

Jesteś inżynierem oprogramowania. Co teraz?

BRAKUJĄCY
PLIK
README

PRZEWODNIK
DLA POCZĄTKUJĄCYCH
INŻYNIERÓW OPROGRAMOWANIA

CHRIS RICCOMINI
DMITRIY RYABOY

Tytuł oryginału: The Missing README: A Guide for the New Software Engineer

Tłumaczenie: Anna Zawila, Tadeusz Zawila

ISBN: 978-83-283-9408-7

Copyright © 2021 by Chris Riccomini and Dmitriy Ryaboy. Title of English-language original: The Missing README: A Guide for the New Software Engineer, ISBN 9781718501836, published by No Starch Press Inc. 245 8th Street, San Francisco, California United States 94103. The Polish-language edition Copyright © 2022 by Helion S.A. under license by No Starch Press Inc. All rights reserved.

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 the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek 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 wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://onepress.pl/user/opinie/brapli>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: onepress@onepress.pl

WWW: <https://onepress.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

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

SPIS TREŚCI

| | |
|--|-----------|
| O autorach | 13 |
| Podziękowania | 14 |
| Wstęp | 15 |
| 1. Podróż, która Cię czeka | 17 |
| Cel Twojej podróży | 18 |
| Mapa Twojej podróży | 19 |
| Góra Żółtodzioba | 19 |
| Rzeka Szkolenia | 21 |
| Przylądek Kontrybutora | 21 |
| Ocean Operacyjny | 22 |
| Zatoka Kompetencji | 22 |
| Pełną parą naprzód! | 23 |
| 2. Świadome dochodzenie do kompetencji | 24 |
| Jak się uczyć, żeby się nauczyć | 25 |
| Przygotuj się do nauki | 25 |
| Ucz się przez działanie | 26 |
| Eksperymentuj z kodem | 27 |
| Czytaj | 27 |
| Obejrzyj prezentacje | 29 |
| Uczestnicz w spotkaniach i konferencjach (z umiarem) | 29 |
| Wymiana doświadczeń i współpraca z doświadczonymi inżynierami | 30 |
| Eksperymentuj z projektami pobocznymi | 31 |
| Zadawanie pytań | 32 |
| Zrób research | 32 |
| Ramy czasowe | 33 |

| | |
|---|-----------|
| Pokaż wykonaną pracę | 33 |
| Nie przeszkadzaj | 34 |
| Preferuj multicast, komunikację asynchroniczną | 35 |
| Zaplanuj żądania synchroniczne | 35 |
| Pokonywanie przeszkód w rozwoju | 36 |
| Syndrom impostora | 36 |
| Efekt Dunninga-Krugera | 37 |
| Co należy, a czego nie należy robić | 38 |
| Wejść na wyższy poziom | 38 |
| 3. Praca z kodem | 39 |
| Entropia oprogramowania | 40 |
| Dług technologiczny | 40 |
| Rozwiązywanie problemu długu technologicznego | 42 |
| Zmienianie kodu | 43 |
| Użyj algorytmu zmiany dotychczasowego kodu | 44 |
| Pozostaw kod czystszy niż ten, który zastałeś | 45 |
| Wprowadzaj zmiany przyrostowo (stopniowo) | 47 |
| Bądź pragmatyczny w refaktoryzacji | 47 |
| Użyj IDE | 47 |
| Stosuj najlepsze praktyki systemu kontroli wersji | 48 |
| Unikanie pułapek | 49 |
| Użycie nudnych technologii | 49 |
| Nie daj się zbałamucić | 51 |
| Nie rozwidlaj bez commitowania do źródła | 52 |
| Oprzyj się pokusie, by napisać to od nowa | 53 |
| Co należy, a czego nie należy robić | 54 |
| Wejść na wyższy poziom | 55 |
| 4. Pisanie działającego kodu | 56 |
| Programowanie defensywne | 57 |
| Unikaj wartości zerowych | 57 |
| Spraw, aby zmienne były niezmiennie | 57 |
| Używanie podpowiadania typów | |
| i statycznego sprawdzania typów | 58 |
| Sprawdzanie poprawności danych wejściowych | 58 |
| Korzystaj z wyjątków | 60 |
| Precyzyjnie określaj wyjątki | 61 |
| Rzucanie wyjątków wcześniej, | |
| przechwytywanie wyjątków późno | 61 |
| Inteligentne ponawianie prób | 62 |

