

Wprowadzenie

- Dla kogo przeznaczona jest ta książka? (26)
- Wiemy, co sobie myślisz (27)
- Metapoznanie: myślenie o myśleniu (29)
- A oto, co TY możesz zrobić, aby zmusić mózg do posłuszeństwa (31)
- Przeczytaj koniecznie (32)
- Zespół recenzentów technicznych (34)
- Podziękowania (35)

Rozdział 1. Zapewnianie zadowolenia klienta

- Szlakami Macieja wchodzi do internetu (38)
- W większości projektów trzeba uwzględnić dwa główne zagadnienia (39)
- Rozwój oprogramowania metodą wielkiego wybuchu (40)
- Przenieśmy się w czasie - dwa tygodnie później (41)
- Rozwój oprogramowania metodą wielkiego wybuchu kończy się zwykle WIELKIMI PROBLEMAMI (42)
- Doskonały rozwój oprogramowania polega na... (45)
- Dochodzenie do celu dzięki ITERACJOM (46)
- Każda iteracja to miniprojekt (50)
- Każda iteracja prowadzi do rozwoju oprogramowania o WYSOKIEJ JAKOŚCI (50)
- Klient BĘDZIE zmieniał zdanie (56)
- Wprowadzenie poprawek to Twoje zadanie (56)
- Musisz poradzić sobie z POWAŻNYMI problemami... (56)
- Iteracje automatycznie uwzględniają zmiany (58)
- Oprogramowanie jest gotowe dopiero w momencie jego UDOSTĘPNIENIA (61)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (62)

Rozdział 2. Określanie potrzeb klientów

- Orbity Oriona przystępują do modernizacji (64)
- Porozmawiaj z klientem, aby uzyskać WIĘCEJ informacji (67)
- W obłokach z klientem (68)
- Czasem sesje bujania w obłokach wyglądają tak jak na rysunku... (70)
- Dowiedz się, co użytkownicy NAPRAWDĘ robią (71)
- Wymagania muszą być zorientowane na KLIENTA (73)
- Rozwijaj wymagania na podstawie informacji zwrotnych od klienta (75)
- Opowieści użytkownika opisują, CO należy zrealizować w projekcie... a szacunki określają, KIEDY należy to zrobić (77)
- Rozmowa przy stanowisku pracy (81)
- Gra w pokera planistycznego (82)
- Osądź zasadność założeń (85)
- DŁUGIE w realizacji opowieści użytkownika to ZŁE opowieści użytkownika (88)
- Celem jest zbieżność (91)
- Cykl przechodzenia od wymagań do szacunków (94)
- Na zakończenie można oszacować czas trwania całego projektu (97)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (99)

Rozdział 3. Planowanie z myślą o sukcesie

- Klienci chcą otrzymać oprogramowanie OD RAZU! (102)
- Określanie priorytetów razem z klientem (105)
- Wiemy, co znajdzie się w wydaniu 1.0 (prawdopodobnie) (106)
- Jeśli szacowany czas jest za długi, zmień priorytety (107)
- Im więcej osób, tym mniej korzyści (109)
- Dochodzenie do rozsądnego wydania 1.0 (110)
- Iteracje powinny być krótkie i przyjemne (117)
- Porównywanie planu z rzeczywistością (119)
- Szybkość uwzględnia niespodziewane wydarzenia (121)
- Programiści myślą w kategoriach dni UTOPIJNYCH (122)
- Twórcy oprogramowania myślą w kategoriach dni REALNYCH (123)
- Co zrobić, jeśli iteracja jest za długa? (124)
- Uwzględnij szybkość PRZED zaplanowaniem iteracji (125)
- Czas na dokonanie oceny (129)
- Radzenie sobie z wkurzonymi klientami (130)
- Duża tablica na ścianie (132)
- Jak zrujnować życie osobiste członków zespołu? (135)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (137)

Rozdział 4. Przystępowanie do prawdziwej pracy

- Poznajemy iSwoon (140)
- Łączny czas realizacji zadań (143)
- Uwzględniaj tylko niewykonane zadania (145)
- Umieszczanie zadań na tablicy (146)
- Rozpoczynanie pracy nad zadaniami (148)
- Zadanie jest w toku tylko wtedy, kiedy jest W TOKU (149)
- Co zrobić, jeśli pracuję jednocześnie nad dwoma zadaniami? (150)
- Pierwsze spotkanie "na stojaka" (153)
- Zadanie 1: utworzenie klasy Date (154)
- Krótkie spotkanie robocze: dzień 5, koniec tygodnia 1 (160)
- Krótkie spotkanie robocze: dzień 2, tydzień 2 (166)
- Przerywamy ten rozdział... (170)
- Musisz kontrolować niezaplanowane zadania (171)
- Nieoczekiwane prace podwyższają poziom zadań do wykonania (173)
- Szybkość pomaga, ale... (174)
- Mamy dużo do zrobienia... (176)
- ...jednak DOKŁADNIE wiemy, na czym stoimy (177)
- Wszystko o Szybkości (178)

