WYDANIE II

PROGRAMISTAMI DA SIĘ ZARZĄDZAĆ!

ZASADY I NARZĘDZIA POMOCNE W ZARZĄDZANIU ZESPOŁAMI PROGRAMISTÓW

MICKEY W. MANTLE | RON LICHTY

Helion

Tytuł oryginału: Managing the Unmanageable: Rules, Tools, and Insights for Managing Software People and Teams (2nd Edition)

Tłumaczenie: Maksymilian Gutowski

ISBN: 978-83-283-6977-1

Authorized translation from the English language edition, entitled MANAGING THE UNMANAGEABLE: RULES, TOOLS, AND INSIGHTS FOR MANAGING SOFTWARE PEOPLE AND TEAMS, 2nd Edition by MICKEY MANTLE; RON LICHTY, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright © 2020 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 SA, Copyright © 2021.

Microsoft® Windows®, Microsoft Excel®, and Microsoft Office® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Figures 5.1, 7.7, 7.8: Photos by Mickey W. Mantle

Figure 9.2: Photo by Ron Lichty

Cohn, Mike, Succeeding with Agile, 1st Ed., © 2010. Reprinted and Electronically reproduced by permission of Pearson Education, Inc., New York, NY

Figures 5.2, 9.3, 9.4, 9.5, 9.6: Screenshots of Microsoft Excel © Microsoft 2019

Cover photo by Vasilius/Shutterstock

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.

Autorzy oraz Helion SA 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. Autorzy oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA
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/progd2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

2

Zrozumieć programistów

Programiści bardzo się od siebie różnią na wielu płaszczyznach, a trzeba być za pan brat z dziedziną programowania, żeby móc to w pełni docenić. W większości firm kierownictwo wysokiego szczebla postrzega wszystkich programistów jako jednolitą masę, co jest poważnym błędem. Szefom takim jak Bill Gates z Microsoftu, John Warnock z Adobe Systems i Mark Zuckerberg z Facebooka udało się nie wpaść w tę pułapkę, ponieważ w głębi serca każdy z nich był programistą.

Dlaczego ma to znaczenie? Być może nie powinno mieć, ale jednak ma. Po latach zarządzania programistami niezmiennie zachwycają nas istotnie dzielące ich różnice oraz to, jak bardzo odmienne podejścia do rozwiązywania problemów i motywowania pracowników musimy przyjmować. Jasne jest dla nas jedno: trzeba przede wszystkim zrozumieć, kim naprawdę jest każdy z programistów, aby móc nimi z powodzeniem zarządzać.

Należy zwrócić tutaj uwagę na bardzo ważną rzecz: nie zauważyliśmy, by wiek, płeć i pochodzenie etniczne lub kulturowe stanowiły w większym stopniu o takich różnicach. Z doświadczenia wyniesionego z zatrudniania setek programistów i zarządzania nimi wiemy, że programiści różnią się od siebie tym, co tkwi w ich wnętrzu, a nie jakimikolwiek zewnętrznymi cechami. Wyszkolenie i doświadczenie oczywiście mają pewne znaczenie, ale najważniejszymi wyróżnikami są indywidualne talenty i przyrodzone cechy.

Programistów można zrozumieć na różne sposoby. Przyjrzymy się im teraz z kilku perspektyw:

- dyscyplina programowania,
- rodzaj programisty,
- specjalizacja dziedzinowa,
- wymagania i umiejętności,
- lokalizacja i relacje,
- charakter pokoleniowy,
- typ osobowości.

Kup książkę Poleć książkę

Dyscyplina programowania

Pierwsza perspektywa wiąże się ze zrozumieniem wszystkich dyscyplin programowania, do których można ich przypisać. Uwzględniamy zazwyczaj następujące grupy:

- programiści systemów wbudowanych i IoT,
- programiści front-endowi,
- programiści back-endowi,
- programiści baz danych,
- web developerzy i inni twórcy skryptów,
- programiści full stack.

