownikiem, serwery danych i usług sieciowych itp.

Rysunek 12.5 pokazuje część pociągu wersji dystrybucyjnych w oparciu o definicję Leffingwella.



**RYSUNEK 12.5.** Struktura pociągu wersji dystrybucyjnych

Jest to bardzo bogata koncepcja z wieloma stopniami szczegółowości, uwzględniająca poziomy portfela i wersji dystrybucyjnych. Jak wspomniałem w rozdziale 6., pociąg wersji dystrybucyjnych jest oparty na modelu rejestru produktu przedsięwzięcia składającego się z trzech poziomów rejestrów: rejestru portfela (z eposami będącymi własnością kadry zarządzającej portfelem), rejestru programów (z cechami będącymi w posiadaniu właścicieli produktu) i rejestrami zespołów (z historyjkami nadającymi się do sprintu, będącymi w posiadaniu właściciela produktu). Rysunek 12.5 pokazuje jedynie poziom zespołów. Szczegóły dotyczące planowania portfela i wersji dystrybucyjnej zostaną omówione odpowiednio w rozdziałach 16. i 17.

Pociąg na poziomie zespołów z rysunku 12.5 pokazuje dziewięć zespołów zgrupowanych w trzy klastry cech. Każdy zespół w ramach swojego obszaru cech wykonuje własny sprint, pobierając rzeczy do wykonania z rejestru produktu dla tego obszaru funkcjonalności. Zespoły koordynują i integrują swoją pracę w każdym sprincie, używając techniki takiej jak scrum scrumów.

Tak często jak tylko jest to możliwe z praktycznego punktu widzenia, przeprowadzana powinna być integracja oraz testowanie w całym obszarze cech. Niektóre zespoły decydują się poświęcić ostatni sprint przed odjazdem pociągu na *uszczerbienie* tego, co zostało zbudowane w poprzednich sprintach, a następnie zintegrowane i przetestowane w różnych obszarach funkcjonalności (na przykład na rysunku 12.5 takim sprintem mógłby być sprint numer 4). Potrzeba wprowadzania sprintu uszczelniającego powinna maleć wraz ze wzrostem umiejętności zespołu.

Czasy trwania oraz momenty rozpoczęcia sprintów we wszystkich zespołach biorących udział w wersji dystrybucyjnej powinny być identyczne. Dzięki temu wszystkie sprints będą rozpoczynać się i kończyć w tym samym dniu. Pozwala to na synchronizację nie tylko w obrębie danej funkcjonalności, ale również pomiędzy wszystkimi zespołami biorącymi udział w pracach nad produktem.

Potencjalnie zdatny do wdrożenia przyrost produktu (lub też przyrostowa wersja dystrybucyjna) jest dostępny po ustalonej liczbie sprintów, która w przypadku rysunku 12.5 wynosi cztery. Światodomość, iż kolejna wersja dystrybucyjna będzie wypuszczona w ścisłe określonym momencie, pozwala organizacji dopasować swoje pozostałe działania do tej daty. W punktach wypuszczenia wersji firma może wybrać pomiędzy wypuszczeniem przyrostu produktu do klientów (jeżeli jest to akceptowalne z biznesowego punktu widzenia) i użyciem go do potwierdzenia, iż cała praca zrealizowana w poszczególnych obszarach funkcjonalności została zintegrowana i przetestowana poprzez zlecenie przeglądu wewnętrznego.

Każde planowanie pociągu wersji dystrybucyjnych rozpoczyna się od planowania wersji dystrybucyjnej obejmującego wszystkie zespoły zaangażowane w pracę nad potencjalnie zdatnym do wdrożenia przyrostem produktu (patrz rysunek 12.5). Oznacza to, że w spotkaniu tym może brać udział jednocześnie kilkaset osób. Muszę przyznać, że obserwowanie na żywo takiego spotkania jest fascynujące. Oto jak przebiega planowanie na taką skalę.

Po pierwsze: potrzebna jest duża sala konferencyjna! Spotkaniu przewodniczy szef właścicieli produktu (patrz rysunek 9.13) i to on jest głównym animatorem. Poszczególne zespoły scrumowe gromadzą się w tym samym miejscu sali (najczęściej w pobliżu pustej ściany, na której mogą powiesić swoje artefakty). Zespoły przeznaczone do tego samego obszaru funkcjonalności zbierają się w klastry. Po przedstawieniu przez szefa właścicieli produktu ogólnego obrazu potencjalnie nadającego się do wdrożenia przyrostu produktu zespoły budujące cechy zbierają się razem. Właściciele produktu odpowiedzialni za poszczególne obszary funkcjonalności przedstawiają następnie ogólne koncepcje w swoich obszarach na nadchodzący pociąg wersji dystrybucyjnej.

W kolejnym kroku indywidualne zespoły scrumowe rozpoczynają planowanie swoich sprintów poprzez wkładanie cech w kolejne iteracje. Ta aktywność nazywana jest tworzeniem mapy sprintów — będę o niej pisał w rozdziale 18. Ponieważ zespoły scrumowe pracują faktycznie nad czymś większym, co będzie dostarczone przez wiele zespołów, niemal na pewno pojawią sięewnętrzne zależności pomiędzy zespołami. Żeby poradzić sobie z tymi zależnościami, członek zespołu może podejść do innego zespołu (być może niosąc w ręku karteczkę samoprzylepną) i zapytać, czy jego członkowie będą mogli zrealizować opisaną na karteczce pracę w trakcie nadchodzącego pociągu wersji dystrybucyjnej. Jeżeli odpowiedź będzie pozytywna, zespół zgłaszający potrzebę będzie mógł zobowiązać się do zrealizowania cechy zewnętrzna zależnością.

Przez cały czas osoby mające obowiązki rozciągające się na wiele zespołów, czyli między innymi właściciele produktów odpowiedzialni za całe obszary funkcjonalności oraz architekci systemu, mogą poruszać się od stołu do stołu i upewniać się, że wszyscy rozumieją szeroką wizję oraz spójny plan nadchodzącego pociągu wersji dystrybucyjnej. Każdy z zespołów może oczywiście również poprosić jedną z tych osób o podejście i udzielenie niezbędnej pomocy.

Po skończeniu sprintów pociągu wersji dystrybucyjnej i dotarciu do punktu wypuszczenia potencjalnie nadającego się do wdrożenia przyrostu produktu (odjazdu pociągu) przeprowadzamy aktywności inspekcjno-adaptacyjne na poziomie pociągu wersji dystrybucyjnej. Pierwszą z nich jest przegląd wszystkich rzeczy umieszczonych w pociągu wersji dystrybucyjnej. Drugą jest retrospekcja

na poziomie pociągu wersji dystrybucyjnej — skupiona na osiągnięciu lepszej wydajności w przyszłych pociągach. Na końcu przychodzi pora na zaplanowanie następnego pociągu wersji dystrybucyjnej.

## Zakończenie

W tym rozdziale omawiałem różne konstrukcje zespołów scrumowych. Zacząłem od wielofunkcyjnych zespołów budujących cechy, posiadających umiejętności pozwalające na pobranie cech dla użytkownika z rejestru produktu i samodzielne zrealizowanie ich. Następnie porównałem tego typu zespoły z zespołami budującymi komponenty — zajmującymi się wybranymi modułami, zasobami lub obszarami architektury systemu, czyli częściami, które wymagają integracji z cechami dla użytkownika. Dalej pokazałem możliwości połączenia obu typów zespołów jako metody pozwalającej organizacji na płynne przejście do stanu, w którym większość zespołów stanowić będą zespoły budujące cechy, a każdy z nich odpowiedzialny będzie za współdzielony kod.

Kolejną rzeczą omówioną w tym rozdziale była koordynacja wielu zespołów scrumowych, po czynając od tradycyjnej praktyki scrumowej zwanej scrumem scrumów, a następnie przechodząc do koncepcji zwanej pociągiem wersji dystrybucyjnych, która pozwala na koordynowanie dużej liczby zespołów scrumowych. W następnym rozdziale odejdę od tradycyjnych ról zespołu scrumowego i omówię rolę menedżerów w organizacji scrumowej.



# Rozdział 13

## MENEDŻEROWIE

---

Czy w świecie, w którym zespoły organizują się samodzielnie, jest jeszcze miejsce dla menedżerów? Absolutnie tak. Chociaż środowisko Scrum nie mówi nic szczególnego o roli menedżerów, oni sami odgrywają nadal ważną rolę w organizacji zwinnej. Tak czy inaczej istnieje wiele ról niewiązanych ze Scrumem, które mimo to mają kluczowe znaczenie dla działania całej organizacji. (Rolą nie-podlegającą Scrumowi jest księgowy — nie zdarzyło mi się jednak do tej pory spotkać członka zespołu scrumowego, który nie chciałby dostać swojego wynagrodzenia!).

W tym rozdziale omówię obowiązki menedżerów obszarów funkcjonalności (zwanych również menedżerami zasobów) — przykładem takiej roli są menedżerowie procesu deweloperskiego, menedżerowie kontroli jakości i dyrektorzy artystyczni w organizacji scrumowej. Na koniec zajmę się rolą menedżera projektu w Scrumie.

Ten rozdział jest przeznaczony bezpośrednio dla organizacji, które zatrudniają menedżerów obszarów funkcjonalności i menedżerów projektów. Jeżeli Twoja organizacja należy do małych i nie zatrudnia zbyt licznej kadry zarządzającej, możesz przejść do dalszej części książki. Z drugiej strony, lektura tego rozdziału może okazać się wartościowa z punktu widzenia tego, czego możesz potrzebować w przyszłości w miarę rozwoju Twojej firmy.

### Wprowadzenie

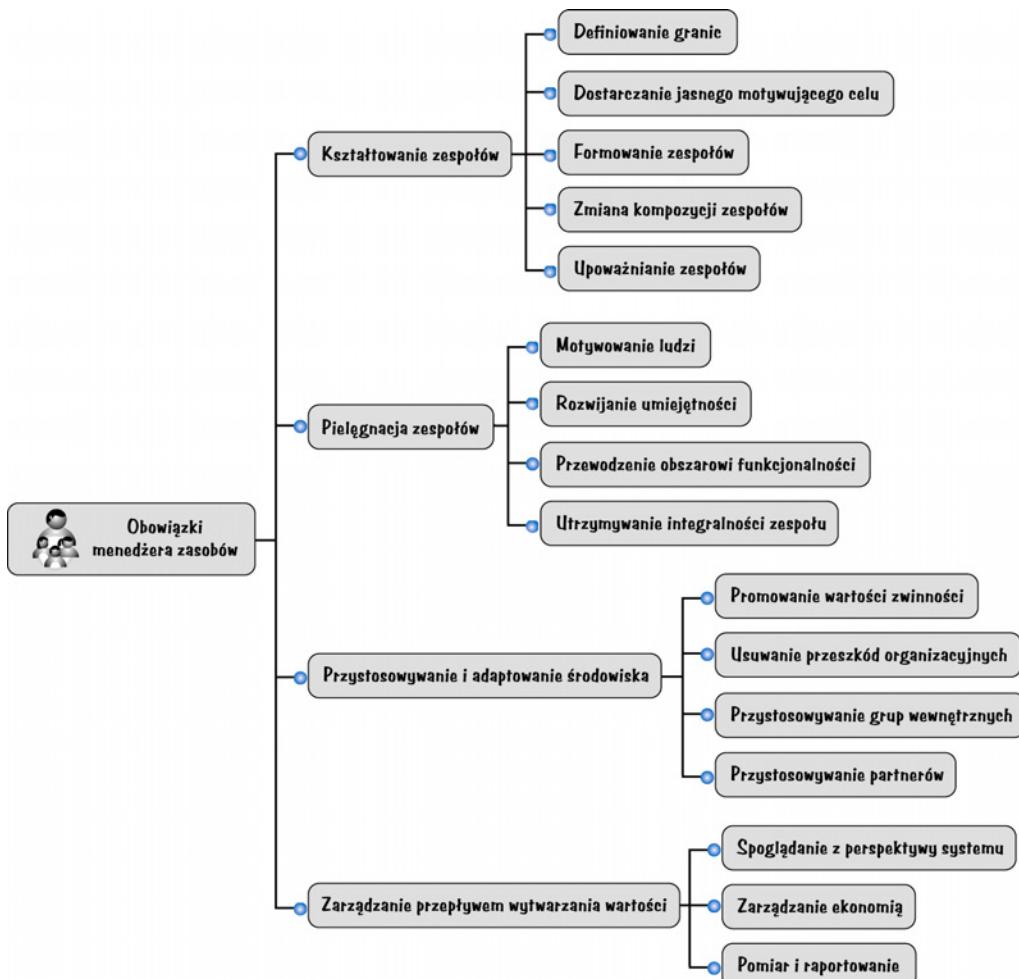
Według sondażu zwinności z 2011 roku przeszkodą numer jeden w adaptacji Scruma jest przesiadecie o utracie kontroli menedżerskiej (patrz rysunek 13.1 — źródło: [Verson One, 2011]).

Strach przed utratą znaczenia roli menedżera jest nieuzasadniony. W organizacjach scrumowych menedżerowie nadal zachowują ważne obowiązki (patrz rysunek 13.2).

Menedżerowie zasobów są odpowiedzialni w szczególności za kształtowanie i pielęgnowanie zespołów, przystosowywanie środowiska pracy i zarządzanie przypływem dostarczania wartości.



**RYSUNEK 13.1.** Największe wątpliwości podczas wprowadzania Scruma



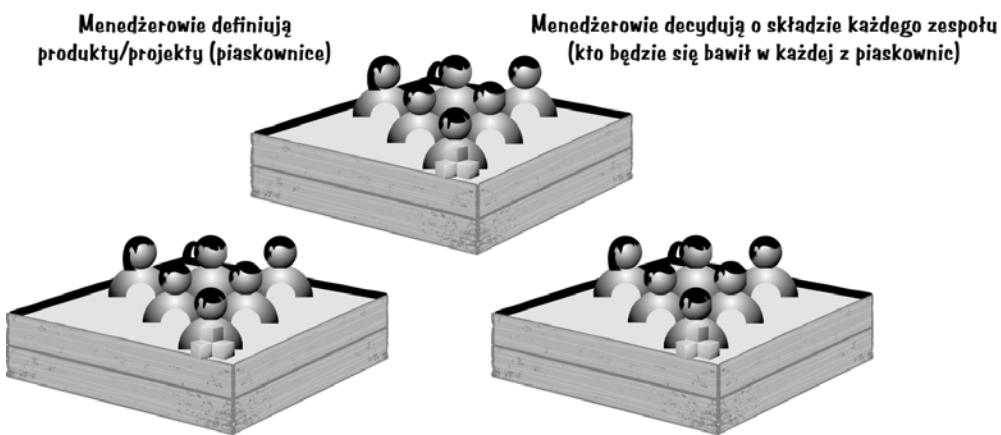
**RYSUNEK 13.2.** Obowiązki menedżera zasobów w organizacji scrumowej

## Kształtowanie zespołów

Menedżerowie kształtują zespoły, czyli przeprowadzają proces obejmujący definiowanie granic, wskazywanie jasnego, motywującego celu, formowanie zespołów, zmianę ich kompozycji i nadawanie im uprawnień.

### Definiowanie granic

W rozdziale 11. opisałem, w jaki sposób samoorganizujący się zespół zarządza swoimi działaniami wobec środowiska, w którym został umieszczony. Środowisko to podlega jednak wpływom menedżerów (patrz rysunek 13.3).



**RYSUNEK 13.3.** Menedżerowie definiują granice

Rzadko zdarza się, aby samoorganizujący się zespół decydował, jakie produkty lub projekty chce realizować. Przykładowo: jeżeli firma wytwarza oprogramowanie dla księgowości, zespół nie może podjąć decyzji, że zajmie się oprogramowaniem do kontroli świateł drogowych. Takie decyzje niemal zawsze podejmują menedżerowie — to oni decydują o granicach (piaskownicach), w ramach których zespoły mogą dokonywać samoorganizacji.

Powiedzmy, że zespoły zajmują się budowaniem zamków z piasku — rolą menedżera jest decydowanie, ile zamków z piasku zbudować (ile potrzeba piaskownic), a także jakiego rozmiaru piaskownice są potrzebne, aby w każdej z nich zespół mógł dokonać samoorganizacji i stworzyć swój zamek. Przekładając to na tematykę IT — menedżerowie w organizacji budującej oprogramowanie dla księgowości decydują, jakiego typu mają to być aplikacje, oraz wyznaczają granice poprzez wskazanie, czy zespoły deweloperskie powinny przekazać swoją pracę zespołom wdrożeniowym, czy też same muszą dokonać wdrożenia w ramach każdego sprintu.

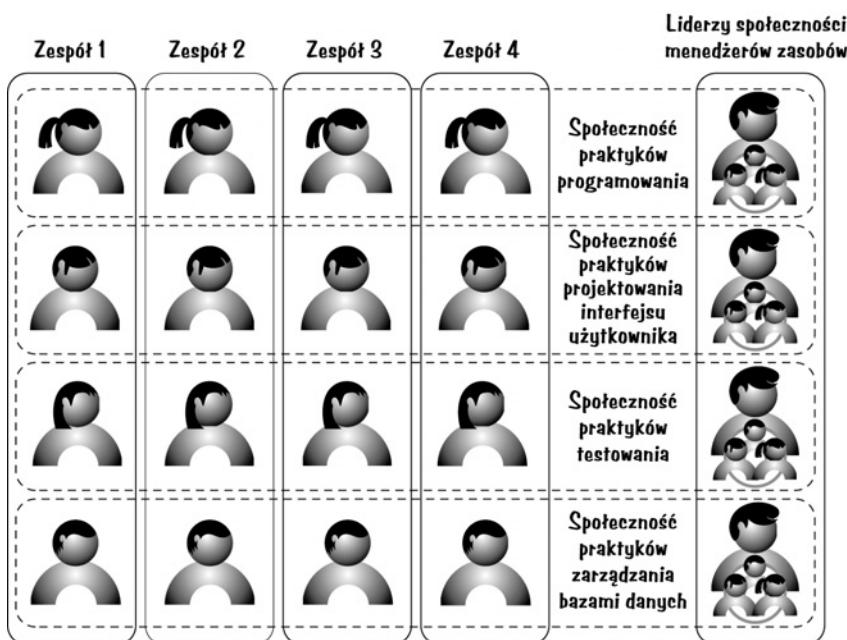
## Dostarczanie jasnego motywującego celu

Menedżerowie dostarczają również jasnego motywującego celu każdemu z zespołów. Ten cel daje zespołowi sens i kierunek działania. Używając dalej porównania do piaskownic: menedżerowie oznajmiają, że chcą mieć zamek z piasku, który wygra w weekendowych zawodach na najlepszy zamek z piasku. Właściciel produktu w zespole scrumowym może dalej uściślić ten cel, mówiąc: „Stwórzcie zamek średniowieczny z wieżami i otaczającą go fosą”.

## Formowanie zespołów

Zespoły zazwyczaj nie formują się samodzielnie (członkowie zespołów nie decydują, kto znajdzie się w zespole). To zadanie menedżerów. Wracając ponownie do przykładu z piaskownicami: menedżerowie niemal zawsze decydują, kto będzie się bawił w danej piaskownicy — nie robią tego członkowie poszczególnych zespołów. Z całą pewnością członkowie zespołu mogą i powinni wyrażać własne poglądy na temat procesu formowania zespołu — na przykład zgłaszając chęć znalezienia się w danym zespole lub przesłuchując kandydatów do istniejącego zespołu. Jednak w większości organizacji to menedżerowie podejmują ostateczną decyzję, upewniając się, iż skład zespołu bilansuje się z możliwościami i ograniczeniami biznesowymi organizacji.

W środowisku Scrum menedżerowie zasobów reprezentujący różne dyscypliny i społeczności praktyków współpracują ze sobą w celu wybrania członków wielozadaniowych zespołów scrumowych (patrz rysunek 13.4).



**RYSUNEK 13.4.** Menedżerowie zasobów wspólnie tworzą zespoły scrumowe

Na rysunku 13.4 każdy poziomy wiersz reprezentuje obszar funkcjonalności lub społeczność praktyków składającą się z ludzi o podobnych umiejętnościach (mogą to być deweloperzy, projektanci interfejsu użytkownika, testerzy lub administratorzy baz danych). Każdy obszar funkcjonalności ma swojego menedżera.

Menedżerowie funkcjonalności są wspólnie odpowiedzialni za wybieranie właściwych osób ze swojego obszaru do uformowania zespołów scrumowych — reprezentowanych na rysunku 13.4 przez kolumny. Starają się oni uformować zespoły posiadające dostatecznie zróżnicowane umiejętności w odpowiedniej ilości, tak aby wszyscy członkowie cechowali się uzupełniającymi się umiejętnościami typu T (patrz rozdział 11.).

## Zmiana kompozycji zespołów

Menedżerowie mają również obowiązek zmiany składu zespołu, jeśli uważają, że takie działanie poprawi kondycję i wydajność samego zespołu, a przez to również całej organizacji.

Załóżmy dla przykładu, że Fred jest osobą o małej wydajności w zespole deweloperskim. Ponadto ma złe nastawienie i źle wpływa na zdolność zespołu do wydajnej pracy. Jak należy z nim postąpić?

Po pierwsze, oczekwałbym od pozostałych członków zespołu Freda omówienia z nim istniejącej sytuacji i chęci udzielenia pomocy jemu i zespołowi. Jeżeli nie przyniesie to pozytywnych skutków, zadanie uczynienia z Freda bardziej efektywnego członka zespołu przejmie trener zespołu — mistrz młyna. Jeżeli trenowanie nie zadziała, sytuacja Freda zostanie najprawdopodobniej przedstawiona na zewnątrz zespołu scrumowego jego menedżerowi (osobie będącej bezpośrednim przełożonym Freda w organizacji), ponieważ sam mistrz młyna nie posiada uprawnień do zwalniania i zatrudniania ludzi.

Od tego momentu problemem wydajności Freda będzie zajmował się jego menedżer (być może współpracując z przedstawicielem działu kadry), próbując znaleźć rozwiązanie z poszanowaniem praw swojego pracownika. Będzie on chciał zapewne porozmawiać z mistrzem młyna i zespołem deweloperskim, aby lepiej zrozumieć zaistniałą sytuację. Mając pełny obraz problemu, menedżer może zdecydować o natychmiastowym usunięciu Freda z zespołu scrumowego i przypisaniu go do innego zespołu, gdzie będzie lepiej pasował. Dodatkowo menedżer może wprowadzić plan naprawczy wobec Freda (w jego obecnym lub nowym zespole). Niepowodzenie planu naprawczego będzie podstawą możliwej decyzji o zwolnieniu pracownika.

Chociaż to menedżerowie, a nie mistrzowie młyna i członkowie zespołu mają prawo do zwalniania ludzi, członkowie zespołu są zdecydowanie zaangażowani w ten proces, dbając o prawidłową konstrukcję swojego zespołu.

Menedżerowie mogą być również zmuszeni do zmiany składu zespołu, jeśli zmiana taka zapewni organizacji lepszą zdolność dostarczania wartości w całym swoim portfelu produktów. Zatem chociaż preferujemy długi czas życia naszych zespołów, niezbędne może okazać się przesunięcie od czasu do czasu osoby o szczególnych umiejętnościach z jednego zespołu do innego, który pilnie potrzebuje tych umiejętności. Menedżerowie muszą przeprowadzać tego typu zmiany z należytą ostrożnością, ponieważ mają one wpływ na skład obu zespołów jednocześnie.

## Upoważnianie zespołów

Aby zespoły mogły się samodzielnie organizować, muszą posiadać odpowiednie uprawnienia, co wymaga autoryzacji i zaufania ze strony menedżera. Jednym z głównych sposobów upoważniania zespołów jest oddelegowanie do nich obowiązków, których głównym celem jest umożliwienie samoorganizacji i zarządzania sobą. Należy zaznaczyć, że zespoły nie uzyskują wszelkich praw „menedżerskich” (jak powiedzieliśmy wcześniej, członkowie zespołu Freda nie mogą go zwolnić z powodu niskiej wydajności). Zespół może jednak uzyskać typowe prawa związane z zarządzaniem swoimi aktywnościami.

Dla każdej aktywności lub prawa do podejmowania decyzji, jakie menedżer przenosi na zespół, ustalany jest indywidualny poziom autoryzacji. Appelo definiuje siedem takich poziomów przedstawionych w tabeli 13.1 — przy każdym z nich umieszczony został przykład [Appelo, 2011].

**TABELA 13.1.** Siedem poziomów autoryzacji z przykładami według Appelo

Poziom	Nazwa	Opis	Przykład
1	Poinformowanie	Menedżer podejmuje decyzję i przekazuje ją zespołowi.	Przeniesienie do nowego biura.
2	Sprzedanie	Menedżer przekonuje zespół o nowej decyzji.	Decyzja o przejściu na Scrum.
3	Konsultacja	Menedżer prosi zespół o opinię przed podjęciem decyzji.	Wybranie nowego członka zespołu.
4	Porozumienienie	Menedżer i zespół podejmują decyzję wspólnie.	Wybór logo dla działu biznesowego.
5	Doradzenie	Menedżer udziela rady, aby wypłynąć na decyzję zespołu.	Wybór architektury lub komponentu.
6	Zapytanie	Menedżer pyta zespół o podjętą decyzję.	Długość sprintu.
7	Oddelegowanie	Menedżer oddelegowuje pełne prawo do podjęcia decyzji na zespół.	Najlepsze praktyki pisania kodu.

Poziomy te rozciągają się od jednego ekstremum — *poinformowania* — polegającego na podjęciu decyzji przez menedżera i przekazaniu jej zespołowi, do drugiego — *oddelegowania* — gdzie to zespół ma pełne prawo podjęcia decyzji.

Tam, gdzie menedżerowie oddelegowują zadania, wierzą również w prawidłowe przeprowadzenie swoich obowiązków przez zespół. Zespół z kolei ufa, że jego menedżerowie nie będą podejmowali działań stojących w sprzeczności z oddelegowanymi uprawnieniami. Menedżer nie powinien podejmować decyzji osobiście, jeśli wcześniej przekazał prawo do jej podjęcia zespołowi.

Menedżerowie powinni również pomagać członkom zespołu w zdobywaniu wzajemnego zaufania. Mogą to robić przez wyznaczanie odpowiednich granic dla środowiska, w którym operować ma zespół. Wskazanie granic zaufania pomaga formować zaufanie wewnętrzzespołowe. Zadaniem menedżerów jest także uświadamianie samoorganizującemu się zespołowi wagi przyjmowanych zobowiązań ze względu na brak menedżera w zespole, który pilnowałby, aby praca została zrealizowana. Ich kolejnym zadaniem powinno być podtrzymywanie nastawienia muszkieterów wśród członków zespołu, tak aby każdy mógł być pewny, że wszyscy są odpowiednio zaangażowani w realizację celów założonych przez cały zespół.

## Pielęgnacja zespołów

Po uformowaniu zespołów menedżerowie muszą je pielęgnować. Pielęgnacja nie oznacza *zarządzania zespołami*. Chodzi bardziej o przekazywanie ludziom pozytywnej energii, skupianie się na rozwijaniu kompetencji, przewodzenie w obszarze funkcjonalności i utrzymywanie integralności zespołów.

### Motywowanie ludzi

Wskazanie jasnego motywującego celu jest podstawą umożliwiającą przekazywanie członkom zespołu pozytywnej energii. Przez przekazywanie pozytywnej energii rozumieć nieustanne poszukiwanie sposobów motywowania ludzi, tak aby sami pragnęli wykonywać dobrą pracę. Wszyscy chcemy pracować w wyluzowanym środowisku pozwalającym na dostarczanie wartości — rolą menedżerów jest stwarzanie takiego środowiska. Poprzez właściwe zarządzanie menedżerowie mogą pozytywnie wpływać na wewnętrzną motywację i energię członków zespołów.

Z drugiej strony, istnieje możliwość podejmowania przez menedżerów decyzji dających zupełnie odwrotne skutki — takich, które zabierają energię ze środowiska i prowadzą do utraty motywacji przez ludzi. Przykładem mogą być menedżerowie funkcjonalności, którzy w przeszłości przyzwyczajeni byli do przypisywania zadań ludziom w swoim obszarze. Takie postępowanie w Scrumie spowoduje spadek pozytywnej energii wśród osób poprzez podważenie fundamentalnej zasady samoorganizacji. W wyniku na szwank narażona zostanie zdolność zespołu do dostarczania wartości.

### Rozwijanie umiejętności

Każdy członek zespołu w organizacji scrumowej odpowiada przed menedżerem funkcjonalności lub menedżerem zasobów, który zazwyczaj nie jest mistrzem młyna lub właścicielem produktu. Tak jak w przypadku organizacji niestosującej Scruma tacy menedżerowie mają za zadanie aktywnie trenować swoich bezpośrednich podwładnych i asystować im w osiąganiu ich własnych celów kariery poprzez oferowanie możliwości rozwoju umiejętności i dostarczanie okresowej informacji zwrotnej na temat ich efektywności.

Menedżerowie powinni rozwijać środowisko, w którym ludzie muszą nieustannie uczyć się i poszerzać krąg swoich umiejętności. Muszą jasno zakomunikować, iż nauka jest nie tylko wskazana, ale stanowi wręcz priorytet dla każdej osoby, zespołu czy też całego pionu organizacyjnego.

Działania takie jak dawanie ludziom czasu na szkolenia i uczestnictwo w konferencjach mówią same za siebie. W takim środowisku wspierającym naukę menedżerowie pobudzają pracowników do rozwijania swoich umiejętności domenowych, wiedzy technicznej, kreatywnego myślenia itd.

Menedżerowie muszą również regularnie przekazywać swoje uwagi i opinie zespołom oraz indywidualnym pracownikom. W wielu organizacjach, które nie używają Scruma, ocena wydajności pracowników odbywa się raz do roku. W organizacjach scrumowych, które zdecydowały się na przeprowadzanie tego typu ocen, oczekuje się od menedżerów kontynuowania tej praktyki. Niemniej jednak organizacje, które przyswoiły sobie fundamentalne wartości i zasady Scruma, szybko zdają sobie sprawę, że ocena wydajności pracownika raz (lub dwa razy) do roku ma się nijak do taktu zespołu scrumowego, który realizuje działania i zdobywa wiedzę w krótkich okresach sprintów. Roczne oceny wydajności mogą również sprzyjać wrogiemu współzawodnictwu w zespole, zamiast

wspierać samoorganizację z nastawieniem muszkieterów. Ocenianie w sposób indywidualny może również źle wpływać na nadzczną ocenę całego zespołu poprzez promowanie samodzielnego działania — ludzie optymalizują swoje postępowanie pod względem oceny osobistej kosztem zespołu. Organizacje, które pomyślnie wdrożyły Scrum, zaczynają szybko kwestionować wartość przeprowadzania rocznych ocen, zdając sobie sprawę, że mogą one spowodować więcej złego niż dobrego.

Nie oznacza to wcale, że w organizacjach scrumowych nie dokonuje się ocen osobistych. Chodzi jedynie o to, aby menedżerowie dostosowali częstotliwość wyrażania swoich ocen do pętli zdobywania wiedzy zespołu, z którego wywodzą się ich bezpośredni podwładni. Jednym z możliwych rozwiązań tego problemu jest ocenianie co sprint. Indywidualna ocena powinna być również odpowiednio wpasowana w kontekst wpływu (lub braku wpływu) wydajności danej osoby na wydajność całego zespołu.

## Przewodzenie obszarowi funkcjonalności

Podobnie jak w organizacjach niestosujących Scruma, tak i w organizacjach scrumowych menedżerowie przewodzą swoim obszarom funkcjonalności.

Posiadają oni zazwyczaj dobrą wiedzę na temat podlegającego im zakresu funkcjonalności i mogą przewodniczyć ideowo. Tego typu przewodnictwo nie oznacza przypisywania zadań swoim bezpośrednim podwładnym lub mówienia im, jak mają wykonywać swoją pracę. Takie działania osłabłyby samoorganizujący się zespół. Przewodzenie ma polegać na udzielaniu wsparcia w zakresie zachowania spójności i trenowania w tym konkretnym obszarze funkcjonalności.

Na przykład w firmach tworzących gry komputerowe artyści podlegają dyrektorowi artystycznemu, który sam również jest artystą o wysokich umiejętnościach. Dyrektor artystyczny jest liderem pozostałych artystów i pomaga im ustanowić standardy artystyczne dla danej gry, a następnie przejrzeć pracę poszczególnych artystów, aby zapewnić spójność całości dzieła. Nie chcemy, aby artysta w jednym z zespołów scrumowych tworzył dzieło gotyckie, a artysta w drugim zespole stosował kreskówki. Dyrektor artystyczny przewodzi całemu obszarowi i pomaga w zapewnieniu spójnego wyniku końcowego o wysokiej jakości.

Przewodzenie menedżerów funkcjonalności objawia się również przez wyznaczenie właściwych standardów dla zarządzanego obszaru, a także przez promowanie pewnych inicjatyw w tych obszarach. Oto przykład: powiedzmy, że dyrektor ds. jakości chce wybrać nowe narzędzia do automatyzacji testów, które będą mogły być używane przy różnych projektach deweloperskich. Chcąc osiągnąć ten cel, dyrektor może poprosić podlegające mu osoby zajmujące się testowaniem i należące do różnych zespołów scrumowych (tak jak pokazuje to rysunek 13.4) o pomoc w dokonaniu właściwego wyboru.

## Utrzymywanie integralności zespołu

W rozdziale 11. stwierdziłem, że jednostką monetarną metod zwinności jest zespół. Ponieważ każda osoba w zespole reprezentuje jego jednostkę pojemności<sup>1</sup>, menedżerowie powinni aktywnie działać w kierunku utrzymania integralności zespołu. Oznacza to między innymi przeciwdziałanie wyciąganiu ludzi z zespołu w połowie sprintu do pracy nad faworyzowanymi przez siebie projektami czy też nizkim nieuzasadnionemu przypisywaniu ludzi do pracy w wielu zespołach jednocześnie.

<sup>1</sup> Każda osoba w dobrze zintegrowanym zespole powiększa jego zdolność produkcyjną w każdym sprintie — przyp. tłum.

Biorąc pod uwagę argumenty ekonomiczne uzasadniające utrzymywanie zespołów długowiecznych, menedżerowie powinni również starać się zachowywać zespoły w niezmienionym składzie tak długo, jak długo ma to sens z punktu widzenia finansów. Po zakończeniu projektu menedżerowie powinni w pierwszej kolejności spróbować przypisać cały zespół do następnego zadania. Powinni to zrobić, zanim zmniejszą wartość zespołu przez rozbicie go na kawałki i utracą wartość wynikającą z jego spójności.

## Przystosowywanie i adaptowanie środowiska

Przystosowanie pojedynczego zespołu lub nawet działu IT, ewentualnie działu produkcji do pracy w Scrumie to dobry początek. Żeby jednak można było wykorzystać niesamowite zalety Scruma, zasady zwinności muszą zostać przyjęte w całym łańcuchu biznesowym, poczynając od dostawców, a na klientach kończąc. Za odpowiednie przystosowywanie i adaptowanie środowiska (łańcucha biznesowego) — poprzez promowanie metod zwinnych, usuwanie przeszkód organizacyjnych, dostosowywanie do siebie grup wewnętrznych oraz partnerów — odpowiadają menedżerowie.

### Promowanie wartości zwinności

Menedżerowie muszą wyznawać wartości i zasady zwinności. Muszą je rozumieć i szczerze w nie wierzyć, żyć nimi i zachęcać innych do takiego samego postępowania. Zbyt często podczas moich wykładów lub trenowania zespołów scrumowych słyszę: „Tak, tak, wszystko to ma dla nas sens, ale żeby faktycznie wprowadzić Scrum w życie, musimy jeszcze przekonać zarząd. Szkoda, że nie ma ich tutaj razem z nami”. Te zespoły mają absolutną rację. Jeśli mają osiągnąć długotrwały sukces, będą potrzebowały wsparcia zarządu.

Pewnego razu podczas obiadu zaangażowałem się w dyskusję z zespołem menedżerskim organizacji, która właśnie zaczynała adaptację Scruma. Podczas rozmowy wspomniałem, iż menedżerowie powinni unikać wyjmowania „w locie” pracownika z zespołu i przydzielania mu tymczasowych zadań w jakimś innym projekcie ze względu na możliwe zakłócenie pracy. Jeden z menedżerów odpowiedział nieśmiało, choć szczerze: „Zgoda, ale ja robię to na okrągło i nie uważam, żeby było w tym coś złego. O czym jeszcze powiniensem wiedzieć jako menedżer w organizacji przyjmującej metody zwinności, abym mógł lepiej dostosować swoje zachowanie oraz środowisko do promowania tych metod?“.

W odpowiedzi na to pytanie zacząłem wykład na temat zupełnie podstawowych wartości i zasad zwinności (na wzór treści rozdziału 3.), aby uświadomić jemu i jego kolegom, w jaki sposób menedżer może wesprzeć zasady zwinności, zamiast nieswiadomie pracować przeciwko nim. Prawdziwe promowanie tych zasad jest możliwe jedynie dzięki właściwym zachowaniom dzień po dniu.

### Usuwanie przeszkód organizacyjnych

Menedżerowie pracują ramię w ramię z mistrzami młyna w celu usuwania przeszkód. Chociaż to mistrz młyna jest osobą mającą za zadanie pozbywać się przeszkód organizacyjnych, wiele z nich — szczególnie tych natury środowiskowej — wymaga interwencji ze strony menedżerów.

## Przystosowywanie grup wewnętrznych

Zazwyczaj pierwszymi w kolejce do przyjęcia Scruma są zespoły inżynierijne i IT. Założymy, że po pewnym czasie pierwsza z takich grup w każdym sprincie osiąga wysoki poziom umiejętności w wytwarzaniu cech mających wartość dla klienta. Jednak dopóki owe cechy nie zostaną udostępnione klientowi, nie można mówić o dostarczeniu jakiekolwiek wartości. Co jeśli grupa zajmująca się wdrożeniami nie działa w sposób zwinny i nie może lub nie chce wdrażać cech co kilka tygodni? Czy taka organizacja może twierdzić, że posiada status instytucji scrumowej o dużej wydajności, w sytuacji kiedy nie jest w stanie dostarczyć wartości klientowi w odpowiednim czasie?

Co zrobić, jeśli podobne niedopasowanie pojawia się za działem produkcyjnym? Być może sprzedaż i marketing operują według zupełnie odmiennych zasad. Ich nastawienie może się wyrażać na przykład tak: „Wy w dziale produkcji używajcie sobie procesu, jakiego wam się tylko podoba, pod warunkiem że jesteście w stanie z góry odpowiedzieć na moje szczegółowe pytania i dotrzymać daty, którą przekazałem już klientowi”. A może w dziale kadr system rekrutacyjny ciągle opiera się na starym opisie stanowisk i nie przewiduje zatrudniania ludzi posiadających umiejętności typu T, pragnących pracować w samoorganizujących się zespołach.

W takim środowisku nie jesteśmy w stanie skorzystać z pełnych, długotrwałych zysków oferowanych przez Scrum. Menedżerowie (również ci mający władzę wykonawczą) są zobowiązani do modyfikacji środowiska, tak aby osiągnąć dobre wewnętrzne warunki dla różnych grup, takich jak nadzór, finanse, sprzedaż, wdrożenia czy wsparcie użytkowników. Menedżerowie muszą patrzeć całościowo i dostosowywać całościowo do metod zwinności.

## Przystosowywanie partnerów

Dlaczego zatrzymywać się wyłącznie na przystosowaniu wewnętrznym? Menedżerowie powinni pomagać organizacji w promowaniu bardziej zinnego podejścia w zarządzaniu dostawami i outsourcingiem. Organizacji nie uda się osiągnąć pełnego sukcesu w Scrumie, jeśli w kontaktach ze swoimi zewnętrznymi partnerami będzie podążać utartą ścieżką negocjacji na wyciągnięcie ręki z silnym zaangażowaniem kontraktów.

Zamiast tego menedżerowie powinni promować metody zwinności podczas kontaktów ze swoimi partnerami. Na przykład najprostszą formą outsourcingu jest leasing zespołu scrumowego od firmy zewnętrznej. Zamiast przeprowadzać skomplikowany proces tworzenia zespołu o wysokiej wydajności, menedżerowie mogą kupić dostęp do takiego zespołu stworzonego już przez kogoś innego. W ten sposób organizacja używa Scruma zgodnie z opisem przedstawionym w tej książce, ale zespół deweloperski (i być może mistrz młyna) jest „w posiadaniu” kogoś poza organizacją.

Chcąc osiągnąć taki poziom zinnego przystosowania do partnerów, menedżerowie powinni rozważyć rozwiązania alternatywne do spisywania z klientami kontraktów o ustalonej cenie. Tego typu kontrakty niemal natychmiast doprowadzają do nieporozumień pomiędzy organizacją i jej kontrahentami (Kontrahenci chcą dostarczyć jak najmniej się tylko da, aby zrealizować kontrakt i zmaksymalizować swój przychód, a organizacja chce otrzymać jak najwięcej za ustaloną cenę). Trudno uznać to za zinną metodę działania. Menedżerowie powinni zmienić ten styl angażowania się w relacje biznesowe.

## Zarządzanie przepływem wytwarzania wartości

Patrząc całościowo, menedżerowie w środowisku Scrum są odpowiedzialni za ustalanie strategicznego celu, a następnie ekonomiczne zarządzanie zasobami na drodze do osiągnięcia tego celu. Menedżerowie zarządzają przepływem wytwarzania wartości, patrząc z punktu widzenia budowanych systemów, zarządzając finansami, a także mierząc i raportując.

### Spoglądanie z perspektywy systemu

Aby efektywnie zarządzać przepływem wytwarzania wartości, menedżerowie muszą patrzeć z perspektywy systemów. Jedną z większych przeszkód, jaką widziałem podczas adaptowania Scruma, była niechęć menedżerów do myślenia o systemie jako całości i skupianie się jedynie na swoich własnych obszarach wiedzy. Często słyszę: „Tak, ale robienie tego, co proponujesz, wymagałoby zmiany w strukturze raportowania lub w opisach kluczowych stanowisk”. Kiedy ludzie mówią takie rzeczy, często słyszę również: „Nie wyobrażam sobie, abym mógł dokonać jakiekolwiek zmiany w swoim otoczeniu, która spowodowałaby lepszą zgodność z wartościami i zasadami Scruma, a także z pozostałą częścią organizacji”.

Takie myślenie bardzo utrudnia jakiekolwiek wewnętrzne dostosowanie do metod zwinności i może doprowadzić do stanu, w którym różne części organizacji dosłownie pracują na niekorzyść ogólnego dobra systemu. Menedżerowie pracujący w organizacji scrumowej i chcący realizować długofalowy zysk z wysoko wydajnego procesu scrumowego muszą pragnąć patrzeć całościowo.

### Zarządzanie ekonomią

Organizacje oczekują od menedżerów bycia zaufanymi strażnikami powierzonych im zasobów finansowych. W związku z tym menedżerowie wyższego szczebla w organizacji scrumowej nadal zajmują się zarządzaniem ekoniemią (na przykład zyskami i stratami) swoich obszarów. Menedżerowie obszarów funkcjonalności lub też menedżerowie zasobów mogą nie mieć bezpośrednich obowiązków związanych z generowaniem przychodu, ale nadal ponoszą odpowiedzialność za sposób wydawania powierzonych im pieniędzy.

Menedżerowie (prawdopodobnie na poziomie wykonawczym) są również zobowiązani do nadzoru nad ekonemią na wyższym poziomie organizacji. Jest to często widoczne poprzez ich zaangażowanie w zarządzanie portfelem i nadzór korporacyjny. Zarządzając portfelem, menedżerowie decydują, jakie projekty deweloperskie otrzymają środki finansowe, w jakim stopniu oraz w jakiej kolejności będą realizowane. Kiedy projekt wejdzie już w fazę produkcji, menedżerowie analizują nieustanny strumień informacji zwrotnej płynący z iteracyjnego i inkrementalnego wysiłku deweloperskiego i jeśli zachodzi taka potrzeba, przerywają dany wysiłek, gdy z ekonomicznego punktu widzenia jest on dłużej nieuzasadniony (patrz rozdział 16.).

### Obserwacja pomiarów i raportowanie

Na żądanie menedżerów tworzonych jest wiele raportów i pomiarów, stąd do nich należy decyzja o mierzeniu i raportowaniu tylko tego, co ma faktyczny związek z przepływem wytwarzania wartości. Cel ten można osiągnąć, zapewniając zgodność pomiarów i raportów z wartościami i zasadami Scruma.

Kilka zasad Scruma, które mogą pomóc menedżerom w ich podejściu do pomiarów i raportowania, opisałem w rozdziale 3. Oto kilka przykładów:

- Skupiąj się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę. Chcąc osiągnąć ten cel, mierz raczej, kiedy i jak często praca jest zakłóczana, a nie to, jak dobrą jesteś w utrzymywaniu wysokiego wykorzystania swoich pracowników.
- Mierz postęp przez działające, zweryfikowane zasoby. Czy ma to jakiekolwiek znaczenie, że dostarczysz produkt na czas i w ustalonym budżecie, jeśli produkt ten nie spełnia oczekiwaniów odbiorców? Skup się na mierzeniu wartości dostarczonej (działających i zweryfikowanych zasobach), ale nie trać z oczu zmiennych (daty ukończenia, zakresu, budżetu i jakości) potrzebnych do dostarczenia wartości.
- Zorganizuj przepływ zapewniający szybką informację zwrotną. Dostosuj swoje pomiary tak, aby określić, jak szybko domykany jest cykl zdobywania wiedzy (przyjęcia założeń, budowania, pobierania informacji zwrotnej, inspekcji i adaptacji).

Ten ostatni pomiar jest kluczem do **rachunku ekonomicznego uwzględniającego innowacje**, który ma miejsce w każdej organizacji tworzącej produkty lub usługi w warunkach ogromnej niepewności [Ries, 2011]. Rachunek ekonomiczny uwzględniający innowacje używa pomiarów rodzących działania, mających na celu określenie szybkości zdobywania wiedzy jako kluczowej miary postępu w kierunku generowania wartościowego wyniku biznesowego. Rachunek ekonomiczny uwzględniający innowacje bazuje na trzech krokach:

1. Stworzenie możliwie minimalnego produktu pozwalającego na ustalenie wartości bazowych dla rodzących działania pomiarów określających, w jakim miejscu organizacja lub produkt znajdują się w chwili obecnej.
2. Próba poprawienia pomiarów rodzących działania z wartości bazowych do wartości idealnych lub pożądanych poprzez serię przyrostowych poprawek produktu.
3. Jeżeli pomiary rodzące działania wykażą, że produkt wykazuje możliwy do zademonstrowania postęp w pożądanym kierunku, pozostajemy na bieżącej ścieżce. W przeciwnym razie dokonujemy **zwrotu** w kierunku nowej strategii i zaczynamy cały proces od początku.

Koncepcję zwrotu oraz zachowania bieżącej ścieżki opiszę bardziej szczegółowo w rozdziałach 14., 16. i 17.

## Menedżerowie projektów

Do tej pory pisałem o roli menedżera obszaru funkcjonalności lub też menedżera zasobów. A co z rolą menedżera projektu? Czy w organizacji jest miejsce na taką rolę? Czy taka rola ma swoje miejsce w Scrumie?

## Obowiązki menedżera projektu w zespole scrumowym

Często spotykanym błędnym przekonaniem jest traktowanie mistrza młyna jako „zwinnego menedżera projektu” lub też menedżera projektu o innym tytule. Istnieją pewne powierzchowne podobieństwa pomiędzy mistrzem młyna i menedżerem projektu — na przykład jeden i drugi zajmują się usuwaniem przeszkód. Niemniej jednak rola lidera służby różni się znacząco od roli skupionego na zarządzaniu i kontroli menedżera projektu.

Aby odpowiedzieć na pytanie: „Gdzie ma swoje miejsce menedżer projektu?”, przyjrzyjmy się jego kluczowym obowiązkom zdefiniowanym przez Instytut Zarządzania Projektami [PMI, 2008], pokazanym w tabeli 13.2.

**TABELA 13.2.** Tradycyjne obowiązki związane z zarządzaniem projektem

Aktynośc zarządzania projektem	Opis
Integracja	Identyfikacja, definiowanie, łączenie, unifikowanie i koordynowanie różnych procesów i aktywności podczas zarządzania projektem.
Zakres	Definiowanie i kontrolowanie, co jest, a co nie jest włączone do projektu, oraz upewnianie się, że wszystkie niezbędne prace są włączone do projektu.
Czas	Zarządzanie w kierunku ukończenia projektu na czas poprzez definiowanie, co i kiedy należy zrobić, a także jakie zasoby są do tego niezbędne.
Jakość	Definiowanie wymogów i (lub) standardów jakościowych, prowadzenie działań zapewniających jakość oraz monitorowanie i przechowywanie wyników aktywności związanych z zapewnieniem jakości.
Zespół (zasoby ludzkie)	Organizowanie zespołu projektowego, zarządzanie nim i przewodzenie mu.
Ryzyko	Planowanie, identyfikowanie, analizowanie, przeciwdziałanie, monitorowanie i kontrolowanie ryzyka projektowego.
Zaopatrzenie	Zakup produktów, usług lub wyników na zewnątrz zespołu.

Z całą pewnością powyższe obowiązki są bardzo ważne i powinny być wypełniane. Zatem jeśli nie ma menedżera projektu, kto zajmuje się ich wypełnianiem?

Sposób dystrybucji tradycyjnych obowiązków menedżera projektów wśród różnych ról scrumowych, a także wśród innych menedżerów przedstawia tabela 13.3.

Patrząc na tabelę 13.3, można powiedzieć, że osoba, która była menedżerem projektu, w zależności od swoich umiejętności i preferencji może przyjąć jedną z trzech ról scrumowych. Menedżerowie projektów są doskonałym materiałem na mistrzów młyna, jeśli tylko potrafią pozbyć się nawyku rządzenia i kontrolowania.

Analizując dokładniej tabelę 13.3, zauważysz również, że rola właściciela produktu zakłada nie mniejszą liczbę obowiązków niż rola mistrza młyna. Zatem menedżer projektu może dokonać transformacji we właściciela produktu, o ile tylko posiada odpowiednią wiedzę domenową i inne umiejętności potrzebne do skutecznego wykonywania tej roli. Rzadziej spotykanym przypadkiem jest przyjęcie przez menedżera projektu z odpowiednią wiedzą techniczną roli członka zespołu deweloperskiego.

**TABELA 13.3.** Dystrybucja obowiązków zarządzania projektem w organizacji scrumowej

Aktywność zarządzania projektem	Właściciel produktu	Mistrz młyna	Zespół deweloperski	Inny menedżer
Integracja	✓			✓
Zakres	Poziom makro		Poziom sprintu	
Czas	Poziom makro	Pomaga zespołowi produktywnie wykorzystać czas	Poziom sprintu	
Koszt	✓		Historyjka/zadanie	
Jakość	✓	✓	✓	✓
Zespół (zasoby ludzkie)			✓	Tworzenie
Komunikacja	✓	✓	✓	✓
Rzyko	✓	✓	✓	✓
Zaopatrzenie	✓			✓

## Pozostanie przy roli niezależnego menedżera projektu

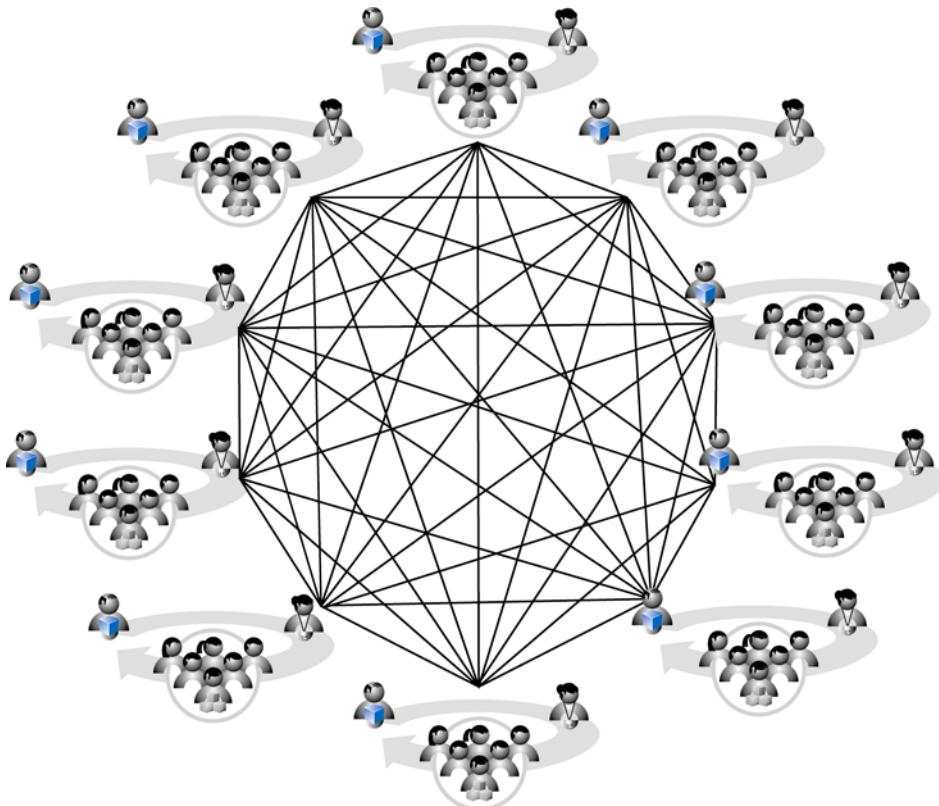
Można by przyjąć, że menedżerowie projektów stają się mistrzami młyna, właścicielami produktów lub członkami zespołu deweloperskiego. Jednak nie we wszystkich przypadkach ma miejsce jedna z tych trzech transformacji. Firmy prowadzące duże i skomplikowane projekty deweloperskie decydują się pozostawić rolę niezależnego menedżera projektu, kiedy zadania logistyczne i koordynacyjne są na tyle przytaczające, że trudno oczekiwąć realizowania ich na bieżąco przez same zespoły.

Ja osobiście wolę, kiedy całą logistiką i koordynacją zajmują się mistrzowie młyna przypisani do danego wysiłku deweloperskiego. Zespoły scrumowe nie powinny oczekwać, że ktoś spoza zespołu zajmie się koordynowaniem na ich rzecz. Takie podejście powoduje myślenie w stylu „Jeśli ktoś inny jest odpowiedzialny za koordynację, w takim razie my nie jesteśmy”.

Logistyka i wzajemne zależności w małych projektach deweloperskich realizowanych przez kilka zespołów scrumowych mogą być bez problemu obsłużone na zasadzie codziennej koordynacji międzyzespołowej (przy użyciu takich technik jak scrum scrumów — patrz rozdział 12.). Ale co zrobić, jeśli w naszym wysiłku deweloperskim biorą udział dziesiątki, a nawet setki zespołów scrumowych, czyli setki lub nawet tysiące deweloperów?

Podobnie jak w przypadku zasady „jeden produkt, jeden rejestr produktu” przedstawionej w rozdziale 6. zespoły powinny zacząć od zasady „każdy zespół koordynuje się samodzielnie”. Dopuszczamy jednak poluzowanie wymagania jednego rejestru na jeden produkt w przypadku napotkania problemu skali. Na tej samej zasadzie możemy być zmuszeni do pozostawienia jednego lub więcej menedżerów produktów lub programów celem koordynacyjnego wsparcia wszystkich ruchomych części.

Zanim jednak pospieszymy ścieżką pozostawienia roli poświęconej wyłącznie koordynacji ze względu na dużą liczbę zespołów, powinniśmy zrobić krok wstecz i przyjrzeć się kanałom komunikacji pomiędzy samymi zespołami. Z mojego doświadczenia w tego typu sytuacjach wynika, że rzadko kiedy kanały komunikacyjne pomiędzy wszystkimi zespołami są w pełni podpięte (patrz rysunek 13.5).



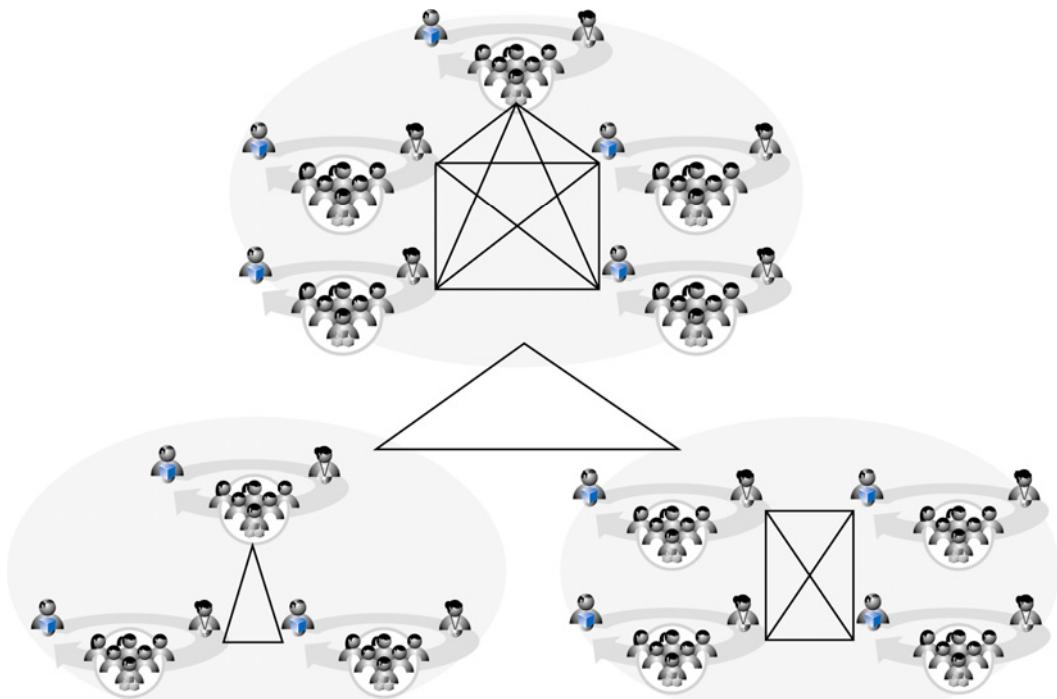
**RYSUNEK 13.5.** Rzadko zdarza się, aby zespoły posiadały w pełni podpięte kanały komunikacyjne

Bardziej prawdopodobne jest to, że zespoły łączą się w obszary związane z daną cechą lub jej odpowiednikiem i tam kanały komunikacyjne działają z odpowiednią intensywnością, natomiast połączenia między klastrami są luźniejsze (patrz rysunek 13.6).

W takich przypadkach zespoły scrumowe mogą z łatwością koordynować zależności w ramach klastra, ale kto zajmie się koordynacją pomiędzy klastrami? Domyślona odpowiedź brzmi: same zespoły. W wielu przypadkach takie podejście funkcjonuje zupełnie poprawnie. Wzajemna współpraca może zostać zrealizowana w formie scruma scrumów, w trakcie którego przedstawiciele klastrów spotykają się ze sobą i dyskutują problemy wymagające koordynacji.

Jeżeli istnieje duża liczba różnych klastrów, nawet tego typu koordynacja w stylu scruma scrumów może okazać się trudna. Widziałem, że w takich przypadkach organizacje decydują się kierować wysiłkiem koordynacyjnym poprzez menedżera projektu lub produktu (patrz rysunek 13.7).

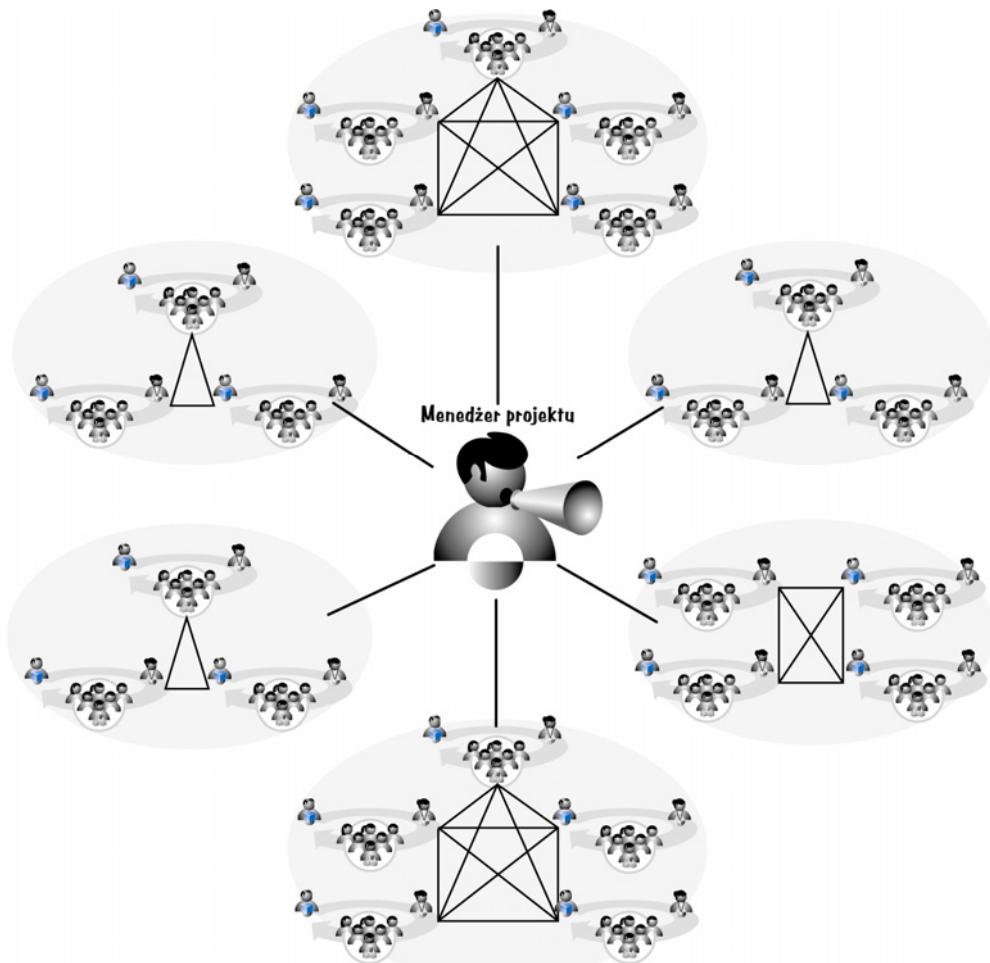
Osobiście wolałbym nie mieć menedżera projektu w centrum koordynacji. Takie podejście rodzi ryzyko przesuwania odpowiedzialności za koordynację swoich działań przez zespoły scrumowe na inny podmiot.



**RYSUNEK 13.6.** Zespoły często formują klastry wzajemnej współpracy

Potrafię jednak zrozumieć, że przy odpowiednio dużej skali przedsięwzięcia posiadanie osoby lub osób skupiających cały czas na działaniach logistycznych i koordynacyjnych daje komfort utrzymywania pałeczki w powietrzu. Aby jeszcze bardziej podkreślić, że indywidualne zespoły nie mogą oddelegowywać swoich zadań koordynacyjnych wewnętrz klastra na kogoś innego, staram się myśleć o menedżerze projektu jak o asystencie (na zasadzie lidera służby) zespołów scrumowych. Od menedżera projektu w tej roli oczekuje się patrzenia w perspektywie całego systemu i pilnej współpracy z każdym klastrem lub indywidualnymi zespołami w celu zapewniania, iż wszyscy posiadają odpowiednią świadomość konieczności koordynacji działań między zespołami. Sam obowiązek koordynacji spoczywa jednak na zespołach.

Takie wykorzystanie menedżera projektu może być również pomocne w przypadku wysiłków deweloperskich, w których Scrum jest jedynie małym wycinkiem znacznie większego projektu mającego na celu stworzenie produktu lub usługi. Przykładami takich sytuacji są podwykonawcy, wewnętrzne zespoły nieużywające Scruma oraz inne wewnętrzne działy związane z dostarczaniem produktu. Logistyka niezbędna przy współpracy z podwykonawcami może być zajęciem bardzo angażującym i czasochłonnym. Przy tak wielu ruchomych częściach dobrze jest mieć kogoś skupionego całkowicie na logistyce (patrz rysunek 13.8).

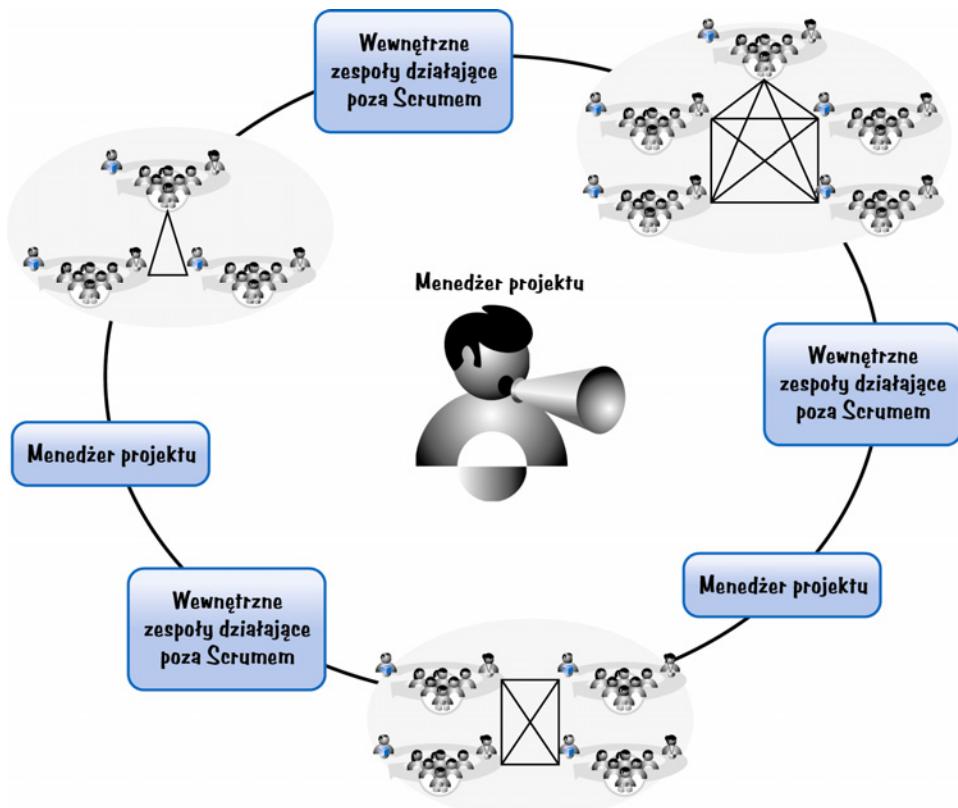


**RYSUNEK 13.7.** Kierowanie koordynacją przez menedżera projektu

Jeszcze raz należy podkreślić, że celem nie jest powierzenie kierownictwa menedżerowi projektu. Osoba ta ma dbać o to, aby wszyscy dobrze rozumieli zależności pomiędzy różnymi obszarami oraz komunikowali się w sposób pozwalający jak najefektywniej koordynować własną pracę z innymi zespołami.

## Zakończenie

W tym rozdziale opisałem rolę menedżerów obszarów funkcjonalności w organizacji używającej Scruma. Pogrupowałem ich obowiązki w kategorie kształtowania zespołów, pielęgnacji zespołów, przystosowywania i adaptowania środowiska, a także zarządzania przepływem wytwarzania wartości.



**RYSUNEK 13.8.** Menedżer projektu w złożonym projekcie, w którym udział bierze wiele podmiotów

Podsumowanie obowiązków menedżerów obszarów funkcjonalności w organizacji o tradycyjnym modelu produkcji i w Scrumie przedstawia tabela 13.4.

Chociaż większość tego rozdziału poświęcona była menedżerom obszarów funkcjonalności, na koniec zająłem się również rolą menedżera projektu. Wskazałem obowiązki tej roli w tradycyjnym modelu produkcji, jak również obowiązki przekładające się na trzy podstawowe role w Scrumie. Wspomniałem o złożonych sytuacjach, w których organizacje uważają za pomocne wprowadzenie jednego lub więcej menedżerów produktu obok trzech podstawowych ról scrumowych.

Ten rozdział kończy drugą część książki. W następnym rozdziale zajmiemy się planowaniem, zaczynając od opisania istotnych zasad planowania w Scrumie.

**TABELA 13.4.** Porównanie zadań menedżera obszaru funkcjonalności w tradycyjnym środowisku produkcji i w Scrumie

Tradycyjny model produkcji	Scrum
Przypisuje ludzi do projektów	Współtworzy wspaniałe zespoły
Zatrudnia i zwalnia	Tak samo
Skupia uwagę na rozwoju pracowników	Tak samo
Ocenia pracowników	Nadal zaangażowany, ale często informacja zwrotna ma większe znaczenie i musi być spięta z zespołem
Przypisuje zadania członkom zespołu (czasem)	Pozwala członkom zespołu na samodzielne organizowanie się i wybieranie swoich zadań
Ustanawia standardy dla obszaru funkcjonalności obowiązujące między projektami	Tak samo
Zachęca do inicjatyw w specyficznych obszarach funkcjonalności	Tak samo
Posiada dobrą wiedzę praktyczną w obszarze funkcjonalności i jest w stanie pomóc, kiedy zachodzi taka potrzeba	Tak samo
Jest wytrenowany w przenoszeniu bezpośrednich podwładnych między zespołami	Skupia się na utrzymaniu integralności zespołu
Usuwa przeszkody	Tak samo
Skupia się na własnym obszarze funkcjonalności	Przyjmuje perspektywę postrzegania całości dla lepszego przystosowania i wytwarzania wartości
Zarządza ekoniemią (rachunek zysków i strat)	Tak samo
Monitoruje wskaźniki i raportuje	Dostosowuje mierzone rzeczy oraz tworzone raporty do zasad zwinności, aby skupiać się na przepływie wytwarzania wartości



Część III

## PLANOWANIE

---



## Rozdział 14

# ZASADY PLANOWANIA W SCRUMIE

---

Istnieje obiegowe przekonanie, że prace deweloperskie w Scrumie startują bez planowania. Zaczynamy pierwszy sprint i wymyślamy resztę szczegółów „w locie”. To nieprawda. W Scrumie prowadzamy prawdziwe planowanie. Dokładnie mówiąc, planujemy na wielu różnych poziomach szczegółowości i w wielu różnych momentach. Niektórym osobom może się wydawać, że przykłada my mniejszą uwagę do planowania, ponieważ większość z aktywności tego typu ma miejsce w samą porę zamiast na samym początku prac. Z moich doświadczeń wynika jednak, że zespoły pracujące w Scrumie spędzają więcej czasu na planowaniu, niż miałyby to miejsce w modelu tradycyjnym.

W tym rozdziale rozwinę kilka z zasad Scruma opisanych w rozdziale 3., skupiając się na tym, jak przekładają się one na planowanie. W ten sposób przygotuję sobie grunt do rozważań w rozdziale 15. na temat różnych poziomów planowania, jakie mają miejsce w Scrumie. W kolejnych rozdziałach opiszę szczegółowo planowanie portfela, produktu, wersji dystrybucyjnej i sprintu.

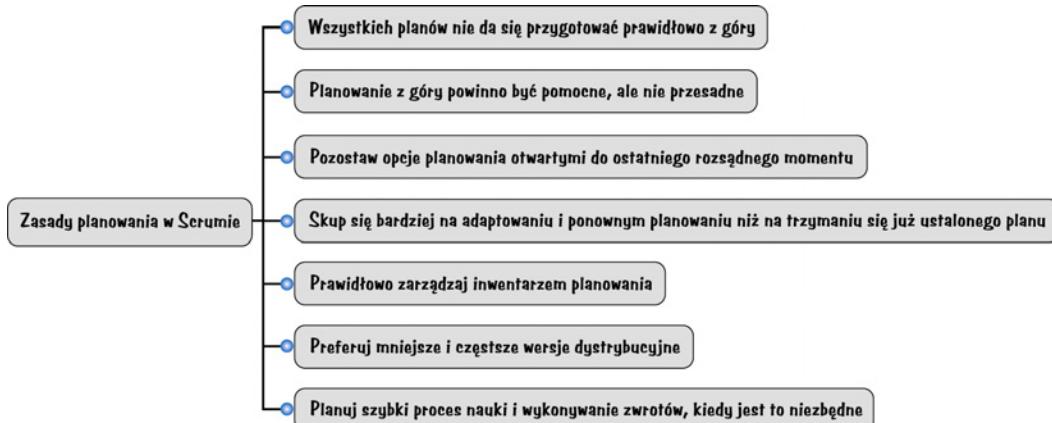
## Wprowadzenie

Rozdział 3. opisuje kluczowe zasady Scruma, z których część ma fundamentalne znaczenie dla naszego podejścia do planowania w tym środowisku. Ten rozdział podkreśla zasady przedstawione na rysunku 14.1.

Pozostałe zasady Scruma (takie jak praca w krótkich przedziałach czasu, stosowanie taktu itp.) zostaną podkreślone w kolejnych rozdziałach poświęconych planowaniu.

## Nie zakładaj, że z góry uda Ci się wszystko dobrze zaplanować

Tradycyjne, przewidujące podejście do planowania zakłada stworzenie szczegółowego planu przed rozpoczęciem prac deweloperskich. Celem jest zrobienie go w prawidłowy sposób, tak aby reszta prac mogła pójść według ustalonego porządku. Niektórzy argumentują, że bez tego planu nie będziemy wiedzieć, w jakim kierunku zmierzamy, i nie będziemy w stanie koordynować ludzi oraz ich działań, szczególnie w przypadku większych wysiłków deweloperskich wymagających pracy wielu zespołów. Ten argument zawiera ziarno prawdy.



**RYSUNEK 14.1.** Zasady planowania w Scrumie

Scrumowe podejście do planowania jest w zgodzie z fundamentalną zasadą inspekcji i adaptacji. Produkując przy użyciu Scruma, uważamy, że nie jesteśmy w stanie przewidzieć wszystkiego w prawidłowy sposób z góry i dlatego nie próbujemy stworzyć *wszystkich* artefaktów planowania już na samym początku. Produkujemy jednak wcześniej *pewne* artefakty w ilości zapewniającej odpowiednią równowagę pomiędzy planowaniem z góry i planowaniem w samą porę.

## Planowanie z góry powinno być pomocne, ale nie przesadne

Przyjrzyjmy się przykładowi ilustrującemu zasadę mówiącą, iż *planowanie z góry powinno być pomocne, ale nie przesadne*.

Mieszkam w Colorado, gdzie narciarstwo stoi na światowym poziomie. Od czasu do czasu jeżdżę na nartach w celach rekreacyjnych — nie można powiedzieć, abym był ekspertem w tej dziedzinie sportu. Mój przyjaciel — John — uprawia narciarstwo w sposób ekstremalny. Szczerze przyznam, że czasem też chciałbym jeździć na nartach w taki sposób, ale nie mam odpowiednich umiejętności i nie jestem dostatecznie szalony. Pewnego razu John chwalił się zdjęciami z przygód na szczególnie niebezpiecznej górze. Z ciekawości zadałem mu pytanie: „Czy kiedy stoisz na górze i przygotowujesz się do zjazdu, planujesz całą tracę do samego dołu?”.

Kiedy skończył się śmiać, odpowiedział: „Nie, gdybym tak zrobił, najzwyczajniej w świecie zabili bym się” i dodał: „Wybieram jakieś miejsce w dole i przyjmuję sobie za cel dotarcie do niego. Planuję być może pierwsze dwa lub trzy skręty. Patrząc na to realistycznie, jakkolwiek planowanie dalej byłoby niemożliwe i niebezpieczne”.

Zapytałem: „Dlaczego?”.

Odpowiedział: „Widoczna powierzchnia śniegu nie jest tym, czym się wydaje z góry, ponieważ oświetlenie i inne czynniki oszukują wzrok. Poza tym gdzieś tam w dole prawie na pewno są jakieś drzewa, których nie widzę ze szczytu — jeżeli będąc na górze, zdecyduję się zrobić skręt w prawo i potem faktycznie wykonam to założenie, mogę wjechać prosto w te drzewa. Do tego nie da się przewidzieć 15-latka na snowboardzie przelatującego nad moją głową i krzyczącego »Uważaj, chłopie!«. Nigdy nie wiadomo, kiedy będziesz musiał zmienić kierunek i z jakiego powodu”.

Przypominam sobie, że kiedy skończył swoje wyjaśnienia, pomyślałem sobie „To mi przypomina pewien interesujący projekt deweloperski, nad którym pracowałem”. Nigdy nie jesteś w stanie przewidzieć z dużą dokładnością, kiedy będziesz musiał zmienić kierunek lub dlaczego. Tam, gdzie proszono mnie o przygotowanie szczegółowego wstępnego planu, tworzyłem taki, ale nie przypominam sobie ani jednego przypadku, kiedy takie podejście zadziałało. Nie przypominam sobie, abym po skończeniu dzieła kiedykolwiek spojrzał wstecz i powiedział „Zrobiliśmy to perfekcyjnie!”. W pewnym sensie wczesne planowanie przypomina próbę przewidzenia każdego skrętu, stojąc na szczytce wznieśienia. Planowanie na takim poziomie szczegółowości jest marnotrawstwem. Wiara w plan do tego stopnia, iż ignoruje się rzeczywiste dane, jest skrajnie niebezpieczna.

Większość z nas była kiedyś zaangażowana w prace deweloperskie, podczas których poziom szczegółowości początkowego planowania był wręcz absurdalny. Czy to oznacza, że nie powinniśmy w ogóle planować z góry? Nie, takie postępowanie byłoby niedbalstwem i szaleństwem. John niewątpliwie przygotowuje plan początkowy — studiuje główne cechy terenu, aby zyskać większą pewność siebie przed ruszeniem w dół. Szaleństwem byłoby jednak planować do takiego momentu, w którym reakcja na rzeczywiste dane będzie bardzo trudna lub kosztowna. Podobnie jak John musimy znaleźć odpowiednią równowagę pomiędzy przewidywaniem z góry i planowaniem w samą porę.

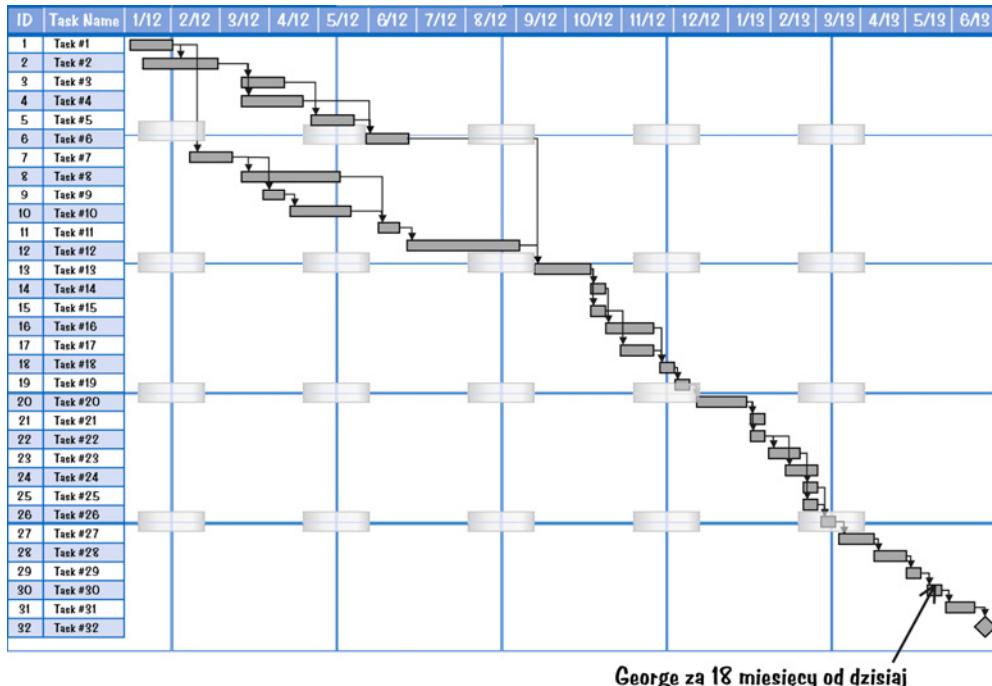
## Pozostaw opcje planowania otwartymi do ostatniego rozsądnego momentu

Chcąc osiągnąć prawidłową równowagę pomiędzy planowaniem z góry i planowaniem w samą porę, powinniśmy kierować się zasadą pozostawiania istotnych decyzji otwartymi do ostatniego momentu, kiedy ich podjęcie ma jeszcze racjonalny sens. Oznacza to wstrzymanie się z planowaniem w samą porę do momentu, kiedy będziemy posiadać lepsze informacje. Po co podejmować wczesne decyzje w oparciu o kiepskie dane? Przedwczesne decyzje mogą być bardzo kosztowne, a także — jak zauważył John — niebezpieczne.