| | |
|--|-----------|
| Stosuj systemy idempotentne | 63 |
| Czyszczenie zasobów | 64 |
| Rejestrowanie | 64 |
| Użyj poziomów dziennika | 65 |
| Przechowuj dzienniki atomowe | 67 |
| Dbaj o szybkość dzienników | 67 |
| Nie rejestruj wrażliwych danych | 68 |
| Metryki | 69 |
| Wykorzystanie standardowych bibliotek metryk | 70 |
| Zmierz wszystko | 72 |
| Ślady | 73 |
| Konfiguracja | 74 |
| Nie kombinuj z konfiguracją | 75 |
| Rejestrowanie i sprawdzanie poprawności wszystkich konfiguracji | 76 |
| Podaj wartości domyślne | 76 |
| Konfiguracja powiązana z grupą | 76 |
| Traktuj konfigurację jako kod | 77 |
| Utrzymuj porządek w plikach konfiguracyjnych | 77 |
| Nie edytuj wdrożonej konfiguracji | 77 |
| Narzędzia | 77 |
| Co należy, a czego nie należy robić | 79 |
| Wejdz na wyższy poziom | 80 |
| 5. Zarządzanie zależnościami | 81 |
| Podstawy zarządzania zależnościami | 82 |
| Wersjonowanie semantyczne | 83 |
| Zależności przechodnie | 84 |
| Piekło zależności | 85 |
| Unikanie piekła zależności | 87 |
| Wyizoluj zależności | 88 |
| Celowe dodawanie zależności | 89 |
| Przypinanie wersji | 89 |
| Wąski zakres zależności | 91 |
| Chroń się przed zależnościami cyklicznymi | 91 |
| Co należy, a czego nie należy robić | 92 |
| Wejdz na wyższy poziom | 92 |

| | |
|---|------------|
| 6. Testowanie | 93 |
| Wielość zastosowań testów | 93 |
| Typy testów | 94 |
| Narzędzia testowe | 97 |
| Biblioteki atrap | 97 |
| Frameworki testowe | 98 |
| Narzędzia do poprawy jakości kodu | 99 |
| Pisanie własnych testów | 100 |
| Pisz czyste testy | 101 |
| Nie przesadzaj z testowaniem | 102 |
| Determinizm w testach | 103 |
| Ziarno generatora liczb losowych | 104 |
| Nie wywołuj zdalnych systemów w testach jednostkowych | 104 |
| Wstrzykiwanie zegarów | 105 |
| Unikaj usypiania i przerywania pracy | 106 |
| Zamykanie gniazd sieciowych i uchwytów plików | 107 |
| Powiązanie z portem 0 | 107 |
| Generowanie unikalnych ścieżek do plików i baz danych | 108 |
| Wyizoluj i usuń pozostałości stanu testów | 108 |
| Nie uzależniaj się od kolejności testów | 109 |
| Co należy, a czego nie należy robić | 109 |
| Wejdź na wyższy poziom | 110 |
| | |
| 7. Przeglądy kodu | 111 |
| Dlaczego warto przeglądać kod? | 112 |
| Uzyskiwanie przeglądu kodu | 113 |
| Przygotuj swój przegląd | 113 |
| Zmniejszanie ryzyka dzięki przeglądom projektów | 114 |
| Nie wysyłaj recenzji w celu wyzwolenia testów | 115 |
| Przechodzenie przez duże zmiany w kodzie | 115 |
| Nie przywiązuj się | 116 |
| Bądź empatyczny, ale nie toleruj chamstwa | 116 |
| Bądź proaktywny | 117 |
| Przeglądanie kodu | 117 |
| Ocena wniosków o przeprowadzenie przeglądu | 117 |
| Zarezerwuj czas na przeglądy | 118 |
| Zrozum zmianę | 118 |
| Przełącz wyczerpującą informację zwrotną | 119 |
| Uznaj to, co dobre | 119 |
| Rozróżniaj problemy, sugestie i drobiazgi | 120 |