Istnieje oczywiście wiele specjalizacji, które umykają powyższej klasyfikacji. Te sześć kategorii jednak obejmuje lwią część ogółu programistów na świecie.

Przy omówieniu tych różnorakich dyscyplin warto wspomnieć o specjalizacji, jaką jest analiza i obróbka danych. Praktykuje ją stale powiększająca się grupa specjalistów, którzy łączą w swojej pracy analizę danych z obsługą baz danych i programowaniem, aby interpretować olbrzymie zbiory informacji. W przeszłości można by ich było zakwalifikować do programistów baz danych, ale współcześnie wyróżniają ich zdolności analityczne i biegła znajomość narzędzi służących do korzystania z obszernych zasobów gromadzonych danych i do zarządzania nimi. Uważamy, że ta konkretnie grupa znajduje się na obrzeżach społeczności standardowych programistów, więc choć przedstawione w tej książce treści mogą okazać się przydatne w zarządzaniu analitykami danych, nie podejmujemy się tutaj opisania ich odrębnej dyscypliny.

Musimy także wspomnieć o DevOps/DevSecOps. Jeżeli Twoim programistom przydzielono także rolę DevOps, może to mieć wpływ na charakter przynależności do wspomnianych grup, wobec czego do tej specjalizacji przejdziemy już po omówieniu sześciu dyscyplin.

Programiści systemów wbudowanych i IoT

Programowanie systemów wbudowanych to specjalizacja, która pojawiła się niedługo po wprowadzeniu generacji mikroprocesorów, których niewielkie rozmiary pozwalały na używanie ich do sterowania konkretnymi urządzeniami. Choć kod pierwotnie był wykonywany bezpośrednio przez sprzęt bez pomocy systemu operacyjnego, obecnie niemal wszystkie systemy wbudowane działają na (stosunkowo) lekkich systemach operacyjnych i wykonują konkretne zadania przy użyciu czujników i interfejsów kontrolujących mechaniczne, sprzętowe i połączone systemy, takie jak elektronika lotnicza, routery sieciowe, światła uliczne, zmywarki, samochody i niemal dowolne inne przedmioty.

Programiści IoT (ang. *Internet of Things* — internet rzeczy) zajmują się podobnymi rzeczami z uwzględnieniem objęcia połączeniem sieciowym tradycyjnie prostych urządzeń i zwyczajnych przedmiotów. Urządzenia IoT poza swoimi standardowymi funkcjami

Kup książkę Poleć książkę

mają możliwość łączenia się z usługami opartymi na chmurze, które komunikują się z nimi, aby monitorować, kontrolować i gromadzić dane na potrzeby analizy i prezentacji.

Programiści systemów wbudowanych i IoT mogą mierzyć się z ograniczeniami przetwarzania, pamięci i wydajności w czasie rzeczywistym, którymi przedstawiciele innych dyscyplin programowania zwykle się nie przejmują (choć prawdopodobnie powinni).

Programiści front-endowi

Większość programistów to programiści front-endowi[1]. Odpowiadają oni za interfejs użytkownika (UI) i użyteczność (UX), a także — z punktu widzenia użytkownika — za ogólne działanie aplikacji, w tym za komunikację i wymianę danych z back-endem. Tworzą oni m.in. następującego typu aplikacje:

- gry na konsole do gier, takie jak PlayStation, Xbox i Nintendo;
- aplikacje na inteligentne telewizory i inne urządzenia multimedialne, takie jak Apple TV i Amazon Fire TV;
- aplikacje na telefony i tablety z iOS lub Androidem;
- aplikacje instalowane w systemach operacyjnych Windows, macOS i Linux;
- coraz większą liczbę aplikacji przeglądarkowych (gdzie przeglądarka internetowa sama w sobie jest jedną z najważniejszych aplikacji front-endowych).