## Skup się bardziej na adaptowaniu i ponownym planowaniu niż na trzymaniu się już ustalonego planu

Jednym z problemów występujących w wielu projektach deweloperskich jest przywiązywanie zbyt dużej uwagi do ustalonego z góry planu oraz zbyt małej uwagi do planowania w sposób ciągły. Jeżeli spędżymy dużą ilość czasu na początku, tworząc wysoce przewidywalny plan, i będziemy uznawać, że jest poprawny, cały impet pojedzie w kierunku trzymania się tego planu, zamiast aktualniania go tak, aby odzwierciedlał zmiany. Przyjmując scrumowy sposób myślenia, który zakłada, że nie da się stworzyć wszystkich planów z góry i nie można wyeliminować zmian, bardziej niż przestrzeganie planu będziemy cenić możliwość zareagowania na zmiany i przeplanowania.

W latach osiemdziesiątych ubiegłego wieku pomagałem w tworzeniu dużych planów lub pracowałem jako konsultant w firmach, które same tworzyły takie plany. Wiesz zapewne, o jakiego typu planach mówię — dużych wykresach Gantta, które drukowaliśmy (na wielu stronach), sklejaliśmy razem i wieszaliśmy na ścianie (rysunek 14.2).



**RYSUNEK 14.2.** Duży plan w formie wykresu Gantta

W kilku przypadkach spędziliśmy do sześciu tygodni, tworząc wysoce przewidywalne plany. Po stworzeniu stawały się one mapą dla projektów. Podobnie jak system prawny zakłada „niewinność do momentu udowodnienia winy”, plany te „zakładają poprawność do momentu udowodnienia niepoprawności”. Do sytuacji tej wydaje się pasować pewne powiedzenie przypisywane często armii szwajcarskiej, ale pochodzące prawdopodobnie z *Przewodnika przetrwania SAS* [Wiseman, 2010]: „Kiedy zgubisz się w lesie, a mapa nie zgadza się z terenem, we wszystkich przypadkach bardziej zaufaj terenowi” (patrz rysunek 14.3).

W przypadku dowolnego produktu nasza mylna wiara w mapę doprowadzała nas do konkluzji, że postęp może i powinien być mierzony jako zgodność lub odstępstwo od oryginalnego planu. Kiedy pojawiają się odstępstwa od planu, nasze pragnienie trzymania się go zaśłania przed nami fakt, iż sama mapa może być błędna. Jeżeli mapa stanie się ważniejsza niż teren, oznacza to, że straciliśmy poczucie rzeczywistości, w której musimy się poruszać.

Używając Scruma, uznajemy wstępny plan za pomocny, ale uważamy za niezbędne odczytywanie ukształtowania terenu i adaptowanie się do niego. Jest to rozsądne przekonanie, biorąc pod uwagę fakt, że układając plan, mamy najmniejszą możliwą wiedzę na temat produktu w całym cyklu jego życia. W związku z tym plany wstępne bardzo dokładnie odzwierciedlają naszą początkową ignorancję.

W Scrumie preferujemy planowanie po zweryfikowaniu założeń. Używamy naszej potwierdzonej wiedzy do nieustannego tworzenia lepszych, bardziej użytecznych planów. Nie martwimy się tym, że nasze plany mogą być błędne, ponieważ wiemy, że wkrótce zastąpimy je planami bardziej dokładnymi. Pracujemy w krótkich sprintach trwających od kilku tygodni do nie więcej niż miesiąca, więc nawet jeśli jesteśmy w błędzie, nie spędzimy zbyt dużo czasu na złej ścieżce, zanim będziemy mieli okazję dostosować kurs.



**RYSUNEK 14.3.** Kiedy mapa nie zgadza się z terenem, zaufaj terenowi

Chociaż większość znanych mi zespołów scrumowych nie stosuje wykresów Gantta, cenią one sobie pewną formę planowania długoterminowego i je przeprowadzają. Dokładniej rzecz biorąc, zespoły scrumowe planują na różnych poziomach szczegółowości, o czym będę pisał w rozdziale 15. To, czego nie chcemy, to zbyt mocne przywiązywanie do planów uniemożliwiające ich modyfikację w przypadku zmiany lub poznania istotnych informacji, na które trzeba w jakiś sposób zareagować.

## Prawidłowo zarządzaj inwentarzem pracy

W rozdziale 3. omówiłem obowiązującą w Scrumie kluczową zasadę zarządzania inwentarzem (pracą cząstkową). Zachowanie odpowiedniej równowagi pomiędzy planowaniem z góry i planowaniem w samą porę wymaga uwiadomienia sobie jednej istotnej rzeczy — tworzenie dużego inwentarza przewidywanych, niepotwierdzonych artefaktów planowania niesie ze sobą ryzyko poważnego marnotrawstwa.

Prawidłowe zarządzanie naszym inwentarzem artefaktów planowania ma znaczenie ekonomiczne.

Weźmy poprzedni przykład dużego, zaplanowanego z góry wykresu Gantta. W miarę postępu prac deweloperskich weryfikujemy nasze założenia poprzez zdobywanie wiedzy na temat tego, co faktycznie robimy, i dowiadujemy się, gdzie nasz oryginalny plan był błędny. Niestety w tym momencie możemy jedynie żałować marnotrawstwa wynikającego z konieczności wycofania się z tamtych założeń i ponownego zaplanowania przyszłości w świetle właśnie zdobytej wiedzy.

W ten sposób powstają minimum trzy formy marnotrawstwa. Po pierwsze, zmarnowany wysiłek włożony w wytworzenie części planów, które teraz trzeba porzucić. Po drugie, potencjalne ryzyko dużego marnotrawstwa podczas uaktualniania tego planu i po trzecie, zmarnowane szanse zainwestowania naszego czasu w bardziej wartościowe działania (takie jak dostarczanie działającego oprogramowania o wysokiej jakości) zamiast robienia czegoś, co i tak będzie później wymagało poprawek.

Zawsze staram się zestawić ilość planowania w danym czasie z prawdopodobieństwem, że to, co robię, wpłynie jedynie na wzrost marnotrawstwa w przypadku następzenia zmiany. Na przykład na rysunku 14.2 mógłbym mieć umieszczone imię George obok zadania, którego planowane rozpoczęcie ma miejsce za 18 miesięcy od dnia dzisiejszego. Jakie Twoim zdaniem są szanse, że George będzie faktycznie pracował nad tym zadaniem za 18 miesięcy? Zapewne bliskie零!

Skoro istnieje tak duże ryzyko błędnego przewidywania odlegiej przyszłości przez plan, po co planować aż tak daleko? Zazwyczaj robimy to, chcąc odpowiedzieć na pytania takie jak: „Kiedy skończymy?” lub „Ilu ludzi będziemy potrzebowali do tego projektu?”. Czy możemy odpowiedzieć z jakąkolwiek pewnością na te pytania, nie przewidując całej potrzebnej do wykonania pracy?

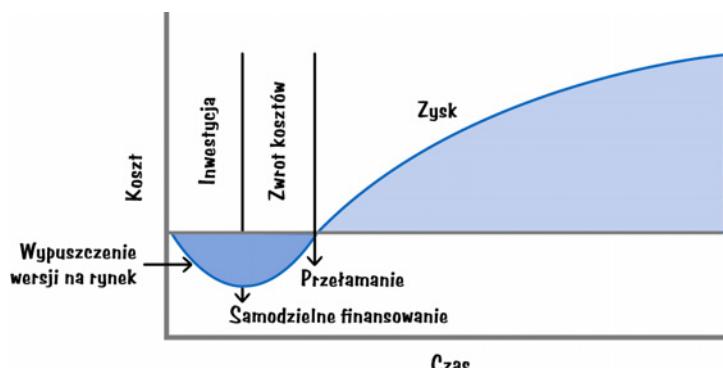
To, kiedy produkt zostanie wypuszczony na rynek i jakie cechy zdążymy w nim umieścić przed ustaloną datą, są ważnymi kwestiami, z którymi trzeba umieć sobie poradzić. Nie możemy jednak oszukiwać sami siebie myśleniem, że znamy prawidłowe odpowiedzi, tylko dlatego że przeprowadziliśmy długofalowe zgadywanie o niskim stopniu pewności. W kilku kolejnych rozdziałach poświęć uwagę tego typu pytaniom z perspektywy planowania.

## Preferuj mniejsze i częstsze wersje dystrybucyjne

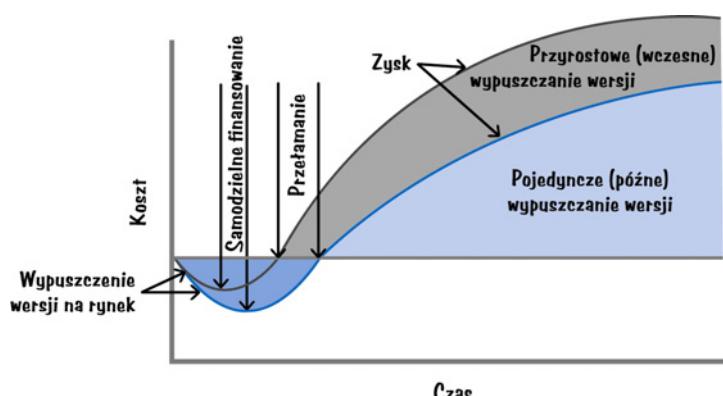
W Scrumie preferowane są mniejsze i częstsze wersje dystrybucyjne, ponieważ umożliwiają one szybsze zdobycie informacji zwrotnej oraz zysku z inwestycji. Niemal we wszystkich przypadkach jesteśmy w stanie poprawić dochody z cyklu życia produktu dzięki stosowaniu produkcji przyrostowej i wypuszczaniu wielu wersji dystrybucyjnych zawierających małe porcje cech nadających się do sprzedaży.

Przyjrzyj się kosztom i zyskom produktu o pojedynczej wersji dystrybucyjnej pokazanym na rysunku 14.4 (źródło: [Denne i Cleland-Huang, 2003]). Na początku prac deweloperskich wydajemy pieniądze (rozpoczynamy okres inwestycyjny) bez żadnego zwrotu. Wypuszczenie produktu ma miejsce tam, gdzie krzywa biegnie w dół podczas okresu inwestycyjnego. W końcu docieramy do punktu, w którym możemy samodzielnie finansować swoją działalność — jest to miejsce zrównoważenia się kosztów produkcji z osiąganyimi zyskami. Kiedy zyski przeważą nad kosztami, wchodzimy w okres zwrotu — zaczynamy odzyskiwać to, co zainwestowaliśmy. Kiedy całkowity zysk zrówna się z całkowitym kosztem, docieramy do punktu przełamania. Od tego momentu zaczynamy w końcu odnotowywać zyski.

W celu zilustrowania zalet mniejszych i częstszych wdrożeń założymy, że będziemy mieć dwie wersje dystrybucyjne zamiast jednej (patrz rysunek 14.5). W tym przypadku szybciej docieramy do punktów samodzielnego finansowania, przełamania i zysku, poprawiając tym samym całkowity zwrot z inwestycji w produkt.



RYSUNEK 14.4. Ekonomia pojedynczej wersji dystrybucyjnej



RYSUNEK 14.5. Ekonomia wielu wersji dystrybucyjnych

W ramach przykładu (w oparciu o [Patton, 2008]) przyjrzyj się poprawie zysków z inwestycji w modelu z założeniami podanymi w tabeli 14.1.

TABELA 14.1. Założenia dla modelu zwrotu z inwestycji

Zmienna	Wartość
Dochód (wszystkie cechy)	300 tys. zł/miesiąc
Dochód (połowa cech)	200 tys. zł/miesiąc
Dochód (jedna trzecia cech)	150 tys. zł/miesiąc
Opóźnienie od wdrożenia do przychodów	1 miesiąc
Koszt produkcji	100 tys. zł/miesiąc
Koszt wdrożenia	100 tys. zł na każde wdrożenie

Wyniki z tabeli 14.2 pokazują, że pojedyncze wdrożenie wersji po 12 miesiącach ma współczynnik zwrotu z inwestycji rzędu 9,1%. Jeżeli zamiast tego zdecydujemy się na wdrożenia co pół roku, wskaźnik ten wzrośnie do 15,7%, a w przypadku wdrożeń co kwartał osiągniemy zwrot na poziomie 19,5%.

**TABELA 14.2.** Zwrot z inwestycji dla różnych cyklów wdrażania

	Pojedyncze wdrożenie (12 miesięcy)	Wdrożenia co pół roku	Wdrożenia co kwartał
Całkowity koszt	1,3 mln zł	1,4 mln zł	1,6 mln zł
Całkowity przychód z dwóch lat	3,6 mln zł	4,8 mln zł	5,25 mln zł
Całkowity zysk netto z dwóch lat	2,3 mln zł	3,4 mln zł	3,65 mln zł
Zainwestowana gotówka	1,3 mln zł	0,7 mln zł	0,45 mln zł
Wewnętrzna stopa zwrotu (w zastępstwie wskaźnika ROI)	9,1%	15,7%	19,5%

Takie podejście ma też swoje ograniczenia. Po pierwsze, dla każdego produktu istnieje minimalny zestaw cech kwalifikujących do wdrożenia lub sprzedaży. W związku z tym nie możemy w nieskończoność zmniejszać rozmiarów wersji, ponieważ w pewnym momencie stanie się ona zbyt mała, aby nadawać się do sprzedaży. Poza tym małe i częste wersje dystrybucyjne nie będą pasować do pewnych typów rynków. Niemniej jednak jeśli Twój rynek jest otwarty na otrzymywanie częściowej wartości wcześniej, powinieneś przestrzegać tej ważnej zasady częstszych i mniejszych wersji dystrybucyjnych.

## Planuj szybki proces nauki i wykonywanie zwrotów, kiedy jest to niezbędne

Żadna ilość prób przewidzenia lub odgadnięcia czegoś z góry nie zastąpi wykonania konkretnej pracy, szybkiego poznania jej wyników i, w miarę potrzeby, dokonania zwrotu. Przez zwrot rozumiem zmianę kierunku działań przy jednoczesnym bazowaniu na zdobytej właśnie wiedzy. Ries definiuje dokonywanie zwrotów jako „strukturyzowaną korekcję kursu, zaprojektowaną do przetestowania nowych kluczowych hipotez odnośnie do produktu, strategii i mechanizmów wzrostu” [Ries, 2011]. Tak jak narciarz John musimy być przygotowani na wykonanie szybkiego zwrotu, kiedy tylko dowiemy się, że bieżący plan stracił sens.

Jak pisałem w rozdziale 3., naszym celem jest szybkie i ekonomiczne przejście przez pętlę zdobywania wiedzy. Zatem tworząc nasze plany, powinniśmy pamiętać, że kluczowym celem jest zdobywanie wiedzy. Otrzymując szybką informację zwrotną, możemy stwierdzić, czy nasze plany prowadzą nas w dobrym kierunku. Jeśli tak nie jest, możemy wykonać zwrot w nowym kierunku.

## Zakończenie

W tym rozdziale opisałem kilka z zasad planowania w Scrumie. Zasady te pozwalają nam planować w sposób ekonomiczny poprzez ograniczenie aktywności tego typu na początku i zrównoważenie ich bardziej szczegółowym planowaniem w samą porę, kiedy wiemy więcej na temat tego, co budujemy, i jak należy to coś zbudować. W kolejnych pięciu rozdziałach pokażę na przykładach, jak zastosować te zasady w praktyce w szerszym kontekście planowania scrumowego na wielu poziomach szczegółowości.

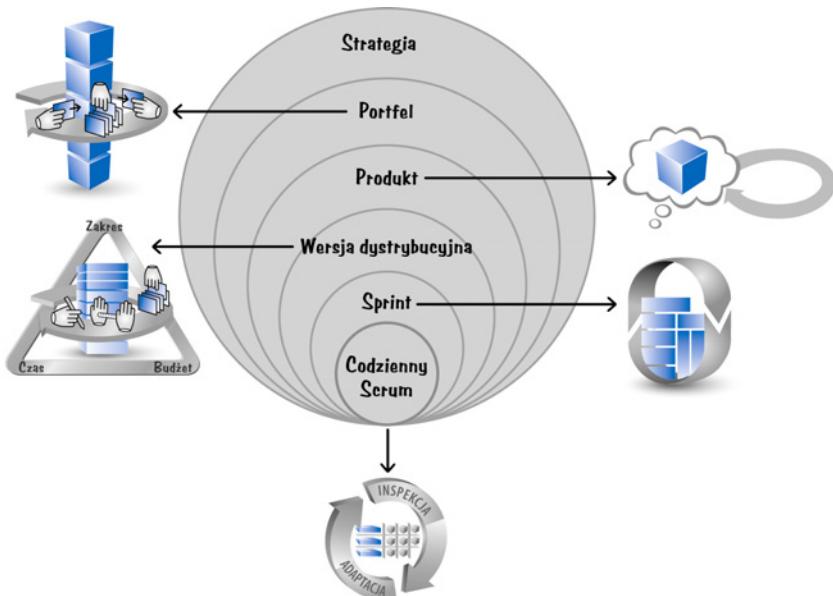
## Rozdział 15

# PLANOWANIE WIELOPOZIOMOWE

W projektach scrumowych planujemy wielokrotnie w czasie produkcji oprogramowania i na wielu różnych poziomach. W tym rozdziale przedstawię ogólny opis całej hierarchii różnorodnych aktywności związanych z planowaniem w Scrumie oraz ich powiązań między sobą, poczynając od samej góry i idąc w dół. W kilku kolejnych rozdziałach rozwinę głębiej tematy planowania portfela, produktu (przedstawianie wizji), wersji dystrybucyjnej, sprintu.

## Wprowadzenie

Podczas wytwarzania produktu w Scrumie planowanie odbywa się wielokrotnie i na różnych poziomach (patrz rysunek 15.1).



**RYSUNEK 15.1.** Różne poziomy planowania

Na najwyższym poziomie znajduje się planowanie strategii — ma ono fundamentalne znaczenie dla osiągnięcia sukcesu przez firmę, ale wykracza poza zakres tej książki. Formalnie Scrum definiuje jedynie planowanie sprintu oraz planowanie codzienne (poprzez codzienne działania scrumowe). Niemniej jednak większość organizacji może skorzystać na planowaniu portfela, produktu i wersji dystrybucyjnej, dlatego podsumuję podejścia do każdego z nich w tym rozdziale, a następnie omówię je bardziej szczegółowo w dalszych rozdziałach.

Tabela 15.1 podsumowuje pięć typów planowania, zwracając uwagę na potrzebny czas, uczestników, przedmiot będący tematem planowania i wyniki planowania na każdym poziomie.

**TABELA 15.1.** Szczegóły planowania na poszczególnych poziomach

Poziom	Horyzont czasowy	Kto	Przedmiot planowania	Wyniki planowania
Portfel	Możliwie rok lub dłużej	Interesariusze i właściciele produktów	Zarządzanie portfelem produktów	Rejestr portfela i zbiór produktów
Produkt (wizja produktu)	Do kilku miesięcy lub dłużej	Właściciel produktu, interesariusze	Ewolucja wizji i produktu wraz z upływem czasu	Wizja produktu, mapa drogowa i cechy na wysokim poziomie ogólności
Wersja dystrybucyjna	Od trzech (lub mniej) do dziewięciu miesięcy	Cały zespół scrumowy, interesariusze	Nieustanne równoważenie wartości dla klienta i jakości z ograniczeniami zakresu, harmonogramem i budżetem	Plan wersji dystrybucyjnej
Sprint	Każda iteracja (od jednego tygodnia do jednego miesiąca kalendarzowego)	Cały zespół scrumowy	Jakie cechy dostarczyć w następnym sprincie	Cel sprintu i rejestr sprintu
Codzienne działania scrumowe	Każdego dnia	Mistrz młyna, zespół deweloperski	Jak zrealizować cechy, które zespół zobowiązał się zrealizować	Inspekcja bieżących postępów i adaptacja mająca na celu jak najlepszą organizację nadchodzącego dnia pracy

Do zilustrowania planowania na każdym poziomie użyję przykładu przeprojektowywania witryny internetowej Scrum Alliance ([www.scrumalliance.org](http://www.scrumalliance.org)). Organizacja ta skupiła się w 2006 roku na światowej promocji Scruma — niestety jej witryna była wtedy w okropnym stanie. Wyglądała przeciennie, była trudna w nawigacji i uboga w treść. Kiedy obejmowałem stanowisko dyrektora zarządzającego Scrum Alliance pod koniec roku 2006, jedną z pierwszych rzeczy, o jaką poprosiła rada dyrektorów, było stworzenie nowej, znacznie lepszej witryny. Stałem się właścicielem produktu dla tego przedsięwzięcia. Opiszę całą hierarchię planowania, jakie przeprowadziliśmy w celu stworzenia nowej strony internetowej.

## Planowanie portfela

Planowanie portfela (lub zarządzanie portfelem) to aktywność mająca na celu stwierdzenie, nad jakimi produktami należy pracować, w jakim porządku i jak długo. Chociaż planowanie portfela jest koncepcyjnie na wyższym poziomie niż planowanie produktu (ponieważ dotyczy całego zbioru produktów), jedną z danych wejściowych podczas jego przeprowadzania jest nowo powstała wizja produktu powstała w trakcie planowania produktu.

W 2006 roku Scrum Alliance była względnie młodą organizacją, a jej portfel zawierał wyłącznie trwający nieustannie proces rozwoju własnej witryny internetowej. Po przedstawieniu wstępnej wizji nowej strony internetowej Scrum Alliance rada dyrektorów (interesariusze rejestru portfela Scrum Alliance) zatwierdziły prace deweloperskie nad pierwszą wersją dystrybucyjną witryny.

## Planowanie produktu (tworzenie wizji produktu)

Celem planowania na poziomie produktu (które ja określam również mianem **tworzenia wizji produktu**) jest uchwycenie esencji potencjalnego produktu i stworzenie ogólnego planu jego budowy. Planowanie zaczyna się od przedstawienia samej wizji, a następnie stworzenia rejestru produktu na dużym poziomie ogólności i często również mapy drogowej produktu.

### Wizja

**Wizja produktu** dostarcza jasnego opisu obszaru, w którym interesariusze, tacy jak użytkownicy lub klienci, otrzymują wartość. W naszym przypadku użytkownikami było 10 tysięcy członków Scrum Alliance na całym świecie w owym czasie (pod koniec roku 2011 liczba ta wzrosła do 150 tysięcy). Klientem płacącym za nowy produkt była rada dyrektorów Scrum Alliance działająca z ramienia członków.

Nasza wizja nowej witryny Scrum Alliance przedstawała się następująco:

*Nowa witryna Scrum Alliance będzie zaufanym źródłem wiedzy na temat Scruma dla ludzi na całym świecie zainteresowanych tą tematyką. Witryna będzie bogata w treść i cechy — stanie się pierwszym przystankiem w Internecie pozwalającym zdobyć więcej wiedzy na temat Scruma lub podjąć współpracę w zakresie własnych zainteresowań związanych ze Scrumem.*

### Rejestr produktu wysokiego poziomu

Po ustaleniu wizji produktu następnym krokiem jest wygenerowanie początkowej wersji rejestru produktu na bardzo ogólnym poziomie. W przypadku przeprojektowanej witryny Scrum Alliance w roku 2006 mieliśmy już rosnący rejestr elementów produktu dla cech, które interesują i użytkownicy chcieli widzieć w nowej, poprawionej stronie internetowej.

Wśród elementów rejestru produktu znajdowały się następujące eposy:

*Jako certyfikowany trener Scruma chciałbym mieć możliwość umieszczania informacji na temat moich publicznych wykładów dotyczących Scruma na stronie internetowej Scrum Alliance, dzięki czemu społeczność będzie mogła poznać czas i miejsce oferowanych przez mnie zajęć.*

*Jako potencjalny student chciałbym mieć możliwość przeglądania dostępnych publicznie zajęć dotyczących Scruma, dzięki czemu mógłbym znaleźć zajęcia odpowiadające moim kryteriom uczestnictwa.*

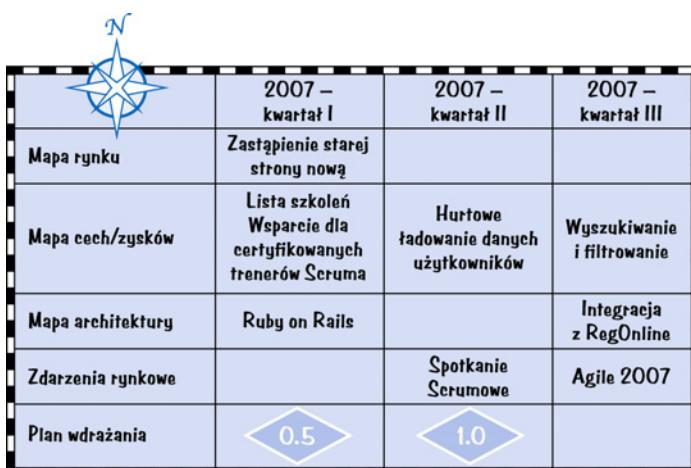
Gdyby nasz produkt był zupełnie nowy, musielibyśmy rozpoczęć od wygenerowania minimalnego zestawu wymagań, które zapełniłyby rejestr produktu, i przypisać oceny przynajmniej najbardziej ogólnym elementom tego rejestru. W naszym przypadku mieliśmy już jakieś elementy w rejestrze, które mogliśmy przyjąć za punkt startowy dla idei, które miały zostać włączone do wizji nowej strony internetowej.

## Mapa drogowa produktu

Po stworzeniu wizji produktu i rejestrze produktu wysokiego poziomu pomocne jest zbudowanie mapy drogowej produktu (określonej czasem mianem mapy drogowej wersji dystrybucyjnej). **Mapa drogowa produktu** dostarcza informacji na temat przyrostowego sposobu budowania i dostarczania produktu w czasie uwzględniających istotne czynniki mające wpływ na poszczególną wersję.

Dzisiaj wiele organizacji stara się osiągnąć **ciągłą dystrybucję**, czyli wdrażanie działających cech, kiedy tylko jest to możliwe. Jeżeli Twoja organizacja skupia się na takim działaniu, być może nie musisz tworzyć mapy produktu. Niemniej jednak nawet jeśli starasz się wdrażać produkt w sposób nieustający, jego mapa drogowa może okazać się użytecznym narzędziem pozwalającym Twojej organizacji myśleć o większych zbiorach cech, zależnościach, które mogą dyktować konieczność realizowania pewnych cech mniej więcej w tym samym czasie, a także o tym, kiedy dane cechy powinny stać się dostępne.

Rysunek 15.2 pokazuje mapę drogową produktu w formie promowanej przez Luke'a Hohmanna [Hohmann, 2003].



**RYSUNEK 15.2.** Mapa drogowa dla produktu strony internetowej Scrum Alliance

Na mapie tej widoczne są dwie wersje dystrybucyjne — po jednej w każdym z pierwszych dwóch kwartałów roku 2007. Wersja „0.5” w pierwszym kwartale 2007 była pierwszą wersją naszej nowej witryny internetowej. Wybraliśmy taki numer, ponieważ planowaliśmy, że w tej pierwszej wersji znajdzie się mniej niż połowa cech starej witryny Scrum Alliance, ale będzie ona zawierała nowe cechy, lepsze od swoich odpowiedników w wersji starej. Oczekiwane cechy skupione były wokół publikowania wszelkich dostępnych publicznie szkoleń poświęconych Scrumowi na całym świecie, a także wokół podstawowego wsparcia dla certyfikowanych trenerów Scruma. Wersja 0.5 była wersją z ustalonąm zakresem, ponieważ wiedzieliśmy, jakie konkretnie cechy chcemy posiadać w nowej wersji, zanim będziemy gotowi pozbyć się wersji starej. Nie wiedzieliśmy natomiast, ile czasu zajmie nam przygotowanie tych nowych cech. O tym, jak ustalić datę wdrożenia dla wersji z ustalonąm zakresem, napiszę w rozdziale 18.

Wersja 1.0 była wersją z ustaloną datą dystrybucji. Wiedzieliśmy, że chcemy, aby wypuszczenie tej wersji zbiegło się w czasie z konferencją Scrum Alliance (zwaną Spotkaniem Scrumowym), która rozpoczęła się 7 maja 2007 roku w Portland w stanie Oregon. Naszym celem było posiadanie zestawu atrakcyjnych cech dostępnych w pierwszym dniu tej konferencji. Nie wiedzieliśmy jedynie, ile cech uda nam się umieścić w tej wersji dystrybucyjnej. Problemem określania zawartości wersji o ustalonej dacie dystrybucji zajmiemy się w rozdziale 18.

Podsumowując, na początkowej wersji mapy drogowej dla strony internetowej Scrum Alliance umieściliśmy zarówno wersję z ustalonąm zakresem (0.5), jak i wersję z ustaloną datą (1.0).

Niezależnie od tego, jaki produkt wytwarzasz, pod koniec planowania na poziomie produktu powinieneś mieć wizję produktu, rejestr produktu wysokiego poziomu wypełniony ocenionymi historyjkami użytkownika i (opcjonalnie) mapę drogową produktu. Możesz również stworzyć inne artefakty dające decydentom większą pewność przy podejmowaniu decyzji o rozpoczęciu produkcji.

Wyniki planowania na poziomie produktu stają się danymi wejściowymi do planowania portfela. W trakcie tego spotkania rada dyrektorów zatwierdziła początkową wersję 0.5 przeredagowanej strony internetowej.

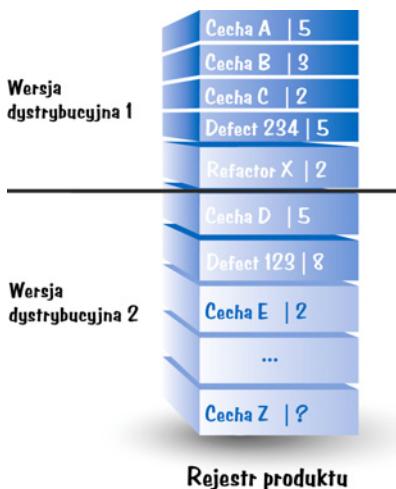
## Planowanie wersji dystrybucyjnej

**Planowanie wersji dystrybucyjnej** to szukanie kompromisu pomiędzy zakresem, czasem i budżetem przyrostowych wersji produktu.

Dla większości projektów deweloperskich wskazane jest przeprowadzenie początkowego planowania wersji dystrybucyjnej po planowaniu produktu, ale przed rozpoczęciem pierwszego sprintu związanego z tą wersją dystrybucyjną. W tym momencie możesz stworzyć początkowy **plan wersji dystrybucyjnej**, który zbilansuje to, ile rzeczy będziesz w stanie stworzyć w tej wersji, z czasem, kiedy wersja będzie dostępna.

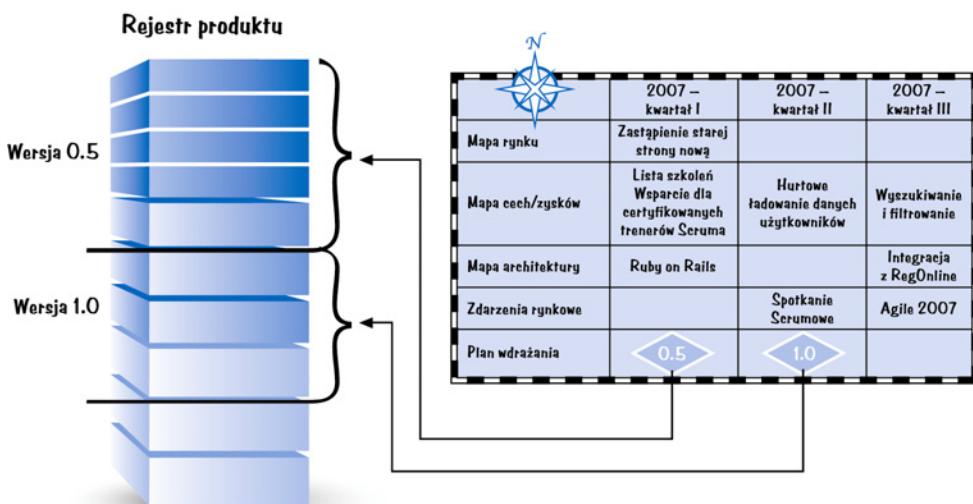
Chcąc zyskać jakiekolwiek pojęcie na temat cech możliwych do dostarczenia w ustalonym czasie lub czasu potrzebnego do dostarczenia określonego zestawu cech, musisz stworzyć i nadać oceny odpowiedniej liczbie elementów rejestru produktu.

Prostą metodą wizualizacji wersji dystrybucyjnej jest narysowanie linii biegnącej w poprzek rejestru produktu (patrz rysunek 15.3). Wszystkie elementy powyżej tej linii przeznaczone są do danej wersji dystrybucyjnej. W miarę zdobywania coraz lepszej wiedzy na temat produktu linia może przesuwać się w górę lub w dół. O tym, gdzie początkowo należy umieścić tę linię, będę pisał w rozdziale 18.



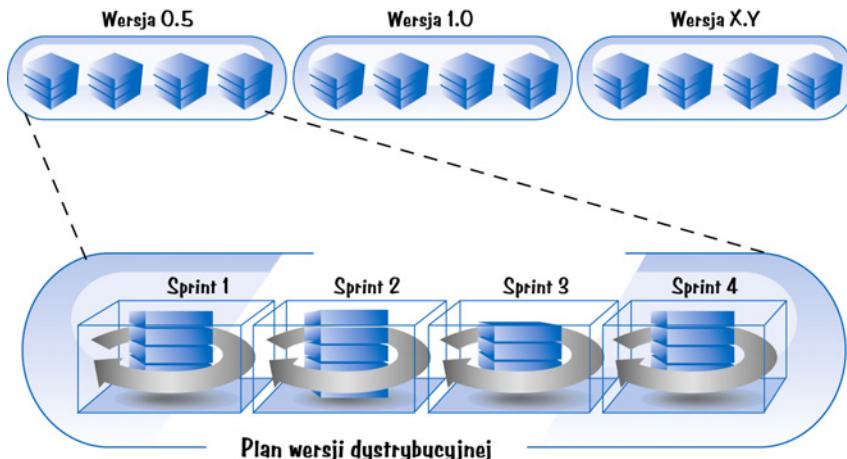
**RYSUNEK 15.3.** Linia wersji dystrybucyjnej w rejestrze produktu

Teraz możesz w łatwy sposób powiązać mapę drogową produktu z rejestrzem produktu i bardziej dokładnie opracować zawartość przynajmniej najbliższych wersji dystrybucyjnych wskazanych na mapie (patrz rysunek 15.4). Wersja dystrybucyjna na mapie drogowej produktu odpowiada zestawowi cech w rejestrze produktu.



**RYSUNEK 15.4.** Odwzorowanie mapy drogowej produktu na rejestr produktu

Plan wersji dystrybucyjnej musi mieć powiązany ze sobą wymiar czasowy, który może być wyrażony w formie liczby sprintów potrzebnych do ukończenia wersji. Większość wersji dystrybucyjnych posiada liczbę cech niemożliwą do zrealizowania w czasie pojedynczego sprintu (patrz rysunek 15.5).



**RYSUNEK 15.5.** Wersja dystrybucyjna może obejmować jeden lub więcej sprintów

Podczas planowania wersji dystrybucyjnej możesz zaryzykować próbę zgadnięcia cech, które zostaną dostarczone podczas kilku pierwszych sprintów. Będzie to pomocne w przypadku konieczności koordynowania prac wielu zespołów lub jeśli nasz zespół musi wcześniej zgłosić konieczność otrzymania odpowiedniego sprzętu, narzędzi lub pomocy osób trzecich. Przewidywanie zawartości więcej niż kilku sprintów w przód jest niewskazane i narusza zasadę planowania w samą porę i w stopniu wystarczającym do płynnego przepływu pracy.

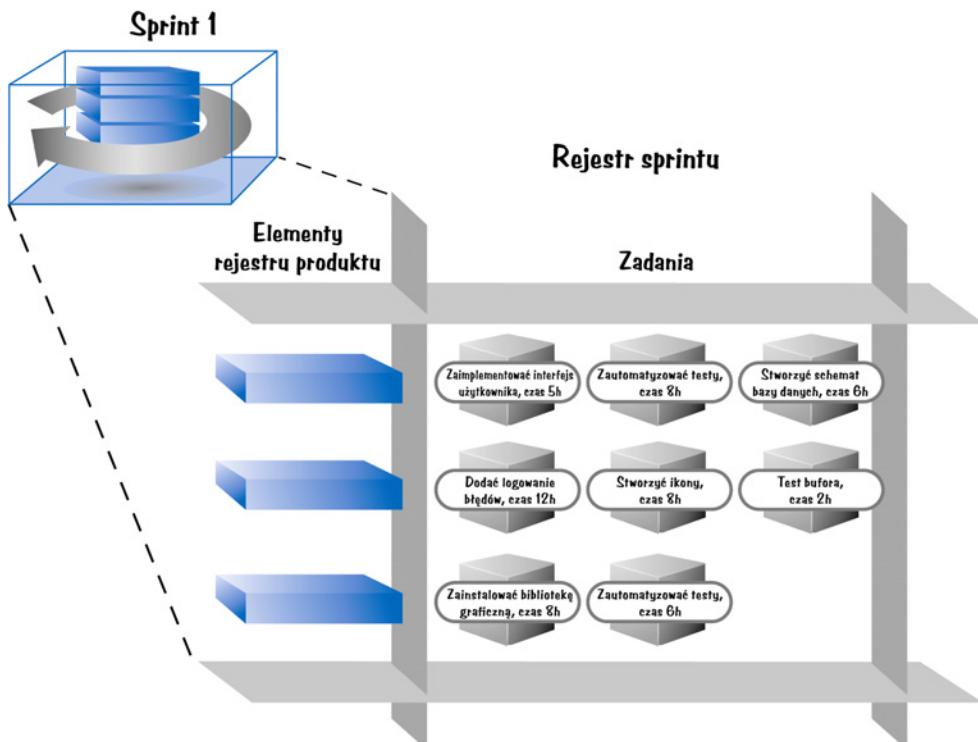
## Planowanie sprintu

Elementy rejestru produktu, nad którymi zespół scrumowy będzie pracował w następnym sprintie, są uzgadniane podczas planowania sprintu (mającego miejsce na początku każdego sprintu). W trakcie tej aktywności zespół tworzy rejestr sprintu, czyli opis pracy w formie zadań, które muszą zostać zrealizowane, aby uznać element rejestru za ukończony (patrz rysunek 15.6).

Podczas planowania sprintu zespół wykonuje kolejny stopień szczegółowego planowania w samą porę. Na temat planowania sprintu będę pisał dokładniej w rozdziale 19.

## Planowanie codzienne

Najbardziej szczegółowy rodzaj planowania ma miejsce w trakcie codziennych działań scrumowych. Jak zapewne sobie przypominasz, jest to aktywność, w trakcie której członkowie zespołu spotykają się ze sobą i każdy z nich kolejno opowiada, co udało mu się zrobić od poprzedniego zebrania, nad czym zamierza pracować w dniu dzisiejszym i czy widzi jakieś przeszkody w realizacji swoich zadań.

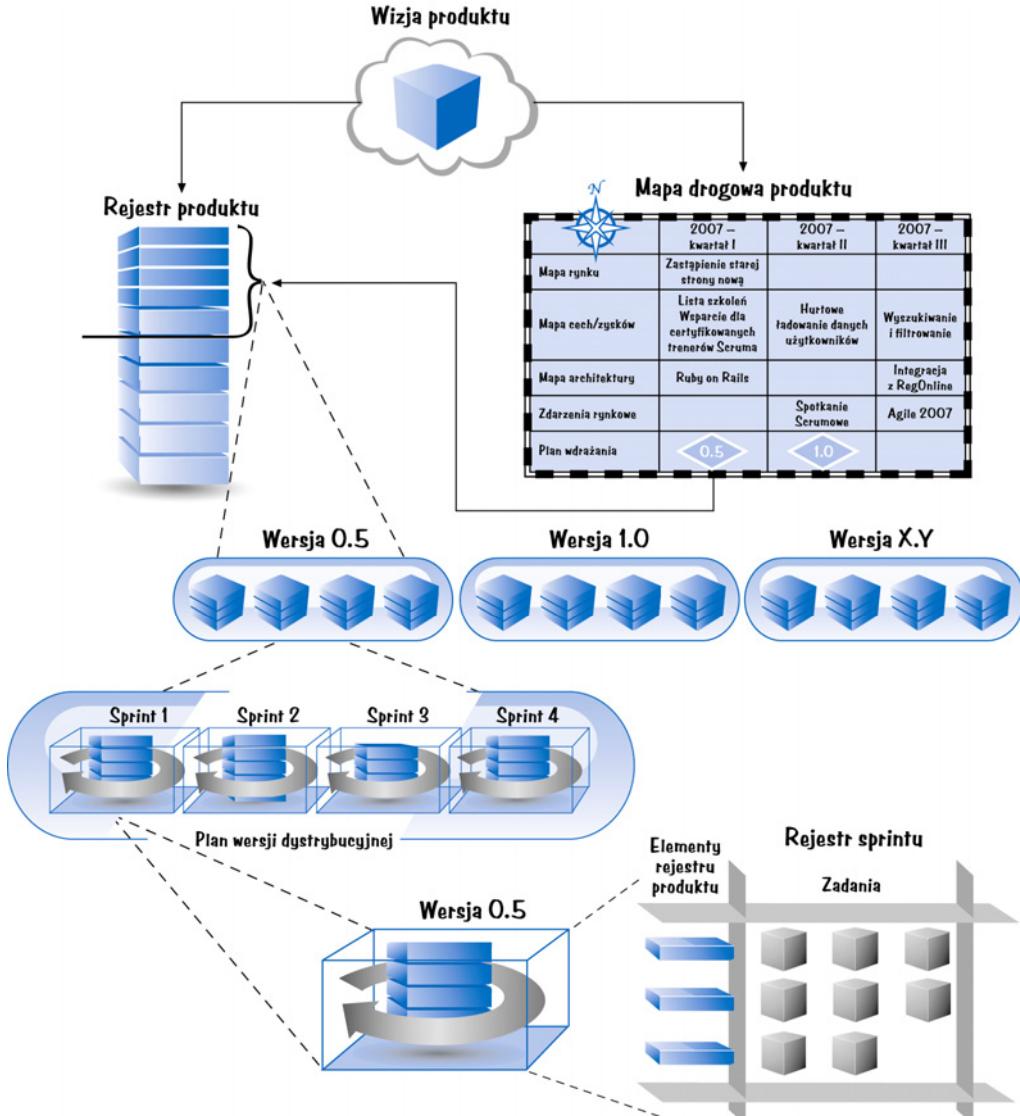


**RYSUNEK 15.6.** Każdy sprint posiada swój rejestr

Podczas codziennych działań scrumowych członkowie zespołu wspólnie opisują w łatwy do zrozumienia sposób ogólny plan tego dnia. To pozwala na przekazywanie istotnych informacji (alarmów) odnośnie do potrzebnych zasobów. Na przykład ktoś może powiedzieć: „Dzisiaj zamierzam pracować nad zadaniem procedury składowanej. Powiniensem być gotowy przed obiadem. Ktakolwiek ma przypisane zadanie dopisania logiki biznesowej, proszę, żeby pamiętała, że ta rzecz ma krytyczne znaczenie dla ukończenia naszej pracy w tym sprintie i w związku z tym praca nad nią powinna zacząć się zaraz po obiedzie”. Taki komunikat pozwala szybko zidentyfikować potencjalne blokady w pracy i zapewnić lepszy przepływ w trakcie wykonania sprintu.

## Zakończenie

Ten rozdział zilustrował przebieg planowania na różnych poziomach szczegółowości w projekcie realizowanym z użyciem Scruma. Rysunek 15.7 pokazuje w sposób graficzny artefakty wytwarzane na tych poziomach (z wyłączeniem poziomów portfela i planowania codziennego) oraz ich wzajemne powiązania.



**RYSUNEK 15.7.** Hierarchiczne planowanie w Scrumie

W kolejnych rozdziałach zajmę się bardziej szczegółowo tematyką planowania portfela, produktu, wersji dystrybucyjnej i sprintu.



## Rozdział 16

# PLANOWANIE PORTFELA

---

Większość organizacji chce lub musi produkować więcej niż jeden produkt jednocześnie. Potrzebują one sposobu na podejmowanie ekonomicznie uzasadnionych decyzji podczas zarządzania swoimi portfelami produktów. Sam proces zarządzania lub też nadzoru nad portfelem powinien być dodatkowo zgodny z podstawowymi zasadami zwinności, w przeciwnym wypadku ujawni się całkowity brak zadowolenia z zastosowania podejścia zwinnego na poziomie indywidualnych produktów. Ten rozdział przedstawia 11 strategii planowania portfela, pogrupowanych według podejścia do tworzenia harmonogramu, napływu i odpływu produktów. Na koniec omówię kwestię inwestowania dodatkowej pracy w produkty już przetwarzane (aktywne).

## Wprowadzenie

**Planowanie portfela** (lub też zarządzanie portfelem) to aktywność mająca na celu wskazanie, nad którymi elementami rejestru portfela należy pracować, w jakiej kolejności i jak długo. Elementem rejestru portfela może być produkt, przyrost produktu (pojedyncza wersja dystrybucyjna produktu) lub projekt (jeżeli organizacja preferuje planowanie pracy wokół projektów). Używane w tym rozdziale słowo *produkt* odnosi się do wszystkich typów elementów z rejestru portfela.

Z własnego doświadczenia mogę powiedzieć, że większość organizacji (zwinnych lub innych) bardzo kiepsko radzi sobie z planowaniem na poziomie portfela. W wielu z nich proces planowania na tym poziomie stoi w zupełnej sprzeczności z podstawowymi zasadami zwinności. Kiedy taka sytuacja ma miejsce, decyzje podejmowane na poziomie portfela powodują przerwanie szybkiego i płynnego potoku pracy. W tym rozdziale wyjaśnię, jak uniknąć takiego nieporozumienia poprzez planowanie portfela w zgodzie z zasadami zwinności.

## Czas

Planowanie portfela jest nigdy niekończącą się aktywnością. Jak długo istnieją produkty przeznaczone do rozwijania lub utrzymywania, musimy zarządzać portfelem.

Jak widać na rysunku 15.1, planowanie portfela ma związek z kolekcją produktów i w związku z tym stoi na wyższym poziomie hierarchii niż planowanie (tworzenie wizji) pojedynczego produktu. Bycie na wyższym poziomie nie oznacza wcale, że planowanie portfela poprzedza planowanie produktu. Przeciwnie — wyniki planowania lub też tworzenia wizji produktu są ważnymi danymi wejściowymi do planowania portfela. Podczas planowania portfela dane te służą do podjęcia decyzji o tym, czy finansować produkt i w jakiej kolejności umieścić go w **rejestrze portfela**. Nie oznacza to jednak wcale, że planowanie portfela jest przeznaczone wyłącznie dla nowych wizji produktów. Przeprowadzane jest ono w regularnych odstępach czasu w celu przejrzenia produktów już przetwarzanych (będących w produkcji, użytkowanych lub aktualnie sprzedawanych).

## Uczestnicy

Ponieważ planowanie portfela skupia się zarówno na nowych produktach, jak i **produkach aktywnych**, jego uczestnikami powinni być wewnętrzni interesariusze, właściciele poszczególnych produktów i opcjonalnie, chociaż często, starsi architekci i liderzy techniczni.

Interesariusze muszą posiadać dostatecznie szeroką perspektywę biznesową, aby nadawać prawidłowe priorytety elementom rejestru portfela i podejmować decyzje odnośnie do aktywnych produktów. W niektórych organizacjach interesariusze formują wspólnie komitet zatwierdzający lub też radę nadzorczą, ewentualnie inne podobne ciało, które ma za zadanie nadzorować proces planowania portfela.

W planowaniu portfela uczestniczą również właściciele produktów, jako ich przywódcy i adwokaci odpowiedzialni za niezbędne zasoby.

Aby zapewnić uwzględnienie ważnych zależności technicznych podczas podejmowania decyzji związanych z zarządzaniem portfelem, często w spotkaniu biorą udział starsi architekci i liderzy techniczni.

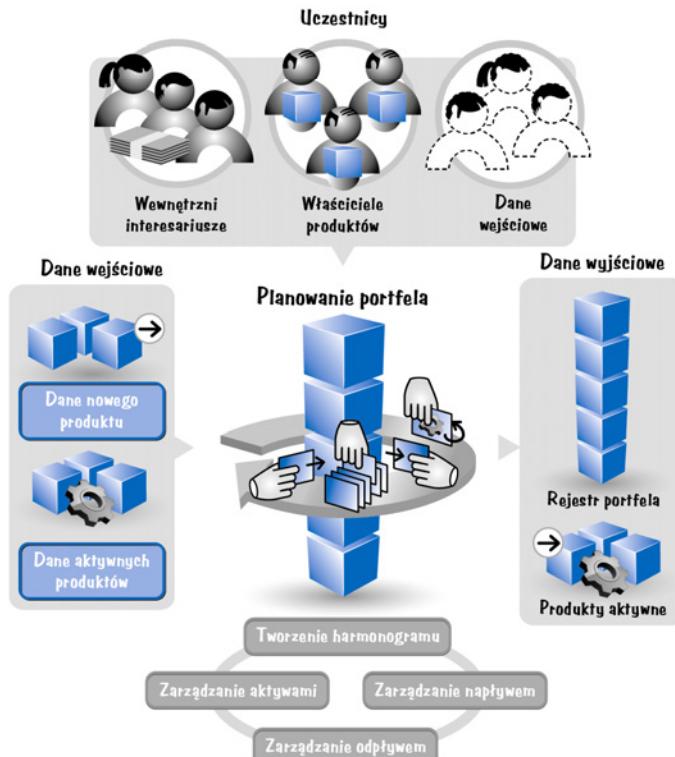
## Proces

Przebieg aktywności planowania portfela ilustruje rysunek 16.1.

Jak napisałem wcześniej, dane wejściowe do planowania portfela zawierają nowe wizje produktów (kandydatów dołączenia do rejestru portfela) i produkty aktywne. Nowe produkty „przybywają” z informacjami zebranymi podczas tworzenia ich wizji — są to między innymi: koszt, czas trwania produkcji, wartość, ryzyko. Produkty aktywne „przybywają” ze swoim zestawami danych — informacjami zwrotnymi od klientów, uaktualnionymi kosztami i harmonogramami prac, a także ocenami zakresu, poziomami dłużu technicznego i danymi rynkowymi, które pomagają określić ich dalszą ścieżkę.

Wynikiem planowania portfela są dwie rzeczy. Pierwsza to rejestr portfela będący posortowaną według priorytetów listą przyszłych projektów — zatwierdzonych, ale jeszcze nieprzekazanych do produkcji. Druga rzecz to zbiór aktywnych produktów — nowych zatwierdzonych do bezzwłocznej produkcji, a także produktów będących już w produkcji, którym dano zielone światło do kontynuacji.

Otrzymanie tych danych wyjściowych wymaga zaangażowania uczestników spotkania w cztery formy aktywności: tworzenie harmonogramu, zarządzanie napływem i odpływem oraz zarządzanie produktami aktywnymi. Strategie związane z powyższymi kategoriami przedstawia rysunek 16.2.



**RYSUNEK 16.1.** Aktywność planowania portfela

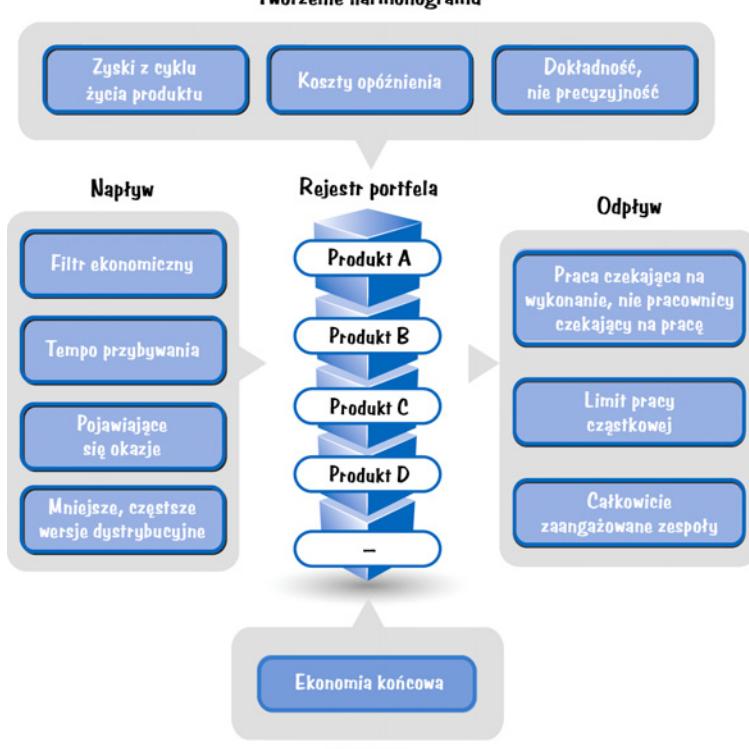
Strategie tworzenia harmonogramu pomagają w określeniu właściwego porządku produktów w rejestrze portfela. Strategie napływu dają uczestnikom wskazówki na temat tego, kiedy umieścić elementy w rejestrze portfela. Strategie odpływu informują uczestników o konieczności usunięcia produktu z rejestru. Strategia produktów aktywnych służy do podjęcia decyzji o zachowaniu, dokonaniu zwrotu, dostarczeniu lub zakończeniu produktu, który aktualnie znajduje się w stanie aktywnym.

Pozostała część tego rozdziału omawiać będzie każdą z 11 strategii tworzących wymienione cztery kategorie.

## Strategie tworzenia harmonogramu

Planowanie portfela musi skutkować przypisaniem ograniczonych zasobów firmy do produktów z zachowaniem zasad ekonomii. Chociaż istnieje wiele sposobów decydowania o kolejności produktów, ja skupiam się na trzech strategiach:

- optymalizacją pod względem zysków z cyklu życia produktu;
- wliczaniem kosztów opóźnienia;
- estymacją z zachowaniem dokładności, nie precyzji.



**RYSUNEK 16.2.** Strategie planowania portfela

## Optymalizacja pod względem zysków z cyklu życia

Aby określić pozycję produktu w portfelu, musimy zdecydować, którą zmienną chcemy mierzyć w celu upewnienia się, że nasze wskaźniki optymalizacyjne odnoszą skutek. Reinertsen rekomenduje używanie środowiska ekonomicznego, w którym wszystkie decyzje i kompromisy mierzone są standardową i użyteczną jednostką miary: **zyskami z cyklu życia** [Reinertsen, 2009b]. Bazując na tej rekomendacji, naszym celem powinna być sekwencja elementów rejestru produktu maksymalizująca całkowite zyski z cyklu życia.

Dla danego produktu zyski z cyklu życia to całkowity potencjał zysków, jakie może przynieść ten produkt w całym cyklu swego życia. W przypadku planowania portfela interesuje nas optymalizacja zysków z cyklu życia całego portfela, a nie pojedynczego produktu. W związku z tym będziemy musieli przeprowadzić optymalizację indywidualnych produktów [Poppdieck i Poppdieck, 2003]. Stąd celem strategii optymalizowania pod względem zysków z cyklu życia jest znalezienie takiej sekwencji elementów rejestru produktu, która zapewnia największe zyski z cyklu życia w całym portfelu (porównaj to dla przykładu z obliczaniem kosztów opóźnienia przedstawionym w kolejnej sekcji).

Reinertsen dowodzi również, że dwiema najważniejszymi zmiennymi potrzebnymi do oceny zysków z cyklu życia są koszt opóźnienia i czas trwania (którego dobrymi reprezentantami są wysiłek lub rozmiar produktu). Sugeruje on wybranie jednego z trzech podejść do harmonogramu przedstawionych w tabeli 16.1 w oparciu o to, jak blisko siebie znajdują się (lub też nie) te dwie zmienne dla wszystkich produktów w portfelu.

**TABELA 16.1.** Różne zasady układania harmonogramu dla portfela

(Jeżeli) koszt opóźnienia	(i) czas trwania/rozmiar	(wtedy) podejście do harmonogramu
Taki sam dla wszystkich produktów	Zróżnicowany w zależności od produktu	W pierwszej kolejności praca najkrótsza
Zróżnicowany w zależności od produktu	Taki sam dla wszystkich produktów	W pierwszej kolejności zadanie o największym koszcie opóźnienia
Zróżnicowany w zależności od produktu	Zróżnicowany w zależności od produktu	Ważona najkrótsza praca w pierwszej kolejności

Kiedy wszystkie produkty mają taki sam koszt opóźnienia, preferowaną strategią jest wykonanie w pierwszej kolejności najkrótszej pracy. Jeżeli wszystkie produkty mają taki sam rozmiar (czas trwania), zalecaną strategią jest praca w pierwszej kolejności nad produktami o wysokim koszcie opóźnienia. Gdy zarówno koszt opóźnienia, jak i czas trwania są zmienne (co jest rzeczą normalną w produkcji oprogramowania), optymalnym z punktu widzenia ekonomii podejściem do harmonogramu jest **ważona najkrótsza praca w pierwszej kolejności** — wyliczana jako koszt opóźnienia dzielony przez czas trwania (lub wysiłek niezbędny do zaimplementowania).

Dalej omówię koszt opóźnienia oraz prognozowanie wysiłku (kosztu) produktów w portfelu.

## Wyznaczanie kosztu opóźnienia

Po uporządkowaniu elementów w rejestrze portfela zachodzi konieczność pracowania nad pewnymi produktami, zanim będziemy pracować nad innymi. Te produkty, nad którymi nie rozpoczęliśmy pracy, natychmiast będą miały opóźniony start i w związku z tym opóźnioną datę dostarczenia, z czym wiąże się mierzalny koszt.

Jak napisałem w rozdziale 3., koszt opóźnienia jest źródłem istotnych informacji służących do podejmowania świadomych decyzji ekonomicznych. Niemniej jednak większość organizacji nie jest nawet w stanie odpowiedzieć na tak proste pytanie jak „Jaki będzie koszt opóźnienia w zyskach z cyklu życia produktu, jeśli opóźnimy jego wdrożenie o jeden miesiąc?“.

Ignorowanie kosztu opóźnienia sprawia, że większość organizacji decyduje się ułożyć swój portfel według prostej (i często błędnej) zasady „najwyższych zysków w pierwszej kolejności“ (patrz tabela 16.2).

**TABELA 16.2.** Przykład użycia kosztu opóźnienia do określenia kolejności projektów w portfelu

Projekt A	Projekt B
Zwrot z inwestycji	20%
Koszt opóźnienia (1 miesiąc)	5 tysięcy zł
	75 tysięcy zł

W tym przykładzie projekt A ma stopę zwrotu z inwestycji na poziomie 20%, a projekt B tę samą stopę na poziomie 15%. Stosując strategię najwyższych zysków, wykonujemy projekt A przed projektem B, ponieważ pierwszy z nich ma wyższą stopę zwrotu. Chociaż takie podejście wydaje się być rozsądne, nie bierzemy pod uwagę kosztu opóźnienia każdego z produktów, który może mieć znaczący wpływ na wyliczenia zysków z cyklu życia produktu. Co na przykład, jeśli projekt A ma koszt opóźnienia rzędu 5 tysięcy złotych na miesiąc, a projekt B 75 tysięcy złotych na miesiąc (zgodnie z tabelą 16.2)? W takiej sytuacji odraczanie prac nad projektem B, aby pracować nad projektem A, ma o wiele większy wpływ na ogólny zysk z cyklu życia w całym portfelu.

Koszt opóźnienia uzmysławia fakt, iż czas wpływa lub może wpływać na wszystkie zmienne. W powyższym przykładzie zwroty z inwestycji dla projektu A i B zostały policzone w oparciu o pewne założenia bazujące na czasie (na przykład na tym, kiedy rozpoczną się i zakończą prace deweloperskie, jakie zasoby będą dostępne w tym czasie, ile będą one kosztować, jaką cenę ludzie będą skłonni zapłacić za produkt wraz z upływem czasu, jakiego rodzaju ryzyka biznesowe i technologiczne będą mogły mieć miejsce, jakie będzie ich prawdopodobieństwo i związane z nimi dodatkowe koszty). Wystarczy opóźnić lub przyspieszyć prace deweloperskie, a wartości tych zmiennych najprawdopodobniej ulegną zmianie. Zatem koszt opóźnienia nie jest jedynym czynnikiem wartym rozważenia podczas ustalania priorytetów elementów w rejestrze portfela. Pod uwagę trzeba również wziąć wymiar czasu, ponieważ ma on wpływ na wszystkie pozostałe zmienne priorytetów, takie jak koszt, zyski, wiedza i ryzyko.

Najczęściej słyszane przeze mnie zarzuty pod adresem kosztu opóźnienia dotyczą braku jasności co do sposobu obliczania jego wartości. W większości przypadków są one nieuzasadnione, ponieważ do wyliczenia kosztu opóźnienia wystarczą dwa różne modele arkuszy kalkulacyjnych wyliczających zyskowność (jeden z uwzględnieniem kosztu opóźnienia i drugi bez).

Leffingwell proponuje model wyliczania kosztu opóźnienia będący sumą trzech atrybutów produktu [Leffingwell, 2011]:

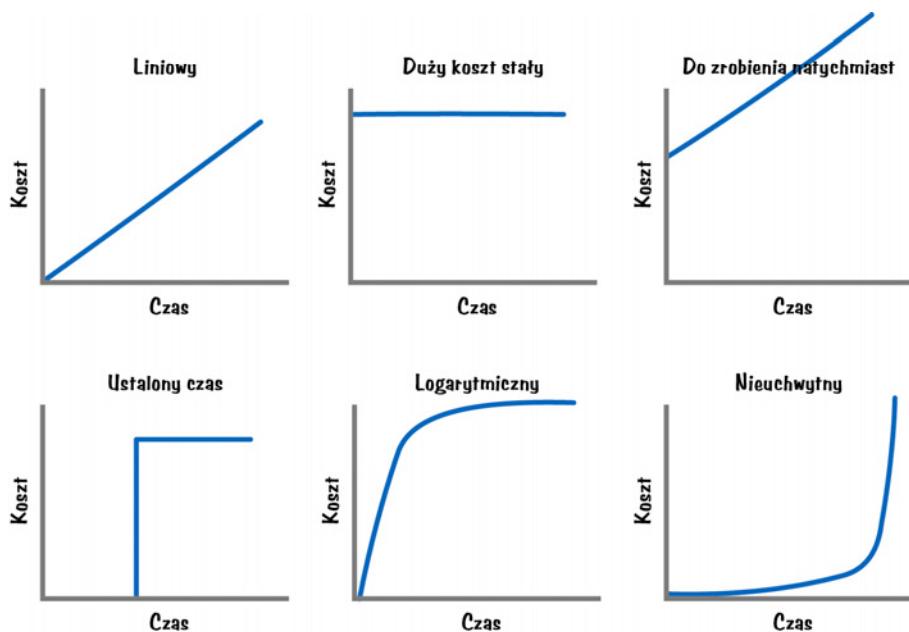
- wartości dla użytkownika — potencjalnej wartości widzianej oczami użytkownika;
- wartości czasu — spadku wartości dla użytkownika w miarę upływu czasu;
- redukcji ryzyka i (lub) umożliwiania okazji — wartości w kontekście łagodzenia ryzyka lub eksploracji pewnej okazji.

Aby obliczyć koszt opóźnienia dla produktu, każdemu z tych trzech atrybutów przypisywana jest jego własna wartość kosztu opóźnienia w skali od 1 (najniższy koszt) do 10 (najwyższy koszt). Całkowity koszt opóźnienia produktu jest sumą tych poszczególnych kosztów opóźnienia.

Innym często skutecznym podejściem do podejmowania świadomych decyzji odnośnie do harmonogramu jest charakteryzowanie ogólnego profilu kosztu opóźnienia (patrz rysunek 16.3).

Każdy z profili na rysunku 16.3 jest opisany bardziej szczegółowo w tabeli 16.3.

Jeżeli precyzyjne wyliczenie kosztu opóźnienia jest zbyt czasochłonne lub podatne na błędy, rozważ wybór odpowiedniego profilu opóźnienia (lub stworzenie nowego) i użyj go zamiast konkretnej wartości podczas podejmowania decyzji dotyczących harmonogramu prac.



RYSUNEK 16.3. Profile kosztów opóźnienia

TABELA 16.3. Opis profili czasu opóźnienia

Nazwa profilu	Opis
Liniowy	Produkt, którego koszt opóźnienia rośnie w stałym tempie.
Duży koszt stały	Produkt, który w przypadku niepodjęcia bezwzględnego działań akumuluje jednorazowo stały koszt opóźnienia. Przykładem może być produkt, za który otrzymamy konkretną zapłatę dopiero po jego dostarczeniu.
Do zrobienia natychmiast	Produkt, który „musimy zrobić teraz”, ponieważ już teraz doświadczamy agresywnie rosnącego kosztu opóźnienia. Przykładem może być produkt, który brak powoduje utratę zysków lub oszczędności w tempie postępującym wraz z upływem czasu.
Ustalony czas	Produkt, który musi być dostarczony w ścisłe określonym dniu w przeszłości i w związku z tym aż do tego dnia ma zerowy koszt opóźnienia. Po przekroczeniu tego terminu doświadczamy pełnego kosztu przynajmniej początkowego opóźnienia.
Logarytmiczny	Produkt, który bardzo wcześnie generuje większość kosztu opóźnienia, a jego dalszy wzrost nie jest aż tak agresywny.
Nieuchwytny	Produkt (lub nakład pracy), który przez większość czasu „najwyraźniej” nie posiada kosztu opóźnienia, a następnie nieoczekiwane nabieranie go w bardzo dużej ilości. Przykładem może być sposób traktowania dłużu technicznego przez większość organizacji. Dzisiaj może się wydawać, że koszt opóźnienia wynikający z niespłacania dłużu technicznego jest niewielki lub wręcz zerowy. Niemniej jednak, jak opisałem to w rozdziale 8., dług techniczny może osiągnąć punkt przegięcia, po którym przekroczeniu koszt opóźnienia pozostały pracy staje się zauważalny i jest bardzo duży.

Czy koszt opóźnienia dotyczy organizacji, które wytwarzają produkty w gałęziach przemysłu podlegających ścisłym regulacjom, takich jak produkcja urządzeń medycznych lub opieka medyczna, czyli tam, gdzie zachowanie zgodności z wymogami oraz bezpieczeństwo pacjentów mają fundamentalne znaczenie? Te absolutnie krytyczne czynniki muszą być brane pod uwagę podczas ustalania priorytetów, chociaż ich specyfika może mieć wpływ na koszt opóźnienia wyrażony w formie zysków z cyklu życia.

Na przykład w Stanach Zjednoczonych organizacje zajmujące się finansowaniem ochrony zdrowia muszą używać we wnioskach, kartach medycznych oraz innych transakcjach elektronicznych określonych kodów identyfikujących diagnozy oraz procedury medyczne. W chwili pisania tej książki obowiązujący standard tych kodów to *Międzynarodowa klasyfikacja chorób* (ang. *International Classification of Diseases*) w wersji dziewiątej (ICD-9-CM). Niemniej jednak 1 października 2013 roku ICD-9-CM zostanie zastąpiony przez nowy standard ICD-10-CM. W tym czasie organizacje podlegające amerykańskiej ustawie HIPPA (ang. *Health Insurance Portability and Accountability Act*) z roku 1996 będą musiały zacząć działać w zgodzie z ICD-10-CM. Wiele z nich dysponuje portfelem produktów, które będą wymagały modernizacji — w stylu uderzająco podobnym do problemu roku dwutysięcznego (Y2K) na przełomie wieków. Ponieważ wszystkie produkty w portfelu remontowym mają profil kosztu opóźnienia o ustalonym czasie (widoczny na rysunku 16.3), racjonalne ułożenie harmonogramu prac będzie wymagało od tych organizacji określenia wysokości kosztu opóźnienia dla każdego z produktów w sytuacji, gdyby *nie* zostały one zmodernizowane przed pierwszym dniem października 2013. Powiedzmy, że produkt o zupełnie kluczowym znaczeniu mógłby wygenerować stratę rzędu 100 milionów złotych na rok, podczas gdy inny niezgodny produkt wygenerowałby jedynie 5 milionów straty na rok. Stąd widać, że wyliczony koszt opóźnienia jest kluczową zmienną mającą wpływ na ekonomiczne uporządkowanie portfela remontowego.

## **Staraj się obliczać dokładnie, ale nie precyzyjnie**

W celu dobrego ułożenia elementów w rejestrze portfela potrzebujemy również rozumieć związek z nimi wysiłek czy koszt (ponieważ koszt ma wpływ na zyski z cyklu życia). Ze względu na bardzo ograniczone dane w chwili, kiedy potrzebujemy ocen po raz pierwszy, określając ich rozmiar, staramy się osiągnąć dokładność, ale nie precyzję.

W rozdziale 7. napisałem, że niektóre organizacje preferują ocenianie elementów rejestru portfela przy użyciu rozmiarów ubraniowych zamiast przesadnie precyzyjnych liczb. Każdy taki rozmiar odpowiada pewnemu zakresowi kosztów (przykład odwzorowania rozmiarów na przedziały liczbowe kosztów pokazuje tabela 16.4).

**TABELA 16.4.** Przykład ocen z użyciem rozmiarów ubraniowych

Rozmiar	Przybliżony przedział kosztów
Bardzo mały (XS)	od 10 tysięcy do 25 tysięcy złotych
Mały (S)	od 25 tysięcy do 50 tysięcy złotych
Średni (M)	od 50 tysięcy do 125 tysięcy złotych
Duży (L)	od 125 tysięcy do 350 tysięcy złotych
Bardzo duży (XL)	powyżej 350 tysięcy złotych

W tabeli tej przybliżony przedział kosztów uwzględnia koszt pracy (będący na ogół większościowym kosztem produktu w organizacji), jak również wydatki kapitałowe i inne koszty materiałowe związane z wysiłkiem deweloperskim.

Zaletami oceniania przy użyciu rozmiarów ubraniowych są szybkość, wystarczająca dokładność oraz dostarczanie informacji na poziomie portfela umożliwiających podjęcie konkretnych działań.

Jaka dokładność jest wystarczająca? Pozwól sobie odpowiedzieć przykładem. We wspomnianej wcześniej organizacji dział inżynierijny spędzał dawniej sporo czasu, starając się nadawać bardzo precyzyjne oceny. Pracownicy tego działu nie byli przekonani, czy rozmiary ubraniowe będą odpowiednio dokładne, ale wszyscy zgodzili się spróbować. W niedługim czasie dział marketingowy zgłosił pomysł nowego projektu. Dział inżynierijny przedyskutował ten pomysł i nadał mu rozmiar średni (M).

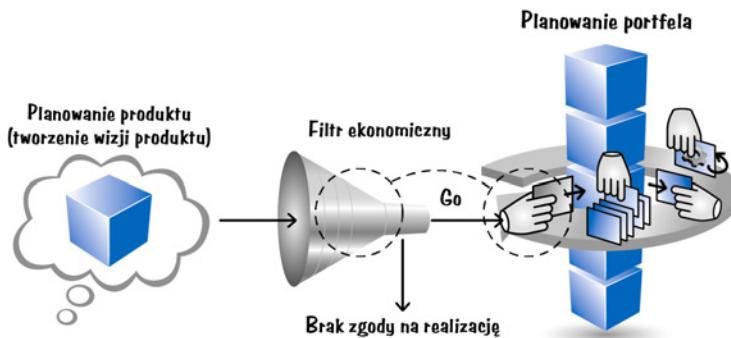
Na tej podstawie dział marketingowy był w stanie zdecydować, czy korzyści z realizacji przekroczą koszt średniego projektu (od 50 tysięcy do 125 tysięcy). Ta informacja była równie pomocna jak wypracowywana wcześniej wielkim kosztem wartość, powiedzmy, 72 381,27 — na pierwszy rzut bardziej precyzyjna, ale niemal na pewno niewłaściwa. Ta organizacja przekonała się, że przestały koszty są dostatecznie dokładne i eliminują marnotrawstwo bez nadmiernego podnoszenia oczekiwania lub też oferowania fałszywego poczucia bezpieczeństwa.

## Strategie napływu

W rozdziale 17. będę pisał o tym, że proces planowania nowego produktu ujawnia szczegóły dotyczące jego wizji, a także ma na celu zgromadzenie informacji potrzebnych decydentom do podjęcia decyzji o finansowaniu prac nad tym produktem. Strategie napływu dotyczą sposobów stosowania filtrów ekonomicznych organizacji w celu podjęcia takiej decyzji o finansowaniu. Mają one również za zadanie: równoważenie tempa umieszczania produktów w rejestrze portfela z tempem wy ciągania produktów z tego rejestru, szybkie wykorzystanie chwilowych okazji, kiedy te zostaną ujawnione, a także unikanie powstawania wąskich gardeł w portfelu dzięki zastosowaniu mniejzych, ale częstszych wersji dystrybucyjnych.

## Zastosowanie filtra ekonomicznego

Wynikiem planowania produktu jest jego wizja oraz zestaw informacji umożliwiających osiągnięcie jasnego poziomu pewności związanego z aktywnością tego planowania (patrz rozdział 17.). Wynik ten stanowi dane wejściowe przy planowaniu portfela (patrz rysunek 16.1). W oparciu o nie organizacja powinna podjąć decyzję o rozpoczęciu lub porzuceniu dalszych prac deweloperskich nad produktem. Działania te określam mianem nakładania filtra ekonomicznego na nowy produkt i sprawdzania, czy spełnia on kryteria organizacji uprawniające do rozpoczęcia finansowania (patrz rysunek 16.4).

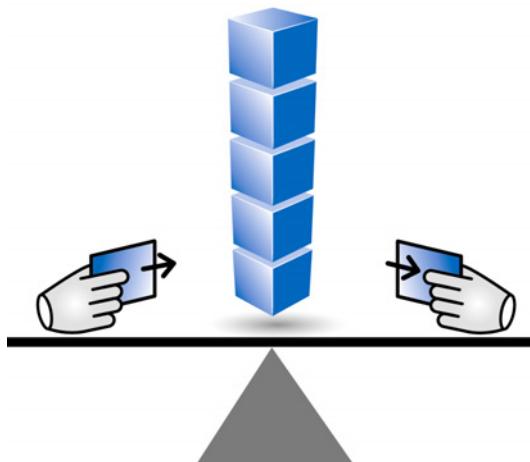


**RYSUNEK 16.4.** Stosowanie filtru ekonomicznego

Chociaż każda organizacja powinna zdefiniować filtr ekonomiczny najlepiej pasujący do jej polityki finansowania, dobry filtr ekonomiczny powinien szybko zapewniać zielone światło wszelkim okazjom dającym szanse na ogromne zyski w relacji do kosztów produkcji. Niemal każde inne rozwiązanie (o ile nie występują jakieś warunki szczególne) powinno zostać odrzucone. Jeżeli wynikowa wartość wynikająca z wytworzenia produktu jest zdecydowanie większa od kosztu jego produkcji, dyskutowanie tej kwestii powinno być zwięzłe — zatwierdzamy i przechodzimy do ustalenia kolejności w rejestrze portfela. Jeżeli okaże się, że przed podjęciem decyzji sprzeczamy się o małą różnicę w kosztach lub wartości, powinniśmy odrzucić produkt ze względu na brak ogólnego poparcia ekonomicznego dla jego produkcji. W większości organizacji jest zwyczajnie zbyt wiele okazji do stworzenia produktów o wysokiej jakości, aby skupiać się na okazjach niepewnych.

### Równoważ tempo przybywania z tempem ubywania

W praktyce chcielibyśmy, aby produkty napływały do rejestru portfela równomiernie do tempa usuwania ich z tego rejestru (patrz rysunek 16.5).



**RYSUNEK 16.5.** Równoważenie tempa przybywania z tempem ubywania

Nie chcemy przeciągać rejestru portfela przez umieszczanie w nim zbyt wielu produktów jednocześnie. Takie działanie spowoduje przeciążenie systemu.

W celu zilustrowania, dlaczego tak się stanie, założymy, że chcesz pójść na obiad do swojej ulubionej restauracji. Wsiadasz do samochodu i jedziesz na miejsce. Dojeżdżając, zauważasz duży autobus turystyczny, z którego wysiada właśnie cała gromada głodnych seniorów i udaje się wprost do restauracji.

Co zrobiłbyś w takiej sytuacji? Czy mimo wszystko poszedłbyś do restauracji i próbował zjeść tam obiad? Jeśli tak, jakie Twoim zdaniem byłyby konsekwencje wejścia do restauracji w tym samym czasie całej gromady głodnych seniorów? Istnieje duże prawdopodobieństwo, że ich obecność przeciąży możliwości restauracji. Decydując się na pozostanie, godzisz się na długie i frustrujące doświadczenie. Być może powinieneś wrócić do swojego samochodu i pojechać do innej restauracji!

Wiele organizacji przeprowadza coroczne spotkanie planowania strategicznego zazwyczaj w okolicach trzeciego kwartału roku finansowego. Jednym z wyników takiego planowania jest kompletna lista produktów, nad którymi organizacja będzie pracować w następnym roku finansowym. Produkty te zostają następnie jednocześnie umieszczone w rejestrze portfela, co prowadzi do przeciążenia procesu planowania portfela.

Nie sugeruję w ten sposób, że organizacje nie powinny przeprowadzać planowania strategicznego. Powinny one zdefiniować swój strategiczny kierunek, ale bez uwzględniania *wszystkich* detali (na poziomie produktów) opisujących sposób realizacji tej strategii. Decydowanie raz na rok o całej pracy na następny rok finansowy lub dłużej, a następnie wstawianie jednocześnie wszystkich elementów reprezentujących tę pracę do rejestru portfela jest decyzyją o fundamentalnym znaczeniu (w wielu organizacjach nieodwracalną), podjętą w obliczu wielkiej niepewności i naruszającą zasadę pozostawiania opcji otwartymi do ostatniego możliwego momentu (patrz rozdział 14.).

Decydowanie o całym portfelu produktów jednocześnie narusza również zasadę stosowania rozsądnych ekonomicznie rozmiarów zapotrzebowania (o czym była mowa w rozdziale 3.). Przetwarzanie dużej porcji produktów w celu określenia ich kolejności w rejestrze produktu jest bardzo kosztowne i wnosi potencjalne marnotrawstwo (ponieważ próbujemy zaplanować na rok lub dłużej do przodu). Wysoki koszt wynika nie tylko z dużej ilości produktów do przetworzenia, ale również jest wynikiem tego, że duża liczba elementów w rejestrze portfela komplikuje układanie harmonogramu prac, o którym pisałem wcześniej w tym rozdziale. Określenie dobrej kolejności jest o wiele mniejszym problemem, jeżeli jest mniej elementów do ułożenia. Mówiąc dokładniej: przy małej liczbie elementów rejestru portfela każda forma harmonogramu, która unika przesadnie nierozsądnej priorytetyzacji, jest wystarczająco dobra.