| | |
|--|------------|
| Nie przybijaj pieczątki na przeglądach | 120 |
| Nie ograniczaj się do narzędzi dostępnych na stronach internetowych | 121 |
| Nie zapominaj o sprawdzaniu testów | 121 |
| Dąż do konkluzji | 122 |
| Co należy, a czego nie należy robić | 123 |
| Wejść na wyższy poziom | 123 |
| 8. Dostarczanie oprogramowania | 124 |
| Fazy dostarczania oprogramowania | 125 |
| Strategie rozgałęziania | 126 |
| Faza kompilowania | 128 |
| Wersja pakietów | 129 |
| Oddzielne pakowanie różnych zasobów | 130 |
| Faza wydawania | 131 |
| Nie rób z wydań kukułczych jaj | 132 |
| Publikowanie pakietów w repozytorium wydania | 132 |
| Zachowaj niezmienność wydań | 133 |
| Częste wydawanie | 134 |
| Zachowaj przejrzystość w kwestii harmonogramu wydań | 134 |
| Publikowanie dzienników zmian oraz informacji o wydaniu | 134 |
| Faza wdrażania | 135 |
| Automatyzacja wdrożeń | 135 |
| Dokonaj atomizacji wdrożeń | 136 |
| Wdrażaj aplikacje w sposób niezależny | 137 |
| Faza rozpowszechniania | 138 |
| Monitoruj rozpowszechnianie | 138 |
| Zwiększanie wydajności dzięki flagom funkcji | 139 |
| Zabezpiecz kod za pomocą wyłączników automatycznych | 140 |
| Równoległe wersje usług i progi | 140 |
| Wdrażaj w trybie ciemnym | 142 |
| Co należy, a czego nie należy robić | 145 |
| Wejść na wyższy poziom | 145 |
| 9. Pełnienie dyżurów telefonicznych | 147 |
| Jak działa dyżur telefoniczny? | 148 |
| Ważne umiejętności w kontekście dyżurów telefonicznych | 149 |
| Bądź dostępny | 149 |
| Bądź uważny | 150 |

| | |
|--|------------|
| Ustalaj priorytety pracy | 150 |
| Komunikuj się w jasny sposób | 151 |
| Śledź swoją pracę | 152 |
| Postępowanie w przypadku incydentów | 153 |
| Ocena | 154 |
| Koordynacja | 155 |
| Łagodzenie skutków incydentu | 156 |
| Rozwiązanie | 157 |
| Dodatkowe zadania | 160 |
| Udzielanie wsparcia | 162 |
| Nie bądź bohaterem | 164 |
| Co należy, a czego nie należy robić | 165 |
| Wejść na wyższy poziom | 165 |
| 10. Proces projektowania technicznego | 167 |
| Stożek procesu projektowania technicznego | 168 |
| Myślenie o projektowaniu | 169 |
| Zdefiniuj problem | 170 |
| Przeprowadź analizę | 171 |
| Przeprowadzaj eksperymenty | 172 |
| Daj sobie czas | 173 |
| Pisanie dokumentacji projektowej | 173 |
| Wiedz, po co piszesz | 174 |
| Naucz się pisać | 175 |
| Aktualizuj dokumentację projektową | 176 |
| Korzystanie z szablonu dokumentacji projektowej | 177 |
| Wprowadzenie | 178 |
| Stan obecny i kontekst | 178 |
| Motywacja do zmian | 178 |
| Wymagania | 178 |
| Potencjalne rozwiązania | 179 |
| Proponowane rozwiązanie | 179 |
| Projektowanie i architektura | 179 |
| Plan testów | 181 |
| Plan wprowadzenia | 181 |
| Kwestie nierozstrzygnięte | 181 |
| Dodatek | 181 |
| Współpraca przy projekcie | 181 |
| Zrozumienie procesu przeglądu projektu w zespole | 181 |
| Nie zaskakuj ludzi | 182 |