Programiści front-endowi mogą odpowiadać za UX, ale nie oznacza to, że zajmują się jego projektowaniem — to zadanie lepiej zostawić projektantowi UX, czyli komuś z wykształceniem lub doświadczeniem z dziedziny projektowania wrażeń z interakcji. Projektanci UX muszą natomiast ściśle współpracować z programistami front-endowymi, aby mieć pewność, że implementacja ich koncepcji jest realna i bezproblemowa, mając na uwadze inne ograniczenia aplikacji.

Programiści front-endowi mogą mieć swoje specjalizacje (np. tworzenie na potrzeby iOS lub Androida, macOS lub Windows, PlayStation lub Xbox), ale muszą też posiadać umiejętności ogólne i być w stanie wykorzystać z możliwie pozytywnym skutkiem zatwierdzone frameworki i zestawy narzędzi. Ponieważ dostępnych frameworków jest coraz więcej, ważne jest, aby odpowiednio dobrać narzędzia. Wybór ten nie powinien zależeć od pojedynczych programistów, lecz zostać podjęty jako kluczowa decyzja w sprawie architektury.

[1] W pierwszym wydaniu książki nazwaliśmy tę kategorię programistami klienta, ale w ciągu minionej dekady zaczęto tworzyć coraz mniej aplikacji z myślą wyłącznie o komputerach lub wybranych urządzeniach; niemal wszystkie współczesne aplikacje łączą się z usługami back-endowymi, choćby w minimalnym stopniu. Przy rozszerzonych możliwościach łączności i przesyłu, zwłaszcza w obliczu wprowadzenia 5G, podział aplikacji na front-end i back-end staje się uniwersalny.

Kup książkę Poleć książkę

Programiści back-endowi

Programiści back-endowi odpowiadają za rozwój usług i udostępnianie danych potrzebnych dla front-endowców.

W przeszłości często odpowiadali za stawianie i konfigurowanie serwerów, ale upowszechnienie się usług opartych na chmurze w znacznej mierze wyeliminowało konieczność wykonywania takich prac. Obecnie odpowiadają oni za prowadzenie ewaluacji konkurujących ze sobą usług opartych na chmurze, wybór potrzebnych konfiguracji, procesów tworzenia wersji zapasowych i usług, a także tworzenie kluczy bezpieczeństwa, przyznawanie dostępu i monitorowanie stanu systemu oraz raportów.

Po przygotowaniu infrastruktury back-endowej programiści we współpracy z front-endowcami określają wymagania aplikacji oraz logiki biznesowej i programują potrzebne usługi. Podobnie jak programiści front-endowi, back-endowcy mogą przebierać w licznych frameworkach i narzędziach, które pomagają im we wdrażaniu wymaganych usług.

Programiści baz danych

Programiści baz danych to zupełnie inna kategoria. Zazwyczaj posługują się zupełnie innym językiem programowania, używają innych narzędzi i piszą programy, które zwracają innego typu wyniki niż te, które tworzą front-endowcy i back-endowcy. Programiści baz danych zazwyczaj zajmują się wyłącznie organizacją, magazynowaniem i ekstrakcją danych dla użytkownika końcowego lub aplikacji. Po utworzeniu bazy danych programiści back-endowi i front-endowi mogą bezpośrednio pobierać dane z bazy, używając współpracującego z nią frameworka.

Istnieją dwie szersze kategorie programistów baz danych, o których istnieniu należy wiedzieć: programiści OLTP (ang. *online transaction processing* — przetwarzanie transakcji online) i programiści magazynów danych. Programiści OLTP koncentrują się na operacjach wejścia-wyjścia o wysokiej częstotliwości, a programiści magazynów danych na dużych operacjach wejścia-wyjścia o niskiej częstotliwości. Choć skupiają się one na rozwiązywaniu różnych problemów, programiści z obydwu kategorii korzystają z podobnych narzędzi i podejść do produkcji.

