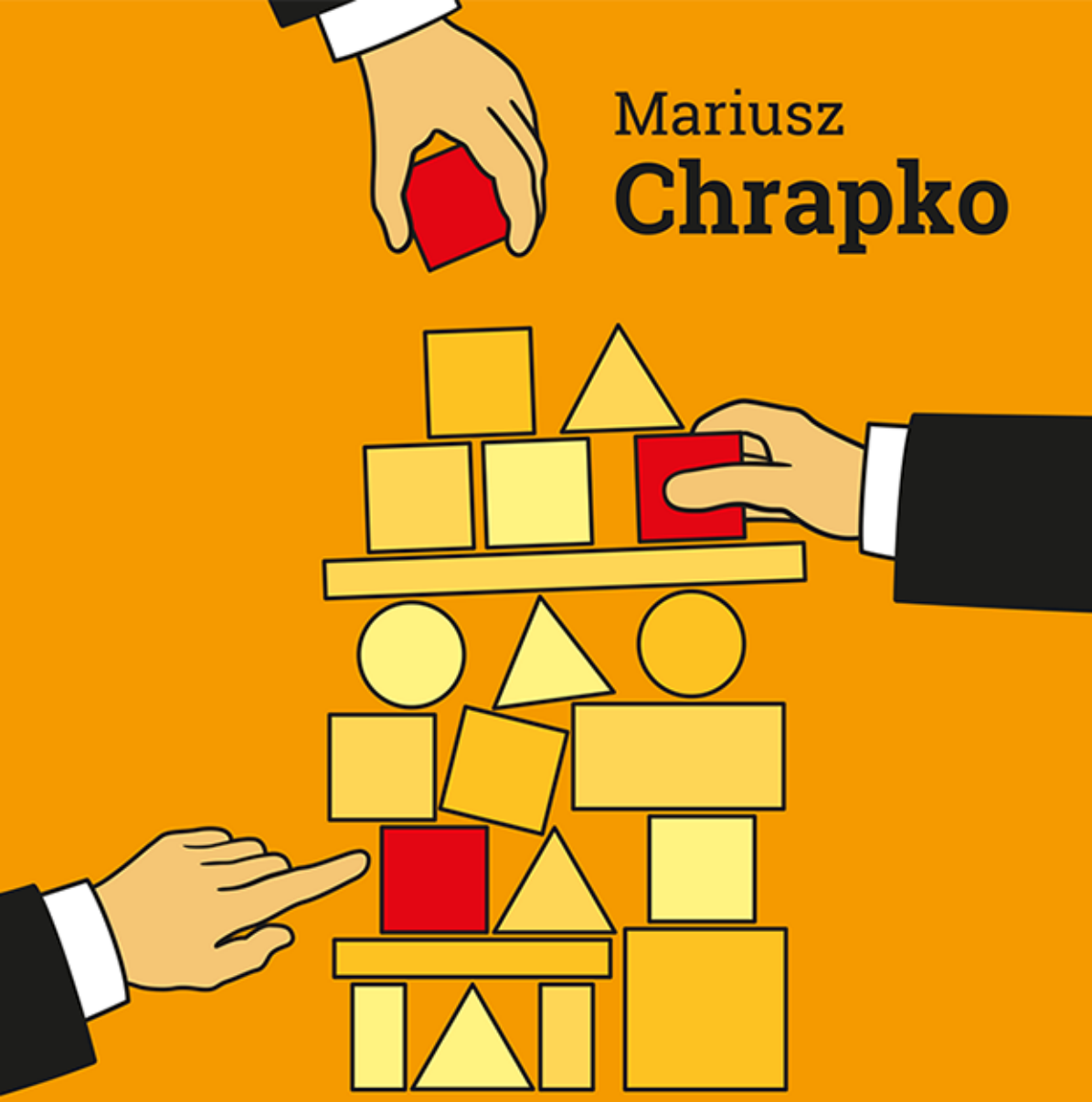


Mariusz
Chrapko

SCRUM

O zwinnym **zarządzaniu projektami**

WYDANIE II ROZSZERZONE

one press

Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek 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.

Redaktor prowadzący: Ewelina Burska
Ilustracje: Ola Bułhak
Projekt okładki: Ola Bułhak

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/scrum2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-2519-2

Copyright © Helion 2015

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

SPIS TREŚCI

O AUTORZE 7

PODZIĘKOWANIA 9

WSTĘP DO DRUGIEGO WYDANIA 13

WPROWADZENIE 17

ZAMIAST WSTĘPU 19

1 **MYŚLENIE ODWROTNE.** Czym jest agile? 21
Szkoła przetrwania 21
Wodospad 22
Pan Samochodzik i tworzenie oprogramowania 27
Ludzie z kryjówek 28
Wychodzenie z kryjówki 30
Myślenie odwrotne 31
Zdrowie szaleńców 32
Manifest agile 33

2 **LEKCJE PŁYWANIA.** O zwinnym zarządzaniu projektami 37
Klasyka i jazz 37
Piankowe wyzwanie 44
Dama z łasiczką 46
Agile i pornografia 48
Scrum 52
Lekcje pływania 58

Kup książkę Poleć książkę

3 **KONTRABASISTA.** Nowe role i obowiązki **61**
Lekcja z *Kontrabasisty* 61
Scrum Master 62
Cechy dobrego Scrum Mastera 67
Wybór Scrum Mastera 74
W dużej organizacji 77
Właściciel Produktu 77
Czy kierownik projektu jest potrzebny w Scrumie? 91

4 **ŁAWA PRZYSIĘGŁYCH.** O hodowaniu zwinnych zespołów projektowych **107**
Ugotowani 107
Zespół = nirwana projektu 109
Szlachetny cel 113
Uprawa zwinnych zespołów 115
Wielofunkcyjne zespoły 130
Zespoły komponentowe 131
Ława przysięgłych 134

5 **PLATON, IDEE I RYBY.** Jak zwinnie zarządzać wymaganiami? **141**
Jaszczurka czy dinozaur? 141
Diabeł tkwi w… komunikacji 143
Historyjki użytkownika 150
ZaINWESTuj w dobre historyjki 157
Rejestr produktu i ocean plazmy 168
Historyjki, epiki, tematy… thriller 170
Wymagania jak góra lodowa 171
Pielęgnacja 172
Praca z dużym rejestrem 174
Tylko 150, reszta nie ma znaczenia 175

6 **LIZANIE ZNACZKÓW I CHIRURGIA MÓZGU.** Szacowanie projektów w Scrumie **177**
Kadzidłowy dym, ekstatyczny trans 177
O istocie szacowania 178
W punktach czy w osobodniach? 200
Trochę techniki i człowiek… się nie gubi 205

7 **O ŻEGLOWANIU I OBIERANIU CEBULI.** Planowanie projektów w Scrumie **215**
Obsesja planowania 215
Zwinne planowanie 216
Żeglowanie i obieranie cebuli 218
Jak się do tego zabrać? 220

Kup książkę Poleć książkę

Plan wydania 239
Planowanie na dużą skalę 241
Jak śledzić postęp prac? 245

8 **BIEGNIJ, FORREST, BIEGNIJ.** Sprint **251**
Jak na nartach po Barcelonie 251
Planowanie sprintu 252
Plan sprintu 265
Gotowi!… Do startu!… Gotowi?! 266
W dużych projektach 268
Zrobione czy niezrobione? 272
Puste taczki 274
Komunikacja 277
Śledzenie postępu prac 279

9 **W POKOJU SZTABOWYM.** Codzienne scrumy **285**
Wniosek o zakończenie wojny 285
Codzienny scrum 290
Jak się komunikować? 302
Zespoły rozproszone 305
Scrum of Scrums 309

10 **FURTKA DO OGRODU.** O tym, jak zorganizować przegląd sprintu **313**
W sklepie z zabawkami 313
Przegląd sprintu 314
Metoda Kawasakiego 316
Zarażać dobrą energią 317
Furtka do ogrodu 318
Atak na Ziemian 319

11 **MYŚLODSIEWNIA.** Retrospektywa na koniec sprintu **321**
Myślodsiewnia 321
Oczyszczenie 322
Najczęściej pomijana praktyka 323
Lekcja anatomii 324
Próba ogarnięcia kuwety 333

12 **WSPÓLNY REJS.** Historia z morza wzięta **343**

SKOROWIDZ **347**

Kup książkę Poleć książkę

Kup książkę Poleć książkę

1

MYŚLENIE ODWROTNE

Czym jest agile?