Aby przewyciążyć zalewanie rejestru portfela wszystkimi produktami jednocześnie, możemy zacząć wprowadzać produkty do rejestru w bardziej regularnych odstępach czasu, na przykład co miesiąc (lub co kwartał), a nie tylko raz do roku. Takie działanie znacznie redukuje wysiłek (i koszt) potrzebny do przejrzenia i wstawienia nowych produktów do portfela, a także zapewnia lepszą ogólną stabilność i przewidywalność w planowaniu portfela.

Powinniśmy się również skupić na mniejszych produktach (przyjrzyj się strategii mniejszych i częstszych wersji dystrybucyjnych). To powinno skutkować jednostajnym strumieniem wykonanych produktów i tym samym zwalnianiem zasobów, które mogą pobierać kolejne nowe produkty z rejestru portfela w stałym tempie. Regularne pobieranie produktów z rejestru portfela pozwoli zrównoważyć odpływ z napływem.

Ostatnia rzecz. Kiedy rozmiar rejestru portfela zacznie rosnąć, możemy zacząć sterować napływem nowych produktów do rejestru portfela. Jednym ze sposobów na to jest podniesienie kryteriów akceptacji w filtrze ekonomicznym, tak aby przepuszczać jedynie produkty o wysokiej wartości. To zmniejszy tempo wstawiania i pomoże w osiągnięciu równowagi z tempem odpływu.

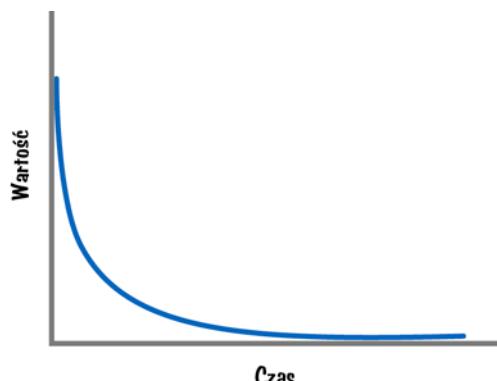
## Szybko wykorzystuj nadarzające się okazje

Planowanie portfela musi uwzględniać **nadarzające się okazje**. Nadarzająca się okazja to coś, co wcześniej było nieznane lub traktowane jako niemożliwe do wystąpienia, a zatem uważane za niewartego poświęcania pieniędzy.

Dla przykładu: pewna organizacja, w której pracowałem, brała udział w internetowym rynku zakładów. Jej działalność biznesowa podlega ścisłym regulacjom pozwalającym oferować jej zakłady bukmacherskie. Organy regulujące ten rynek na całym świecie są do pewnego stopnia nieprzewidywalne — szczególnie w przypadku hazardu — stąd bardzo trudno jest stwierdzić, czy i kiedy będzie można zaoferować swoje usługi w danym rejonie świata. Pracując w takim środowisku, trzeba być przygotowanym na nadarzające się okazje, ponieważ dane regulacje prawne mogą ulec zmianie wraz ze zmianą rządów.

Jedną z takich okazji była możliwość udostępnienia zakładów dla wyścigów konnych w stanie Kalifornia. Stan ten posiada znaczną liczbę torów wyścigowych, co w przypadku zmiany prawa czyniło tę okazję bardzo lukratywną (trzeba dodać, że taka zmiana faktycznie miała miejsce — nielegalne od lat zakłady bukmacherskie w Internecie stały się legalne od maja 2012 roku). Gdyby organizacja była przyzwyczajona do planowania strategicznego raz do roku w październiku (przed zmianą prawa), przegapiłaby okazję do wykorzystania tej nadarzającej się okazji — chyba że zdecydowałaby się podjąć ryzyko zbudowania giełdy zakładów dla rynku, który nie istniał i mógł nigdy nie dojść do skutku.

Tego typu nadarzająca się okazja musi zostać wykorzystana szybko. Bycie drugim w segmencie zakładów internetowych w Kalifornii dałoby bardzo mały lub wręcz zerowy udział w rynku. Ten często spotykany przypadek bardzo szybkiego spadku wartości nadarzającej się okazji wraz z upływem czasu ilustruje rysunek 16.6.



**RYSUNEK 16.6.** Wartość wielu nadarzających się okazji bardzo szybko maleje

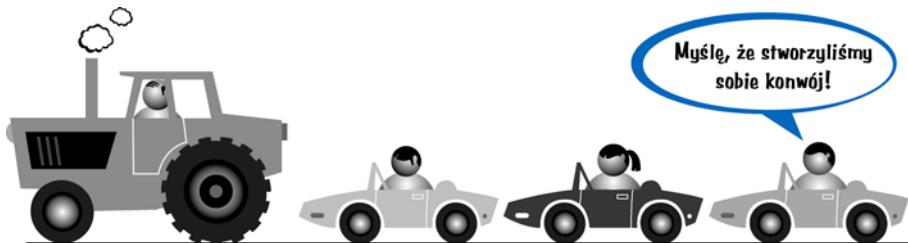
Przy braku szybkich działań, kiedy tylko okazja stanie się dostępna, niemal natychmiast tracimy całą jej wartość ekonomiczną, a próbując podążyć za tym celem nieco później (na przykład w trakcie następnej corocznej sesji planowania strategicznego), popełniamy błąd finansowy.

Jeżeli organizacja regularnie i dostatecznie często dokonuje oceny okazji, na przykład co miesiąc, i dysponuje dobrze funkcjonującym filtrem ekonomicznym przy jednoczesnym wypuszczaniu małych wersji dystrybucyjnych i stosowaniu limitu pracy cząstkowej, nigdy nie będzie musiała czekać długo na rozważenie nadarzającej się okazji.

## Planuj mniejsze i częstsze wersje dystrybucyjne

Jak pisałem w rozdziale 14., ekonomia mniejszych, bardziej regularnych wersji dystrybucyjnych jest bardzo przekonująca. Rysunki 14.4 i 14.5 wraz z tabelą 14.1 ilustrują, że dzieląc produkt na serię mniejszych przyrostowych wersji dystrybucyjnych, niemal zawsze możemy zwiększyć zyski z cyklu życia.

Oprócz tej znaczącej korzyści jest jeszcze jeden powód, dla którego chcemy zarządzać portfelem mniejszych i częstszych wersji dystrybucyjnych — unikanie efektu konwoju (patrz rysunek 16.7).



**RYSUNEK 16.7.** Duże produkty w rejestrze portfela tworzą konwój

Co się stanie, jeśli będziesz jechał wiejską drogą z jednym pasem ruchu i utkniesz za dużym ciągnikiem rolniczym (jak ten pokazany na rysunku 16.7)? Istnieje duże prawdopodobieństwo, że Ty i cały sznur mniejszych samochodów zostaniecie na długo uwięzieni za większym i wolniejszym pojazdem. Przyczyna powstania konwoju jest oczywista — duży ciągnik blokuje całą drogę (współ-dzielony zasób).

Taki sam scenariusz będzie miał miejsce, jeśli wpuścimy do rejestru portfela duże produkty. Duże produkty wymagają znacznych zasobów przez długi okres. Zasoby te przestaną być dostępne dla wielu mniejszych produktów, które trafią do kolejki za dużym projektem. Czekając w tej kolejce, każdy z nich będzie generował koszty opóźnienia. Kiedy zsumujemy ze sobą wszystkie koszty opóźnień tych małych produktów i wezmijmy pod uwagę niezaprzecjalną ekonomię wykonywania mniejszych, przyrostowych wersji dystrybucyjnych, jasne okaże się, że duże produkty wnoszą istotną stratę do zysków z cyklu życia.

Żeby zwalczyć ten problem, niektóre organizacje wprowadzają specjalną politykę rozmiarową podczas planowania portfela, która określa dopuszczalną wielkość wysiłku deweloperskiego. Spottedkany przeze mnie przykładem takiej polityki było przepuszczanie jedynie produktów, dla których wysiłek deweloperski był nie dłuższy niż dziewięć miesięcy. Wszelkie propozycje większych produktów były natychmiast odrzucone, a osobom przedstawiającym te produkty proponowano, aby znalazły sposób na ich wdrożenie w formie mniejszych i częstszych wersji dystrybucyjnych.

Pracowałem również z organizacjami, które kierowały się mottem „Nie możemy nigdy założyć, że kiedykolwiek powstanie druga wersja jakiegokolwiek z naszych produktów”. Takie przekonanie stoi w zupełnej sprzeczności z wykonywaniem małych, częstych wersji dystrybucyjnych. Jeżeli uważaemy, że nigdy nie będziemy mieli okazji wypuścić drugiej wersji, naturalną konsekwencją tego podejścia jest umieszczenie w pierwszej wersji wszystkiego, co się da, a także wszystkiego, co naszym zdaniem może być przydatne w przyszłości. Skutkiem tego nie tylko tworzymy większe wysiłki deweloperskie, ale niemal z całą pewnością opóźniamy produkcję cech o wysokiej wartości w innych produktach, pracując w tym czasie nad cechami o niskiej wartości w naszym dużym produkcie. Takie podejście z ekonomicznego punktu widzenia ma działanie niszczące. Organizacje muszą jasno dawać do zrozumienia, że kolejne wersje mogą być i będą wykonywane w oparciu o ich indywidualne oceny ekonomiczne, a także że planowanie z założeniem pojedynczej wersji jest wysoce niezalecane.

## Strategie odpływu

Strategie zarządzania odpływem pomagają organizacjom w podejmowaniu decyzji o wyciągnięciu produktu z rejestru portfela. Opiszę trzy takie strategie:

- skupianie się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę;
- ustanawianie limitu pracy częściowej;
- czekanie na cały zespół.

### Skup się na pracy czekającej na realizację, nie na pracownikach czekających na pracę

Kluczowa strategia określania, kiedy należy wyciągnąć produkt z rejestru portfela, to pamiętanie o zasadzie, którą omówiłem w rozdziale 3. — skup się na pracy czekającej na realizację, nie na pracownikach czekających na pracę. Zasada ta mówi, że praca czekająca na realizację wprowadza o wiele większe marnotrawstwo i szkodę ekonomiczną niż pracownicy czekający na pracę. Jest to sprzeczne ze sposobem zarządzania swoim portfelem przez wiele organizacji.

Powszechnie, ale mylne podejście przy wdrażaniu produktów do prac deweloperskich to:

1. Wyciągnąć produkt z rejestru portfela i przypisać ludzi do pracy nad nim.
2. Czy wszyscy ludzie są wykorzystani w 100% (100% ich czasu zajmuje praca)?

Jeśli nie, powtórz krok 1.

Takie podejście spowoduje, że wszyscy będą bardzo zajęci, ale doprowadzi również do spowolnienia prac nad wszystkimi produktami i zwiększy liczbę popełnianych błędów. O wiele lepszym podejściem jest rozpoczęcie pracy nad produktem, tylko jeśli możemy zapewnić dwie rzeczy: dobry przepływ pracy nad nowym produktem i brak przerw w przepływie pracy przy innych aktywnych produktach. Ta strategia jest używana przy ścisłej koordynacji z kolejną strategią — ustanawianiem limitu pracy częściowej.

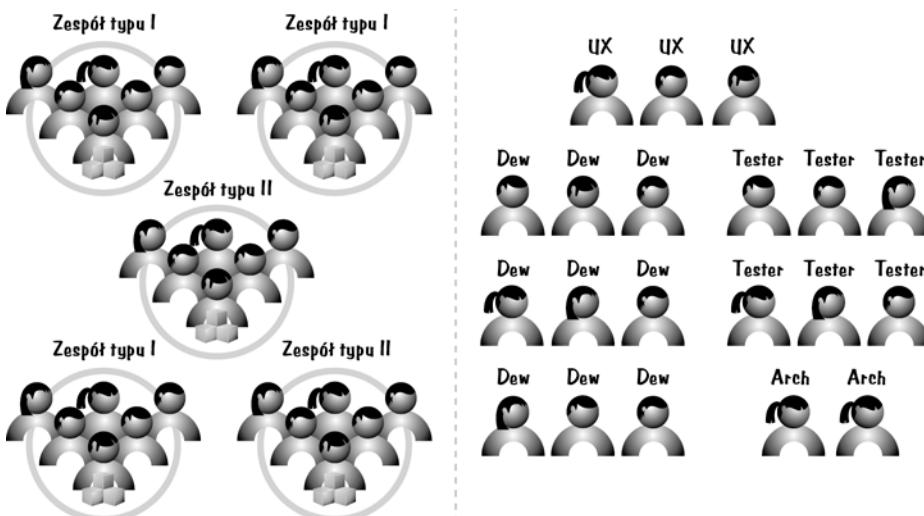
## Ustal limit pracy cząstkowej

Rozważ następujący scenariusz. Czy byłes kiedyś w restauracji i widziałeś wolne stoliki, a mimo to obsługa nie posadziła Cię przy żadnym z nich? Jeśli tak, wiesz, że jest to frustrujące. Być może po myślałeś sobie: „Dlaczego mnie nie posadzą? Przecież mają wolne stoliki. Nie interesują się mną?”. Założmy, że kilku z kelnerów zachorowało w owym dniu. W takiej sytuacji mądry restaurator nie powinien Cię posadzić. Co się stanie, jeśli jednak to zrobi? Być może będziesz musiał czekać 45 minut, zanim kelner podejdzie do Twojego stolika. Nie wiem jak Ty, ale ja nie byłbym szczęśliwy, czekając 45 minut, zanim ktoś do mnie podejdzie i porozmawia ze mną! Wolałbym, gdyby od razu postawiono sprawę jasno — „Przepraszamy, ale czterech naszych kelnerów jest dzisiaj na zwolnieniu lekarskim, więc trzeba czekać do 45 minut, zanim będziemy mogli pana posadzić”. Taka informacja dałaby mi możliwość zaczekania lub pójścia gdzie indziej.

Jeszcze gorzej byłoby, gdyby jednak posadzono mnie przy dostępnym stoliku i spróbowało obsłużyć. Ucierpiałaby na tym obsługa pozostałych klientów znajdujących się w restauracji. Obsadzenie zbyt wielu stolików w stosunku do dostępnej liczby kelnerów sprawi, że każdy z nich będzie przepracowany i wszyscy odczuja to negatywnie. Właśnie dlatego mądry restaurator nie pozwoli na obsadzanie stolików ponad dostępne zasoby.

Gdybyśmy tylko potrafili podążyć śladem mądrego restauratora podczas planowania portfela. Nie powinniśmy nigdy wyciągać z portfela produktów ponad dostępne zasoby do ich wykonania. Takie postępowanie zmniejszy dostępne moce przerobowe dla każdego z produktów (doprowadzając do opóźnienia każdego z nich), a także pogorszy jakość pracy nad wszystkimi produktami. Wykonywanie pracy w wolniejszym tempie i z gorszą jakością nie jest strategią sukcesu.

Jak zatem określić właściwy limit pracy cząstkowej? W rozdziale 11. omówiłem ideę reprezentowania przez zespoły jednostki pojemności, której powinniśmy użyć do ustalenia limitu pracy cząstkowej. Wiedząc, ile mamy zespołów scrumowych oraz nad jakimi produktami mogą one pracować, możemy stwierdzić, które prace deweloperskie możemy realizować jednocześnie (patrz rysunek 16.8).



**RYSUNEK 16.8.** Zespoły są jednostką pojemności umożliwiającą ustalenie limitu pracy cząstkowej

Lewa strona rysunku 16.8 pokazuje trzy zespoły mogące pracować nad I typem produktów i dwa zespoły mogące pracować nad II typem produktów. Ta informacja byłaby doskonałym punktem startowym do ustalenia maksymalnej liczby produktów każdego typu, nad jakimi nasza organizacja może pracować jednocześnie. Wyobraź sobie, o ile trudniej byłoby wyliczyć prawidłową liczbę jednoczesnych projektów deweloperskich, gdybyśmy posługiwali się wyłącznie liczbą pracowników o określonych umiejętnościach (prawa strona rysunku 16.8).

## Zaczekaj na cały zespół

Ostatnią strategią odpływu jest wstrzymanie pracy nad produktem do momentu gotowości całego zespołu scrumowego. Organizacje naruszające zasadę „skupiania się na pracy czekającej na realizację, a nie na pracownikach czekających na pracę” często zaczynają pracę nad produktem w chwili, kiedy tylko część osób jest dostępna. Ich sposób myślenia może być następujący: „Kilku deweloperów ma trochę wolnego czasu, niech zatem zaczną popychać do przodu następny produkt”.

Ta strategia ma wadę, ponieważ doprowadzi do zablokowania jeszcze większej ilości pracy w innych produktach, spowalniając tempo ich dostarczania i generując znaczne koszty opóźnienia.

Ponieważ w Scrumie jednostką pojemności jest zespół, nie powinniśmy rozpoczynać pracy nad produktem, jeżeli nie dysponujemy całym zespołem scrumowym. Z punktu widzenia Scruma takie postępowanie jest pozbawione sensu. Niekompletny zespół scrumowy nie jest w stanie doprowadzać cech do stanu spełniającego kryteria ukończenia.

Jednym możliwym do rozważenia odstępstwem są produkty wymagające wielu zespołów scrumowych. Założmy, że mamy produkt wymagający trzech zespołów scrumowych. Jeżeli jeden z zespołów scrumowych jest dostępny w całości i rozsądne wydaje się, aby mógł on rozpocząć prace deweloperskie, wtedy rozważyłbym wdrożenie prac nad produktem. Pozostałe zespoły mogą być włączane do prac, kiedy tylko będą dostępne.

## Strategie aktywności

Strategie zarządzania produktami aktywnymi mają za zadanie pomóc nam w stwierdzeniu, czy w przypadku aktywnie rozwijanego produktu należy zachować bieżącą ścieżkę, wykonać zwrot, dostarczyć gotowe rozwiązanie lub może zakończyć cały projekt. Tego typu decyzje powinniśmy podejmować w regularnych odstępach czasu (powiedzmy pod koniec każdego sprintu), a także od czasu do czasu niezależnie od cyklu, kiedy wystąpią warunki wyjątkowe wymagające przeanalizowania naszych aktywnych produktów.

Istnieje wiele różnych strategii, które moglibyśmy tutaj rozważyć, nie zapominając jednocześnie, że działa nadzorcze w organizacjach posiadają zapewne swoje własne wytyczne dotyczące sposobów radzenia sobie z produktami aktywnymi. Ja skupię się jednak tylko na jednej z nich — ekonomii końcowej. Powinna być to strategia nadzędna kierująca decydowaniem — będąca w zgodzie z podstawowymi zasadami Scruma i zwinnością opisywanymi przeze mnie w tej książce.

## Wykorzystanie ekonomicznej analizy marginalnej

Patrząc z perspektywy ekonomicznej, cała praca włożona w produkt do momentu podjęcia decyzji jest „kosztem utopionym”. Przy podejmowaniu następnego kroku w przód interesuje nas wyłącznie ekonomiczna analiza marginalna. Powinniśmy pytać siebie, czy wyłożenie dodatkowej porcji pieniędzy jest opłacalne z punktu widzenia dodatkowego zwrotu z inwestycji. Podjęcie tej decyzji bez patrzenia na koszty już poniesione nie jest łatwe.

Wykorzystując analizę marginalną, możemy zdecydować, co zrobić z produktami, które są aktualnie rozwijane. Podczas analizy z punktu widzenia ekonomii końcowej istnieją cztery podstawowe opcje do wyboru:

- zachować — kontynuować prace deweloperskie nad produktem;
- dostarczyć — przerwać prace nad produktem i dostarczyć go w obecnej postaci;
- wykonać zwrot — przyjąć to, czego się nauczyliśmy, i zmienić kierunek działania;
- przerwać — zakończyć prace nad produktem i go porzucić.

Proces podejmowania decyzji związany z tymi czterema opcjami przedstawia rysunek 16.9.



**RYSUNEK 16.9.** Proces podejmowania decyzji w odniesieniu do produktów aktywnych bazujący na analizie marginalnej.

Jeżeli kolejna inwestycja w bieżący produkt jest uzasadniona ekonomicznie, najbardziej prawdopodobnym wyborem będzie *zachowanie*. Jest to scenariusz, według którego dokonujemy przeglądu aktywnego produktu i dochodzimy do wniosku, że powinniśmy kontynuować wydawanie pieniędzy na jego rozwój.

Jeżeli dalsze inwestowanie w produkt nie jest uzasadnione ekonomicznie, powinniśmy zdecydować, czy chcemy dostarczyć go tak jak jest, wykonać zwrot lub też pozbyć się produktu.

Jeżeli stworzony przez nas produkt zawiera minimalny zestaw kwalifikujący do dystrybucji, możemy rozważyć jego *dostarczenie*. Jeśli tak nie jest, idziemy złą ścieżką i jeśli myślimy, że istnieje inna warta eksploracji, możemy *dokonać zwrotu* na nową drogę produktu. Ta opcja będzie wymagała prawdopodobnie ponownego przeprowadzenia planowania produktu w celu rozważenia tej nowej ścieżki (patrz rozdział 17.).

Jeśli w końcu dalsze finansowanie inwestycji nie znajduje uzasadnienia, jesteśmy niezadowoleni z obecnego stanu i ewentualnych możliwych *zwrotów* w kierunku prac, jedyną rozsądnią opcją wydaje się *przerwanie* produktu.

Ignorowanie analizy marginalnej może skutkować bardzo nierozsądnymi zachowaniami. Oto pytanie warte rozważenia: Czy jeśli w Twojej organizacji wydasz pierwszą złotówkę na prace deweloperskie, mogą zaistnieć jakiekolwiek okoliczności skłaniające Cię do przerwania dalszej produkcji? Ja wielokrotnie z zaskoczeniem słyszałem stwierdzenia osób mówiących, że ich organizacje nigdy nie decydują się na przerwanie produktu (lub robią to bardzo rzadko) po wydaniu na niego chociaż złotówki — ich strategia sprowadza się do stwierdzenia „jeżeli powiedziało się A, trzeba powiedzieć B”.

Jedno z uzasadnień nieprzerywania produktów przez firmę zaskoczyło mnie szczegółowo. Zapytałem zarząd IT: „Załóżmy, że rozpoczęcie pracy nad produktem, który waszym zdaniem będzie cenny dla 100% waszych klientów, a jego koszt wytworzenia to milion złotych. Po wydaniu tego miliona dowiadujecie się, że produkt będzie miał wartość jedynie dla 10% waszych klientów, a jego całkowita produkcja pochłonie 10 milionów. Czy wydalibyście następne dziewięć milionów na dokończenie tego projektu?”. Ich odpowiedź: „Nie rozumiesz naszej księgowości. Jeśli przewiemy projekt po wydaniu jednego miliona przed jego wdrożeniem, dział IT odnotuje wpływ tego jednego miliona akonto swojego budżetu. Jeżeli wydamy pozostałe dziewięć milionów i wdrożymy system chociaż na jeden dzień, cały jego koszt zostanie przeniesiony do jednostki biznesowej i to ona będzie nim obciążona”.

Ten przykład jasno pokazuje, że próba przechytrzenia systemu księgowego doprowadziła do utraty zdrowego rozsądku.

Analiza marginalna jest niezwykle funkcjonalnym narzędziem pozwalającym postępować w sposób właściwy i ujawniającym wszelkiego typu głupie i marnotrawne zachowania. Powinna to być Twoja główna strategia postępowania podczas podejmowania decyzji odnośnie do przyszłości aktywnych produktów.

## Zakończenie

W tym rozdziale omówiłem 11 ważnych strategii służących do planowania portfela (zarządzania portfelem). Moim celem nie było przedstawienie kryteriów wyboru tych strategii do własnego użytku. Każda z nich jest uzupełnieniem pozostałych. Największą korzyść osiągniesz, stosując wszystkie z nich jednocześnie. Gdybym jednak został zmuszony do wybrania tylko jednej strategii z każdej kategorii, skupiłbym się na koszcie opóźnienia, mniejszych i częstszych wersjach dystrybucyjnych, ograniczaniu pracy częstekowej i analizy marginalnej.

W następnym rozdziale omówię planowanie produktu (tworzenie wizji produktu). Wynikiem tego procesu są kandydatury produktów do rozważenia w czasie planowania portfela.

## Rozdział 17

# PLANOWANIE PRODUKTU (TWORZENIE WIZJI PRODUKTU)

---

Przed rozpoczęciem wytwarzania wartości dla klienta w pierwszym sprincie potrzebujemy początkowego rejestru produktu. Do jego stworzenia potrzebujemy z kolei wizji produktu. Wiele organizacji uznaje również za wskazane stworzenie mapy drogowej produktu, która definiuje potencjalną serię przyrostowych wersji dystrybucyjnych. Twój organizacja może również preferować tworzenie innych artefaktów. Proces tworzenia tych artefaktów określę mianem planowania na poziomie produktu lub tworzenia wizji produktu.

W tym rozdziale opiszę podejście do tworzenia wizji współgrające z zasadami Scruma. Nadaje się ono również dla organizacji, które próbują tworzyć produkty przy użyciu Scruma, ale nadal stosują politykę wstępnego zatwierdzania niespełniającą kryteriów zwinności.

## Wprowadzenie

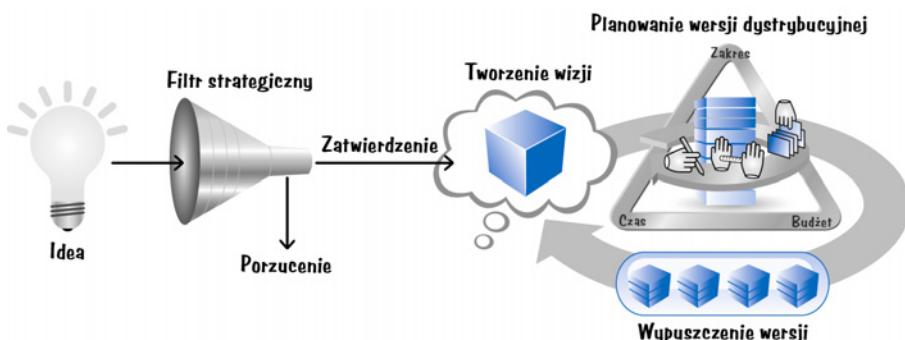
Powiedzmy, że w Twojej głowie zrodziła się intrygująca idea nowego produktu lub następnej wersji produktu już istniejącego. Celem stworzenia wizji jest rozwinięcie tej idei, opisanie esencji potencjalnego produktu i wygenerowanie przybliżonego planu jego stworzenia. Pod koniec tworzenia wizji powinieneś posiadać wystarczającą pewność, aby móc przekazać tę ideę na spotkaniu planowania portfela (patrz rozdział 16.), gdzie będziesz mógł zdecydować, czy chcesz sfinansować dalszy poziom bardziej szczegółowych prac deweloperskich.

Tworzenia wizji produktu w Scrumie nie należy mylić ze znacznie bardziej skomplikowaną ceremonią — intensywnym planowaniem całego projektu produktu. W Scrumie nie uważamy, że możemy znać (lub powinniśmy próbować poznać) wszystkie możliwe detale dotyczące produktu przed rozpoczęciem prac. Rozumiemy jednak, że finansowanie produktu nie może się odbyć bez jakiejkolwiek wizji — szczegółów wystarczających do zrozumienia klientów, cech, rozwiązań na wysokim poziomie, a także szacunków odnośnie do spodziewanych kosztów produktu.

Nie poświęcamy zbyt dużo czasu lub też energii na stworzenie wizji, ponieważ chcemy bezzwłocznie przejść dalej, poza fazę domysłów, w której *myślmy*, że wiemy, jakie potrzeby ma klient i jakie rozwiązanie będzie skuteczne, do fazy szybkiej pętli zwrotnej — sprintów przynoszących konkretną wartość dla klienta. W końcu dopiero kiedy faktycznie zaczniemy implementować rozwiązanie poprzez ciąg iteracji w naszym skomplikowanym środowisku, zdobędziemy potwierdzoną wiedzę na temat realiów, w których nasz produkt musi funkcjonować i rozwijać się.

## Czas

Tworzenie wizji związane z planowaniem na poziomie produktu jest aktywnością ciągłą, a nie jednorazowym wydarzeniem (patrz rysunek 17.1).



**RYSUNEK 17.1.** Tworzenie wizji produktu jest aktywnością nieustającą

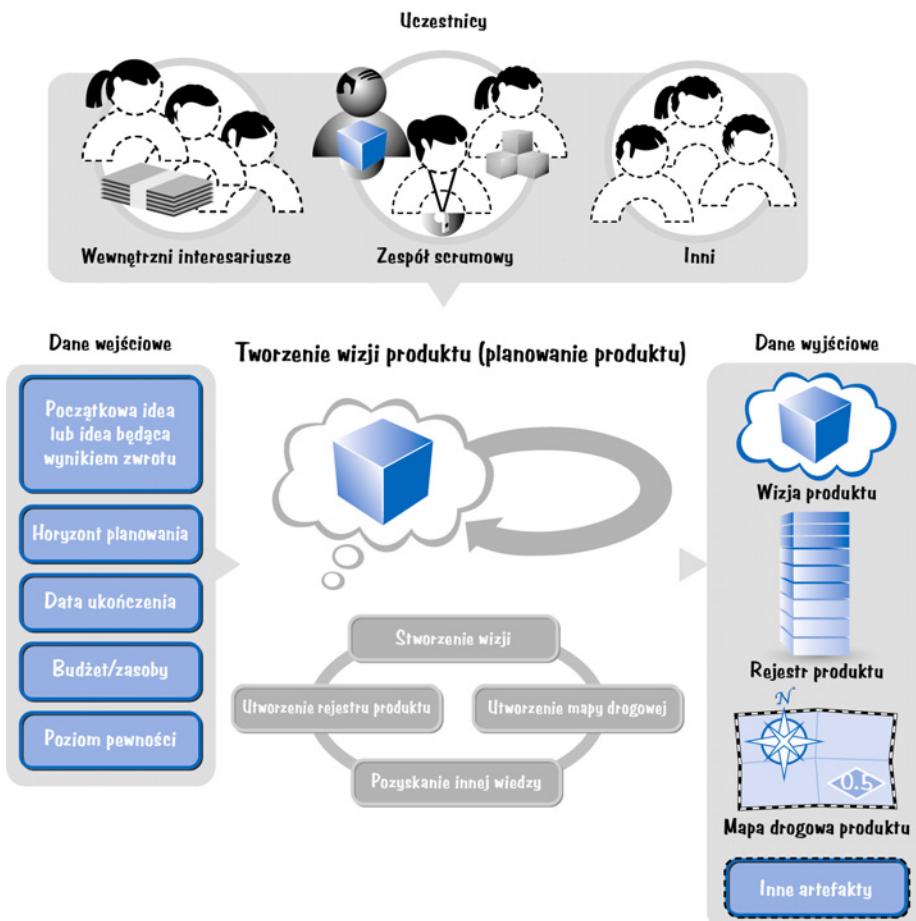
Stworzenie wizji rozpoczyna się od idei produktu, na którą wpadła jakaś osoba lub zespół. Idea ta przechodzi następnie przez **filtル strategiczny** organizacji w celu sprawdzenia, czy jest zgodna ze strategicznym kierunkiem organizacji, a zatem warta dalszego analizowania i inwestowania.

Po pomyślnym przejściu idei przez filtr strategiczny zaczynamy wstępne tworzenie wizji. W trakcie tego procesu staramy się zrozumieć przyszły produkt w stopniu wystarczającym do zdefiniowania tego, czym powinna być pierwsza, minimalna wersja dystrybucyjna naszego produktu. Takie działanie pozwoli nam bardzo szybko dostarczyć rzeczywistą wartość przy niskim koszcie. Dodatkowo użytkownicy i klienci dostaną do rąk coś namacalnego tak szybko, jak będzie to możliwe, co pozwoli nam uzyskać informacje zwrotne potwierdzające lub odrzucające nasze założenia odnośnie do docelowych klientów, pożądanych cech i całego rozwiązania. Otrzymane dane mogą być w zgodzie z naszymi oczekiwaniemi i wzmacniać nasze pragnienie utrzymania bieżącej wizji. Z drugiej strony, sytuacja może okazać się zupełnie odmienną od tego, czego oczekiwaliśmy, zmuszając nas do dokonania zwrotu względem oryginalnego rozwiązania, przejrzenia tego, co robimy, i odpowiedniego dostosowania planu.

## Uczestnicy

Podczas wstępnego tworzenia wizji jedynym obowiązkowym uczestnikiem jest właściciel produktu. Zazwyczaj jednak właściciel produktu nadzoruje tworzenie wizji wspólnie z jednym lub kilkoma wewnętrznymi interesariuszami, którzy razem z nim wykonują pracę mającą na celu stworzenie

wizji. Ponadto w różnorodnych zadaniach tworzenia wizji biorą udział także inni specjaliści związani z takimi obszarami jak analiza rynku, rozwój modeli biznesowych, projektowanie interfejsu użytkownika czy architektura systemu. Aktywność tworzenia wizji przedstawia rysunek 17.2 (opcjonalni uczestnicy oraz artefakty są zaznaczone przerywaną linią).



**RYSUNEK 17.2.** Tworzenie wizji (planowanie) produktu

W idealnym przypadku we wstępny tworzeniu wizji produktu bierze również udział mistrz młyna i zespół deweloperski, który będzie wytwarzał faktyczną wartość dla klienta podczas kolejnych sprintów. Ich zadaniem jest dostarczenie cennych informacji podczas tworzenia wizji produktu oraz wyeliminowanie potrzeby przekazania wizji innemu zespołowi w celu zbudowania produktu. Często jednak organizacja wstrzymuje się z wyborem zespołu scrumowego do momentu zakończenia wstępnej wizji produktu, co uniemożliwia włączenie go w aktywność tworzenia tej wizji. Mistrz młyna i zespół deweloperski powinni jednak zostać włączeni we wszystkie zadania związane ze zmianą wizji, kiedy tylko cały zespół scrumowy (właściciel produktu, mistrz młyna i zespół deweloperski) będzie uformowany i ruszą prace deweloperskie.

## Proces

Głównym parametrem wejściowym do utworzenia początkowej wizji produktu jest pomysł (idea), który przeszedł przez filtr ekonomiczny, natomiast w przypadku ponownego rozważania wizji byłby to pomysł zmieniony (w wyniku zwrotu). Jest to pomysł, który podlegał modyfikacjom lub był rewidowany w wyniku informacji uzyskanych od użytkowników i klientów, zmian w finansowaniu, nieprzewidywalnych posunięć konkurencji lub innych istotnych zmian pojawiających się w złożonym środowisku, w jakim pomysł ten musi funkcjonować.

Potrzebujemy również innych danych wejściowych. Po pierwsze: wskazania pewnego horyzontu planowania — jak daleko w przyszłość powinniśmy sięgać, tworząc wizję produktu. Po drugie: musimy wiedzieć, kiedy (jeśli w ogóle) ma nastąpić koniec aktywności związanych z wypracowywaniem wizji, a także jakie zasoby i w jakiej ilości będziemy mieć przydzielone w celu przeprowadzania planowania produktu. I w końcu po trzecie: musimy znać **poziom zaufania**, czyli mówiąc inaczej „definicję ukończenia” dla planowania produktu. Poziom zaufania to zestaw informacji potrzebnych interesariuszom do komfortowego podjęcia decyzji o finansowaniu (lub zaprzestaniu finansowania) bardziej szczegółowych prac deweloperskich. O tym, czym jest rozsądny poziom zaufania, będę pisał w dalszej części tego rozdziału. Na koniec dodam jeszcze, że wszystkie dane do planowania produktu na rysunku 17.2 powinny być analizowane jednocześnie, a nie jedne po drugich.

Na tworzenie wizji składa się kilka różnych aktywności, z których każda generuje ważne wyniki, takie jak wizja produktu czy też początkowy rejestr produktu. Często tworzona jest również prosta mapa drogowa produktu pokazująca serię przyrostowych wersji dystrybucyjnych. W tym czasie możemy również wykonywać inne aktywności, które pomogą nam osiągnąć właściwy poziom zaufania w ekonomiczny sposób.

## Przykład ZODC

Aby zilustrować aktywność tworzenia wizji, posłużę się nową, fikcyjną ideą produktu o nazwie ZmyślnaOpiniaDlaCiebie (w skrócie ZODC). Firma Opinia Całkowita jest liderem w przeglądach produktów i usług dostarczanych przez klientów. Jej podstawowa działalność to udostępnianie forum, na którym ludzie mogą wymieniać między sobą opinie na temat produktów i usług. Przychody firmy rosły w małym tempie przez kilka ostatnich lat, udowadniając jej dochodowość. Opinia Całkowita ma jednak wielu konkurentów wypuszczających innowacyjne cechy z alarmującą częstotliwością i w związku z tym potrzebuje pilnie nowej, innowacyjnej usługi, która pozwoli jej odskoczyć do przodu.

Opinia Całkowita dysponuje zaangażowanym zespołem marketingowym, który nieustannie monitoruje media społecznościowe, aby sprawdzać, jak klienci postrzegają jej usługi. Dzięki temu zespół dowiedział się, że wiele użytkowników spędza zbyt dużo czasu na ich stronie internetowej, próbując oddzielić „autentyczne” opinie na temat produktów od opinii „podejrzanych”. Dodatkowo wielu użytkowników twierdzi, że ogromna liczba opinii na temat niektórych produktów (na przykład odtwarzaczy DVD) lub usług (na przykład chińskich restauracji w centrum miasta) bardzo utrudnia im przebrnięcie przez masę informacji i uzyskanie dobrego obrazu całości.

Taki wywiad na rynku doprowadził zespół marketingowy do pomysłu na ZODC — rewolucyjny sposób identyfikowania, filtrowania i przeglądania w sieci opinii za pomocą agenta wyszukiwania, którego można wytrenować. Ich zdaniem ta idea może zostać przekształcona w innowacyjną usługę, której poszukuje firma. Ludzie z działu marketingu tworzą jednostronicową notatkę na temat ZODC, która zawiera listę ogólnych cech, opis docelowych klientów i kluczowe korzyści z posiadania usługi. Zespół wysyła następnie ten opis do Komitetu Zatwierdzania Nowych Produktów, który przegląda go w trakcie jednego z regularnie odbywających się Przeglądów Nowych Pomysłów (przeprowadzanych co drugą środę każdego miesiąca).

Wyższa kadra zarządzająca (z której wywodzi się Komitet Zatwierdzania Nowych Produktów) zgadza się z opinią marketingu, że ZODC stanowi znaczącą szansę dla Opinii Całkowitej do wyróżnienia się na swoim rynku. Komitet desygnuje następnie Roberta, przedstawiciela biznesowego z działu marketingu strategicznego, na właściciela produktu dla ZODC.

Zarząd zatwierdził środki na dwa tygodnie pracy w celu dokończenia wizji. W tym czasie członkowie komitetu zatwierdzającego dokonają przeglądu wyników stworzonej wizji i podejmą decyzję o finansowaniu pierwszych prac deweloperskich nad ZODC. Oprócz Roberta zarząd ustanowił jeszcze dwóch ekspertów, analityka rynku i kilku interesariuszy, którzy mają brać udział w stworzeniu wizji produktu. Zarząd nie wyraził jednak jeszcze zgody na większe wydatki potrzebne do stworzenia pełnego zespołu scrumowego.

Robert został poproszony o wykorzystanie dostępnych dla niego zasobów do wyprodukowania następujących rzeczy:

- wstępnej wizji produktu, rejestru produktu i mapy drogowej produktu;
- sprawdzenia głównego założenia, że użytkownicy będą zdecydowanie woleli wyniki przefiltrowane za pomocą ZODC od nieprzefiltrowanych danych (w dalszej części rozdziału opiszę, jak Robert i jego koledzy dostarczą tej potwierzonej wiedzy);
- opisu innych istotnych założeń (hipotez) odnośnie do potencjalnych użytkowników oraz cech, które powinna przetestować pierwsza wersja produktu;
- kilku kluczowych środków (metod działania) służących do sprawdzenia pozostałych założeń i zdobycia wiedzy na temat tego, czy pierwsza wersja ZODC spełniała pokładane w niej oczekiwania;
- listy pytań (znanych nieznanych), na które trzeba będzie znaleźć odpowiedź.

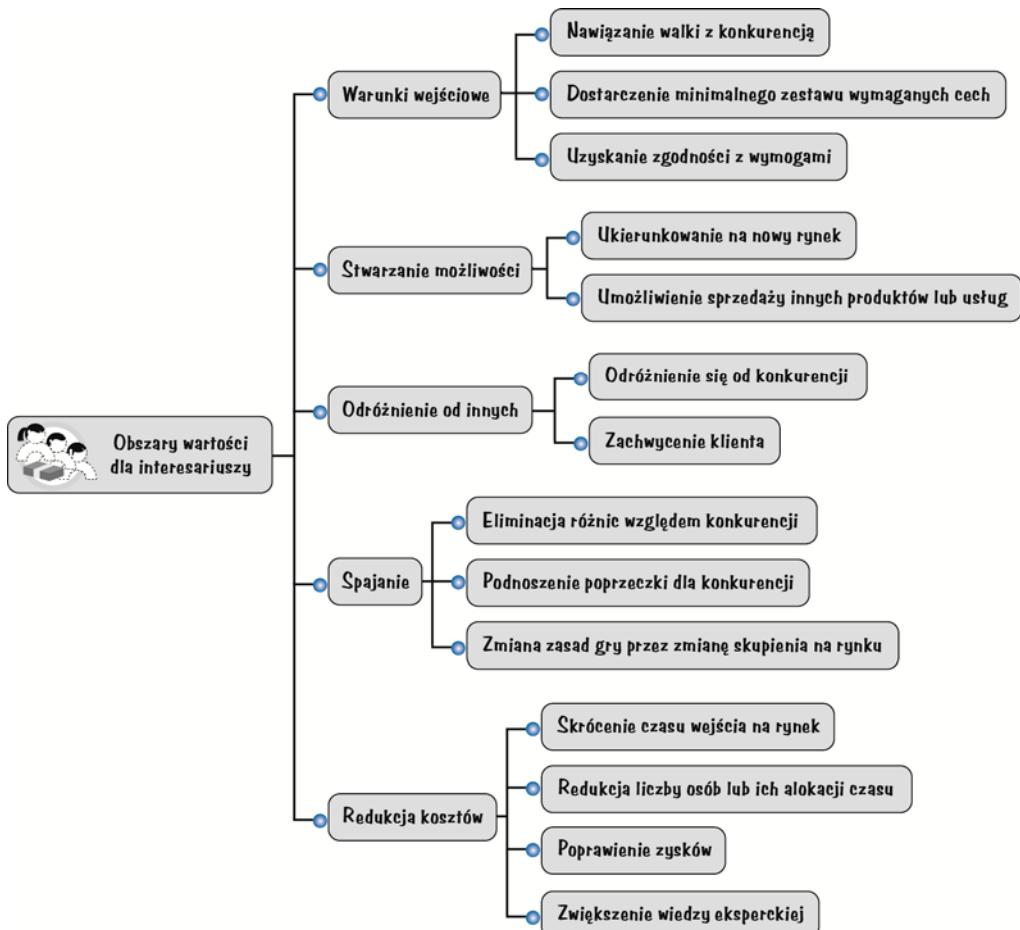
Bez tych informacji wyższa kadra zarządzająca nie będzie posiadała dostatecznego poziomu ufności, aby podjąć świadomą decyzję o kontynuacji prac deweloperskich nad pierwszą wersją produktu.

## Tworzenie wizji

Robert i pozostali interesariusze zaczynają od stworzenia wspólnej, przekonującej wizji ZODC. W Scrumie wizja nie jest szczegółowym, kilkusetstronicowym dokumentem. Jeśli potrzebujemy aż tyle miejsca do opisania naszej wizji, najprawdopodobniej sami jej nie rozumiemy. Wizje nawet złożonych produktów powinny dać się łatwo wyrazić i wyznaczać spójny kierunek dla ludzi, którzy będą je realizować. Weźmy za przykład wizję prezydenta Kennedy'ego o wyprawie na Księżyce:

„Uważam, że ten naród powinien zaangażować się w osiągnięcie celu, jakim będzie wyładowanie na Księżyku i bezpieczny powrót na Ziemię, przed końcem obecnej dekady” [Kennedy, 1961]. W mniej niż pięćdziesięciu słowach Kennedy był w stanie wyrazić śmiałą, jednoznaczną wizję, której realizacja wymagała ostatecznie współpracy tysięcy ludzi budujących złożone systemy składające się z setek tysięcy powiązanych ze sobą komponentów.

Podczas tworzenia produktów i usług wizja jest często wyrażana w kategoriach dostarczania wartości interesariuszom. Przykładami mogą być obszary wartości z kategorii przedstawionych na rysunku 17.3.



**RYSUNEK 17.3.** Obszary wartości dla interesariuszy

Sposób wyrażania wizji może być dowolny, poczynając od stwierdzenia w stylu Kennedy’ego do fikcyjnego artykułu przeglądowego w magazynie. Przykłady form wizji dla niektórych popularnych produktów i usług zostały przedstawione w tabeli 17.1 (w oparciu częściowo o [Highsmith, 2009]). Powinieneś wybrać formę, która będzie najlepiej odpowiadać Twojej organizacji, grupie zajmującej się tworzeniem pomysłów czy w końcu samej idei.

**TABELA 17.1.** Popularne formy przedstawiania wizji

Forma	Opis
Wyrażenie windowowe	Napisz krótki (wymagający od 30 sekund do jednej minuty) opis wizji produktu. Wyobraź sobie, że wszedłeś właśnie do windy z inwestorem i chcesz szybko przekazać mu informację na temat wizji swojego produktu. Czy byłbyś w stanie zrobić to w trakcie jazdy windy?
Karta katalogowa produktu	Napisz pierwszego dnia kartę katalogową produktu. Spróbuj zmieścić ją na frontowej stronie jednostronicowej ulotki marketingowej.
Wizja pudełka produktu	Narysuje pudełko, w którym będziesz chciał umieścić produkt, kiedy ten będzie gotowy do wypuszczenia na rynek. Czy jesteś w stanie wymyślić trzy lub cztery najistotniejsze punkty do zilustrowania na tym pudełku? (Rozrysowanie 15 punktów jest łatwiejsze od narysowania trzech lub czterech).
Slajdy na spotkanie z użytkownikami	Stwórz dwa lub trzy slajdy prezentacji, których użyłbyś do przedstawienia produktu na konferencji z użytkownikami (lub w trakcie innego podobnego wydarzenia). Unikaj wyrażania swojej treści w formie punktów.
Ogłoszenie prasowe	Napisz ogłoszenie prasowe, które chciałbyś wypuścić, kiedy produkt stanie się dostępny. Dobre ogłoszenia prasowe komunikują w jasny sposób to, co jest warte zakomunikowania, na maksymalnie jednej stronie.
Przegląd w magazynie	Stwórz побieżny fikcyjny przegląd napisany przez osobę, która dokonała analizy Twojego rozwiązania w najbardziej popularnym magazynie związanym z branżą.

W firmie Opinia Całkowita Robert i interesariusze postanawiają opisać ZODC w formie ogłoszenia prasowego. Zaczynają od zidentyfikowania kilku obszarów **wartości dla interesariuszy** (z rysunku 17.3), które powinna dostarczyć ZODC — patrz tabela 17.2.

**TABELA 17.2.** Potencjalne obszary wartości ZODC dla interesariuszy

Obszar	Opis
Redukcja kosztów/ oszczędność czasu	ZODC musi oszczęścić swoim użytkownikom znaczącą ilość czasu podczas wyszukiwania opinii.
Odróżnienie od innych/ zachwycenie klienta	ZODC musi dostarczyć swoim użytkownikom bardzo mile zaskakujące doświadczenie. Muszą oni odnieść wrażenie, że usługa wykonała za nich kawał dobrej roboty, pozwalając im na podjęcie świadomego zakupu.
Spajanie/podnoszenie poprzeczków dla konkurencji	ZODC powinien wytworzyć znaczny chaos wśród konkurencji. Ich bieżące rozwiązania powinny niemal natychmiast wyglądać na przestarzałe w porównaniach. ZODC ustanowi nowy standard dla usług przeglądania opinii, którzy inni będą musieli mozołnie osiągnąć.

W oparciu o te obszary wartości dla interesariuszy Robert i pozostali stworzą następujące ogłoszenie prasowe (wyrażenie wizji):

*Firma Opinia Całkowita ogłosiła w dniu dzisiejszym pomyślne uruchomienie swojej nowej usługi ZmyślnaOpiniaDlaCiebie. Usługa ta udostępnia użytkownikom swojego własnego agenta, którego można wytrenować do przeszukiwania Internetu i identyfikowania obiektywnych i właściwych opinii o produktach lub usługach.*

Dorota Nowak — zagozały użytkownik opinii internetowych — mówi: „Mam swojego osobistego asystenta, który naśladuje moje metody filtrowania opinii dostępnych w sieci. To zadziwiające — uczę go, co lubię, czego unikam, a potem ZmysłnaOpiniaDlaCiebie przedziera się przez Internet i znajduje opinie na temat produktów lub usług i automatycznie usuwa te nieobiektywne lub fałszywe. To, co kiedyś zajmowało mi całą wieczność, teraz odbywa się błyskawicznie. Ta usługa zapewnia niesamowitą oszczędność czasu!”

J. Kowalski, prezes Opinii Całkowitej, powiedział: „Jesteśmy zachwyceni, mogąc zaofferować pierwszą na świecie inteligentną usługę do przeglądania opinii w sieci. Od momentu powstania Internetu ludzie zaczęli organizować się w społeczność sieciową. Niestety każda społeczność może stać się czasem zbyt hałaśliwa i trudno jest oddzielić właściwe opinie od tych złych. Nasza bardzo zmysłna usługa wykonuje żmudną pracę przebijania się przez ogromną masę informacji zawierających opinie, eliminując po drodze podejrzane i zwracając jedynie te użyteczne. Czytamy jedynie opinie, których lekturę rozważylibyśmy po spędzeniu długich godzin na samodzielnym przeczesywaniu Internetu”.

Nowa ZmysłnaOpiniaDlaCiebie jest dostępna za darmo pod adresem  
[www.zmyslnaopinia.dlaciebie.pl](http://www.zmyslnaopinia.dlaciebie.pl).

## Tworzenie rejestru produktu wysokiego poziomu

Kiedy mamy już wizję, jesteśmy gotowi do stworzenia elementów rejestru produktu wysokiego poziomu. Chociaż istnieje wiele sposobów reprezentowania elementów rejestru produktu, ja preferuję historyjki użytkownika (omawiane szczegółowo w rozdziale 5.). Używając terminologii historyjek użytkownika, w trakcie planowania produktu chcę stworzyć eposy — bardzo duże historyjki użytkownika, które odpowiadają planowaniu na poziomie produktu. Te historyjki na poziomie eposów są zgodne z wizją i stanowią następny poziom szczegółowości dla wyższej kadry zarządzającej i zespołu scrumowego.

Historyjki piszą zazwyczaj ci sami ludzie, którzy stworzyli wizję — właściciel produktu, interesariusze i jeśli to możliwe mistrz młyna oraz zespół deweloperski. Ja wyznaję ogólną zasadę, zgodnie z którą wszyscy członkowie mojego zespołu scrumowego są zaangażowani w pisanie tych historyjek. Jeśli jednak tworzenie produktu nie zostało jeszcze zatwierdzone/sfinansowane (o czym wspomniałem wcześniej), pełny zespół scrumowy może być niedostępny podczas tworzenia wizji. W takich przypadkach właściciel produktu może poprosić o przysługę zainteresowane produktem osoby z odpowiednią wiedzą techniczną i zaprosić je do pomocy przy pisaniu historyjek.

W firmie Opinia Całkowita ZODC nie została jeszcze zatwierdzona, zatem nie ma zespołu deweloperskiego przypisanego do pracy nad tym produktem. Robert i interesariusze proszą zatem Julię, doświadczonego architekta systemów, aby wzięła udział w burzy mózgów poświęconej tworzeniu historyjek. W trakcie tej sesji tworzą serię początkowych eposów, wśród których znajdują się następujące:

*Jako typowy użytkownik chcę nauczyć ZODC, jakiego typu opinie odrzucać, dzięki czemu usługa będzie wiedzieć, jakich cech użyć podczas odrzucania opinii.*

*Jako typowy użytkownik chcę prostego, podobnego do Google interfejsu do wyszukiwania opinii, dzięki czemu nie będę musiał spędzać zbyt dużo czasu, opisując, jaki wynik chcę osiągnąć.*

Jako typowy użytkownik chcę, aby ZODC monitorowała Internet w poszukiwaniu nowych opinii na temat produktów i usług, którymi jestem zainteresowany, filtrowała je i raportowała, dzięki czemu nie będę musiał nieustannie prosić usługi o wykonywanie tych rzeczy dla mnie.

Jako zaawansowany użytkownik chcę wskazać ZODC, których źródeł ma używać podczas wyszukiwania opinii dla mnie, dzięki czemu nie będę otrzymywać opinii z witryn, których nie lubię lub którym nie ufam.

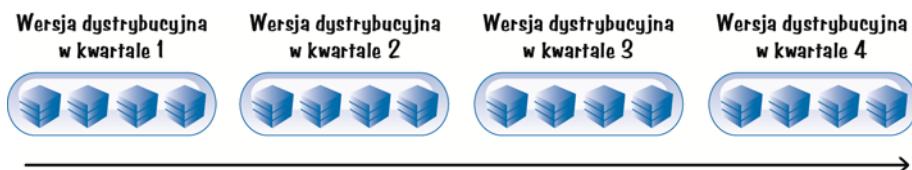
Jako dystrybutor produktu chcę mieć możliwość pokazania zatwierdzonego przez ZODC podsumowania opinii na temat mojego produktu na mojej stronie internetowej, dzięki czemu ludzie będą mogli bezzwłocznie zobaczyć, co rynek myśli na temat mojego produktu, poprzez zaufane źródło, jakim jest ZODC.

## Definiowanie mapy drogowej produktu

Mając początkową wizję i rejestr produktu wysokiego poziomu, możemy zdefiniować naszą początkową mapę drogową — serię wersji dystrybucyjnych, dzięki którym osiągniemy pewną część lub całość wizji naszego produktu. W Scrumie zawsze budujemy w sposób przyrostowy. Kiedy ma to sens, staramy się również wypuszczać nasz produkt w sposób przyrostowy, skupiając się na mniejszych, częstszych wersjach dystrybucyjnych, w których dostarczamy część rozwiązania, idąc w kierunku kompletnego produktu. Mapa drogowa produktu jest początkowym przeglądem tych przyrostowych wdrożeń. Oczywiście nie potrzebujemy mapy drogowej, jeśli planujemy jedynie jedną małą wersję dystrybucyjną.

Częste wypuszczanie wersji nie oznacza konieczności ustanawiania agresywnych terminów końcowych — ich obecność powodowałaby ich częste przekraczanie. Zamiast tego w każdej wersji dystrybucyjnej skupiamy się na małym **minimalnym zestawie cech kwalifikujących do dystrybucji**, co do którego istnieje konsensus w społeczności interesariuszy. Minimalny zestaw cech kwalifikujących do dystrybucji to najmniejszy możliwy zbiór cech obowiązkowych, czyli takich, które zwykle muszą być w wersji dystrybucyjnej, aby spełnić oczekiwania klienta odnośnie do wartości i jakości. Niektórzy określają ten zestaw cech **minimalnym opłacalnym produktem** lub **minimalnym zestawem cech kwalifikującym do sprzedaży**. Choćż możemy zdecydować o dostarczeniu więcej niż tylko minimalnego zestawu cech kwalifikujących do dystrybucji w danej wersji, klienci nie dostrzegliby wystarczającej wartości, gdybyśmy dostarczyli cokolwiek mniejszego. W związku z tym istotne jest, aby zdefiniować taki minimalny zestaw cech.

Aby spełnić wymagania minimalnego zestawu cech kwalifikującego do dystrybucji, niektóre organizacje w celu uproszczenia mapy drogowej produktu stosują strategię ustalonych, okresowych wersji dystrybucyjnych — na przykład co kwartał — patrz rysunek 17.4.



RYSUNEK 17.4. Ustalone, periodyczne wersje dystrybucyjne

Takie podejście ma kilka zalet. Po pierwsze, łatwo można zrozumieć i przekazać wszystkim zainteresowanym (wewnątrz i na zewnątrz organizacji) przewidywany czas pojawiania się wersji dystrybucyjnych. Po drugie, ustanowiony zostaje pewien rytm czy też takt wypuszczania wersji, który pozwala zarządzać zasobami w przewidywalny sposób oraz synchronizować plany pomiędzy niezależnymi grupami.

Używając tej strategii, nadal definiujemy minimalny zestaw cech kwalifikujących do dystrybucji dla każdej wersji. Jeżeli wyprodukowanie minimalnego zestawu cech wymaga mniejszej ilości czasu niż wynikająca z ustalonego czasu dystrybucji, stworzone zostaną dodatkowe cechy o wysokiej wartości. Ustalone, okresowe wersje dystrybucyjne nie zawsze będą możliwe do zastosowania, zwłaszcza jeśli ich produkcja jest sterowana przez zdarzenia zewnętrzne (takie jak konferencje lub ustalony czas dystrybucji innego produktu pod wspólną marką), jednak zalety tej metody sprawiają, że jest ona warta rozważenia.

Każda wersja dystrybucyjna na mapie drogowej powinna posiadać jasno zdefiniowany **cel wersji dystrybucyjnej**, który komunikuje jej przeznaczenie i spodziewany wynik. Cel wersji dystrybucyjnej powstaje przez rozważenie wielu czynników, włączając w to docelowych klientów, problemy architektoniczne na wysokim poziomie, znaczące wydarzenia na rynku itd.

Tworząc mapę drogową produktu, powinniśmy rozważyć klientów oraz ich możliwy podział na segmenty rynku. Mapa powinna wyrażać, jak i kiedy odpowiademy na potrzeby tych różnych segmentów. W przypadku ZODC początkowym rynkiem są indywidualni klienci zainteresowani czytaniem pomocnych opinii przed zakupem produktu lub usługi. Dalsze rozwijanie wizji ZODC dzieli ten rynek na „typowych użytkowników” oraz „użytkowników zaawansowanych” — oczekujących dokładniejszej kontroli nad działaniem ZODC. Zespół decyduje, że początkowym celem będzie użytkownik typowy.

Zespół tworzący wizję ZODC może również przewidzieć przyszłą bazę klientów w postaci dystrybutorów produktów lub usług, którzy mogliby użyć ZODC do dostarczenia obiektywnej historii opinii z całego Internetu na własnej stronie internetowej. Zanim jednak dystrybutorzy dostrzegą wystarczającą wartość, aby zapłacić za usługę, Opinia Całkowita będzie musiała najpierw przekształcić ZODC w zaufaną markę na rynku zbierania i filtrowania opinii.

Tworząc mapę drogową produktu, powinniśmy również rozważyć problemy technologiczne i architektoniczne wysokiego rzędu. Na przykład w przypadku ZODC podstawowym problemem technologicznym jest stwierdzenie możliwych sposobów dostępu do usługi. Zespół decyduje na początek udostępnić usługę poprzez przeglądarkę internetową, ale w dłuższym horyzoncie czasowym przewiduje również aplikacje dedykowane na określone urządzenia mobilne sterowane takimi systemami operacyjnymi jak iOS, Android, a być może również na inne urządzenia nadające się do korzystania z ZODC. W jeszcze dalszej przyszłości zespół chciałby również udostępnić otwarty interfejs programistyczny, z którego mogliby korzystać partnerzy ZODC.

Definiując mapę drogową produktu, powinniśmy wziąć również pod uwagę wszelkie znaczące wydarzenia rynkowe, które mogą mieć wpływ na czas wdrażania naszych przyszłych wersji. Przegląd Całkowity na przykład zawsze uczestniczy w corocznej konferencji Social Media Expo. Robert i interesariusze zgodzili się, że posiadanie wersji dystrybucyjnej na tegorocznej konferencji (mniej więcej za trzy miesiące) byłoby doskonałą okazją do uzyskania komentarzy na temat usługi.

Naszym celem podczas tworzenia mapy drogowej produktu jest uwzględnienie wszelkich czynników, które uważamy za istotne i pomocne w ustaleniu docelowej serii wersji dystrybucyjnych naszego rozwiązania. Pamiętaj jednak, że ta mapa drogowa jest jedynie surowym przybliżeniem jednej

lub kilku najbliższych wersji dystrybucyjnych. Musimy zarezerwować sobie prawo do aktualnienia mapy w miarę dostępności lepszych informacji.

Musimy również rozważyć, jak daleko w przód powinna sięgać nasza mapa drogowa produktu. Chociaż nasza wizja może mieć szerokie horyzonty i być dostatecznie głęboka, aby wymagać wielu lat na pełną realizację, jest mało prawdopodobne, abyśmy chcieli podejmować próbę stworzenia szczegółowej mapy obejmującej całą tę wizję. W Scrumie tworzymy mapę drogową wybiegającą na tyle w przód, na ile uważamy to za rozsądne i pożądane. Jak daleko w przód powinna wybiegać Twoja mapa, zależy będzie od Twoich specyficznych warunków. W wersji minimalnej będzie ona musiał pokrywać czas, jaki chcesz, aby ludzie zarezerwowali na ten projekt.

Robert i interesariusze ZODC uważają, że zrealizowanie w 100% ich wizji zajmie kilka lat, ale sam Robert uważa, że budowanie mapy tak daleko w przód nie byłoby praktyczne, biorąc pod uwagę niski poziom ich potwierzonej wiedzy, a także szybkość, z jaką zmienia się rynek opinii wyrażanych w sieci. Robert i interesariusze zgadzają się na prostą dziewięciomiesięczną mapę drogową, taką jak pokazana na rysunku 17.5.



	Kwartał 3 – Rok 1	Kwartał 4 – Rok 1	Kwartał 1 – Rok 2
Mapa rynku	Pierwsze wdrożenie	Lepsze wyniki Więcej platform	Użytkownicy zaawansowani
Mapa cech/korzyści	Proste uczenie Proste filtrowanie	Złożone uczenie Złożone zapytania	Definiowanie źródłem Nauka przez przykłady
Mapa architektury	100 tysięcy bieżących użytkowników	iOS i Android	Interfejs dla usług sieciowych
Wydarzenia rynkowe	Social Media Expo	Konferencja użytkowników Przeglądu Całkowitego	
Harmonogram wersji dystrybucyjnych	1.0	2.0	3.0

**RYSUNEK 17.5.** Mapa drogowa ZmyślniejOpiniiDlaCiebie

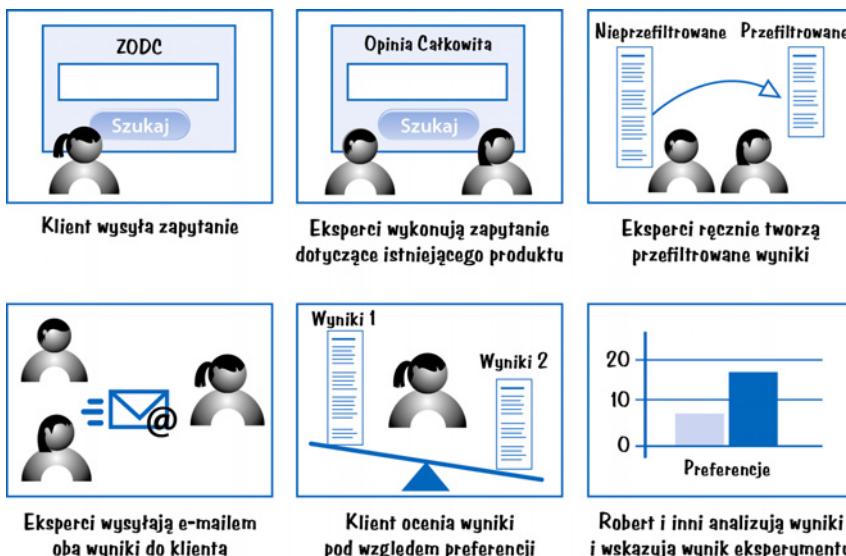
## Inne aktywności

Tworzenie wizji może obejmować również inne rodzaje prac, które osoby zaangażowane wspólnie uznaly za potrzebne w celu osiągnięcia docelowego poziomu ufności. Być może chcemy przeprowadzić jakieś proste badanie rynku pod kątem klientów i użytkowników końcowych. Lub porównać proponowany produkt z innymi już dostępnymi na rynku. Inna możliwość to chęć stworzenia modelu biznesowego, który pomoże nam zdecydować, czy produkt przejdzie przez „filtr ekonomiczny” organizacji.

Niektóre organizacje mogą rozłożyć prace związane z tworzeniem wizji na jeden lub więcej sprintów. W takim przypadku zespół przypisany do tego zadania (nazwijmy go zespołem scrumowym do tworzenia wizji produktów) utrzymuje uporządkowany według priorytetów rejestr pracy

i realizuje ją w krótkich cyklach (być może jednodygodniowych sprintach). Niektóre z tych sprintów mogą być poświęcone pozyskiwaniu wiedzy, jak zostało to opisane w rozdziale 5. Sprinty pozyskiwania wiedzy to między innymi budowanie prototypów, udowadnianie koncepcji zachowania produktu lub kluczowej cechy architektury.

W przypadku ZODC Robert i jego zespół (z ekspertami technicznymi włącznie) decydują się przeprowadzić w trakcie tworzenia wizji jeden sprint pozyskiwania wiedzy. Zanim zainwestują w tworzenie zautomatyzowanego systemu, Robert chce najpierw wykonać prosty test porównujący, aby udowodnić założenie, że przefiltrowane przez ZODC opinie są faktycznie bardziej pomocne dla użytkowników niż opinie nieprzefiltrowane (patrz rysunek 17.6).



**RYSUNEK 17.6.** Tablica historyjek dla sprintu pozyskiwania wiedzy dla ZODC

Podczas sprintu tworzenia wizji zespół buduje atrapę pojedynczej strony internetowej (HTML przypominający wyglądem wyszukiwarkę Google dla ZODC), na której mała, przykładowa grupa użytkowników może wysłać zapytanie dotyczące wybranego przez nich produktu lub usługi i otrzymać w odpowiedzi dwa zestawy wyników. Pierwszy z nich będzie zawierał nieprzefiltrowane wyniki, których zazwyczaj można by się spodziewać w odpowiedzi na zapytanie. Drugi zestaw będzie zawierał wyniki pozbawione „podejrzanych opinii”. Użytkownicy nie zostaną poinformowani, które wyniki zostały przefiltrowane, a które nie.

Użytkownikom biorącym udział w teście mówi się wcześniej, że wyniki zapytania będą gotowe następnego dnia i zostaną przesłane mailem, ponieważ — o czym oni nie wiedzą — Robert nie ma zamiaru budować w tym momencie technologii pozwalającej zautomatyzować generowanie przefiltrowanych wyników. Zamiast tego prosi ekspertów o ręczne przefiltrowanie danych i dostarczenie użytkownikom zarówno przefiltrowanych, jak i nieprzefiltrowanych wyników. Robert i jego zespół następnie przepytują wszystkich użytkowników przykładowej grupy, aby dowiedzieć się, które wyniki były dla nich lepsze i dlaczego.

Celem tego wczesnego testu jest przeprowadzenie podstawowej weryfikacji kluczowego założenia dla usługi ZODC — przekonania, że użytkownicy będą zachwyceni przefiltrowanym przez ZODC zestawem opinii. Jeżeli eksperci nie są w stanie ręcznie wygenerować przekonujących wyników, zdolność Przeglądu Całkowitego do stworzenia systemu eksperckiego, który wniesie jakąś wartość na rynek, stanie pod dużym znakiem zapytania.

Zarząd prosi również Roberta o opisanie innych kluczowych założeń/hipotez pozostających do udowodnienia odnośnie do potencjalnych użytkowników i cech, a także środków umożliwiających zweryfikowanie tych założeń. W celu realizacji tej pracy Robert będzie współpracował z ludźmi od marketingu i innymi. Zamiast przeprowadzania rozległych i czasochłonnych badań rynkowych Robert planuje wykorzystanie pierwszej powstającej wersji jako eksperymentalnego narzędzia do sprawdzenia, co ludzie faktycznie myślą o ZODC i czego tak naprawdę potrzebują pod względem dostarczanych cech.

## Przeprowadzanie wizji w sposób ekonomiczny

Przeprowadzanie wizji powinno odbyć się z zachowaniem zasad ekonomii. Aktywność ta powinna być traktowana jak inwestycja w pozyskanie wiedzy niezbędnej dla zarządu do podjęcia świadomej decyzji odnośnie do finansowania prac zmierzających do stworzenia produktu w oparciu o przedstawioną ideę. Jeżeli zbytnio ograniczymy tworzenie wizji, może okazać się, że jesteśmy nieprzygotowani do przeprowadzenia pierwszego sprintu tworzenia wartości dla klienta. Z drugiej strony, zbyt długotrwałe tworzenie wizji może doprowadzić do powstania dużego inwentarza artefaktów planowania produktu, które trzeba będzie modyfikować lub porzucić, kiedy zaczniemy zdobywać wiedzę potwierdzoną.

W wielu organizacjach praca związana z tworzeniem wizji nosi nazwę **czarterowania projektu, powstawania projektu lub inicjalizacji projektu**. W pewnych organizacjach proces czarterowania jest częścią mechanizmu nadzoru nad fazami projektu. W tym kontekście czarterowanie bardzo często sprowadza się do ciężkiego, ceremonialnego procesu sterowanego planem, bazującego na przewidzianych danych. Te szczegółowe, ale wciąż czekające na potwierdzenie dane tworzą plany niepewne, dające jedynie iluzję komfortu podczas podejmowania decyzji o finansowaniu lub jego wstrzymaniu.

Dodatkowo podejście z dużym nakładem prac na początku nie współgra ze zwinnym procesem deweloperskim w Scrumie, w którym planowanie jest rozłożone w czasie. Tę różnicę można porównać do powiedzenia „Możesz tworzyć oprogramowanie przy użyciu Scruma, ale zanim zatwierdzimy prace deweloperskie, będziemy potrzebować takich samych artefaktów, jakich zawsze wymagaliśmy: rozległych, ustalonych z góry wymagań, pełnego budżetu i precyzyjnego harmonogramu”. Przy tego typu braku dopasowania trudno będzie organizacji osiągnąć długotrwałe, dające wysoką wartość korzyści z używania Scruma.

W Scrumie upraszczamy tworzenie wizji do minimum. Wykonujemy jedynie wystarczającą ilość planowania w formie przewidywania i pozyskiwania wiedzy w oparciu o naturę produktu, jego rozmiar i poziom ryzyka. Pozwalamy na to, aby szczegóły pozostałych artefaktów powstawały w samą porę. Naszym celem jest podjęcie dzisiaj najlepszej możliwej decyzji przy użyciu racjonalnych informacji, uzyskanych w sposób oszczędzający finanse i czas. Uznajemy, że to, co naszym zdaniem produkt potrafi, ulegnie zmianie, kiedy faktycznie coś zbudujemy i poddamy ocenie przez klientów i użytkowników.

Znalazłem kilka wytycznych pomocnych przy tworzeniu wizji w sposób uwzględniający ekonomiczne przedsiewzięcia (patrz rysunek 17.7).

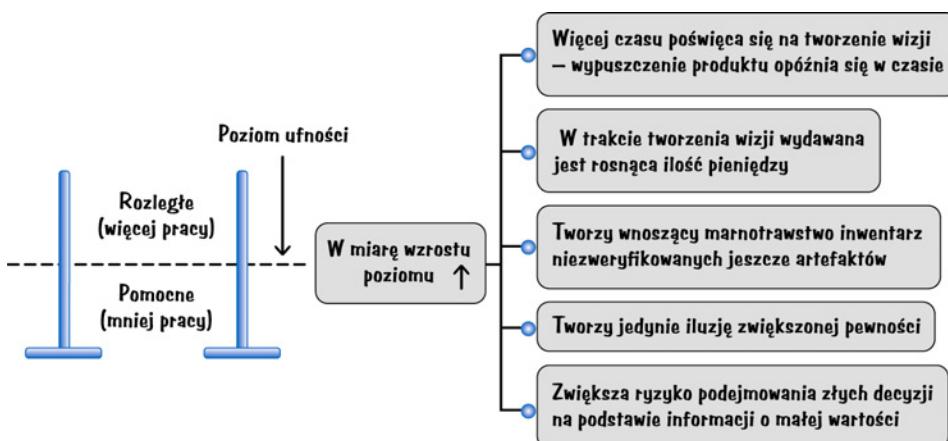


**RYSUNEK 17.7.** Wytyczne ekonomicznego tworzenia wizji produktu

### Celuj w realistyczny poziom ufności

Poziom ufności definiuje minimalny zakres i typ informacji oczekiwany przez decydentów do osiągnięcia przez nich wystarczającej pewności przy podejmowaniu decyzji o finansowaniu lub wstrzymania finansowania następnego etapu. Możesz traktować poziom ufności jak pewną poprzeczkę, nad którą trzeba przeskoczyć, zanim będziemy mogli zakończyć tworzenie wizji i przekazać produkt pod obrady planowania portfela — gdzie zostanie do niego zastosowany filtr ekonomiczny w celu sprawdzenia, czy spełnia kryteria finansowania narzucone przez organizację. Jeśli spełnia, możemy przejść do sprawdzania kluczowych założeń i budowania produktu.

Wysokość tej poprzeczki ma realne konsekwencje ekonomiczne (patrz rysunek 17.8).



**RYSUNEK 17.8.** Konsekwencje zbyt wysokiego ustawienia poprzeczki dla poziomu ufności

Im wyższa poprzeczka, tym więcej potrzebujemy do przeskoczenia nad nią. Dodatkowy czas spędzony na tworzeniu wizji najprawdopodobniej opóźni czas wypuszczenia produktu na rynek, a to opóźnienie będzie miało swój koszt (patrz rozdział 3.). Za czas tworzenia wizji trzeba również zapłacić, zatem im wyższa poprzeczka, tym wyższy koszt jej przeskoczenia. Przewidywanie w większym zakresie tworzy więcej pracy cząstkowej (inwentarza), która z łatwością może zostać zmaranowana, jeśli nadziejde zmiana. Dodatkowo większość pracy cząstkowej nie ma swojego potwierdzenia (na przykład artefakty planowania przewidujące, co może się wydarzyć w przyszłości), zatem dalsze podnoszenie poprzeczki nie wprowadza większej pewności co do słuszności naszych wysiłków. I w końcu więcej pracy może w praktyce zwiększyć ryzyko podjęcia przez nas złej decyzji o rozpoczęciu projektu ze względu na iluzję pewności, jaką daje masa wytworzonych artefaktów planowania. Większa liczba artefaktów nie implikuje większej pewności lub też lepszej decyzji odnośnie do finansowania.

Jak wspomniałem w rozdziale 14., planowanie z góry powinno być pomocne, ale nie rozległe, stąd musimy ustawić nasz próg na poziomie pomocnym, a nie rozległym. Co dokładnie oznaczają poziomy pomocne i rozległe, zależy od specyfiki organizacji i produktu. Niektóre organizacje bez problemu podejmują decyzje w warunkach dużej niepewności, podczas gdy inne przed przystąpieniem do dalszych prac wymagają wysokiego poziomu pewności. Wraz ze wzrostem potrzeby pewności rośnie również wysiłek potrzebny do zebrania danych i nabycia wiedzy potwierzonej. Istnieje pewien faktyczny limit ilości potwierzonej wiedzy, jaką jesteśmy w stanie wytworzyć przed przejęciem do prac deweloperskich, rozpoczęciem budowania czegoś i sprawdzeniem tego przez naszych użytkowników. Podchodź zatem realistycznie do wysokości ustawionego poziomu pewności.

Poziom ufności przy tworzeniu wizji następnej wersji dystrybucyjnej długowiecznego, kluczowego systemu będzie najprawdopodobniej niższy niż w przypadku tworzenia wizji nowego, wysoce innowacyjnego i potencjalnie drogiego produktu.

Przegląd Całkowity odszedł od rozległego planowania w przód na poziomie produktu. Komitet zatwierdzający zgodził się z założeniem, że poziom ufności powinien być „dostatecznie dobry” lub „ledwie wystarczający”, aby rozpocząć prace deweloperskie, w trakcie których firma będzie mogła sprawdzić przyjęte założenia na użytkownikach. Komitet zatwierdzający nie oczekuje pełnego planu projektu z rozbiciem na zadania dla każdej osoby i jej czasem pracy. Zamiast tego chce jasności co do celu następnej porcji prac deweloperskich, a także tego, jak Robert planuje zmierzyć ich wyniki i zdecydować o dalszym, najlepszym kierunku działania.

## Skup się na krótkim horyzoncie

Nie próbuj tworzyć wizji zbyt wielu rzeczy jednocześnie. Skup się głównie na cechach obowiązkowych pierwszego kandydata na wersję dystrybucyjną. Jeżeli planujemy w odniesieniu do szerokiego horyzontu, istnieje duże zagrożenie, że tracimy czas, planując rzeczy, które mogą nigdy nie mieć miejsca. Ponadto jeśli tworzymy nowy, innowacyjny produkt, większość z naszych założeń nie została jeszcze zweryfikowana, zatem istnieje spore prawdopodobieństwo, że kiedy poddamy nasz produkt ocenie bardzo niepewnego środowiska klientów, dowiemy się czegoś istotnego, co zmotywuje nas do adaptacji naszej wizji i planów dotyczących budowanego produktu.

W przypadku ZODC mapa drogowa Roberta sięga dziewięciu miesięcy w przód, ale on sam skupia się na pierwszej wersji dystrybucyjnej. Wszyscy zaangażowani w projekt wiedzą, że dopóki nie będą posiadać zweryfikowanej usługi, której będą mogli używać klienci, zgadują jedynie, jaki

powinien być właściwy zestaw cech. Zatem próba roztaczania wizji zbyt daleko w przód będzie wymagała od nich przyjęcia założeń na podstawie innych założeń, co narusza zasadę posiadania małej ilości krótkoterminowych ważnych założeń.

## Działaj szybko

Tworzenie wizji nie powinno być długim, przeciągającym się procesem. Powinno trwać krótko i przebiegać wydajnie. Im szybciej skończymy, tym szybciej będziemy mogli zacząć budować coś namacalnego, co wykorzystamy do zweryfikowania poprawności naszego sposobu myślenia oraz przyjętych założeń.

Czas poświęcony na tworzenie wizji powinien zostać uwzględniony podczas wyliczania czasu potrzebnego na dostarczenie rozwiązania. Zegar rynkowy zaczyna tykać w chwili pojawiения się okazji biznesowej (stworzenia idei) i nie przestaje do momentu, aż dostarczymy produkt. Niepotrzebnie dłuża aktywność tworzenia wizji opóźni dostarczenie produktu, a koszt tego opóźnienia może okazać się bardzo wysoki. Ekonomia szybkiego postępowania podczas tworzenia wizji jest bardzo przekonująca — Smith i Reinertsen zauważają, że jest to jedna z najmniej kosztownych okazji do zredukowania czasu całego procesu [Smith i Reinertsen, 1998].

Szybkie działanie promuje również przekonanie o konieczności bezzwłocznego podejmowania decyzji na temat produktu. To z kolei pomaga w sprawnym identyfikowaniu niezbędnych zasobów i angażowaniu ich w realizację prac związanych z tworzeniem wizji.