Choć znaczenie baz danych i danych jako takich wzrasta od lat, zakres obowiązków programistów z tej dziedziny zmalał ze względu na coraz większe możliwości systemów baz danych i narzędzi. Nadal możesz wprawdzie potrzebować starszego programisty — najprawdopodobniej architekta baz danych — który dopilnuje, by struktura rekordów i pól danych umożliwiała łatwy dostęp, niemniej większość wcześniej wykonywanych przez takich programistów prac powierza się zautomatyzowanym narzędziom lub aplikacjom pisanym przez programistów front-endowych i back-endowych. Nawet kluczowe niegdyś zadania związane z dostrajaniem baz danych i dbaniem o ich odpowiednie skalowanie są dziś w znacznej mierze zautomatyzowane w Oracle, SQL Server i innych systemach bazodanowych.

Kup książkę Poleć książkę

Ponadto różnice między systemami baz danych z biegiem lat zatarły się, a wiele z „podstawowych" umiejętności technicznych programisty bazy danych znajduje zastosowanie na przestrzeni różnych systemów. Te same podstawowe pojęcia są obecne we wszystkich takich systemach, które używają poleceń SQL (i ich odpowiedników w API) do uzyskiwania dostępu do danych i zapewniania programistom back-endowym dostępu do baz danych za pośrednictwem interfejsów ODBC (ang. *Open Database Connectivity* — otwarte łącze baz danych). Może się zatem wydawać, że specjalista od jednego z takich systemów będzie tak samo dobrze radził sobie z pozostałymi, lecz z obserwacji wiemy, że doświadczenie bezpośredniej styczności z określoną bazą danych jest konieczne, aby móc wykonywać na niej jakiekolwiek działania wykraczające poza dość podstawowe czynności.

Głównym aktualnie trendem w technologiach baz danych jest stosowanie systemów opartych na chmurze. Sprzyjają one zmniejszaniu ilości pracy programistycznej i administracyjnej, a w znacznej mierze utrzymują się samodzielnie. Systemy takie jak Amazon Redshift, Google Big Query i Snowflake ograniczyły w dużym stopniu złożoność działań administracyjnych (tworzenia wersji zapasowych, optymalizacji zapytań itd.) i zapewniają wydajność oraz niezawodność, których potrzeba dziś do wykonywania operacji na „big data".

Architekci baz danych są jak mechanicy samochodowi. Możesz przyjąć, że każdy mechanik będzie umiał wymienić oponę lub wycieraczkę, ale w przypadku konieczności przeprowadzenia poważniejszych prac nigdy przecież nie powierzyłbyś porsche mechanikowi, który nie miał styczności z samochodami tej marki.

To samo można powiedzieć o architektach baz danych: możesz przyjąć, że każdy programista baz danych będzie w stanie napisać raport pobierający dane np. z bazy Oracle, ale nie powierzyłbyś poważniejszych prac nad architekturą bazy danych komuś, kto nie może pochwalić się doświadczeniem w pracy z konkretnym systemem, którego będzie używać się na produkcji (np. Oracle, SQL Server, Hadoop lub Aurora, Redshift i Relational Database Service należących do Amazon Web Services).

Web developerzy i inni twórcy skryptów

Wielu web developerów korzysta z zupełnie innych zestawów narzędzi programistycznych niż inni programiści, którzy w większości prac programistycznych posługują się poważnymi językami programowania, takimi jak C, C++, C#, Java, Kotlin, Golang, Python lub Ruby. Web developerzy z kolei wykonują swoje zadania głównie za pomocą języków znaczników (np. HTML, XML, CSS, ASP/JSP) i narzędzi skryptowych (np. Perl lub Python, PHP, JavaScript). Część prac przebiega według schematu „skopiuj, wklej i zmodyfikuj", co polega na kopiowaniu działającego kodu i ewentualnym przekształceniu go tak, aby wykonywał inne zadanie. Niekiedy korzystają także z narzędzi ułatwiających pisanie i wdrażanie skryptów, takich jak Adobe Dreamweaver i Chrome DevTools. Oznacza to, że choć programiści zajmujący się samym front-endowym programowaniem interfejsów użytkownika witryn internetowych mogą skorzystać na formalnym kształceniu informatycznym, nie jest im ono tak potrzebne jak programistom z innych dziedzin.