Większość naszych wielkich wynalazków i genialnych osiągnięć zawdzięczamy lenistwu, czy to narzuconemu, czy dobrowolnemu. Umysł nasz lubi być karmiony, jak łyżeczką, pomysłami innych ludzi, jeśli się go jednak pozbawi tej pożywki, zaczyna, zrazu niechętnie, myśleć samodzielnie, a tego rodzaju myślenie, proszę pamiętać, jest myśleniem oryginalnym i może przynieść bardzo cenne rezultaty.

Agatha Christie, *Zatrute pióro*

Szkoła przetrwania

Jestem wielkim fanem Beara Gryllsa, bohatera programu telewizyjnego emitowanego przez Discovery Channel — *Ultimate Survival* (polski tytuł: *Szkoła przetrwania*). Dzielny Bear, podróżnik i były żołnierz służb specjalnych SAS 21 (*Special Air Service*), uzbrojony jedynie w nóż, manierkę, krzesiwo i ubranie, pokazuje, jak przetrwać w absolutnie ekstremalnych warunkach. Pamiętam jeden z odcinków (właściwie chyba był to pilot pierwszego sezonu), który był kręcony w Górach Skalistych, w USA. Duże wrażenie zrobiła na mnie scena, w której Bear schodził w dół rzeki (było może z 9 metrów) samym środkiem wodospadu. Jeśli ktoś z Was widział ten odcinek, to wie, że Bear pokonywał go trochę „na raty". Najpierw pierwszy etap — dojście do wodospadu. Potem drugi — zdobycie małej półki skalnej, oczywiście niewidocznej ze względu na ogromną masę lodowatej wody, wpadającej wprost na Beara. Kolejny moment to zejście na półkę skalną, ale jak? Za pomocą drabinki zrobionej z linek paralotni, na której nasz bohater wylądował na początku programu. I wreszcie ostatni etap — skok z kilku metrów do ujścia rzeki. Myślę, że opisana scena „pokonywania wodospadu" bardzo dobrze pokazuje pułapkę, w jaką wpada większość z nas, stosując dobrze znany wszystkim tzw. *waterfall model* (z ang. *metodyka kaskadowa*). Na pierwszy rzut oka wydaje nam się, że nie ma innej drogi. Kiedy się pokonuje wodospad, można sobie przecież

Kup książkę Poleć książkę

znacznie skrócić drogę, tym samym szybciej dotrzeć do zamierzonego celu. Do kogoś, kto nie ma dużego doświadczenia w rozwijaniu oprogramowania, taki argument może faktycznie trafiać. Opiera się przecież na bardzo logicznych przesłankach — najpierw musimy wiedzieć, czego chce od nas klient (wymagania biznesowe), żebyśmy mogli myśleć o specyfikacji technicznej. Dopiero kiedy już mamy te dwa etapy za sobą, możemy rozpocząć prace architektoniczne i zająć się dokumentacją wybranych rozwiązań. Gdy już nam się to uda, wreszcie mekka! — upragnione pisanie kodu. Potem już tylko testy, retesty i — uwaga! — Wielki Finał, czyli demonstracja efektów naszej pracy klientowi, który dostaje oponę zawieszoną na linie zamiast wymarzonej huśtawki[1].

Tak sobie myślę, że gdybym ja, podobnie jak Bear Grylls, został rzucony na pastwę losu i musiał przetrwać w najdzikszych miejscach na świecie, na pewno nie schodziłbym w dół wodospadem. Znając moje fizyczne możliwości, dodatkowo biorąc pod uwagę fakt, że nie służyłem w SAS 21, z pewnością poszukałbym jakiejś innej drogi w dół — niekoniecznie takiej, która skazałaby mnie na połamanie sobie rąk i nóg na śliskich kamieniach lub na nabawienie się hipotermii od lodowatej wody. Metody agile (z ang. *zwinny*) to szybka droga do celu, która omija wodospad, bystrza i wszystko, co może się zdarzyć po drodze.

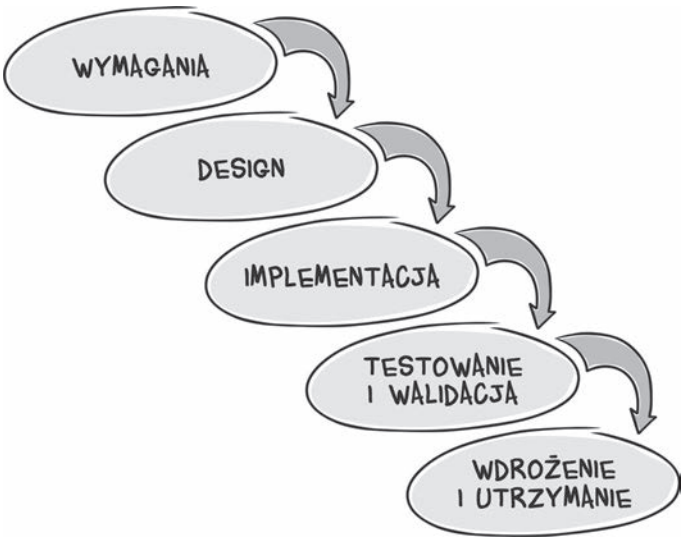
Wodospad

Z powstaniem zwinnych metod tworzenia oprogramowania było trochę tak, jak z powstaniem życia na Ziemi. Ich podstawowe idee, wyznawane wartości i zasady ewoluowały wraz z dyscypliną inżynierii oprogramowania, a więc od początku jej istnienia (przełom lat 50. i 60. XX w.). Niewątpliwym katalizatorem, który przyczynił się do ich rozwoju, był świat tradycyjnych metod wytwórczych, które najpełniej definiuje podejście zwane kaskadowym (od ang. *waterfall*). Polega na tym, że aktywności projektowe realizowane są liniowo (sekwencyjnie) — płyną niczym Nil, tworząc imponujące kaskady wodne (dwie z nich mają postać potężnych wodospadów, nazywanych Ripon i Owena). Cykl życia projektu dzielony jest na określone fazy, które wzajemnie od siebie zależą. Najpierw ustalane są potrzeby klienta (wymagania biznesowe), potem tłumaczy się je na język zrozumiały dla programistów (wymagania funkcjonalne), żeby z kolei, na ich podstawie, można było zaprojektować

[1] *http://www.projectcartoon.com/cartoon/32* (dostęp: 23 września 2014).