| | |
|---|------------|
| Burza mózgów i dyskusje projektowe | 183 |
| Udział w projektowaniu | 184 |
| Co należy, a czego nie należy robić | 185 |
| Wejdz na wyższy poziom | 185 |
| 11. Tworzenie ewoluujących architektur | 187 |
| Zrozumienie złożoności | 188 |
| Projektowanie z myślą o ewolucji | 189 |
| Nie będziesz tego potrzebował | 189 |
| Zasada najmniejszego zdziwienia | 191 |
| Enkapsulacja wiedzy domenowej | 192 |
| Ewoluujące API | 193 |
| Staraj się, żeby API były małe | 193 |
| Udostępnianie dobrze zdefiniowanych API dla usług | 194 |
| Zachowaj zgodność zmian API | 195 |
| API wersji | 197 |
| Ewoluujące dane | 198 |
| Wyizoluj bazy danych | 198 |
| Używanie schematów | 200 |
| Zautomatyzuj migrację schematów | 201 |
| Zachowaj zgodność schematów | 204 |
| Co należy, a czego nie należy robić | 205 |
| Wejdz na wyższy poziom | 206 |
| 12. Agile'owe planowanie | 208 |
| Manifest Agile | 209 |
| Framework planowania zwinnego | 209 |
| Scrum | 211 |
| Historyjki użytkownika | 211 |
| Zadania | 212 |
| Szacowanie w punktach | 213 |
| Klasyfikacja zadań do zrobienia | 214 |
| Planowanie sprintu | 214 |
| Stand-upy | 215 |
| Przeglądy | 216 |
| Retrospektywy | 217 |
| Roadmapy | 218 |
| Co należy, a czego nie należy robić | 220 |
| Wejdz na wyższy poziom | 220 |

| | |
|---|------------|
| 13. Praca z menedżerami | 221 |
| Czym zajmują się menedżerowie | 222 |
| Komunikacja, cele i procesy rozwojowe | 222 |
| Spotkania 1:1 | 223 |
| PPP | 225 |
| Cele i kluczowe rezultaty (OKR) | 226 |
| Ocena pracownika | 228 |
| Zarządzanie w górę | 229 |
| Uzyskaj informację zwrotną | 229 |
| Przełącz informację zwrotną | 230 |
| Omów swoje cele | 232 |
| Podejmij działanie, gdy coś nie działa | 233 |
| Co należy, a czego nie należy robić | 235 |
| Wejdź na wyższy poziom | 235 |
| | |
| 14. Ścieżka kariery | 237 |
| Do poziomu seniora i jeszcze dalej | 237 |
| Doradztwo zawodowe | 238 |
| Trzymaj się litery T | 238 |
| Uczestnictwo w programach inżynierskich | 240 |
| Kieruj swoim awansem | 240 |
| Ostrożnie zmieniaj miejsca pracy | 242 |
| Trzymaj się własnego tempa | 243 |
| Refleksje końcowe | 244 |



PODRÓŻ, KTÓRA CIĘ CZEKA

Twoja podróż jako inżyniera oprogramowania obejmuje całą karierę zawodową. Po drodze będziesz mieć wiele przystanków — jako student, inżynier, lider techniczny, a może nawet menedżer. Większość nowych inżynierów rozpoczyna pracę z podstawami technicznymi, ale z niewielkim doświadczeniem praktycznym. Kolejne rozdziały poprowadzą Cię w kierunku pierwszego kamienia milowego Twojej kariery, który osiągniesz, gdy będziesz w stanie bezpiecznie dostarczać zmiany w kodzie i płynnie współpracować ze swoim zespołem.



Osiągnięcie pierwszego kamienia milowego jest trudne — potrzebne informacje są rozproszone w internecie lub, co gorsza, ukryte w czyjejs głowie. W niniejszej książce zebrano najważniejsze informacje, które są potrzebne do osiągnięcia sukcesu. Ale jak wygląda praca inżyniera oprogramowania, który odnosi sukcesy, i jak dość do tego etapu?

Cel Twojej podróży

Każdy zaczyna pracę jako młodszy inżynier. Aby awansować, trzeba posiadać kompetencje w kilku podstawowych obszarach.

WIEDZA TECHNICZNA. Znasz podstawy informatyki. Masz umiejętność posługiwania się zintegrowanymi środowiskami programistycznymi (*integrated development environments, IDE*), kompilatorami, debuggerami i frameworkami (platformami) testowymi. Znasz się na ciągłej integracji, metrykach i monitorowaniu, konfiguracji i pakietach. Proaktywnie tworzysz i ulepszasz kod testowy. Podejmując decyzje architektoniczne, bierzesz pod uwagę także operacje (eksploatację).