Kup książkę Poleć książkę

Z drugiej strony tendencja do przerzucania obliczeń z serwera na przeglądarkę przy użyciu JavaScriptu i frameworków opartych na JavaScripcie i Node zmieniła sposób, w jaki tworzy się strony internetowe. Web developerzy od dawna musieli się mierzyć z problemami związanymi z kompatybilnością przeglądarek. Obciążenie strony klienta odpowiedzialnością za wykonywanie obliczeń pogłębiło te problemy, wymuszając na web developerach zapoznanie się z bardziej tradycyjnymi dyscyplinami programowania, a tym samym wykształcenie się nowej kategorii pracowników, którzy potrafią odnaleźć się w roli programistów front-endowych. Web developerzy muszą uczyć się programowania po stronie klienta i być gotowi mierzyć się ze związanymi z tym wyzwaniami.

Specjalistyczna wiedza potrzebna do opanowania szerokiego przeglądu dostępnych narzędzi, frameworków, języków, protokołów i API przeglądarek sprawia, że tacy programiści stają się coraz istotniejszą częścią zespołu programistycznego.

Programiści full stack

Opisy powyższych pięciu kategorii programistów są dość uogólnione, a wysoce wykwalifikowani programiści zazwyczaj potrafią zająć się wszystkimi tymi rodzajami prac. Programiści mają jednak skłonność do zajmowania się w szczególności jedną z tych kategorii i osiągają najlepsze rezultaty, gdy pracują nad preferowanym przez siebie rodzajem kodu.

Za sprawą upowszechniania się internetu, a zwłaszcza pojawienia się przetwarzania w chmurze jako praktycznej alternatywy dla samodzielnie hostowanych serwerów z czasem zaczął się rozwijać nowy typ programisty, który tworzy aplikacje, pracując na całym stosie środowisk programistycznych — front-endzie, back-endzie, bazach danych i witrynach internetowych. Na utalentowanych programistów „full stack" panuje istotny popyt, zwłaszcza wśród start-upów, dla których zatrudnienie programisty zdolnego do wykonywania zadań, którymi wcześniej zajmowało się kilku specjalistów, jest w oczywisty sposób efektywne kosztowo i sprzyja wydajności pracy ze względu na ograniczenie ilości problemów związanych z komunikacją.

Jedną z głównych przyczyn wzrostu znaczenia programistów full stack jest obrodzenie narzędziami, zwanymi na ogół *frameworkami*, które pozwalają na wykorzystanie znacznej części niszowej wiedzy, jaką programista musi posiadać, aby z powodzeniem radzić sobie z różnymi komponentami aplikacji. Orientacja w tym, jakich frameworków używać i jak je ze sobą łączyć, jest sztuką samą w sobie. Dobry dobór frameworków, narzędzi programistycznych i usług w chmurze pozwala na opracowywanie większości aplikacji bez konieczności wynajdywania koła na nowo lub zajmowania się „programowaniem niskopoziomowym". Dopracowanie aplikacji tak, aby działała szczególnie sprawnie, często jednak wymaga ingerencji wysoce uzdolnionego specjalisty.

Programiści full stack są dostatecznie kompetentni w każdej z powyższych dziedzin, ale zazwyczaj są ekspertami jedynie w jednej z nich. Zetknęliśmy się wprawdzie z programistami full stack, którzy wydawali się równie sprawni we wszystkich tych obszarach, ale uznajemy to raczej za wyjątek od reguły (który zasługuje na odpowiednio wyjątkowe wynagrodzenie).