Kup książkę Poleć książkę

określone rozwiązania techniczne (projekt systemu). Kolejna faza to implementacja wybranych rozwiązań, które są integrowane, testowane i utrzymywane (ang. *maintenance*) w wyniku wdrożenia. Zwróćcie uwagę na to, że każda faza w tym podejściu stanowi domkniętą całość. Jej produkty wyjściowe (*outputs*) stanowią wejścia (*inputs*) do fazy następnej, co pokazuje rysunek 1.1.



Rysunek 1.1. Cykl kaskadowy projektu

Bardzo dobrze pamiętam problemy wynikające ze stosowania metody kaskadowej, z którymi sam, jako początkujący kierownik projektu, musiałem się kiedyś zmierzyć. Przede wszystkim dość szybko doszedłem do wniosku, że stałe wymagania w projekcie są tak rzadkie jak oscarowe role u Arnolda Schwarzeneggera. Wyobraźcie sobie sytuację, że skończyliście prace związane z przygotowaniem niskopoziomowego projektu systemu (ang. *Low Level Design*). Nagle dzwoni klient i mówi, że chciałby trochę rozbudować dwie ostatnie funkcjonalności, o których rozmawialiście pięć dni temu, i dodatkowo dorzucić jedną nową. Do dzisiaj myśleliście, że wszystko jest pod kontrolą. Nagle cały świat wywraca się do góry nogami, grawitacja nie działa zgodnie z powszechnym prawem ciążenia, a czasoprzestrzeń zakrzywia się w nieprawdopodobny wręcz sposób. Scena żywcem wyjęta z *Incepcji* Christophera Nolana[2]. Chciałoby się włamać przez sen do świadomości klienta i zaszczepić mu ideę cierpliwego czekania i „niewrzucania" nam dodatkowej roboty. Ale…

[2] *http://www.imdb.com/title/tt1375666/* (dostęp: 23 września 2014).

Kup książkę Poleć książkę

nie ma sprawiedliwości na tym świecie. Musimy kilka rzeczy przeprojektować, przeplanować i starać się jakoś podnieść morale sfrustrowanego zespołu. Nie trzeba już chyba dodawać, jak bardzo takie „wrzutki" zwiększają koszty prowadzonego projektu.

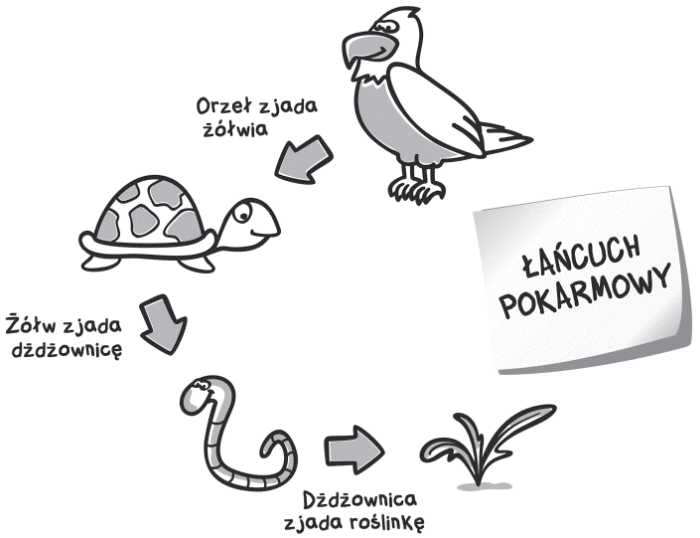
Kolejnym problemem, który napotkałem przy okazji stosowania modelu kaskadowego, jest formalne pozbawienie prawa głosu naszego klienta w trakcie prowadzenia prac rozwojowych. Celowo napisałem „formalne", gdyż klient i tak kontaktował się z nami na wiele różnych sposobów i demonstrował swoje widzimisię. I nic nie mogliśmy zrobić. Nie pomagało alarmowanie o tym problemie wyższego kierownictwa, które, jak jedna z pluszowych zabawek stojących przy wejściu do Smyka, natrętnie powtarzało: „Takie jest życie! Dobrze wiesz, że jest to nasz strategiczny partner (czytaj: dojna krowa) i musimy to jakoś znieść. Głowa do góry! Następnym razem będzie lepiej". Nie było.

Model kaskadowy umożliwia różnego rodzaju „ucieczki" błędów z jednej fazy do drugiej. Wyobraźcie sobie np., że na etapie testów systemowych nagle dowiadujecie się, że określone rozwiązanie zostało źle zaprojektowane i musicie wrócić do wcześniejszej fazy, rozgrzebać architekturę systemu, zaimplementować odpowiednie poprawki, zrobić testy i wdrożyć zmiany na środowisko produkcyjne klienta. W tym momencie Wasze plany biorą w łeb. To jest trochę tak, jak z wyjazdem na weekendowy biwak na Mazury. Pakujemy się: śpiwór, karimata, ubrania, ręczniki (oczywiście wszystko razy kilka, chyba że nie bierzemy żony i dzieci), buty, kosmetyczka, latarka, zapałki, saperka, ładowarka do telefonu, konserwy… Wreszcie wyjeżdżamy z Krakowa. Nawet nam sprawnie poszło, mimo że jest piątek, siedemnasta. Jedziemy już około dwóch godzin, nagle żona, patrząc błędnym wzrokiem na przejeżdżające sąsiednim pasem auta, pyta: „Krzysiek, a namiot?". Możecie sobie wyobrazić, jaki jest dalszy ciąg tej historii. Z modelem kaskadowym jest podobnie. Im później się zorientujemy, że nie mamy namiotu, tym więcej nas kosztuje powrót po niego.

W modelu kaskadowym nad sukcesem każdej fazy czuwa osobny zespół, który skupia ludzi o bardzo zbliżonych kompetencjach. Na przykład za fazę wymagań odpowiadają analitycy biznesowi, analitycy systemowi oraz inżynierowie wymagań. Na etapie projektowania pierwsze skrzypce grają projektanci i architekci. Później do gry wchodzą programiści, którzy implementują przyjęte wcześniej rozwiązania, a wyniki swoich prac przekazują dalej testerom, którzy z kolei sprawdzają, czy wszystko działa zgodnie z wymaganiami, a więc z tym, czym zajmował się pierwszy zespół. Jeżeli wszystko jest przetestowane, a znalezione błędy poprawione, produkt trafia

Kup książkę Poleć książkę

w ręce wdrożeniowców, a w następnej kolejności zespołu zajmującego się pracami utrzymaniowymi. Proces ten przypomina trochę łańcuch pokarmowy, w którym mamy do czynienia z szeregiem różnych organizmów ustawionych w takiej kolejności, że każdy z nich jest źródłem pożywienia dla kolejnego. Na rysunku 1.2 bliżej niezidentyfikowana roślina jest pokarmem dla dżdżownicy, którą zjada ze smakiem na śniadanie mały żółwik, będący niezłym obiadowym kąskiem dla zmęczonego lataniem orła, wprost uwielbiającego te opancerzone stwory. Mamy tu więc i „zjadających", i „zjadanych".



Rysunek 1.2. Łańcuch pokarmowy

W modelu kaskadowym zespół, który zbiera i analizuje wymagania, jest pierwszym ogniwem łańcucha projektowego. Wytwory jego prac (lista wymagań klienta) są „zjadane" przez zespół projektantów („zjadających"), którzy z kolei dostarczają cennego pożywienia (projekt systemu) zespołom programistów. Łańcuszek ten zamyka się na etapie, kiedy klient konsumuje „gotowy produkt". W przyrodzie łańcuchy pokarmowe są długie i wzajemnie poprzeplatane — tworzą sieci różnego rodzaju zależności pokarmowych. I to jest coś, czego nie bierze się pod uwagę w podejściu kaskadowym. Tam bowiem zakłada się płynne przejście pomiędzy jedną fazą a drugą.