WYKONANIE. Tworzysz wartość, rozwiązując problemy za pomocą kodu, a ponadto rozumiesz związek między swoją pracą a biznesem. Zbudowałeś i wdrożyłeś małe i średnie funkcje. Pisziesz, testujesz i sprawdzasz kod. Uczestniczysz w dyżurach telefonicznych i rozwiązujesz problemy operacyjne. Jesteś proaktywny i niezawodny. Uczestniczysz w konferencjach technicznych, grupach dyskusyjnych, rozmowach kwalifikacyjnych i prezentacjach.

KOMUNIKACJA. Wyrażasz się w sposób klarowny zarówno w formie pisemnej, jak i ustnej. Potrafisz skutecznie przekazywać i przyjmować informacje zwrotne. Proaktywnie zwracasz się o pomoc i starasz się wyjaśnić niejednoznaczne sytuacje. Poruszasz trudne kwestie i identyfikujesz problemy w sposób konstruktywny. Udzielasz pomocy, gdy jest to możliwe, a także zaczynasz wywierać wpływ na współpracowników. Dokumentujesz swoją pracę. Tworzysz przejrzystą dokumentację projektową i prosisz o informacje zwrotne. W kontaktach z innymi ludźmi wykazujesz się cierpliwością i empatią.

LIDEROWANIE. Samodzielnie pracujesz nad zadaniami, które mają dobrze określony zakres. Szybko uczysz się na błędach. Dobrze radzisz sobie ze zmianami i z niejednoznacznością. Aktywnie uczestniczysz w planowaniu projektów i prac na dany kwartał. Pomagasz nowym członkom zespołu we wdrażaniu się do pracy. Przekazujesz swojemu przełożonemu istotne informacje zwrotne.

Mapa Twojej podróży

Abyś mógł dotrzeć do celu, potrzebna będzie Ci mapa. Pozostała część tego rozdziału pomoże Ci poruszać się zarówno po tej książce, jak i po początkach Twojej kariery. Zaczynamy od Góry Żółtodzioba, gdzie startują wszyscy początkujący. Stamtąd udajemy się w dół, kierując się nurtem Rzeki Szkolenia, gdzie zaczynasz pisać kod i uczyć się lokalnych konwencji i procesów. Kolejnym etapem jest Przyładek Kontrybutora, na którym zaczniesz dostarczać istotne funkcje. Zaś samo dostarczanie funkcji oznacza, że trzeba skoczyć na głęboką wodę i przetrwać sztormy szalejące na Oceanie Operacyjnym. W końcu razem dotrzemy do bezpiecznej przystani w Zatoce Kompetencji.

Wiele akapitów opatrzyliśmy adnotacjami z odnośnikami do rozdziałów. Książkę można czytać linearnie lub przeskakiwać do konkretnych rozdziałów, na których najbardziej Ci zależy. Wiele odniesień do rozdziałów pojawia się w konspekcie więcej niż raz; jest to celowy zabieg. Rozdziały są pogrupowane tematycznie, ale tematy, które poruszamy, będą dotyczyły całej kariery zawodowej. Za każdym razem, gdy będziesz wracać do tego materiału, odkryjesz coś nowego i wyciągniesz nowe wnioski.

Góra Żółtodzioba

Rozpoczynasz swoją podróż jako żółtodziób. Zapoznaj się z firmą, zespołem i ze sposobem załatwiania spraw. Uczestnicz w spotkaniach wprowadzających. Skonfiguruj środowisko programistyczne i dostęp do systemu, a także dowiedz się o obowiązujących w zespole procesach i jego regularnych spotkaniach. Zaczynaj czytać dokumentację i prowadzić dyskusje z kolegami z zespołu. Stań się kontrybutorem, uzupełniając braki w dokumentacji, które znajdziesz w trakcie procesu wdrażania Cię w Twoje obowiązki.