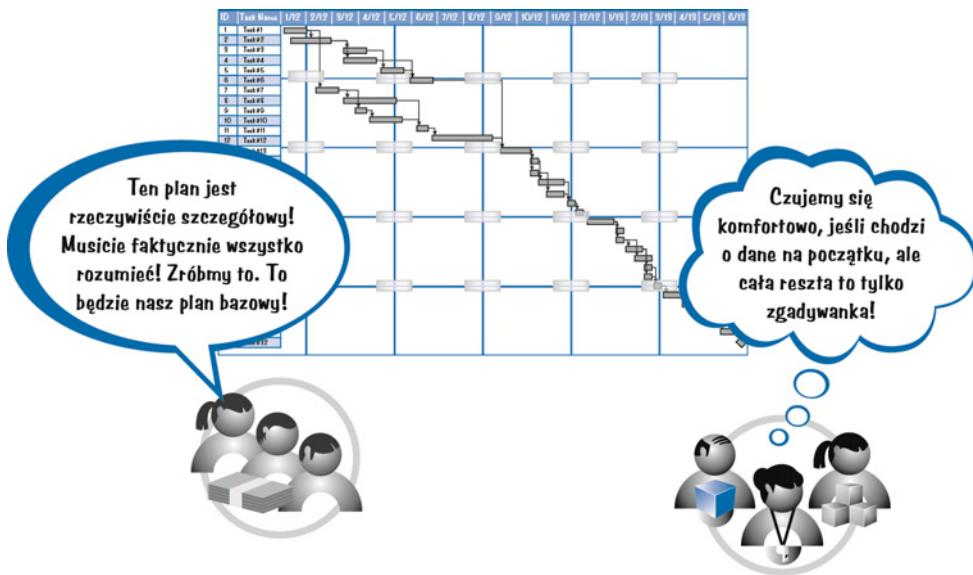
Jednym ze sposobów promowania szybkiego postępowania jest wskazanie oczekiwanej daty zakończenia prac zespołowi zajmującemu się wizją (jako jednej z informacji wejściowych do planowania produktu). Nie każda idea będzie wymagać takiej samej ilości czasu do stworzenia wizji. Jak wspomniałem wcześniej, idea nowego, innowacyjnego produktu może wymagać więcej czasu na zaplanowanie niż wzbogacenie lub uaktualnienie produktu, który istnieje już od dawna. W obu przypadkach nadal chcemy jednak nałożyć pewne racjonalne ograniczenia na pracę związaną z tworzeniem wizji, tak abyśmy mogli przejść szybko do fazy sprawdzania naszych założeń dzięki rzeczywistej informacji zwrotnej.

W przypadku ZODC Robert i inni mają dwa tygodnie na zakończenie prac związanych z wizją. Aby sprostać temu terminowi, Robert będzie musiał zaangażować się w pełnym wymiarze czasu. Eksperci w dziedzinie filtrowania będą musieli poświęcić połowę swojego czasu w drugim tygodniu prac, kiedy wykonywany będzie sprint pozyskiwania wiedzy. Osoba zajmująca się analizą rynku będzie potrzebna przez dwa dni pierwszego tygodnia.

## Płać za wiedzę potwierzoną

Oceniąj aktywności tworzenia wizji z ekonomicznego punktu widzenia w oparciu o to, jak przekładają się one na zdobywanie wiedzy potwierzonej odnośnie do klientów końcowych, docelowego zestawu cech lub samego rozwiązania. Bądź powściągliwy w odniesieniu do aktywności przewidujących, które generują informacje o dużym stopniu niepewności, czyli uważane za poprawne, ale tak naprawdę jeszcze niezweryfikowane przez klientów i użytkowników. Te aktywności „nabywają” informacje o małej wartości i stanowią nie tylko kiepski zwrot z inwestycji, ale również wprowadzają potencjalne marnotrawstwo, ponieważ może się okazać, że po zdobyciu wiedzy potwierzonej będzie my musieli ją porzucić lub na nowo wypracować wiedzę, ponieważ posiadana okazała się błędną.

Poza tym generowanie informacji o niskiej wartości i dużym stopniu niepewności może zaciemnić nasz osąd i doprowadzić do przekonania, że rozumiemy naszą sytuację lepiej, niż jest w rzeczywistości. W wyniku tego podejmujemy istotne decyzje pod wpływem iluzji pewności (patrz rysunek 17.9).



**RYSUNEK 17.9.** Podejmowanie decyzji w warunkach iluzji pewności

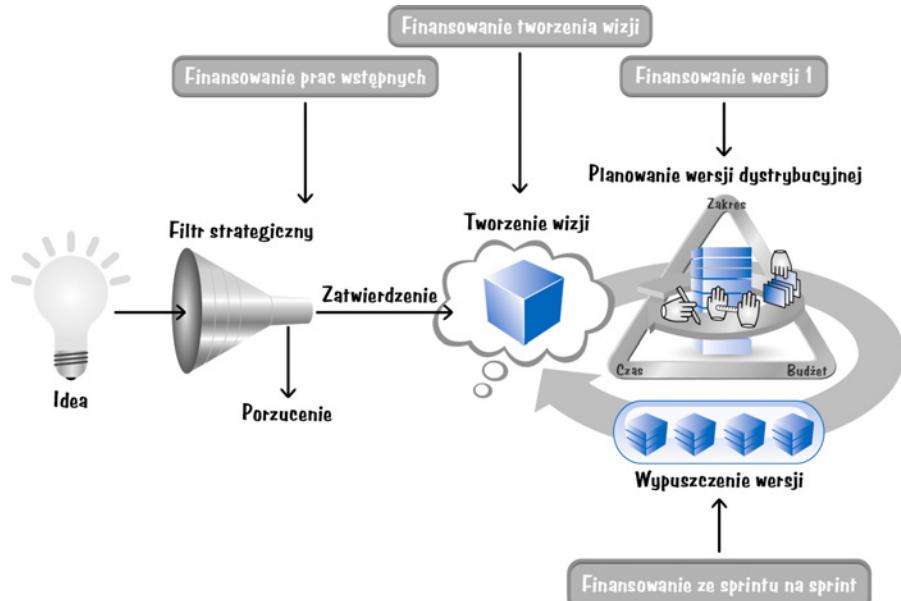
W przypadku ZODC zwartość rejestru produktu oraz mapy drogowej stanowi informację niepewną. Robert uważa, że to, co wyprodukował, jest dobrą próbą zgadnięcia, czego chcą użytkownicy i kiedy mniej więcej będą mogli to dostać. Niemniej jednak zawartość obu tych artefaktów będzie ulegać zmianom wraz z pozyskiwaniem przez zespół wiedzy potwierdzonej w trakcie prac deweloperskich. Dlatego też Robert jest ostrożny pod względem ilości szczegółów wytwarzanych w tym czasie.

Podczas tworzenia wizji ZODC zarząd jest gotowy zapłacić za wiedzę potwierdzoną dotyczącą kluczowego założenia, że użytkownicy będą preferować dane przefiltrowane nad te nieprzefiltrowane. Uważają, że ekonomiczne jest kupienie tej informacji podczas tworzenia wizji, zanim zainwestują znacznie większe środki do zdobycia tej samej wiedzy później. O wiele mniej rozsądne byłoby wydanie znacznej kwoty pieniędzy na zbudowanie pierwszej wersji ZODC i dopiero wtedy przekonanie się, że użytkownicy nie preferują wyników przefiltrowanych.

## Stosuj przyrostowe lub prowizoryczne finansowanie

Zawsze rozważaj finansowanie prac deweloperskich nad produktem w sposób przyrostowy lub prowizoryczny (patrz rysunek 17.10).

Decyzje dotyczące finansowania są (lub przynajmniej powinny być) podejmowane w sposób ciągły i modyfikowane w miarę dostępności lepszych informacji. Przy naszym pierwszym przejściu przez tworzenie wizji nie powinniśmy próbować generować takiej ilości informacji, aby zatwierdzić i sfinansować cały zakres przyszłych prac deweloperskich. Wystarczy informacja pozwalająca sfinansować prace potrzebne do zdobycia kolejnej porcji istotnych danych z rzeczywistego środowiska lub też opinii dotyczących naszych klientów, cech lub samego podejścia do rozwiązania.



**RYSUNEK 17.10.** Finansowanie w sposób przyrostowy lub prowizoryczny

Stosując **finansowanie przyrostowe**, wydalibyśmy pieniądze jedynie na tę pierwszą, małą część prac deweloperskich, a następnie wrócilibyśmy do decyzji o dalszym finansowaniu po uzyskaniu kluczowej wiedzy potwierzonej, za której zdobycie zapłaciliśmy. Stosując finansowanie przyrostowe, możemy zredukować zakres tworzenia wizji oraz czas potrzebny do jej ukończenia.

Pamiętaj również, że sam fakt przydzielenia środków finansowych nie oznacza wcale, że faktycznie je wydamy. Kiedy ze sprintu na sprint zaczyna do nas docierać informacje zwrotne, możemy zdecydować o wykonaniu zwrotu do nowej wizji lub najzwyczajniej przerwać prace deweloperskie nad produktem (więcej szczegółów na ten temat znajdziesz w rozdziale 16.).

W przypadku ZODC polityka Przeglądu Całkowitego polega na elastycznym finansowaniu i unikaniu przydziału dużych środków, a raczej płaceniu w stopniu wystarczającym do potwierdzenia kolejnego ważnego założenia. Zarząd, na podstawie uzyskanych informacji i zdobytej wiedzy, może kontynuować wydawanie przydzielonych już pieniędzy lub też zaprzestać finansowania dalszych prac deweloperskich.

## Ucz się szybko i wykonuj zwroty (czyli stosuj strategię „szybkiej porażki”)

Tworzenie wizji jest częścią cyklu szybkiego zdobywania wiedzy i wykonywania zwrotu. Czasami podejście to jest określane mianem **szybkiej porażki**. Mówiąc najprościej, zarządzamy naszymi zasobami w sposób rozsądny i wydajny, aby jak najsprawniej przeprowadzić tworzenie wizji. Następnie szybko i tanio potwierdzamy naszą wiedzę i założenia w odniesieniu do klientów, cech lub samego rozwiązania, aby sprawdzić, czy wizja i plany produktu spełniają oczekiwania biznesowe. Jeśli okaże się, że tak nie jest, dokonujemy bezzwłocznego zwrotu i wytwarzamy kolejną, bardziej odpowiednią wizję tego produktu lub zwyczajnie „ubijamy” produkt, zatrzymując tym samym wszelkie dalsze wydatki.

Chęć podjęcia świadomej decyzji zmiany kierunku lub przerwania pracy nad produktem w oparciu o racjonalne informacje umożliwia nam znaczne zredukowanie ryzyka finansowego. Zazwyczaj o wiele mniej kosztowne jest szybkie wystartowanie i przekonanie się, że popełniliśmy błąd, niż wydawanie z góry dużej ilości pieniędzy i poświęcanie czasu w celu zapewnienia, iż podejmujemy „właściwą” decyzję tylko po to, aby przekonać się ostatecznie, że byliśmy jednak w błędzie. Strategia szybkiej porażki jest możliwa, tylko jeśli jesteśmy skłonni do „zabicia” produktu już po rozpoczęciu wydawania na niego pieniędzy (zobacz omówienie tematu analizy marginalnej w rozdziale 16.).

W przypadku ZODC celem jest bardzo szybkie (i optymalne pod względem kosztów) uzyskanie informacji zwrotnej przez uruchomienie początkowej wersji umożliwiającej naukę i filtrowanie tak, aby ludzie mogli zacząć korzystać z usługi. Jeżeli na podstawie nadchodzących opinii zespół przekona się, że przefiltrowane wyniki nie są uważane przez docelowych użytkowników za istotnie lepsze od wyników nieprzefiltrowanych, firma może poświęcić trochę więcej czasu na próbę poprawienia algorytmu filtrującego. Niemniej jednak jeśli po zainwestowaniu rozsądnej ilości zasobów firma nadal nie jest w stanie uzyskać mierzalnie lepszego zestawu przefiltrowanych wyników, może to być czas na wykonanie zwrotu i przerwanie prac nad produktem lub rozważenie innego kierunku działań będącego w zgodzie ze zdobytą właśnie wiedzą.

## Zakończenie

W tym rozdziale opisałem w sposób dogłębny tworzenie wizji (planowanie na poziomie produktu). Zilustrowałem to przykładem fikcyjnej firmy tworzącej wizję, rejestr produktu wysokiego poziomu oraz mapę drogową usługi ZmyślnaOpiniaDlaCiebie. Pokazałem również, jak sprint pozyskiwania wiedzy może pomóc w zdobyciu poziomu ufności niezbędnego do zakończenia tworzenia wizji. Następnie przyszła pora na przewodnik tworzenia wizji w sposób ekonomiczny, pozwalający na lepsze zrównoważenie wykonywanego na początku planowania produktu z następującą później scrumową pracą wytwarzania wartości dla klienta.

W rozdziale 18. będę omawiał przejęcie wyników tworzenia wizji i użycie ich podczas planowania wersji dystrybucyjnej.



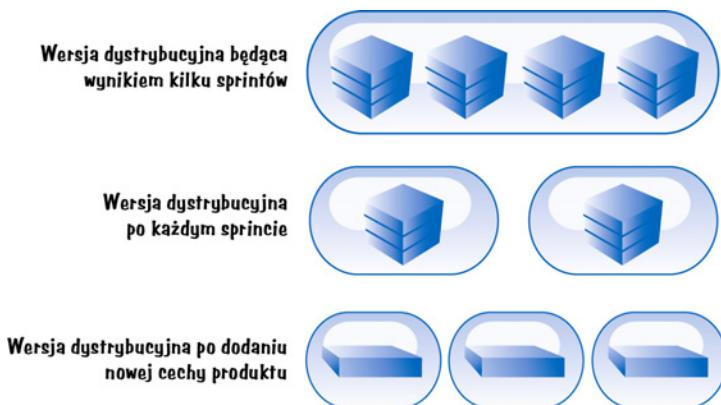
## Rozdział 18

# PLANOWANIE WERSJI DYSTRYBUCYJNEJ (PLANOWANIE DŁUGOTERMINOWE)

Planowanie wersji dystrybucyjnej dotyczy długiego horyzontu czasowego, dzięki czemu można odpowiedzieć na pytania takie jak „Kiedy zostaną zakończone prace?” czy „Jakie nowe cechy mogą znaleźć się w produkcie do końca roku?”. Plan wersji dystrybucyjnej musi równoważyć wartość dla użytkownika przy zachowaniu odpowiedniej jakości z wszelkimi możliwymi ograniczeniami zakresu, planu i budżetu. W tym rozdziale omówię, w jaki sposób planowanie wersji dystrybucyjnej łączy się z procesem scrumowym, a także jak przeprowadzić planowanie w przypadku wersji dystrybucyjnych z ustaloną datą wypuszczenia i ustalonym zakresem zmian.

## Wprowadzenie

Każda organizacja musi posiadać odpowiedni harmonogram dostarczania nowych cech produktu swoim klientom (patrz rysunek 18.1).



**RYSUNEK 18.1.** Różne harmonogramy wypuszczenia wersji dystrybucyjnych

Chociaż wynik wykonania sprintu nadaje się do potencjalnego wypuszczenia na rynek, wiele organizacji decyduje się nie dostarczać nowych cech produktu po każdym sprincie. Zamiast tego łączą one wyniki wielu sprintów w jedną większą całość.

Inne organizacje dopasowują okresy trwania sprintu do harmonogramu wypuszczania nowych wersji produktu. W takich przypadkach pod koniec każdego sprintu powstaje wersja produktu nadająca się do potencjalnego wypuszczenia.

Niektóre organizacje nie czekają nawet do końca sprintu, lecz wypuszczają nową wersję produktu w momencie ukończenia danej cechy — jest to praktyka określana często mianem **ciąglej dystrybucji**. Takie organizacje dostarczają nową cechę lub poprawkę do niej natychmiast po zakończeniu produkcji, czasami nawet kilka razy w ciągu dnia.

Każda organizacja, niezależnie od przyjętego modelu dystrybucji (po każdym sprincie, co kilka sprintów lub w sposób ciągły), uznaże za przydatne poświęcenie odrobiny czasu na planowanie wysokiego rzędu dla długiego horyzontu czasowego. Tego typu planowanie określam mianem planowania wersji dystrybucyjnej. Jeżeli **planowanie wersji dystrybucyjnej** nie pasuje do specyfiki Twojego procesu, możesz użyć innego określenia, które będzie lepiej odzwierciedlać kontekst. Oto kilka synonimów, o których wiem, że są stosowane w różnych organizacjach:

- Planowanie długoterminowe — sugeruje spojrzenie w przyszłość sięgającą dalej niż pojedynczy sprint.
- Planowanie według kamieni milowych — ponieważ wersje dystrybucyjne zazwyczaj pokrywają się z bardziej znaczącymi kamieniami milowymi, takimi jak ważne spotkania z użytkownikami lub zakończenie minimalnego zestawu cech definiujących produkt nadający się do wypuszczenia na rynek.

Niezależnie od wybranej nazwy planowanie wersji dystrybucyjnej celuje w pewien moment w przyszłość, w którym zbilansowane zostaną tak ważne zmienne jak data wypuszczenia produktu, zakres niezbędnych działań i budżet.

## Czas

Planowanie wersji dystrybucyjnej nie jest wydarzeniem jednorazowym, ale raczej aktywnością powtarzaną w każdym sprincie (patrz rysunek 18.2). Planowanie wersji dystrybucyjnej jest logiczną konsekwencją tworzenia wizji czy też planowania na poziomie produktu.

Celem planowania produktu jest przedstawienie wizji tego, czym powinien on być. Celem planowania wersji dystrybucyjnej jest wskazanie następnego logicznego kroku do osiągnięcia celów założonych w produkcie.

Przed rozpoczęciem prac wiele organizacji używających Scruma przeprowadza wstępne planowanie mające na celu wytworzenie roboczego planu wersji dystrybucyjnej. Zazwyczaj zajmuje to około jednego lub dwóch dni, chociaż dokładny czas trwania różni się w zależności od zakresu i ryzyka danej wersji oraz stopnia zaznajomienia członków zespołu z tworzonym dziełem.

Przy tworzeniu nowego produktu ten wstępny plan wersji dystrybucyjnej nie będzie kompletny i szczególnie precyzyjny. Będziemy go uaktualniać w takcie prac nad wersją, w miarę zdobywania nowej wiedzy. Plany wersji dystrybucyjnej mogą być weryfikowane w ramach przeglądu każdego sprintu lub w ramach przygotowań do przeprowadzenia następnego sprintu.



**RYSUNEK 18.2.** Kiedy powinno mieć miejsce planowanie wersji dystrybucyjnej

## Uczestnicy

Planowanie wersji dystrybucyjnej wymaga współpracy pomiędzy interesariuszami i całym zespołem scrumowym. W pewnym momencie wszystkie te osoby muszą być zaangażowane, ponieważ dla osiągnięcia najlepszego stosunku zawartości produktu do jego jakości niezbędne będzie podejmowanie kompromisów zarówno ze strony biznesowej, jak i technicznej. Stopień zaangażowania każdego z uczestników całego procesu może ulegać zmianom w czasie.

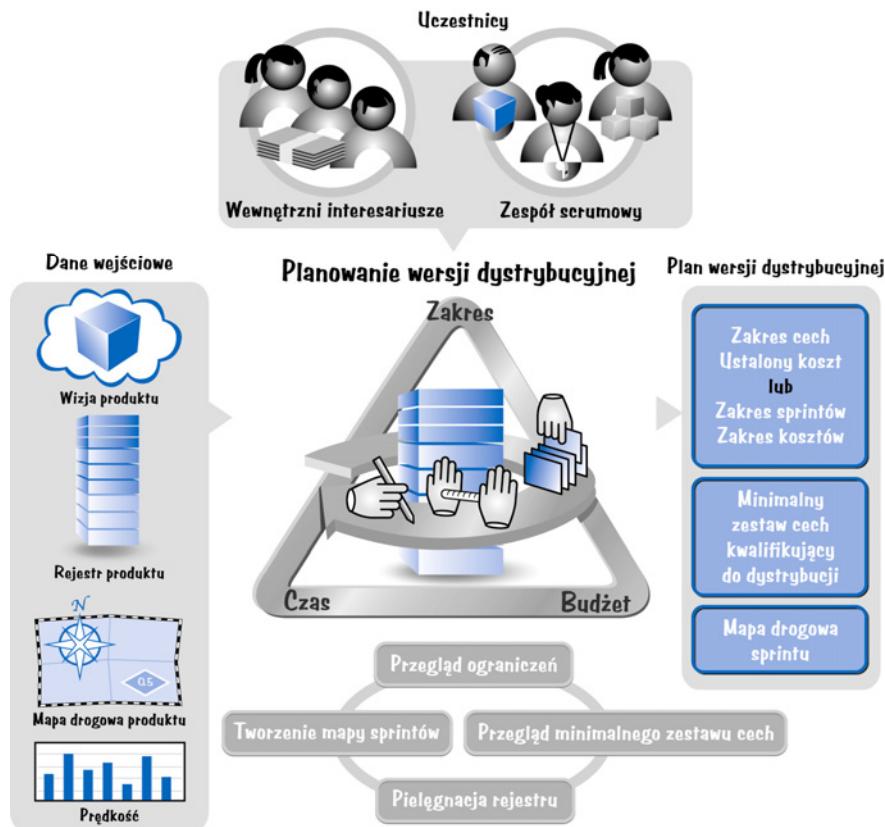
## Proces

Sposób planowania wersji dystrybucyjnej przedstawia rysunek 18.3.

Danymi wejściowymi przy planowaniu wersji dystrybucyjnej są wyniki planowania produktu, czyli wizja produktu, rejestr produktu wysokiego poziomu, a także mapa drogowa. Potrzebujemy również prędkości zespołu lub zespołów, które będą pracować nad wersją produktu. Dla zespołu już istniejącego używamy jego znanej prędkości. Dla nowych zespołów prognozujemy prędkość w trakcie planowania wersji dystrybucyjnej (tak jak zostało to opisane w rozdziale 7.).

Jednym z działań powtarzających się podczas planowania wersji dystrybucyjnej jest potwierdzenie narzuconych ograniczeń odnośnie do zakresu prac, czasu i budżetu, a także sprawdzenie, czy nie należy ich uaktualnić w oparciu o czas, jaki już minął, a także dostępną wiedzę na temat produktu i jego bieżącej wersji dystrybucyjnej.

Kolejnym zadaniem podczas planowania wersji dystrybucyjnej jest pielęgnacja rejestru produktu. Obejmuje ona tworzenie, ocenianie, a także priorytetyzowanie uszczegółowionych historyjek w oparciu o historyjki ogólne. Wspomniane zadania powinny mieć miejsce w kilku różnych momentach:



**RYSUNEK 18.3.** Aktywności planowania wersji dystrybucyjnej

- po przeprowadzeniu planowania produktu, ale przed wstępny planowaniem wersji dystrybucyjnej;
- w ramach aktywności związanych ze wstępny planowaniem wersji dystrybucyjnej;
- w miarę potrzeb podczas każdego sprintu (więcej informacji na temat pielęgnowania rejestru produktu znajdziesz w rozdziale 6.).

Każda wersja dystrybucyjna powinna posiadać dobrze zdefiniowany minimalny zestaw cech kwalifikujący do dystrybucji. Początkowa lista tych cech mogła być już zdefiniowana podczas przedstawiania wizji produktu.

Podczas planowania wersji dystrybucyjnej przeglądamy zawsze minimalny zestaw cech produktu dla tej wersji, aby upewnić się, iż faktycznie reprezentują one minimalną użyteczną postać produktu z punktu widzenia klienta.

Wiele organizacji podczas planowania wersji dystrybucyjnej tworzy również mapę drogową dla sprintu określającą, w którym sprintie do rejestru mogą zostać dopisane poszczególne historyjki. Mapa drogowa sprintu nie przewiduje wybiegania daleko w przód. Jest ona raczej przydatna w wizualizacji nieodległej przyszłości, aby pomóc nam lepiej zarządzać wewnętrznymi zależnościami i ograniczeniami nałożonymi na zasoby, a także lepiej koordynować działania współpracujących ze sobą zespołów.

Wyniki przeprowadzonego planowania wersji dystrybucyjnej są określane wspólnie mianem „plan wersji dystrybucyjnej”. Plan ten informuje w stopniu odpowiednim do aktualnego stanu produkcji o dacie zakończenia prac, zakresie nowych cech produktu, jakie wprowadzimy, i spodziewanym koszcie. Jego celem jest również jasne zakomunikowanie oczekiwanej minimalnego zestawu cech produktu dla bieżącej wersji dystrybucyjnej. I w końcu plan często pokazuje odwołanie niektórych elementów rejestru produktu na sprinty w ramach danej wersji dystrybucyjnej.

## Ograniczenia wersji dystrybucyjnej

Celem planowania wersji dystrybucyjnej jest wskazanie tego, co stanowi najbardziej wartościową kolejną wersję, a także jaki jest oczekiwany poziom jakości. To, w jaki sposób osiągniemy ten cel, będzie zależeć od tak ważnych zmiennych jak zakres, czas i budżet.

Ograniczenia tego typu (niekoniecznie wszystkie) zostaną najprawdopodobniej ustanowione podczas planowania produktu. W rozdziale 17. wprowadzona została fikcyjna firma Przegląd Całkowity. W tym rozdziale będziemy śledzić jej poczynania po stworzeniu nowego produktu — ZODC — dającego się trenować agenta wyszukiwania opinii internetowych. W mapie drogowej dla ZODC Robert i jego zespół ustalili, że korzystnie byłoby wypuścić pierwszą wersję ZODC na nadchodzącą konferencję Social Media Expo. Stąd ZODC wersja 1 posiada ograniczenie w postaci ustalonej daty wdrożenia — produkt musi być gotowy przed określonym dniem (rozpoczęciem Social Media Expo). Pozostałe ograniczenia (zakresu i budżetu) są elastyczne.

Przykłady możliwych kombinacji ograniczeń elastycznych i sztywnych pokazuje tabela 18.1.

**TABELA 18.1.** Możliwe kombinacje ograniczeń

Typ projektu	Zakres	Czas	Budżet
Wszystko ustalone (niezalecane)	Ustalony	Ustalony	Ustalony
Ustalony zakres i czas (niezalecane)	Ustalony	Ustalony	Elastyczny
Ustalony zakres	Ustalony	Elastyczny	Ustalony (nie do końca)
Ustalony czas	Elastyczny	Ustalony	Ustalony

Przyjrzymy się tym różnym kombinacjom pod kątem ich wpływu na planowanie wersji dystrybucyjnej.

### Wszystko sztywne

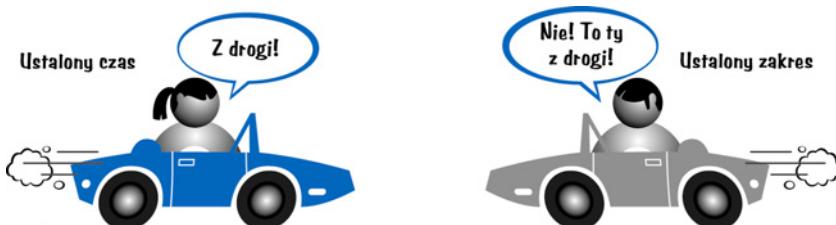
Zgodnie z opisem przedstawionym przeze mnie w rozdziale 3. tradycyjne, sterowane planem podejście do prac deweloperskich zakłada, że wymagania są znane lub mogą być przewidziane z góry, a ich zakres nie zmieni się. Bazując na tych przekonaniach, możemy stworzyć kompletny plan oraz ocenić koszt i harmonogram prac. W tabeli 18.1 podejście to jest opisane wyrażeniem „wszystko ustalone”.

W Scrumie nie wierzymy w możliwość ustalenia wszystkiego w prawidłowy sposób już na samym początku. Stąd wnioskujemy, że podejście w stylu „wszystko ustalone” również nie zadziała. Przeprowadzając planowanie wersji dystrybucyjnej dla produktu przy użyciu Scruma, wymagamy, aby przynajmniej jedna z tych zmiennych była elastyczna.

## Ustalony zakres i czas

Jednym z podejść jest ustalenie zarówno zakresu, jak i czasu oraz pozwolenie na elastyczny budżet. Ta metoda ma wiele wad. Po pierwsze, w wielu organizacjach zwiększanie budżetu po rozpoczęciu prac deweloperskich nie jest możliwe lub prawdopodobne. Po drugie, z mojego doświadczenia wynika, że w takim podejściu usztywniamy dwie zmienne, które bardzo trudno jest przewidzieć. W praktyce nawet jeśli rozpoczęliśmy działania z przekonaniem, że pracujemy nad wersją o ustalonym zakresie i czasie, jedna z tych zmiennych nie utrzyma swojej wartości.

Weźmy za przykład problem roku dwutysięcznego (Y2K). Wiele organizacji starających się sprostać temu zagrożeniu posiadało znaną liczbę aplikacji, które wymagały uaktualnienia przed 31 grudnia 1999 roku. Sporo z nich zdecydowało się ustalić zakres i czas, umożliwiając jednocześnie dostosowywanie budżetu. Ostatecznie jednak wszystkie przekonały się, że niezależnie od tego, jak bardzo zwiększą swoje budżety, nie dadzą rady ukończyć całej potrzebnej pracy przed sztyno ustanowionym terminem 31 grudnia. Daty nie dały się przesunąć, więc przesunięciu uległ zakres. Pod tym względem czas i zakres nieustannie grają ze sobą w cykora! (Patrz rysunek 18.4).



**RYSUNEK 18.4.** Ustalony czas i zakres grają ze sobą w cykora

W pewnym momencie, kiedy zaczyna brakować czasu, jedna ze zmiennych — czas lub zakres — musi się poddać. Jeżeli żadna nie chce tego zrobić, wynikiem ich „zderzenia” będzie najprawdopodobniej wygenerowanie dużej porcji długu technicznego.

Usztywnianiu zakresu i czasu przy jednoczesnym uelastycznieniu budżetu przyświeca założenie, że zwiększenie zasobów potrzebnych do rozwiązywania danego problemu wpłynie na ilość pracy, jaką uda nam się pomyślnie zrealizować i (lub) zredukuje czas potrzebny na wykonanie tej pracy. Niewątpliwie zdarzają się przypadki w trakcie prac deweloperskich, kiedy taka sytuacja ma miejsce. Możemy na przykład zdecydować o wydaniu dodatkowych pieniędzy, aby przyspieszyć wykonanie danego zadania (być może przez zapłaceniu podwykonawcy premii celem umieszczenia naszej pracy przed pracą dla innych). W tym przypadku wydajemy pieniądze, aby kupić czas.

Kupowanie czasu lub zakresu ma jednak swoje granice. Pracy deweloperskiej bardzo często nie da się skompresować — dodanie większej ilości zasobów lub wydanie większej ilości pieniędzy nie pomoże, a może nawet zaszkodzić. Bardzo obrazowe przykłady takich działań podaje Fred Brooks: „Dziewięć kobiet nie urodzi jednego dziecka po miesiącu ciąży” [Brooks, 1995].

Podczas pracy nad produktem „elastyczny budżet” oznacza często „dodawanie kolejnych ludzi”. Niemniej jednak zgodnie z ostrzeżeniami Brooksa i naszymi własnymi doświadczeniami „dodawanie kolejnych ludzi do spóźnionego projektu softwarowego opóżnia go jeszcze bardziej” [Brooks, 1995]. Są przypadki, kiedy włączenie w prace ludzi o odpowiednich umiejętnościach we wczesnej fazie wersji dystrybucyjnej może pomóc. Robienie tego samego pod sam koniec rzadko kiedy wpłynie na pomyślne zakończenie wersji z ustalonym zakresem i czasem.

W wielu organizacjach rzeczywistość wygląda jednak tak, że elastyczny budżet rzadko przekłada się na przesuwanie kolejnych pracowników do projektu. Zwiększenie nakładów oznacza najczęściej, że ci sami ludzie pracują w nadgodzinach — szczególnie jeśli są to pracownicy etatowi. Intensywna praca w nadgodzinach celem sprostania wymogom ustalonego zakresu i czasu powoduje wypalanie naszej kadry i przeczy scrumowej zasadzie pracy w podtrzymywalnym tempie.

Jeżeli okaże się, że daliśmy się wciągnąć w pracę nad wersją o ustalonym zakresie i czasie, iteracyjne i przyrostowe podejście Scruma do produkcji pozwoli nam szybciej zrozumieć, że jesteśmy w kłopotach, da nam więcej czasu na zbilansowanie ograniczeń zakresu, czasu oraz budżetu i w konsekwencji osiągnięcie pozytywnego wyniku.

Do tej pory pisałem o tym, że wersje, gdzie wszystko jest ustalone lub ustalone są zakres i czas, narzucają zbyt wiele ograniczeń przy pracy nad produktem. Pozostają nam zatem dwie inne, bardziej realistyczne opcje: ustalony zakres lub ustalony czas.

## Ustalony zakres

Model z ustalonym zakresem nadaje się do sytuacji, w których zakres prac jest faktycznie o wiele bardziej istotny niż czas. Jeżeli w tym modelu zabraknie nam czasu, a nie ukończyliśmy jeszcze wszystkich cech, przedłużamy pracę, aby mieć pewność, że spełniłyśmy kryterium minimalnego zestawu cech kwalifikujących do dystrybucji. Nie mówię, że jest to model z ustalonym zakresem i budżetem, ponieważ budżet nie może być sztywny. Jeśli damy zespołowi ludzi więcej czasu na dokończenie prac, będą oni oczekiwali zapłaty! Innymi słowy, oferując więcej czasu na dokończenie ustalonego zakresu, musimy również zaoferować więcej pieniędzy na zapłacenie za ten czas.

Często scenariusz ustalonego zakresu wynika z ogólnie przesadzonej ilości cech. Lepszym rozwiązaniem może okazać się rozważenie mniejszych, częstszych dystrybucji z ustaloną datą. W organizacjach, w których różne grupy (takie jak zespoły deweloperskie, marketingowe i wsparcia użytkownika) muszą koordynować swoje działania, przesuwanie daty może zakłócić działania innych zespołów. Mimo to w dalszej części rozdziału wyjaśnię, jak zaplanować wersję z ustalonym zakresem w Scrumie, na wypadek gdybyś znalazł się w sytuacji, kiedy ustalony zakres ma większe znaczenie niż czas.

## Ustalony czas

Ostatnim z podejść przedstawionych w tabeli 18.1 jest ustalony czas. Wiele osób, ze mną włącznie, uważa to podejście za najbardziej zbliżone do zasad Scruma. Mówiąc najprościej, możemy ustalić zarówno czas, jak i budżet, ale zakres musi pozostać elastyczny.

Zasada Scruma polegająca na tworzeniu w pierwszej kolejności cech o najwyższym priorytecie powinna łagodzić dyskomfort wynikający z konieczności porzucania cech. Kiedy skończy nam się czas w wersji z ustaloną datą dystrybucji, wszystko, co nie zostało zbudowane, powinno mieć niższy priorytet od rzeczy już zbudowanych. O wiele łatwiej jest podjąć decyzję o wysłaniu oprogramowania do klientów, jeżeli brakujące cechy mają niską wartość. Jeżeli brakuje nam cech o wysokości wartości, naprawdopodobniej, o ile tylko będzie to możliwe, przesuniemy datę wysyłki.

Zasada ta będzie obowiązywać jedynie, jeśli cechy o wysokiej jakości będą faktycznie wykonane zgodnie z ustaloną przez nas definicją ukończenia. Nie chcemy scenariusza, według którego najbardziej oczekiwane cechy są zrealizowane na 75 lub 90 procent, a my musimy decydować się na porzucenie jednej lub kilku z nich, aby móc dokończyć pozostałe na 100%.

Model z ustalonym czasem będzie jeszcze łatwiejszy w realizacji, jeśli będziemy potrafili zdefiniować naprawdę mały minimalny zestaw cech kwalifikujący do dystrybucji. Jeżeli potrafimy w sposób komfortowy dostarczyć minimalny zestaw cech kwalifikujący do dystrybucji w ustalonym czasie, jesteśmy w dobrej formie, ponieważ z definicji pozostałe cechy są jedynie mile widziane.

Wersje dystrybucyjne z ustaloną datą (czasem) pasują idealnie do nacisku, jaki Scrum kładzie na ograniczenia czasowe. Ustalając limit czasu dla wersji dystrybucyjnej, ograniczamy ilość pracy, jaką ludzie mogą wykonać, i zmuszamy ich do podejmowania trudnych decyzji odnośnie do priorytetów, których tak czy inaczej nie da się uniknąć.

## Zmienna jakość

Jeżeli zbytnio ograniczymy zakres, czas i budżet, spowodujemy „uelastycznenie” jakości. W wyniku tego możemy dostarczyć klientowi rozwiązanie niespełniające jego oczekiwania. Elastyczność pod względem jakości może również doprowadzić do powstania dlużu technicznego, który będzie utrudniał w przyszłości modyfikowanie lub rozszerzanie naszego produktu — o czym pisałem w rozdziale 8.

## Uaktualnianie ograniczeń

Istotną częścią ciągłego procesu planowania wersji dystrybucyjnej jest branie pod uwagę naszej bieżącej wiedzy i w oparciu o nią przeglądanie ograniczeń celem sprawdzenia, czy nie należy ich ponownie zrównoważyć. Na przykład: co powinien zrobić Robert i jego zespół w Opinii Całkowej, jeśli zbliża się do terminu końcowego pierwszej wersji ZODC i będzie pewne, że nie uda im się ukończyć minimalnego zestawu cech kwalifikującego do dystrybucji? Ponieważ jest to wersja z ustaloną czasem, pierwszą z brzegu strategią jest porzucenie cech o niskiej wartości. Założymy jednak, że w tym przypadku, aby sprostać wymaganiom czasowym, musieliby oni porzucać cechy obojęzkowe wchodzące w skład minimalnego zestawu kwalifikującego do dystrybucji.

Być może właściwym rozwiązaniem jest zdefiniowanie mniejszego zestawu cech kwalifikujących do dystrybucji. Na przykład początkowa wersja ZODC może skupić się na filtrowaniu opinii o restauracjach jedynie z małej liczby ustalonych źródeł. Robert i jego zespół muszą określić, czy zmniejszenie zakresu spowoduje spadek wartości dla klienta do poziomu nieakceptownego. Jeśli zapadnie decyzja, że Przegląd Całkowy nie może porzucać cech bez znacznej utraty wartości, firma może rozważyć zwiększenie liczby ludzi (budżetu) lub porzucić nadzieję uruchomienia usługi tuż przed Social Media Expo (zmiana czasu).

Tego typu decyzje musimy podejmować nieustannie, a następnie powracać do nich i jeszcze raz rozważać je w trakcie prac deweloperskich.

## Pielęgnacja rejestru produktu

Podstawową aktywnością planowania wersji dystrybucyjnej jest pielęgnowanie rejestru produktu w celu osiągnięcia założonych celów pod względem wartości i jakości. W trakcie tworzenia wizji (planowania produktu) tworzymy rejestr produktu wysokiego poziomu (być może zawierający historyjki w postaci eposów), a następnie używamy go do zdefiniowania minimalnego zestawu cech kwalifikujących do każdej dystrybucji. Wiele z tych elementów rejestru jest zbyt dużych, aby być przydatnymi podczas planowania wersji dystrybucyjnej.

Na przykład podczas tworzenia wizji ZODC Robert przedstawił przybliżoną ideę tego, które cechy wysokiego poziomu będą dostępne na Social Media Expo. Wyobraźmy sobie, że jego mapa drogowa wskazuje skupienie się wersji 1.0 na cechach „podstawowej nauki” i „podstawowego filtrowania” odpowiadających następującym elementom rejestrów produktu:

*Jako typowy użytkownik chcę nauczyć ZODC, jakiego typu opinie odrzucać, dzięki czemu usługa będzie wiedzieć, jakich cech użyć podczas odrzucania opinii.*

*Jako typowy użytkownik chcę prostego, podobnego do Google interfejsu do wyszukiwania opinii, dzięki czemu nie będę musiał spędzać zbyt dużo czasu, opisując, jaki wynik chcę osiągnąć.*

Podczas planowania wersji elementy te okażą się zbyt duże, aby można było nad nimi pracować. W celu ich uszczegółowienia Robert i jego zespół przeprowadzą warsztaty pisania historyjek (patrz rozdział 5.) w ramach spotkania planowania wersji lub jeszcze przed tym spotkaniem. Wynikiem przeprowadzenia warsztatów będzie powstanie znacznej liczby szczegółowych historyjek w rejestrze produktu, na przykład takich:

*Jako typowy użytkownik chcę wskazać ZODC, aby ignorowała opinie zawierające konkretne słowa kluczowe, które moim zdaniem są oznaką stronnictwa, dzięki czemu nie będę musiał oglądać opinii zawierającej te słowa.*

*Jako typowy użytkownik chcę wybrać kategorię produktu lub usługi, dzięki czemu będę mógł pomóc ZODC skupić się wyłącznie na odpowiednich opiniach.*

Kiedy historyjki staną się dostatecznie małe, zespół będzie mógł przypisać im oceny (patrz rozdział 7.), komunikując w ten sposób ogólną ideę odnośnie do kosztów. (Pewna ilość ocen jest niezbędna w trakcie planowania wersji dystrybucyjnej. Również nowe historyjki powstające w trakcie tworzenia wersji będą musiały być oceniane podczas powtarzających się spotkań planowania wersji). Uczestnicy planowania wersji nadadzą następnie priorytety ocenionym historyjkom w oparciu o cel wersji i ograniczenia. Przy każdej kolejnej zmianie priorytetów powinni oni zachować czujność, aby zawsze mieć na uwadze identyfikację wynegocjowanego wcześniej minimalnego zestawu cech kwalifikujących do dystrybucji.

## Definiowanie minimalnego zestawu cech kwalifikującego do dystrybucji

Jak opisałem w rozdziale 17., minimalny zestaw cech kwalifikujący do dystrybucji stanowi najmniejszą możliwą liczbę cech „obowiązkowych”, czyli takich, które zwyczajnie muszą znaleźć się w wersji dystrybucyjnej, jeśli chcemy spełnić oczekiwania klienta odnośnie do wartości i jakości. Istotną częścią planowania wersji dystrybucyjnej jest dokładna ponowna ocena i dostosowanie tego, co stanowi faktyczny zestaw cech kwalifikujących do dystrybucji dla danej wersji. Nieustannie dostosowujemy ten zestaw w miarę otrzymywania szybkiej informacji zwrotnej z naszych sprintów i zdobywania wiedzy potwierzonej.

Problemem, który często zauważam w organizacjach, jest niezdolność do osiągnięcia porozumienia w odniesieniu do tego, co stanowi faktyczny zestaw cech kwalifikujących do dystrybucji. Wielu interesariuszy najzwyczajniej nie jest w stanie się dogadać. Posiadanie słabo zdefiniowanego zestawu takich cech lub zestawu, na który ludzie zgadzają się jedynie z nastawieniem obronno-atakującym, stanowi przeszkodę dla przejrzystego procesu decyzyjnego w trakcie planowania wersji. Założymy na przykład, że kończy nam się czas. Które cechy powinniśmy porzucić? Brak jasności odnośnie do zestawu kwalifikującego do dystrybucji może utrudnić podjęcie takiej decyzji.

W Scrumie to właściciel produktu jest ostatecznie odpowiedzialny za zdefiniowanie zestawu cech kwalifikującego do dystrybucji. Oczywiście może on i powinien dokonać tego wyboru w bliskiej współpracy z odpowiednimi interesariuszami i zespołem scrumowym.

Dla niektórych koncepcja minimalnego zestawu cech kwalifikującego do dystrybucji może być nieintuicyjna — dlaczego w ramach wersji dystrybucyjnej nie próbujemy dostarczyć największego, zamiast najmniejszego, zestawu cech? Prosta odpowiedź jest taka, że największy zestaw cech najprawdopodobniej kosztuje najczęściej, zajmuje najczęściej czasu, wymaga największej ilości pracy i wiąże się z nim największe ryzyko. Z drugiej strony, najmniejszy możliwy zestaw cech powinien kosztować najmniej, zajmować najmniej czasu i mieć najmniejsze ryzyko. Myślenie w sposób minimalistyczny pozwala nam lepiej współgrać z zasadą dostarczania mniejszych i częstszych wersji dystrybucyjnych, jak zostało to opisane w rozdziale 14.

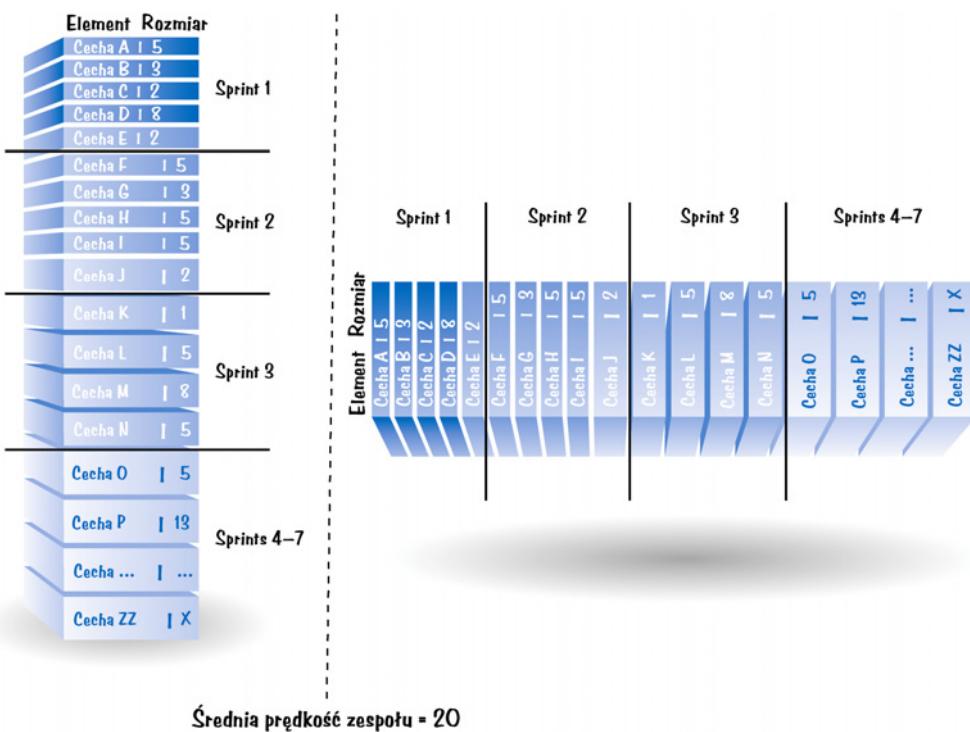
Minimalny zestaw cech kwalifikujący do dystrybucji powinien zostać zdefiniowany, kiedy znane są rozmiary cech (nadane podczas pielęgnacji rejestru produktu). Nie wszyscy zgadzają się z tym twierdzeniem. Niektórzy uważają, że minimalny zestaw cech kwalifikujący do dystrybucji powinien zostać zdefiniowany niezależnie od kosztów — inaczej mówiąc, zestaw ten powinien być zbudowany z minimalnych cech zdatnych do dystrybucji, które razem przekroczą poziom wartości dla użytkownika w danej wersji dystrybucyjnej (niezależnie od kosztów tych cech). Początkowy zestaw cech kwalifikujący do dystrybucji może zostać przewidziany bez danych na temat kosztów, ale ponieważ wszystkie nasze decyzje w ramach planowania wersji dystrybucyjnej powinny być podejmowane z uwzględnieniem ekonomii, znajomość przyszłych kosztów stanowi kluczowe kryterium ekonomicznej wiarygodności minimalnego zestawu cech kwalifikującego do dystrybucji. Jeżeli uznamy, że nasz minimalny zestaw cech jest nieakceptowalny ekonomicznie, być może będzie to sygnał do wykonania zwrotu.

## Tworzenie mapy sprintów (umieszczanie elementów rejestru produktu w slotach)

W każdym sprincie zespół pracuje nad zbiorem elementów rejestru produktu. Zespół i właściciel produktu wstrzymują się z decyzją, nad którymi elementami rejestru pracować w danym sprincie do momentu planowania sprintu. Czy to oznacza, że nie powinniśmy zajmować się przydzielaniem elementów rejestru produktu do poszczególnych sprintów w trakcie planowania sprintów?

Absolutnie nie! Niektóre zespoły uważają, że szybkie, wczesne utworzenie mapy (lub umieszczenie w slotach) elementów rejestru produktu, które będą wkrótce produkowane, jest pomocne. Na przykład utworzenie mapy dla kilku sprintów w wielzespołowym środowisku może pomóc zespołom lepiej skoordynować swoją pracę.

Utworzenie mapy wymaga posiadania odpowiednio uszczegółowanego, ocenionego i posortowanego pod względem priorytetów rejestru produktu. Na podstawie prędkości naszego zespołu możemy oszacować elementy rejestru produktu w każdym sprincie, grupując razem elementy, których sumaryczny rozmiar odpowiada średniej prędkości zespołu. Wynik może przypominać lewą stronę rysunku 18.5. Niektórzy wolą przedstawiać mapę sprintów w formie poziomej (prawa strona rysunku 18.5), aby lepiej ukazać linię czasu. Widziałem zespoły, które umieszczały swoją poziomo zorientowaną mapę sprintów nad planem projektu (na przykład w formie wykresu Gantta) opisującym pracę zespołów niescrumowych, aby lepiej zwizualizować wzajemne ułożenie i wskazać punkty styku pomiędzy scrumową i niescrumową pracą deweloperską.



**RYSUNEK 18.5.** Tworzenie mapy sprintów — przypisywanie elementów rejestru produktu do kolejnych sprintów

Jeżeli prace deweloperskie wykonuje pojedynczy zespół scrumowy, mapę możemy stworzyć podczas wstępniego planowania wersji dystrybucyjnej, uzyskując w ten sposób przybliżony obraz kolejności powstawania cech w ramach tej wersji dystrybucyjnej. Aktywność ta może nam również pomóc zreorganizować rejestr produktu poprzez zgrupowanie elementów w sposób bardziej naturalny i wydajny. Możemy również dokonać reorganizacji pracy, aby zapewnić wynik sprintu w formie umożliwiającej nam zdobycie informacji zwrotnej i podjęcie konkretnych działań na tej podstawie.

Produkując przy pomocy wielu zespołów, możemy spróbować przypisać elementy rejestru produktu do przyszłych sprintów, pomagając w ten sposób w zarządzaniu zależnościami między nimi. Jednym z możliwych podejść jest użycie planowania z rozwijaniem w przód (ang. *rolling lookahead planning* [Cohn, 2006]), w którym każdy zespół rozważa elementy rejestru produktu nie tylko na nadchodzący sprint, ale również na dwa (a czasem więcej) kolejne sprinty. Dzięki temu każdy z zespołów scrumowych może sprawdzić, który zespół pracuje nad danymi elementami rejestru i kiedy mniej więcej można spodziewać się realizacji danej rzeczy.

Załóżmy na przykład, że nad ZODC będą pracować trzy zespoły scrumowe. Zespół numer 1 skupi się na całościowym przetwarzaniu zapytań użytkownika. Jego zadaniem będzie umożliwienie użytkownikom wyspecyfikowania zapytania dotyczącego opinii, wykonania go i otrzymania wyników. Zespół numer 2 skupi się na silniku sztucznej inteligencji, który będzie zawierał logikę wskażającą, w jaki sposób analizować i odrzucać opinie. Zespół numer 3 zajmie się podpinaniem do różnych źródeł danych w Internecie w celu pobrania opinii — kandydatów do analizy.

Te trzy zespoły muszą koordynować swoje działania, aby mieć pewność, że wyprodukowany zostanie minimalny zestaw cech nadający się do dystrybucji i będzie on gotowy na Social Media Expo. W takiej sytuacji rozsądne jest, aby wszystkie trzy zespoły brały udział we wspólnym planowaniu wersji dystrybucyjnej.

Podczas wstępniego planowania wersji każdy z zespołów przedstawia swoje założenia odnośnie do tego, kiedy będzie pracować nad swoimi elementami rejestru produktu. W trakcie intensywnej dyskusji ktoś z zespołu numer 1 może powiedzieć: „Myślimy, że będziemy gotowi do wykonania funkcji *ignorowania opinii zawierających określone słowa kluczowe* w sprintie numer 2. Niemniej jednak będziemy potrzebować zespołu numer 3, aby móc odbierać dane przynajmniej z jednego źródła internetowego, zanim wystartuje ten sprint lub zaraz na jego początku”. Członkowie zespołu numer 3 mogą przeanalizować swoją mapę sprintów, aby sprawdzić, czy planują mieć przynajmniej jedno aktywne źródło w sprintie numer 2. Jeżeli nie, oba zespoły mogą przedyskutować wzajemną zależność i sprawdzić, jakie modyfikacje muszą poczynić po swoich stronach.

Jeżeli zdecydujemy się na utworzenie pewnej wczesnej mapy sprintów, musimy zdawać sobie sprawę, że mapa ta będzie ewoluować w trakcie powstawania wersji. Tak czy inaczej decyzja o tym, nad którymi cechami każdy z zespołów będzie pracował w danym sprintie, jest podejmowana w ostatnim możliwym momencie — w trakcie planowania nadchodzącego sprintu.

Niektóre organizacje (szczególnie wytwarzające oprogramowanie tylko w jednym zespole) preferują zupełnie odmienne podejście — przypisywanie minimalnej lub wręcz zerowej ilości elementów rejestru do kolejnych sprintów. Zespoły w tego typu firmach wychodzą z założenia, że wysiłek potrzebny na stworzenie mapy jest nieuzasadniony z punktu widzenia wartości, jaką dostarcza. Dla tych zespołów spotkanie wstępniego planowania wersji produktu nie przewiduje kroku tworzenia mapy, a jeśli nawet, to na aktywność tę przeznaczana jest mała ilość czasu.

## Planowanie wersji z ustaloną datą

Jak wspomniałem wcześniej, wiele organizacji stosujących Scrum woli wersje dystrybucyjne z ustaloną datą. Kroki potrzebne do przeprowadzenia planowania wersji z ustaloną datą dystrybucji przedstawione zostały w tabeli 18.2.

**TABELA 18.2.** Kroki potrzebne do przeprowadzenia planowania wersji z ustaloną datą dystrybucji

Krok	Opis	Uwagi
1	Określ, ile sprintów będzie w tej wersji dystrybucyjnej	Jeżeli każdy sprint będzie miał taką samą długość, jest to proste zadanie kalendarzowe, ponieważ wiesz, kiedy wystartuje pierwszy sprint i kiedy powinieneś dostarczyć gotowy produkt.
2	Dokonaj dostatecznie głębokiej pielęgnacji rejestru przez utworzenie elementów rejestru, nadanie im ocen oraz priorytetów.	Potrzebujemy dostatecznie dużo elementów rejestru produktu, ponieważ staramy się określić, które z nich damy radę dostarczyć na określony dzień.
3	Zmierz lub spróbuj przewidzieć prędkość zespołu jako przedział wartości.	Określ średnią górną i dolną prędkość zespołu (patrz rozdział 7.).
4	Pomnóż wolniejszą prędkość przez liczbę sprintów. Odlicz tę liczbę punktów w rejestrze produktu i narysuj linię.	To jest linia „będziemy mieć”.
5	Pomnóż szybszą prędkość przez liczbę sprintów. Odlicz tę liczbę punktów w rejestrze produktu i narysuj drugą linię.	To jest linia „być może będziemy mieć”.

Jako przykładu użyjmy ZODC. Wersja numer 1 jest powiązana z początkiem konferencji Social Media Expo, która startuje w poniedziałek 26 września. Firma uważa, że posiadanie pierwszej wersji możliwej do pokazania w trakcie tej konferencji będzie doskonałym kamieniem milowym na drodze do realizacji wizji produktu.

Chociaż przed chwilą wyobraźliśmy sobie pracę trzech zespołów nad ZODC, w tym przykładzie wróćmy jednak do początkowego założenia, według którego nad naszym produktem pracować będzie jeden zespół. Ponieważ Robert i pozostali chcą rozpoczęć sprint numer 1 pierwszego tygodnia lipca i skończyć 23 września (w piątek przez rozpoczęciem Expo), mogą łatwo wyliczyć, ile sprintów będą mieli do wykonania w trakcie tej wersji. Zakładając, że sprinty będą miały taką samą długość w całym zakresie czasu, co jest normą w Scrumie, ZODC w wersji 1 będzie się składał z sześciu dwutygodniowych (dziesięciodniowych) sprintów. Wygląd tych sprintów na tle kalendarza pokazuje rysunek 18.6.

Lipiec							Sierpień							Wrzesień							
S	M	T	W	TH	F	S	S	M	T	W	TH	F	S	S	M	T	W	TH	F	S	
Sprint 1							31	1	2	3	4	5	6	4	5	6	7	8	9	10	
	3	4	5	6	7	8	7	8	9	10	11	12	13	11	12	13	14	15	16	17	
	10	11	12	13	14	15	16	14	15	16	17	18	19	20	18	19	20	21	22	23	24
Sprint 2	17	18	19	20	21	22	23	21	22	23	24	25	26	27	25	26	27	28	29	30	31
	24	25	26	27	28	29	30	28	29	30	31				25	26	27	28	29	30	31

**RYSUNEK 18.6.** Kalendarz sprintów dla ZODC, wersja 1.0

Następnie Robert i pozostali ustalają, ile pracy zespół jest w stanie wykonać w ciągu sześciu sprintów. Stosując podejście opisane przeze mnie w rozdziale 7., powiedzmy, że obliczają przedział swojej prędkości na 18 do 22 punktów historyjkowych w każdym sprintie. Stąd w trakcie opracowywania całej wersji zespół powinien być w stanie ukończyć od 108 do 132 punktów historyjkowych pracy.

Teraz muszą określić, jakie historyki będą reprezentowane w tym przedziale punktowym. Kończąc planowanie produktu, Robert i zespół mieli kilka historyjek wysokiego poziomu (eposy i tematy). Jak pisałem wcześniej, zespół ZODC przeprowadził następnie warsztaty pisania historyjek użytkownika, aby utworzyć bardziej szczegółowe elementy rejestru produktu i nadać im oceny. Opierając się na tych ocenach, właściciel produktu z pomocą zespołu deweloperskiego i interesariuszy przypisał im priorytety.

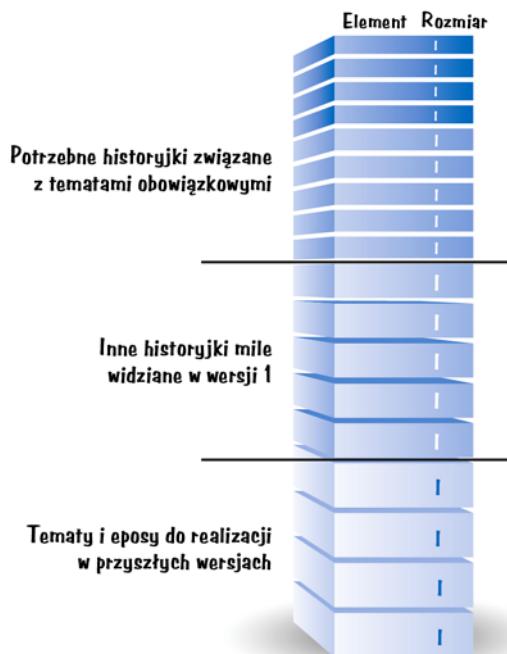
W ramach tego procesu Robert i zespół musieli określić zestaw cech obowiązkowych, reprezentujących minimalny zestaw kwalifikujący do dystrybucji. Moją żelazną zasadą jest wymaganie, aby realizacja minimalnego zestawu cech kwalifikujących do dystrybucji wymagała mniej niż 100% czasu przydzielonego dla danej wersji. Z dwóch powodów wolę, aby była to wartość w okolicach 60% lub 70%.

- Jeżeli skończy nam się czas dla danej wersji dystrybucyjnej, a wszystkie cechy przeznaczone dla tej cechy są elementami obowiązkowymi, to które z nich porzucimy? Zgodnie z definicją, jeśli wszystkie cechy są obowiązkowe, nie powinniśmy porzucać żadnej z nich. Jeśli zdefiniujemy wersję na około 60 – 70% cech obowiązkowych, a pozostały czas przeznaczmy na cechy mile widziane, będziemy mogli porzucić te drugie, jeśli zajdzie potrzeba zmiany zakresu.
- Co zrobić, jeżeli mamy 100% cech obowiązkowych w wersji, a nagle zajdzie konieczność dodania kolejnej cechy obowiązkowej? Innymi słowy, cecha, o której istnieniu nie wiedzieliśmy wcześniej, zostaje odkryta i musi zostać koniecznie wykonana, aby otrzymać opłacalną wersję dystrybucyjną. Jak znajdziemy dla niej miejsce? Jeśli zdefiniowaliśmy wersję w taki sposób, aby posiadała kilka cech mile widzianych, możemy porzucić jedną lub kilka z nich i na ich miejsce wstawić nowo odkrytą cechę obowiązkową.

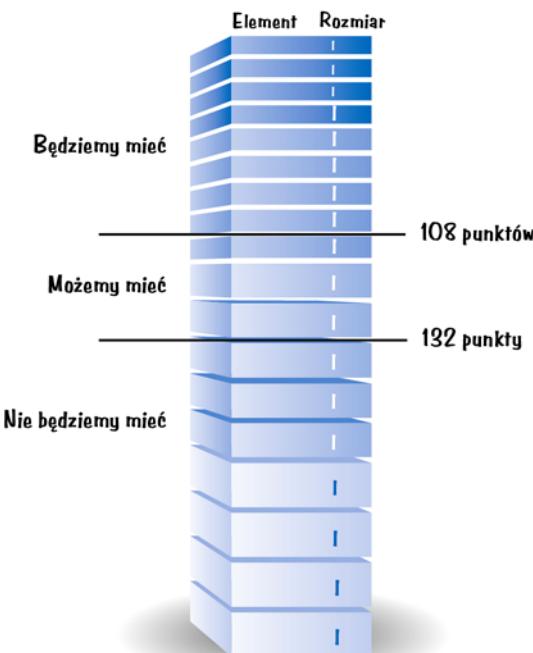
Efektem takich działań będzie rejestr produktu przypominający swoją strukturą ten pokazany na rysunku 18.7. Widać na nim cały rejestr produktu w postaci rozumianej aktualnie przez Roberta i cały zespół (zawiera on również tematy i eposy niezaprojektowane dla wersji bieżącej).

Robert i zespół mogą następnie zastosować wyniki wstępного wyliczenia prędkości, według których zespół ocenił, że jest w stanie ukończyć pracę wartą od 108 do 132 punktów w ciągu sześciu sprintów. Zespół może wizualnie przedstawić miejsce, do którego dotrze, odejmując 108 punktów od szczytu rejestru, a następnie odliczając dalej w dół do 132 punktów (patrz rysunek 18.8).

Jak widzisz, te dwie linie dzielą rejestr produktu na trzy części (cechy, które będziemy mieć, możemy mieć i których nie będziemy mieć). Podejście to ilustruje naszą zdolność do odpowiedzenia w formie przedziału na pytanie „Co dostanę w następnej wersji dystrybucyjnej?”. We wczesnej fazie wersji trudno jest precyzyjnie odpowiedzieć na to pytanie. Odpowiedź w formie przedziału jest dokładna i jednocześnie komunikuje niepewność naszego stwierdzenia — im szerszy przedział, tym mniej jesteśmy pewni.



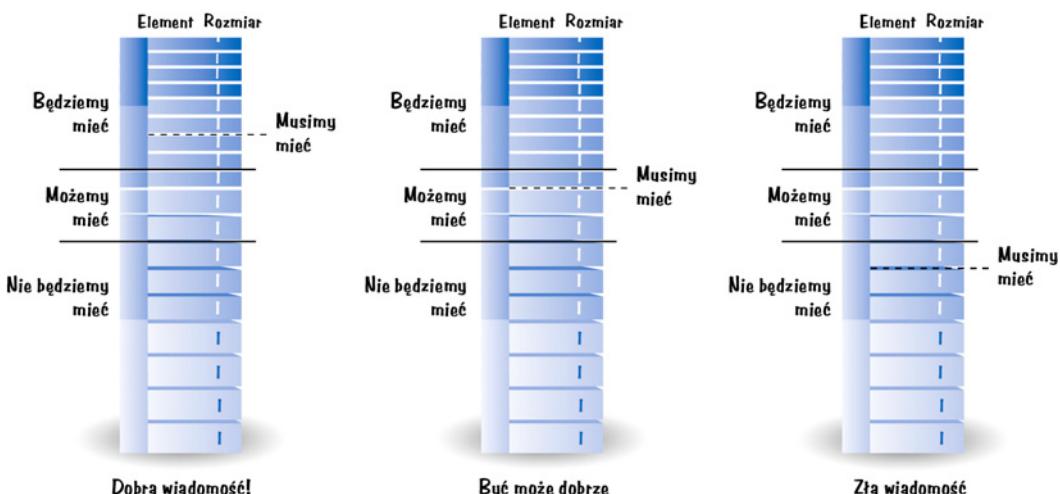
RYSUNEK 18.7. Rejestr produktu gotowy do planowania wersji dystrybucyjnej



RYSUNEK 18.8. Określanie zakresu cech dla wersji dystrybucyjnej z ustaloną datą

Robertowi i jego zespołowi wystarczy naniesienie jedynie linii cech obowiązkowych (rysunek 18.7) na rejestr produktu (rysunek 18.8), aby przekonać się, że są w dobrej formie, jeśli chodzi o plan wersji dystrybucyjnej. Zauważ, że linia cech obowiązkowych oddziela minimalny zestaw cech kwalifikujących do dystrybucji (powyżej linii) od pozostały części rejestru produktu.

Skrajnie lewa część rejestru produktu na rysunku 18.9 komunikuje bardzo pozytywną sytuację. Możesz zinterpretować ją jako „Będziemy mieć nasze cechy obowiązkowe”. Powinniśmy przystąpić do tworzenia wersji dystrybucyjnej.



**RYSUNEK 18.9.** Lokalizacja cech obowiązkowych w odniesieniu do przedziału cech możliwych do dostarczenia

Środkowy rejestr produktu z rysunku 18.9 możemy zinterpretować jako „Będziemy mieć większość naszych cech obowiązkowych, ale może zdarzyć się tak, że nie będziemy mieć ich wszystkich”. Jasno widać, że z tym scenariuszem wiąże się większe ryzyko niż ze scenariuszem poprzednim. Jedna z możliwości to zaakceptowanie ryzyka, że nie uda nam się osiągnąć kompletu cech obowiązkowych i iść dalej. Ponieważ planujemy szybkie pozyskiwanie wiedzy, możemy rozpoczęć prace nad tą wersją dystrybucyjną i wykonać kilka sprintów. W tym momencie możemy ponownie ocenić naszą sytuację i kontynuować pracę nad wersją dystrybucyjną lub przerwać ją (o czym pisałem w poprzednich rozdziałach). Ponadto informacja zwrotna, jaką uzyskamy w oparciu o pracę już wykonaną, może wykazać, że pewne cechy uznawane pierwotnie za część zestawu niezbędnego do wdrożenia nie są tak naprawdę cechami obowiązkowymi, dzięki czemu jesteśmy w lepszej sytuacji, niż nam się mogło wydawać.

Możemy ewentualnie rozważyć ustanowienie nowej, późniejszej daty wdrożenia, którą bardzo łatwo obliczymy, lub skierować większą liczbę ludzi do wysiłku deweloperskiego, aby zwiększyć prędkość (jeśli uważamy, że takie rozwiążanie może pomóc). W tym momencie niektóre organizacje mogą również zdecydować się na zaciągnięcie dłużu technicznego przez przycinanie zadań celem wykonania wszystkich obowiązkowych cech w ustalonym czasie, ale przy pogorszonej jakości. Jeżeli jednak w bieżącej wersji zaciągamy dług, powinien on zostać spłacony w następnej wersji, co spowoduje zmniejszenie dostarczonej wartości.

Skrajnie prawy rejestr produktu na rysunku 18.9 można zinterpretować jako: „Nie będziemy mieć naszych cech obowiązkowych”. Być może nie powinniśmy rozpoczynać tej wersji dystrybucyjnej albo powinniśmy zmienić datę wdrożenia lub rozważyć zwiększenie zasobów. Jeśli w takim scenariuszu zdecydujemy się na zaciągnięcie dłużu technicznego, będzie on zapewne bardzo duży.

Jeżeli mimo wszystko rozpoczęliśmy pracę nad wersją, musimy oczywiście powracać do jej planu w każdym sprintie i uaktualnić go w oparciu o naszą bieżącą wiedzę.

Na przykład pod koniec każdego sprintu posiadamy dodatkowe dane dotyczące prędkości, które dla nowego zespołu nieposiadającego swojej historii prędkości lub dla zespołu wykonującego zupełnie odmienną pracę od tej, do której był przyzwyczajony, będą wpływały na średnią liczbę punktów, jaką zespół ten jest w stanie wykonać w ciągu sprintu. Inna rzecz, której możesz się spodziewać, to zmiana samych elementów w rejestrze produktu. Nowe elementy mogą się ujawniać, inne mogą zostać przesunięte poza bieżącą wersję lub usunięte, jeśli dowiedzieliśmy się, że nie będziemy ich potrzebować teraz lub w ogóle. Chcąc zakomunikować w sposób wizualny nowy plan wersji dystrybucyjnej, moglibyśmy namalować kolejną wersję rejestru z rysunku 18.8.

## Planowanie wersji z ustalonym zakresem

Chociaż wersje z ustaloną datą wdrożenia są bardzo popularne w Scrumie, to co zrobić, jeśli w Twoim produkcie bardziej liczy się zakres niż czas wdrożenia? Co zrobić, jeśli Twój minimalny zestaw cech kwalifikujący do wdrożenia składa się w większości z cech obowiązkowych i jesteś skłonny wprowadzić pewien poślizg w dacie wdrożenia, aby tylko mieć je wszystkie?

Jeżeli jesteś w takiej sytuacji, to czy faktycznie wyizolowałeś zestaw cech obowiązkowych do absolutnego minimum? Od czasu do czasu słyszę zarzuty w stylu: „Ale przecież my implementujemy standard — nie można wypuścić produktu na rynek, obsługując jedynie połowę tego standardu”. Chociaż to twierdzenie może być prawdziwe, w większości standardów na ogół występują części opcjonalne, których nie musimy implementować już teraz (przykładem może być obsługa rodzących się standardów HTML i CSS przez przeglądarki internetowe). W innych przypadkach możemy być w stanie wejść na rynek z niepełną implementacją standardu i poinformować klientów, które jego części obsługujemy, a których nie obsługujemy.

Teza, którą chcę udowodnić, to że jeśli będziemy myśleć w sposób przyrostowy i agresywnie kierować się w stronę prawdziwego minimalnego zestawu cech kwalifikującego do wdrożenia, będziemy mogli zazwyczaj zamienić wersję z ustalonym zakresem w kilka mniejszych wersji z ustaloną datą wdrożenia. Kiedy minimalny zestaw cech kwalifikujący do wdrożenia stanie się mały, dominującą rolę przyjmie zazwyczaj inne ograniczenie (takie jak czas).

Załóżmy, że zawdzieliśmy zestaw cech obowiązkowych do absolutnego minimum i naszym głównym ograniczeniem tej wersji jest czas, w którym będziemy mogli ją dostarczyć. W takim przypadku przeprowadzamy planowanie wersji zgodnie z tabelą 18.3.

Jeżeli budujemy wersję z ustalonym zakresem, musimy znać wszystkie potrzebne cechy już na samym jej początku. Możemy je znać, jeśli budujemy prosty lub znany nam produkt. Jednak w przypadku produktów innowacyjnych wiele z cech ujawni się i będzie ewoluować w trakcie produkcji. Z całą pewnością mamy pewne pojęcie na temat spodziewanych cech na samym początku i nim będziemy posługiwać się w trakcie wstępnego planowania wersji. Niemniej jednak musimy być przygotowani na nieustanne rewidowanie planu wersji dystrybucyjnej w miarę zmiany sposobu pojmowania przez nas cech obowiązkowych.

**TABELA 18.3.** Kolejne kroki planowania wersji z ustalonym zakresem

Krok	Opis	Uwagi
1	Przeprowadź pielęgnację rejestru produktu (przez tworzenie, ocenianie i nadawanie priorytetów), tak aby znalazły się w nim przynajmniej te elementy, które chcielibyśmy mieć w tej wersji.	Ponieważ jest to wersja z ustalonym zakresem, musimy wiedzieć, które elementy rejestru należą do tego zakresu.
2	Określ całkowity rozmiar elementów rejestru, jakie mają zostać dostarczone w tej wersji.	Jeżeli mamy rejestr produktu z ustalonimi rozmiarami historyjek, sumujemy nadane oceny wszystkich elementów, które chcemy mieć w danej wersji dystrybucyjnej.
3	Zmierz lub oceń prędkość zespołu jako przedział.	Określ średnią szybszą i wolniejszą prędkość zespołu.
4	Podziel całkowity rozmiar elementów rejestru przez szybszą prędkość i zaokrągluj wynik do następnej liczby całkowitej.	To wskaże nam, jakiej minimalnej liczby sprintów potrzebujemy, aby dostarczyć żądane cechy.
5	Podziel całkowity rozmiar elementów rejestru przez wolniejszą prędkość i zaokrągluj wynik do następnej liczby całkowitej.	To wskaże nam, jakiej największej liczby sprintów potrzebujemy, aby dostarczyć żądane cechy.

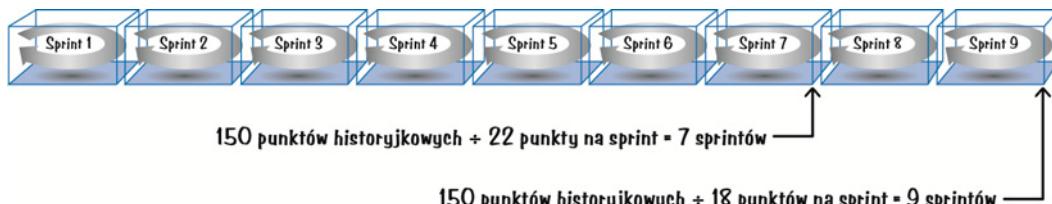
Jeśli przeprowadzamy spotkanie planowania rejestru na początku wersji, musimy najpierw dokonać pielęgnacji rejestru produktu, tak jak robiliśmy to podczas planowania z ustaloną datą dystrybucji. Różnica polega na tym, że podczas planowania wersji z ustaloną datą staramy się mieć mniej niż 100% elementów obowiązkowych, aby móc zareagować na niepewność. W przypadku planowania z ustalonym zakresem chcemy, aby cały zakres wersji składał się z cech obowiązkowych. Naszym celem jest ukończenie wszystkich cech obowiązkowych w odpowiednim czasie. Jeżeli w trakcie pojawi się jakaś cecha obowiązkowa, dodamy ją zwyczajnie do wersji i pchniemy do przodu datę wdrożenia.

Podczas planowania z ustaloną datą wdrożenia wiedzieliśmy dokładnie, ile sprintów mamy do dyspozycji. Podczas planowania z ustalonym zakresem musimy policzyć, ile sprintów będzie potrzebnych, aby dostarczyć ustalony zbiór cech.

Żeby wykonać obliczenia matematyczne, potrzebujemy przedziału prędkości naszego zespołu (podobnie jak miało to miejsce w przypadku planowania z ustaloną datą wdrożenia). Założymy, że prędkość naszego zespołu w sprintach dwutygodniowych waha się pomiędzy 18 a 22 punktami. Chcąc odpowiedzieć na pytanie, kiedy dostaniemy ustalony zestaw cech, sumujemy ich rozmiary, a następnie dzielimy otrzymany wynik przez większą i mniejszą prędkość zespołu. Wynikiem jest przedział sprintów, pomiędzy którymi dostarczone zostaną wszystkie cechy.

Powiedzmy, że w następnej wersji chcemy umieścić cechy o wartości 150 punktów historyjkowych. Jeżeli podzielimy 150 przez 18 (wolniejszą prędkość naszego zespołu) i zaokrąglimy wynik, otrzymamy dziewięć sprintów. Jeżeli podzielimy 150 przez 22 (szybszą prędkość naszego zespołu) i zaokrąglimy, otrzymamy siedem sprintów. Oba wyniki zostały przedstawione w formie graficznej na rysunku 18.10.

Zauważ, że kolejny raz na zadane nam pytanie odpowiedzieliśmy przedziałem. W tym przypadku pytanie brzmi: „Ilu sprintów będącie potrzebować, aby ukończyć wersję dystrybucyjną zawierającą 150 punktów historyjkowych pracy?”. Nasza odpowiedź to od siedmiu do dziewięciu sprintów. Ponieważ są to sprinty dwutygodniowe, naszą odpowiedzią mogłyby również być od 14 do 18 tygodni.



RYSUNEK 18.10. Wyniki planowania ustalonego zakresu

## Obliczanie kosztów

Obliczenie kosztu wersji z ustaloną datą wdrożenia lub ustalonym zakresem jest proste (patrz tabela 18.4).

TABELA 18.4. Obliczanie kosztów wersji dystrybucyjnej

Krok	Opis	Uwagi
1	Określ, kto jest w zespole.	Założ, że skład zespołu nie ulegnie zmianom w trakcie sprintu lub pomiędzy sprintami.
2	Określ długość sprintu.	Założ, że wszystkie sprints będą miały jednakową długość.
3	Na podstawie składu zespołu i długości sprintu określ koszty osobowe wykonania sprintu.	Jeżeli poprzednie założenia są prawdziwe, jest to bardzo proste i może się jedynie odrobić skomplikować, jeśli skład zespołu lub długość sprintu ulegają zmianom.
4a	Dla wersji z ustaloną datą wdrożenia pomnóż liczbę sprintów tej wersji przez koszt każdego sprintu.	Wynikiem będzie stały koszt osobowy dla tej wersji dystrybucyjnej.
4b	Dla wersji z ustalonym zakresem pomnóż większą i mniejszą liczbę sprintów przez koszt pojedynczego sprintu.	Wynikiem jest przedział kosztów osobowych związanych z tą wersją dystrybucyjną. Jedna z wartości reprezentuje mniejszy oczekiwany koszt wersji, a druga większy koszt tej wersji.

Założmy, że skład zespołu przypisanego do prac deweloperskich jest mniej więcej stały. Innymi słowy, nie wyciągamy ludzi z zespołu i nie dokładamy do niego nowych ludzi, a jeśli już zajdzie taka konieczność, staramy się, aby zmiany były niewielkie, a przesuwani ludzie mieli porównywalne zarobki.

Na podstawie tych założeń możemy z łatwością określić koszt przypadający na każdy sprint, ponieważ wiemy, kto jest w zespole oraz jak długo trwają sprints. Jeżeli usuniemy z tej analizy inne koszty (takie jak koszt kapitału), co na ogół ma sens, ponieważ przy produkcji oprogramowania główną rolę odgrywają koszty ludzkie, koszt osobowy będzie dobrym odpowiednikiem całkowitych kosztów sprintu.

Żeby dokończyć obliczenia, musimy znać liczbę sprintów w ramach wersji. W przypadku wersji z ustaloną datą wdrożenia wiemy dokładnie, ile to będzie sprintów, więc mnożymy tę liczbę przez koszt pojedynczego sprintu i otrzymujemy koszt wersji dystrybucyjnej.

W przypadku wersji z ustalonym zakresem mamy przedział sprintów. W naszym poprzednim przykładzie wyliczyliśmy przedział od siedmiu do dziewięciu sprintów, stąd koszt wersji będzie się wałał pomiędzy siedmiokrotnością a dziewięciokrotnością kosztu sprintu. Większość organizacji zdecyduje się ustalić budżet w oparciu o górną wartość tego przedziału, ponieważ ukończenie wersji może ostatecznie zająć dziewięć sprintów. Gdybyśmy ustalili budżet jedynie na siedem sprintów, mogliby nam zabraknąć środków na dokończenie wersji.