Model kaskadowy, przez surowe rozgraniczenie prac wykonywanych przez różne zespoły, powoduje rozmycie odpowiedzialności za rozwój produktu. Każdy zespół skupia się tylko na swojej działce i w żaden sposób nie czuje się odpowiedzialny za to, co robi zespół kolejny. Każdy zamyka się

Kup książkę Poleć książkę

w swoim silosie. Przypomina mi to pewną historię. Kilka lat temu byłem na weselu u mojego znajomego. Nie wiem jak Wy, ale ja, delikatnie mówiąc, nie najlepiej znoszę wszelkiej maści zabawy weselne, które się przy tej okazji odbywają. No, ale przecież nie wypadało odmówić. Gra była bardzo prosta. Polegała na tym, że goście weselni — zarówno ci, którzy zgłosili się dobrowolnie, jak i tacy jak ja, którzy dostali się do zabawy z „łapanki" — mieli utworzyć koło i kolejno podawać sobie małą łyżeczkę do herbaty. W tym czasie zespół pastwił się nad wesołymi weselnikami, grając skoczne „umpa… umpa…", od czasu do czasu robiąc niespodziewane przerwy. I właśnie te „przerwy", ni z gruchy ni z pietruchy, powodowały, że człowiek chciał jak najszybciej pozbyć się tej okropnej łyżeczki i czym prędzej wcisnąć ją w ręce sąsiada. Jest to postawa, którą wyzwalała zasada tej zabawy, że gdy tylko muzyka przestaje grać, a ktoś zostanie z „problemem" w ręku, wypada z gry. Projekty w modelu kaskadowym przypominają bardzo zabawę z łyżeczką. Każdy zespół chce jak najszybciej „wypchnąć" to, co zrobił, do sąsiedniego zespołu — „Niech oni się tym martwią". „To nie jest już nasz problem". Ile razy słyszeliście takie hasła w swoim projekcie? Pamiętacie, jak nieraz dany problem (błąd, poprawka) był przerzucany między programistami, testerami, utrzymaniowcami lub osobami z obsługi klienta? „Przecież my zrobiliśmy to tak, jak było w wymaganiach, to ONI zawalili, nie MY!". Albo: „MY tego nie będziemy naprawiać, bo myśmy tego nie robili, to ICH robota". W modelu kaskadowym poszczególne zespoły przypominają trochę monady, o których mówił Leibniz. Monady nie mają drzwi ani okien. Są światem samym w sobie. Żyją w radykalnej separacji. Nie komunikują się, bo — używając metafory, którą posłużył się niemiecki filozof — do tego potrzebne są drzwi i okna[3].

Winston Royce, który jako pierwszy w artykule *Managing the Development of Large Software System* (1970) opisał model kaskadowy, powiedział bardzo ciekawą rzecz: „Podoba mi się ten pomysł, ale jego implementacja jest ryzykowna i ma tendencję do błędów"[4]. Jest swoistym paradoksem, że wiele osób uważa tego człowieka za twórcę metodyki kaskadowej — człowieka, który widział w niej duże zagrożenie.

[3] Zob. F. Copleston, *Historia filozofii*, tłum. J. Marzęcki, t. IV, Instytut Wydawniczy PAX, Warszawa 1995, s. 299 – 300.

[4] W. Royce, *Managing the Development of Large Software System*, Proceedings of IEEE WESCON 26 (August): 1 – 9, *http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf* (dostęp: 23 września 2014).

Kup książkę Poleć książkę

Pan Samochodzik i tworzenie oprogramowania

Metody agile powstały jako reakcja na założenie, które od samego początku było głęboko zakorzenione w inżynierii oprogramowania: „proces tworzenia oprogramowania jest procesem, który niczym nie różni się od procesu produkcyjnego". Jest linia produkcyjna, są stanowiska robocze (maszynowe, ręczne lub mieszane), pogrupowane według kolejnych operacji procesu technicznego. Idea „linii produkcyjnej" w momencie powstania była alternatywnym rozwiązaniem wobec dotychczasowej produkcji rzemieślniczej i jej utworzenie stanowiło niekwestionowaną zasługę amerykańskiego koncernu Ford. Inżynierowie oprogramowania popełnili jednak zasadniczy błąd, próbując przenieść ten pomysł na grunt projektów software'owych. Co gorsza, na tym założeniu osadzona została cała dotychczasowa filozofia zarządzania ludźmi. Na czym ten błąd polegał? Wyobraź sobie, Drogi Czytelniku, że awansowałeś i zostałeś kierownikiem projektu w zakładzie Tomasza N.N. (tytułowy bohater książki *Pan Samochodzik*), któremu znudziła się już praca historyka sztuki, postanowił więc zacząć masową produkcję pokracznego wehikułu, zbudowanego na bazie rozbitego *Ferrari 410 Superamerica*. Jako menedżer odpowiadasz za linię produkcyjną. Natychmiast eliminujesz wszystkie pojawiające się defekty; jesteś wściekły i nie tolerujesz błędów popełnianych przez pracowników, którzy obsługują linię; traktujesz ich jak kolejny trybik maszyny, który w razie potrzeby zawsze można wymienić na inny; no i jesteś mistrzem wszelkich instrukcji — wszystko musi „tykać" jak w szwajcarskim zegarku i na wszystko jest standardowa procedura. Aha, jeszcze jedno: nie znosisz eksperymentów — nie ma czasu sprawdzać, czy coś się da wykonywać lepiej, czy nie, to nie jest przecież Twoja rola.

Kup książkę Poleć książkę

Takie podejście faktycznie ma sens i sprawdza się podczas pracy przy linii produkcyjnej, np. samochodów. Ogromnym błędem jest jednak próba zaaplikowania tego modelu do projektów software'owych, w których kluczową rolę odgrywa tzw. „czynnik ludzki". To od stopnia zaangażowania ludzi, ich kreatywności, umiejętności akceptowania problemów, radzenia sobie z konfliktami, zdolności komunikacji, pracy w grupie zależy sukces danego projektu. Stosowanie mechanizmów zaczerpniętych ze świata produkcji może prowadzić do postaw zupełnie odwrotnych — stłumienia inicjatywy, braku odwagi w wyrażaniu swoich opinii i pomysłów, chowania się do „kryjówek", z których nie trzeba się zbytnio wychylać, żeby przeżyć kolejny dzień w pracy.

Ludzie z kryjówek

Ponieważ prywatnie zawsze byłem i dalej jestem „fanatykiem" filozofii w każdym wydaniu, przytoczę krótki fragment *Myślenia według wartości* ks. Józefa Tischnera (na pewno kojarzycie jego kultową już dzisiaj *Filozofię po góralsku*[5], jeśli nie — polecam!):

> *Człowiek w kryjówce chroni się przed światem i przed innymi. Przyszłość nie obiecuje człowiekowi nic wielkiego, pamięć przeszłości podsuwa mu przed oczy same doznane porażki, przestrzeń nie zaprasza do żadnego ruchu. Wprawdzie w kryjówce nadzieja nie znika bez reszty, tylko maleje, ale maleje do tego stopnia, że staje się jedynie nadzieją przetrwania*[6].