Być może firma organizuje szkolenie dla nowo zatrudnionych pracowników, aby pomóc im w rozpoczęciu pracy. Programy szkoleniowe uczą, jak działa firma, umożliwiają zapoznanie się z organizacją i przedstawiają kierownictwo firmy. Programy dla nowo zatrudnionych pozwalają również poznać nowych pracowników z innych działów — czyli przyszłych kolegów i przyszłe koleżanki. Jeśli w Twojej firmie nie ma programu dla nowo zatrudnionych, poproś swojego przełożonego o wyjaśnienie „schematu organizacyjnego” (kto jest za co odpowiedzialny i kto komu podlega), funkcji różnych działów i ich wzajemnych powiązań; rób przy tym notatki.

PRAWO CUNNINGHAMA I WIATA ROWEROWA

Radzimy Ci udokumentować konwencje, procedury wdrażania do pracy i inne ustne tradycje obowiązujące w zespole. Otrzymasz wiele komentarzy i poprawek. Nie bierz tych komentarzy do siebie. Nie chodzi o to, by napisać doskonały dokument, ale raczej o to, by napisać wystarczająco dużo, by wywołać dyskusję, która doprecyzuje szczegóły. Jest to odmiana *prawa Cunninghama*, które mówi, że „najlepszym sposobem na uzyskanie właściwej odpowiedzi w internecie nie jest zadawanie pytań, ale zamieszczanie złych odpowiedzi”.

Przygotuj się na to, że błahe dyskusje mogą się przeciągać, co jest zjawiskiem zwanym *wiatą rowerową*. Szopa dla rowerów to alegoria autorstwa C. Northcote’a Parkinsona, opisująca komisję, której powierzono przegląd projektów elektrowni. Komisja zatwierdza plany w ciągu kilku minut, ponieważ są one zbyt skomplikowane, żeby w ogóle o nich rozmawiać. Następnie członkowie komisji spędzają 45 minut na omawianiu materiałów do budowy wiaty rowerowej obok tej elektrowni. W pracy technicznej często pojawia się kwestia wiaty rowerowej.

Niektóre firmy wprowadzają dodatkowe procesy dla nowych inżynierów oprogramowania, aby pomóc im w uzyskaniu dostępu do systemów, skonfigurowaniu środowiska programistycznego oraz sprawdzeniu i zbudowaniu kodu. Jeśli taki proces nie istnieje, masz szansę go stworzyć! Zapisz, co robisz w trakcie przygotowań. (Zob. rozdział 2.).

Powinieneś otrzymać niewielkie zadanie, aby nauczyć się podstaw wprowadzania zmian w kodzie i wprowadzania ich do środowiska produkcyjnego. Jeśli nie, należy poszukać takiej możliwości lub poprosić o wprowadzenie jakiejś przydatnej, choć niewielkiej zmiany. Może to być coś tak małego jak aktualizacja komentarza; celem jest zrozumienie kroków, a nie wywarcie wrażenia. (Zob. rozdział 2. i rozdział 8.).

Skonfiguruj edytor kodu lub zintegrowane środowisko programistyczne (IDE). Użyj IDE, z którego korzysta Twój zespół; jeśli nie znasz tego środowiska, znajdź samouczek w internecie. Poznanie IDE pozwoli zaoszczędzić wiele czasu w przyszłości. Skonfiguruj IDE tak, aby stosowało konwencje formatowania kodu zespołu; zapytaj, czym one są i jak je stosować. (Zob. rozdział 3.).

Upewnij się, że Twój menedżer dodaje Cię do meetingów zespołowych i firmowych — codziennych spotkań, planowania sprintu, retrospektyw, spotkań z udziałem całej organizacji itd. Przypomnij swojemu menedżerowi, aby umówił się z Tobą na spotkanie indywidualne, jeśli takie przeprowadza. (Por. rozdział 12. i rozdział 13.).

Rzeka Szkolenia

Po wykonaniu zadań dla początkujących podejmiesz się pierwszej prawdziwej pracy dla zespołu. Prawdopodobnie będziesz pracować na już istniejącym repozytorium kodu. To, co znajdziesz, może Cię zdezorientować lub onieśmielić. Zadawaj pytania, a sam zespół powinien często sprawdzać Twoją pracę. (Zob. rozdział 3. i rozdział 7.).