Kup książkę Poleć książkę

DevOps

DevOps jest strategią wdrażania oprogramowania, stworzoną z myślą o zmniejszeniu dystansu dzielącego programistów od personelu operacyjnego. Choć praktyczna implementacja DevOps bywa bardzo zróżnicowana, strategia i pierwotne wdrożenie DevOps często bierze się z inicjatywy starszych architektów zespołów programistycznych i operacyjnych, po czym obowiązki związane ze sprawowaniem pieczy nad takim przedsięwzięciem, konserwacją i doskonaleniem procesów przekazywane są zespołowi programistycznemu, który zajmuje się zadaniami związanymi z tworzeniem, wdrażaniem, monitorowaniem i (w razie konieczności) prowadzeniem napraw. Sprawia to, że zespół programistyczny zamiast pisać kod i „przerzucać go do działu IT", staje się partnerem działu operacyjnego oraz pozostałych stron odpowiedzialnych za stawianie i obsługę infrastruktury informatycznej.

Pojawienie się usług opartych na chmurze wpłynęło korzystnie na wykonalność tej strategii, jako że firma może teraz „wynająć" infrastrukturę IT — serwery, systemy operacyjne, bazy danych i inne prekonfigurowane, gotowe do użycia pakiety — i wyeliminować potrzebę orientowania się w szczegółowych technikaliach infrastrukturalnych, które do tej pory były zrozumiałe jedynie dla doświadczonych architektów IT. Dzięki temu programiści, zwłaszcza full stack, mogą z powodzeniem i w sposób ciągły kodować oraz wdrażać w chmurze gotowe aplikacje front-endowe i back-endowe. W istocie ciągła integracja (ang. *continuous integration* — CI) i ciągłe wdrażanie (ang. *continuous deployment* — CD) leżą u podstaw DevOps.

DevOps kładzie również nacisk na rozwój skryptów wdrażania i monitorowania, które automatyzują wykonywanie nieodzownie związanych z tym zadań. O ile pierwotnie korzystano z samodzielnie tworzonych skryptów bash i opracowywanych w firmach narzędzi, o tyle współcześnie można sięgnąć po szeroki wybór otwartego oprogramowania i komercyjnych narzędzi, które przypominają bardziej aplikacje niż skrypty.

Usługi w chmurze poszerzają zakres swoich ofert i ulepszają narzędzia, aby ułatwić konfigurację serwerów, ich wdrażanie, monitorowanie i zarządzanie nimi, co usprawnia realizację inicjatyw DevOps.

To wszystko brzmi idealnie! Diabeł jednak oczywiście tkwi w licznych szczegółach, do których należy się odnieść. Często lekceważy się fakt, że mało który programista jest zainteresowany byciem dyspozycyjnym przez 24 godziny na dobę przez cały tydzień, a tego właśnie często wymaga się w DevOps. To jeden z istotnych „ludzkich aspektów" DevOps, które również należy uważnie śledzić, aby móc nimi zarządzać.

Choć historie o sukcesach DevOps podkreślają zwiększoną częstotliwość wdrożeń i wyższą jakość oprogramowania, trzeba pamiętać o kosztach. Ponieważ programiści należący do wszystkich omówionych wcześniej kategorii mogą obecnie uczestniczyć we wdrażaniu, monitorowaniu i naprawianiu ewentualnych błędów, stosowanie DevOps może doprowadzić do całkowitego zakłócenia procesu programistycznego, jako że programiści zmuszeni są do „gaszenia pożarów" na bieżąco, dopóki zespół nie przyswoi sobie docelowego rytmu produkcji, wdrażania, monitorowania i naprawiania. Opanowanie DevOps może być czasochłonne, przed dojściem do tego punktu właściwe postępy w produkcji mogą być ograniczone.

Kup książkę Poleć książkę

Ponieważ firma musi być skłonna pogodzić się z przestojami związanymi z problemami pojawiającymi się przy wdrażaniu aplikacji, DevOps wspiera przyjmowanie praktyk takich jak TDD (ang. *test-driven development*) lub automatyczne testowanie i wdrażanie, które prowadzą do uzyskania wyższej jakości i produktywności. Dotyczy to w szczególności aplikacji B2B, gdzie umowy o gwarantowanym poziomie usług z podanymi wymaganiami dotyczącymi czasu bezawaryjnej pracy (np. trzy lub cztery dziewiątki: 99,9% lub 99,99%) sprawiają, że przestoje są nie do zaakceptowania. W takich przypadkach firma musi postępować ostrożnie i przyjąć szeroki zakres sposobów myślenia, zasad i praktyk w ramach przyswajania DevOps.