Styl zarządzania, który próbuje odzwierciedlać świat produkcji, „spycha" członków zespołów projektowych właśnie do „kryjówek". Zauważcie, w ilu projektach, w których sami uczestniczyliście, mieliście do czynienia z sytuacją, gdy kierownik zawsze brał sprawy w swoje ręce w obawie o Wasze kompetencje. Współpracowałem kiedyś z firmą, w której spotkałem się z dość komiczną sytuacją, a przypominała mi ona dawne działania Głównego Urzędu Kontroli Prasy, Publikacji i Widowisk cenzurującego publikacje prasowe, radiowe i telewizyjne w PRL-u. Każdy e-mail, który Scrum Master lub lider techniczny wysyłał do klienta, musiał być wcześniej wnikliwie przeanalizowany i autoryzowany przez kierownika projektu. Z aptekarską wręcz „precyzją"

[5] J. Tischner, *Historia filozofii po góralsku*, Znak, Kraków 1997.

[6] J. Tischner, *Myślenie według wartości*, Znak, Kraków 2000, s. 412 – 413.

Kup książkę Poleć książkę

kierownik studiował każdą wiadomość, która wychodziła poza firmę, i nanosił kluczowe — jak twierdził — poprawki. Kiedyś zapytałem go, dlaczego to robi. Szybko dostałem odpowiedź, że jego ludzie nie mają wyczucia tzw. „kwestii politycznych", więc nie będzie z tego względu ryzykował utraty kontraktu z klientem, który jest jego jedyną „dojną krową". Niestety, nie byłem w stanie go przekonać, że w ten sposób zabija inicjatywę w zespole i — w efekcie — kształtuje mentalność „ludzi z kryjówek", którzy wcześniej czy później przejdą do defensywy i przestaną wykazywać jakąkolwiek wolę twórczego działania. Bo, koniec końców, po co podejmować takie działanie, skoro zawsze istnieje ryzyko, że może się to źle skończyć dla projektu?

Innym poważnym błędem menedżerów wychowanych w świecie produkcji jest zabijanie indywidualności. Przez „indywidualistę" rozumiem osobę mającą swoje zdanie, kreatywną, patrzącą na problemy, które wcześniej czy później pojawią się w projekcie, zawsze w sposób nowatorski i inny od wszystkich proponowanych. Nie chodzi mi jednak o „artystę", który pojawia się w pracy, kiedy chce, a gdy już przyjdzie, to wszystko, co robi, traktuje jako środek do samopodniety. Obecność tego rodzaju ludzi w zespole projektowym jest bardzo destruktywna i trzeba dobrze się zastanowić, zanim zatrudnimy takich „gagatków" do pracy w naszej firmie. Dla kierownika, który swoją inspirację czerpie z linii produkcyjnej, programista-indywidualista jest tylko źródłem problemów. Tymczasem to często *wyjątkowość* decyduje o sukcesie danego przedsięwzięcia. Ten, kto oglądał film *Lśnienie* Stanleya Kubricka, z rewelacyjną rolą Jacka Nicholsona, z pewnością wie, o czym mówię[7].

Ostatnią rzeczą, o której chciałbym wspomnieć w kontekście tradycyjnego stylu zarządzania, jest koncentracja na realizacji zadań. W zarządzaniu wzorującym się na produkcji samochodów nie ma czasu na myślenie o tym, jak coś zrobić. Zadania muszą być wykonywane mechanicznie, żeby ze wszystkim zdążyć na czas. Tymczasem w projektach software'owych tak się nie da. Oczywiście wszyscy o tym wiemy, niemniej często jest tak, że kiedy już dostaniemy projekt do ręki, rzucamy się w wir pracy, najczęściej nie mając odpowiedniej ilości czasu na planowanie, analizę nowych metod i technologii, na szkolenia, czytanie fachowych książek, na wspólne dyskusje o problemach. W ubiegłym roku miałem przyjemność współpracować z firmą, która „programowała na odległość" dla niemieckiego zleceniodawcy — dużej korporacji z, wydawałoby się, uporządkowaną strukturą i tzw. „dojrzałym ładem organizacyjnym". Jednym z zadań, do których zostałem zatrudniony,

[7] *http://www.imdb.com/title/tt0081505/* (dostęp: 23 września 2014).

Kup książkę Poleć książkę

było usprawnienie procesów szacowania parametrów projektu. Chodziło o to, żeby nauczyć ludzi przygotowywać WBS-a (ang. *Work Breakdown Structure*), odpowiednio definiować zadania projektowe, a także wdrożyć jedną z technik estymacyjnych. Początkowo wszystko szło jak po maśle. Zespół z Polski bardzo szybko się wdrożył. Wspólnie przygotowaliśmy kilka arkuszy estymacyjnych do wcześniej złożonych zamówień. I tu zaczęły się schody. Nasi niemieccy koledzy nie mogli zrozumieć, dlaczego w wycenie uwzględniliśmy analizę rozwiązań architektonicznych, czas spędzony na telekonferencjach, a także zrobienie prototypu sprawdzającego jedno z możliwych rozwiązań. Ta firma w ogóle nie brała pod uwagę, że zanim coś zrobimy, musimy najpierw się zastanowić, jak to zrobić. Spotkać się, pogadać, pomyśleć nad rozwiązaniem. Nie da się ludzi podpiąć do klawiatury i kazać im pisać kod.

Wychodzenie z kryjówki

Agile to rodzina metod, które pomagają ludziom zarządzanym według reguł linii produkcyjnej wyjść z kryjówek.

> *Kryjówka ma jakiś próg, który trzeba przekroczyć. Próg znaczy koniec kryjówki i początek nowej przestrzeni. Jeśli się widzi koniec, a nie widzi się początku, będzie się mimo wszystko więcej kochać złudzenie niż prawdę*[8].

Jaki początek proponują metody agile? Przede wszystkim dzięki iteracyjnej metodzie pracy, w której projekt podzielony jest na kilka mniejszych kawałków (iteracji), akceptują „prawo" członków zespołów do robienia błędów. Zastanówcie się przez chwilę, na ile ślepych zaułków w trakcie realizacji Waszych projektów natrafiliście (bez względu na ich rozmiar i złożoność). Ile było meandrów? Ile razy waliliście głową w mur, nie wiedząc, co robić dalej — w którą stronę iść? Agile zakłada, że projekty są podatne na błędy. Jest rzeczą zupełnie normalną, że często zmienia się kierunek prac rozwojowych. Klient co kilka tygodni dostaje fragment działającego produktu, może go sobie oglądnąć, nacieszyć się nim, zgłosić swoje zastrzeżenia oraz zaproponować nowe pomysły. To jest duża siła tego podejścia. Pamiętam, że jako początkujący kierownik projektu, przygotowując harmonogram prac na podstawie metodyki kaskadowej, zawsze miałem problem ze zmiennością wymagań. Zbierałem nawet metrykę śledzącą ich fluktuację (Ile pojawiło się nowych? Ile

[8] J. Tischner, *Myślenie według wartości*, dz.cyt., s. 427.

Kup książkę Poleć książkę

zmieniono starych? Ile te zmiany nas kosztowały?), a wszystko po to, żeby mieć „haka" na klienta na wypadek, gdyby przyszło mu do głowy czepiać się nas, że po raz kolejny nie zdążyliśmy z osiągnięciem zaplanowanego kamienia milowego, bo w trakcie implementacji kilka razy zmieniły się wymagania i pierwotny zakres prac poszerzył się o trzy nowe funkcjonalności. Muszę przyznać, że zawsze byłem bezsilny. Kiedy jednak zacząłem stosować zwinne praktyki projektowe, wszystko się zmieniło. Agile „oswoił" zmianę. Pokazał, że „nie taki diabeł straszny…".