Rozdział 5. Tworzenie oprogramowania na podstawie doskonałych projektów

- Zespół pracujący nad iSwoon ma poważne problemy (180)
- Taki projekt narusza zasadę jednego zadania (183)
- Wykrywanie wielu obowiązków w projekcie (186)
- Przechodzenie od wielu obowiązków do jednego zadania (189)
- Projekt powinien być zgodny z SRP, a także z zasadą DRY (190)
- Krótkie spotkanie robocze po zakończeniu refaktoryzacji (194)
- Niezaplanowane zadania to wciąż zadania (196)
- Częścią Twojego zadania jest przeprowadzenie prezentacji (197)

- Kiedy wszystko jest gotowe, iteracja jest ukończona (200)

Rozdział 6. Programowanie defensywne

- Podpisałeś nowy kontrakt na aplikację MuzMachina Pro (206)
- Praca nad interfejsem GUI (210)
- Demonstracja nowej MuzMachiny klientowi (213)
- Zaczniemy od KONTROLI WERSJI (216)
- Najpierw skonfiguruj projekt... (218)
- ...a następnie prześlij i pobierz kod (219)
- Większość narzędzi do kontroli wersji próbuje rozwiązywać problemy za Ciebie (220)
- Serwer próbuje SCALIĆ zmiany (221)
- Jeśli system nie potrafi scalić zmian, informuje o konflikcie (222)
- Następne iteracje, następne opowieści (226)
- Mamy kilka wersji oprogramowania (228)
- Opisowe komentarze dodane przy przesyłaniu ułatwiają znalezienie starszego oprogramowania (230)
- Teraz możesz pobrać wersję 1.0 (231)
- (Awaryjne) krótkie spotkanie robocze (232)
- Oznaczanie wersji (233)
- Oznaczenia, gałęzie, pnie - co jeszcze? (235)
- Poprawianie wersji 1.0 - tym razem na poważnie (236)
- Teraz mamy DWIE wersje kodu bazowego (237)
- Kiedy NIE należy tworzyć gałęzi? (240)
- Zen poprawnego rozgałęziania (240)
- Co zapewnia system kontroli wersji... (242)
- System kontroli wersji nie może sprawdzić, czy kod działa (243)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (244)

Rozdział 6.5. Wstaw element a w pole b...

- Programiści nie potrafią czytać w myślach (248)
- Budowanie projektu w jednym kroku (249)
- Ant - narzędzie do budowania projektów w języku Java (250)
- Projekty, właściwości, cele i zadania (251)
- Dobre skrypty kompilacji... (256)
- Dobre skrypty kompilacji wykraczają POZA podstawy (258)
- Skrypty kompilacji to też kod (260)
- Nowy, weź dwójkę (261)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (262)

Rozdział 7. Pojawiają się problemy

- ZAWSZE coś pójdzie źle (264)
- Są trzy sposoby postrzegania systemu (266)
- Testy czarnej skrzynki dotyczą przede wszystkim danych WEJŚCIOWYCH i WYJŚCIOWYCH (267)
- Testy szarej skrzynki ZBLIŻAJĄ Cię do kodu (268)
- Testy białej skrzynki wymagają wiedzy o wnętrzu systemu (271)
- Testowanie WSZYSTKIEGO w jednym kroku (276)

- Automatyzacja testów przy użyciu platformy testowej (278)
- Używanie platformy do uruchamiania testów (279)
- Sterowanie CI za pomocą narzędzia CruiseControl (282)
- Testy gwarantują działanie programu, prawda? (284)
- Przetestowanie całego kodu wymaga sprawdzenia KAŻDEJ GAŁĘZI (292)
- Użyj raportu pokrycia, aby zobaczyć, które metody są sprawdzane (293)
- Uzyskanie wysokiego pokrycia kodu nie zawsze jest proste (295)
- Krótkie spotkanie robocze (297)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (300)