Posiadając historyczną wiedzę na temat kosztu punktu historyjkowego, można wyliczyć koszt wersji jeszcze inaczej. Jeżeli dysponujesz danymi wskazującymi, ile punktów historyjkowych zostało zrealizowanych w poprzednim okresie (powiedzmy, w ciągu roku), i podzielisz ten koszt przez całkowity koszt pracy zespołu, poznasz koszt punktu historyjkowego. Zakładając, że ten sam koszt punktu można zastosować do bieżącej wersji dystrybucyjnej, będziesz w stanie ocenić, ile mniej więcej kosztować będzie 150-punktowa wersja (mnożąc 150 przez historyczny koszt jednego punktu), zanim jeszcze przystąpisz do wstępniego planowania wersji.

## Komunikacja

Ważnym aspektem planowania wersji dystrybucyjnej jest komunikowanie postępów. Chociaż można użyć dowolnej metody informowania o postępach, większość zespołów stosuje do tego celu jakąś formę wykresu spalania lub rozpalania. Przyjrzymy się, w jaki sposób komunikować stan wersji dystrybucyjnej z ustalonym zakresem i ustaloną datą dystrybucji.

### Komunikowanie postępów dla wersji z ustalonym zakresem

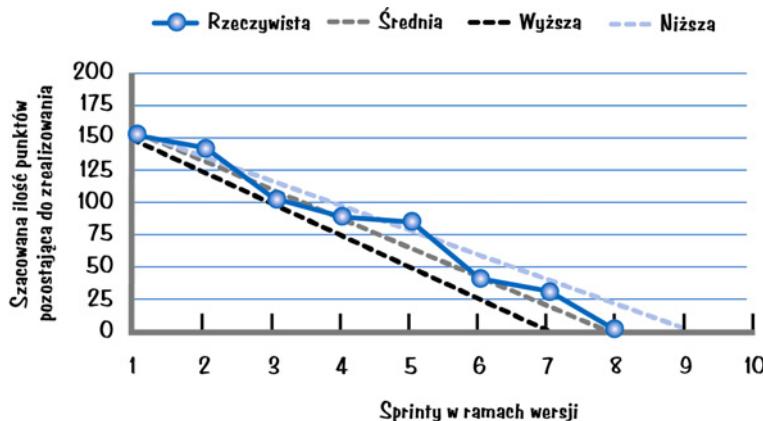
W przypadku wersji z ustalonym zakresem wiemy, jaki rozmiar pracy chcemy zrealizować. Celem jest zakomunikowanie, jak wygląda nasza pozycja względem tego celu.

#### Wykres spalania dla wersji z ustalonym zakresem

Wykres spalania dla wersji z ustalonym zakresem pokazuje całkowitą ilość nieukończonej pracy, pozostającej do zrobienia po każdym sprintie, a potrzebnej do osiągnięcia bieżącego celu wersji. Na tego typu wykresie wartości osi pionowej wyskalowane są w takich samych jednostkach, jakich używamy do oceniania rozmiaru naszych elementów rejestru produktu (zazwyczaj są to punkty historyjkowe lub idealne dni). Oś pozioma reprezentuje sprinty (patrz rysunek 18.11).

Używając przykładu z wcześniejszej części tego rozdziału, na początku prac deweloperskich (pod koniec wstępniego planowania wersji) mamy 150 punktów historyjkowych, co odpowiada wartości w chwili startu sprintu numer 1. Pod koniec każdego sprintu aktualniemy wykres, aby pokazać całkowitą ilość pracy pozostającą do zrobienia. Różnica pomiędzy ilością pracy pozostającą do wykonania na początku sprintu i na jego końcu to prędkość sprintu — reprezentowana na rysunku 18.11 przez linię „rzeczywistą”.

Na wykresie spalania możemy również przedstawić spodziewane wyniki. Wykres z rysunku 18.11 pokazuje trzy linie przewidujące moment ukończenia wersji, z których każda odpowiada przewidywanej prędkości zespołu. Jeżeli zespół jest w stanie pracować ze swoją większą prędkością 22 punktów na sprint, to zakończy prace po siedmiu sprintach. Jeżeli działa ze swoją wolniejszą prędkością 18 punktów, być może będzie potrzebował dziewięciu sprintów. Z kolei pracując ze swoją średnią prędkością 20 punktów, będzie potrzebował ośmiu sprintów.

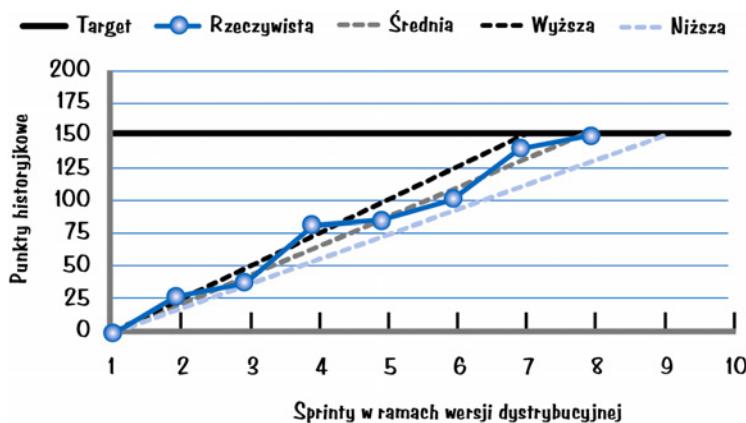


**RYSUNEK 18.11.** Wykres spalania dla wersji z ustalonym zakresem

Istnieje kilka odmian wykresu spalania, niemniej jednak wszystkie są do siebie podobne pod względem wykazywania skumulowanej pracy pozostającej do wykonania w celu zrealizowania celu wersji.

### Wykres rozpalania dla wersji z zakresem

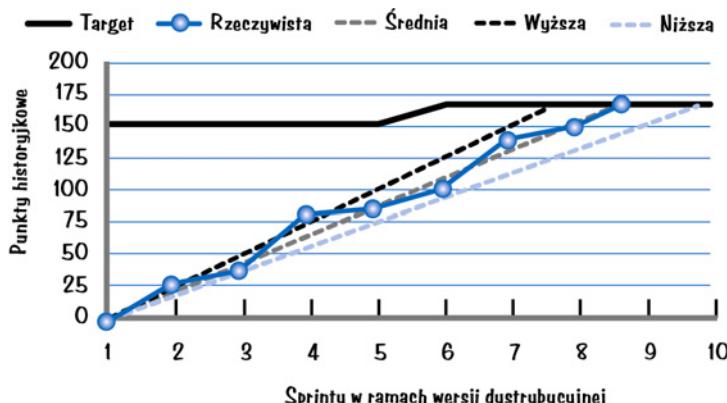
Wykres rozpalania dla wersji z ustalonym zakresem pokazuje całkowitą ilość pracy w wersji jako cel lub linię docelową oraz nasz postęp w kierunku osiągnięcia tego celu (patrz rysunek 18.12). Jednostki osi poziomej i pionowej odpowiadają tym z wykresu spalania.



**RYSUNEK 18.12.** Wykres rozpalania dla wersji z ustalonym zakresem

Na tym wykresie pod koniec każdego sprintu podnosimy całkowitą liczbę zrealizowanych punktów o punkty zrealizowane w tym sprincie. Celem jest rozpalanie do momentu osiągnięcia docelowej liczby punktów dla realizowanej wersji dystrybucyjnej. Podobnie jak w przypadku wykresu spalania wykres ten pokazuje te same trzy linie, wskazując spodziewaną liczbę sprintów potrzebnych do osiągnięcia zamierzzonego celu.

Niektórzy ludzie preferują wykres rozpalania ze względu na łatwy sposób przedstawiania zmiany zakresu wersji. Na przykład: jeśli zwiększymy zakres bieżącej wersji (czyli wersja ta przestanie mieć ustalony zakres!), w sprincie, w którym zakres uległ zmianie, podnosimy linię docelową, wskazując tym samym, że od tego momentu istnieje nowy, wyższy cel (patrz rysunek 18.13).



**RYSUNEK 18.13.** Wykres rozpalania dla wersji ze zmiennym zakresem

Zmianę zakresu można również pokazać na wykresie spalania dla wersji (patrz [Cohn, 2006]).

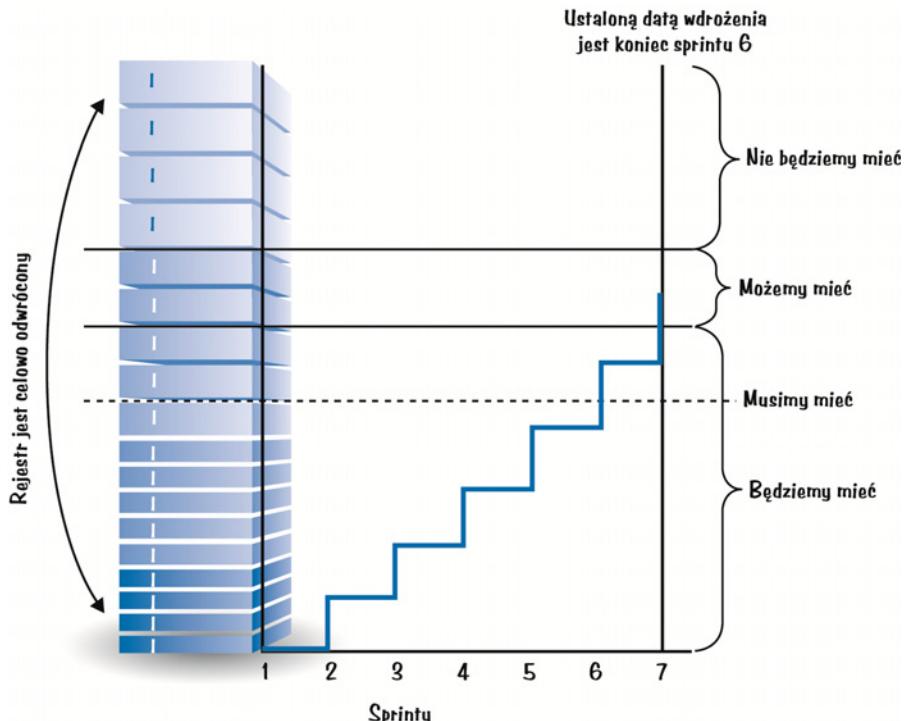
## Komunikowanie postępów dla wersji z ustaloną datą

W przypadku wersji z ustaloną datą dystrybucji znamy liczbę sprintów, zatem naszym celem jest zakomunikowanie zakresu cech, które spodziewamy się ukończyć, oraz postęp w tym kierunku ze sprintu na sprint. Tradycyjne wykresy spalania i rozpalania nie nadają się do planowania z ustaloną datą, ponieważ zakładają one, że wiesz, do jakiego maksymalnego zakresu powinieneś się wypalić lub rozpalić. Pamiętaj, że jest to planowanie z ustaloną datą, zatem wraz z upływem czasu staramy się wyliczyć i zakomunikować zawężający się zakres cech, jakie mogą zostać dostarczone przed tą datą.

Rysunek 18.9 pokazuje możliwy sposób wizualizacji zakresu cech, jaki spodziewamy się osiągnąć w wersji z ustaloną datą. Uaktualniając wykres z rysunku 18.9, pod koniec każdego sprintu otrzymujemy skuteczną metodę komunikowania przewidywanego zakresu cech, jakie uda nam się wykonać w wersji z ustaloną datą dystrybucji. W oparciu o ten wykres będziemy mogli również zrozumieć, jak bardzo prawdopodobne jest, że przed ustaloną datą będziemy mieć do dyspozycji wszystkie cechy obowiązkowe.

Jeżeli chcemy utrzymywać jeden wykres pokazujący historię naszych postępów w kierunku osiągnięcia zakresu końcowego, możemy stworzyć wyspecjalizowany wykres rozpalania w formie pokazanej na rysunku 18.14.

Wykres ten składa się z takich samych elementów jak wykresy na rysunku 18.9, ale na rysunku 18.14 rejestr produktu jest (celowo) odwrócony do góry nogami. W ten sposób element o najwyższej priorytecie znajduje się teraz na samym dole, a nie na szczytce rejestru. Elementy o niższym priorytecie znajdują się wyżej w rejestrze. Odwrócenie rejestru w taki sposób usuwa problem konieczności posiadania wiedzy na temat zakresu elementów rejestru produktu w danej wersji dystrybucyjnej, którą trzeba obowiązkowo mieć w przypadku tradycyjnych wykresów spalania i rozpalania.



**RYSUNEK 18.14.** Wykres rozpalania dla wersji z ustaloną datą dystrybucji (z odwróconym rejestrzem produktu)

Wykres pokazuje przewidywany zakres cech, jaki spodziewamy się otrzymać pod koniec sprintu numer 6 (na początku sprintu numer 7). W każdym sprintie rozpalamy wykres, aby pokazać ukończone w nim cechy. Zatem pod koniec sprintu numer 1 (na początku sprintu numer 2) pojawia się pionowa linia wskazująca, ile cech udało nam się ukończyć w sprintie numer 1. Dzięki takiemu podejściu możemy obserwować nasze postępy w kierunku ukończenia cech obowiązkowych. Dla uproszczenia wykres ten nie zawiera linii trendów, ale można byłoby je z łatwością dodać, aby na podstawie wcześniejszych sprintów wyekstrapolować spodziewany zakres, z jakim zakończymy tę wersję.

## Zakończenie

W tym rozdziale pogłębiłem opis planowania wersji dystrybucyjnej, omawiając zagadnienia takie jak czas, kiedy planowanie wersji ma miejsce, osoby zaangażowane w planowanie, aktywności związane z planowaniem wersji i elementy będące wynikiem stworzonego planu wersji. Podałem również szczegóły dotyczące planowania wersji z ustaloną datą wdrożenia i ustalonym zakresem oraz komunikowania postępów w trakcie każdej z tych wersji.

Ten rozdział kończy część trzecią. W następnym rozdziale omówię kolejny poziom planowania — planowanie sprintu, które umieściłem w części czwartej poświęconej aktywnościom sprintowym.



Część IV

## WYKONYWANIE SPRINTÓW

---



## Rozdział 19

# PLANOWANIE SPRINTU

---

Wersja dystrybucyjna składa się zazwyczaj z wielu sprintów, z których każdy dostarcza pewną wartość dla klienta lub użytkownika. Sprint zaczyna się od planowania, w trakcie którego zespół scrumowy zbiera się, aby ustalić cel sprintu i wskazać, co będzie w stanie dostarczyć w jego trakcie. W tym rozdziale omawiam umiejscowienie planowania sprintu w środowisku Scrum oraz sposób jego przeprowadzania.

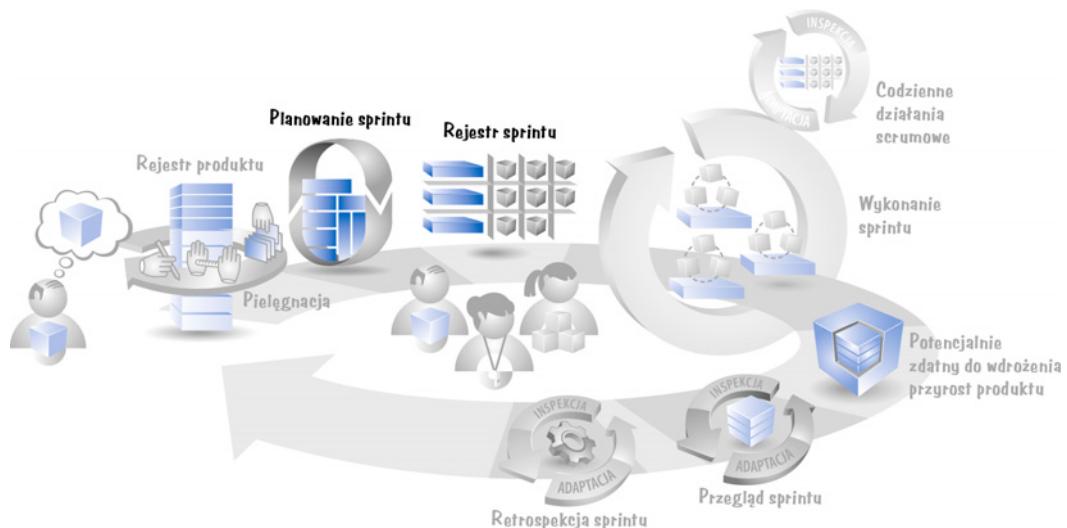
## Wprowadzenie

Rejestr produktu może reprezentować wiele tygodni lub miesięcy pracy, czyli znacznie więcej zadań, niż można zrealizować w ciągu jednego, krótkiego sprintu. Zespół scrumowy przeprowadza planowanie sprintu, aby określić podzbiór najważniejszych elementów rejestru produktu do zrealizowania w nadchodzącym sprincie. Podczas planowania zespół scrumowy uzgadnia cel sprintu, a zespół deweloperski wskazuje konkretne elementy rejestru, które są zgodne z tym celem i które może z dużą dozą prawdopodobieństwa dostarczyć przed końcem sprintu. Aby uzyskać pewność pod względem tego, co może zostać dostarczone, zespół deweloperski planuje, jak zamierza zrealizować wskazane elementy rejestru produktu. Wybrane elementy rejestru oraz plan ich realizacji tworzą rejestr sprintu.

## Czas

Planowanie sprintu jest wykonywaną na bieżąco, powtarzającą się aktywnością, która ma miejsce na początku każdego sprintu, kiedy w oparciu o naszą najlepszą dostępną wiedzę możemy zdecydować, nad czym będziemy pracować w nadchodzącym sprincie (patrz rysunek 19.1).

W przypadku sprintu o długości od dwóch do czterech tygodni planowanie nie powinno zająć więcej niż 4 – 8 godzin.



**RYSUNEK 19.1.** Kiedy rozpoczyna się planowanie sprintu

## Uczestnicy

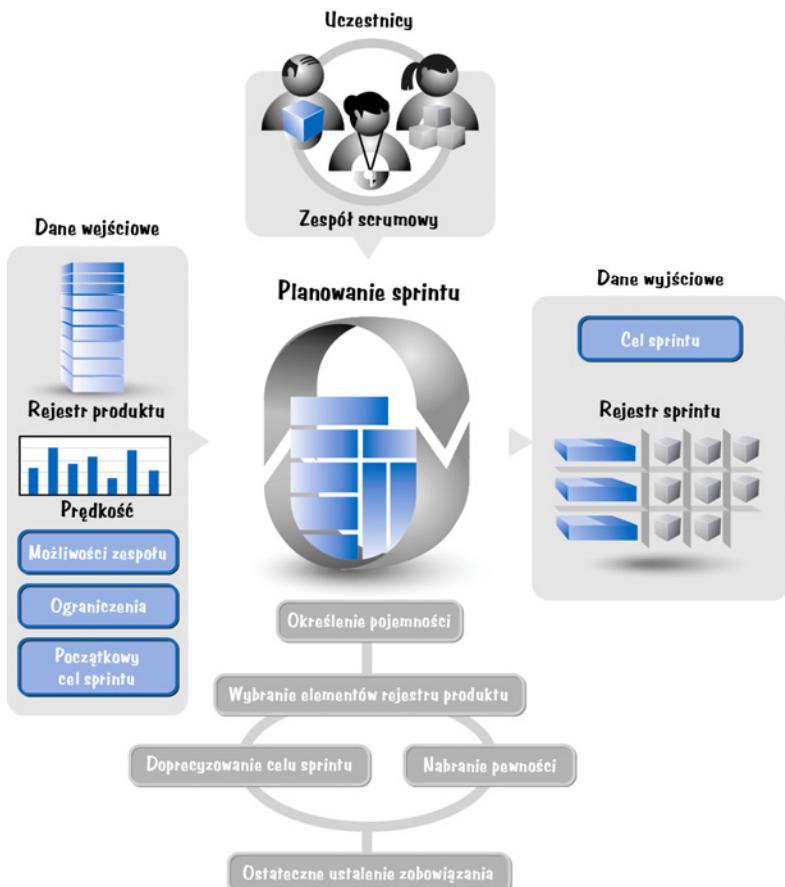
W planowaniu sprintu bierze udział cały zespół scrumowy. Właściciel produktu przedstawia cel sprintu, prezentuje ułożony według priorytetów rejestr produktu i odpowiada na wszelkie pytania, jakie zespół może mieć odnośnie do elementów rejestru produktu. Zespół deweloperski przystępuje do pracy mającej na celu wskazanie elementów, które może wykonać, i zobowiązuje się do ich realizacji pod koniec planowania sprintu. Mistrz młyna, jako trener Scruma, obserwuje aktywność planowania, zadaje badawcze pytania i działa z zamiarem udzielenia pomocy w zapewnieniu pozytywnego wyniku. Ponieważ mistrz młyna nie ma władzy kierowniczej nad zespołem deweloperskim, nie może on zdecydować w jego imieniu, jakiego rodzaju zobowiązanie powinien podjąć. Mistrz młyna może jednak poddać krytyce zobowiązanie zespołu, aby zapewnić, że cel jest odpowiedni i da się zrealizować.

## Proces

Aktywność planowania sprintu została zilustrowana na rysunku 19.2.

Planowanie sprintu bazuje na danych wejściowych, umożliwiających zespołowi określenie wartości, jakie będzie w stanie realistycznie dostarczyć pod koniec iteracji. Dane te zostały opisane w tabeli 19.1.

Pierwszą i najważniejszą informacją podczas planowania sprintu jest poddany wcześniejszej pielęgnacji rejestr produktu, w którym znajdują się najwyżej elementy spełniające ustalone przez zespół scrumowy kryteria definicji gotowości (patrz rozdział 6.). Zazwyczaj oznacza to, że znajdujące się najwyżej elementy mają dobrze zdefiniowane kryteria akceptacji, odpowiedni rozmiar, przyisaną ocenę oraz priorytet.

**RYSUNEK 19.2.** Aktywność planowania sprintu**TABELA 19.1.** Dane wejściowe do planowania sprintu

Dane wejściowe	Opis
Rejestr produktu	Elementy rejestru produktu znajdujące się na szczytce zostały poddane pielęgnacji i są w stanie gotowości.
Prędkość	Historia prędkości zespołu jest wskaźnikiem pozwalającym określić praktyczną ilość pracy, jaką można mu przydzielić na sprint.
Ograniczenia	Zidentyfikowane ograniczenia biznesowe lub techniczne, które mogą wpływać fizycznie na możliwości dostarczenia określonych cech przez zespół.
Zdolności produkcyjne zespołu	Zdolności produkcyjne biorą pod uwagę, jacy pracownicy są w zespole, jakie każdy z nich posiada umiejętności oraz przez jaki czas będzie dostępny w nadchodzącym sprintie.
Początkowy cel sprintu	Jest to cel biznesowy, którego realizacji w trakcie sprintu spodziewa się właściciel produktu.

Zaangażowani właściciele produktów wkraczają na spotkanie planowania produktu z jasnym przeświadczenie wartości, jakich dostarczenia oczekują od zespołu pod koniec sprintu. Mogą mieć na myśli zestaw ściśle określonych elementów rejestru produktu o wysokim priorytetyce. „Chciałbym, aby w tym sprintie zrealizowanych zostało pięć elementów znajdujących się najwyżej w rejestrze produktu” — lub coś bardziej ogólnego: „Chciałbym, aby pod koniec tego sprintu użytkownicy mogli przesyłać proste zapytania oparte na słowach kluczowych”. Znajomość celu sprintu pozwala zespołowi zrównoważyć rywalizujące ze sobą priorytety. Właściciel produktu powinien zakomunikować swój początkowy cel sprintu w sposób dostatecznie powściągliwy, aby nie wywrzeć nacisku na zespół deweloperski i przez to doprowadzić go do zadeklarowania większej ilości pracy, niż jest w stanie realistycznie dostarczyć.

Fakt, iż właściciel produktu wie, czego chce, nie oznacza koniecznie, że zespół deweloperski jest w stanie dostarczyć to w trakcie sprintu. Realistyczne zobowiązanie można osiągnąć jedynie poprzez współpracę (a czasami negocjacje) pomiędzy właścicielem produktu i członkami zespołu deweloperskiego. Uczestnicy planowania sprintu muszą mieć okazję do przeanalizowania i przedyskutowania potencjalnych alternatywnych działań dających wartość, a następnie zdecydowania, jakie rozwiązanie będzie praktyczne po uwzględnieniu możliwości zespołu, jego przewidywanej prędkości i znanych ograniczeń.

W celu uzyskania pewności co do swoich możliwości zespół deweloperski tworzy plan określający, w jaki sposób osiągnięty zostanie cel sprintu. Wybrane elementy rejestru produktu i plan tworzą wspólnie rejestr sprintu (pokazany na rysunku 19.2). Większość zespołów dzieli każdy docelowy element rejestru produktu na zbiór zadań z przypisanymi rozmiarami, które wspólnie stanowią wspomniany plan. Zespoły przyjmujące takie podejście zazwyczaj stosują pomocną zasadę dzielenia zadań na mniejsze, wymagające maksymalnie 8 godzin pracy (z nielicznymi wyjątkami). Na takim poziomie szczegółowości zespół może jasno wskazać, co faktycznie trzeba zrobić i czy wypisane zadania można zrealizować w dostępnym czasie.

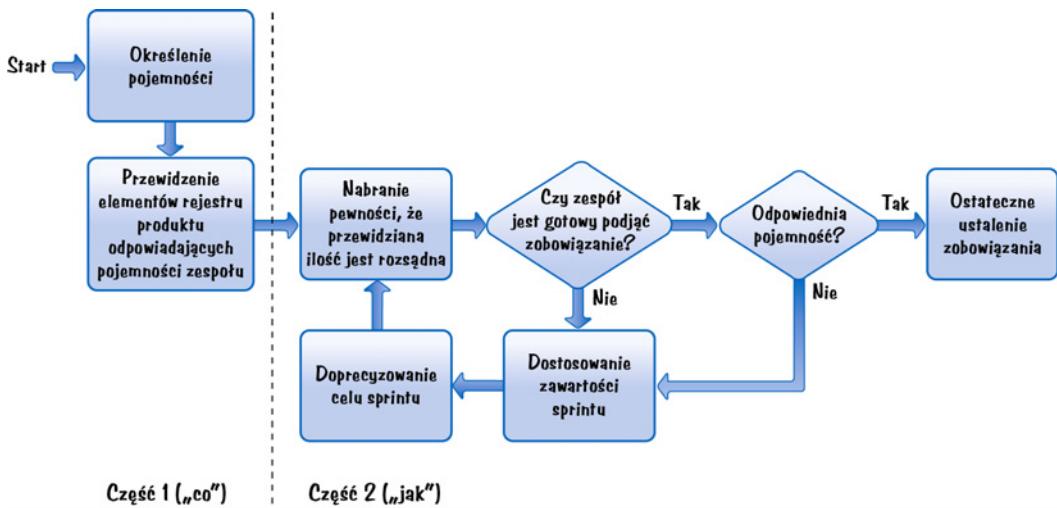
Pod koniec każdego planowania sprintu praca, którą zespół deweloperski zobowiązał się zrealizować, jest komunikowana poprzez ustalony ostatecznie cel i rejestr sprintu.

## Podejścia do planowania sprintu

Opiszę dwa podejścia do planowania sprintu: dwuczęściowe i jednoczęściowe planowanie sprintu.

### Dwuczęściowe planowanie sprintu

Jednym z podejść do planowania sprintu jest podzielenie tej aktywności na dwie części (patrz rysunek 19.3). Podczas pierwszej części (części „co”) zespół deweloperski określa swoją pojemność pracy, a następnie przewiduje, jakie elementy rejestru produktu będzie w stanie dostarczyć przed końcem sprintu. Zatem jeśli zespół uważa, że może wykonać 40 punktów historyjkowych, wskaże pracę o właśnie takiej wartości.



**RYSUNEK 19.3.** Dwuczęściowe podejście do planowania sprintu

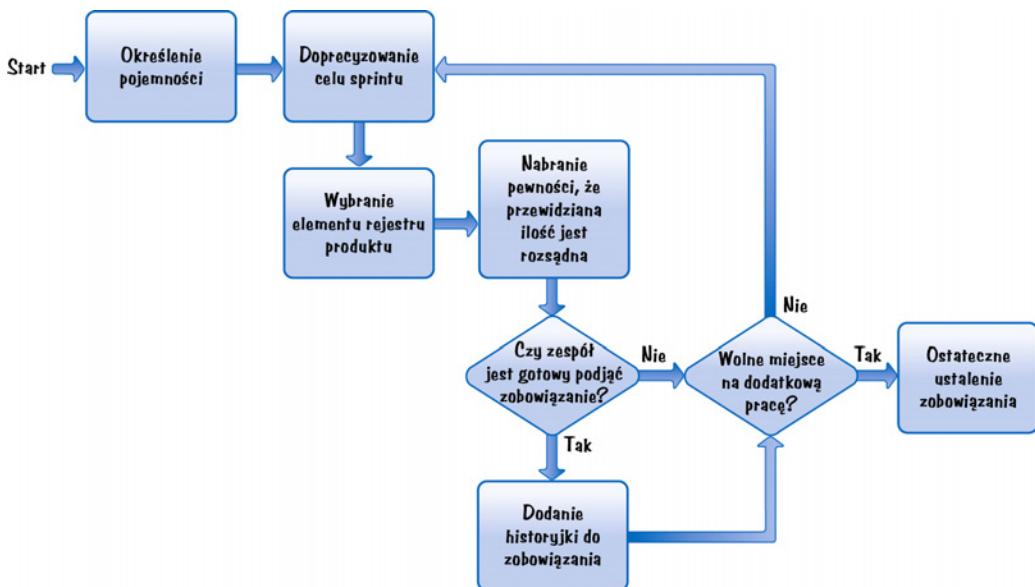
W trakcie drugiej części (części „jak”) zespół nabywa przekonania odnośnie do swojej zdolności do zrealizowania elementów przewidzianych w części pierwszej poprzez stworzenie planu. Większość zespołów tworzy ów plan, rozpisując wybrane elementy rejestru produktu na zadania, a następnie oceniając wysiłek (wyrażony w godzinach) potrzebny na wykonanie każdego z nich. Wyliczony czas potrzebny na zadania zostaje porównany z pojemnością zespołu. W ten sposób można sprawdzić, czy początkowe zobowiązanie było realistyczne.

Jeżeli zespół przekona się, że wybrał zbyt wiele lub zbyt mało, lub też wybrał elementy rejestru, których nie da się realistycznie wykonać w tym samym sprincie ze względu na jedno lub więcej ograniczeń, może dostosować swoją prognozę i być może sam cel sprintu tak, aby pasowały one do dostępnej pojemności i ograniczeń. Kiedy prognoza zespołu zostanie dopasowana do jego pojemności i ograniczeń, nastąpi ostateczne ustalenie zobowiązania i zakończenie spotkania planowania sprintu.

## Jednocięściowe planowanie sprintu

Alternatywnym i najczęściej przeze mnie obserwowanym podejściem do planowania sprintu jest planowanie jednostopniowe, w którym na przemian wybiera się element rejestru, a następnie używa pewność odnośnie do możliwości jego realizacji. Podejście to ilustruje rysunek 19.4.

Stosując to podejście, zespół zaczyna od określenia swojej pojemności. Wyliczona pojemność może wymusić dopecoyzowanie celu sprintu. Następnie zespół wybiera element rejestru produktu i zdobywa pewność, że element ten zmieści się bez problemu w sprincie, biorąc pod uwagę inne elementy włączone już do ewoluującego zobowiązania zespołu. Cały cykl jest następnie powtarzany do momentu, aż zabraknie pojemności do wykonania jakiekolwiek dodatkowej pracy. W tym momencie następuje ustalenie ostatecznego zobowiązania i zakończenie planowania sprintu.



**RYSUNEK 19.4.** Jednocześciowe podejście do planowania sprintu

## Określanie pojemności

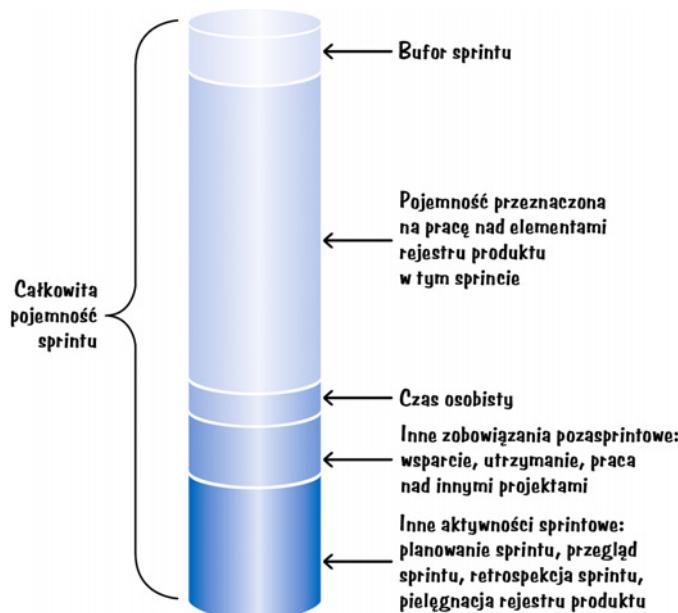
Istotną aktywnością na początku planowania sprintu jest określenie dostępnej pojemności zespołu pozwalającej na realizację pracy w trakcie sprintu. Wiedza na temat pojemności pozwala zespołowi zdeterminować, co jest w stanie dostarczyć w trakcie trwania sprintu.

## Czym jest pojemność?

Rysunek 19.5 przedstawia różnorodne czynniki wpływające na pojemność (zdolność do) pracy nad elementami rejestru produktu w trakcie nadchodzącego sprintu. Wśród nich znajdują się czas potrzebny na inne aktywności scrumowe, zobowiązania pozasprintowe, czas osobisty oraz potrzeba pewnego bufora.

Powiedzmy, że zespół wykonuje dwutygodniowe (dziesięciiodniowe) sprinty. Musimy od razu przyjąć, że w praktyce zespół nie ma dziesięciu dni, które może poświęcić na wykonanie sprintu. Wiemy na przykład, że w trakcie dwutygodniowego sprintu mniej więcej jeden dzień musi zostać zarezerwowany na aktywności planowania, przeglądu i retrospekcji sprintu. Wiemy również, że zespół powinien zarezerwować mniej więcej 10% swojego czasu na asystowanie właścielowi produktu w pielęgnacji rejestru produktu (pisaniu, doprecyzowywaniu, ocenianiu i nadawaniu priorytetów elementom rejestru produktu), aby zapewnić gotowość elementów do realizacji.

Zespół musi również określić, ile czasu powinien zarezerwować dla siebie na pracę poza sprintem — udzielanie wsparcia bieżącemu produktowi, utrzymywanie innego produktu lub pracę niezwiązaną z bieżącym sprintem. Zespół powinien również pamiętać, że nie całe osiem godzin każdego dnia można poświęcić zadaniom sprintowym. Istnieje pewien narzut czasu niezbędny do spełniania roli dobrego obywatela organizacji — uczestniczenia w spotkaniach, odpowiadania na e-maile itd.



**RYSUNEK 19.5.** Pojemność zespołu w sprintie

Oprócz tego zespół powinien wiedzieć, czy ktokolwiek z jego członków ma zarezerwowany czas dla siebie w trakcie sprintu, ponieważ to redukuje jego całkowitą pojemność.

Po usunięciu czasu przeznaczonego na inne aktywności scrumowe, pracę poza sprintem i czas osobisty to, co pozostało, stanowi pojemność zespołu przeznaczoną na realizację elementów rejestru produktu w danym sprintie. Powinniśmy jednak jeszcze zarezerwować pewną ilość tej pojemności na rzeczy, które mogą pójść niezgodnie z planem. Na przykład nie ma możliwości, aby nasze oceny były doskonałe, stąd wybrane elementy mogą okazać się odrobinę większe, niż myśleliśmy. Poza tym zawsze coś może pójść nie tak (i zazwyczaj idzie). Rozsądnym podejściem jest posiadanie pewnego bufora czasu na nieprzewidziane problemy.

Istnieje wiele możliwych sposobów określania właściwego rozmiaru bufora na rzeczy niespodziewane (przykłady znajdziesz w opracowaniu [Cohn, 2006]). W praktyce bufor ten jest ustalany w sposób empiryczny po wykonaniu przez zespół kilku sprintów i nabraniu lepszego zrozumienia co do czasu, jaki powinien zostać zarezerwowany na obsługę niepewności występujących w trakcie prac deweloperskich.

Po zdefiniowaniu bufora zespół może ostatecznie wskazać swoją pojemność przeznaczoną na prace w bieżącym sprintie.

## Pojemność w punktach historyjkowych

Jakiej jednostki miary powinniśmy użyć do pomiaru pojemności? Dwie oczywiste odpowiedzi to: takich samych jednostek jakich używamy w rejestrze produktu (zazwyczaj punkty historyjkowe lub idealne dni) lub takich samych jednostek jakich używamy do określania rozmiarów zadań w rejestrze sprintu (roboczogodziny).

Prędkość zespołu jest wyrażona w jednostkach używanych przez elementy rejestru produktu (powiedzmy punktach historyjkowych). Zatem jeśli pojemność wyrazimy w punktach historyjkowych, wyznaczenie jej wielkości będzie wyglądać tak samo jak przewidzenie docelowej prędkości naszego zespołu w nadchodzącym sprincie.

Chcąc określić pojemność/prędkość zespołu na nadchodzący sprint, zacznij od średniej prędkości zespołu z długiego okresu wstecz lub prędkości zespołu z poprzedniego sprintu (określonej czasem mianem podejścia „wczorajszej pogody”). Następnie sprawdź, czy nadchodzący sprint będzie się czymś różnił od typowych lub poprzednich sprintów (może się okazać, że nie różni się wcale). Otrzymasz w ten sposób realistyczną pojemność (przewidywaną prędkość) na nadchodzący sprint.

Załóżmy na przykład, że średnia prędkość naszego zespołu to 40 punktów historyjkowych na dwutygodniowy sprint. Jednak sprint, który planujemy, będzie miał miejsce w trakcie dwóch ostatnich tygodni grudnia w Polsce, co oznacza, że wielu członków naszego zespołu będzie brało czas wolny, aby spędzić święta z rodziną. Używając średniej prędkości 40 punktów, wzięlibyśmy zbyt dużo pracy do tego sprintu. Znacznie lepiej będzie przyjąć, że realistyczna pojemność zespołu w trakcie tego sprintu wypadnie gdzieś w okolicach 20 punktów.

## Pojemność w roboczogodzinach

Pojemność można również wyrazić w roboczogodzinach. Sposób wyrażania zdolności zespołu do wykonywania pracy na poziomie zadań w trakcie dwutygodniowego sprintu w roboczogodzinach przedstawiony został w tabeli 19.2.

**TABELA 19.2.** Określanie potrzebnego wysiłku w roboczogodzinach

Osoba	Dni dostępne (pomniejszone o czas osobisty)	Dni na inne aktywności scrumowe	Godziny na dzień	Dostępne roboczogodziny
Grzegorz	10	2	4 – 7	32 – 56
Beata	8	2	5 – 6	30 – 36
Rafał	8	2	4 – 6	24 – 36
Szymon	9	2	2 – 3	14 – 21
Helena	10	2	5 – 6	40 – 48
Razem				140 – 197

Obliczanie pojemności pokazane w tabeli 19.2 przebiega następująco. Na początku członkowie zespołu mówią, ile dni mogą przepracować w ciągu sprintu (ilość czasu niedostępnego odpowiada czasowi osobistemu na rysunku 19.5). Ponieważ Beata i Rafał planują wziąć udział w dwudniowym szkoleniu, każde z nich ma tylko osiem dni dostępnych w tym sprincie. Szymon planuje trzydniowy weekend, zatem ma dziewięć dni dostępnych.

Następnie członkowie zespołu określają, ile czasu trzeba zarezerwować na inne aktywności scrumowe. Zostawiają jeden dzień na planowanie, przegląd i retrospekcję sprintu. Potrącają również czas przeznaczony na asystowanie właścicielowi produktu w trakcie pielęgnacji rejestru produktu. Razem daje to o dwa dni mniej dla każdej osoby na wykonywanie pracy na poziomie zadań sprintowych.

Członkowie zespołu określają następnie, ile godzin w ciągu dnia mogą poświęcić na pracę w sprincie. Każda osoba podaje przedział, który uwzględnia wszelką pracę nadmiarową niezwiązaną z elementami w rejestrze sprintu (ten narzut odpowiada kawałkowi o nazwie „inne zobowiązania pozasprintowe” na rysunku 19.5). Szymon na przykład ma tylko połowę swojego czasu na pracę nad tym produktem, stąd ocenia, że będzie mógł mu poświęcić od 2 do 3 godzin dziennie w ciągu nadchodzącego sprintu.

Po uwzględnieniu czasu osobistego, innych aktywności scrumowych i zobowiązań pozasprintowych zespół z tabeli 19.2 ocenia swoją pojemność na 140 – 197 roboczych godzin dla zadań w rejestrze sprintu.

Osobiście ostrzegłbym zespół przed braniem 197 godzin pracy, ponieważ w sprincie nie pozostały żaden zapas czasu. Lepszą strategią dla tego zespołu podczas podejmowania zobowiązania na bieżący sprint byłoby użycie pojemności większej od 140 godzin, ale zdecydowanie mniejszej od 197 godzin.

## Wybieranie elementów rejestru produktu

Niezależnie od wybranego podejścia do planowania sprintu musimy wybrać elementy z rejestru produktu, które staną się kandydatami do zobowiązania. Wybór można przeprowadzić na kilka sposobów. Jeżeli mamy określony cel sprintu, wybieramy elementy rejestru produktu zgodne z tym celem. Jeżeli cel sprintu nie został formalnie wskazany, z reguły bierzemy elementy znajdujące się na szczytce rejestru produktu. Zaczęlibyśmy od elementu na samej górze, następnie przeszlibyśmy do elementu poniżej itd. Jeżeli zespół nie mógłby podjąć zobowiązań wobec kolejnego elementu na szczytce (na przykład ze względu na brak potrzebnych umiejętności w zespole), wzięty zostałaby następny, który na pierwszy rzut oka wydaje się możliwy do zrealizowania w oparciu o bieżące ograniczenia.

Jedną ze stosowanych przeze mnie reguł podczas wybierania elementów rejestru produktu jest niewybieranie rzeczy, których nie jesteśmy w stanie ukończyć. Zatem jeśli następny element rejestru produktu jest zbyt duży, aby można było ukończyć go w sprincie przy uwzględnieniu pozostały elementów, na których ukończenie już się zgodziliśmy, powinniśmy podzielić go na dwa lub więcej mniejszych elementów, z których każdy miałby wartość dla klienta, lub rozważyć wybór innego — możliwego do zrealizowania — elementu. Przed wybieraniem elementów rejestru, które są kiepsko zdefiniowane lub nie można ich zrealizować w sprincie ze względu na brak zasobów lub istniejące zależności, powinna chronić nas dobra *definicja gotowości*.

Reguła „zaczynaj tylko to, co jesteś w stanie ukończyć” bazuje na zasadach mówiących, że powinniśmy ograniczać ilość pracy cząstkowej oraz że rozpoczęwanie czegoś i nieukończenie jest źródłem różnych form marnotrawstwa. Obie te zasady omawiałem w rozdziale 4. przy okazji zasady niezmieniania celu w trakcie trwania sprintu. Ponadto zezwalanie na przechodzenie elementów z jednego sprintu na kolejny przecież celowi osiągania potencjalnie nadającego się do wdrożenia przyrostu produktu pod koniec każdego sprintu.

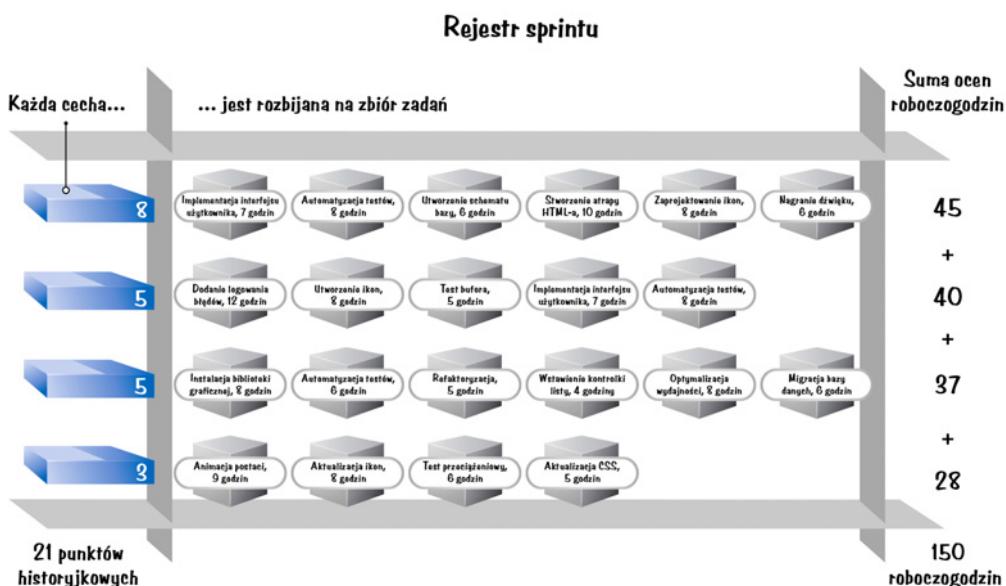
## Nabieranie pewności

Jednym ze sposobów nabierania pewności jest użycie przewidzianej prędkości do sprawdzenia, czy przyjęte zobowiązanie jest realistyczne. Jeżeli przewidywana prędkość w sprintie to 25 punktów historyjkowych, a nasz zespół wybrał pracę wartą 45 punktów, powinno mu to dać do myślenia. Powinniśmy przynajmniej zacząć pytać, skąd zespół bierze przekonanie o możliwości zrealizowania takiego zobowiązania. Wykorzystana w ten sposób wiedza na temat prędkości jest idealnym środkiem umożliwiającym sprawdzenie i zbilansowanie proponowanego zobowiązania.

Ryzyko związane z wykorzystaniem prędkości jako jedynego środka uzyskiwania pewności polega na tym, że zobowiązanie wyglądające dobrze pod względem liczb może nadal być niewykonalne. Na przykład: jeśli określiliśmy prędkość na 25 punktów, a całkowite zobowiązanie zespołu wynosi 21 punktów historyjowych, można uważać, że jest ono rozsądne. Nie będziemy mieć jednak pewności, czy elementy rejestru o wartości 21 punktów mogą być zrealizowane, dopóki nie zjedziemy niżej do poziomu zadań — tam znaleźć możemy problemy związane z zależnościami, brakiem odpowiednich umiejętności, a także inne trudności sprawiające, że wykonanie całości pracy przez zespół stanie pod znakiem zapytania.

Większość zespołów scrumowych zyskuje niezbędny poziom pewności przez rozbicie elementów rejestru produktu na zadania potrzebne do ich zrealizowania zgodnie z ustaloną wspólnie przez cały zespół scrumowy definicją ukończenia. Zadania te mogą zostać następnie ocenione (zazwyczaj w roboczogodzinach) i odjęte od pojemności zespołu. Rozbijanie elementów rejestru produktu na zadania jest formą projektowania i planowania ich realizacji w samą porę.

Wynikiem tych działań jest rejestr produktu. Przykładowy rejestr sprintu przedstawiony został na rysunku 19.6. Widać na nim elementy rejestru produktu (o całkowej wartości 21 punktów), które zespół uważa za możliwe do zrealizowania przed końcem sprintu. Rejestr sprintu pokazuje również plan (w formie zadań) dostarczenia tych elementów rejestru i tym samym zrealizowania celu sprintu.



**RYSUNEK 19.6.** Rejestr sprintu pokazujący elementy rejestru produktu i plan zadań

Czy podejmowane przez zespół zobowiązanie jest dobre lub też złe?

Jeżeli przewidywana prędkość wynosi 25 punktów, to 21 punktów wydaje się być wartością rozsądnią. Aby przekonać się, czy zobowiązanie jest dobre, spójrzmy jeszcze na rozpisane zadania. Po zsumowaniu zadań dla wszystkich czterech elementów rejestru wyszło 150 roboczogodzin. Założymy, że zespołem przydzielonym do pracy w tym sprincie jest ten opisany w tabeli 19.2. Posiada on pojemność w przedziale od 140 do 197 roboczogodzin. Zobowiązanie na poziomie 150 roboczogodzin wydaje się dawać zupełny komfort bezpieczeństwa — pozostaje spory zapas godzin na poradzenie sobie z ewentualnymi trudnościami.

Jednak komfortowe umiejscowienie 150 godzin w przedziale między 140 a 197 nie gwarantuje jeszcze dobrej jakości zobowiązania. Zwróć uwagę, że w tabeli 19.2 Szymon jest dostępny jedynie przez dwie lub trzy godziny dziennie przez dziewięć z dziesięciu dni sprintu. Oznacza to, że Szymon ma tylko od 14 do 21 roboczogodzin dostępnych w tym sprincie. Co jeśli Szymon jest jedyną osobą mogącą pracować nad zadaniami związanymi z interfejsem użytkownika? W takim przypadku zespół może nie być w stanie zobowiązać się do realizacji czterech elementów rejestru z rysunku 19.6, ponieważ istnieje ryzyko, że wraz z wyczerpaniem się pojemności Szymona wyczerpie się pojemność dla zadań związanych z interfejsem użytkownika.

Chociaż w Scrumie zwyczajowo nie przypisujemy członków zespołu do zadań w trakcie planowania sprintu (zauważ, że na rysunku 19.6 nie widać imion pracowników umieszczonych na zadaniach), musimy przynajmniej pobicieżnie rozważyć pojemność naszych umiejętności — inaczej możemy podjąć błędne zobowiązanie. Samo prawidłowe umiejscowienie naszego zobowiązania w przedziale ocenionej sumarycznej pojemności zespołu nie oznacza jeszcze, że nie zabraknie nam jej w określonym obszarze umiejętności.

Z tego powodu niektóre zespoły decydują się umieścić na każdym zadaniu notkę informującą, która z osób najprawdopodobniej będzie się nim zajmowała. Taki krok jest często niepotrzebny i wnosi potencjalne marnotrawstwo ze względu na nieoczekiwane wydarzenia w trakcie sprintu, które będą wymagały przepisywania zadań. Przypisywanie zadań do poszczególnych osób może być również szkodliwe, jeżeli działanie takie redukuje współposiadanie zadań przez zespół. Lepszą strategią (o której będę pisał w rozdziale 20.) jest pozwolenie członkom zespołu na dobieranie zadań w sposób doraźny w samą porę w czasie trwania sprintu.

## Doprecyzowywanie celu sprintu

Cel sprintu podsumowuje znaczenie biznesowe sprintu i jego wartość. Początkową postać celu sprintu powinien przedstawić na spotkaniu planowania sprintu właściciel produktu. Jednak w trakcie tego spotkania cel może zostać doprecyzowany w miarę wspólnej pracy uczestników nad realistycznym określeniem tego, co może zostać dostarczone w sprincie.

## Ostateczne ustalenie zobowiązania

Pod koniec planowania sprintu zespół deweloperski ustala ostateczne zobowiązanie co do wartości biznesowej, jaką zamierza dostarczyć przed końcem sprintu. Na zobowiązanie składają się cel sprintu i wybrane elementy rejestru produktu.

Jak wspomniałem w rozdziale 2., niektórzy ludzie wolą nazywać wartość biznesową, jaką zespół deweloperski deklaruje się dostarczyć w trakcie sprintu, *prognozą*. Ja preferuję użycie słowa *zobowiązanie*. Niezależnie od tego, którego terminu będziesz używał, obowiązywać będą te same, opisane przeze mnie w tym rozdziale podejścia do planowania sprintu.

Niuanse pomiędzy tymi dwoma określeniami mogą jedynie wpływać na zakres tego, co zespół deweloperski uzna za możliwe do dostarczenia, a także na sposób radzenia sobie zespołu scrumowego z nowymi informacjami pojawiającymi się w trakcie wykonywania sprintu (omówienie tematu zmian występujących w trakcie sprintu znajdziesz w rozdziale 4.).

## Zakończenie

W tym rozdziale rozwinąłem opis planowania sprintu, omawiając, kiedy ma miejsce ta aktywność i kto bierze w niej udział. Opisałem dwa różne podejścia do planowania sprintu. W pierwszym z nich zespół wybiera zbiór elementów rejestru produktu, a następnie zdobywa pewność co do faktycznej możliwości dostarczenia go w całości. W drugim podejściu po wybraniu elementu rejestru produktu bada się, czy może on zostać włączony do rosnącego stopniowo zobowiązania zespołu. Wyjaśniłem również dwa różne sposoby określania pojemności zespołu do realizacji pracy. W następnym rozdziale opiszę wykonanie sprintu po jego zaplanowaniu.

# Rozdział 20

## WYKONANIE SPRINTU

---

Wykonanie sprintu to praca, jaką zespół scrumowy realizuje, aby osiągnąć cel sprintu. W tym rozdziale skupię się na zasadach i technikach, którymi zespół scrumowy posługuje się, planując, zarządzając, działając i komunikując się w trakcie wykonywania sprintu.

### Wprowadzenie

Wykonanie sprintu jest miniprojektem samym w sobie — przeprowadzana jest cała niezbędna praca potrzebna do dostarczenia przyrostu produktu potencjalnie nadającego się do wdrożenia.

### Czas

Wykonanie sprintu zajmuje większość czasu przeznaczonego na sprint. Zaczyna się po planowaniu sprintu i kończy przed przeglądem sprintu (patrz rysunek 20.1).

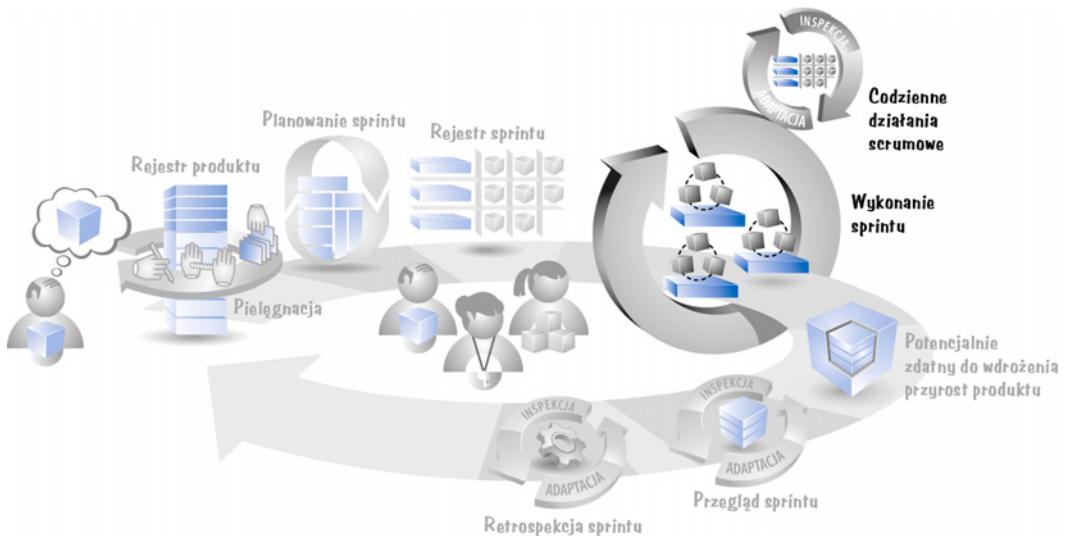
W przypadku dwutygodniowego sprintu wykonanie go może zająć do ośmiu z dziesięciu dostępnych dni.

### Uczestnicy

W czasie wykonywania sprintu członkowie zespołu deweloperskiego organizują się samodzielnie i ustalają najlepszy sposób osiągnięcia celu przyjętego w trakcie planowania sprintu.

W wykonaniu sprintu udział bierze mistrz młyna — jako trener, animator i osoba odpowiedzialna za usuwanie przeszkód. Jego zadaniem jest zrobienie wszystkiego, co możliwe, aby pomóc zespołowi w osiągnięciu sukcesu. Mistrz młyna nie przypisuje pracy zespołowi i nie mówi, jak ten ma ją wykonywać. To samoorganizujący się zespół decyduje o tych sprawach.

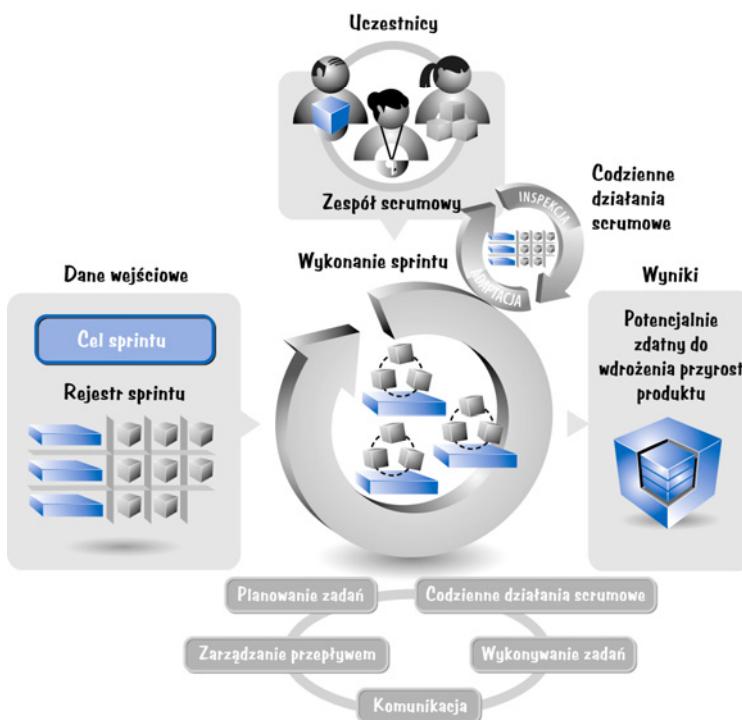
Właściciel produktu musi być dostępny w czasie wykonania sprintu, aby udzielać odpowiedzi na pytania wyjaśniające, przeglądać realizowaną na bieżąco pracę i przekazywać zespołowi swoje opinie na jej temat, omawiać wszelkiego typu poprawki celu sprintu, jeżeli zaistnieją ku temu warunki, a także weryfikować spełnienie kryteriów akceptacji dla poszczególnych elementów rejestru produktu.



**RYSUNEK 20.1.** Kiedy ma miejsce wykonanie sprintu

## Proces

Aktywność wykonania sprintu została przedstawiona na rysunku 20.2.



**RYSUNEK 20.2.** Aktywność wykonania sprintu

Danymi wejściowymi do wykonania sprintu — stworzonymi podczas planowania sprintu — są cel sprintu i rejestr sprintu. Wynikiem wykonania sprintu jest potencjalnie zdatny do wdrożenia przyrost produktu, czyli zestaw elementów rejestru produktu zrealizowanych z wysokim poziomem pewności, czyli spełniających kryteria definicji ukończenia przyjęte wspólnie przez zespół scrumowy (patrz rozdział 4.). Wykonanie sprintu obejmuje planowanie, zarządzanie i komunikowanie pracy, której realizacja jest niezbędna do stworzenia tych działających i przetestowanych cech produktu.

## Planowanie wykonania sprintu

Podczas planowania sprintu zespół tworzy *plan* mówiący, w jaki sposób osiągnięty zostanie cel sprintu. Większość zespołów tworzy rejestr sprintu, który zazwyczaj stanowi listę elementów rejestru produktu wraz z przypisanymi do nich zadaniami, wycenionymi w roboczych godzinach (patrz rysunek 19.6). Chociaż zespół mógłby się pokusić o stworzenie pełnego planu wykonania sprintu na poziomie zadań (odpowiednika planu projektu dla sprintu, być może w formie wykresu Gantta), działanie takie byłoby nieuzasadnione z punktu widzenia ekonomii.

Po pierwsze, nie ma powodu do utworzenia dla zespołu 5 – 9 osób wykresu Gantta mówiącego, kto i kiedy powinien wykonywać pracę w trakcie następnej krótkiej iteracji sprintowej. Po drugie, nawet jeśli zespół chciałby utworzyć wykres Gantta, robienie tego zaraz po rozpoczęciu prac byłoby nieadekwatne do sytuacji. Wykonywanie sprintu to w rzeczywistości czas próby. Otrzymujemy duże porcje wiedzy w oparciu o to, co budujemy i testujemy. Wiedza ta zburzy nawet najlepiej przygotowany z góry plan. Przygotowywanie takiego planu będzie zatem marnotrawstwem cennego czasu zespołu i pociągnie za sobą jeszcze większą szkodę, kiedy zajdzie potrzeba przystosowywania go do realiów sprintu.

Oczywiście pewna porcja planowania z góry pomaga w ujawnieniu ważnych zależności pomiędzy zadaniami. Na przykład: jeśli wiemy, że cecha, którą tworzymy w trakcie sprintu, musi zostać oddana specjalnemu testowi wydajnościowemu, rozsądne będzie takie ułożenie prac w sprintie, aby test ten mógł zostać rozpoczęty przynajmniej dwa dni przed końcem wykonania sprintu.

Dobrą zasadą wykonania sprintu jest podchodzenie do planowania zadań w sposób doraźny zamiast próbowania ułożenia z góry planu realizacji całej pracy [Goldberg i Rubin, 1995]. Pozwól, aby zadania planowane były w sposób ciągły w trakcie sprintu w miarę adaptowania się zespołu do ewoluujących warunków.

## Zarządzanie przepływem

Do obowiązków zespołu należy zarządzanie przepływem pracy, aby móc osiągnąć cel sprintu. Musi on podejmować takie decyzje, jak ile pracy zespół powinien zrównoleglić, kiedy należy zacząć pracę nad poszczególnymi elementami, jak powinna zostać zorganizowana praca na poziomie zadań, jaka praca wymaga realizacji i kto powinien ją wykonać.

Chcąc znaleźć odpowiedzi na te pytania, zespół powinien porzucić stare zachowania, takie jak próby utrzymywania wszystkich swoich członków na poziomie 100% wydajności (czego konsekwencje zostały opisane w rozdziale 2.), przekonanie, że praca musi być realizowana w sposób sekwencyjny, a także że każda osoba powinna skupiać się wyłącznie na swojej części rozwiązania.

## Praca równoległa i działanie w kopcu

Istotną częścią zarządzania przepływem jest określenie, nad iloma elementami rejestru produktu zespół powinien pracować równolegle celem zmaksymalizowania wartości dostarczonej pod koniec sprintu. Praca nad zbyt wielką liczbą elementów jednocześnie zmusza członków zespołu do wykonywania wielu zadań równolegle, co z kolei zwiększa czas potrzebny na wykonanie poszczególnych elementów i najprawdopodobniej pogarsza ich jakość.

Rysunek 20.3 pokazuje prosty przykład, którego używam podczas moich wykładów do zilustrowowania kosztu wykonywania wielu zadań jednocześnie.

Litera	Liczby	Liczby rzymskie
a	1	i →
b	2	ii →
c	3	iii →
d	4	iv →
e	5	v →
f	6	vi →
g	7	vii →
h	8	viii →
i	9	ix →
j	10	x →

Litera	Liczby	Liczby rzymskie
a	1	i
b	2	ii
c	3	iii
d	4	iv
e	5	v
f	6	vi
g	7	vii
h	8	viii
i	9	ix
j	10	x ↓

Wiersz po wierszu (wielozadaniowość)  
Średni czas = 35 sekund

Kolumna po kolumnie (jedno zadanie)  
Średni czas = 16 sekund

**RYSUNEK 20.3.** Koszt wielozadaniowości

Celem jest zbudowanie dwóch identycznych tabel przez wypisanie liter od a do j, liczb od 1 do 10 oraz rzymskich liczb od I do X. Jedna z tabel jest budowana według wierszy, a druga według kolumn. Tabela budowana wierszami reprezentuje wielozadaniowość (wykonaj zadanie związane z literą, następnie zadanie związane z liczbą, dalej zadanie związane z liczbą rzymską i na końcu zgłoś sekwencję dla następnej litery i liczb). Tabela budowana według kolumn odpowiada zadaniom realizowanym pojedynczo.

Wyniki tego zadania, pokazane na rysunku 20.3, wskazują, że większość ludzi buduje tabelę według kolumn mniej więcej o połowę szybciej niż tabelę według wierszy. Spróbuj sam, mierząc swoje czasy, a przekonasz się, że to prawda! Ponadto jeśli ludzie popełniają jakieś błędy, robią to zazwyczaj, wypełniając tabelę wiersz po wierszu. Wyobraź sobie odpowiednik tego marnotrawstwa w przypadku wykonywania wielu zadań jednocześnie w złożonym projekcie.

Praca nad zbyt małą liczbą zadań jest równie marnotrawna co praca nad zbyt dużą liczbą zadań. Powoduje ona niedostateczne wykorzystanie umiejętności członków zespołu i ich pojemności, co prowadzi do realizacji mniejszej ilości pracy i dostarczania mniejszej wartości.

Aby znaleźć odpowiednią równowagę, rekomenduję zespołom pracę nad taką liczbą elementów, które będą wymagały pełni umiejętności typu T (patrz rozdział 11.) oraz pojemności zespołu (patrz rozdział 19.), ale bez przeciążania. Celem jest redukcja czasu potrzebnego do wykonania

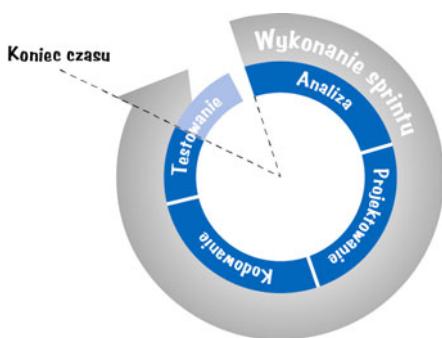
poszczególnych elementów przy jednoczesnej maksymalizacji wartości dostarczanej w ciągu sprintu (czyli zazwyczaj liczby elementów ukończonych w trakcie sprintu).

Często do opisania tego podejścia stosuje się termin **działanie w roju**, oznaczający podejmowanie przez członków zespołu posiadających wolne moce przerobowe pracy mającej na celu ukończenie elementów, które już zostały rozpoczęte przed przystąpieniem do pracy nad nowymi elementami. Zespoły, które nadal myślą w kategoriach indywidualnych ról, kończą z pewnymi członkami goniącymi do przodu ze swoją pracą i pozostałymi tkwiącymi w bagnie nieukończonych zadań. Oto przykład myślenia w sposób indywidualny, a nie zespołowy: „Być może testerzy mają jeszcze swoją pracę do zrobienia, ale ja już skończyłem implementować tę cechę, więc mogę zabrać się za kodowanie następnej”. W zespole, który działa w roju, ludzie rozumieliby, że lepiej jest pozostać skupionym na obecnej pracy i pomóc w dokończeniu testowania, zamiast biec do przodu i zaczynać pracę nad następną cechą.

Niektoří mylnie uważają, że działanie w roju jest strategią mającą na celu utrzymanie wszystkich członków zespołu na poziomie 100% wydajności. Tak nie jest. Gdybyśmy chcieli zapewnić wykorzystanie ludzi w 100%, zaczeliibyśmy pracę nad wszystkimi elementami rejestru produktu jednocześnie! Czemu tak nie robimy? Ponieważ masowe wykonywanie wielu zadań jednocześnie pozwalające na osiągnięcie takiego stanu doprowadziłoby w konsekwencji do spowolnienia potoku ukończonych elementów. Działanie w roju ma za zadanie pomóc zespołowi skupić się na celu, a nie na zadaniach, co w rezultacie pozwala ukończyć większą liczbę rzeczy szybciej.

Chociaż działanie w roju faworyzuje pracę nad mniejszą liczbą elementów w tym samym czasie, nie oznacza to absolutnie konieczności pracy wyłącznie nad jednym elementem rejestru produktu w danej chwili. Skupianie się wyłącznie nad jednym elementem może mieć sens w pewnym kontekście, ale mówienie, że wszyscy członkowie zespołu powinni wspólnie zająć się jednym elementem, może być potencjalnie niebezpieczne. To, jaka liczba elementów rejestru nadaje się do jednoczesnej realizacji, będzie zależeć od tego, co faktycznie trzeba zrobić, jakie umiejętności posiada zespół oraz jakie inne uwarunkowania mają miejsce w chwili podejmowania decyzji o rozpoczęciu prac nad kolejnym elementem lub wstrzymania się z rozpoczęciem prac.

Innym niebezpiecznym podejściem byłoby zastosowanie na poziomie sprintu myślenia rodem z procesu kaskadowego i potraktowanie wykonania sprintu jako miniaturowego projektu sekwencyjnego. Oznaczałoby to rozpoczęcie pracy nad wszystkimi elementami rejestru w tym samym czasie. Zaczeliibyśmy od przeanalizowania elementów przeznaczonych do realizacji w sprintie, następnie zaprojektowalibyśmy je, zakodowali i w końcu przetestowali (patrz rysunek 20.4).



RYSUNEK 20.4. Miniproces kaskadowy w trakcie wykonania sprintu — zły pomysł

Chociaż to podejście może wydawać się logiczne, jest bardzo ryzykowne. Co jeśli zespołowi zabraknie czasu i nie zdoła przeprowadzić całego niezbędnego testowania? Czy będziemy mieć nadający się potencjalnie do wdrożenia przyrost produktu? Nie — racjonalnie skonstruowana definicja ukończenia nigdy nie pozwoli nazwać nieprzetestowanych cech mianem ukończonych. Używając ministrategii kaskadowej, moglibyśmy skończyć z każdą cechą ukończoną na 90%, ale żadna z nich nie byłaby gotowa w 100%. Właściciel produktu nie będzie miał żadnej wartości ekonomicznej z częściowo wykonanej pracy.

## Którą pracę rozpocząć?

Zakładając, że nie zaczynamy pracy nad wszystkimi elementami rejestru produktu jednocześnie, w pewnym momencie zespół musi wskazać element, który weźmie w następnej kolejności do realizacji.

Najprostszą metodą jest wybranie następnego elementu o najwyższym priorytecie według właściciela produktu (co odpowiada jego pozycji w rejestrze produktu). Takie podejście ma oczywistą zaletę polegającą na tym, że wszelkie nieukończone elementy rejestru w danym sprintie mają niższy priorytet od tych ukończonych.

Niestety to najprostsze podejście nie zawsze jest możliwe do zastosowania w praktyce, ponieważ wzajemne zależności oraz ograniczenia pod względem pojemności umiejętności mogą wymusić realizację elementów rejestru w innej kolejności. Zespół deweloperski musi mieć możliwość podejmowania tego typu decyzji w sposób doraźny według własnego rozeznania sytuacji.

## Jak zorganizować pracę nad zadaniami?

Podejmując pracę nad elementem rejestru produktu, zespół deweloperski musi zdecydować, w jaki sposób wykonać związaną z nim pracę na poziomie zadań. Myśląc o pojedynczym elemencie rejestru produktu w kategoriach kaskadowych, zaczelibyśmy od jego przeanalizowania, przechodząc następnie do projektowania, implementowania i w końcu testowania.

Przekonanie, że istnieje wyłącznie jeden z góry ustalony porządek pracy (na przykład trzeba zacząć od zbudowania, zanim będzie można przeprowadzić jakiekolwiek testowanie), odbiera zespołowi szansę na wykonywanie rzeczy w inny, być może bardziej wydajny sposób. Często słyszę, jak nowe zespoły mówią: „Co będą robić nasi testerzy we wczesnej fazie sprintu, kiedy nie będzie jeszcze żadnych cech gotowych do testowania?”. Zazwyczaj odpowiadam, że w zespołach stosujących **programowanie sterowane testami**, czyli pisanie testów przed rozpoczęciem prac deweloperskich, „testerzy” jako *pierwi* zaczynają pracę nad cechą [Crispin i Gregory, 2009]!

Plagą szerzącą się wśród wielu zespołów jest myślenie w kategoriach ról. To, czego potrzebujemy w zamian, to myślenie skupione na dostarczaniu wartości, pozwalające członkom zespołu doraźnie wybierać zadania i wskazywać, kto będzie nad nimi pracował. Dzięki temu minimalizowany jest czas, w którym praca czeka na realizację, a także częstotliwość i rozmiar pracy, jaką członkowie zespołu muszą „przekazywać” między sobą. Może to na przykład oznaczać, że dwóch członków zespołu siądzie razem pierwszego dnia sprintu i zacznie bardzo intensywnie posuwać prace do przodu poprzez liczne cykle tworzenia testów, implementacji, wykonania testów i wprowadzania poprawek. Takie podejście pozwala na swobodny przepływ pracy (żadne z zadań nie będzie zablokowane), umożliwia bardzo szybkie otrzymanie informacji zwrotnej, dzięki czemu wszelkie problemy są identyfikowane

i bezzwłocznie likwidowane. Kolejna korzyść to możliwość pracowania zespołu z umiejętnościami typu T w roju nad elementem rejestru celem jego ukończenia.

## Jaką pracę trzeba zrealizować?

Jakie zadania musi wykonać zespół, aby móc powiedzieć, że element rejestru został ukończony? Jest to decyzja, którą podejmuje sam zespół. Właściciele produktu i menedżerowie muszą ufać w to, że członkowie zespołu są odpowiedzialnymi profesjonalistami, posiadającymi głęboką motywację do realizacji naprawdę imponującej pracy. Muszą oni zachęcać te jednostki do niezbędnych działań, dzięki którym powstaną innowacyjne rozwiązania w warunkach uwzględniających ekonomię całego przedsięwzięcia.

Oczywiście właściciele produktów i menedżerowie mają swój wpływ na to, jaka praca na poziomie zadań będzie realizowana. Po pierwsze, właściciele produktów dbają o to, aby został zdefiniowany zakres cechy i jej kryteria akceptacji (część definicji gotowości opisanej w rozdziale 6.) — dwie rzeczy, które określają granice pracy na poziomie zadań.

Właściciele produktów i menedżerowie określają również biznesową stronę definicji ukończenia. Na przykład: jeśli biznes wymaga, aby cechy stworzone w każdym sprintie były wdrażane u klienta pod koniec każdego sprintu, ma to wpływ na zadania, jakie będzie musiał zrealizować zespół (uruchomienie nowych cech na serwerach produkcyjnych oznacza więcej pracy niż tylko ich zbudowanie i przetestowanie).

Podsumowując, właściciel produktu musi współpracować z zespołem, aby podjęte zostały decyzje techniczne mające swoje określone konsekwencje biznesowe i to wszystko przy zachowaniu zasad ekonomii. Niektóre z tych decyzji będą bardziej osadzone w technicznych realiach definicji ukończenia. Na przykład zespół scrumowy może wspólnie zdecydować, że posiadanie automatycznych testów regresyjnych (mających swój koszt ekonomiczny, ale również przynoszących określony zysk) jest ważne, i w ten sposób wpływać na pracę na poziomie zadań (wymuszając konieczność tworzenia testów i uruchamiania ich za każdym razem).

Inne decyzje mają ścisły związek z cechami. Często istnieje pewien stopień swobody co do ilości wysiłku, jaki zespół powinien włożyć w realizację cechy. Na przykład udoskonalenie lub też wypolerowanie cechy może być technicznie możliwe, ale w oczach właściciela produktu zwyczajnie nie warte ponoszenia dodatkowych kosztów w danej chwili lub w ogóle. Z drugiej strony, przycinanie rogów projektu lub chodzenie na skróty pod względem czasu i sposobu przeprowadzania testów ma również swoje konsekwencje ekonomiczne, które muszą zostać rozważone (przeczytaj omówienie tematu dłużu technicznego w rozdziale 8.). Oczekuje się, że zespół będzie współpracował z właścicielem produktu, dyskutując o tego typu kompromisach i podejmując wspólnie decyzje uzasadnione ekonomicznie.

## Kto wykonuje pracę?

Kto powinien pracować nad zadaniem? Oczywista odpowiedź to: osoba, która potrafi najlepiej i najszybciej zrealizować dane zadanie. Co jeśli ta osoba jest niedostępna? Być może wykonuje już inne, ważniejsze zadanie lub jest chora i nie ma jej w biurze, a zadanie musi zostać wykonane bezzwłocznie.

Istnieje pewna liczba czynników, które mogą i powinny wpływać na to, kto będzie pracował nad zadaniem. Wspólnym obowiązkiem zespołu jest rozważenie tych czynników i podjęcie właściwej decyzji.

Kiedy członkowie zespołu posiadają umiejętności typu T, kilka osób z tego zespołu ma możliwość pracowania nad każdym z zadań. Gdy pewne umiejętności nakładają się na siebie, zespół może zebrąć ludzi w rój, aby przejść przez zadania, które powstrzymują realizację elementu rejestru produktu w trakcie wykonywania sprintu — tym samym czyniąc cały zespół bardziej wydajnym.

## Codzienne działania scrumowe

Codzienne działania scrumowe to kluczowa, wykonywana każdego dnia aktywność inspekcji i adaptacji pomagająca zespołowi osiągnąć szybszy i bardziej elastyczny przepływ pracy w kierunku rozwiązania. Jak pisałem w rozdziale 2., codzienne działania scrumowe to ograniczona w czasie (15-minutowa) aktywność, która ma miejsce co 24 godziny. Aktywność ta służy do inspekcji, synchronizacji i codziennego planowania adaptacyjnego mających na celu pomóc samoorganizującemu się zespołowi w lepszej realizacji swojej pracy.

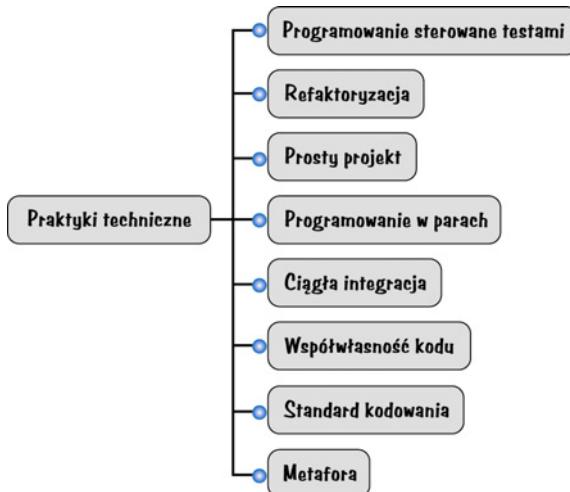
Celem codziennych działań scrumowych jest zebranie razem ludzi zaangażowanych w osiągnięcie celu sprintu i podzielenie się ogólnym obrazem sytuacji, dzięki czemu wszyscy będą mogli zrozumieć, ile potrzeba jeszcze wysiłku, nad którymi elementami rejestru należy rozpocząć pracę i jak najlepiej rozłożyć zadania pomiędzy członków zespołu. Codzienne działania scrumowe pozwalają również unikać czekania. Jeżeli pojawił się problem, który blokuje przepływ pracy, zespół nigdy nie musi czekać więcej iż jeden dzień na jego omówienie. Wyobraź sobie, że członkowie zespołu spotykaliby się tylko raz na tydzień — zablokowałiby sobie w ten sposób korzyści płynące z szybkiej pętli informacji zwrotnej (patrz rozdział 3.). Podsumowując, codzienne działania scrumowe mają kluczowe znaczenie dla zarządzania przepływem.

## Realizacja zadań — praktyki techniczne

Od członków zespołu deweloperskiego oczekuje się dobrych umiejętności technicznych. Nie oznacza to, że do używania Scruma potrzebujesz zespołu supergwiazd. Niemniej jednak praca w krótkich, ograniczonych czasowo iteracjach, których oczekiwany wynikiem jest potencjalnie nadający się do dystrybucji przyrost produktu, mobilizuje zespoły do wykonania pracy przy zachowaniu kontroli nad długiem technicznym. Jeżeli zespołowi brakuje odpowiednich umiejętności technicznych, istnieje spore prawdopodobieństwo, że nie uda mu się osiągnąć dobrego poziomu zwinności potrzebnego do dostarczania wartości biznesowej w sposób ciągły przez długi czas.