Metody zwinne promują opisany wcześniej indywidualizm. W tym sensie są drogą pod prąd tradycyjnego stylu zarządzania. Dzięki stosowanym metodom i narzędziom tworzą coś, co moglibyśmy nazwać demokratycznym stylem zarządzania. Kierownik projektu nie jest już „królem", który panuje nad ludem, niezależnie od układów politycznych i systemów. Jest bardziej sługą i mentorem osób, które angażują się w projekt. Jego rola musi więc zostać odpowiednio przedefiniowana. Dodatkowo o „demokracji" świadczy fakt, że „władza została oddana w ręce ludu". Odtąd to zespół, a nie kierownik projektu, decyduje o tym, jak zdefiniować poszczególne zadania projektowe oraz kto się nimi zajmie. Rola kierownika musi więc zostać ustalona na nowo, w kontekście wartości i zasad, które przynosi ze sobą idea zwinności. Będzie o tym więcej w rozdziale 3., poświęconym rolom w Scrumie.

Myślenie odwrotne

Metody agile powstały w wyniku próby spojrzenia na proces tworzenia oprogramowania w nieco inny — *odwrotny* sposób. Na myśl przychodzi mi tutaj historia, którą przeczytałem w książce Paula Ardena *Cokolwiek myślisz, pomyśl odwrotnie*[9]. Rzecz zdarzyła się przed olimpiadą w Meksyku, która odbyła się w 1986 r. Był to czas, kiedy technika skoków wzwyż polegała na tym, że sportowcy w trakcie skoku „przerzucali" swoje ciało równolegle do poprzeczki. Na wspomnianej olimpiadzie pojawił się, wtedy jeszcze mało znany zawodnik, Dick Fosbury, który podbiegł do poprzeczki ustawionej na rekordowej wysokości 2 m 24 cm. Wybił się i zamiast skoczyć jak wszyscy, obrócił się do poprzeczki plecami. Unosząc nogi, pokonał ją tyłem. Jego styl skakania obowiązuje do dzisiaj i zasłynął jako „flop Fosbury'ego". Dick

[9] P. Arden, *Cokolwiek myślisz, pomyśl odwrotnie*, tłum. O. Siara, Insignis, Kraków 2008, s. 7.

Kup książkę Poleć książkę

Kup książkę Poleć książkę

SKOROWIDZ

A

Albrecht Allan, 197
analityk
  biznesowy, 83, 89, 130, 144, 145, 148
  systemowy, 130
Arden Paul, 31
Arystoteles, 143, 202
Atlassian, 79, 275

B

Beck Kent, 34, 151
Bezos Jeff, 125
błędy, 260, 261
Boehm Barry, 184
Brass Dick, 74
Brown Tim, 123
burndown chart, *Patrz:* wykres spalania

C, Ć

Cannon-Brokkes Mike, 275, 276
Chambers Harry, 94
Chief Architect, *Patrz:* Główny Architekt
Chief Engineer, *Patrz:* Główny Inżynier
Chief Product Owner, *Patrz:* Właściciel Produktu Główny
Chief Scrum Master, *Patrz:* Scrum Master Główny
ciąg
  Fibonacciego, 208
  geometryczny, 208
Class Owner, *Patrz:* Właściciel Klasy
Cockburn Alistar, 34
Codziennik, 306
Coelho Paulo, 184
Cohn Mike, 157, 167, 190, 217, 272, 297
cross-functional teams, *Patrz:* zespół wielofunkcyjny
Cskiszentmihalyi Mihaly, 140
cykl
  wytwórczy, 51
  życia projektu, *Patrz:* projekt cykl życia
czas trwania, 186, 187, 193
ćwiczenie Piankowe wyzwanie, 44

D

daily scrum, *Patrz:* scrum codzienny
Dama z łasiczką, 47
De Pree Max, 68
definicja ukończenia, 273
definition of done, *Patrz:* definicja ukończenia, kryterium ukończenia pracy
DeLuca Jeff, 34
DeMarco Tom, 115
Demiurg, 40
Derby Esther, 324
diagram rybiej ości Ishikawy, 331
dokumentacja, 35
Drucker Peter, 95
DSDM, 34, 51
Dunbar Robin, 175
duration, *Patrz:* czas trwania

Kup książkę Poleć książkę

Dynamic Systems Development Methodology, *Patrz:* DSDM
dzień idealny, 188, 201, 202, 203, 204, 205, 230, 246, 252

E

effort, *Patrz:* pracochłonność
elicitation, 148
Entergy, 113
entropia, 41, 42
entroskop, 41
epik, 88, 170, 171, 172, 176, 195, 225, 226, 244
estymacja, 30, 124, 163, 179, 181, 195, 199, 200, 203, 206, 207, 211, 234, *Patrz też:* szacowanie
Extreme Progamming, 34, 49, 50, 151, 165, 170

F

Farquhar Scott, 276
FDD, 50
feature team, *Patrz:* zespół funkcjonalny
Feature-Driven Development, 34, *Patrz:* FDD
feng shui, 116
Feynman Richard, 104
Fforde Jasper, 41
Fisher Darin, 79
Fosbury Dick, 31
Fowler Martin, 151
frequent delivery, 52
Furman Richard, 277

G

Gallo Carmine, 314
Główny Architekt, 50
Główny Inżynier, 87
godzina idealna, 257, 266
Goodger Ben, 79
Google, 79
Google Chrome, 79
Google Docs, 306
gra innowacyjna, 66
Gran Torino, 99
Greenleaf Robert, 55, 68
grooming, 172, 267, 297

H

Hamel Gary, 113
hand-over, 149
Heidegger Martin, 209
Heller Michał, 86
hermeneutyka, 142
historyjka użytkownika, 64, 90, 151, 157, 159, 168, 170, 171, 176, 208, 210, 212, 221, 241, 244, 252, 255, 265, 267, 268, 280
  badawcza, 165, 196, *Patrz też:* spike
  cechy, 157, 158, 159, 160, 162, 163, 165, 166
  INVEST, 158
  karta, 154
  konfirmacja, 156
  konwersacja, 155
  łączenie, 159
  tworzenie, 164
  wielkość, 164, 189, 198, 221, 268
  zamykanie, 272
Hobbes Thomas, 156
Hohmann Luk, 329
homo socius, 143

I

ideal day, *Patrz:* dzień idealny
idealna godzina, *Patrz:* godzina idealna
idealny dzień, *Patrz:* dzień idealny
impediment backlog, *Patrz:* przeszkoda rejestr
impediments, *Patrz:* zespół przeszkody
implementacja, 23, 24, 38, 130, 273
inkrement, 48
Innovation Games, 329
inspekcja kodu, 64, 73, 124
integrowanie, 23
interesariusz, 40, 42, 82, 179, 277, 319
INVEST, 158
inżynier wymagań, 144, 145, 148, 149
inżynieria oprogramowania, 22
iteracja, 48, 55, 315, *Patrz też:* sprint