Rozdział 8. Zapewnianie poprawności kodu

- Pisz testy NA POCZĄTKU, a nie na końcu (302)
- NAJPIERW testy (303)
- Witamy w świecie wytwarzania sterowanego testami (303)
- Pierwszy test... (304)
- ...kończy się całkowitym niepowodzeniem (305)
- Doprowadź testy do koloru ZIELONEGO (306)
- Czerwone, zielone, refaktoryzacja... (307)
- W TDD testy STERUJĄ rozwojem kodu (312)
- Ukończenie zadania oznacza, że napisałeś wszystkie potrzebne testy i kończą się one sukcesem (314)
- Kiedy kod przejdzie testy, idź dalej! (315)
- Prostota oznacza unikanie zależności (319)
- Zawsze pisz kod, który można przetestować (320)
- Kiedy wystąpią trudności, przyjrzyj się projektowi (321)
- Wzorec strategii pozwala tworzyć wiele wersji jednego interfejsu (322)
- Przechowuj kod testowy razem z testami (325)
- Testy prowadzą do powstania lepszego kodu (326)
- Więcej testów zawsze oznacza więcej kodu (328)
- Wzorce strategii, luźne powiązanie, zastępowanie obiektów (329)
- Potrzebujemy wielu odmiennych, choć podobnych obiektów (330)
- A gdyby tak wygenerować obiekty? (330)
- Obiekty zastępcze zastępują prawdziwe obiekty (331)
- Obiekty zastępcze to działające zastępniki obiektów (332)
- Dobre oprogramowanie można przetestować (335)
- Niełatwo być zielonym (336)
- Dzień z życia programisty stosującego TDD (338)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (340)

Rozdział 9. Wszystkie elementy łączą się ze sobą...

- Iteracja jest prawie ukończona... (342)
- ...jednak możesz zrobić jeszcze wiele rzeczy (343)
- TRZEBA przeprowadzić testy systemu... (348)
- ...ale KTO ma to zrobić? (349)
- Testy systemu wymagają kompletnego oprogramowania (350)
- Dobre testy systemu wymagają DWÓCH cykli iteracji (351)
- Więcej iteracji oznacza dodatkowe problemy (352)
- Życie (i śmierć) błędu (358)

- Znalazłeś błąd i co dalej? (360)
- Anatomia raportu o błędzie (361)
- Jest jeszcze wiele rzeczy, które MÓGŁBYŚ zrobić... (362)
- Czas na przegląd iteracji (366)
- Przykładowe pytania z przeglądu iteracji (367)
- OGÓLNA lista priorytetów zadań DODATKOWYCH (368)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (370)

Rozdział 10. Jeśli nie jest zepsute... i tak lepiej to naprawić

- Czym jest działające oprogramowanie? (372)
- Potrzebujesz planu następnej iteracji (374)
- Szybkość pozwala uwzględnić... RZECZYWISTOŚĆ BIZNESOWĄ (381)
- Klient NADAL jest najważniejszy (382)
- Oprogramowanie innych zespołów to NADAL tylko oprogramowanie (384)
- Akceptacja klienta? Jest! (387)
- Testowanie kodu (392)
- Houston, mamy problem... (393)
- Krótkie spotkanie robocze (394)
- Nie ufaj NIKOMU (395)
- Zespół bez procesu (400)
- Zespół z procesem (401)

Rozdział 11. Profesjonalne usuwanie błędów

- W poprzednim odcinku (404)
- Najpierw musisz porozmawiać z klientem (406)
- Pierwszy priorytet: umożliwienie zbudowania oprogramowania (412)
- Moglibyśmy naprawić kod... (414)
- ...ale trzeba naprawić funkcje systemu (415)
- Określ, które funkcje działają (416)
- TERAZ wiesz, co (nie) działa (419)
- Co zrobiłbyś w tej sytuacji? (419)
- Oszacuj czas pracy przy użyciu testów punktowych (420)
- O czym informują Cię wyniki testów punktowych? (422)
- Intuicja członków zespołu ma znaczenie (424)
- Poinformuj klienta o szacunkowym czasie naprawy błędów (426)
- Sytuacja wygląda dobrze... (430)
- ...i kończysz iterację sukcesem! (431)
- I klient jest zadowolony (432)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (434)

Rozdział 12. Proces w praktyce

- Definiowanie procesu rozwoju oprogramowania (436)
- Dobry proces prowadzi do dobrego oprogramowania (437)
- Wymagany jest strój wieczorowy (442)
- Wybrane materiały dodatkowe (444)
- Więcej wiedzy = lepszy proces (445)
- Narzędzia do Twojej programistycznej skrzynki narzędziowej (446)

Dodatek A Pięć najważniejszych tematów (których nie poruszyliśmy)

- Numer 1. Diagramy klas w notacji UML (450)
- Numer 2. Diagramy sekwencji (452)
- Numer 3. Opowieści użytkownika i przypadki użycia (454)
- Numer 4. Testy systemu a testy jednostkowe (456)
- Numer 5. Refaktoryzacja (457)

Dodatek B Narzędzia dla doświadczonych programistów

- Techniki rozwoju (460)
- Zasady rozwoju (462)

Skorowidz (465)