Jeżeli używasz Scruma do produkcji oprogramowania, członkowie zespołu muszą umieć stosować najlepsze techniki programistyczne. Nie mam tutaj na myśli jakichś wydumanych umiejętności, ale takie, które były i są w użyciu od dekad i mają kluczowe znaczenie dla osiągnięcia sukcesu w Scrumie i przypuszczalnie w każdym innym podejściu do wytwarzania oprogramowania — na przykład w ciągłej integracji, testowaniu automatycznym, refaktoryzacji, programowaniu sterowanym testami itd. Dzisiaj społeczność metod zwinnych określa te techniki mianem programowania ekstremalnego

[Beck i Andres, 2004], chociaż większość z nich pochodzi z czasów, zanim powstało to określenie (podzbiór praktyk należących do programowania ekstremalnego znajdziesz na rysunku 20.5).



**RYSUNEK 20.5.** Podzbiór technik wchodzących w skład programowania ekstremalnego

Weźmy za przykład testowanie automatyczne, niezbędne do wspierania kilku technik z rysunku 20.5. Zespół deweloperski, który nie skupia się na automatyzowaniu swoich testów, zacznie bardzo szybko zwalniać i podejmować nieustannie rosnące ryzyko. W pewnym momencie może dojść do sytuacji, w której ręczne przeprowadzenie testów regresyjnych dla zrealizowanych wcześniej cech wymagać będzie całego czasu przeznaczonego na wykonanie sprintu. W celu opanowania sytuacji zespół może zdecydować, że nie będzie wykonywał testów manualnych w każdym sprintie, a to pozwoli na mnożenie się błędów i zwiększanie poziomu długiego technicznego (zwiększenie ryzyka). Nie uda Ci się osiągnąć zwinności na długo, jeśli nie zacznesz automatyzować swoich testów.

Podobne argumenty można przytoczyć w odniesieniu do pozostałych technik. Większość zespołów deweloperskich osiąga długotrwałe zyski ze Scruma tylko i wyłącznie poprzez stosowanie solidnych praktyk technicznych podczas wykonywania pracy na poziomie zadań.

## Komunikowanie

Jedną z zalet realizacji pracy przez małe zespoły w krótkich ograniczonych czasowo iteracjach jest brak konieczności tworzenia złożonych wykresów i raportów do zakomunikowania postępów! Chociaż można użyć dowolnej, odpowiednio przejrzystej metody raportowania stanu prac, większość zespołów jako swoich podstawowych **radiatorów informacji** używa tablicy zadań i wykresu spalania lub rozpalania.

## Tablica zadań

**Tablica zadań** to prosta, ale bardzo funkcjonalna metoda komunikowania poprzez jedno spojrzenie postępu prac w sprincie. Z formalnego punktu widzenia tablica zadań pokazuje ewoluujący w czasie stan rejestru sprintu (patrz rysunek 20.6).



RYSUNEK 20.6. Przykład tablicy zadań

Każdy element rejestru produktu planowany do zrealizowania w sprincie jest przedstawiony na tej tablicy w formie zestawu zadań niezbędnych do ukończenia danej cechy. Wszystkie zadania startują od pozycji w kolumnie „do zrobienia”. Kiedy zespół uzna, że można zacząć pracę nad elementem, jego członkowie wybierają związane z nim zadania w kolumnie „do zrobienia” i przenoszą je do kolumny „w trakcie realizacji”. Kiedy zadanie zostanie wykonane, zostaje przeniesione do kolumny „zrealizowane”.

Oczywiście rysunek 20.6 jest tylko przykładem wyglądu tablicy zadań. Zespół może umieścić na tablicy dodatkowe kolumny, jeśli uzna, że warto pokazać przepływ pracy przez inne stany. Istnieje alternatywne zwinne podejście o nazwie kanban [Anderson, 2010], które używa właśnie takiej szczegółowej tablicy zadań do przedstawienia przepływu pracy przez różne stany.

## Wykres spalania sprintu

Każdego dnia podczas wykonywania sprintu członkowie zespołu uaktualniają swoje oceny odnośnie do ilości pracy pozostającej do wykonania dla każdego nieukończonego zadania. Moglibyśmy utworzyć arkusz wizualizujący te dane. Tabela 20.1 pokazuje przykład 15-dniowego sprintu, na który początkowo składało się 30 zadań (tabela nie pokazuje wszystkich zadań i dni).

Liczba godzin pozostających dla każdego zadania w tabeli 20.1 wyznacza pewien ogólny trend spadkowy w trakcie sprintu — wynika on z pracy nad zadaniami i kończeniem ich. Jeżeli zadanie nie zostało jeszcze rozpoczęte (nadal widnieje w kolumnie „do zrobienia”), jego rozmiar jest taki sam dzień po dniu, aż do momentu rozpoczęcia pracy nad nim. Oczywiście zadanie może okazać się większe, niż początkowo zakładano, i stąd ilość czasu może rosnąć dzień po dniu, zamiast spaść (patrz zadanie numer 4 w dniach 4. i 5. — tabela 20.1) lub utrzymywać niezmienioną wartość,

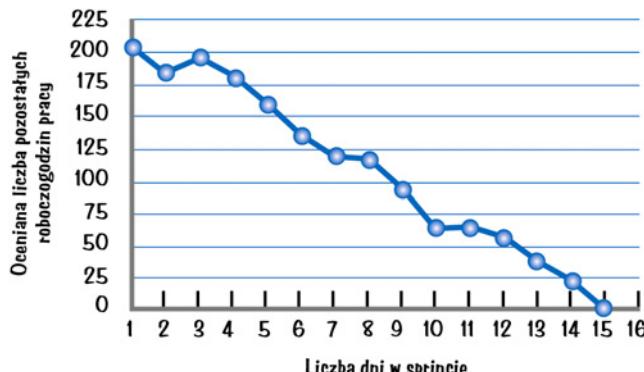
**TABELA 20.1.** Rejestr sprintu z oceną wysiłku potrzebnego do zrealizowania każdego zadania dzień po dniu

Zadania	D1	D2	D3	D4	D5	D6	D7	D8	D9	...	D15
Zadanie 1	8	4	4	2							
Zadanie 2	12	8	16	14	9	6	2				
Zadanie 3	5	5	3	3	1						
Zadanie 4	7	7	7	5	10	6	3	1			
Zadanie 5	3	3	3	3	3	3	3				
Zadanie 6	14	14	14	14	14	14	14	8	4		
Zadanie 7						8	6	4	2		
Zadanie 8 – 30	151	139	143	134	118	99	89	101	84		0
Razem	200	180	190	175	155	130	115	113	90		0

mimo że zespół zaczął już nad nim pracować (patrz zadanie numer 1 w dniach 4. i 5. — tabela 20.1), ponieważ poprzedniego dnia nie zrobiono nic w zakresie tego zadania lub pomimo prac nad nim pozostały czas potrzebny do jego wykonania nie uległ zmianie.

Nowe zadania mogą być dodawane w dowolnej chwili do elementów rejestru, które zespół zobowiązał się zrealizować w trakcie sprintu. Na przykład w 6. dniu (tabela 20.1) zespół odkrył, że brakowało zadania numer 7 i dodał je. Nie ma powodu, aby unikać dodawania zadań do rejestru sprintu. Reprezentuje on rzeczywistą pracę, jaką zespół musi wykonać, aby ukończyć element rejestru produktu, do którego realizacji się zobowiązał. Zgoda na dodawanie nieprzewidzianych wcześniej zadań do rejestru sprintu nie jest luką w systemie pozwalającą wprowadzać dodatkową pracę do sprintu. Jest to jedynie potwierdzenie faktu, iż w czasie planowania sprintu możemy nie być w stanie zdefiniować pełnego zestawu zadań potrzebnych do zaprojektowania, zbudowania, zintegrowania i przetestowania elementów rejestru przyjętych przez zespół do realizacji. W miarę coraz lepszego rozumienia naszej pracy możemy i powinniśmy aktualniwać rejestr sprintu.

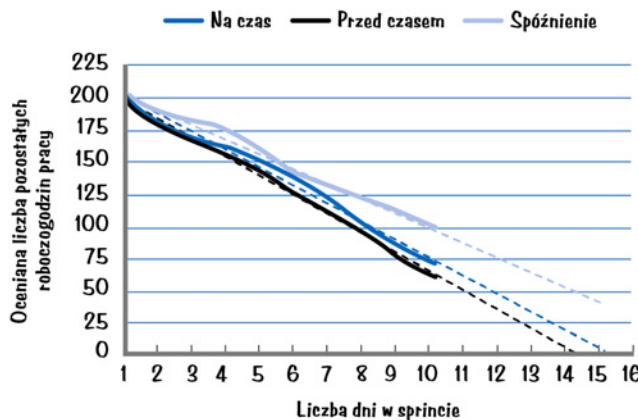
Jeżeli do narysowania wykresu użyjemy wartości z wiersza „Razem” tabeli 20.1, będących sumą pozostałych roboczych godzin we wszystkich niezrealizowanych jeszcze zadaniach, otrzymamy jeszcze jeden z artefaktów Scruma służących do komunikowania postępów — wykres spalania sprintu (patrz rysunek 20.7).



RYSUNEK 20.7. Wykres spalania sprintu

W rozdziale 18. opisałem wykresy spalania wersji dystrybucyjnej. Tam na osi pionowej umieszczone były punkty historyjkowe lub idealne dni, natomiast na osi poziomej znajdowały się numery kolejnych sprintów (patrz rysunek 18.11). Na wykresach spalania sprintu wartości na osi pionowej to przewidywane roboczogodziny pracy czekające na wykonanie, natomiast wartości na osi poziomej to kolejne dni sprintu. Rysunek 20.7 pokazuje, że mamy 200 roboczogodzin pozostających do wykonania w pierwszym dniu sprintu i zero roboczogodzin w dniu 15. (ostatniego dnia trzytygodniowego sprintu). Każdego dnia aktualniśmy ten wykres, aby pokazać całkowity spodziewany wysiłek potrzebny do zrealizowania wszystkich nieskończonych jeszcze zadań.

Podobnie jak w przypadku wykresów spalania dla wersji dystrybucyjnych wykresy spalania dla sprintu są pomocą podczas śledzenia postępów i mogą służyć jako główny wskaźnik przewidywania, kiedy cała praca zostanie ukończona. W dowolnej chwili na podstawie danych historycznych moglibyśmy wyznaczyć linię trendu, która wskaże przewidywany termin zakończenia przy założeniu utrzymania bieżącego tempa pracy i niezmienionego zakresu (patrz rysunek 20.8).



RYSUNEK 20.8. Wykres spalania sprintu z liniami trendu

Na tym rysunku widać nakładające się na siebie trzy linie trendu ilustrujące trzy różne sytuacje. Kiedy linia trendu przecina oś poziomą w pobliżu końca sprintu, możemy uznać, że jesteśmy we

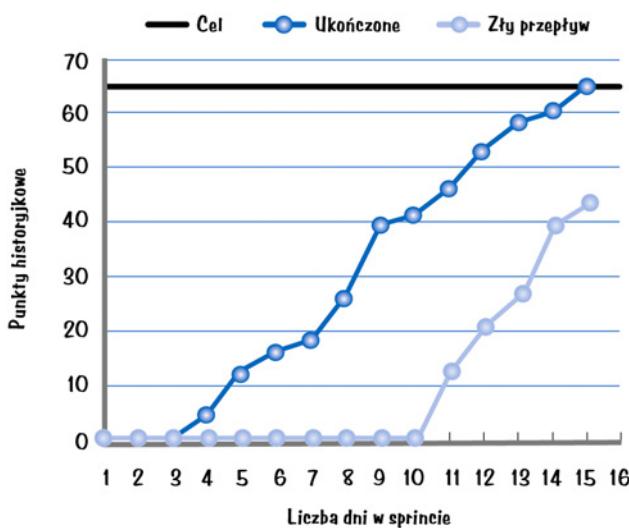
względnie dobrej formie („na czas”). Jeśli linia ląduje zdecydowanie na lewo od końca sprintu, możemy sprawdzić, czy jesteśmy w stanie wziąć bezpiecznie do sprintu więcej pracy („przed czasem”). Jeśli jednak linia ląduje zdecydowanie na prawo („spóźnienie”), jest to wyraźny sygnał, że nie działamy w oczekiwany tempie lub wzięliśmy zbyt wiele pracy do sprintu (lub te dwie rzeczy mają miejsce jednocześnie!). Kiedy tak się stanie, powinniśmy przeprowadzić dogłębne śledztwo, które ujawni, skąd biorą się takie, a nie inne liczby i pozwoli podjąć ewentualne środki zaradcze. Dokonując projekcji linii trendu, otrzymujemy kolejny zestaw danych, które powiększą naszą świadomość radzenia sobie z zarządzaniem przepływem w sprincie.

Rejestr sprintu i wykres spalania sprintu używają zawsze przybliżonego *pozostalonego* wysiłku. Nie mają one na celu uchwycenia wysiłku już poniesionego. W Scrumie nie ma powodu do pamiętania *rzeczy dokonanych*, chociaż Twój organizacja może mimo wszystko chcieć takich danych z powodów pozascrumowych, takich jak księgowość kosztów, czy też dla celów podatkowych.

## Wykres rozpalania sprintu

Tak jak wykres rozpalania wersji pokazuje postęp wersji, tak wykres rozpalania sprintu wizualizuje postęp sprintu. Oba typy wykresów pokazują ilość pracy zrealizowaną w kierunku osiągnięcia celu — w pierwszym przypadku jest to cel wersji dystrybucyjnej, w drugim cel sprintu.

Przykład wykresu rozpalania sprintu pokazuje rysunek 20.9.



RYSUNEK 20.9. Wykres rozpalania sprintu

W przypadku wykresu rozpalania sprintu praca może być reprezentowana przez roboczogodziny (tak jak w przypadku wykresu spalania) lub przez punkty historyjkowe (jak pokazuje to rysunek 20.9). Wiele osób preferuje punkty historyjkowe na swoich wykresach rozpalania, ponieważ pod koniec sprintu jedyną rzeczą, jaka ma znaczenie dla zespołu scrumowego, jest praca o wartości biznesowej, jaką udało się wykonać w trakcie sprintu, a ta mierzona jest w punktach historyjkowych (lub idealnych dniach), a nie w roboczogodzinach.

Ponadto jeśli ukończone elementy rejestru produktu mierzymy w punktach historyjkowych, już na pierwszy rzut oka będziemy mogli stwierdzić, jak wygląda przepływ pracy i jak zespół radzi sobie z kompletowaniem elementów rejestru produktu w sprincie. W celu zilustrowania tej uwagi na wykresie z rysunku 20.9 umieszczona została trzecia linia trendu o nazwie „Zły przepływ” (w normalnych warunkach ta linia nie znalazłaby się na wykresie, została ona tutaj dodana wyłącznie w celach porównawczych). Linia „złego przepływu” pokazuje, jak mógłby wyglądać wykres rozpalania sprintu, gdyby zespół rozpoczął zbyt wiele elementów rejestru produktu jednocześnie, a tym samym opóźnił ich ukończenie do późnej fazy sprintu i nie spełnił celu sprintu ze względu na zmniejszoną prędkość wynikającą ze zbyt dużej liczby rzeczy wykonywanych jednocześnie, pracy nad dużymi elementami, których realizacja jest czasochłonna, lub podejmowania innych działań skutkujących złym przepływem.

## Zakończenie

W tym rozdziale opisałem wykonanie sprintu, które zajmuje w tym sprincie większość czasu. Podkreśliłem fakt, iż wykonanie sprintu nie jest sterowane żadnym przygotowanym z góry planem specyfikującym, jaka praca będzie realizowana, kiedy będzie realizowana lub kto będzie ją wykonywał. Przeciwnie, wykonanie sprintu przebiega w sposób doraźny, bazując na umiejętnościach zespołu, informacjach zwrotnych uzyskanych w oparciu o rzeczy już wykonane oraz nieprzewidywalnych okolicznościach sprintu. Nie oznacza to wcale, że wykonanie sprintu ma charakter chaotyczny — jest raczej sterowane przez zastosowanie dobrych zasad zarządzania przepływem, które pozwalają stwierdzić, ile pracy realizować równolegle, jaką pracę rozpoczęć, jak ją zorganizować, kto powinien ją wykonać i ile wysiłku należy w nią zainwestować. W tym kontekście wskazałem codzienne działania scrumowe jako bardzo wartościowy czynnik w zarządzaniu przepływem. Wspomniałem również o wadze dobrych praktyk technicznych, umożliwiających osiągnięcie wysokiego poziomu zwinności. Na koniec zająłem się różnymi metodami komunikowania postępu sprintu przez zespół scrumowy. Te metody to tablica zadań, wykres spalania sprintu i wykres rozpalania sprintu. W następnym rozdziale omówię aktywność przeglądu sprintu, która ma miejsce po wykonaniu sprintu.

## Rozdział 21

# PRZEGŁĄD SPRINTU

---

Pod koniec sprintu zespół przeprowadza dwie bardzo ważne aktywności inspekcyjno-adaptacyjne: przegląd sprintu i retrospekcję sprintu. Przegląd sprintu skupia się na samym produkcie. Retrospekcja z kolei ma na celu analizowanie *procesu*, jakiego zespół używa do budowania produktu.

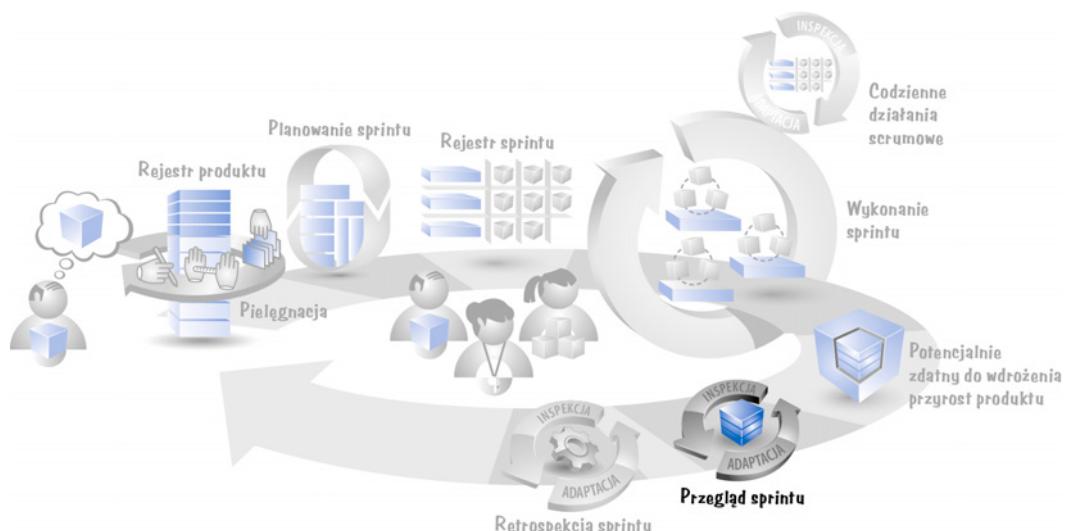
W tym rozdziale opiszę przegląd sprintu — cel tej aktywności, uczestników i pracę potrzebną do jej przeprowadzenia. Na koniec wskażę kilka powszechnie spotykanych problemów w trakcie przeglądu sprintu.

## Wprowadzenie

Podczas planowania sprintu planujemy pracę. Podczas wykonywania sprintu wykonujemy pracę. Podczas przeglądu sprintu dokonujemy inspekcji (i adaptacji) wyników pracy (potencjalnie zdaneego do wdrożenia przyrostu produktu). Przegląd sprintu ma miejsce pod koniec każdego sprintu, tuż za wykonaniem sprintu i przed (a czasami za) retrospekcją sprintu (patrz rysunek 21.1).

Przegląd sprintu daje wszystkim wnoszącym wkład w wysiłek mający na celu stworzenie produktu szansę inspekcji i adaptacji tego, co zostało zbudowane do tej pory. W trakcie tej aktywności można obiektywnie spojrzeć na bieżący stan produktu i ujawnić wszelkie niewygodne prawdy na jego temat. Jest to czas zadawania pytań, czynienia obserwacji i dyskutowania na temat tego, jaką drogą iść dalej w zaistniałych realiach.

Przegląd sprintu ze względu na pomoc, jaką daje organizacji w tworzeniu pomyślnego produktu, jest jedną z najistotniejszych pętli zdobywania wiedzy w środowisku Scrum. A ponieważ sprinty są krótkie, pętla ta jest bardzo szybka, co pozwala na częste poprawki kursu i nadawanie produkcji właściwego kierunku. Gdybyśmy zamiast tego odraczali proces pętli zwrotnej do znacznie późniejszego momentu i zakładali, że wszystko przebiega zgodnie z pewnym planem bazowym, otrzymalibyśmy zapewne to, do czego wielu jest przyzwyczajonych — zaskoczenie, rozczarowanie i frustrację.



**RYSUNEK 21.1.** Kiedy ma miejsce przegląd sprintu

## Uczestnicy

Przegląd sprintu daje zespołowi scrumowemu istotną szansę otrzymania informacji zwrotnej od ludzi, którzy zazwyczaj nie są dostępni na co dzień w trakcie wykonywania sprintu. Dla tych osób przegląd sprintu jest pierwszą okazją do obejrzenia i przedyskutowania pracy zrealizowanej w trakcie sprintu. Stąd też w aktywności tej powinny brać udział wszystkie zainteresowane strony — patrz tabela 21.1.

**TABELA 21.1.** Uczestnicy przeglądu sprintu

Rodzaj uczestników	Opis
Zespół scrumowy	Właściciel produktu, mistrz młyna i zespół deweloperski powinni być obecni, aby móc wspólnie wysłuchać opinii i odpowiedzieć na pytania dotyczące sprintu i przyrostu produktu.
Wewnętrzni interesariusze	Właściciele firmy, dyrektorzy wykonawczy i menedżerowie powinni zobaczyć poczynione postępy i zasugerować na gorąco poprawki co do kierunku. W przypadku wewnętrznej produkcji w spotkaniu powinni brać udział wewnętrzni użytkownicy, eksperci z danej dziedziny, a także menedżerowie biznesowi, do których ostatecznie trafi gotowy produkt.
Inne zespoły wewnętrzne	W przeglądzie sprintu mogą wziąć udział przedstawiciele sprzedaży, marketingu, wsparcia użytkowników, działu prawnego, a także ludzie z innych zespołów scrumowych lub pozascrumowych. Każdy z takich uczestników może dostarczyć informacji zwrotnej w zakresie własnej wiedzy eksperckiej lub też zsynchronizować swoją pracę z pracą zespołu scrumowego.
Zewnętrzni interesariusze	Klienci zewnętrzni, użytkownicy i partnerzy mogą dostarczyć cennych informacji zespołowi scrumowemu, a także innym uczestnikom spotkania.

Wszyscy członkowie zespołu scrumowego (właściciel produktu, mistrz młyna i zespół deweloperski) powinni być obecni w trakcie każdego przeglądu sprintu, aby móc opisać zrealizowane rzeczy, odpowiedzieć na pytania i przyjąć pierwsze pochwały.

W spotkaniu powinni również uczestniczyć wewnętrzni interesariusze, tacy jak właściciele obszarów biznesowych (którzy być może płacą za zbudowanie systemu), zarząd wykonawczy oraz menedżerowie zasobów i inni. Ich opinie mają kluczowe znaczenie dla zapewnienia działania zespołu w kierunku uzasadnionego ekonomicznie wyniku. Ponadto spotkania przeglądowe stanowią wygodną okazję do poznania stanu wysiłku deweloperskiego. W przypadku wytwarzania produktu wewnętrznego użytkownikami są osoby z tej samej organizacji — ich przedstawiciele powinni pojawiać się na przeglądach sprintu razem z ekspertami w danej dziedzinie, którzy stanowią doskonale źródło cennych opinii o aktualnym stanie budowanego produktu.

Również inne osoby z organizacji mogą chcieć wziąć udział w przeglądzie sprintu. Często na spotkaniach pojawiają się handlowcy i osoby z marketingu. Potrafią oni doskonale ocenić, czy produkt zmierza w kierunku sukcesu na swoim rynku sprzedaży. Inne grupy, takie jak wsparcie użytkowników, dział prawny i dział zgodności mogą również pojawiać się na spotkaniach, aby być na bieżąco z postępami zespołu scrumowego, udzielać mu swoich opinii, a także lepiej określić moment rozpoczęcia swojej części pracy w obszarze tworzonego produktu.

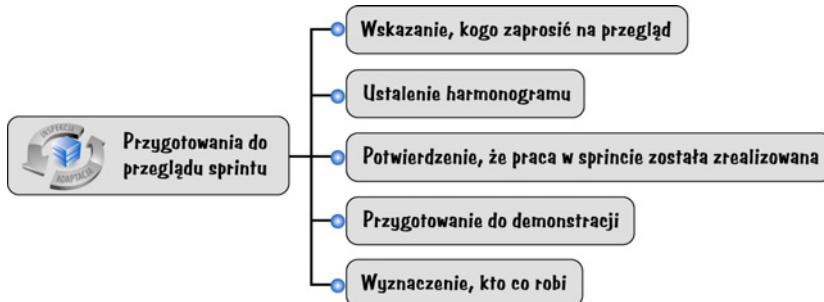
Inne zespoły deweloperskie działające w zbliżonym obszarze mogą wysyłać swoich przedstawicieli, aby stwierdzić, w jakim kierunku zmierza produkt, i udzielić informacji na temat swoich własnych działań oraz możliwości ich wpływu na bieżący wysiłek deweloperski.

Dobrym pomysłem jest przynajmniej okresowe zapraszanie na spotkania interesariuszy wewnętrznych, takich jak faktyczni klienci lub użytkownicy budowanego systemu. Dzięki ich obecności w sali konferencyjnej możemy otrzymać informacje z pierwszej ręki zamiast wiedzy pośredniej pochodzącej od interesariuszy wewnętrznych. Obecność interesariuszy zewnętrznych na każdym spotkaniu nie jest konieczna, szczególnie jeśli wiemy, że niektóre przeglądy będą bogate w wewnętrzne dyskusje, które najlepiej przeprowadzać wyłącznie z udziałem interesariuszy wewnętrznych. Jeśli zdecydujemy się jednak włączyć w spotkania interesariuszy zewnętrznych, to o ile nie jest to tylko jedna osoba, trzeba zastanowić się, których z potencjalnie wielu klientów i użytkowników powinniśmy zaprosić. Przy wyborze interesariuszy zewnętrznych należy kierować się zdrowym rozsądkiem, a także wyczulением na potrzeby i osobowości zapraszanych osób.

## Przygotowanie

Chociaż przegląd sprintu jest aktywnością nieformalną, zespół scrumowy musi poczynić pewne minimalne przygotowania (patrz rysunek 21.2).

Przygotowania te obejmują podjęcie decyzji, kogo zaprosić, ustalenie harmonogramu spotkań, potwierdzenie, iż praca w sprincie została zrealizowana, przygotowanie do demonstracji, a także wyznaczenie osoby prowadzącej spotkanie i demonstrującej nowe cechy.



**RYSUNEK 21.2.** Przygotowania do przeglądu sprintu

## Wskazanie, kogo zaprosić na przegląd

W pierwszej kolejności zespół scrumowy powinien określić, kto ma regularnie uczestniczyć w przeglądach sprintów. Celem jest zaproszenie do sali konferencyjnej właściwych osób i uzyskanie od nich jak najwartościowszych informacji. O ile nie ma przesłanek do niezapraszania kogoś lub jakiejś grupy, wyślij zaproszenie do jak najszerzego kręgu osób i pozwól im samodzielnie wybrać — jeśli będą zainteresowani, sami przyjdą do sali konferencyjnej i wezmą udział w spotkaniu.

Od czasu do czasu zespół będzie musiał ograniczyć audytorium. Na przykład zajdzie potrzeba, aby zespół skupił się na pewnej osobie lub grupie, której opinia ma kluczowe znaczenie podczas przeglądu sprintu. Ewentualnie zespół może w trakcie sprintu budować cechę dla wybranego klienta i w związku z tym nie może zaprosić na przegląd klientów stanowiących konkurencję.

Jeśli podejrzewasz, że taka sytuacja może mieć miejsce, zidentyfikuj podstawową grupę osób, które mogą zostać zaproszone na każdy przegląd sprintu, a następnie co sprint wysyłaj oddzielne zaproszenie do wyselekcjonowanych grup lub klientów.

## Ustalenie harmonogramu

Przegląd sprintu musi posiadać swój harmonogram (kiedy, gdzie i jak długo ma trwać). Spośród czterech wymaganych i występujących okresowo aktywności Scruma (planowania sprintu, codziennych działań scrumowych, przeglądu sprintu i retrospekcji sprintu) przegląd sprintu jest najtrudniejszy do zaplanowania, ponieważ biorą w nim udział osoby spoza zespołu scrumowego. Pozostałe trzy cykliczne aktywności wymagają obecności jedynie osób z zespołu scrumowego i w związku z tym mogą mieć miejsce i czas dobrane dogodnie dla wszystkich jego członków.

Aby ułatwić sobie ustalenie harmonogramu, zaczniź od sprawdzenia, kiedy najwygodniej będzie wziąć udział w tym spotkaniu kluczowym interesariuszom (grupie podstawowej, o której mówiłem wcześniej) — powiedzmy, że będą to piątkowe popołudnia, zaczynając od godziny drugiej — a następnie ustal pozostałe aktywności sprintowe wokół tego ustalonego czasu. Jeżeli będziemy używać sprintów o jednakowej długości (powiedzmy dwutygodniowych) — o czym pisałem w rozdziale 4. — dzięki regularnemu taktowi będziemy mogli ustalić wszystkie albo przynajmniej większość spotkań przeglądów sprintów (co drugi piątek o drugiej po południu). Daje to podwójny zysk redukcji zadań i kosztów administracyjnych, a także zwiększenia frekwencji.

Czas trwania przeglądu sprintu ulega zmianom w zależności od kilku czynników, poczynając od długości sprintu, rozmiaru zespołu, a także od tego, czy w spotkaniu bierze udział wiele zespołów. Zazwyczaj jednak przegląd sprintu nie przekracza czterogodzinnego ograniczenia czasowego. Wiele zespołów uznaje za użyteczną regułę jednej godziny na każdy dzień sprintu. Innymi słowy: dla dwutygodniowego sprintu przegląd powinien trwać nie dłużej niż dwie godziny, a dla czterogodzinnego nie więcej niż cztery godziny.

## Potwierdzenie, że praca w sprintie została zrealizowana

W trakcie przeglądu sprintu zespół ma prawo zaprezentować jedynie pracę, która została ukończona, czyli spełnia ustaloną wspólnie definicję ukończenia. (Więcej na ten temat znajdziesz w rozdziale 4.). To oznacza, że w pewnym czasie *przed* przeglądem sprintu ktoś musiał stwierdzić, które elementy rejestru produktu są ukończone — skąd inaczej zespół scrumowy wiedziałby, jakie elementy rejestru zaprezentować?

Ostateczna decyzja o tym, czy praca została ukończona, należy do właściciela produktu. Jak wspomniałem w rozdziale 9., właściciel produktu powinien wykonywać przeglądy elementów rejestru produktu w samą porę, czyli kiedy tylko staną się one dostępne w trakcie sprintu. Dzięki temu kiedy przyjdzie pora na przegląd sprintu, zespół będzie wiedział, które elementy są gotowe.

Nie wszyscy zgadzają się z opinią, że właściciel produktu powinien przeglądać prace przed przeglądem sprintu. Niektórzy praktycy uważają, że właściciel produktu powinien formalnie akceptować pracę wyłącznie w trakcie przeglądu sprintu. Twierdzą, że jeśli właścicielowi produktu pozwala się przeglądać pracę w trakcie sprintu, może on żądać zmian wykraczających poza zakres sprintu, czyli takich, które zmieniają cel sprintu i najprawdopodobniej wpłyną niszczącą na wykonanie sprintu (patrz rozdział 4.).

Jest to potencjalne ryzyko, ale zalety wczesnych przeglądów przeprowadzanych przez właściciela produktu (szybka pętla zwrotna) znacznie przewyższają ewentualne wady takiego podejścia. Ponadto jeśli właściciel produktu zobaczy pracę zespołu po raz pierwszy w trakcie spotkania przeglądu sprintu, to znaczy, że zobaczy ją zbyt późno. Oto dlaczego. Właściciel produktu musi być dostępny w trakcie wykonywania sprintu, aby odpowiadać na pytania i wyjaśniać elementy rejestru produktu. W trakcie wykonywania tych obowiązków powinien on również dokonywać przeglądu postępów zespołu i dostarczać kluczowych opinii „w locie”, pozwalających zespołowi reagować bezzwłocznie w sposób wydajny pod względem kosztów. Odraczanie tej pętli informacji zwrotnej do przeglądu sprintu generowałoby niepotrzebną pracę i najprawdopodobniej frustrowałoby zespół („Dlaczego nie wspomniałeś o tym w trakcie sprintu, kiedy mogliśmy to poprawić w prosty sposób?”). Również interesariusze mogliby być poirytowani („Ta cecha mogła nadawać się potencjalnie do dystrybucji, gdybyście tylko poprawili te rzeczy w trakcie sprintu!”).

Ponadto powstaje jeszcze jeden problem. Właściciel produktu odrzucający lub kwestionujący pracę w trakcie przeglądu sprintu może stwarzać wrażenie, że działa przeciwko pozostałym członkom zespołu scrumowego. Ten brak porozumienia może zostać odebrany przez interesariuszy jako stary, dobrze znany problem my kontra wy. Właściciel produktu i zespół deweloperski pracują w tym samym zespole scrumowym i w związku z tym w trakcie przeglądu sprintu powinni prezentować się jako jedność.

## Przygotowanie do demonstracji

Ponieważ cała praca, jaką zespół prezentuje w trakcie przeglądu sprintu, jest ukończona (nadaje się do potencjalnego wdrożenia), przygotowania do jej pokazania nie powinny zająć zbyt wiele czasu. Celem jest przedstawienie wszystkiego w sposób przezroczysty, dający szansę na inspekcję i adaptację produktu, a nie tworzenie przedstawienia w stylu Hollywood mającego wzbudzić podekscytowanie wśród widzów.

Przegląd sprintu powinien być nieformalnym spotkaniem z małą ilością ceremonii i dużą wartością. Spędzanie masy czasu na tworzeniu i polerowaniu prezentacji PowerPoint nie znajduje tutaj uzasadnienia. Poza tym byłbym bardzo zaniepokojony, gdyby na spotkaniu przeglądu sprintu zamiast działającego oprogramowania pokazano by mi prezentację PowerPoint. Zastanawiałbym się: „Czy faktycznie coś ukończyliśmy?” Dlaczego nie pokażecie mi tego, co właśnie zrobiliście?”.

Większość zespołów stosuje zasadę poświęcania przygotowaniom do przeglądu sprintu od 30 minut do godziny na każdy tydzień sprintu. Dodatkowo wiele z nich przyjmuje założenie, że będą pokazywane jedynie te artefakty, których powstanie było konsekwencją realizacji celu sprintu.

Oczywiście istnieją wyjątki od tej reguły. Współpracowałem kiedyś z organizacją, która tworzyła systemy dla armii Stanów Zjednoczonych. Na większości przeglądów sprintów pojawiali się urzędnicy rządowi (z działów administracyjnych). Od czasu do czasu jednak wśród uczestników planowanego spotkania był wymieniany generał dowodzący armii. W takich momentach zespół ze zrozumiałych względów inwestował więcej czasu w przygotowanie i wypolerowanie wszystkiego!

## Wyznaczenie, kto co robi

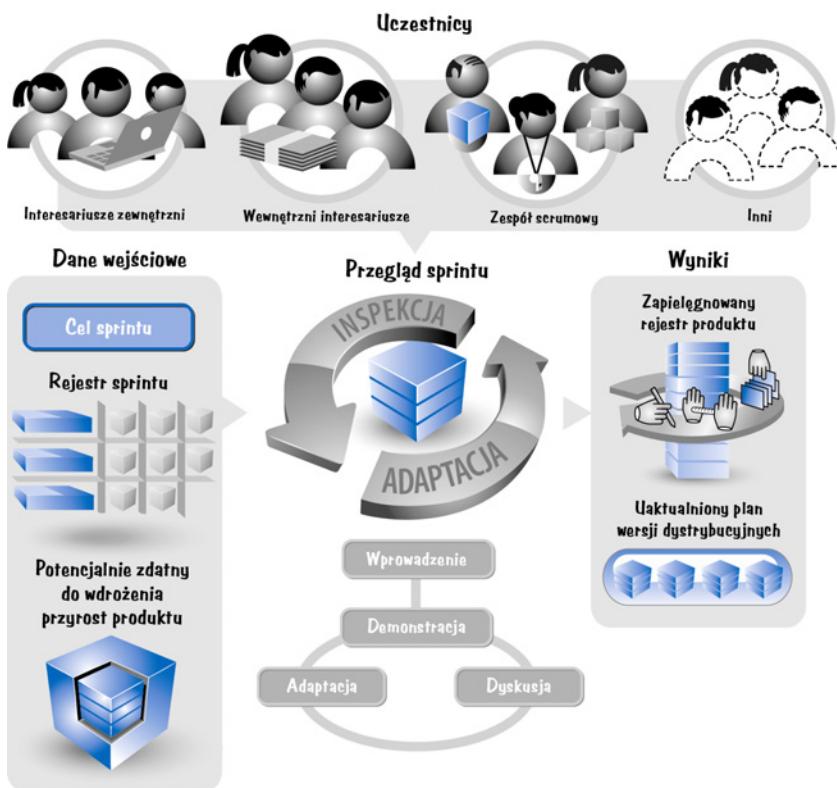
Przed przeglądem sprintu zespół musi zdecydować, kto konkretnie zajmie się prowadzeniem przeglądu i kto będzie demonstrował ukończoną pracę. Zazwyczaj osobą prowadzącą spotkanie jest mistrz młyna, ale samo rozpoczęcie, połączone z powitaniem interesariuszy i streszczeniem wyników sprintu, może należeć do właściciela produktu. Co do demonstrowania ukończonej pracy preferuję sytuację, w której każdy członek zespołu deweloperskiego ma szansę praktycznego zademonstrowania działających rzeczy w trakcie przeglądu zamiast wybierania jednej i tej samej osoby do tej roli w każdym kolejnym przeglądzie. Staram się jednak nie przykładać zbytniej wagi do tego, kto będzie demonstrował pracę. Pozwalam podjąć decyzję zespołowi scrumowemu przy założonym celu maksymalizacji zysków z aktywności przeglądu.

## Podejście

Aktywność przeglądu sprintu przedstawiona została na rysunku 21.3.

Danymi wejściowymi do przeprowadzenia przeglądu sprintu są rejestr sprintu i (lub) cel sprintu, a także potencjalnie zdatny do wdrożenia przyrost produktu, który zespół właśnie wytworzył.

Wynikami przeglądu sprintu są poddany pielęgnacji rejestr produktu i uaktualniony plan wersji dystrybucyjnych.



**RYSUNEK 21.3.** Aktywność przeglądu sprintu

Powszechnie stosowane podejście do przeglądu sprintu uwzględnia przedstawienie podsumowania lub też streszczenia tego, co zostało i nie zostało wykonane w odniesieniu do celu sprintu, przedstawienie potencjalnie nadającego się do dystrybucji przyrostu produktu, dyskusję na temat aktualnego stanu produktu oraz przyjęcie dalszego kierunku rozwoju produktu.

## Podsumowanie

Przegląd sprintu rozpoczyna jeden z członków zespołu scrumowego (zazwyczaj właściciel produktu), prezentując cel sprintu, elementy rejestru produktu związane z tym celem, a także opis przyrostu produktu, który został faktycznie osiągnięty podczas minionego sprintu. Te informacje stanowią podsumowanie lub też streszczenie wyników sprintu w odniesieniu do jego celów.

Jeżeli wyniki nie spełniają celu, zespół scrumowy wyjaśnia przyczyny takiej sytuacji. Ważne jest, aby przegląd sprintu nie stał się areną wzajemnego obwiniania. Jeżeli cel nie został osiągnięty, wszyscy uczestnicy powinni powstrzymać się od prób szukania winnych. Celem przeglądu jest opisanie tego, co zostało osiągnięte, i użycie tej informacji do obranego najlepszego kierunku dalszego postępowania.

## Demonstracja

Przegląowi sprintu często nadaje się niesłusznie etyktkę „demonstracji sprintu” lub tylko „demonstracji”. Chociaż demonstracja jest całkiem pomocna w trakcie przeglądu sprintu, to nie ona jest celem całego spotkania.

Najważniejszym aspektem przeglądu sprintu jest analiza i dogłębia dyskusja oraz współpraca uczestników umożliwiająca pojawienie się i wykorzystanie produktywnych propozycji adaptacji. Demonstrowanie tego, co zostało faktycznie zbudowane, jest zwyczajnie bardzo skutecznym sposobem rozpalenia dyskusji wokół czegoś konkretnego. Nic tak nie pobudza rozmowy jak możliwość faktycznego zobaczenia działającej rzeczy.

Zgodnie ze wstępymi ustaleniami jeden lub kilku członków zespołu scrumowego demonstruje wszystkie ważne aspekty przyrostu produktu zbudowanego w trakcie sprintu. W niektórych organizacjach, takich jak firmy produkujące gry komputerowe, bardziej produktywne może okazać się pozwolenie interesariuszom na samodzielne wypróbowanie wyników, na przykład przez zagranie w przyrostową grę stworzoną w trakcie sprintu.

A co jeśli nie ma niczego do zademonstrowania? Jeśli zespół nie zdołał osiągnąć czegokolwiek i faktycznie nie ma co pokazać, przegląd sprintu naprawdopodobniej skupi się na tym, dlaczego nic nie zostało zrobione i jak brak postępów w tym sprincie wpłynie na przyszłe prace. Z drugiej strony, jeśli tego, co zostało zbudowane, nie da się zademonstrować w prosty sposób, stojmy przed innym problemem. Założmy dla przykładu, że zespół zajmował się jedynie pracą związaną z budową architektury (budował „kod spajający”). W takim przypadku zespół deweloperski może twierdzić, że prezentowanie kodu nie ma sensu lub jest niepraktyczne. Taki pogląd jest niemal zawsze nieprawdziwy. Oto dlaczego.

Aby zespół mógł pracować wyłącznie nad „kodem spajającym”, musiałby najpierw przekonać właściciela produktu do umieszczenia w sprincie wyłącznie technicznych (inżynierijnych) elementów rejestru produktu. Jak pisałem w rozdziale 5., dopuszcając takie elementy, właściciel produktu musi wiedzieć, jak stwierdzić, że praca została ukończona. Poza tym większość tworzonych definicji gotowości mówi, że zespół scrumowy powinien rozumieć, w jaki sposób zademonstrować element w trakcie przeglądu sprintu.

Zespół musi posiadać przynajmniej pewien zestaw testów demonstrujących fakt ukończenia pracy w stopniu satysfakcyjnym właściciela produktu. Testy te musiały dać wynik pozytywny, ponieważ w trakcie przeglądu sprintu zespół może pokazać jedynie prace, które zostały ukończone. Czyli pierwsza, najprostsza rzecz, jaką można zrobić, to zademonstrować te testy w trakcie przeglądu sprintu. Zazwyczaj jednak jeśli członkowie zespołu zastanowią się trochę nad tym problemem, mogą znaleźć znacznie lepszy sposób. Fakt, iż coś jest trudne do zademonstrowania, nie usprawiedliwia dostatecznie wykluczenia tego czegoś z prezentacji.

## Dyskusja

Demonstracja przyrostu produktu pozwala na przeprowadzenie dogłębnej dyskusji. Wszyscy uczestnicy spotkania są zachęcani do dzielenia się obserwacjami, uwagami i do uczestniczenia w rozmowie na temat produktu oraz jego kierunku. Przegląd sprintu nie jest jednak aktywnością mającą na celu rozwiązywanie problemów — do tego celu służy inne spotkanie.

Zdrowa dyskusja pozwala uczestnikom, którzy nie są członkami zespołu scrumowego, na zadawanie pytań, zrozumienie aktualnego stanu produktu, a także pomaganie w wyznaczeniu dalszego kierunku rozwoju. W tym samym czasie członkowie zespołu scrumowego zyskują lepsze zrozumienie biznesowej i marketingowej strony produktu dzięki informacjom zwrotnym mówiącym o jego zbieżności z celem, jakim jest zachwycenie klientów i użytkowników.

## Adaptacja

Dzięki demonstracji i dyskusji zespół jest w stanie zadawać pytania i odpowiadać na nie. Oto co może być przedmiotem takich pytań:

- Czy interesariuszom spodobało się to, co widzieli?
- Czy chcieliby zobaczyć jakieś zmiany?
- Czy to, co budujemy, jest nadal dobrym pomysłem na rysunku lub dla naszych wewnętrznych klientów?
- Czy nie brakuje nam jakiejś istotnej cechy?
- Czy nie inwestujemy (wkładamy niepotrzebny wysiłek deweloperski) w cechę, której nie potrzebujemy?

Zadawanie tego typu pytań i odpowiadanie na nie pozwala uzyskać informacje będące podstawą do zaadaptowania rejestru produktu i planów wersji dystrybucyjnej.

W rozdziale 6. opisałem naturalny proces pielęgnacji rejestru produktu w trakcie przeglądu sprintu. W miarę lepszego rozumienia przez wszystkich bieżącego wysiłku deweloperskiego i jego kierunku w rejestrze często tworzone są nowe elementy, a istniejącym nadaje się inne priorytety lub usuwa, gdyż nie są dłużej potrzebne. Pielęgnacja może również wpływać na to, co zespół będzie robił w następnym sprincie.

W rozdziale 18. napisałem, że pielęgnacja przeprowadzana w trakcie przeglądu sprintu może mieć również wpływ na szerszy plan wersji dystrybucyjnej. Na przykład opierając się na dyskusji przeprowadzonej w trakcie przeglądu i jej konkluzjach, możemy zmienić jedną z kluczowych zmiennych planowania wersji: zakres, czas lub budżet. Być może w wyniku przeglądu aktualnego stanu produktu zdecydujemy się przerwać prace nad jedną z kluczowych cech (zmienić zakres). Taka decyzja bez wątpienia wpłynie na bieżący plan wersji.

Przegląd sprintu daje nam szansę identyfikacji sposobów adaptowania się i reagowania na zmiany w chwili, kiedy nas na to jeszcze stać — pod koniec każdego sprintu.

## Problemy przeglądu sprintu

Przeglądy sprintu nie są pozbawione problemów. Pracując z wieloma organizacjami używającymi Scruma, zauważałem kilka powtarzających się bolączek, dotyczących głównie zatwierdzania rzeczy w trakcie spotkania, małej frekwencji i dużych projektów.

## Zatwierdzanie

Zatwierdzanie w trakcie przeglądu sprintu może być źródłem problemów. Pierwsze pytanie brzmi, czy przeglądy sprintu są odpowiednim miejscem do zatwierdzania (akceptowania) elementów rejestru produktu? Jak wspomniałem wcześniej, zanim jeszcze zacznie się przegląd sprintu, właściciel produktu musi przejrzeć prace, aby stwierdzić, czy faktycznie zostały one wykonane (spełniają uzgodnioną wcześniej definicję ukończenia). W związku z tym przegląd sprintu nie powinien być formalnym wydarzeniem akceptowania pracy. Elementy rejestru produktu powinny zostać przyjęte przez właściciela produktu przed przeglądem.

Załóżmy jednak, że w trakcie przeglądu sprintu jeden z interesariuszy zajmujących w organizacji wysokie stanowiska nie zgadza się co do stanu cechy — jego zdaniem nie została ona ukończona. Mimo że taka opinia jest cenna, uważam, że skoro właściciel produktu zadeklarował wykonaną pracę jako ukończoną, jest ona ukończona. W rozdziale 9. pisałem o uprawnieniach właściciela produktu jako centralnego punktu dowodzenia produktem. Aby ten centralny punkt obowiązywał w praktyce, właściciel produktu musi posiadać pełne prawo do ostatecznego akceptowania lub odrzucania pracy i nie może być ono usurpowane przez jakiegokolwiek innego uczestnika przeglądu sprintu — niezależnie od tego, z jak wysokiego szczebla pochodzi ten uczestnik.

Nie oznacza to wcale, że właściciel produktu powinien ignorować uwagi na temat niezgodności cech z oczekiwaniemi interesariuszy. Kiedy taka sytuacja nastąpi, właściwym sposobem postępowania jest przewidzenie zmiany dla danej cechy poprzez stworzenie elementu rejestru produktu odzwierciedlającego nowe zachowanie, jakiego spodziewa się interesariusz, i wstawienie go do rejestru celem przygotowania do realizacji w jednym z nadchodzących sprintów. Właściciel produktu powinien również przeanalizować taki przypadek, aby stwierdzić, dlaczego rozminął się z interesariuszami, i wprowadzić niezbędne zmiany, dzięki którym taka sytuacja nie powtórzy się.

## Mała frekwencja

Przegląd sprintu należy traktować jako ważną aktywność inspekcyjno-adaptacyjną, na którą warto poświęcić swój czas. Mimo to wiele organizacji cierpi z powodu słabej frekwencji.

Jedną z głównych przyczyn słabego uczestnictwa jest zbyt napięty kalendarz interesariuszy mających na głowie inne zobowiązania o „wysokim priorytecie”, uniemożliwiające im uczestniczenie w przeglądach sprintu. Przeładowanie harmonogramu zajęć w stopniu zmuszającym do porzucania pewnych zobowiązań jest wyraźnym sygnałem złego funkcjonowania organizacji. W takich przypadkach rekomenduję zaprzestanie pracy nad produktami o niskim priorytecie do momentu, kiedy staną się one na tyle ważne, że interesariusze zaczną pojawiać się na ich przeglądach. Jeśli taki dzień nigdy nie nastąpi, będzie to oznaczało, że te produkty o niskiej wartości nigdy nie były tak cenne (w odniesieniu do innych produktów w portfelu), aby warto było nad nimi pracować.

Czasami sporadyczna frekwencja jest wynikiem braku przekonania ludzi do tego, że zespół scrumowy jest w stanie wyprodukować cokolwiek wartego przejrzenia w przeciągu kilku tygodniu. To zdanie jest szczególnie prawdziwe w odniesieniu do firm, które zaczynają pracę w Scrumie. Interesariusze są przyzwyczajeni do znacznie dłuższych okresów pomiędzy przeglądami, a przeglądy, w których brali do tej pory udział, mogły być rozczarowujące.

Najlepszym sposobem poradzenia sobie z tym problemem jest budowanie w każdym sprintie potencjalnie zdatnego do wdrożenia przyrostu produktu. Kiedy zespoły działają w ten sposób, większość osób zdaje sobie sprawę, że częste przeglądy są warte ich czasu i pozwalają im udzielać szybkiej informacji zwrotnej, którą zespół może wykorzystać w praktyce.

## Duże projekty

W przypadku dużego wysiłku deweloperskiego obejmującego wiele zespołów scrumowych rozsądne może okazać się przeprowadzanie wspólnego przeglądu sprintu. Jest to przegląd obejmujący pracę zrealizowaną przez ściśle powiązane ze sobą zespoły.

Takie podejście ma kilka zalet. Po pierwsze, interesariusze uczestniczą tylko w jednym przeglądzie sprintu zamiast w kilku. Po drugie, jeżeli praca tych zespołów miała zostać zintegrowana, dobrze będzie, jeśli przegląd skupi się na tym, co zostało faktycznie zintegrowane, a nie na kolekcji samodzielnych przyrostów produktu. Aby osiągnąć taki cel, wszystkie zespoły muszą zadbać o to, aby ich kryteria ukończenia uwzględniały testowanie integracyjne (które tak czy inaczej powinno mieć miejsce).

Wadą przeprowadzania wspólnego przeglądu sprintu dla wielu zespołów jest to, że samo spotkanie trwa dłużej i może wymagać większej sali konferencyjnej, niż potrzebna byłaby dla pojedynczego zespołu scrumowego.

## Zakończenie

W tym rozdziale podkreśliłem cel przeglądu sprintu jako kluczowej pętli pozyskiwania wiedzy w trakcie prac deweloperskich przy użyciu Scruma. Przegląd sprintu dotyczy szerokiego kręgu uczestników, którzy mają za zadanie dokonać inspekcji i adaptacji bieżącego produktu. Chociaż przegląd sprintu jest aktywnością nieformalną, zespół scrumowy przeprowadza pewne minimalne przygotowania pozwalające zapewnić wartościowe wyniki. Podczas przeglądu sprintu zespół scrumowy streszcza, co działało się w trakcie sprintu i jakie rzeczy udało się zrealizować, a także demonstruje przyrost produktu będący wynikiem wykonania sprintu. Uczestnicy spotkania prowadzą żywiołową dyskusję — zachęca się wszystkich do zadawania pytań, dzielenia się swoimi obserwacjami i sugestiami. Dzięki tej dyskusji można przeprowadzić pielęgnację rejestru produktu i uaktualnić plan wersji dystrybucyjnej.

W następnym rozdziale skupię się na aktywności inspekcjno-adaptacyjnej samego procesu, czyli na retrospekcji sprintu.



## Rozdział 22

# RETROSPEKCJA SPRINTU

---

Scrum przewiduje dwie aktywności inspekcyjno-adaptacyjne pod koniec każdego sprintu: przegląd sprintu i retrospekcję sprintu. W poprzednim rozdziale opisałem przegląd sprintu, w trakcie którego zespół i interesariusze dokonują inspekcji samego produktu. Przenieśmy teraz naszą uwagę na retrospekcję sprintu, podczas której zespół scrumowy analizuje proces używany do wytwarzania produktu.

Zacznę od przedstawienia ogólnego celu retrospekcji sprintu i jej uczestników. Następnie opiszę potrzebne przygotowania oraz główne zadania związane z retrospekcją sprintu. Najważniejsze z tych zadań mają miejsce po retrospekcji, kiedy uczestnicy wdrażają zidentyfikowane poprawki.

## Wprowadzenie

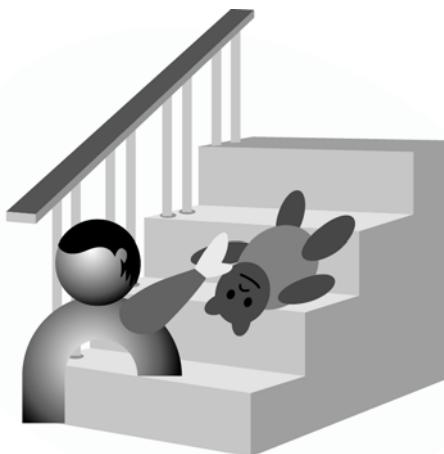
Norm Kerth — wynalazca nowoczesnego podejścia do retrospekcji — we wstępie do swojej książki *Project Retrospectives*<sup>1</sup> podsumowuje cel retrospekcji, cytując fragment z *Kubusia Puchatka* [Kerth, 2001]:

*Przedstawiam wam Misia Puchatka, który właśnie w tej chwili schodzi po schodach. Tuk-tuk, tuk-tuk, zsuwa się Puchatek na grzbiecie, do góry nogami, w tyle za Krzysiem, który go ciągnie za przednią łapkę. Odkąd Puchatek siebie pamięta, jest to jedyny sposób schodzenia ze schodów, choć Miś czuje czasami, że mógłby to robić zupełnie inaczej, gdyby udało mu się przestać tuktać choćby jedną chwilę i dobrze się nad tym zastanowić.*

Retrospekcje sprintów dają całemu zespołowi scrumowemu szansę zaprzestania ciągłego obijania się o schody i pomyślenia przez chwilę (patrz rysunek 22.1). W ograniczonym czasie poświęconym na retrospekcję zespoły mogą spokojnie prześledzić to, co się wydarzyło, przeanalizować sposób wykonywania pracy, zidentyfikować sposoby poprawienia sytuacji i przygotować plany wdrożenia poprawek. Wszystko, co ma wpływ na sposób tworzenia produktu przez zespół, może być przedmiotem badania i dyskusji, włączając w to wszelkie procesy, praktyki, metody komunikacji, środowisko, artefakty, narzędzia itd.

---

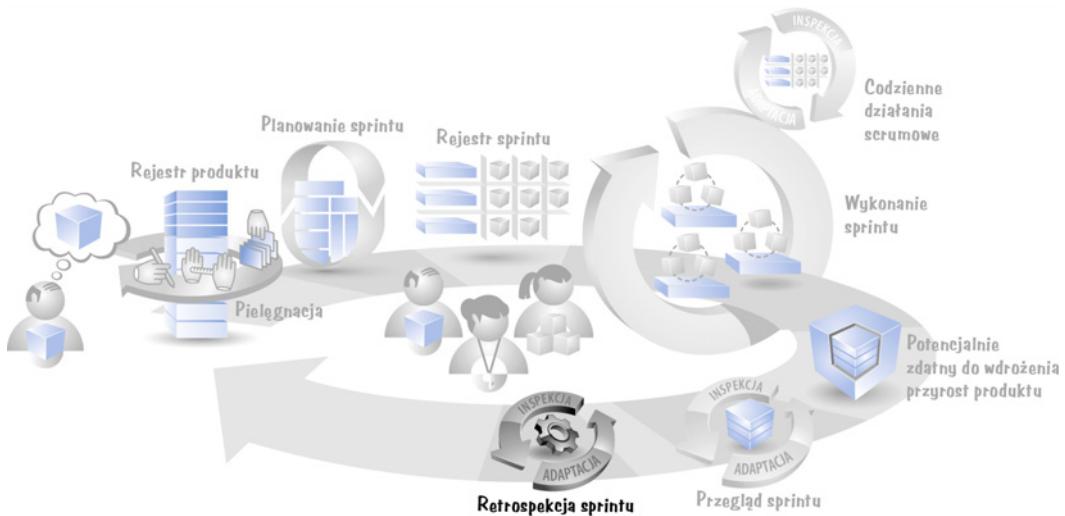
<sup>1</sup> Restrospekcje projektów — przyp. tłum.



**RYSUNEK 22.1.** Kubuś Puchatek ilustrujący potrzebę retrospekcji

Retrospekcja sprintu jest jedną z najistotniejszych i najmniej docenianych praktyk w środowisku scrumowym. Jej waga wynika z faktu, iż daje zespołowi szansę dostosowania Scruma do swoich własnych, unikatowych warunków. Ignorowanie tej aktywności wynika ze źle uformowanych poglądów ludzi, zdaniem których zabiera ona cenny czas przeznaczony na „prawdziwą” pracę, czyli projektowanie, budowanie i testowanie.

Retrospekcja sprintu jest kluczowym czynnikiem nieustannego poprawiania procesu (jednej z zalet Scruma). Chociaż niektóre organizacje wstrzymują się z retrospekcją do końca całego projektu, zespoły scrumowe dokonują retrospekcji w każdym sprincie (patrz rysunek 22.2), co pozwala im skorzystać z doświadczeń i danych, zanim te ulegną zatarciu.



**RYSUNEK 22.2.** Kiedy ma miejsce retrospekcja sprintu

Dzięki spotykaniu się pod koniec każdego sprintu w celu przeprowadzenia inspekcji i adaptacji swojego procesu scrumowego zespół może stosować metodę wczesnego i przyrostowego pozyskiwania wiedzy w trakcie całego procesu deweloperskiego i w ten sposób znacząco wpływać na wynik całego projektu.

W pozostałej części rozdziału opiszę szczegółowo podejście do wykonania retrospekcji sprintu. Nie pozwól jednak, aby szczegóły doprowadziły Cię do przekonania, że retrospekcja to jakaś rozbudowana ceremonia. Aktywność ta może sprowadzać się do zwykłego spotkania członków zespołu i przedyskutowania pytań takich jak:

- Co zadziałało dobrze w sprincie i jest warte kontynuowania?
- Co nie zadziałało w sprincie i w związku z tym nie powinniśmy tego dłużej robić?
- Co powinniśmy zacząć robić lub poprawiać?

W oparciu o przeprowadzone dyskusje członkowie zespołu wskazują kilka możliwych poprawek, które wprowadzają stopniowo do swojego procesu.

## Uczestnicy

Ponieważ retrospekcja sprintu jest czasem refleksji nad procesem, niezbędne jest uczestnictwo całego zespołu scrumowego — wszystkich członków zespołu deweloperskiego (projektujących, budujących i testujących produkt), mistrza młyna i właściciela produktu. Każda z tych osób ma swój własny sposób postrzegania. Suma tych różnorodnych perspektyw ma kluczowe znaczenie podczas identyfikowania ulepszeń procesu.

Mistrz młyna uczestniczy w spotkaniu zarówno jako integralny uczestnik całego procesu, ale również jako autorytet w dziedzinie procesu dla zespołu scrumowego (patrz rozdział 10.). Bycie autorytetem nie oznacza, że mistrz młyna powinien mówić zespołowi, w jaki sposób należy zmienić proces. Przeciwnie, mistrz młyna może wskazać, gdzie zespół nie przestrzega ustalonego przez siebie procesu. Jest on również cennym źródłem wiedzy i pomysłów dla zespołu.

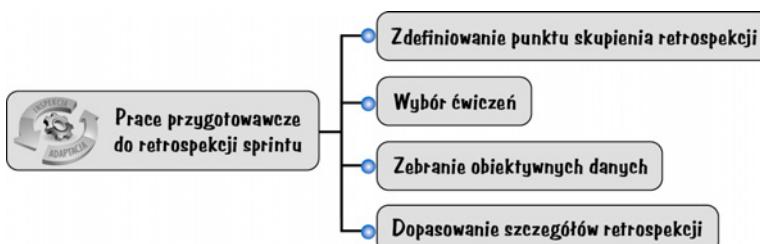
Niektórzy przekonują, że obecność właściciela produktu na retrospekcji sprintu może stanowić przeszkodę do pełnej szczerości pozostałych uczestników i zniechęcać ich do ujawniania trudnych problemów. Chociaż istnieje ryzyko takiego zachowania w pewnych organizacjach, właściciel produktu jest kluczowym elementem procesu scrumowego i jako taki powinien być uwzględniony w dyskusjach na temat tego procesu. Jeżeli nie ma zaufania pomiędzy właścicielem produktu i zespołem deweloperskim lub też istnieje małe poczucie komfortu zniechęcające do mówienia w sposób otwarty, być może właściciel produktu powinien powstrzymać się od uczestnictwa w spotkaniach do momentu, kiedy mistrz młyna wytrenuje wszystkie zaangażowane osoby w kierunku tworzenia bardziej komfortowego środowiska o większym poziomie wzajemnej ufności.

Zakładając istnienie środowiska zaufania i bezpieczeństwa, efektywnie działający właściciel produktu ma kluczowe znaczenie dla szybkiego i elastycznego przepływu wartości biznesowej i w związku z tym powinien brać udział w retrospekcji sprintu. Właściciel produktu jest kanałem, poprzez który wymagania płyną do zespołu. A jeśli coś jest nie tak z przepływem wymagań w procesie scrumowym? Być może elementy rejestru produktu nie są dostatecznie dobrze pielęgnowane na początku planowania sprintu. Zespółowi trudno będzie wypracować ewentualną poprawkę procesu przy nieobecności właściciela produktu.

Z drugiej strony, interesariusze lub menedżerowie niebędący członkami zespołu scrumowego powinni uczestniczyć w retrospekcji sprintu wyłącznie, jeśli zostaną zaproszeni przez zespół. Choć przejrzystość jest podstawową wartością w Scrumie, w rzeczywistości mało która organizacja osiąga poziom zaufania pozwalający osobom niebędącym w zespole scrumowym na regularne uczestniczenie w jego retrospekcjach. Członkowie zespołu muszą mieć poczucie bezpieczeństwa, jeśli mają przeprowadzić szczerą dyskusję bez zniechęcenia wywołanego obecnością kogoś z zewnątrz. Jeśli zespół nie ma dostatecznego komfortu do wyjawiania prawdziwych problemów ze względu na obecność osób trzecich, retrospekcje stracą na efektywności.

## Przygotowania

Przed rozpoczęciem retrospekcji sprintu trzeba przeprowadzić pewne przygotowania (patrz rysunek 22.3).



**RYSUNEK 22.3.** Prace przygotowawcze do retrospekcji sprintu

W przypadku sprintów o krótkim czasie trwania lub zespołów stosujących dobrze funkcjonującą, prostą formę retrospekcji przygotowania powinny zająć bardzo małą ilość czasu.

### Zdefiniowanie punktu skupienia retrospekcji

Każda retrospekcja powinna posiadać dobrze zdefiniowany punkt skupienia. Typowym punktem skupienia jest przejrzenie wszystkich aspektów procesu zespołu scrumowego użytego podczas bieżącego sprintu. Niemniej jednak czasami zespół może wybrać inny temat, opierając się na tym, co aktualnie ma duże znaczenie dla zespołu i gdzie wszyscy widzą potrzebę wprowadzenia usprawnień. Na przykład:

- Poprawa własnych umiejętności w zakresie programowania sterowanego testami.
- Sprawdzenie, dlaczego budujemy to, co uważamy za oczekiwane przez klientów, a po prezentacji okazuje się często, że ich zdaniem źle zrozumieliśmy ich potrzeby lub pominęliśmy bardzo istotny fakt zawarty w wymaganiach.

Ustalenie i zakomunikowanie punktu skupienia przed rozpoczęciem retrospekcji pozwala zespołowi scrumowemu stwierdzić, czy ktokolwiek spoza zespołu powinien zostać zaproszony na spotkanie. Poza tym znajomość tematu retrospekcji przed jej rozpoczęciem pozwala zespołowi wybrać odpowiednie ćwiczenia retrospekcjne i daje czas na zebranie niezbędnych danych umożliwiających płynne przeprowadzenie tej aktywności.

Posiadanie prawa wyboru konkretnego punktu skupienia może pomóc długowiecznym, wydajnym zespołom scrumowym w nieustannym wydobywaniu mieralnej wartości z retrospekcji sprintów. Na przykład w pewnej organizacji, którą trenowałem, był doświadczony, dojrzały zespół scrumowy, którego członkowie bardzo dobrze współpracowali ze sobą od niemal trzech lat. Przeszli razem przez wiele sprintów i zaczęli dochodzić do wniosku, że przeprowadzanie retrospekcji skupionej wyłącznie na bieżącym sprincie miało często niską wartość. Jeden z członków zespołu zauważał: „Przez długi czas retrospekcje sprintów wydawały się nieodzowne, ale teraz często mam wrażenie, jakby były jedynie sztuką dla sztuki”. Od tego momentu zaczęliśmy wykonywać krótsze, bardziej zwięzłe retrospekcje, pozwalające zespołowi i zaproszonym gościom na wgryzanie się w bardzo specyficzne problemy i dociekanie sedna ich przyczyn. Dzięki temu zespół kontynuował zdobywanie wiedzy i poprawianie swoich umiejętności mimo bogatego doświadczenia w Scrumie. Zawsze istnieje miejsce na wzrost, tyle że jego pobudzanie może wymagać bardziej skupionych retrospekcji.

## Wybór ćwiczeń

Po ustaleniu punktu skupienia i uczestników nadchodzącej retrospekcji sprintu możemy wybrać ćwiczenia mające na celu pomóc uczestnikom w zaangażowaniu, myśleniu, eksploracji i wspólnym podejmowaniu decyzji. W trakcie typowej retrospekcji przeprowadzane są następujące ćwiczenia:

- stworzenie i prześledzenie linii zdarzeń sprintu,
- burza mózgów ujawniająca spostrzeżenia,
- grupowanie i głosowanie nad spostrzeżeniami.

Możemy zdecydować o zmianie wykonywanych ćwiczeń celem wsparcia określonego punktu skupienia lub określonej grupy uczestników. Możemy również spróbować wykonać nowe ćwiczenia, aby wprowadzić element świeżości do procesu. Dodatkowe ćwiczenia znajdziesz w opracowaniach *Project Retrospectives* [Kerth, 2001] i *Agile Retrospectives* [Derby i Larsen, 2006].

Uczestnicy nie muszą decydować już na wstępie, które ćwiczenia będą realizowane. W praktyce o wiele lepsze może okazać się dobieranie ćwiczeń w samą porę podczas retrospekcji w oparciu o to, co zdaniem uczestników może najlepiej zadziałać w danej sytuacji. W trakcie przygotowań dobrze jest wskazać te ćwiczenia, które wymagać będą danych lub innego zaopatrzenia. Bądź przygotowany, ale zachowaj elastyczność.

## Zebranie obiektywnych danych

Ponieważ retrospekcja sprintu jest wykonywana w krótkim czasie (wiele zespołów przyjmuje ograniczenie czasowe), wszelkie dłuższe prace niezbędne do zebrania danych powinny mieć miejsce przed rozpoczęciem retrospekcji.

Znamy zarówno punkt skupienia, jak i możliwe ćwiczenia dla nadchodzącej retrospekcji, powinniśmy zatem mieć dobre pojęcie na temat ewentualnych danych, które trzeba zebrać (może być tak, że żadne dane nie będą potrzebne). Dane obiektywne są sztywne (nie podlegają negocjacjom). Może to być na przykład lista zdarzeń w sprincie połączona z czasem ich wystąpienia, liczba elementów rejestru produktu, które zostały rozpoczęte, ale nie ukończone, wykres rozpalania cech dla sprintu ilustrujący przepływ ukończonej pracy. W tej konkretnej chwili nie zajmujemy się zorganizowaniem lub analizą danych, naszym zadaniem jest jedynie zebranie informacji, tak aby były one dostępne podczas retrospekcji.

## Dopasowanie szczegółów retrospekcji

Retrospekcje sprintów, podobnie jak przeglądy, mają miejsce pod koniec każdego sprintu, często bezpośrednio po zakończeniu przeglądu sprintu. W każdym sprintie powinny odbywać się w tym samym miejscu i tego samego dnia. Niemniej jednak, w przeciwieństwie do przeglądu sprintu, od czasu do czasu konieczna może być zmiana miejsca, dnia lub czasu trwania konkretnej retrospekcji, aby zapewnić lepszą realizację punktu skupienia, dostosowanie do uczestników spoza zespołu lub realizację specyficznych ćwiczeń zaplanowanych na to spotkanie.

Dokładny czas trwania retrospekcji zależy od takich czynników jak liczba osób w zespole, staż zespołu, stopień kolokacji zespołu itd. Z mojego doświadczenia wynika, że nowe zespoły mają tendencję do poświęcenia zbyt małej ilości swojego czasu na retrospekcje. Trudno jest przeprowadzić spotkanie, które będzie miało jakieś produktywne wyniki, w mniej niż 60 minut. Ja zawsze przeznaczam na tę aktywność około półtorej godziny w przypadku sprintów dwutygodniowych i proporcjonalnie więcej dla dłuższych sprintów.

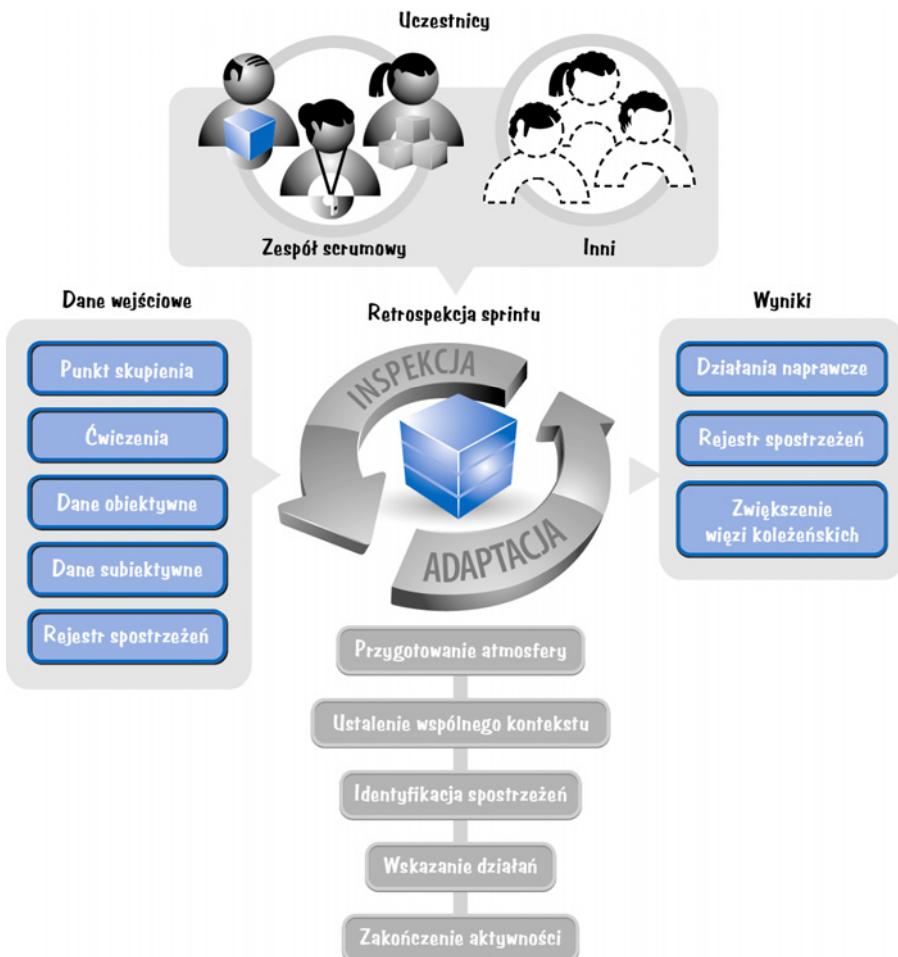
Zespół scrumowy powinien wybrać lokalizację retrospekcji, która będzie jak najbardziej pomocna w wypracowaniu pozytywnych wyników. Niektóre zespoły decydują się na przeprowadzenie retrospekcji w swoim miejscu pracy, gdzie wiszą ich duże tablice i wykresy. Dzięki temu mają dostęp do zasobów właściwej informacji. Inne z kolei wolą spotkać się z dala od tego miejsca, by móc w celu uzyskania środowiska o mniejszym nasyceniu emocjonalnym, w którym wszyscy będą mieli mniej zahamowań i będą mówić bardziej otwarcie. Jeszcze raz podkreślę, że samo miejsce nie ma takiego znaczenia, jak stworzenie bezpiecznego środowiska, w którym członkowie zespołu będą mogli swobodnie mówić, co myślą.

Najczęściej animatorem retrospekcji sprintu jest mistrz młyna, ale rola ta równie dobrze może zostać powierzona dowolnemu innemu członkowi zespołu, który czuje się na siłach do jej wypełnienia. W pewnych przypadkach, kiedy poruszane są drażliwe tematy i animator współpracujący ściśle z zespołem może być mało efektywny, pomocne może okazać się ściągnięcie wyszkolonej, neutralnej osoby z zewnątrz do poprowadzenia spotkania. W organizacjach dysponujących wieloma zespołami, z których każdy posiada własnego mistrza młyna, bardzo odkrywcze może okazać się przeprowadzanie retrospekcji przez mistrzów młyna w innych zespołach niż ich własny. Kto będzie animatorem retrospekcji sprintu, powinno zostać ustalone w czasie przygotowań do tej aktywności.

## Podejście

Aktywność retrospekcji sprintu ilustruje rysunek 22.4.

Dane wejściowe do retrospekcji sprintu obejmują uzgodniony wcześniej punkt skupienia, a także wszelkie ćwiczenia i materiały, jakich zespół może chcieć używać w trakcie retrospekcji. Oprócz tego większość retrospekcji wymaga pewnych przygotowanych wcześniej danych obiektywnych. Tym, czego można się bez wątpienia spodziewać wśród danych wejściowych, są subiektywne opinie dotyczące bieżącego sprintu wnoszone przez każdego członka zespołu. Do danych wejściowych zalicza się również rejestr spostrzeżeń wyprodukowany podczas poprzednich retrospekcji.



**RYSUNEK 22.4.** Aktywność retrospekcji sprintu

W wyniku retrospekcji powinna powstać lista konkretnych działań naprawczych, które zespół zgodził się wykonać w ciągu następnego sprintu. Może również powstać rejestr spostrzeżeń zebrań w trakcie bieżącej retrospekcji, którymi zespół nie będzie się zajmował w kolejnym sprincie, ale być może weźmie je pod uwagę w przyszłości. W wyniku retrospekcji członkowie zespołu powinni również spodziewać się poprawy wzajemnych stosunków koleżeńskich.

Chociaż istnieje wiele różnych podejść do retrospekcji, celem większości jest znalezienie odpowiedzi na następujące pytania:

- Co poszło dobrze w tym sprincie, więc chcielibyśmy kontynuować to w przyszłości?
- Co nie poszło dobrze w tym sprincie i w związku z tym powinniśmy przestać robić to w przyszłości?
- Co powinniśmy zacząć robić lub poprawić?

Podejście (podobne do opisanego w opracowaniu [Derby i Larsen, 2006]), które ja uznaję za użyteczne, polega na przygotowaniu odpowiedniej atmosfery dla retrospekcji, utworzeniu wspólnego kontekstu pomiędzy uczestnikami, zidentyfikowaniu spostrzeżeń, które mogą przełożyć się na poprawki, wskazaniu konkretnych działań naprawczych do wykonania w następnym sprintie i zakończeniu spotkania.

Kroki te pokazuje rysunek 22.4. Kolejne sekcje będą wyjaśniać każdy z nich.

## Przygotowanie odpowiedniej atmosfery

Podczas retrospekcji ludzie są proszeni o przeanalizowanie zachowania i wydajności swojego zespołu oraz wskazanie konkretnych rekomendacji prowadzących do jeszcze lepszych wyników w przyszłości. Umieszczenie zespołu (a tym samym również siebie) pod mikroskopem może nie być przyjemnym doświadczeniem. Zatem dobrym sposobem rozpoczęcia retrospekcji jest wprowadzenie atmosfery, dzięki której każdy będzie miał poczucie komfortu w trakcie tego spotkania.

Ludzie muszą czuć, że mogą wyrażać swoje opinie bez strachu o możliwą karę. Zespół powinien ustalić pewną regułę lub też porozumienie, które jasno stwierdza, że wyrażenie opinii i wyciąganie na światło dzienne wszelkich niejasności jest bezpieczne. Pomocne jest również jednoznaczne wskazanie w tej regule, iż celem spotkania jest skupienie się na systemie organizacyjnym i procesie, a nie na poszczególnych pracownikach i w związku z tym zupełnie akceptowalna jest eksploracja rzeczy, które posłyły źle.

Zdarzą się przypadki, kiedy problemy będą miały charakter personalny. Retrospekcja nie jest miejscem na ich rozwiązywanie. Celem tego spotkania jest poprawianie procesu zespołu scrumowego, a nie przypisywanie win lub też zwracanie uwagi poszczególnym osobom. Ustanawiając atmosferę spotkania, zadaj o to, aby podstawowe zasady spotkania wspierały koncepcję środowiska wolnego od wzajemnego obwiniania.

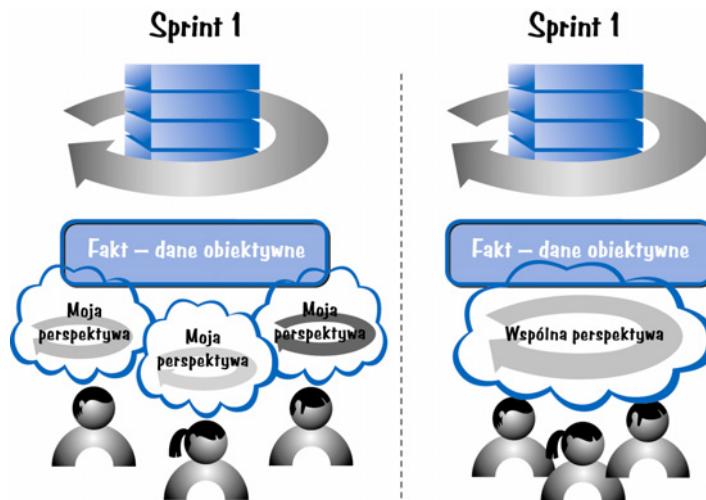
Równie ważne jest zachęcenie wszystkich do aktywnego uczestniczenia w spotkaniu. Retrospekcja będzie miała małą skuteczność, jeśli ludzie przyjmą rolę pasywną. Zatem tworząc atmosferę, dobrze jest zachęcić ludzi do mówienia. W niektórych zespołach odbywa się to przez poproszenie każdego uczestnika osobno o powiedzenie kilku słów o swych aktualnych odczuciach i poziomie energii. To, jakie pytanie zostanie konkretnie zadane, nie jest aż tak istotne — chodzi raczej o zaproszenie do powiedzenia czegoś i tym samym wprowadzenie w nastrój rozmowy.