J

Jacobellis Nick, 48
Jaspers Karl, 49, 99
Jeffries Ron, 154

Kup książkę Poleć książkę

JIRA, 79, 275, 282
Jobs Steve, 32, 81, 313, 317

K

kampania Think Different, 32
Kanban, 51
Kawasaki Guy, 316
Kennedy John Fitzgerald, 80
kierownik projektu, 63, 71, 77, 89, 90, 91, 92, 93, 95, 96, 97, 103, 115, 122, 124, 179, 280, 290
kierownik projeku, 206
klika, 112, 114
kod
  flagowy MKS, 272
  inspekcja, *Patrz:* inspekcja kodu
komunikacja, 143, 144, 277, 302, 305
  niewerbalna, 144, 302, 303
  osmotyczna, 52
  werbalna, 144
komunikat JA, 337, 338
komunikator internetowy, 305
konwersacja, 151, 155, 157, 164, 168, 173
kryterium
  akceptacyjne, 157, 210, 268, 316
  ukończenia pracy, 56
Kubrick Stanley, 29

L, Ł

Larsen Diana, 324
lean, 51
Lean Manufacturing, 51
lejek niepewności, 184, 232, 233
Lencioni Patrick, 289
Leonardo da Vinci, 47
liczba Dunbara, 175, 176
Line Product Owner, *Patrz:* Właściciel Produktu Liniowy
linie kodu, 188
Lipnack Jessica, 306
lista kontrolna, 156
Lister Timothy, 115
Low Level Design, *Patrz:* projekt niskopoziomowy
Lowndes Leil, 82
ludzie
  na kształt litery I, 129
  na kształt litery T, 123, 129, 138
Lumet Sidney, 134
łańcuch projektowy, 25
łącznik, 307, 308

M

maintenance, 23
Malle Louis, 48
manifest agile, 34
Manifest zwinnego tworzenia oprogramowania, 34, 49
Manifesto for Agile Software Development, 34
mapa drogowa, 35
McConnell Steve, 184
McLuhann Marshall, 305
McManus Ray, 52
menedżer produktu, 77, 83
metoda
  5W, 331
  agile, 22, 27, 30, 31, 48, 49, 200, 216, 217
  Codziennika, *Patrz:* Codziennik
  Crystal, 34, 52
  DSDM, *Patrz:* DSDM
  Dużej sali, 271
  iteracyjna, 30
  kaskadowa, *Patrz:* model kaskadowy
  Łącznika, *Patrz:* łącznik
  Scrum, 34, 50, 52, 53, 54, 56
    żargon, 55
  zwinna, *Patrz:* metoda agile
mikrozarządzanie, 94, 323
model
  CMMI, 35
  Kano, 66, 226
  kaskadowy, 21, 22, 23, 24, 25, 26, 30, 35, 37, 50, 130, 143, 147, 185, 200, 227
  sekwencyjny, *Patrz:* model kaskadowy
  Tukcmana, 66
  względnego ważenia Karla Wiegersa, 66
MoSCoW, 51
mowa ciała, *Patrz:* komunikacja niewerbalna
Mozart Wolfgang Amadeusz, 37

Kup książkę Poleć książkę

N

Na blacie, *Patrz:* Triangulacja
Nierenberg Gerard, 161
Nolan Christopher, 23
Nonaka Ikujiro, 138, 139

O

osmotic communication, *Patrz:* komunikacja osmotyczna
osmotyczna komunikacja, *Patrz:* komunikacja osmotyczna
osobodzień, *Patrz:* dzień idealny

P

Page Scott, 137, 138
Parmenides, 209
Penderecki Krzysztof, 43
pet project, *Patrz:* pieszczoszek
Pichler Roman, 267, 319
pieszczoszek, 79
Planning Poker, 199, 207, 208, 222
planowanie, *Patrz: projekt plan, sprint planowanie, wydanie planowanie*
  do przodu, 269
Platon, 148
Polanyi Michael, 118
Poppendieck Mary, 51
Poppendieck Tom, 51
Popper Karl, 183
pracochłonność, 126, 186, 193, 197, 201, 203, 226
priorytet, 225, 226, 229, 253, 255, 265, 279
priorytetyzacja wymagań, 51
proces, 63
  empiryczny, 40, 43, 44, 46, 48
  powtarzalny, 38, 40, 46
product backlog, *Patrz:* rejestr produktu
Product Owner, *Patrz:* Właściciel Produktu
produkt
  rejestr, *Patrz:* rejestr produktu
  wydanie, 88
programista, 145, 147, 148, 149, 150, 157, 161, 168, 226
programowanie w parach, 73
projekt
  cykl życia, 22
  interesariusz, *Patrz:* interesariusz
  kaskadowy, 143
  niskopoziomowy, 23
  plan, 36, 38, 216, 217
  przeszkoda, *Patrz:* przeszkoda
  ryzyko, 228
  systemu, *Patrz:* system projekt
  szacowanie, *Patrz:* szacowanie
  złożoność, 40, 41
  zorientowany
    na datę, 237, 238
    na funkcjonalności, 239
projektowanie
  liniowe, 22
  sekwencyjne, *Patrz:* projektowanie liniowe
próżniactwo społeczne, 128
przekazanie, *Patrz:* hand-over
przeszkoda, 300, 301
  rejestr, 301
przywództwo służebne, 55, 64, 68
punkt, 188, 189, 193, 202, 203, 226, 230, 246, 252, 263
  funkcyjny, 188, 197
punkty, 200
Putnam Doug, 126

Q

QSM, 125
Quantitative Software Management, *Patrz:* QSM

R

refinement, 173
rejestr
  produktu, 51, 55, 78, 80, 130, 168, 171, 172, 176, 220, 221, 244
    pielęgnacja, 172, 173
    pluralizm, 174
  przeszkód, *Patrz:* przeszkoda rejestr
  sprintu, 55
Release Kick-Off, 88
retrospektywa, 51, 290, 321, 323, 324, 329, 331, 335
  analiza, 325, 331
  czas trwania, 338

Kup książkę Poleć książkę

ćwiczenia, 326, 329, 334, 335
decyzja, 325, 331
meble, 339
przygotowanie sceny, 325, 326
zamknięcie, 325, 333
zbieranie danych, 325, 328, 329
Ringelmann Max, 128
Robertson James, 148
Robertson Suzanne, 148
rola projektowa, 50, 61
Royce Winston, 26
Rubinstein Artur, 38
ryzyko, 228, 242