Uczenie się jest bardzo ważne dla Twojego rozwoju. Dowiedz się, w jaki sposób tworzony, testowany i wdrażany jest kod. Zapoznaj się z pull requestami (czyli powiadomieniami od członków zespołu, że ich kod do jest gotowy do przeglądu przed wprowadzeniem do repozytorium) i przeglądami kodu. Nie bój się prosić o więcej informacji. Zapisz się na konferencje o nowych technologiach, dołącz do nieformalnych spotkań podczas lunchu, grup dyskusyjnych, programów mentorskich itp. (Por. rozdział 2., rozdział 5., rozdział 6. oraz rozdział 8.).

Teraz nadszedł czas na zbudowanie relacji z menedżerem. Poznaj jego styl pracy, zrozum, jakie ma oczekiwania, i porozmawiaj z nim o swoich celach. Jeśli Twój menedżer organizuje indywidualne spotkania, spodziewaj się, że przeprowadzi z Tobą kilka sesji na początek. Menedżerowie zazwyczaj chcą śledzić postępy, więc zapytaj swojego menedżera, jak go informować o stanie prac. (Patrz rozdział 13.).

Prawdopodobnie weźmiesz też udział w swojej pierwszej sesji planowania — zazwyczaj będzie to spotkanie dotyczące planowania sprintu. Możesz także uczestniczyć w retrospektywach lub meetingach dla całej organizacji. Zapytaj o przegląd mapy drogowej i procesu planowania rozwoju. (Zob. rozdział 12.).

Przyłądek Kontrybutora

Do Przyłądka Kontrybutora dotrzesz wtedy, gdy zaczniesz pracować nad większymi zadaniami i funkcjami. Zespół ufa Ci już bowiem na tym etapie i pozwala na bardziej niezależną pracę. Dowiedz się, jak pisać kod nadający się do wprowadzenia na produkcję, który jest przyjazny dla operatora, właściwie zarządza zależnościami i przechodzi testy na czysto. (Zob. rozdział 3., rozdział 4., rozdział 5. oraz rozdział 6.).

Teraz to Ty powinieneś również pomagać kolegom z zespołu. Angażuj się w przeglądy kodu i oczekuj, że koledzy z zespołu będą pytać o Twoje pomysły i opinie. Twoi koledzy mogli zapomnieć, że niedawno dołączyłeś do zespołu, więc zadawaj pytania, gdy czujesz się zdezorientowany. (Por. rozdział 2., rozdział 7. oraz rozdział 10.).

W większości firm obowiązują kwartalne cykle planowania i wyznaczania celów. Uczestnicz w planowaniu pracy zespołu i współpracuj ze swoim menedżerem przy ustalaniu celów i kluczowych wyników (CKW). (Por. rozdział 12. i rozdział 13.).

Ocean Operacyjny

Podczas pracy nad większymi zadaniami dowiesz się więcej o tym, jak kod jest dostarczany użytkownikom. Podczas dostarczania wiele się dzieje, dochodzi bowiem do testowania, budowania, wydawania, wdrażania i rozwijania. Dopracowanie tego procesu wymaga umiejętności. (Zob. rozdział 8.).

Po wprowadzeniu zmian będziesz musiał się zająć obsługą oprogramowania zespołu. Praca operacyjna wiąże się z dużym stresem i wymaga wytrwałości — niestabilność będzie miała bowiem wpływ na klientów. Będziesz usuwać błędy w oprogramowaniu na bieżąco, korzystając z metryk, dzienników i narzędzi do śledzenia. Na tym etapie możesz zostać poproszony o pełnienie dyżurów telefonicznych. Dzięki pracy operacyjnej dowiesz się, jak kod zachowuje się w rękach użytkowników, a także nauczysz się chronić swoje oprogramowanie. (Zob. rozdział 4. i rozdział 9.).

Zatoka Kompetencji

Twój zespół będzie liczył na to, że poprowadzisz teraz mały projekt. Będziesz musiał stworzyć dokument w postaci technicznego projektu i pomóc przy samym planowaniu projektu. Projektowanie oprogramowania wiąże się z nowym poziomem złożoności. Nie należy poprzestawać na pierwszym projekcie; należy zbadać możliwości kompromisów i zaplanować ewolucję systemu w czasie. (Zob. rozdział 10., rozdział 11. oraz rozdział 12.).