## Ustalenie wspólnego kontekstu

Grupa ludzi może przeżyć to samo doświadczenie, a jednak interpretować je w zupełnie odmienny sposób. Aby dokonać skutecznej inspekcji bieżącego sprintu, wszyscy muszą osiągnąć ten sam poziom zrozumienia (znaleźć się w tym samym kontekście co pozostali uczestnicy).

Ustalenie wspólnego kontekstu wymaga przystosowania indywidualnego (odmiennego) kontekstu każdego uczestnika (patrz lewa strona rysunku 22.5) do wspólnej perspektywy zespołu (patrz prawa strona rysunku 22.5).

Lewa strona rysunku 22.5 pokazuje, że każda osoba może postrzegać sprint inaczej w oparciu o swoje własne doświadczenia w jego trakcie i nie dostrzegać całościowego obrazu zdarzeń, osiągnięć i porażek. Jeżeli pozwoli się na dominowanie indywidualnych opinii, retrospekcja może zamienić się w debatę nad tymi opiniami, zamiast skupić się tym, co można poprawić dzięki wspólnemu dostrzeżeniu problemów.



**RYSUNEK 22.5.** Równoważenie perspektyw w celu uzyskania wspólnego kontekstu

Ustalając wspólny kontekst, należy obowiązkowo zadbać o to, aby retrospekcja bazowała na obiektywnym, całościowym obrazie sprintu. Po wprowadzeniu wszystkich w nastrój dyskusji przedstaw obiektywne dane, takie jak elementy rejestru produktu, które zespół zobowiązał się zrealizować, elementy ukończone, liczba zgłoszonych błędów itd. To, jakie dokładnie dane zostaną przedstawione, zależy będzie od punktu skupienia retrospekcji. Chociaż większość obiektywnych danych zazwyczaj zbiera się w trakcie przygotowań do retrospekcji, niektóre z nich mogą zostać zszytezowane wspólnie przez uczestników w jej trakcie. Takie działanie doda energii zespołowi do skupienia się nad ważnymi informacjami. Niezależnie od tego, czy obiektywne dane zostaną zebrane przed retrospekcją, czy też w jej trakcie, sam fakt ich zebrania ma kluczowe znaczenie dla ustalenia wspólnego fundamentu na faktach, a nie na opiniach.

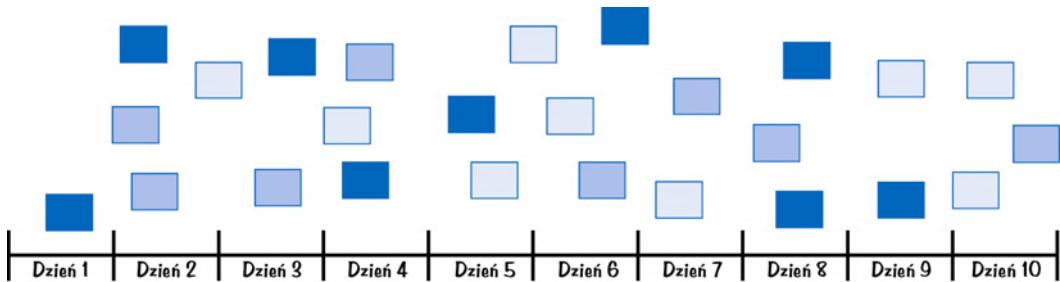
Oparcie na danych obiektywnych nie oznacza wcale, że dane subiektywne są bez znaczenia. Ka da osoba wnosi na spotkanie swoje subiektywne dane odwzorowujące osobist  interpretacj  sprintu. Je eli dane subiektywne nie zostan  ujawnione i przedyskutowane, dany uczestnik spotkania mo e uzna ,  e pozosta i prze ili sprint w podobny sposób. To niedopasowanie w snego kontekstu mo e utrudni  ludziom zrozumienie wzajemnych uwag i sugestii.

Istnieje szereg  wicze , jakie uczestnicy mo e wykona , aby wypracować wspólny kontekst zarówno odnośnie do obiektywnych, jak i subiektywnych danych. Dwa najcz ciej spotykane to linia zdarze  i sejsmograf emocji.

### Linia zdarze 

Tworzenie **linii zdarze ** jest prost , a jednocze nie bardzo skuteczn  metod  budowania wspólnego artefaktu, który przedstawia w sposób wizualny przebieg zdarze  w sprintie. Zdarzenia mo e by  zapisane tak: „Popsuta kompilacja”, „Przerwanie w celu usuni cia usterki produkcyjnej”, „Sabina wr ci a z urlopu”.

Najcz ciej zaczynamy od narysowania linii czasu na tablicy i pozwalamy uczestnikom umie ci na niej karteczki samoprzyklepne reprezentuj ce znaczące zdarzenia, które miały miejsce podczas sprintu (patrz rysunek 22.6). Zespo y, których czonkowie nie pracuj  w jednym miejscu, mo g  do tego celu wykorzystać tablic  wspo dzielon  sieciowo.



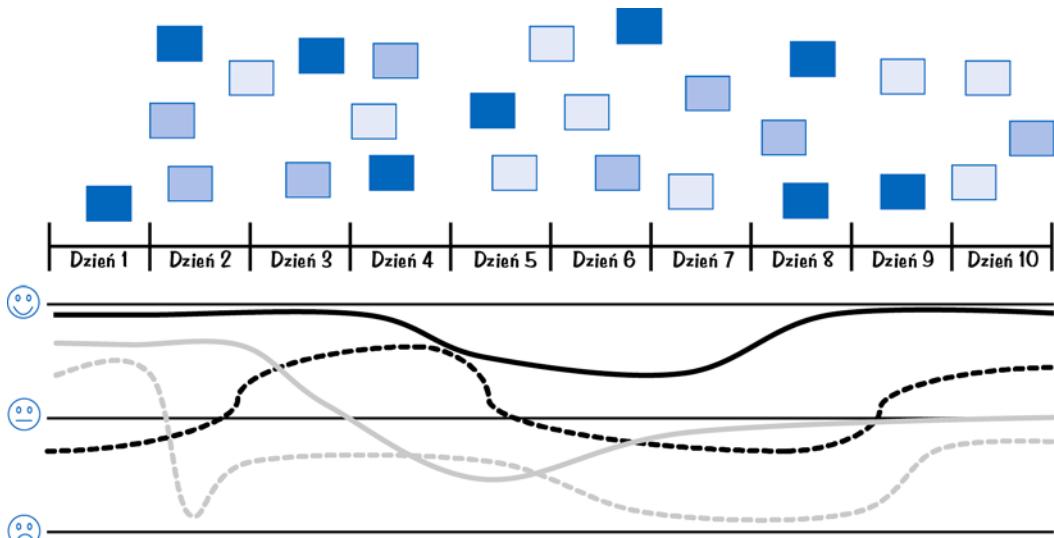
**RYSUNEK 22.6.** Linia zdarzeń sprintu

Karteczki są umieszczone na tablicy w sposób chronologiczny. Takie czasowe ułożenie zdarzeń doskonale wizualizuje przepływ aktywności w trakcie sprintu i dostarcza wspólnego kontekstu pozwalającego na szybką identyfikację brakujących lub zapomnianych zdarzeń.

Niektóre zespoły stosują karteczki w różnych kolorach reprezentujące różne typy zdarzeń (na przykład zielona karteczka to zdarzenie o charakterze technicznym, żółta to zdarzenie o charakterze organizacyjnym, a czerwona to zdarzenie o charakterze osobistym). Inne zespoły używają różnych kolorów do przedstawienia poziomu energii lub uczuć (na przykład karteczka zielona to zdarzenie pozytywne, żółta to zdarzenie neutralne, a czerwona negatywne).

### Sejsmograf emocji

Wiele zespołów oprócz linii zdarzeń tworzy również **sejsmografię emocji**. Jest to graficzne przedstawienie emocjonalnych wzlotów i upadków uczestników w trakcie trwania sprintu (patrz rysunek 22.7). Stworzenie sejsmografu emocji pozwala rozszerzyć wspólny kontekst ponad dane obiektywne (to co się wydarzyło) i uwzględnić pewne dane subiektywne (co zespół czuł w związku z wydarzeniami).



**RYSUNEK 22.7.** Sejsmograf emocji

Aby utworzyć sejsmograf, każda uczestnicząca osoba rysuje krzywą przedstawiającą jej samopoczucie lub też poziom energii w trakcie trwania sprintu. Wygodnie jest narysować sejsmograf bezpośrednio pod linią zdarzeń, dzięki czemu oba zestawy danych są ze sobą skorelowane. Później uczestnicy mogą przeanalizować te dane celem znalezienia interesujących spostrzeżeń, które posłużą do poprawy procesu.

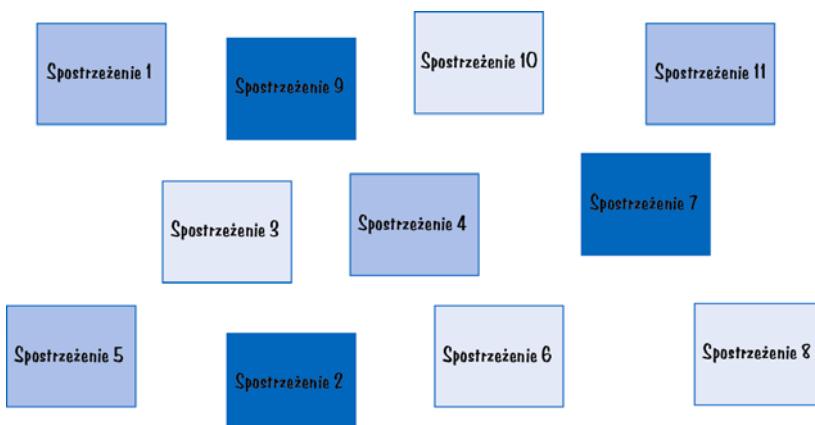
## Identyfikacja spostrzeżeń

Po ustaleniu wspólnego kontekstu uczestnicy mogą przemyśleć to, co widzą, zrozumieć, zinterpretować dane i dojść do spostrzeżeń pozwalających na poprawę procesu. Osiągnięcie takiego wyniku wymaga skupienia na poziomie systemowym (całościowym). Skupianie się wyłącznie na jednym aspekcie (patrzenie w sposób lokalny) może spowodować, że zespół przegapi wielki obraz sytuacji. Skupienie na poziomie systemowym pozwala również na pozostawienie z tyłu wszelkich powierzchownych przekonań i dotarcie do prawdziwych przyczyn problemów.

Uczestnicy powinni zacząć od przeanalizowania danych wspólnego kontekstu. Na przykład mogą spojrzeć na swoją linię zdarzeń oraz sejsmograf emocji i zadać następujące pytania, które pomogą odkryć spostrzeżenia:

- Co poszło dobrze?
- Co nie poszło dobrze?
- Gdzie widać jakieś możliwości zrealizowania rzeczy w inny sposób?

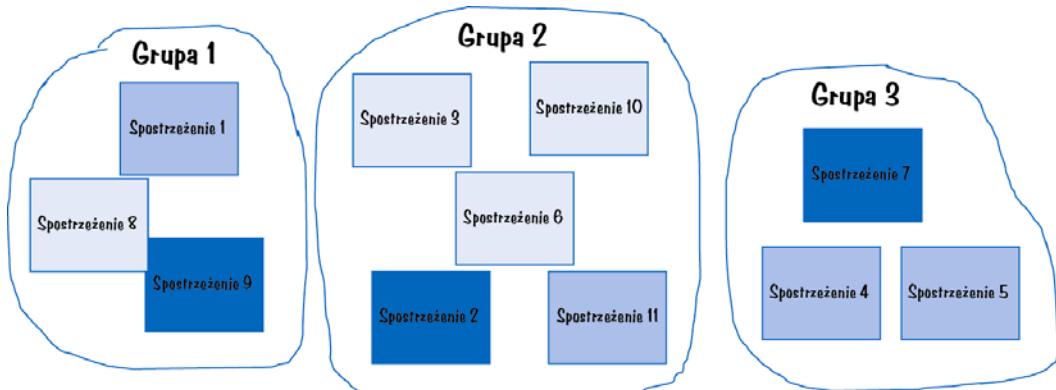
Często uczestnicy są proszeni o zapisanie swoich spostrzeżeń na karteczkach samoprzylepnych i umieszczenie ich na ścianie lub innej powierzchni, tak aby każdy mógł je widzieć (patrz rysunek 22.8).



**RYSUNEK 22.8.** Ściana z karteczkami spostrzeżeń retrospekcji

Innym źródłem spostrzeżeń może być utrzymywany przez zespół **rejestr spostrzeżeń**, czyli posortowana według priorytetów lista wcześniejszych spostrzeżeń, wobec których nie podjęto jeszcze żadnych działań. Jeżeli taki rejestr istnieje, należy go przejrzeć, aby stwierdzić, które spostrzeżenia uczestnicy chcieliby włączyć w bieżącą retrospekcję. Istniejące już spostrzeżenia mogą mieć swoje karteczki umieszczone na ścianie obok nowych.

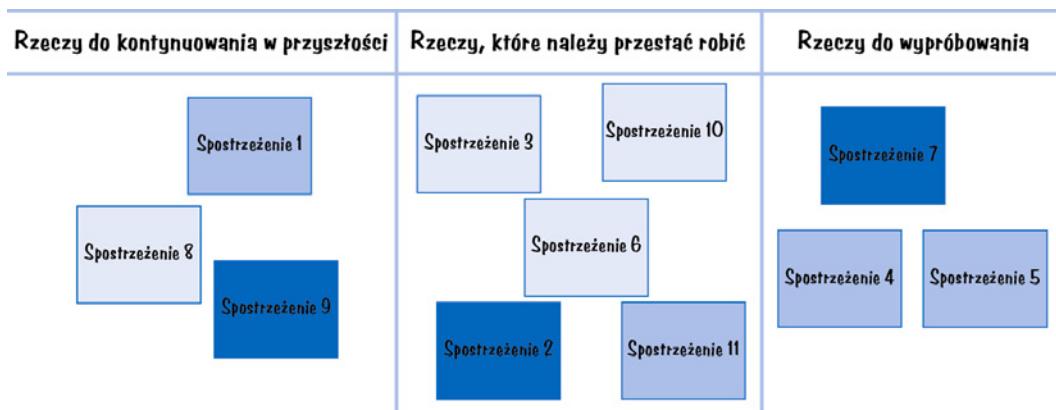
Po umieszczeniu karteczek na ścianie uczestnicy będą musieli je poukładać. Wiele zespołów robi to poprzez ćwiczenie nazywane **cichym grupowaniem**, polegające na połączeniu w zbiory identycznych lub bardzo podobnych do siebie spostrzeżeń (patrz rysunek 22.9).



**RYSUNEK 22.9.** Karteczki ze spostrzeżeniami połączone w grupy podobieństwa

Podczas cichego grupowania wszyscy razem katalogują spostrzeżenia bez prowadzenia ustnej dyskusji, opierając się wyłącznie na ruchach karteczek jako środkach komunikacji i koordynacji pomiędzy uczestnikami. Ciche grupowanie jest szybkie i wydajne.

Inne zespoły decydują się przed rozpoczęciem retrospekcji podzielić ścianę na trzy obszary (takie jak rzeczy, które należy kontynuować, rzeczy, których wykonywania należy zaprzestać, i rzeczy do wypróbowania). Następnie kiedy tworzone są karteczki ze spostrzeżeniami, uczestnicy mogą umieścić każdą z nich na ścianie pod jedną z kategorii (patrz rysunek 22.10).



**RYSUNEK 22.10.** Karteczki ze spostrzeżeniami umieszczone w ustalonych wcześniej grupach

Mimo wstępnego przypisania karteczek do kategorii nadal warto, aby uczestnicy przeprowadzili ciche grupowanie i zebraли razem karteczki o podobnym znaczeniu.

Po utworzeniu wspólnego kontekstu i przeanalizowaniu danych z karteczek uczestnicy powinni zidentyfikować kilka obszarów poprawy w swoim procesie scrumowym, czyli w swojej wspólnej

metodzie pracy mającej na celu dostarczanie wartości. Niektóre ze spostrzeżeń mogą prowadzić do głębszych dyskusji pomiędzy uczestnikami mających na celu lepsze zrozumienie przyczyn, ważnych wzorców lub związków. Po przedyskutowaniu wszystkich spostrzeżeń umieszczonych na ścianie nadchodzi pora określenia, co należy zrobić z wszystkimi tymi informacjami.

## Wskazanie działań

Spostrzeżenia to nasz sposób rozumienia lub też postrzegania sposobów poprawy stanu rzeczy. Chcąc wydobyć długotrwąłą wartość z tych spostrzeżeń, musimy przejść od dyskusji nad nimi do konkretnych działań. Na przykład: jeżeli spostrzeżenie to „Tracimy zbyt dużo czasu, ponieważ system zarządzania wersją ma ciągle usterki”, działaniem pozwalającym przezwyciężyć ten problem może być „Niech Tamara zaaplikuje łatki wypuszczane przez producenta systemu zarządzania wersją, aby zaczął on działać bardziej stabilnie”. Tamara jest członkiem zespołu deweloperskiego i może zrealizować to zadanie w trakcie sprintu.

Uczestnicy powinni również poświęcić czas na przejrzenie akcji mających na celu poprawienie procesów podjętych w poprzednim sprintie. Jeżeli akcje te nie zostały ukończone (lub nawet rozpoczęte), wszyscy muszą wiedzieć, dlaczego tak się stało, zanim przystąpią do zajmowania się nowymi spostrzeżeniami. Mogą oni zdecydować o realizacji poprzednich akcji lub przynajmniej nadać im wyższy priorytet w porównaniu z nowymi spostrzeżeniami, których właśnie dokonali.

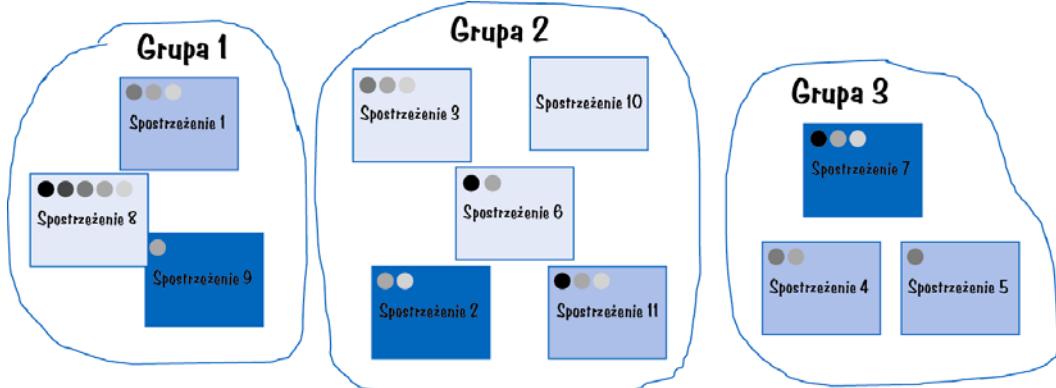
## Wskazanie spostrzeżeń

Trzeba mieć świadomość, że w trakcie retrospekcji bardzo często identyfikowanych jest o wiele więcej spostrzeżeń, niż zespół scrumowy jest w stanie „przetrawić” i zastosować w trakcie krótkiego zakresu czasu. W związku z tym uczestnicy powinni w pierwszej kolejności określić, które działania naprawcze należy podjąć bezzwłocznie, a które można odłożyć na później. W wielu zespołach uczestnicy proszeni są o nadanie priorytetów w oparciu o ich własne przekonania dotyczące wag spostrzeżeń lub też ich własny entuzjazm wobec wprowadzenia konkretnych poprawek. Często te dwie rzeczy nie idą w parze. Możemy się zgadzać co do tego, że pewne spostrzeżenie jest ważne, ale ponieważ nie ma chętnych do zrealizowania wiążącej się z nim pracy, być może nie jest ono najlepszym wyborem na teraz. Jeżeli uczestnicy wykazują entuzjazm wobec poprawki, będą o wiele bardziej chętni do podjęcia konkretnych kroków.

Jednym ze sposobów nadawania priorytetów spostrzeżeniom jest użycie **głosowania kropkowego** — patrz rysunek 22.11.

Podczas głosowania kropkowego każdy uczestnik otrzymuje małą liczbę (być może od trzech do pięciu) kolorowych kropek. Następnie wszyscy razem umieszczają swoje kropki na karteczkach ze spostrzeżeniami, które ich zdaniem trzeba jak najszybciej przekształcić w konkretne działania. Dana osoba może umieścić wszystkie swoje kropki na jednej karteczce lub rozproszyć je na różnych spostrzeżeniach. Po przeprowadzeniu głosowania karteczki z najwyższą liczbą kropek powinny być rozważane w pierwszej kolejności.

Ile dokładnie spostrzeżeń należy wybrać do przekształcenia w konkretne działania? To zależy od tego, jak dużo ze swojej pojemności uczestnicy są gotowi poświęcić spostrzeżeniom i w jakim czasie.



**RYSUNEK 22.11.** Przykład głosowania kropkowego

Zazwyczaj zakresem czasu jest następny sprint. Zatem jeśli zespół realizuje dwutygodniowe sprints, najprawdopodobniej będzie rozwijał spostrzeżenia, które można wprowadzić w życie w ciągu kolejnych dwóch tygodni. Nawet jeśli spostrzeżenie dotyczy zbyt dużej rzeczy, aby można było załączyć ją w ciągu następnego sprintu, uczestnicy mogą zdecydować się na rozpoczęcie prac, tak aby zdemonstrować konkretny postęp na drodze do jej zrealizowania.

Uczestnicy muszą również wskazać, jaką część swojej pojemności mogą poświęcić na wprowadzanie poprawek w następnym sprintie (lub w innym rozważanym przez nich zakresie czasu). Jeżeli zespół planuje poświęcić czas w następnym sprintie na realizację zadań z poprzedniej retrospekcji, będzie miało to wpływ na zdolność do realizacji zadań wynikających z bieżącej retrospekcji.

Spędzanie czasu na pracy wprowadzającej w życie poprawki wynikające z bieżącej retrospekcji oraz poprzednich ogranicza czas zespołu przeznaczony na realizację cech. Zatem jaką ilość czasu zespół powinien zarezerwować dzisiaj na wprowadzanie poprawek, aby otrzymać spodziewany zysk w przyszłości? Odpowiedzenie na to pytanie wymaga pewnych wskazówek ze strony właściciela produktu, co jest jednym z powodów, dla których jego uczestnictwo w spotkaniu retrospekcji jest tak ważne. Jeśli zespół scrumowy nie wyspecyfikuje jawnie czasu poświęconego na wprowadzanie zmian do procesu, najprawdopodobniej wszystkie spostrzeżenia pozostaną nieruszzone.

Kiedy wiemy już, ile czasu poświęcimy na poprawki procesu, uczestnicy są w stanie w przybliżeniu powiedzieć, które ze spostrzeżeń o wysokim priorytetie mogą zostać bezzwłocznie wzięte do realizacji. Niemniej jednak podjęcie ostatecznej decyzji wymaga wcześniejszego ustalenia konkretnych zadań.

### Wybór działań

Nasze spostrzeżenia zostały posortowane według priorytetów i wiemy już mniej więcej, ile czasu chcemy poświęcić pracy nad nimi. Retrospekcja nie przyniesie jednak żadnej konkretnej wartości, o ile nie zdefiniujemy konkretnych, możliwych do wykonania działań w oparciu o nasze spostrzeżenia i nie poprawimy procesu scrumowego.

Większość działań będzie miała formę zadań, jakie jeden lub więcej członków zespołu scrumowego wykona w ciągu nadchodzącego sprintu. Na przykład: jeżeli spostrzeżenie to „Zbyt dużo czasu zajmuje przekonanie się, że proces budowania kodu zakończył się błędem”, działaniem może być „Niech serwer budujący wysyła e-mail za każdym razem, kiedy proces budowania zarejestruje błąd”.

To działanie wymaga pracy w postaci zadań do wykonania przez jednego lub kilku członków zespołu deweloperskiego. Zespół powinien stwierdzić, kto jest w stanie wykonać tę pracę i jak dużo czasu będzie ona wymagać. Dopiero wtedy zespół może być pewny, że praca nad konkretnym spostrzeżeniem da się zrealizować w ramach dostępnej pojemności.

Nie wszystkie spostrzeżenia wymagają konkretnej pracy. Na przykład spostrzeżenie takie jak „Szanuj pozostałych członków zespołu i przychodź codziennie w porę na spotkanie scrumowe” nie powinno wymagać jakiekolwiek pracy ze strony zespołu. Mimo że należy podjąć pewne działanie: „Ludzie powinni zadać sobie trud pojawiania się na czas”, nie zmniejsza ono pojemności zespołu.

Czasami działania mające usuwać przeszkody, za których likwidowanie odpowiada formalnie mistrz młyna, musi wykonać ktoś inny w organizacji. Na przykład dla spostrzeżenia „Nie możemy ukończyć elementu rejestru produktu, ponieważ potrzebujemy najnowszej wersji oprogramowania od innego dostawcy, aby wykonać na nim testy” działaniem może być „Nina będzie pracować z zespołem zamówień, aby otrzymać najnowszą wersję potrzebnego oprogramowania”. Zatem mistrz młyna podejmie współpracę z ludźmi z działu zamówień, aby rozwiązać problem z oprogramowaniem od innego dostawcy, który uniemożliwia zespołowi ukończenie elementu rejestru produktu. To działanie rozpocznie się w następnym sprincie i zajmie część pojemności mistrza młyna. Rozwiążanie takiego problemu może potrwać nawet kilka sprintów.

Określając właściwe działania, musimy pamiętać, że czasami natychmiastowe rozwiązywanie spostrzeżonego problemu może być niemożliwe. Będziemy musieli przeprowadzić dodatkową eksplorację spostrzeżenia przed dokonaniem faktycznej poprawki. W takich przypadkach właściwym podejściem jest przeprowadzenie analizy i zebranie większej ilości danych w kolejnym sprincie, dzięki czemu będziemy mogli lepiej zrozumieć problem.

Na przykład spostrzeżeniem może być „Nie potrafimy zrozumieć, dlaczego dwa w pełni przetestowane komponenty posiadające swoje zestawy testów automatycznych nie chcą działać po stworzeniu jednego zestawu testów automatycznych dla obu komponentów, mimo że każdy z nich jest uruchamiany osobno”. W takiej sytuacji nie istnieje konkretne działanie, które członkowie zespołu mogą podjąć, aby rozwiązaćauważony problem, ponieważ tak naprawdę nie rozumieją, co jest nie tak. Zespół może jednak stworzyć zadania dla poszczególnych członków zespołu mające na celu eksplorację tego problemu w następnym sprincie i określić, jaką część swojej pojemności przeznaczyć na ten cel.

## Rejestr spostrzeżeń

Jak wspomniałem wcześniej, wiele zespołów tworzy rejestr spostrzeżeń (czasem nazywany rejestrem usprawnienia), w którym przechowywane są wszelkie problemy zidentyfikowane podczas retrospekcji, ale nierozwiążane natychmiast. W trakcie następnej retrospekcji sprintu uczestnicy spotkania mogą użyć spostrzeżeń z rejestru jako kandydatów do nadania priorytetów względem nowych spostrzeżeń. Ma to pomóc w wybraniu obszaru, na którym skupiona będzie uwaga w następnym sprincie. Oczywiście rejestr spostrzeżeń powinien być okresowo pielęgnowany, aby w środku znajdowały się wyłącznie wartościowe spostrzeżenia.

Inne zespoły zwyczajnie ignorują wszelkie spostrzeżenia, których nie wprowadziły w życie podczas kolejnego sprintu. Wynika to z przekonania, że jeżeli spostrzeżenie było faktycznie istotne, zostanie ono odkryte ponownie w trakcie następnej retrospekcji.

## Zakończenie retrospekcji

Po wskazaniu działań naprawczych uczestnicy kończą retrospekcję. Wiele zespołów kończy, podsumowując działania, jakie zespół zdecydował się podjąć w oparciu o nabytą wiedzę. Może się to sprowadzać do streszczenia każdego przyjętego do realizacji zadania oraz wskazania, kto będzie nad nim pracował.

Zakończenie jest również dobrym czasem do pochwalmienia ludzi biorących udział w spotkaniu. Każdy uczestnik powinien powiedzieć kilka miłych słów na temat wkładu innych osób. Nie należy przy tym zapominać o osobach spoza zespołu scrumowego, które pomimo napiętego kalendarza znalazły czas na uczestnictwo w retrospekcji.

I w końcu dobrym pomysłem jest zapytanie zespołu o kilka sugestii dotyczących ulepszenia po-dejścia zespołu do wykonywania retrospekcji. W końcu retrospekcja jest częścią środowiska scrumowego i jako taka powinna być przedmiotem inspekcji i adaptacji.

## Wprowadzenie zmian w życie

Aby to, co wydarzyło się w trakcie retrospekcji sprintu, *nie* pozostało jedynie na papierze, uczestnicy powinni wprowadzić w życie działania, które zobowiązali się zrealizować w trakcie spotkania. Niektóre działania (takie jak zobowiązanie wszystkich do przychodzenia o określonej porze na spotkania scrumowe) muszą być jedynie zastosowane w praktyce przez wszystkich członków zespołu pod nadzorem mistrza młyna. Inne trzeba będzie zaplanować podczas nadchodzącej aktywności planowania sprintu.

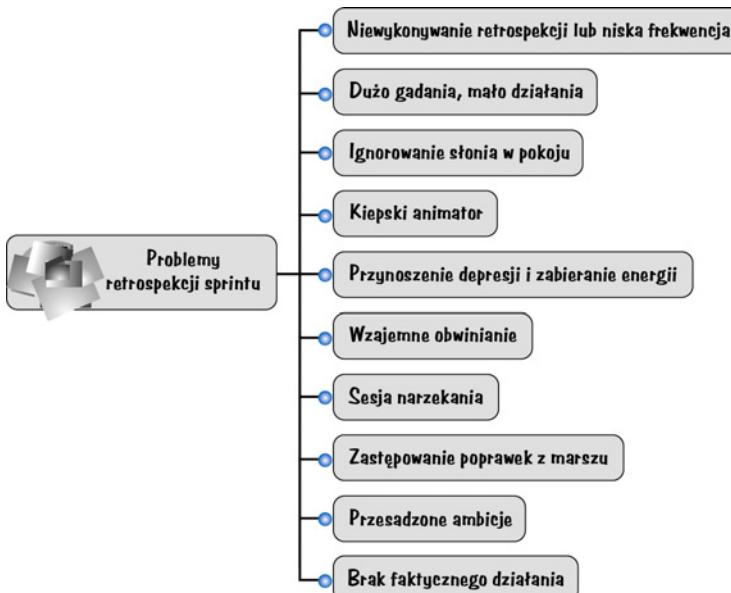
Można to zrobić przez wypełnienie rejestru sprintu zadaniami odpowiadającymi poszczególnym działaniom naprawczym przed planowaniem nowych cech. Pojemność zespołu do pracy nad nowymi cechami zostanie wtedy zmniejszona o szacowany czas potrzebny na wykonanie zadań poprawiających proces. Każde podejście pozwalające zespołowi podjąć dobre zobowiązanie w czasie planowania sprintu z jednoczesną szansą pracy nad zadaniami naprawczymi jest dobre.

Wyjątkiem, który *nie* gwarantuje powodzenia, jest przyjęcie „planu poprawy” reprezentującego pracę odrębną względem tego, co zespół robi w każdym sprincie. Takie dwutorowe podejście będzie niemal zawsze doprowadzać do sytuacji, w której plan naprawczy będzie miał drugorzędne znaczenie wobec planowania cech w sprintach. Chcąc zapewnić wykonanie działań naprawczych, nie oddzielaj ich od głównej pracy, ale łącz z nią!

Działania, które nie wymagają poświęcania czasu przez członka zespołu, najprawdopodobniej trafią na listę przeszkoł mistrza młyna. Działania przeznaczone dla innych zespołów lub całej organizacji mogą zostać wpisane do rejestru właściwego dla ludzi, którzy będą musieli wykonać odpowiednią pracę. Mistrz młyna współpracuje wtedy zazwyczaj z osobami z zewnątrz, aby upewnić się, że odpowiednie działania są faktycznie realizowane.

## Problemy retrospekcji sprintu

Nie zawsze retrospekcje przebiegają idealnie. Pracując z wieloma organizacjami używającymi Scruma, zauważałem cały szereg powtarzających się problemów (patrz rysunek 22.12).



**RYSUNEK 22.12.** Problemy retrospekcji sprintu

Niefortunne podejście polega na rezygnacji z wykonywania retrospekcji lub przeprowadzaniu jej przy bardzo niskiej frekwencji. Powody obu tych sytuacji są bardzo podobne. Bycie przypisany do wielu zespołów jednocześnie może utrudnić uczestnictwo w retrospekcjach ze względu na nachodzące na siebie terminy spotkań. Jest to przeszkoda organizacyjna, którą powinni rozwiązywać menedżerowie. Inna możliwość to zwykłe znudzenie lub brak zaangażowania czy też przekonania do stosowania Scruma wśród członków zespołu. Niektórzy pracownicy chodzą z przekonaniem, że wykonywanie czegoś więcej niż ich regularne obowiązki nie jest warte poświęcania czasu (na przykład inżynierowie oprogramowania mogą uważać, że wszystko, co wychodzi poza zakres programowania i testowania, jest zwykłym marnotrawstwem). Taka postawa wyrasta często z naiwnego podejścia do Scruma i jego skupienia na nieustannym poprawianiu procesu. Czasem jest jednak zupełnie odwrotnie — członkowie zespołu dochodzą do przekonania, że osiągnęli szczyt możliwości w stosowaniu Scruma i w związku z tym niczego więcej nie nauczą się z właśnie wykonanego sprintu, od swoich kolegów i koleżanek lub też na podstawie swoich własnych sukcesów lub porażek. Jeżeli ludzie nie dostrzegają wartości w wykonywaniu retrospekcji sprintu lub też w braniu w niej udziału, rozważ poświęcenie następnego spotkania tego typu na eksplorację właśnie tego problemu.

Czasami niska frekwencja wynika z niewygody uczestników łączących się zdalnie za pośrednictwem telefonu lub wideokonferencji. Jeżeli problemem jest czas spotkania, rozważ inną porę retrospekcji lub też zmieniaj cyklicznie tę porę, tak aby każda z odległych lokalizacji miała co jakiś czas szansę na wzięcie udziału w dogodnym dla siebie czasie. Jeśli problemem jest tylko niewygoda uczestniczenia w spotkaniu zdalnie, rozważ możliwe poprawki w systemie telekomunikacyjnym, a także sposób przeprowadzania ćwiczeń, który bierze pod uwagę uczestników pracujących zdalnie.

Niektóre retrospekcje są bogate w dyskusje, ale nie wynika z nich nic, co można by wprowadzić w życie. Nazywam je retrospekcjami typu *dużo gadania, mało działania*. Jeśli jest tylko dużo gadań, marnujemy swój czas. Rozważ wprowadzenie na spotkanie animatora doświadczonego w prowadzeniu retrospekcji, który pomoże aktywować uczestników do prawdziwego działania.

Inne retrospekcje można obserwować z zafascynowaniem. Widać w ich trakcie pewien krytyczny problem, który ma istotny wpływ na zespół, ale nikt nie chce wyciągnąć go na światło dzienne. Mówiąc inaczej, uczestnicy ignorują obecność słonia w pokoju. Niechęć do „dyskutowania słonia” wynika najprawdopodobniej z pewnych uprzedzeń dotyczących poczucia bezpieczeństwa. Mistrz młyna powinien przejąć inicjatywę i w pierwszej kolejności pomóc zespołowi oraz organizacji rozwiązać problem z brakiem poczucia bezpieczeństwa.

Niektóre retrospekcje są prowadzone w kiepski sposób. Osoba przewodząca spotkaniu, być może niedoświadczony jeszcze mistrz młyna, stara się jak może, ale najwyraźniej nie przynosi to efektów. Być może należy sięgnąć po kogoś z zewnątrz do przeprowadzenia kilku kolejnych retrospekcji.

Są retrospekcje, które powodują depresję i utratę energii. Być może sprinty nie przebiegają dobrze i członkowie zespołu postrzegają retrospekcję jako aktywność, która tylko potęguje kiepski nastój przez zmuszanie wszystkich do ponownego przeżywania złych momentów. Rozważ poświęcenie odrobiny czasu na wprowadzenie odpowiedniej atmosfery na początku spotkania. Ponadto osoba spoza zespołu prowadząca retrospekcję może lepiej pomóc ludziom skupić się na poprawkach niosących pozytywne efekty.

Często retrospekcje są przygnębiające, ponieważ ludzie zaczynają obarczać siebie nawzajem winą i wytykać się palcami. Animator spotkania musi wyeliminować tego typu zachowania, kiedy tylko zauważa pierwsze symptomy, aby nie dopuścić do ich eskalacji.

Retrospekcje mogą przerodzić się w sesje narzekania. Być może niektórzy postrzegają retrospekcję jako sesję terapeutyczną lub spotkanie, na którym mogą sobie ponarzekać na aktualny stan rzeczy (lub oczekiwany przez nich kierunek zmian). Ich celem nie jest poprawienie sytuacji, a jedynie narzekanie. Rozważ zapraszanie na retrospekcję jedynie osób, które mogą wprowadzić prawdziwe zmiany. Postaraj się unikać krytykowania osób nieobecnych na spotkaniu, które ciągle narzekają — postaraj się raczej przeprowadzić z nimi szczerą rozmowę w cztery oczy.

Inną niefortunną sytuacją jest traktowanie przez uczestników retrospekcji jako *faktycznego* czasu wprowadzania poprawek i przez to zmniejszanie wagi poprawiania procesu w trakcie trwania sprintów. Retrospekcja jest doskonałym czasem dla zespołu do porozmawiania o pewnym okresie pracy i zastanowienia się, jak można polepszyć pewne rzeczy. Mistrz młyna powinien promować poprawianie procesu w trakcie trwania sprintu.

Czasami nasze aspiracje wykraczają poza nasze możliwości. Nowe zespoły pozytywnie nastawione i skupione na faktycznym polepszeniu procesu mogą często przesadzić pod względem swoich ambicji i ustawić cele w sposób zupełnie nierealistyczny. Takie postępowanie doprowadzi jedynie do wielkiego zawodu, kiedy zespołowi nie uda się osiągnąć swoich wielkich, ambitnych celów. Mistrz młyna powinien zachować czujność i przypomnieć zespołowi o jego pojemności przeznaczonej na poprawki procesu, a następnie pomóc w urealnieniu ambicji.

Prawdopodobnie najważniejszym problemem ze wszystkich wymienionych jest brak przejścia do wykonania pracy nad poprawkami procesu zidentyfikowanymi w czasie retrospekcji. Jeśli nie zamierzamy działać, nie ma powodu, aby marnować czas na retrospekcje. Za nieustanne poprawianie procesu odpowiada mistrz młyna. Jeśli nie widać postępów w tym zakresie, mistrz młyna musi wspólnie z zespołem ustalić przyczynę takiej sytuacji, a następnie pomóc członkom zespołu w jej przezwyciężeniu.

## Zakończenie

Retrospekcje sprintów są czasem, kiedy zespoły analizują, jak dobrze radzą sobie z użyciem Scruma, i proponują poprawki. Retrospekcja jest aktywnością wymagającą zaangażowania wszystkich członków zespołu scrumowego (a także osób spoza zespołu, jeśli zachodzi taka potrzeba).

Po wykonaniu wstępnych prac retrospekcja zaczyna się od wprowadzenia odpowiedniej atmosfery zapewniającej pozytywny wynik spotkania. Następnie staramy się, aby wszyscy uczestnicy rozumieli sytuację w taki sam sposób. Dalej można rozpocząć odkrywanie spostrzeżeń i wskazywanie akcji prowadzących do poprawienia procesu. Po wszystkim kończymy retrospekcję, ale pamiętamy, że najważniejszą rzeczą jest wprowadzenie w życie działań naprawczych, które zwiększą wydajność zespołu w kolejnym sprincie. Należy również zwracać uwagę na problemy, które mogą uniemożliwić otrzymanie pozytywnego wyniku z retrospekcji, i sprawnie je likwidować.



## Rozdział 23

# CO DALEJ?

---

W 22 rozdziałach tej książki opisałem środowisko Scrum i wyjaśnilem, co moim zdaniem jest sednem Scruma. Po lekturze tych rozdziałów powinieneś rozumieć, jak należy używać Scruma, aby dostarczać innowacyjne rozwiązania, a także czuć, czemu Scrum zaleca stosowanie określonych ról, praktyk, artefaktów i zasad. Jesteś teraz gotowy do zdefiniowania własnej ścieżki naprzód. W tym rozdziale rozwinę myśl, zgodnie z którą nie istnieje uniwersalny, ostateczny cel wdrożenia Scruma. To Ty tworzyesz swoją własną drogę w kierunku zwinności. Na koniec wskażę znaczenie najlepszych praktyk, a także sposób używania Scruma, z jego iteracyjnym i inkrementacyjnym podejściem, będący podstawą do odkrywania drogi naprzód.

### Nie ma stanu końcowego

Każda organizacja ma swoją wizję tego, co chce osiągnąć. Używanie Scruma może pomóc organizacjom w zarządzaniu pracą mającą na celu osiągnięcie tej wizji. Bycie wysoce wydajnym poprzez użycie Scruma i przez to bardziej zwinnym nie jest celem końcowym, a jedynie środkiem bardziej wydajnego i ekonomicznego osiągania celów biznesowych. Skąd zatem będziesz wiedział, że Twoje wykorzystanie Scruma jest pełne?

Nie istnieje definicja ukończenia w procesie przechodzenia lub też wdrażania Scruma. Nie istnieje model dojrzałości pod względem zwinności, taki jak CMMI [SEI, 2010], w którym celem jest osiągnięcie poziomu 5. Próba zdefiniowania stanu „gotowości” dla swojej implementacji Scruma zakłada, że kiedy już osiągniesz ten stan, będzie to maksimum Twoich możliwości. Takie założenie przeczy jednej z zasad Scruma — zasadzie nieustannego udoskonalania czy pracowania cały czas nad lepszym przystosowaniem własnego użycia Scruma do złożonego świata, w którym przyszło Ci tworzyć produkty.

Co gorsza, gdybyśmy spróbowali zdefiniować taki stan końcowy dla przemysłu, dotyczyłyby on wszystkich organizacji, nawet tych, które produkują zupełnie odmienne typy produktów w zupełnie różnych warunkach.

Powiedzenie „W końcu osiągnąłem zwinność!” jest pozbawione sensu. Podsumował to bardzo ładnie Mike Cohn: „Zwinność nie jest czymś, co osiągasz, ale czymś, co osiągasz w coraz większym stopniu” [Cohn, 2010]. Nie istnieje stan końcowy, który można by nazwać zwinnością lub Scrumem. Osiąganie coraz większej biegłości w Scrumie i bycie coraz bardziej zwinnym jest nieustającym procesem, niekończącym się udoskonalaniem, mającym na celu poprawienie wyniku końcowego.

## Odkryj swoją własną ścieżkę

Tak jak nikt nie może powiedzieć Ci, w którym miejscu powinna zakończyć się Twoja implementacja Scruma, tak nikt nie może poprowadzić Cię z góry wyznaczoną ścieżką gwarantującą sukces. Musisz sam zdobyć wiedzę, dokonać inspekcji i adaptacji swojego sposobu działania w oparciu o cele wyznaczone przez Twoją organizację, kulturę i nieustannie zmieniające się środowisko, w którym musisz operować. Szybką metodą zdobycia zwinności może wydawać się próba przejścia ścieżką, którą przeszedł ktoś inny, z wykorzystaniem zdobytej przez niego wiedzy. Niestety żadne dwie organizacje lub nawet dwa zespoły wewnętrz organizacji nie są do siebie podobne. Poruszanie się po czystych śladach może równie dobrze doprowadzić Cię do tego samego złego położenia.

Nie da się uniknąć własnego procesu nauki. Można jedynie starać się szybko domykać własne pętle zdobywania wiedzy, a następnie dokonywać inspekcji zdobytych informacji i adaptować się. Nie sugeruję absolutnie, abyś ignorował sugestie tych, którzy przed Tobą wdrażali zwinność. Przeanalizuj to, jak oni działały i jak dobre skutki to odniosły, a następnie odkryj swoją własną ścieżkę do bycia bardziej zwinnym.

## Stosowanie najlepszych praktyk

Jeżeli nie powinniśmy podążać ścieżkami wyznaczonymi przez innych, jak ma się to do stosowania najlepszych praktyk? Tak jak nie istnieje jedna ścieżka, którą należy podążać, tak też nie istnieje jeden zestaw najlepszych praktyk obowiązujący wszystkie organizacje.

Kiedy jestem proszony o opisanie „najlepszych praktyk” stosowanych przez organizacje w celu stania się bardziej zwinnymi, podaję przykłady. Zawsze jednak umieszczam moją odpowiedź w odpowiednim kontekście innych organizacji, dzięki czemu osoby zadające pytanie mogą ocenić, czy ich zdaniem przyjęcie podobnego podejścia ma sens we własnej organizacji. Nawet skalując w górę wewnętrz organizacji, musimy zachować ostrożność pod względem uniwersalnego zastosowania najlepszych praktyk. Wiele organizacji stara się opisać, co robi odnoszący sukcesy zespół scrumowy, uchwycić tę wiedzę, a następnie przekształcić ją we wzorzec najlepszych praktyk. Takie działanie może być szkodliwe, ponieważ podejście sprawdzające się w jednym zespole niekoniecznie musi zadziałać w innych.

Być może zauważysz, że odnosząc się do najlepszych *praktyk*, użyłem kilkakrotnie słowa *podejście*. Porozmawiamy przez chwilę o tej różnicę. Na przestrzeni całej książki używałem pojęcia *praktyka* w znaczeniu bazowego lub kluczowego aspektu Scruma. Podejście jest konkretną implementacją praktyki scrumowej. Kiedy ludzie pytają mnie o najlepsze *praktyki*, zakładam, że mają na myśli najlepsze *podejścia*.

Chociaż dwa różne zespoły lub organizacje stosują odmienne implementacje Scruma, każdy z nich powinien trzymać się tych samym praktyk Scruma. Na przykład powinny posiadać zespoły scrumowe składające się z właściciela produktu, mistrza młyna i zespołu deweloperskiego. Oba powinny przeprowadzać planowanie sprintu, codzienne działania scrumowe, przeglądy sprintu i retrospekcje sprintu. Niemniej jednak oczekuję, że każdy zespół (lub organizacja) będzie posiadał swoje własne podejście do stosowania tych praktyk. Pokażę to na przykładzie.

Kluczową praktyką Scruma są codzienne działania scrumowe. Jeśli ich nie przeprowadzasz, nie realizujesz Scruma. Podczas codziennych działań scrumowych każdy członek zespołu informuje pozostałe osoby o swoich postępach, dzięki czemu wszyscy mogą całościowo ocenić zakres prac. Kiedy w ciągu dnia ma się odbyć odpowiednie spotkanie, która osoba powinna podać swój status jako pierwsza? Scrum nie definiuje takich rzeczy. Każdy zespół musi przyjąć swoje własne podejście.

Ciekawe podejście dotyczące tego, kto powinien mówić jako pierwszy, poznałem, pracując z zespołem z Vancouver (Kanada). W tym zespole na początku codziennego spotkania scrumowego mistrz młyna rzucał do góry zabawkę — pluszowego łosia. Kto złapał łosia, mówił pierwszy, a po nim kolejno pozostali członkowie zespołu, idąc w lewo od osoby trzymającej łosia. Takie proste, wreszcie głupkowate, ale śmieszne podejście sprawdzało się bardzo dobrze w tym zespole.

Okazało się, że zespół z Vancouver posiadał bliźniaczy zespół w Chinach uformowany kilka miesięcy później. Członek zespołu z Chin zapytał zespół z Vancouver o „politykę lub najlepszą praktykę” wskazywania osoby, która powinna mówić pierwsza podczas codziennych działań scrumowych. Osoba z Vancouver powiedziała osobie z Chin, że w Vancouver wskazują za każdym razem tę osobę, „rzucając łosem”. Rzucanie łosiem musiało przyjąć zupełnie inne znaczenie po przetłumaczeniu na chiński! Podejście, które sprawdzało się dobrze w zespole z Vancouver, nie zadziałałoby zupełnie w przypadku zespołu z Chin. Zespół z Chin przyjął swoje własne podejście.

Scrum definiuje podstawowe praktyki, które muszą być wypełniane. Każdemu zespołowi pozwala się jednak wolną rękę w określeniu podejścia (lub najlepszej praktyki), które dla niego zadziała. Stąd też podejścia są unikatowe dla każdego zespołu i mogą oraz powinny być używane przez inne zespoły jedynie, jeśli ma to sens w ich konkretnym kontekście.

## Używanie Scruma do odkrywania ścieżki naprzód

Niezależnie od tego, czy jesteś nowym użytkownikiem Scruma, czy też używasz już tej metody do tworzenia produktów, możesz wykorzystać zasady Scruma, aby pomóc sobie w wyznaczaniu ścieżki naprzód. Bardzo szczegółowo opisuje to podejście Mike Cohn w swojej książce *Scrumming with Agile* [Cohn, 2009]. Po więcej szczegółowych informacji na ten temat odsyłam Cię do jego wspomnianej książki.

Opiszę sedno tego podejścia w oparciu o przykład. W roku 2007 zostałem zatrudniony przez dużą międzynarodową organizację, aby pomóc jej wdrożyć i używać Scruma. Organizacja ta posiadała stu pracowników IT w Nowym Jorku i czterystu pracowników IT w Bombaju (Indie). W dowolnie wybranym momencie firma ta miała otwartych około 45 różnych projektów.

Organizacja zdecydowała, że każdy nowy zespół, który miał zacząć stosować Scrum, powinien posiadać trenera do pomocy. Przy ograniczonych zasobach trenerów na początku wprowadzanie Scruma w całej organizacji jednocześnie było nierozsądne. W związku z tym, tak jak to ma miejsce w tego typu środowiskach, organizacja wybrała małą liczbę projektów do przeprowadzenia

programu pilotażowego. Celem było stopniowe przenoszenie zespołów do pracy w Scrumie w miarę wzrostu możliwości trenerskich organizacji poprzez zdobywanie praktyki przez samych pracowników firmy.

Projekty pilotażowe miały szerokie spektrum, od utrzymania istniejących systemów po tworzenie nowych, dużych produktów. W oparciu o tę różnorodność każdy zespół scrumowy wdrażał swoją własną implementację środowiska Scrum, używając podejść odpowiadających osobom pracującym w zespole i pracy, jaką miały do wykonania. Utworzono stronę wiki opisującą podejścia stosowane przez różne zespoły — jej zadaniem była pomoc w zdobywaniu wiedzy przez całą organizację i dzielenie się rozwiązaniami, które sprawdzały się w zespołach.

Kilka miesięcy po rozpoczęciu adaptacji nadszedł czas na rozszerzenie skali użycia Scruma z poziomu zespołów na poziom całej organizacji. Utworzyliśmy coś, co Cohn nazywa skrótem ETC (Enterprise Transition Community<sup>1</sup> — [Cohn, 2009]). W naszym przypadku ETC miało nazwę Working Software Group. Ta grupa menedżerów i dyrektorów utrzymywała rejestr elementów opisujących usprawnienia i wykonywała trzytygodniowe sprints. Elementy w rejestrze opisywały inicjatywy zmian na poziomie organizacyjnym (takie jak „Zmienić model premiowania, tak aby był bardziej skupiony na zespołach”) lub istotne przeszkody, które utrudniały pracę jednemu lub więcej zespołom scrumowym („Poprawić stabilność serwerów, tak aby zespoły mogły wykonywać swoje testy”).

Pracując w sprintach nad elementami z tego rejestru, organizacja była w stanie iteracyjnie i inkrementacyjnie czynić postępy na swojej ścieżce do pomyślnej adaptacji Scruma. Nie istniał założony z góry stan końcowy wdrożenia Scruma. Próba jego zdefiniowania z góry byłaby marnotrawstwem podobnym do próby stworzenia pełnej specyfikacji wymagań dla zupełnie nowego, nigdy niezbudowanego produktu, którego nikt w pełni nie rozumiał.

Zamiast tego Working Software Group odbierała informacje od zespołów scrumowych i interesariuszy, a następnie dokonywała przyrostowych poprawek w strukturze organizacyjnej, tak aby lepiej dopasować się do wartości zwinności. Poprzez nieustające zdobywanie wiedzy, inspekcję i adaptację organizacja wyznaczyła właściwą ścieżkę naprzód, która współgrała z jej celami biznesowymi.

Tego typu wzorzec ETC jest obecnie bardzo popularny. Wiele organizacji zdaje sobie sprawę, że użycie Scruma do adaptacji Scruma jest rozsądnym podejściem do iteracyjnego i przyrostowego przekształcenia się w instytucję bardziej zwinną.

## Naprzód!

Bawi mnie sytuacja, kiedy przychodzą do mnie ludzie uważający, że nie można z góry ustalić prawnie wszystkich wymagań i w związku z tym chcą użyć Scruma do produkcji, ale zaraz potem wyjaśniają mi, że nie są gotowi do rozpoczęcia pracy w Scrumie, ponieważ nie rozpracowali jeszcze wszystkich szczegółów tego podejścia!

Stosując Scrum, nie powinieneś próbować zrobić wszystkiego w sposób perfekcyjny za pierwszym razem. To jest niemożliwe! Próba bycia perfekcyjnym zmusi Cię do zgadywania kosztem zdobywania wiedzy, którą zdobędziesz, jeśli zastosujesz Scrum, i wtedy też zobaczysz, co będzie dalej. Z mojego doświadczenia mogę powiedzieć, że kilka pierwszych sprintów nie wygląda różowo.

<sup>1</sup> Dosłownie komitet lub też społeczność transformacji przedsięwzięcia — przyp. tłum.

Trzeba się z tym pogodzić. Moje jedyne oczekiwanie wobec dowolnego zespołu scrumowego jest takie, aby w kolejnym sprintie był lepszy niż w poprzednim. Dlatego też nie opóźniaj startu. Niezależnie od tego, co Twoim zdaniem wiesz na temat swojego użycia Scruma, wyobraź sobie, ile więcej będziesz wiedział, kiedy rozpocznesz i zakończysz kolejny sprint!

Nie oczekuj również, że adaptacja Scruma przebiegnie bezproblemowo. Mogę zagwarantować, że wcześniej czy później Twój organizacji napotka przeszkody, które utrudnią wykonywanie Scruma. Scrum uwidacznia dysfunkcje i marnotrawstwo, które uniemożliwia organizacjom osiągnięcie ich prawdziwego potencjału. To, czego Scrum nie jest w stanie zrobić, to odpowiedzenie na pytania, jak poradzić sobie z tymi problemami. Tę ciężką pracę będą musieli wykonać ludzie w organizacji.

Status quo jest bardzo potężną siłą. O wiele łatwiej jest ludziom ignorować Scrum lub zmienić go, niż zmodyfikować od dawna pokutujące w firmie procesy, zasady czy zachowania. Kultura, która okazuje wrogie nastawienie wobec prób wskazania jej dysfunkcji, bardzo szybko przygasi jasne światło ukazujące to, co odbywa się w półciemiu. Aby przeciwdziałać tej tendencji, musisz być niezłomny i cierpliwy podczas wprowadzania zmian w organizacji. Musisz zrozumieć, że opór przed zmianami jest rzeczą naturalną. Pomagaj przetrwać najgorsze poprzez szerzenie wiedzy na temat zasad stojących u podstaw Scruma, a także celów, jakie chcesz osiągnąć. Pracuj z ludźmi zamiast przeciw ludziom, aby pokonać przeszkody, które uniemożliwiają Twemu zespołowi i organizacji dostrzeżenie pełnych zalet implementacji Scruma.

Mam nadzieję, że ta książka dostarczyła Ci kompendium wiedzy na temat Scruma, dzięki której oświetlisz swoją ścieżkę naprzód. Życzę Ci osiągnięcia jak największych sukcesów w podróży ze Scrumem.



# SŁOWNICZEK

---

## Wprowadzenie

Hasła w słowniku są uporządkowane alfabetycznie. Hasło może być pojedynczym słowem, takim jak *Scrum*, frazą, na przykład *kryteria akceptacji*, lub skrótem, na przykład *TDD*. Jeżeli hasło posiada kilka definicji, są one ponumerowane.

Wzajemne relacje pomiędzy hasłami przedstawiane są przy użyciu następujących odnośników:

- *Patrz odsyła* do hasła, które najlepiej definiuje dany termin, lub do hasła, które definiuje wskazany termin.
- *Patrz również* odsyła do hasła powiązanego z danym terminem.
- *Równoznaczne z* odsyła do hasła o niemal identycznym znaczeniu jak podane.
- *Przeciwne do* wskazuje hasło o znaczeniu zasadniczo odmiennym od bieżącego.

## Definicje

### A

**adaptacja** (ang. *adaptation*) — jeden z trzech filarów empirycznej kontroli nad procesem. Informacja zwrotna służy do dostosowania prac nad powstającym produktem, a także samego procesu, zgodnie z którym jest on produkowany. Patrz również *empiryczna kontrola nad procesem, inspekcja i przejrzystość*.

**agile** — 1. określony zestaw wartości i reguł wyrażony w Manifeście Agile [Beck i inni, 2001]. 2. zbiorcze określenie używane do wskazania grupy powiązanych ze sobą metodologii tworzenia oprogramowania komputerowego opartych na budowaniu w sposób interaktywny i inkrementacyjny. Scrum jest metodologią tworzenia oprogramowania zgodną z ideą agile. Patrz również *programowanie ekstremalne, kanban, Scrum*.

**aktywność** (ang. *activity*) — 1. praktyka scrumowa polegająca na podjęciu pewnego działania lub przeprowadzeniu pewnego procesu. Może to być planowanie sprintu, codzienna praca w Scrumie, przegląd sprintu czy też retrospekcja sprintu. 2. praca wykonywana przez członków zespołu scrumowego, na przykład pisanie kodu, wykonywanie testów, tworzenie ocen szacunkowych itd. Patrz również *praktyka*.

**artefakt** (ang. *artifact*) — widoczny produkt uboczny powstały w trakcie procesu produkcyjnego. Przykładami artefaktów w Scrumie są: rejestr produktu, rejestr sprintu, a także potencjalnie zdany do wdrożenia przyrostowy fragment produktu. Patrz również *praktyka*.

**ATTD** (ang. *acceptance-test-driven-development*) — patrz *produkcja sterowana testami akceptacyjnymi*.

## B

**bezczynni pracownicy, pracownicy czekający na pracę** (ang. *idle workers*) — pracownicy, którzy mają możliwość wykonania dodatkowej pracy, ponieważ ich aktualne obciążenie jest poniżej 100%. Przeciwne do *pracy czekającej na realizację*.

**budowanie iteracyjne** (ang. *iterative development*) — planowana strategia pracy, w której dobry rezultat końcowy osiąga się dzięki wielokrotnemu powtarzaniu pracy. Patrz również *budowanie przyrostowe, iteracja, proces iteracyjny i przyrostowy*.

**budowanie przyrostowe** (ang. *incremental development*) — 1. sposób budowania rozwiązania polegający na zbudowaniu pewnej części przed zbudowaniem całości. 2. strategia faz, według której części produktu są wytwarzane i dostarczane użytkownikom w różnych okresach czasu z zamiarem adaptacji do informacji zwrotnej pochodzącej z zewnętrz. Patrz również *proces interaktywny i przyrostowy, budowanie przyrostowe*.

## C

**całość przed kolejnym krokiem** (ang. *all-before-any*) — określenie sekwencyjnego procesu produkcyjnego, w którym wynik pracy z poprzedniego kroku tego procesu przekazywany jest do następnego kroku w formie 100% zapotrzebowania. Patrz również *rozmiar zapotrzebowania*.

**cecha** (ang. *feature*) — 1. kawałek funkcjonalności biznesowej, która ma znaczenie dla klienta lub użytkownika. 2. pojęcie stosowane przez niektórych do opisania średniej wielkości historyjki użytkownika, która może będzie podzielona na serię mniejszych historyjek, a te z kolei zostaną zaimplementowane i dostarczą wartość cechy. Patrz również *temat, historyjka użytkownika*.

**cechy mile widziane** (ang. *nice-to-have features*) — cechy przewidziane w nadchodzącej wersji dystrybucyjnej, ale możliwe do wykluczenia, jeśli nie starczy zasobów do ichtworzenia. Przeciwne do *cech obowiązkowych, cech wykluczonych*.

**cechy obowiązkowe** (ang. *must-have features*) — zestaw cech, który musi pojawić się w nadchodzącej dystrybucji produktu, aby można było uznać ją za opłacalną. Równoznaczne z *minimalnym zestawem cech kwalifikujących do dystrybucji*. Przeciwne do *cech mile widzianych, cech wykluczonych*.

**cechy wykluczone** (ang. *won't have features*) — zbiór cech, które celowo zostały określone jako niedostępne w nadchodzącej wersji dystrybucyjnej. Przeciwne do *cech obowiązkowych, cech mile widzianych*.

**cel sprintu** (ang. *sprint goal*) — ogólne stwierdzenie celu, jaki właściciel produktu chciałby osiągnąć podczas sprintu. Często cel ten wyrażony jest poprzez wyspecyfikowany zestaw historyjek z rejestru produktu.

**cel wersji dystrybucyjnej** (ang. *release goal*) — jasne wyrażenie celu i spodziewanego wyniku stworzenia wersji dystrybucyjnej. Przy ustalaniu celu rozważanych jest wiele czynników, z docelowymi klientami, problemami architektonicznymi wysokiego rzędu oraz ważniejszymi wydarzeniami na rynku włącznie. Patrz również *wersja dystrybucyjna*.

**celowy dług techniczny** (ang. *targeted technical debt*) — określenie statusu dłużu technicznego, który jest znany i został skierowany do obsługi przez zespół deweloperski. Przeciwne do *nieoczekiwane dług techniczny, znanego dług techniczny*. Patrz również *dług techniczny*.

**ceremonia** (ang. *ceremony*) — wydarzenie związane z określonymi rytuałami lub symboliką, mające miejsce z powodu dobrze wszystkim znanych okazji. Niektóre osoby określają mianem ceremonii tak fundamentalne aktywności scrumowe jak planowanie sprintu, codzienne spotkania scrumowe, przegląd sprintu czy retrospekcja. Patrz również *aktywność, niepotrzebne formalności*.

**ciągła dystrybucja** (ang. *continuous deployment*) — dostarczanie użytkownikom każdej nowej cechy produktu bezpośrednio po jej zbudowaniu, zintegrowaniu i przetestowaniu. Równoznaczne z *integracją*.

**ciągła integracja** (ang. *continuous integration*) — praktyka polegająca na integrowaniu swojej pracy przez członków pojedynczego zespołu lub przez kilka zespołów tak często, aby dawało to użyteczny efekt.

**ciche grupowanie** (ang. *silent group*) — technika ułatwiania służąca ludziom do grupowania związanych ze sobą elementów bez dyskusji i polegająca wyłącznie na rozmieszczaniu i przesuwaniu poszczególnych elementów (zazwyczaj są to karty lub notki samoprzyklepne) jako środków służących do komunikacji i koordynacji pomiędzy uczestnikami. Metoda stosowana często podczas retrospekcji sprintu. Patrz również *retrospekcja sprintu*.

**codzienne działania scrumowe** (ang. *daily Scrum*) — synchronizacja, analiza i planowanie adaptacyjne — działania, które zespół scrumowy podejmuje każdego dnia. Fundamentalna zasada działania w Scrumie ograniczona do nie więcej niż 15 minut. Równoznaczne z *codziennym spotkaniem scrumowym*. Patrz również *analiza i adaptacja*.

**codzienne spotkanie scrumowe** (ang. *daily stand-up*) — powszechnie stosowane podejście do wykonywania codziennych działań scrumowych polegające na przeprowadzeniu spotkania, w trakcie którego wszyscy uczestnicy stoją do momentu zakończenia obrad. Przeprowadzenie spotkania w pozycji stojącej pomaga zachować związkowość i przeciwdziała przedłużaniu ponad zaplanowane ramy czasowe. Patrz również *codzienne działania scrumowe*.

**Cynefin** (ang. *Cynefin*) — środowisko organizujące świadomość. Jego celem jest pomoc w zrozumieniu sytuacji, w której musimy działać i decydować w sposób odpowiedni do bieżących warunków [Snowden i Boone, 2007].

**czarterowanie projektu** (ang. *project chartering*) — praca przygotowawcza niezbędna do zdefiniowania projektu na poziomie dostatecznie szczegółowym, aby można było podjąć decyzję o jego finansowaniu. Równoznaczne z *powstaniem projektu,inicjalizacją projektu*.

## D

**DEEP** (ang. *Detailed appropriately, Emergent, Estimated i Prioritized*) — skrót stworzony przez Romana Pichlera i Mike'a Cohena ułatwiający zapamiętanie kryteriów służących do oceny jakości rejestru produktu. Te kryteria to: *odpowiedni stopień uszczegółowienia, emergencja, przypisanie ocen, przypisanie priorytetów*. Patrz również *rejestr produktu*.

**definicja gotowości** (ang. *definition of ready*) — lista kontrolna warunków, które muszą zostać spełnione podczas planowania sprintu, zanim będzie można uznać, że historyjka z rejestru produktu jest gotowa do umieszczenia w sprincie. Przeciwne do *definicji ukończenia*.

**definicja ukończenia** (ang. *definition of done*) — 1. lista kontrolna różnych rodzajów zadań, jakie zespół musi zrealizować z powodzeniem przed końcem sprintu, aby móc określić swoją pracę mianem potencjalnie nadającego się do wypuszczenia. Absolutne minimum definicji ukończenia powinno owocować kompletnym fragmentem funkcjonalności produktu — takim, który został zaproektowany, zbudowany, zintegrowany, przetestowany, udokumentowany i dostarcza potwierdzoną wartość dla klienta. 2. czasami określana mianem kryteriów akceptacji, które mają zastosowanie do wszystkich historyjek z rejestru produktu. Przeciwne do *definicji gotowości*.

**demonstracja sprintu** (ang. *sprint demo*) — 1. przegląd sprintu, podczas którego gotowe historyjki (zgodnie z kryteriami ukończenia prac) z rejestru produktu są demonstrowane z zamiarem prowadzenia dyskusji pomiędzy członkami zespołu scrumowego a pozostałymi uczestnikami tego spotkania. Patrz również *przegląd sprintu*.

**dług techniczny** (ang. *technical debt*) — 1. termin służący do opisania zobowiązania podejmowanego przez firmę zajmującą się tworzeniem oprogramowania, która zdecydowała się zaprojektować lub skonstruować system. Dług techniczny jest na ogół mały w krótkim horyzoncie czasowym, ale rośnie (również pod względem kosztu ekonomicznego) wraz ze wzrostem stopnia skomplikowania oprogramowania. 2. metafora ułatwiająca komunikację pomiędzy osobami zajmującymi się stroną biznesową i inżynierijną produktu podczas omawiania możliwych negatywnych artefaktów będących następstwem implementacji. Patrz również *natywny dług techniczny, strategiczny dług techniczny, niedostępny dług techniczny*.

**dokładność** (ang. *accuracy*) — jak bardzo wstępna wartość pokrywa się z wartością faktyczną — stopień zbliżenia estymacji do prawdziwej wartości. Na przykład o wypuszczeniu produktu w październiku 2015 roku można powiedzieć, że jest dokładne, jeżeli zdarzenie to nastąpi dowolnego dnia października 2015. Przeciwne do *precyzyj*.

**domena chaosu** (ang. *chaotic domain*) — 1. sytuacja wymagająca natychmiastowej reakcji. Nastąpił kryzys i musimy działać bezzwłocznie, aby zapobiec dalszym stratom i przywrócić przynajmniej względny porządek. 2. jedna z domen w środowisku Cynefin. Patrz również *Cynefin*. Przeciwne do *domeny złożonej, domeny skomplikowanej, domeny nieporządku, domeny prostej*.

**domena nieporządku** (ang. *disorder domain*) — 1. stan niebezpieczny, w trakcie którego praktycznie nie rozumiemy sytuacji, w jakiej się znaleźliśmy. Naszym celem jest wydostanie się z tego stanu. 2. jeden ze stanów środowiska Cynefin. Patrz również *Cynefin*. Przeciwne do *domeny chaosu, domeny złożonej, domeny skomplikowanej, domeny nieporządku*.

**domena prostoty** (ang. *simple domain*) — 1. sytuacja, w której każdy widzi przyczyny i ich skutki. Często właściwa odpowiedź jest oczywista i nie wymaga komentarza. 2. jedna z domen w środowisku Cynefin. Patrz również *Cynefin*. Przeciwne do *domeny chaotycznej, domeny złożonej, domeny skomplikowanej, domeny nieporządku*.

**domena skomplikowana** (ang. *complicated domain*) — 1. sytuacja, w której może istnieć wiele odpowiedzi, ale do ich znalezienia potrzebna jest diagnoza eksperta. 2. jedna z domen w środowisku Cynefin. Patrz również *Cynefin*. Przeciwne do *domeny chaosu, domeny złożonej, domeny nieporządku, domeny prostoty*.

**domena złożona** (ang. *complex domain*) — 1. sytuacja, w której rzeczy są bardziej nieprzewidywalne niż przewidywalne. Jeżeli istnieje prawidłowa odpowiedź, poznamy ją dopiero, patrząc wstecz. 2. jedna z domen w środowisku Cynefin. Patrz również *Cynefin*. Przeciwne do *domeny chaosu, domeny skomplikowanej, domeny nieporządku, domeny prostoty*.

**dystrybucja z ustaloną datą** (ang. *fixed-date release*) — dystrybucja, która musi zostać wypuszczona w pewnym ustalonym dniu w przyszłości. Zakres tej wersji dystrybucyjnej, a być może również jej koszt muszą być elastyczne. Porównaj do *dystrybucji z ustalonym zakresem*.

**dystrybucja z ustalonym zakresem** (ang. *fixed-scope release*) — dystrybucja, która musi posiadać ustalony zestaw cech. Dzień, w którym ta wersja dystrybucyjna musi być dostarczona, i jej koszt są elastyczne. Przeciwne do *dystrybucji z ustaloną datą*.

**działanie w roju** (ang. *swarming*) — zachowanie, zgodnie z którym członkowie zespołu posiadający wolny czas i odpowiednie umiejętności pracują wspólnie (w roju) nad elementem, który został już rozpoczęty przed przejściem dalej i przystąpieniem do pracy nad nowymi elementami. Patrz również *zdolności typu T*.

## E

**eksplotacja** (ang. *exploitation*) — podejmowanie decyzji w oparciu o pewność danych, które posiadamy w tym momencie. Przeciwne do *eksploracji*.

**eksploracja** (ang. *exploration*) — czynność polegająca na pozyskiwaniu lub kupowaniu wiedzy pochodzącej z jakiegoś rodzaju aktywności, na przykład budowanie prototypu, przeprowadzanie badań lub wykonywanie eksperymentów. Przeciwne do *eksplotacji*.

**emergencja** (ang. *emergence*) — 1. indywidualne, lokalne zachowanie, które przekłada się na zachowanie globalne niezależne od swojego oryginalnego źródła. 2. atrybut złożonych systemów adaptacyjnych. 3. w odniesieniu do tworzenia oprogramowania — świadomość, że nie ma możliwości określenia *a priori* właściwego zbioru funkcjonalności, projektów lub planów. Zamiast tego istotne informacje „pojawiają się” w miarę poznawania systemu i zdobywania doświadczenia w oparciu o pracę już wykonaną. Patrz również *złożone systemy adaptacyjne*.

**empiryczna kontrola nad procesem** (ang. *empirical process control*) — styl pracy, który kładzie nacisk na zasady inspekcji, adaptacji i przejrzystości. Przeciwne do *procesu zdefiniowanego*.

**epos** (ang. *epic*) — duża historia użytkownika o rozmiarze sięgającym nawet kilku miesięcy, która może wypełnić całą wersję dystrybucyjną lub nawet kilka z nich. Eposy są dobrym sposobem przechowywania wymagań o dużych rozmiarach. W odpowiednim czasie eposy są stopniowo przekształcane na zbiory małych historyjek użytkownika. Patrz również: *przyszłość, stopniowe uszczegółowianie, temat, historyjka użytkownika*.

**esencja Scruma** (ang. *essential Scrum*) — wartości, zasady i praktyki powiązane z regułami i przetestowanymi sposobami realizowania wytycznych środowiska Scrum. Patrz również: *podejście, praktyka, reguła, środowisko Scrum*.

**estymacja** (ang. *estimation*) — przybliżona ocena wartości, liczby, ilości lub rozmiaru czegoś. W Scrumie estymujemy rozmiar historyjek znajdujących się w rejestrze portfela produktów, rejestrze produktu, a także zadań w rejestrze sprintu. Patrz również *prognoza*.

## F

**filtr ekonomiczny** (ang. *economic filter*) — kryteria decyzyjne używane przez organizację do określenia rachunku ekonomicznego produktu mające na celu podjęcie decyzji o jego finansowaniu. Przeciwne do *filtra strategicznego*.

**filtr strategiczny** (ang. *strategic filter*) — kryteria decyzyjne stosowane przez organizację w celu określenia, czy produkt spełnia założenia strategiczne kwalifikujące go do dalszej analizy. Przeciwne do *filtra ekonomicznego*.

**finansowanie przyrostowe** (ang. *incremental funding*) — finansowanie pewnej części prac nad produktem bez zobowiązania do finansowania całości. Używając finansowania przyrostowego, wydajemy pieniądze jedynie na pierwszą małą część prac deweloperskich, a następnie wracamy do decyzji o finansowaniu po uzyskaniu kluczowej wiedzy, za którą zapłaciliśmy w pierwszym kroku. Patrz również *poziom pewności, wiedza potwierdzona*.

## G

**głosowanie kropkowe** (ang. *dot voting*) — technika pozwalająca uczestnikom wyrazić swoje preferencje odnośnie do zbioru historyjek poprzez umieszczanie kolorowych kropek przy elementach zbioru, które ich zdaniem mają wyższy priorytet niż pozostałe. Historyjki z większą liczbą kropek mają wyższy priorytet od tych z mniejszą ilością. Metoda ta jest często stosowana podczas retrospekcji sprintu. Patrz również *retrospekcja sprintu*.

**grupa** (ang. *group*) — ludzie określani tą samą nazwą (nazwą grupy), ale nie uformowani jeszcze jako zespół, którego członkowie nauczyli się pracować ze sobą i ufać sobie nawzajem. Przeciwnie do *zespołu*.

## H

**historyjka** (ang. *story*) — patrz *historyjka użytkownika*.

**historyjka nadająca się do sprintu** (ang. *sprintable story*) — patrz *historyjka nadająca się do implementacji w sprincie*.

**historyjka użytkownika** (ang. *user story*) — wygodny format służący do wyrażania pożąданiej wartości biznesowej w różnorodnych elementach rejestru produktu. Historyjki użytkownika są konstruowane w taki sposób, aby mogły być zrozumiane w łatwy sposób zarówno przez ludzi ze strony biznesowej, jak i przez inżynierów. Mają prostą strukturę i zazwyczaj wyrażone są w formie zdania takiego jak „Jako <rola użytkownika> chcę osiągnąć <cel>, abym mógł otrzymać <korzyść>”. Są doskonałym miejscem do prowadzenia konwersacji. Mogą być pisane z różnym poziomem uszczegółowienia, a następnie stopniowo wzbogacane o detale. Patrz również *epos, stopniowe udoskonalanie, temat, rola użytkownika*.

**historyjka z rejestru produktu** (ang. *product backlog item — PBI*) — 1. cecha produktu, błąd lub (sporadycznie) zadanie inżynierijne mające wartość z punktu widzenia właściciela produktu. 2. element w rejestrze produktu. Patrz również *rejestr produktu*.

**historyjka zdolna do implementacji** (ang. *implementable story*) — historyjka użytkownika o rozmiarze wystarczająco małym, aby zmieścić się swobodnie w sprincie. Równoznaczne z *historyjką nadającą się do sprintu*.

**historyjki techniczne** (ang. *technical stories*) — historyjki „użytkownika” (należące do rejestru produktu), które nie dostarczają widocznej wartości dla użytkownika, ale w sposób istotny rozbudowują architekturę lub infrastrukturę potrzebną do dostarczenia takiej wartości w przyszłości.

## I

**idealna godzina** (ang. *ideal hour*) — jednostka służąca do określenia rozmiaru pracy wyrażonej przez zadania w rejestrze sprintu jako projektowanie, budowanie, integrowanie i testowanie. Często określana mianem roboczogodziny lub osobogodziny. Patrz również *idealny dzień*.

**idealny dzień** (ang. *ideal day*) — jednostka służąca do estymowania czasu pracy nad historyjką z rejestru produktu zakładającą, że praca dotyczyłaby wyłączenie tej historyjki, bez przerwań i z bezzwłocznym dostępem do wszelkich potrzebnych zasobów. Patrz również *idealna godzina*. Przeciwnie do pojęcia *punkt historyjkowy*.

**inflacja punktowa** (ang. *point inflation*) — niefortunne zjawisko inflacji ocen w rejestrze produktu mające na celu dostosowanie lub zoptymalizowanie nierozsądnie założonej wartości (takiej jak osiągnięcie docelowej prędkości).

**inicjalizacja projektu** (ang. *project initiation*) — patrz *czarterowanie projektu*.

**inspekcja** (ang. *inspection*) — jeden z trzech filarów empirycznej kontroli nad procesem cechujący się przemyślanym badaniem i analizowaniem informacji zwrotnej w celu podjęcia decyzji adaptacyjnych odnośnie do procesu lub produktu. Patrz również *adaptacja, empiryczna kontrola nad procesem, przejrzystość*.

**inspekcja i adaptacja** (ang. *inspect and adapt*) — 1. znana faza w procesie scrumowym odnosząca się do zasad inspekcji i adaptacji w empirycznej kontroli nad procesem. 2. zasada inspekcji produktu lub procesu, a następnie dokonywania adaptacji na podstawie zdobytej wiedzy. 3. kluczowy element pętli zdobywania wiedzy. Patrz również *adaptacja, empiryczna kontrola nad procesem, inspekcja, pętla zdobywania wiedzy*.

**integracja** (ang. *integration*) — połączenie różnych komponentów lub zasobów w celu utworzenia większego, spójnego produktu, który można przetestować jako całość pod względem prawidłowego funkcjonowania. Patrz również *ciągła integracja*.

**interesariusz** (ang. *stakeholder*) — osoba, grupa lub organizacja, która może wpływać na działania podejmowane przez organizację lub podlegać wpływom podejmowanych decyzji.

**interesariusze wewnętrzni** (ang. *internal stakeholders*) — interesariusze znajdujący się wewnątrz organizacji wytwarzającej produkt, na przykład wyższa kadra kierownicza, menedżerowie i użytkownicy wewnętrzni. Patrz również *interesariusze*. Przeciwne do *interesariuszy zewnętrznych*.