S, Ś

samoorganizacja, 95, 96, 97, 103, 140, 280, 302
samotranscendencja, 139
Schwaber Ken, 62, 67, 265
Schwarz Roger, 334
scrum
codzienny, 290, 292, 293, 294, 297, 302, 306, 307, 308, 309
atrybut mówcy, 294
czas, 292
miejsce, 291, 302
na odległość, 303, 304, 305
rola, 61, 63
Scrum Master, 55, 61, 62, 63, 65, 67, 68, 69, 70, 71, 72, 73, 74, 85, 86, 208, 209, 212, 226, 283, 291, 294, 299, 300, 301, 307, 323, 329, 335, 339
Główny, 77, 310
obowiązki, 65
selekcja kandydatów, 75
Scrum of Scrums, 309, 310
servant leadership, *Patrz:* przywództwo służebne
sesja pielęgnacyjna, *Patrz:* grooming
ShipIt Days, 275, 276
Skillman Peter, 44
Sorensen Ted, 80
specjalista, 124, 129
specyfikacja
techniczna, 22
wymagań, 168
spike, 165, 196
spotkanie
codzienne, *Patrz:* scrum codzienny
projektowe, 288, 290
sprint, 55, 64, 150, 164, 166, 221, 222, 230, 253, 321, *Patrz też:* wydanie
cel, 254, 266, 315
demonstracja, 156
długość, 223
informacja zwrotna, 223
planowanie, 172, 219, 252, 256, 258, 262, 265, 267, 268, 271, 277
przegląd, 314, 315, 318, 319
synchronizacja, 270
tablica, 279, 291, 297
zdalna, 281, 292
zamykanie, 85
sprint backlog, *Patrz:* rejestr sprintu
sprint burndown chart, *Patrz:* wykres spalania pracy
stakeholder, *Patrz:* interesariusz
Stamps Jeffrey, 306
Stańko Tomasz, 43
Stewart Potter, 48
story point, *Patrz:* punkt
strefy czasowe, 307
Streisand Barbra, 39
Süskind Patrik, 61
system
architektura, 24
implementacja, *Patrz:* implementacja
kolejkowania pracy, 51
projekt, 23, 25
szacowanie, 179, 181, 184, 186, 206, 207, 212, 220, 222, 231, 235, *Patrz też:* estymacja
w czasie idealnym, 200, 201, 205, 230, 257
w punktach, 189, 192, 193, 195, 197, 198, 200, 202, 203, 205, 230
zwinne, 187, 188, 189, 192, 193, 195, 197, 198
środowisko pracy, 116

T

tablica sprintu, *Patrz:* sprint tablica
tacit knowledge, *Patrz:* wiedza ukryta
Takeuchi Hirotaka, 138, 139
Taylor Cecil, 43

Kup książkę Poleć książkę

telekonferencja, 303, 305
temat, 88, 171, 176, 225, 226, 244
test
  akceptacyjny, 156, 167, 268, 273
  jednostkowy, 273
  regresyjny, 224, 273
tester, 145, 147, 149, 150, 226, 281
testowanie, 23, 24, 130, 132, 167, 224, 273
theme, *Patrz:* temat
Think Different, 32
Tischner Józef, 28, 115, 225
Tomasz z Akwinu, 209
Toyota, 51, 87
Triangulacja, 207, 212, 222
Truman Harry, 73
T-shaped People, *Patrz:* ludzie na kształt litery T

U, V

ujawnianie, *Patrz:* elicitation
user story, *Patrz:* historyjka użytkownika
utrzymywanie, *Patrz:* maintenence
velocity, *Patrz:* zespół prędkość

W

Wake William, 157, 163
warunki środowiskowe, 36
waterfall model, *Patrz:* model kaskadowy
ważenie relatywne, 226
WBS, 30
Welch Jack, 146
wideokonferencja, 212, 304, 305
wiedza ukryta, 57, 118, 119
Wiegers Karl, 66
Wiegersa model względnego ważenia, 66
wielozadaniowość, 120, 128
Witkacy, 46
Właściciel Klasy, 50
Właściciel Produktu, 55, 61, 65, 70, 77, 79, 80, 82, 83, 84, 85, 86, 87, 130, 131, 150, 155, 157, 158, 161, 163, 164, 173, 176, 200, 208, 210, 212, 219, 220, 225, 226, 228, 241, 254, 265, 297, 315, 318
  Główny, 87, 88, 241, 271
  Liniowy, 87, 241
  rekrutacja, 89
Work Breakdown Structure, *Patrz:* WBS
Wujec Tom, 44
wydanie, 221, 241, *Patrz też:* sprint
  długość, 222
  planowanie, 219, 225, 239, 241, 242, 243, 244
wydobywanie, *Patrz:* elicitation
wykres spalania, 245, 299, 306
  liniowy, 245
  pracy, 69
  słupkowy, 246, 247, 248, 249
  sprintu, 282, 283
wymagania, 143, 144, 145, 148, 149, 150, 151, 156, 160, 168, 172, 179, 182, 200, 206
  biznesowe, 22
  funkcjonalne, 22
  pozyskiwanie od klienta, 148, 149
  specyfikacja, 154, 163

Y

Young Jeffrey, 81

Z, Ż

zadanie, 253, 257, 258, 266, 280, 297, 301
  wielkość, 259
zasada ograniczonego zaufania, 156, 179
zespół, 66, 109, 112, 116, 118, 140, 155, 157, 165, 173, 179, 208, 222, 226, 241, 265
  cel, 113, 114
  fazy ewolucji, 66
  funkcjonalny, 50, 130, 131
  komponentowy, 131, 132, 133
  prędkość, 70, 159, 170, 201, 209, 222, 230, 231, 241, 252, 253, 254, 255, 263, 266, 328
  szacowanie, 230, 233, 235
  przeszkody, 67
  rozproszony, 155, 305, 306, 307, 308, 309
  samoorganizujący się, 95
  spotkanie, *Patrz:* scrum codzienny
  wielkość, 125, 126, 127
  wielofunkcyjny, 130, 135, 138, 139, 200
  wirtualny, *Patrz:* zespół rozproszony
żargon, 146, 148

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

Zwinny jak łasica i szybki jak wiatr — od dziś bądź taki dzięki metodzie Scrum!

Współczesny świat w dużej mierze przeniósł się do wnętrza komputerów. Nie chodzi tylko o internet, ale także o bazy danych najrozmaitszych firm i urzędów, programy do obsługi maszyn i sprzętów AGD, skomplikowane systemy logistyczne, magazynowe, handlowe i wszelkie inne. Bez odpowiedniego oprogramowania żaden z tych elementów rzeczywistości nie będzie właściwie działał, a to może być przyczyną małych kłopotów lub wielkich zagrożeń. Jednak stworzenie takiego oprogramowania nie jest rzeczą prostą, tym bardziej że w trakcie pracy nad nim zawsze trzeba liczyć się z modyfikacją pierwotnych założeń. Odpowiedzią na to wyzwanie jest właśnie Scrum.

Scrum to metoda pozwalająca tak zorganizować pracę nad projektem, by możliwe było szybkie i bezbolesne wprowadzanie do niego zmian. Z tej książki, napisanej przez praktyka i wybitnego specjalistę, dowiesz się, jak to działa. Na konkretnych przykładach, okraszonych odwołaniami do popularnych filmów i książek, prześledzisz cały proces tworzenia oprogramowania i zrozumiesz, do czego powinieneś dążyć w swoich własnych projektach z użyciem metody Scrum. Zobaczysz, jak tworzyć zespół i przydzielać role, zarządzać wymaganiami, planować i szacować projekty oraz przeprowadzać sprinty. Jeśli chcesz dogłębnie poznać Scruma i korzystać z niego na co dzień, nie możesz przegapić tej książki!

- **Lekcje pływania** — o zwinnym zarządzaniu projektami
- **Kontrabasista** — nowe role i obowiązki
- **Ława przysięgłych** — o hodowaniu zwinnych zespołów projektowych
- **Platon, idee i ryby** — jak zwinnie zarządzać wymaganiami?
- **Lizanie znaczków i chirurgia mózgu** — szacowanie projektów w Scrumie
- **O żeglowaniu i obieraniu cebuli** — planowanie projektów w Scrumie
- **Biegnij, Forrest, biegnij** — sprint
- **W pokoju sztabowym** — codzienne scrumy
- **Myślodsiewnia** — retrospektywa na koniec sprintu
- **Wspólny rejs** — historia z morza wzięta

SCRUM — sprawdź sam, jak to działa!

Helion
23313 numer katalogowy
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-246-2519-2
9 788324 625192
cena: 59,00 zł

one press