Część blichtru zdążyła już zniknąć. Zaczynasz teraz dostrzegać wady w architekturze, systemie budowania i wdrażania, a także w środowisku testowym. Uczysz się utrzymywać równowagę między regularną pracą a niezbędną konserwacją i refaktoryzacją. Nie próbuj jednak wszystkiego przepisywać. (Zob. rozdział 3.).

Zapewne na tym etapie masz też swoje przemyślenia na temat procesów zespołowych. Zapisz swoje obserwacje — co działa, a co nie — i omów swoje pomysły w indywidualnej rozmowie z menedżerem. (Patrz rozdział 13., „Praca z menedżerami”).

Teraz jest też czas na wyznaczanie celów długoterminowych i ocenę wyników pracy. Współpracuj ze swoim menedżerem, aby zrozumieć proces i uzyskać informacje zwrotne od swoich współpracowników. Omów ze swoim menedżerem aspiracje zawodowe, przyszłą pracę, projekty i pomysły. (Por. rozdział 13. i rozdział 14.).

Pełną parą naprzód!

Masz teraz zarówno mapę, jak i cel swojej podróży dla początkujących. Po wylądowaniu w Zatoce Kompetencji staniesz się pełnoprawnym inżynierem oprogramowania, zdolnym do współpracy z zespołem w celu dostarczania wartościowych funkcji. Dalsza część książki pomoże Ci poruszać się po tej ścieżce. I tak oto zaczyna się nasza podróż.

PROGRAM PARTNERSKI

— GRUPY 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
Helion

Dla inżyniera oprogramowania umiejętność kodowania to zaledwie punkt wyjścia. Większość początkujących programistów przekonuje się o tym już w pierwszych dniach pracy w firmie. Nagle się okazuje, że wielu kluczowych spraw zabrakło w programach nauczania. Mowa tu nie tylko o praktykach związanych z tworzeniem i wdrażaniem kodu, ale także o zachowaniach i metodach współpracy ułatwiających odpowiednie działanie zespołu i w efekcie całego przedsiębiorstwa.

To książka przeznaczona dla osób, które chcą rozpocząć karierę inżyniera oprogramowania. Znajdziesz w niej wiele cennych informacji, które zazwyczaj nie są uwzględniane w programach nauczania informatyki na poziomie studiów inżynierskich czy licencjackich, dowiesz się też, czego możesz się spodziewać w pracy. Poszerzysz swoją wiedzę techniczną dotyczącą pisania kodu nadającego się do wdrożenia w środowisku produkcyjnym, opanujesz zagadnienia efektywnego testowania i przeglądów kodu, ciągłej integracji, a także ciągłego wdrażania, dokumentacji projektowej i najlepszych praktyk w zakresie architektury. Zapoznasz się również z informacjami o umiejętnościach miękkich, takich jak techniki zwinnego planowania, efektywna współpraca z kierownictwem i zarządzanie rozwojem własnej kariery.

Najciekawsze zagadnienia:

- oczyszczanie zastanego i tworzenie poprawnie działającego kodu
- testy deterministyczne i przeglądy kodu
- proces projektowania technicznego
- obsługa incydentów produkcyjnych
- zaawansowane techniki architektoniczne
- agile w praktyce firmowej

CHRIS RICCOMINI jest inżynierem oprogramowania. W ciągu ostatniej dekady pracował w takich firmach jak PayPal, LinkedIn i WePay. Jest również twórcą oprogramowania open source, inwestorem i doradcą w dziedzinie startupów.

DMITRIY RYABOY od dwudziestu lat pracuje jako inżynier oprogramowania i menedżer. Pracował w startupach i instytucjach badawczych. Pomagał w tworzeniu i rozwijaniu wielu projektów open source, w tym Apache Parquet.

onepress



Księgarnia internetowa:
<http://onepress.pl>



HELION SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
onepress@onepress.pl

książkiklasybusiness

ebook dostępny na:

ebookpoint

ISBN 978-83-283-9408-7



9 788328 394087

Cena: 59,00 zł

**PO UKOŃCZENIU
UCZELNI CZAS NA
PRAWDZIWĄ SZKOŁĘ
PROGRAMOWANIA!**

Helion