**interesariusze zewnętrzni** (ang. *external stakeholder*) — interesariusze, którzy przeważnie nie należą do organizacji zajmującej się stworzeniem produktu, na przykład klienci, partnerzy lub przedstawiciele urzędów regulacyjnych. Patrz również *interesariusze*. Przeciwne do *interesariusze wewnętrzni*.

**INVEST** — skrót stworzony przez Billa Wake'a mający na celu łatwe zapamiętanie zestawu kryteriów stosowanych do oceny jakości historyjek użytkownika. Są to kolejno: niezależność (ang. *independent*), negocjalność (ang. *negotiable*), wartościowość (ang. *valuable*), ocenialność (ang. *estimatable*), dobry (mały) rozmiar (ang. *sized correctly*) i testowalność (ang. *testable*). Patrz również *historyjka użytkownika*.

**inwentarz** (ang. *inventory*) — patrz *aktywna praca*.

**iteracja** (ang. *iteration*) — zamknięty proces produkcyjny skupiony na wykonaniu całej niezbędnej pracy potrzebnej do wytworzenia wyniku o wymiernej wartości. Patrz również *budowanie przyrostowe, budowanie iteracyjne*.

## K

**kanban** — podejście zgodne z ideologią Agile nałożone na istniejący proces w celu widocznego przedstawienia przepływu pracy w systemie, ograniczenia aktywnej pracy, a także zmierzenia jej nakładu i zoptymalizowania przepływu. Patrz również *zwinność, aktywna praca*.

**kolejka** (ang. *queue*) — miejsce do przechowywania elementów (inwentarza) w trakcie ich oczekiwania na kolejne działania w potoku pracy. Patrz również *inwentarz, praca cząstkowa*.

**koniec pracy** (ang. *done*) — zobacz *definicja ukończenia*.

**koszt opóźnienia** (ang. *cost of delay*) — koszt finansowy wynikający z opóźnień w pracy lub w osiągnięciu kamienia milowego. Koszt opóźnienia podkreśla ideę przypisywania rzeczywistej wartości finansowej do czasu, a także wagę posiadania wiedzy na temat tej wartości w celu podejmowania ekonomicznie uzasadnionych kompromisów.

**kryteria akceptacji** (ang. *acceptance criteria*) — 1. wyspecyfikowany przez właściciela produktu lub osobę odpowiedzialną za stronę biznesową przedsięwzięcia zewnętrzny zestaw cech jakościowych produktu. Kryteria akceptacji określają spodziewane zachowanie i służą do sprawdzenia, czy historia z rejestru została zrealizowana w sposób pomyślny. 2. kryteria wyjściowe, jakie komponent lub system musi spełnić, aby zostać zaakceptowanym przez użytkownika, klienta lub inną osobę uprawnioną [IEEE 610].

**kurczaki** (ang. *chickens*) — metafora używana przez niektóre zespoły scrumowe do opisania ludzi inwestujących w cel zespołu na poziomie zaangażowania, ale bez poświęcenia. Hasło to nadaje się najlepiej do opisania osób niebędących częścią zespołu scrumowego — pochodzi ze starego dowcipu o kurczaku i świń: „Do powstania jajecznicy na bekonne wystarczyło jedynie zaangażowanie kury i poświęcenie świń”. Przeciwnie do świń.

## L

**lider służby** (ang. *servant leader*) — 1. osoba osiągająca wyniki dla swojej organizacji poprzez skupianie największej uwagi na potrzebach swoich kolegów oraz tych, którym służy. 2. filozofia i praktyka przywództwa bazująca na wsłuchiwaniu się, empatii, leczeniu, świadomości, przekonywaniu, postrzeganiu, przewidywaniu, służbie, poświęceniu i budowaniu wspólnoty. Patrz również *mistrz młyna*.

**linia zdarzeń** (ang. *event timeline*) — graficzne, chronologiczne przedstawienie znaczących zdarzeń, które miały miejsce w określonym przedziale czasu. Jest to jedna z technik powszechnie stosowanych podczas retrospekcji sprintu. Patrz również *retrospekcja sprintu*.

**LRM** — patrz *ostateczny moment podjęcia decyzji*.

## M

**marnotrawstwo** (ang. *waste*) — dowolna działalność, która konsumuje zasoby i nie powoduje dodania wartości do produktu lub usługi dostarczanych klientowi.

**minimalny opłacalny produkt** (ang. *minimum viable product* — MVP) — produkt posiadający wyłącznie te cechy, które pozwalają na jego dystrybucję i nic więcej.

**minimalny zestaw cech kwalifikujący do dystrybucji** (ang. *minimum releasable features* — MRF) — 1. minimalny zestaw cech potrzebnych w produkcji do uznania go za wartościowy — dostatecznie użyteczny dla klientów końcowych, aby ci byli chętni do jego kupienia. 2. cechy złożone z minimalnego zestawu cech kwalifikujących do sprzedaży. Równoznaczne z *cechami obowiązkowymi*. Patrz również *minimalny zestaw cech kwalifikujący do sprzedaży*.

**minimalny zestaw cech kwalifikujący do sprzedaży** (ang. *minimum marketable features — MMF*) — minimalny zbiór funkcjonalności związanych z cechą, który musi zostać dostarczony klientowi, aby ten uznał ją za wartościową (godną zakupienia). Przeciwne do *minimalnego zestawu cech kwalifikującego do dystrybucji*.

**mistrz młyna** (ang. *Scrum master*) — trener, animator, osoba odpowiedzialna za usuwanie przeszkód i służenie zespołowi scrumowemu. Mistrz młyna jest jedną z trzech ról w zespole scrumowym. Jego zadaniem jest nadzorowanie procesu oraz pomaganie zespołowi i nie tylko w podwyższaniu swojej wydajności i zgodności z regułami Scruma obowiązującymi w organizacji. Patrz również *zespół scrumowy, lider służby*.

**MMF** — patrz *minimalny zestaw cech kwalifikujący do sprzedaży*.

**MRF** — patrz *minimalny zestaw cech kwalifikujący do dystrybucji*.

**MVP** — patrz *minimalny opłacalny produkt*.

## N

**nadarzająca się okazja** (ang. *emergent opportunity*) — możliwość, która była wcześniej nieznana lub której wystąpienie było traktowane jako bardzo mało prawdopodobne i w związku z tym nie-warte poświęcania pieniędzy w tym konkretnym momencie.

**nastawienie muszkietera** (ang. *Musketeer attitude*) — 1. jeden za wszystkich, wszyscy za jednego. 2. nastawienie wśród członków zespołu sprowadzające się do tego, iż wszyscyjadą na tym samym wozie i albo zwyciężą wspólnie, albo polegną wspólnie.

**natywny dług techniczny** (ang. *native technical debt*) — forma dlużu technicznego, który narasta ze względu na nieodpowiedzialne zachowanie lub niedopracowane praktyki po stronie zaangażowanych ludzi. Przeciwne do *strategicznego dlużu technicznego, nieuniknionego dlużu technicznego*. Patrz również *dług techniczny*.

**nieoczekiwany dług techniczny** (ang. *happened-upon technical debt*) — określenie statusu dlużu technicznego, o którego istnieniu zespół deweloperski nie miał pojęcia, a który został ujawniony w czasie codziennej pracy nad produktem. Przeciwne do *znanego dlużu technicznego, celowego dlużu technicznego*. Patrz również *dług techniczny*.

**niepewność** (ang. *uncertainty*) — coś nieznanego i nieustalonego. Niepewność jest często traktowana na równi z ryzykiem, ma jednak szerszy zasięg, ponieważ obejmuje zarówno ryzyko (wynik negatywny), jak i okazje (wynik pozytywny). Patrz również *ryzyko*.

**niepewność klienta** (ang. *customer uncertainty*) — brak pewnej wiedzy na temat tego, kim są klienci produktu. Patrz również *niepewność*. Przeciwne do *niepewność końca, niepewność środków*.

**niepewność końca** (ang. *end uncertainty*) — niepewność dotycząca tego, co zostanie zbudowane (produkту). Patrz również *niepewność*. Przeciwne do *niepewności klienta, niepewności środków*.

**niepewność środków** (ang. *means uncertainty*) — niepewność otaczająca metodologię zbudowania czegoś. Patrz również *niepewność*. Przeciwne do *niepewności klienta, niepewności końca*.

**niepotrzebne formalności** (ang. *unnecessary formality*) — 1. ceremonia, która wymaga rzeczywistych nakładów, a przynosi bardzo znikomą lub wręcz zerową wartość (forma marnotrawstwa). 2. „sztuka dla sztuki”. Patrz również *ceremonia, marnotrawstwo*.

**nieunikniony dług techniczny** (ang. *unavoidable technical debt*) — forma dłużu technicznego, którego na ogół nie można przewidzieć, ani też mu zapobiec. Dług taki narasta bez winy zespołu pracującego nad produktem. Przeciwne do pojęć *natywny dług techniczny, strategiczny dług techniczny*. Patrz również *dług techniczny*.

**nieznane nieznane** (ang. *unknown unknowns*) — nieznane rzeczy, o których jeszcze nie wiemy.

## 0

**ograniczanie czasowe** (ang. *timeboxing*) — technika zarządzania czasem, która pozwala zarządzać wydajnością i zakresem prac. Patrz również *ograniczenie czasowe*.

**ograniczenie czasowe** (ang. *timebox*) — ograniczony przedział czasu, w trakcie którego wykonywane jest zadanie. W Scrumie sprinty są ograniczonymi w czasie iteracjami, w trakcie których zespół pracuje w podtrzymywальным tempie, aby ukończyć wybrany fragment pracy cząstkowej. Patrz również *sprint, ograniczanie czasowe*.

**ostateczny moment podjęcia decyzji** (ang. *last responsible moment — LRM*) — strategia polegająca na niepodejmowaniu przedwcześniekszych decyzji, opóźnianiu zaangażowania oraz pozostawianiu ważnych i nieodwracalnych decyzji otwartymi do momentu, kiedy koszt niepodjęcia decyzji przewyższy koszt jej podjęcia.

## P

**PBI** (ang. *product backlog item*) — patrz *historyjka z rejestru produktu*.

**pełnomocnik właściciela produktu** (ang. *product owner proxy*) — osoba wyznaczona przez właściciela produktu do wykonywania jego obowiązków w określonych sytuacjach. Patrz również *właściciel produktu*.

**persona** — 1. archetyp użytkownika stworzony na podstawie danych etnograficznych rzeczywistych użytkowników pomagający w podejmowaniu decyzji odnośnie do cech produktu, nawigacji, interakcji i projektu graficznego. 2. fikcyjna osoba będąca prototypem danej roli użytkownika. Patrz również *historyjka użytkownika*.

**pętla zdobywania wiedzy** (ang. *learning loop*) — pętla ze sprzężeniem zwrotnym mająca na celu zwiększenie wiedzy. Przebiega ona na ogół w następujący sposób: stwórz założenie (lub ustaw cel), zbuduj coś (przeprowadź działania), pobierz informację zwrotną ze zbudowanego produktu, a następnie przeanalizuj ją pod kątem tego, co zostało osiągnięte w porównaniu z początkowymi założeniami.

**pielęgnacja** (ang. *grooming*) — patrz *pielęgnacja rejestru produktu*.

**pielęgnacja rejestru produktu** (ang. *product backlog grooming*) — czynności związane z pisaniem, doprecyzowaniem, ocenianiem i nadawaniem priorytetów historyjkom z rejestru produktu.

**plan wersji dystrybucyjnej** (ang. *release plan*) — 1. wynik planowania wersji dystrybucyjnej. W przypadku wersji dystrybucyjnej z ustaloną datą wypuszczenia plan będzie określał zestaw cech dostępnych w ustalonym przyszłym dniu. W przypadku wersji dystrybucyjnej z ustalonym zakresem plan będzie określał liczbę sprintów i koszt potrzebny do dostarczenia ustalonego zakresu cech. 2. plan, który dostarcza możliwie najdokładniejszych informacji na temat tego, kiedy będzie dostępna wersja dystrybucyjna, jakie będzie zawierać cechy, jak dużo będzie kosztować. Patrz również *dystrybucja z ustaloną datą*, *dystrybucja z ustalonym zakresem*.

**planowanie pokerowe** (ang. *planning poker*) — technika relatywnego oceniania historyjek z rejestru bazująca na osiąganiu porozumienia.

**planowanie portfela** (ang. *portfolio planning*) — aktywność polegająca na określaniu produktów (lub projektów), nad którymi będziemy pracować, kolejności prac nad nimi, a także czasu, jaki poświęcimy każdemu z nich. Czasem działanie to określa się mianem zarządzania portfelem.

**planowanie produktu** (ang. *product planning*) — patrz *tworzenie wizji produktu*.

**planowanie sprintu** (ang. *sprint planning*) — czas, kiedy członkowie zespołu scrumowego spotykają się, aby uzgodnić cel sprintu, a także ustalić, jaki podzbiór historyjek z rejestru produktu mogą wyprodukować podczas nadchodzącego sprintu. Podczas planowania sprintu powstaje rejestr sprintu, którego celem jest pomóc zespołowi w nabraniu przekonania co do wykonalności historyjek, których realizacji chcą się podjąć. Patrz również *rejestr sprintu*, *cel sprintu*.

**planowanie wersji dystrybucyjnej** (ang. *release planning*) — planowanie w długim horyzoncie czasowym, które odpowiada na pytania takie jak: „Kiedy skończymy?”, „Jakie cechy mogą zostać zrealizowane do końca roku?” lub „Ile to będzie kosztować?”. Planowanie wersji dystrybucyjnej musi bilansować ze sobą wartość dla klienta z uwzględnieniem jakości, ograniczenia zakresu, harmonogramu i budżet. Patrz również *plan wersji dystrybucyjnej*.

**pociąg wersji dystrybucyjnych** (ang. *release train*) — podejście polegające na dostosowaniu wizji, planowania i wewnętrznych zależności między zespołami poprzez wprowadzenie synchronizacji międzyspołowej bazującej na wspólnym taktie. Pociąg wersji dystrybucyjnych skupia się na szybkim, elastycznym przepływie na poziomie większego produktu. Patrz również *scrum scrumów*.

**podejście** (ang. *approach*) — specyficzny sposób realizacji zadań praktycznych lub aktywności. Dla przykładu: Scrum definiuje pojęcie retrospekcji sprintu. Sposób przeprowadzenia retrospekcji przez zespół stanowi jego podejście do tego zagadnienia. Może być ono zupełnie odmienne od podejścia innych zespołów. Patrz również *aktywność*, *praktyka*.

**pojemność** (ang. *capacity*) — 1. ilość dostępnych zasobów pozwalających na wykonanie użytecznej pracy. 2. pojęcie pomagające ustalić zakres pracy częściowej przez zapewnienie, iż praca zostanie rozpoczęta, wyłącznie jeśli dostępny czas pozwoli na jej pomyślne wykonanie. Patrz również *praca częściowa*.

**poświęcenie** (ang. *commitment*) — akt przywiązania się do pewnego działania. Scrum zachęca do poświęcenia. Poświęcenie oznacza, że zarówno w dobrych, jak i złych czasach każdy członek zespołu stara się osiągnąć wspólny cel zespołu. Przeciwne do *prognozy*.

**potencjalnie zdatny do wdrożenia przyrost produktu** (ang. *potentially shippable product increment*) — zrealizowana praca, co do której istnieje wysoki stopień pewności, iż jest ona dobrej jakości i nadaje się do dostarczenia klientom końcowym pod koniec sprintu. Potencjalna zdolność do dostarczenia klientom nie oznacza, że nowa funkcjonalność zostanie faktycznie dostarczona. Wdrożenie jest decyzją biznesową. Potencjalna zdolność do wdrożenia to jedynie stan pewności.

**powstanie projektu** (ang. *project inception*) — patrz *czarterowanie projektu*.

**poziom zaufania** (ang. *confidence threshold*) — 1. definicja udanego zakończenia służąca do tworzenia wizji produktu (planowanie na poziomie produktu). 2. zbiór informacji, jakich potrzebuje kierownictwo, aby móc podjąć decyzję o kontynuacji lub wstrzymaniu finansowania dalszych prac produkcyjnych na niższym poziomie.

**praca częstkowa** (ang. *work in progress — WIP*) — praca, która weszła w fazę produkcji, ale nie została jeszcze ukończona i nie jest dostępna dla klienta lub użytkownika. Dotyczy wszystkich zasobów lub elementów produktu, usługi aktualnie realizowanych lub czekających w kolejce na realizację.

**praca czekająca na realizację** (ang. *idle work*) — praca, która nie jest aktywnie realizowana, ponieważ czeka w kolejce. Przeciwne do *bezcennych pracowników*.

**praktyka** (ang. *practice*) — sposób, w jaki pewna zasada powinna być wspierana lub realizowana. Na przykład w Scrumie zasada prezentowania postępów powinna być realizowana w formie przeglądu sprintu. Patrz *aktywność, artefakt, rola, reguła*. Patrz również *zasada, wartości*.

**praktyki techniczne** (ang. *technical practices*) — specyficzne techniki lub praktyki stosowane podczas wykonania sprintu w celu właściwego wykonania pracy mającej na celu dostarczenie cech o rozsądny stanie dłużu technicznego i spełniających kryteria wykonanej pracy, jakie narzucił sobie zespół scrumowy.

**precyza** (ang. *precyzja*) — trafność estymacji. Na przykład powiedzenie, że produkt zostanie wdrożony 7 października 2015 roku, jest bardziej precyzyjne niż powiedzenie, że zostanie on wdrożony w październiku 2015. Przeciwne do *dokładności*.

**prędkość** (ang. *velocity*) — miara szybkości kończenia prac na jednostkę czasu. W przypadku Scruma prędkość jest typowo mierzona jako suma ocen przypisanych do elementów rejestru produktu, które zostały zrealizowane w trakcie sprintu. Prędkość jest podawana w takich samych jednostkach jak elementy rejestru produktu — są to najczęściej punkty historyjkowe lub dni idealne. Prędkość mierzy wydajność (rozmiar tego, co zostało dostarczone), a nie wynik (wartość tego, co zostało dostarczone).

**proces iteracyjny i przyrostowy** (ang. *iterative and incremental process*) — styl pracy deweloperskiej, który kładzie nacisk zarówno na budowanie iteracyjne, jak i przyrostowe. Patrz również *budowanie przyrostowe, budowanie iteracyjne*.

**proces kaskadowy** (ang. *sequential process*) — patrz *proces sterowany planem*.

**proces opisowy** (ang. *prescriptive process*) — patrz *proces sterowany planem*.

**proces przewidujący** (ang. *anticipatory process*) — patrz *proces sterowany planem*.

**proces przewidujący** (ang. *predictive process*) — patrz *proces sterowany planem*.

**proces sterowany planem** (ang. *plan-driven process*) — styl wytwarzania, w którym usiłuje się za-planować i przewidzieć z góry wszystkie cechy, jakie użytkownik chciałby posiadać w produkcie końcowym, a także określić, jak najlepiej zbudować te cechy. Plan działania przekłada się na sekwencję wykonywanych kolejno różnych faz pracy. Równoznaczne z *procesem antycypracyjnym*, *procesem prognozującym*, *procesem perspektywicznym*, *procesem sekwencyjnym*, *tradycyjnym procesem budowania*, *procesem kaskadowym*.

**proces wodospadowy** (ang. *waterfall process*) — patrz *proces sterowany planem*.

**proces zdefiniowany** (ang. *defined process*) — proces z dobrze zdefiniowanymi krokami postępowania. Dobrze zdefiniowany proces przy tych samych danych wejściowych powinien za każdym razem zwrócić identyczne dane wyjściowe (w pewnym przedziale wariancji). Przeciwne do *empirycznej kontroli nad procesem*.

**produkcia sterowana testami akceptacyjnymi** (ang. *ATTD, acceptance-test-driven development*) — proces, w którym uczestnicy przed podjęciem produkcji wspólnie dyskutują testy akceptacyjne w oparciu o przykłady, a następnie wyodrębniają zestaw konkretnych testów akceptacyjnych. Równoznaczne ze *specyfikacją w oparciu o przykład*.

**produkt** (ang. *product*) — 1. wynik wysiłków deweloperskich. 2. towar lub usługa składające się z zestawu namacalnych i nienamacalnych atrybutów satysfakcyjnych klienta, które można wymienić na pieniądze lub inne jednostki o wymiernej wartości. 3. typowo długowieczny, stabilny artefakt, na którym firmy mogą przeprowadzać jeden lub więcej projektów. Przeciwne do *projektu*.

**produkt aktywny** (ang. *in-process product*) — produkt aktualnie rozwijany, będący już w produkcji lub sprzedaży. Patrz również *planowanie portfela*.

**prognoza** (ang. *forecast*) — 1. wyrażanie poglądów, przewidywań oraz ocen co do zdarzeń, których wyniki nie zostały jeszcze zaobserwowane. 2. określenie pochodzące z „Przewodnika po Scrumie” [2011], opisujące wynik wypracowany przez zespół deweloperski w trakcie planowania sprintu. Patrz również *estymacja*. Przeciwne do *poświęcenia*.

**programowanie ekstremalne** (ang. *extreme programming — EP*) — zwinne podejście do programowania stanowiące dopełnienie Scruma. Programowanie ekstremalne wskazuje istotne techniki, których zespół deweloperski używa do zarządzania pracą na poziomie zadań w trakcie wykonania sprintu. Patrz również *zwinność*.

**programowanie sterowane testami** (ang. *test-driven development — TDD*) — 1. ewolucyjne podejście do programowania oparte na tworzeniu niedziałającego testu automatycznego przed napisaniem kodu, który sprawi, że ten test przejdzie. Po napisaniu kodu, który powoduje przejście testu, cały cykl jest powtarzany, łącznie z refaktoryzacją istniejącego kodu zapewniającą spójny projekt pomiędzy różnymi funkcjami. Celem programowania sterowanego testami jest specyfikacja, a nie walidacja — myślenie o projekcie przed napisaniem kodu, tworzenie przejrzystego kodu, który zawsze działa. 2. przykład programowania w stylu „najpierw test”. Patrz również *refaktoryzacja, umiejętności techniczne, programowanie w stylu „najpierw test”*.

**programowanie w stylu „najpierw test”** (ang. *test-first development*) — praktyka polegająca na pisaniu testów przed przystąpieniem do prac deweloperskich. Przykładem takiej praktyki jest programowanie sterowane testami. Patrz również *umiejętności techniczne, programowanie sterowane testami*.

**projekcja historyjek** (ang. *story mapping*) — 1. technika przekształcająca perspektywę postrzegania tematu przez użytkownika na zbiór historyjek. Każde działanie użytkownika na wysokim poziomie jest rozkładane na pewien przepływ pracy, który może być uszczegółowiony jeszcze bardziej w postaci zbioru zadań. 2. dwuwymiarowa reprezentacja tradycyjnej jednowymiarowej listy historyjek z rejestru produktu. Patrz również *rejestr produktu, historyjka użytkownika*.

**projekt** (ang. *project*) — 1. krótkotrwały wysiłek powzięty w celu stworzenia unikatowego produktu, usługi lub wyniku [PMI, 2008]. 2. wysiłek mający swój koniec w momencie osiągnięcia celów będących podstawą do jego podjęcia. Pod względem czasu projekt jest krótszy od cyklu życia produktu. Bardzo często wiele projektów jest realizowanych w pełnym cyklu „od kołyski do grobu” w ramach cyklu życia produktu. Przeciwne do *produktu*.

**przegląd sprintu** (ang. *sprint review*) — aktywność polegająca na inspekcji i adaptacji pojawiającą się po każdym wykonaniu sprintu, w której trakcie zespół scrumowy prezentuje wszystkim zainteresowanym uczestnikom, co udało się osiągnąć w trakcie trwania sprintu. Przegląd sprintu daje szansę każdemu, kto był zaangażowany w wysiłek rozwijania produktu, na przeanalizowanie tego, co zostało zbudowane do tej pory i zaadaptowanie się do tego, co powstanie dalej. Patrz również *inspekcja i adaptacja, demonstracja sprintu*.

**przejrzystość** (ang. *transparency*) — otwarty dostęp do obiektywnych informacji potrzebnych do przeprowadzenia analizy i adaptacji — jeden z trzech filarów empirycznej kontroli nad procesem. Patrz również *adaptacja, empiryczna kontrola nad procesem, inspekcja*.

**przepływ** (ang. *flow*) — 1. płynny i ciągły przepływ pracy od procesu produkcyjnego do zapewnienia, że dostarczana treść posiada właściwą wartość ekonomiczną. 2. unikanie pracy czekającej na realizację z zachowaniem zasad ekonomii. 3. przeciwieństwo ogromnego nakładu pracy, dużej wersji dystrybucyjnej i wielkiego wybuchu.

**przepływ jednoelementowy** (ang. *single-piece flow*) — stan, w którym elementy są produkowane pojedynczo i przepływają (są przeciągane przez) proces deweloperski jako jedna spójna całość.

**punkt historyjkowy** (ang. *story point*) — miara względnego rozmiaru historyjki z rejestru produktu, która bierze pod uwagę takie fakty jak złożoność i rozmiar fizyczny. Ustalana zazwyczaj przez planowanie pokerowe. Patrz również *idealny dzień, planowanie pokerowe, szacunek względny*.

## R

**rachunek ekonomiczny uwzględniający innowacje** (ang. *innovation accounting*) — system księgowy (pomiary) używający czynnych środków do oceny szybkości zdobywania wiedzy jako kluczowego wyznacznika postępu umożliwiającego osiągnięcie rezultatów o wartości biznesowej [Ries, 2011].

**radiator informacji** (ang. *information radiator*) — wyświetlacz prezentujący przechodniom najświezsze, odpowiednio szczegółowe i ważne informacje w łatwy, niewymagający interpretacji sposób.

**refaktoryzacja** (ang. *refactoring*) — technika polegająca na poprawianiu ciała istniejącego kodu przez usprawnienie (uproszczenie) jego wewnętrznej struktury (projektu), ale z zachowaniem istniejącego działania. Refaktoryzacja jest jedną z podstawowych technik służących do zarządzania długiem technicznym. Patrz również *dług techniczny, działania techniczne*.

**reguła** (ang. *rule*) — powszechna praktyka lub ogólnie uznana metoda postępowania w określonej sytuacji. Reguła może zostać naruszona, jeśli okoliczności wskazują na konieczność podjęcia innego sposobu postępowania. Reguły są częścią środowiska Scrum. Patrz również *esencja Scruma, środowisko Scruma*.

**reguła skauta** (ang. *Boy Scout rule*) — 1. zawsze pozostawiaj obóz w stanie czystszy niż ten, który zastałeś. Jeśli znajdziesz rozrzucone na ziemi śmieci, posprzątaj je, nie zastanawiając się, kto je tam pozostawił. 2. za każdym razem, kiedy pracujesz w obszarze kodu, pozostaw źródła w stanie odróżnione lepszym, nie gorszym, od tego, który zastałeś. Patrz również *dług techniczny*.

**rejestr portfela produktów** (ang. *portfolio backlog*) — rejestr, na który składają się produkty, programy, projekty i także eposy wysokiego rzędu. Patrz również *planowanie portfela*.

**rejestr projektu** (ang. *product backlog*) — spis czekających na wykonanie historyjek z rejestru produktu z przypisanymi priorytetami. Patrz również *historyjka z rejestru produktu*.

**rejestr spostrzeżeń** (ang. *insight backlog*) — posortowana według priorytetów lista utworzonych wcześniej spostrzeżeń lub usprawnień procesu, co do których nie podjęto jeszcze żadnych działań. Rejestr spostrzeżeń jest tworzony i używany podczas retrospekcji sprintów. Patrz również *retrospekcja sprintu*.

**rejestr sprintu** (ang. *sprint backlog*) — 1. artefakt wytworzony podczas planowania sprintu i aktualizowany na bieżąco podczas jego wykonania. Ma on za zadanie pomóc samoorganizującemu się zespołowi w lepszym planowaniu i zarządzaniu pracą, która musi zostać zrealizowana, aby osiągnąć cel sprintu. 2. lista historyjek z rejestru produktu „wciągniętych” do sprintu oraz plan ich realizacji — często mający postać zadań z przypisanym do nich czasem trwania w godzinach. Patrz również *idealna godzina, planowanie sprintu, zadanie*.

**retrospekcja** (ang. *retrospective*) — patrz *retrospekcja sprintu*.

**retrospekcja sprintu** (ang. *sprint retrospective*) — aktywność polegająca na inspekcji i adaptacji przeprowadzana pod koniec każdego sprintu. Retrospekcja sprintu jest powtarzającą się okazją dla zespołu scrumowego do dokonania przeglądu swojego procesu (podejścia do realizacji zasad Scruma) i wskazania możliwości poprawy. Patrz również *inspekcja i adaptacja, retrospekcja sprintu*.

**rola** (ang. *role*) — spójny zbiór obowiązków, które mogą zostać zrealizowane przez jedną lub więcej osób. Trzema rolami w Scrumie są: właściciel produktu, mistrz młyna i zespół deweloperski. Patrz również *praktyka, zasada*.

**rola użytkownika** (ang. *user role*) — 1. nazwa dla klasy użytkowników produktu. 2. jeden z kluczowych elementów historyjki użytkownika definiujący odbiorcę wartości dostarczonej przez tę historyjkę. Patrz również *historyjka użytkownika*.

**rozmiar zapotrzebowania** (ang. *batch size*) — moc zbioru elementów przeznaczonych do przetworzenia w pewnym kolejnym kroku. Patrz również *praca cząstkowa*.

**rozwiązań** (ang. *solution*) — produkt lub usługa będące wynikiem wysiłku deweloperskiego.

**ryzyko** (ang. *risk*) — 1. ryzyko zaistnienia niechcianych okoliczności w trakcie zdarzenia — mierzone zarówno prawdopodobieństwem zdarzenia, jak i wagą konsekwencji. 2. jakakolwiek niepewność mogąca mieć negatywny wpływ na daną aktywność. Patrz również *niepewność*.

## S

**samoorganizacja** (ang. *self-organization*) — 1. rozwijająca się w sposób wstępujący właściwość systemu adaptacyjnego, zgodnie z którą organizacja systemu rozbudowuje się z czasem w odpowiedzi na interakcję ze środowiskiem. 2. cecha zespołu deweloperskiego organizującego się w miarę upływu czasu nadzoru menedżerskiego. 3. odzwierciedlenie filozofii zarządzania, zgodnie z którą decyzje operacyjne są oddelegowywane do najniższego możliwego szczebla, czyli do ludzi posiadających najbardziej precyzyjną wiedzę odnośnie do konsekwencji i praktycznych skutków tych decyzji. Patrz również *złożony system adaptacyjny, emergencja*.

**schemat działań dla produktu** (ang. *product roadmap*) — opis wewnętrznych sposobów budowania i wdrażania produktu na przestrzeni czasu wraz ze wskazaniem istotnych czynników mających wpływ na każdą indywidualną wersję dystrybucyjną. Przydatny podczas tworzenia produktu, który będzie miał więcej niż jedną wersję produkcyjną. Patrz również *tworzenie wizji produktu*.

**Scrum** (ang. *młyn*) — termin zapożyczony z rugby. 1. proste środowisko agile służące do zarządzania skomplikowanym procesem rozwijania produktu lub usługi. 2. iteracyjna i przyrostowa metoda wytwarzania produktów i zarządzania pracą. Patrz również *agile, środowisko Scrum*.

**scrum scrumów** (ang. *scrum of scrums — SoS*) — metoda koordynowania pracy wielu zespołów scrumowych. Osoby oddelegowane z każdego zespołu scrumowego (minimum po jednym członku z każdego zespołu) zbierają się, aby przedyskutować wzajemne zależności pomiędzy zespołami. Patrz również *pociąg wersji dystrybucyjnych*.

**Scrummerfall** — patrz *proces kaskadowy w Scrumie*.

**sejsmograf emocji** (ang. *seismograf emocji*) — graficzna reprezentacja emocjonalnych wzlotów i upadków wśród członków zespołu w trakcie trwania sprintu. Technika często używana podczas retrospekcji sprintu. Patrz również *retrospekcja sprintu*.

**SoS** — patrz *scrum scrumów*.

**specyfikacja przez przykład** (ang. *specification by example*) — patrz *produkcja sterowana testami akceptacyjnymi*.

**sprint** — iteracja o krótkim, ustalonym czasie trwania. Ograniczenie czasu wynosi typowo od jednego tygodnia do jednego miesiąca. W tym czasie zespół scrumowy skupia się na wyprodukowaniu potencjalnie zdatnego do wdrożenia przyrostowego fragmentu produktu, który spełnia przyjęte przez zespół kryteria ukończenia. Patrz również *definicja ukończenia, iteracja, potencjalnie zdatny do wdrożenia przyrost produktu*.

**stopniowe udoskonalanie** (ang. *progressive refinement*) — dzielenie w odpowiednim czasie dużych, pozbawionych szczegółów historyjek z rejestru produktu na mniejsze, bardziej sprecyzowane.

**strategiczny dług techniczny** (ang. *strategic technical debt*) — forma dłużu technicznego używana jako narzędzie wspomagające organizacje w lepszym wyznaczaniu szacunków ekonomicznych podczas podejmowania ważnych, często istotnych ze względu na czas decyzji. Patrz również *dług techniczny*.

**synchronizacja** (ang. *synchronization*) — wymuszanie wielu różnych zdarzeń w tym samym czasie. Operacja stosowana często w celu zapewnienia skoordynowanego współdziałania ze sobą wielu zespołów scrumowych przez rozpoczęwanie i kończenie ich sprintów w tych samych dniach. Patrz również *takt*.

**szacunek względny** (ang. *relative size measure*) — wyrażenie całkowitego rozmiaru elementu bez uwzględnienia wszystkich kluczowych czynników, a jedynie rozmiaru tego elementu w porównaniu z innymi elementami. Na przykład element o rozmiarze 2 jest o połowę mniejszy od elementu o rozmiarze 4 — nie mamy jednak pojęcia, jak duże są elementy o rozmiarze 2 i 4 w sensie absolutnym. Patrz również *idealny dzień, punkt historyjowy*.

**szef właścicieli produktu** (ang. *chief product owner*) — nadzorujący właściciel produktu wewnętrz zespołu właścicieli produktu związanych z dużym przedsięwzięciem produkcyjnym. Patrz również *właściciel produktu*.

**szykba informacja zwrotna** (ang. *fast feedback*) — zasada mówiąca, iż informacja zwrotna uzyskana dzisiaj jest o wiele lepsza od takiej samej informacji, która nadziejnie jutro, ponieważ być może pozwoli ona na poprawienie problemu, zanim ten nawarstwi się i tym samym zapobiegnie podążeniu w kierunku niepożądanym z punktu widzenia ekonomii (przegrywaj szybko). Patrz również *szykba porażka*.

**szykba porażka** (ang. *fail fast*) — strategia polegająca na przeprowadzeniu szybkiej próby, odebraniu informacji zwrotnej, przeanalizowaniu jej i dostosowaniu do nowej sytuacji. W przypadku istnienia wysokiego poziomu niepewności często mniej kosztowne jest rozpoczęcie pracy nad produktem, przekonanie się, czy podjęliśmy dobrą decyzję, a w sytuacji, gdy odpowiedź brzmi negatywnie, szybkie przerwanie dalszych działań, zanim wydane zostaną kolejne pieniądze. Patrz również *szykba informacja zwrotna, inspekcja i adaptacja, zwrot*.

**środowiska** (ang. *framework*) — patrz *środowisko Scrum*.

**środowisko Scrum** (ang. *Scrum framework*) — zbiór wartości, zasad i reguł formułujących podstawy działania w oparciu o Scrum. Patrz również *Scrum*.

**świnie** (ang. *pigs*) — metafora używana przez niektóre zespoły scrumowe do opisania ludzi poświęcających się w celu osiągnięcia wspólnego celu (odpowiedzialnych za wynik). Członkowie zespołu scrumowego uważani są za świnie. Patrz również *zespoł scrumowy*. Przeciwne do kurczaków.

## T

**tablica zadań** (ang. *task board*) — sposób przekazywania informacji podczas wykonania sprintu. Jej zadaniem jest komunikowanie wszystkim postępu i przepływu pracy na poziomie zadań wewnętrz sprintu. Patrz również *radiator informacji, zadanie*.

**takt** (ang. *cadence*) — regularny, przewidywalny rytm lub puls. Sprinty o jednakowej długości taktują odpowiednio wysiłek programistyczny. Patrz również *synchronizacja*.

**TDD** — patrz również *programowanie sterowane testami*.

**technika** (ang. *technique*) — zdefiniowana procedura służąca do wykonania pewnych lub wszystkich aktywności lub wsparcia pewnego podejścia. Patrz również *aktywność, podejście*.

**temat** (ang. *theme*) — zbiór powiązanych ze sobą historyjek użytkownika. Temat pozwala w wygodny sposób wskazać, iż dane historyjki mają coś wspólnego ze sobą, na przykład dotyczą tego samego obszaru funkcjonalności. Patrz również *epos, historyjka użytkownika*.

**tempo podtrzymywane** (ang. *sustainable pace*) — odpowiednio mocne tempo pracy zespołu pozwalające na sprawne wytwarzanie wartości biznesowej przez długi czas i jednocześnie zapobiegające wypaleniu zespołu.

**test akceptacyjny** (ang. *acceptance test*) — 1. test mający na celu sprawdzenie, czy zostały spełnione kryteria akceptacji. 2. test określający wartość biznesową, jaką musi dostarczyć każda historyjka w rejestrze. Jego przedmiotem mogą być wymagania funkcjonalne lub niefunkcjonalne — opisujące na przykład wydajność lub niezawodność. Test akceptacyjny wspomaga proces produkcyjny [Crispin i Gregory, 2009]. 3. formalny test przeprowadzany w odniesieniu do potrzeb i wymagań użytkownika oraz procesu biznesowego mający na celu sprawdzenie, czy system spełnia kryteria akceptacji, a także umożliwiający użytkownikowi, klientom lub innym uprawnionym osobom zaakceptowanie lub odrzucenie systemu [IEEE 610].

**tradycyjny proces deweloperski** (ang. *traditional development process*) — patrz *proces sterowany planem*.

**tworzenie wizji produktu** (ang. *envisioning*) — aktywność mająca na celu uchwycenie sedna potencjalnego produktu i stworzenie przybliżonego planu stworzenia tego produktu. Tworzenie wizji produktu zaczyna się od stworzenia wizji, następnie rejestru produktu na bardzo ogólnym poziomie, a często również schematu działań dla produktu. Równoznaczne z *planowaniem produktu*. Patrz również *schema działania dla produktu*.

## U

**umiejętności typu „T”** (ang. *T-shaped skills*) — metafora opisująca osoby o ogromnych przekrójowych umiejętnościach z wyspecjalizowanej dziedziny (może to być na przykład projektowanie interfejsu użytkownika), a także bogatych, chociaż niekoniecznie rozległych umiejętnościach z innych istotnych dziedzin (takich jak dokumentacja i testowanie). Członkowie zespołu z umiejętnościami typu T pozwalają na lepsze działanie w roju. Patrz również *działanie w roju*.

**utrudnienie** (ang. *impediment*) — przeszkoda lub obstrukcja stojąca na drodze do realizacji pewnego zadania. Termin często stosowany do opisania problemu lub blokady uniemożliwiającej zespołowi lub organizacji realizowanie zasad Scruma w sposób efektywny.

## W

**w samą porę** (ang. *JIT — just in time*) — charakterystyka procesu, w którym zasoby lub aktywności związane z przepływem pracy stają się dostępne dokładnie w chwili, kiedy są potrzebne.

**warsztaty pisania historyjek użytkownika** (ang. *user-story-writing workshop*) — warsztaty trwające od kilku godzin do kilku dni, w których trakcie różnorodne zespoły uczestników prowadzą burze mózgów na temat oczekiwanych wartości biznesowych i tworzą szkielety historyjek użytkownika opisujących spodziewane produkty lub usługi. Patrz również *historyjka użytkownika*.

**wartości** (ang. *values*) — 1. rzeczy, które uznajemy za drogie sercu lub cenne. 2. fundament wspólnego porozumienia w działaniu pomiędzy członkami zespołu. Wśród wartości zespołów scrumowych znajdują się: uczciwość, otwartość, odwaga, szacunek, skupienie, zaufanie, temperament i współpraca.

**wartość dla interesariusza** (ang. *stakeholder value*) — wartość, jaką rozwiązywanie dostarcza interesariuszom. Określenie stosowane zamiennie z wartością dla klienta. Patrz również *interesariusz*.

**warunki zadowolenia** (ang. *conditions of satisfaction*) — warunki, które muszą zostać spełnione, aby właściciel produktu mógł uznać, że rejestr produktu został wykonany. Warunki zadowolenia to inaczej kryteria akceptacji, które wyjaśniają oczekiwane zachowanie. Patrz również *kryteria akceptacji*.

**WaterScrum** — nałożenie kaskadowego stylu produkcji na środowisko Scrum. Przykładem takiego działania byłoby wykonanie sprintu analiz, następnie sprintu projektowania, następnie sprintu kodowania i wreszcie sprintu testowania. Równoznaczne z *procesem kaskadowym w Scrumie*.

**wersja dystrybucyjna** (ang. *release*) — 1. zestaw cech, które połączone razem w jeden pakiet będą stanowiły spójną całość dla klientów lub użytkowników. 2. wersja produktu promowana jako zdatna do użycia lub wdrożenia. Wersje wdrożeniowe wyznaczają rytm dostarczania wartości biznesowej i powinny być w zgodzie ze zdefiniowanymi cyklami biznesowymi.

**wiedza potwierdzona** (ang. *validated learning*) — termin zaproponowany przez Riesa [2011] do opisania postępu osiąganego w chwili, kiedy pewne założenia zostały udowodnione lub odrzucone na skutek poddania każdego z nich jednemu lub większej liczbie testów walidacyjnych. Przeciwne do *założenia*.

**wiedza ukryta** (ang. *tacit knowledge*) — wiedza niema, nigdzie nie spisana (chodzi między innymi o rozumienie, intuicję, przeczucia), której wyrażenie przy użyciu języka formalnego jest trudne, ale nie niemożliwe. Przeciwieństwo wiedzy jawnej lub formalnej. Czasami określana mianem „know-how”.

**WIP** — patrz *praca cząstkowa*.

**wizja produktu** (ang. *product vision*) — krótkie stwierdzenie stanu cech produktu, jaki osiągnięty będzie po zakończeniu prac deweloperskich i przeprowadzeniu wdrożenia. Dobra wizja powinna być łatwa do wyrażenia i wskazywać spójny kierunek działania dla osób poproszonych o jej realizację. Patrz również *tworzenie wizji produktu*.

**właściciel produktu** (ang. *product owner*) — centralny ośrodek upoważniony do kierowania produktem. Jedna z trzech ról zespołu scrumowego — pojedynczy głos środowiska interesariuszy skierowany do zespołu scrumowego. Właściciel produktu definiuje, co należy robić oraz w jakiej kolejności. Patrz również *zespoł scrumowy*.

**wodospad** (ang. *waterfall*) — termin odnoszący się do graficznej reprezentacji procesu produkcji, w którym kolejno następujące po sobie fazy pracy są przedstawiane w formie wykresu biegnącego stopniowo w dół podobnie do kaskad wodospadu. Patrz również *proces sterowany planem*.

**WSJF** — patrz *wyważona najkrótsza praca w pierwszej kolejności*.

**wykres rozpalania** (ang. *burnup chart*) — wykres pokazujący postęp prac dążący do *linii celu* wyznaczonej przez wartość na osi pionowej. W miarę końca pracy prac wraz z upływem *czasu* (wyrażonego na osi poziomej) krzywa postępu wzrosi się do góry („rozpalą się”) i zbliża do linii celu. Możemy pokazać spodziewane wyniki na tych wykresach przez wyznaczenie linii trendu i w ten sposób przewidzieć teoretyczny moment zakończenia prac. Przeciwne do *wykresu spalania*.

**wykres spalania** (ang. *burndown chart*) — wykres pokazujący na pionowej osi *ilość pracy* (wyrażoną w godzinach lub jednostkach rejestru produktu) pozostałą do wykonania w *danym czasie*, przedstawionym na poziomej osi wykresu. Wraz z upływem czasu ilość potrzebnej pracy powinna maleć, zatem trend wykresu powinien dążyć do punktu, w którym nie ma nic do zrobienia. Możemy wskazać to miejsce na każdym wykresie spalania poprzez wyznaczenie linii trendu i w ten sposób dowiedzieć się, kiedy teoretycznie praca zostanie ukończona. Przeciwne do *wykresu rozpalania*.

**wymaganie niefunkcjonalne** (ang. *nonfunctional requirement*) — 1. wymóg niezwiązany bezpośrednio z funkcjonalnością, ale nadal istotny — może to być niezawodność, wydajność, użyteczność, utrzymywalność i przenośność elementu z rejestru produktu kwalifikująca go do pełnej akceptacji przez interesariuszy. 2. każdy wymóg niefunkcjonalny jest potencjalnym kandydatem do włączenia w kryteria *definicji ukończenia*.

**wysiłek deweloperski produktu** (ang. *product development effort*) — pełny zakres prac potrzebnych w celu stworzenia lub ulepszenia produktu lub usługi. Przeciwne do *projektu*.

**wytwarzanie w trybie wszystko naraz** (ang. *all-at-once product development*) — wykonywanie wszystkich rodzajów prac (na przykład analiz, projektowania, kodowania, integrowania i testowania) wbrew wszelkim zasadom w ramach jednej iteracji.

**wyważona najkrótsza praca w pierwszej kolejności** (ang. *weighted shortest job first — WSJF*) — optymalny ekonomicznie algorytm planowania pracy w środowisku, w którym zarówno koszt opóźnienia, jak i czas trwania są zmienne. Patrz również *koszt opóźnienia*.

## X

**XP** — patrz *programowanie ekstremalne*.

## Z

**zadanie** (ang. *task*) — praca inżynierska, którą zespół wykonuje w celu zrealizowania historyjki z rejestru produktu. Zadania w większości przypadków są krótkie i reprezentują nie więcej niż kilka godzin do jednego dnia pracy.

**założenie** (ang. *assumption*) — przypuszczenie lub przekonanie, co do których przyjmuje się, iż są prawdziwe, zgodne z rzeczywistością lub pewne, nawet jeśli nie ma potwierdzonej wiedzy pozwalającej na takie przekonanie. Przeciwne do *wiedzy potwierdzonej*.

**zasada** (ang. *principle*) — fundamentalna prawda lub pogląd będący podstawą naszego podejścia do produkcji produktu. Przykładem zasady Scruma jest regularne prezentowanie postępów w pracy. Patrz również *praktyka, wartości*.

**zasada najmniejszego zaskoczenia** (ang. *principle of least astonishment*) — działanie lub wytwarzanie produktów w sposób najmniej zaskakujący dla najbliższych współpracowników.

**zespoł** (ang. *team*) — mała liczba współpracujących ze sobą osób o zróżnicowanych, wielofunkcyjnych umiejętnościach, mających za zadanie realizację wspólnego celu. Członkowie zespołu ufają sobie nawzajem i pracują razem, aby osiągnąć cel, dzieląc się między sobą odpowiedzialnością za wynik. Przeciwne do *grupy*.

**zespoł budujący cechy** (ang. *feature team*) — wielofunkcyjny zespół pracujący z różnymi komponentami, który jest w stanie pobrać z rejestru produktu cechy przeznaczone dla użytkownika końcowego i zrealizować je. Patrz również *zespoł wielofunkcyjny*. Przeciwne do *zespołu budującego komponenty*.

**zespoł budujący komponenty** (ang. *component team*) — zespół zajmujący się tworzeniem jednej lub kilku części składowych większego produktu przeznaczonego dla klienta. Zespoły tego typu wytwarzają aktywa lub komponenty, które używane są następnie przez inne zespoły budujące gotowe rozwiązanie mające wartość dla klienta. Przeciwne do *zespołu budującego cechy produktu*.

**zespoł deweloperski** (ang. *development team*) — samoorganizujący się, wielofunkcyjny zespół ludzi, którzy ponoszą wspólną odpowiedzialność za wszelkie prace niezbędne do wyprodukowania działających i sprawdzonych aktywów.

**zespoł scrumowy** (ang. *Scrum team*) — zespół składający się z właściciela produktu, mistrza młyna i zespołu deweloperskiego dokonującego faktycznego wysiłku produkcyjnego. Patrz również *zespoł deweloperski, właściciel produktu, mistrz młyna*.

**zespoł wielofunkcyjny** (ang. *cross-functional team*) — zespół składający się z osób posiadających wszelkie możliwe umiejętności (takie jak projektowanie interfejsu użytkownika, tworzenie kodu, testowanie) i specjalizacje niezbędne do wykonania pracy, której ukończenie wymaga znajomości więcej niż jednej dziedziny.

**złożony system adaptacyjny** (ang. *complex adaptive system*) — system, w którym wiele jednostek współdziała ze sobą na różny sposób. Sposób współdziałania jest nadzorowany przez proste, lokalne reguły działające na zasadzie nieustającej pętli zwrotnej. Przykładami mogą być giełda papierów wartościowych, mózg, kolonie, a także zespoły scrumowe.

**zmarnowanie innowacji** (ang. *innovation waste*) — utracona możliwość stworzenia innowacyjnego rozwiązania. Sytuacja często występująca, kiedy historyjka w rejestrze produktu narzuca konkretne rozwiązanie.

**zmienna** (ang. *variability*) — rozkład lub dyspersja danych reprezentujących kolejne wyniki. W przemyśle wytwórczym zmienna oznacza zawsze marnotrawstwo. W procesie wytwarzania oprogramowania pewna zmienność jest niezbędna w celu osiągnięcia rozwiązań innowacyjnych. Patrz również *marnotrawstwo*.

**znany dług techniczny** (ang. *known technical debt*) — określenie statusu dłużu technicznego, który jest znany zespołowi deweloperskiemu i został ujawniony w celu przeanalizowania w przyszłości. Przeciwne do *nieoczekiwanej dłużu technicznego, celowego dłużu technicznego*. Patrz również *dług techniczny*.

**zwrot** (ang. *pivot*) — 1. zmiana kierunku, ale z zachowaniem podstawy bazującej na zdobytej wiedzy. 2. zmiana kierunku działania mająca na celu przetestowanie pewnej nowej hipotezy odnośnie do produktu, strategii i maszyny wzrostu [Ries, 2011].

**zyski z cyklu życia** (ang. *lifecycle profits*) — 1. całkowity potencjał zysku z produktu przez cały cykl jego życia. 2. w przypadku planowania portfela produktów całkowity zysk z tego portfela (w przeciwieństwie do zysku z pojedynczego produktu).



# BIBLIOGRAFIA

---

Adkins Lyssa, *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*, Addison-Wesley Professional, 2010.

Anderson David J., *Kanban*, Blue Hole Press, 2010.

Appelo Jurgen, *Management 3.0: Leading Agile Developers, Developing Agile Leaders*, Addison-Wesley Professional, 2011.

Beck Kent, Beedle Mike, van Bennekum Arie, Cockburn Alistair, Cunningham Ward, Fowler Martin, Grenning James, Highsmith Jim, Hunt Andrew, Jeffries Ron, Kern Jon, Marick Brian, Martin Robert C., Mellor Steve, Schwaber Ken, Sutherland Jeff, Thomas Dave, *Manifest Zwinnego Tworzenia Oprogramowania*, [www.agilemanifesto.org/iso/pl/](http://www.agilemanifesto.org/iso/pl/), 2001.

Beck Kent, Andres Cynthia, *Extreme Programming Explained*, 2nd ed., Addison-Wesley Professional, 2004.

Boehm Barry W., *Software Engineering Economics*, Prentice Hall, 1981.

Brooks Frederick P., *Mityczny osobomiesiąc*, Wydawnictwa Naukowo-Techniczne, 2000

Cohn Mike, *User Stories Applied: For Agile Software Development*, Addison-Wesley Professional, 2004.

Cohn Mike, *Agile Estimating and Planning*, Addison-Wesley Professional, 2006.

Cohn Mike, *Succeeding with Agile*, Addison-Wesley Professional, 2009.

Cohn Mike, prezentacja w ramach konferencji Agile 2010, 2010.

Cook Daniel, „The Laws of Productivity: 8 productivity experiments you don't need to repeat”, 2008; prezentacja znaleziona pod adresem <http://www.lostgarden.com/2008/09/rules-of-productivity-presentation.html>.

Crispin Lisa, Gregory Janet, *Agile Testing: A Practical Guide for Testers and Agile Teams*, Addison-Wesley Professional, 2009.

Cunningham Ward, „The WyCash Portfolio Management System”, raport OOPSLA ’92, Object-Oriented Programming Systems, Languages and Applications, Vancouver, 18 – 22 października 1992.

Denne Mark, Cleland-Huang Jane, *Software by Numbers: Low-Risk, High-Return Development*, Prentice Hall, 2003.

Derby Esther, Larsen Diana, *Agile Retrospectives: Making Good Teams Great*, Pragmatic Bookshelf, 2006.

Fowler Martin, „Technical Debt Quadrant”, wpis na stronie <http://martinfowler.com/bliki/TechnicalDebtQuadrant.html>, 2009.

Fowler Martin, Beck Kent, Brant John, Opdyke William, Roberts Don, *Refaktoryzacja. Ulepszanie struktury istniejącego kodu*, Helion, 2011.

Goldberg Adele, Rubin Kenneth S., *Succeeding with Objects: Decision Frameworks for Project Management*, Addison-Wesley Professional, 1995.

Grenning James, „Planning Poker”, 2002; [www.objectmentor.com/resources/articles/PlanningPoker.zip](http://www.objectmentor.com/resources/articles/PlanningPoker.zip).

Highsmith Jim, *Agile Project Management: Creating Innovative Products*, 2nd ed., Addison-Wesley Professional, 2009.

Hohmann Luke, *Beyond Software Architecture*, Addison-Wesley Professional, 2003.

IEEE, IEEE Std 610.12-1990 (rozszerzenie standardu IEEE Std 792-1983), IEEE Standards Board of the Institute of Electrical and Electronics Engineers, New York, 28 września 1990.

Jeffries Ron, „Essential XP: Card, Conversation, Confirmation”, 2001; <http://xprogramming.com/articles/expcardconversationconfirmation>.

Katz Ralph, *The Effects of Group Longevity on Project Communication and Performance*, „Administrative Science Quarterly”, 1982, 27, s. 81 – 104.

Kennedy John Fitzgerald, *Special Message to the Congress on Urgent National Needs*, 22 maja 1961.

Kerth Norm, *Project Retrospectives: A Handbook for Team Reviews*, Dorset House, 2001.

Larman Craig, Vodde Bas, „Lean Primer”, 2009; artykuł dostępny na stronie [www.leanprimer.com/downloads/lean\\_primer.pdf](http://www.leanprimer.com/downloads/lean_primer.pdf).

Laufer Alexander, *Simultaneous Management: Managing Projects in a Dynamic Environment*, American Management Association, 1996.

Leffingwell Dean, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, Addison-Wesley Professional, 2011.

McConnell Steve, „Technical Debt”, wpis na blogu, <http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx>, 2007

Mar Kane, „Technical Debt and the Death of Design: Part 1”, wpis na blogu, <http://kanemar.com/2006/07/23/technical-debt-and-the-death-of-design-part-1>, 2006.

Martin Robert C., *Czysty kod. Podręcznik dobrego programisty*, Helion, 2010.

Page Scott, *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*, Princeton University Press, 2007.

Patton Jeff, przykład wdrażania przyrostowego, kontakt osobisty, 2008.

Patton Jeff, *Telling Better User Stories: Mapping the Path to Success*, „Better Software”, listopad – grudzień 2009, s. 24 – 29.

Pelrine Joseph, „Is Software Development Complex”, wpis na blogu, <http://cognitive-edge.com/blog/entry/4597/is-software-development-complex>, 2011.

Pichler Roman, *Agile Product Management with Scrum: Creating Products That Customers Love*, Addison-Wesley Professional, 2010.

PMI, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 4th ed., Project Management Institute, Inc., 2008.

Poppendieck Mary, Poppendieck Tom, *Lean Software Development: An Agile Toolkit*, Addison-Wesley Professional, 2003.

Putnam Doug, „Team Size Can Be the Key to a Successful Project”, artykuł w ramach cyklu Process Improvement Series QSM, 1996; [www.qsm.com/process\\_01.html](http://www.qsm.com/process_01.html).

Putnam Lawrence H., Myers Ware, *Familiar Metrics Management: Small Is Beautiful — Once Again, „IT Metrics Strategies”*, IV, 8, s. 12 – 16, Cutter Information Corp., 1998.

Reinertsen Donald G., „Types of Processes”, wpis na blogu, [www.netobjectives.com/blogs/Types-of-Processes](http://www.netobjectives.com/blogs/Types-of-Processes), 2009 (a).

Reinertsen Donald G., *The Principles of Product Development Flow: Second Generation Lean Product Development*, Celeritas Publishing, 2009 (b).

Ries Eric, *Metoda Lean Startup. Wykorzystaj innowacyjne narzędzia i stwórz firmę, która zdobędzie rynek*, Helion, 2012.

Schwaber Ken, „Scrum Development Process”, w: J. Sutherland et al., red., *OOPSLA Business Object Design and Implementation Workshop*, Springer, 1995.

Schwaber Ken, *Sprawne zarządzanie projektami metodą Scrum*, Promise, 2005.

Schwaber Ken, Beedle Mike, *Agile Software Development with Scrum*, Prentice Hall, 2001.

Schwaber Ken, Sutherland Jeff, *Przewodnik po Scrumie*, 2011; dostępny na stronie <http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20PL.pdf>.

SEI, „CMMI for Development, Version 1.3”, Software Engineering Institute, Carnegie Mellon University, 2010; artykuł dostępny na stronie [www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm](http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm).

SEI, Second International Workshop on Managing Technical Debt, warsztaty towarzyszące ICSE 2011, Waikiki, Honolulu, 23 maja 2011; materiały dostępne na stronie [www.sei.cmu.edu/community/td2011](http://www.sei.cmu.edu/community/td2011).

Smith Preston G., Reinertsen Donald G., *Developing Products in Half the Time: New Rules, New Tools*, Van Nostrand Reinhold, 1998.

Snowden David J., Boone Mary E., *A Leader's Framework for Decision Making*, „Harvard Business Review”, listopad 2007.

Staats Bradley R., *Unpacking Team Familiarity: The Effects of Geographic Location and Hierarchical Role*, University of North Carolina, Chapel Hill, 2011.

Takeuchi Hirotaka, Nonaka Ikujiro, *The New New Product Development Game*, „Harvard Business Review”, styczeń 1986, s. 137 – 146.

Tuckman Bruce W., *Developmental Sequence in Small Groups*, „Psychological Bulletin”, 1965, 63, s. 384 – 399; artykuł przedrukowany w: *Group Facilitation: A Research and Applications Journal*, 3, wiosna 2001.

VersionOne, „The State of Agile Development: Sixth Annual Survey”, 2011; artykuł dostępny jako PDF w bibliotece Library of White Papers na stronie [www.versionone.com](http://www.versionone.com).

Wake William C., „INVEST in Good Stories, and SMART Tasks”, 2003; [www.xp123.com](http://www.xp123.com).

Wheelwright Steven C., Clark Kim B., *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality*, The Free Press, 1992.

Wiseman John „Lofty”, *SAS Survival Guide: For Any Climate, in Any Situation*, wyd. poprawione, Collins Reference, 2010.

# SKOROWIDZ

## A

- adaptacja, 67, 68
- adaptowanie środowiska, 259
- aktywności Scruma, 48
- aktywność
  - planowania
  - portfela, 293
  - sprintu, 357
    - wersji dystrybucyjnej, 332
  - przeglądu sprintu, 387
  - retrospekcji sprintu, 399
  - wykonania sprintu, 368
- analiza marginalna, 307
- artefakty Scruma, 48
- autoryzacja, 256

## B

- budowa zespołów scrumowych, 239
- budowanie w sposób
  - iteracyjny, 65
  - przyrostowy, 65
- burza mózgów, 125

## C

- całość przed kolejnym krokiem, 79
- cechy
  - mistrza młyna, 216
  - obowiązkowe, 317, 323, 338
  - właściciela produktu, 199, 200
  - zespołu deweloperskiego, 226

- cel sprintu, 99
  - doprecyzowanie, 99
  - zmiana, 100
- cel wersji dystrybucyjnej, 318
- ceremonie, 87
- charakterystyka sprintu, 52
- ciążła dystrybucja, 330
- codzienne działania scrumowe, 55, 374
- cykl życia produktu, 84
- czarterowanie projektu, 321
- czas trwania produkcji, 148

## D

- DEEP, 131
- definicja
  - gotowości, 138, 363
  - ukończenia, 104, 106, 178, 385
- definiowanie kryteriów, 197
- demonstrowanie postępów, 93
- dług techniczny, 23, 167
  - celowy, 183
  - konsekwencje, 169
  - natywny, 168
  - nieunikniony, 168
  - obsługa, 183
  - odkryty, 183
  - przyczyny, 172
  - spłacanie, 186
  - strategiczny, 168
  - ujawnianie, 180
  - zarządzanie, 176
  - znany, 183

dokładność, 153  
 domena  
     chaosu, 41  
     prostoty, 40  
     skomplikowana, 40  
     złożona, 40  
 doprecyzowanie celu sprintu, 99  
 dostarczanie wartości, 85  
 działanie  
     w kopcu, 370  
     w roju, 371

**E**

ekonomia  
     dług technicznego, 178  
     na poziomie rejestru produktu, 196  
     na poziomie sprintów, 196  
     na poziomie wersji dystrybucyjnej, 195  
     pojedynczej wersji dystrybucyjnej, 279  
     wielu wersji dystrybucyjnych, 279  
 eksploracja, 71  
 elementy  
     rejestru produktu, 51, 129, 131  
     skompletowane, 107  
 empiryczna kontrola nad procesem, 67  
 epos, 116

**F**

filtr  
     ekonomiczny, 203, 299, 322  
     strategiczny, 310  
 finansowanie  
     prowizoryczne, 325  
     przyrostowe, 325  
 funkcjonalności, *Patrz rejestr produktu*

**G**

głosowanie kropkowe, 405

**H**

harmonogram  
     dla portfela, 293, 295  
     przeglądu sprintu, 384  
     wypuszczania wersji dystrybucyjnych, 329  
 hierarchia abstrakcji historyjek, 117

hierarchiczne rejesty produkłów, 144  
 historyjki  
     pozyskiwania wiedzy, 123  
     rejestru produktu, 110  
     użytkownika, 111, 113  
         karta, 113  
     kryteria INVEST, 118  
     mały rozmiar, 121  
     negocjowalność, 118  
     niezależność, 118  
     ocena jakości, 131  
     ocenialność, 121  
     potwierdzenie, 115  
     poziom abstrakcji, 116  
     rozmowa, 114  
     testowalność, 122  
     wartościowość, 120

**I**

idealne dni, 52, 156  
 identyfikacja spostrzeżeń, 403  
 inflacja punktowa, 165  
 informacje potwierdzające, 115  
 inspekcja, 67  
 integracja modułów i testowania, 77  
 integralność zespołu, 258  
 INVEST, 118  
 inwentarz, 323  
 iteracje, 34

**J**

jakość produktu, 86  
 jednostki oceny  
     idealne dni, 156  
     punkty historyjowe, 155

**K**

kalendarz sprintów, 341  
 kanban, 41  
 kandydat  
     na mistrza młyna, 219  
     na właściciela produktu, 204  
 karty, 113  
 karty do planowania pokerowego, 158  
 klastry współpracy, 266

komunikacja  
  przezroczysta, 233  
  szerokopasmowa, 232  
komunikowanie postępów, 348, 350  
kontrola empiryczna procesu, 67  
koordynacja zespołów, 244  
koszt  
  eksploracji, 71  
  opóźnienia, 83, 295  
  obliczenia, 296  
  profile, 297  
  pracy symultanicznej, 235  
  produkcji, 170  
  wielozadaniowości, 370  
  zmiany, 72, 74, 171  
krótkie sprints, 94  
kryteria  
  akceptacji, 107  
  DEEP, 131  
  INVEST, 118  
kształtowanie zespołów, 253

## L

limit pracy cząstkowej, 305  
linia zdarzeń, 401

## Ł

łączenie  
  ról, 209, 221  
  zespołów, 243

## M

mapa  
  drogowa produktu, 284, 286, 317  
historyjek, 125  
sprintów, 339  
menedżer projektu  
  czas, 263  
  integracja, 263  
  jakość, 263  
  kierowanie koordynacją, 267  
  ryzyko, 263  
  zakres, 263  
  zaopatrzenie, 263  
  zespół, 263

menedżer zasobów  
  definiowanie granic, 253  
  dostarczanie celu, 254  
  formowanie zespołów, 254  
  motywowanie ludzi, 257  
  obserwacja pomiarów, 261  
  promowanie wartości zwinności, 259  
  przewodzenie obszarowi funkcjonalności, 258  
  przystosowywanie grup wewnętrznych, 260  
  przystosowywanie partnerów, 260  
  raportowanie, 261  
  rozwijanie umiejętności, 257  
  upoważnianie zespołów, 256  
  usuwanie przeszkód organizacyjnych, 259  
  utrzymywanie integralności zespołu, 258  
  zarządzanie ekonimią, 261  
  zmiana składu zespołu, 255  
metoda  
  kanban, 42  
  skauta, 186  
mierzenie postępu, 85  
miniproces kaskadowy, 371  
mistrz młyna, 47, 213  
  agent zmiany, 215  
  autorytet w dziedzinie procesu, 215  
  chronienie przed zakłóceniami, 215  
  cierpliwość, 217  
  mistrz służby, 214  
  proaktywność, 217  
  przezroczystość, 217  
  trenowanie, 213  
  umiejętność stawiania pytań, 216  
  usuwanie przeszkód, 215  
  wiedza, 216  
  współpraca, 217  
model procesu scrumowego, 67  
moment podejmowania decyzji, 69

## N

nadawanie ocen, 147, 150  
najlepsze praktyki, 414  
negocjowanie wymagań, 109  
niepewność  
  klienta, 68  
  konca, 67  
  środków, 67  
niepotrzebne formalności, 87  
niezmiennałość celu, 98

**O**

obiektywne dane, 397  
 obliczanie  
     kosztów, 347  
     kosztu opóźnienia, 296  
     przedziału prędkości, 161  
 obniżanie kosztu zmiany, 73  
 obowiązki  
     menedżera projektu, 263  
     menedżera zasobów, 252  
     mistrza młyna, 213  
     właściciela produktu, 194  
     zespołu deweloperskiego, 224  
 obsługa dłużu technicznego, 183  
 obszary funkcjonalności, 251  
 ocena, 147, 152  
     elementów portfela, 149  
     elementów produktu, 149  
     elementów rejestru, 133, 148, 151  
     zadań, 150  
 ocenianie  
     absolutne, 154  
     względne, 154  
 oczekiwanie  
     na pracę, 81  
     na realizację, 81  
 ograniczanie  
     błędów, 95  
     pracy cząstkowej, 92  
 ograniczenia wersji dystrybucyjnej, 333–336  
 ograniczenie czasowe  
     demonstrowanie postępów, 93  
     motywowanie domykania prac, 94  
     poprawianie przewidywalności, 94  
     unikanie perfekcjonizmu, 93  
     wymuszanie priorytetów, 93  
     zmniejszanie pracy cząstkowej, 92  
 opcje planowania, 275  
 outsourcing, 207

**P**

pełnomocnik właściciela produktu, 210  
 perfekcjonizm, 93  
 persona, 125  
 pętla  
     informacji zwrotnej, 77  
     zdobywania wiedzy, 77

pielęgnacja  
     w procesach sekwencyjnych, 136  
     zespołów, 257  
     rejestru produktu, 134–138, 197, 225, 337  
 pisanie historyjek, 125  
 planowanie  
     codzienne, 287  
     długoterminowe, 329  
     hierarchiczne, 289  
     pokerowe  
         karty, 158  
         reguły, 159  
         skala ocen, 157  
     portfela, 283, 291–308  
         optymalizacja, 294  
     proces, 292  
     strategie, 294  
         strategie aktywności, 306  
         strategie napływu, 299  
         strategie odpływu, 304  
         tworzenie harmonogramu, 293  
         uczestnicy, 292  
     produktu, 202, 283, 309  
         finansowanie, 325  
         obszary wartości, 314, 315  
     proces, 312  
     przedstawianie wizji, 315  
     szybkie działanie, 324  
     szybkie zdobywanie wiedzy, 326  
     uczestnicy, 310  
         wytyczne ekonomiczne, 322  
     sprintu, 52, 99, 287, 355  
         dane wejściowe, 357  
         doprecyzowywanie celu, 365  
         dwuczęściowe, 358  
         jednoczęściowe, 359  
         nabieranie pewności, 364  
         określanie pojemności, 360  
         ustalanie zobowiązania, 365  
         wybieranie elementów rejestru, 363  
     uproszczone, 98  
     w samą porę, 275  
     według kamieni milowych, 330  
     wersji dystrybucyjnej, 285, 303, 329  
     komunikowanie postępów, 348, 350  
     lokalizacja cech obowiązkowych, 344  
     obliczanie kosztów, 347  
     ograniczenia, 333  
     określanie zakresu cech, 343

- proces, 331, 356  
rejestr produktu, 343  
uczestnicy, 331, 356  
z ustaloną datą, 341  
z ustalonym zakresem, 345  
wielopoziomowe, 281  
wykonania sprintu, 369  
z góry na dół, 227, 274  
pociąg wersji dystrybucyjnych, 246  
podejmowanie decyzji, 69, 72, 325  
podział zasad na kategorie, 63  
pojemność  
    w punktach historyjkowych, 361  
    w roboczych godzinach, 362  
    zespołu, 360  
porażka, 124  
porównanie  
    oceniania, 154  
    punktów kontrolnych, 96  
    unikania i akumulowania dłużu, 180  
    wysiłku z dokładnością, 153  
    zaangażowania klienta, 198  
    zadań menedżera, 269  
    zasad, 89  
postęp, 84  
potwierdzenie ukończenia pracy, 385  
poziom  
    ufności, 322  
    zaufania, 312  
poziomy  
    autoryzacji, 256  
    planowania, 281  
pozyskiwanie  
    informacji zwrotnej, 77  
    wiedzy, 76  
    wymagań, 70  
praca  
    adaptacyjna, 75  
    częstkowa, 78, 277, 305, 323  
    czekająca na realizację, 81, 304  
    nad cechą, 372  
    nad zadaniami, 372  
    planowana, 75  
    równoległa, 370  
    sterowana przerwaniami, 41  
prace deweloperskie  
    komercyjne, 205  
    wewnętrzne, 205  
pracownik czekający na pracę, 81, 304  
praktyki scrumowe, 46  
precyza, 153  
prezentacja pracy zespołu, 386  
prędkość, 23, 98, 147, 160  
    nieprawidłowe wykorzystanie, 164  
prognozowanie, 162  
sztuczne zwiększenie, 173  
priorytety elementów rejestru, 134  
problemy retrospekcji sprintu, 409  
proces  
    sekwenencyjny, 62, 65, 112  
    sterowany planem, 64, 66  
    wodoszadowy, 62  
    zdefiniowany, 64  
    zwinnej produkcji, 34  
produkcja  
    oprogramowania, 64  
    przemysłowa, 64  
    sterowana planem, 61, 74  
produkt, 142  
prognoza, forecast, 49  
prognozowanie  
    prędkości, 162  
    ekstremalne, 375  
    sterowane testami, 174, 372  
przedwczesne zakończenie sprintu, 102  
przedział prędkości, 162  
przegląd sprintu, 57, 381  
    podejście, 386  
    problemy, 389  
    przygotowania, 384  
    uczestnicy, 382  
przejrzystość, 67  
przepływ, 139  
    jednoelementowy, 80  
    wytwarzania wartości, 261  
przeprowadzanie wizji, 321  
przerzywanie sprintu, 102  
przewidywanie, 68  
przyczyny dłużu technicznego, 172  
punkt  
    przegięcia, 169  
    skupienia retrospekcji, 396  
punkty  
    historyjkowe, 52, 155  
    kontrolne, 96

**R**

refaktoryzacja kodu, 169  
 rejestr  
     portfela, 149  
     produkту, 34, 50, 111, 283, 316  
         duże produkty, 143  
         elementy, 129  
         emergencja, 132  
         ocenianie elementu, 132  
         pielegnacja, 134  
         podział elementów, 140  
         priorytety elementów, 133  
         stopień uszczegółowienia, 131  
         strumień wymagań, 141  
         wiele produktów, 145  
         wiele zespołów, 144  
     spostrzeżeń, 407  
     sprintu, 53, 148, 364  
 retrospekcja sprintu, 58, 393  
     podejście, 398  
     problemy, 408  
     przygotowania, 396  
     uczestnicy, 395  
     zakończenie, 408  
 rola  
     menedżer projektu, 262  
     menedżer zasobów, 251  
     mistrza młyna, 46, 213  
     właściciel produktu, 46, 193  
      użytkownik, 125  
     zespół deweloperski, 46, 223  
 rozmiar  
     elementu rejestru, 132  
     historyjki, 116  
     kolejki, 82  
     wersji dystrybucyjnej, 148  
     zespołu, 233  
 rozmalary  
     elementów w rejestrze produktu, 51  
     zapotrzebowania, 79  
 rozmowa, 114  
 rozpoznanie inwentarza, 80  
 ryzyko procesu deweloperskiego, 76

**S**

samoorganizacja, 226  
 samospełniająca się przepowiednia, 73

Scrum, 25, 34  
     historyjki użytkownika, 113  
     oceny, 147  
     prędkość zespołu, 147  
     rejestr produktu, 130  
     scrumów, 244  
     sprinty, 91  
     wymagania, 109  
     zasady planowania, 273  
     zasady zwinności, 61  
 Scrummerfall, 66  
 sejsmograf emocji, 402  
 sekwencyjny procesie produkcyjny, 112  
 skala ceremonii, 88  
 skutki dłużu technicznego, 170–172  
 spadek  
     przewidywalności, 171  
     satysfakcji klientów, 172  
 spłacanie dłużu, 184–187  
 sposoby ujawniania dłużu, 182  
 sprint, 52, 91  
     definicja ukończenia, 103  
     krótki okres trwania, 94  
     niezmienność celu, 98  
     ograniczenie czasowe, 92  
     stały czas trwania, 97  
 stały rozmiar sprintów, 97  
 stopień uszczegółowienia wymagań, 113  
 stosowanie najlepszych praktyk, 414  
 strategia „szybkiej porażki”, 326  
 strategie  
     aktywności, 306  
     napływu, 299  
     odpływowi, 304  
     planowania portfela, 294  
     strumień wymagań, 141  
     synchonizacja zespołów, 246  
     szacunek względny, 52  
     szczegóły retrospekcji, 398  
     szef właścicieli produktu, 211

**Ś**

ścieżka naprzód, 415  
 środowisko  
     Cynefin, 38  
     Pragmatic Marketing, 206  
     Scrum, 45, 49, 92

**T**

tablica  
historyjek, 320  
zadań, 375  
takt sprintowy, 98  
techniki programistyczne, 374  
temat, 117  
tempo  
podtrzymywalne, 236  
pracy, 86  
przybywania, 300  
ubywania, 300  
testowanie, 174  
testy akceptacyjne, 107  
tradycyjne formy produkcji, 61  
tworzenie  
komponentów, 208  
mapy historyjek, 127  
mapy sprintów, 339  
rejestru produktu, 316  
wizji produktu, *Patrz planowanie produktu*

**U**

ujawianie długu technicznego, 180  
ukończenie prac, 104, 105, 108  
ułatwianie planowania, 94  
umiejętności  
mistrza młyna, 216  
właściciela produktu, 199  
zespołu deweloperskiego, 226  
upraszczanie planowania, 98  
ustalanie wspólnego kontekstu, 400  
ustalona data wersji, 341  
ustalony zakres wersji, 345  
uszczegóławianie rejestru, 131  
używanie Scruma, 413

**W**

validacja założeń, 76  
warsztaty pisania historyjek, 125  
warunki zadowolenia, 107, 115  
WaterScrum, 66  
wersja dystrybucyjna, 139, 278, 285, 303  
cechy obowiązkowe, 317, 338  
cel, 318  
ograniczenia, 333

wiedza potwierdzona, 75  
wizja produktu, 283  
właściciel produktu, 47, 50, 193  
definiowanie kryteriów, 197  
odpowiedzialność, 201  
pielęgnacja rejestru produktu, 197  
podejmowanie decyzji, 201  
udział w planowaniu, 196  
umiejętności domenowe, 199  
umiejętności interpersonalne, 200  
współpraca z interesariuszami, 199  
współpraca z zespołem, 198  
zarządzanie ekoniemią, 195  
wskazanie  
działaliń, 405  
spostrzeżeń, 405  
wskaźniki wymagań, 110, 111  
wybór  
ćwiczeń, 397  
działaliń, 406  
zadań do realizacji, 373  
wydajność, 86  
jakość produktu, 86  
niepotrzebne formalności, 87  
tempo pracy, 86  
wykonanie sprintu, 54, 367  
komunikowanie, 375  
praktyki techniczne, 374  
proces, 368  
uczestnicy, 367  
zarządzanie przepływem, 369  
wykorzystanie zasobów, 82  
wykorzystywania okazji, 302  
wykres  
Gantta, 276  
rozpalania, 349, 351, 379  
spalania, 348, 376

wyliczanie kosztu opóźnienia, 83  
wymagania, 109–113, 124  
wymagania niefunkcjonalne, 122  
wymuszanie priorytetów, 93  
wynik sprintu, 56  
wzorzec ETC, 416

**Z**

- zalety  
krótkich sprintów, 94  
małych rozmiarów zapotrzebowania, 79  
ograniczeń czasowych, 93  
taktowania, 97  
wdrożenia Scruma, 38
- założenie, 76
- zapisywanie historyjek, 113
- zarządzanie  
czasem, 92  
długiem technicznym, 176  
obsługa, 183  
przyrost dłużu, 177  
ujawnianie, 180
- ekonomią, 195, 261
- inwentarzem, 64, 277
- portfelem, *Patrz* planowanie portfela  
przepływem, 369  
sprintu, 140  
wersji dystrybucyjnych, 139  
wytwarzania wartości, 261
- przyrostem dłużu, 177
- zasada  
metod zwinności, 89, 90  
najmniejszego zaskoczenia, 233  
procesów sterowanych planem, 89, 90
- zasady  
inspekcji, adaptacji i przejrzystości, 67  
planowania, 273, 280  
zwinności, 61  
postęp, 84  
praca cząstkowa, 78  
przewidywanie i adaptacja, 68  
wiedza potwierdzona, 75  
wydajność, 86  
zmienność i niepewność, 64
- zasoby, 251
- zastosowania  
metody kanban, 42  
Scruma, 35
- zespoł  
budujący cechy, 240  
budujący komponenty, 240  
deweloperski, 48, 223  
adaptacja produktu, 225  
długotrwałość, 237  
inspekcja, 225  
komunikacja szerokopasmowa, 232  
pielęgnacja rejestru produktu, 225  
planowanie sprintu, 225  
postawa muszkieterów, 231  
przezroczysta komunikacja, 233  
rozmiar, 233  
samoorganizacja, 226  
tempo pracy, 236  
umiejętności typu T, 229  
wykonanie sprintu, 225  
zróżnicowanie, 229  
wielofunkcyjny, 34  
właścicieli produktu, 209, 211
- zmiany, 72
- zmienianie celu sprintu, 99
- zmienność, 64
- zobowiązanie, commitment, 49
- zróżnicowanie zespołu deweloperskiego, 229
- zwinna produkcja, 34
- zwrot z inwestycji, 95, 280
- zysk z cyklu życia, 294

## Notatki

---

# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄZKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW  
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>