DevSecOps

DevSecOps (czy też SecDevOps, gdzie na bezpieczeństwo kładzie się jeszcze większy nacisk) podnosi stawkę związaną ze stosowaniem strategii DevOps, wprowadzając podejmowanie decyzji i działań związanych z bezpieczeństwem w tym samym tempie i w tej samej skali, co prace zespołu DevOps. DevSecOps daje zespołom programistycznym możliwość przejęcia kontroli nie tylko nad wydajnością produkcji, ale także nad kwestiami bezpieczeństwa i zgodności.

Wprowadzenie specjalistów ds. bezpieczeństwa w ramach DevOps do wskazania i naprawienia problemów z bezpieczeństwem na wczesnym etapie, kiedy produkt dopiero zaczyna powstawać, może jedynie przynieść korzyści. W najbliższej przyszłości największym wyzwaniem będzie znalezienie wystarczająco wielu wyszkolonych i doświadczonych specjalistów z tej dziedziny.

Choć DevOps i DevSecOps to ważne zagadnienia, nie są to dyscypliny programowania, ani nie wiążą się z pracą określonych typów programistów. Są raczej dodatkowym obowiązkiem każdego programisty. Jeżeli zatem stosujesz DevOps, ważne jest, aby związane z tym obowiązki były wymienione w opisie stanowiska pracy każdego programisty, gdyż zmieniają one tryb ich działania. Programiści DevOps/DevSecOps muszą być gotowi przyjąć na siebie odpowiedzialność za zarządzanie swoim czasem, tak aby w razie potrzeby mogli być dostępni 24 godziny na dobę, 7 dni w tygodniu, by rozwiązywać problemy, które wcześniej mogły leżeć w zakresie obowiązków zespołu obsługi IT. Wykracza to poza oczekiwania, do których programiści byli dotąd przyzwyczajeni.

Rodzaje programistów

Na programistów warto spojrzeć z jeszcze jednej perspektywy przy doborze personelu. W poprzednim podrozdziale skupiliśmy się na klasyfikacji według rodzaju pracy wykonywanej przez programistów (tj. systemy wbudowane, front-end, back-end, bazy danych, tworzenie storn internetowych), ale ważne jest też przyjrzenie się wiedzy technicznej, doświadczeniu i fachowym umiejętnościom, jakie programiści mogą wnieść. Można ich zatem skategoryzować następująco:

Kup książkę Poleć książkę

- inżynierowie/architekci systemów,
- programiści systemów,
- programiści aplikacji,
- nie całkiem programiści.

Inżynierowie/architekci systemów

Inżynierowie/architekci systemów są najbardziej umiejętnymi i doświadczonymi programistami w załodze. Umiejętność zrozumienia złożonych zależności łączących wszystkie istotne komponenty systemu (systemy operacyjne i komunikacyjne, bazy danych, dostęp online i offline, bezpieczeństwo, sprzęt, komunikacja między aplikacjami itd.) wymaga rozległego doświadczenia w dziedzinie różnych technologii i systemów. W rozsądnej wielkości zespole zazwyczaj można znaleźć jednego lub co najwyżej dwóch „prawdziwych" inżynierów lub architektów systemów. Świetny programista tego typu korzystnie wpływa na pracę ogółu. Projektuje niezawodne systemy, które imponują swoją elegancką prostotą.

Gracenote założył właśnie jeden z takich wysoce wykwalifikowanych i doświadczonych inżynierów. Mickey współpracował z nim przez ponad dekadę i był świadkiem tego, jak czystość jego projektów i ich implementacji przekładała się na niezwykły, niezawodny, skalowalny i elastyczny serwis. Innym przykładem jest Dave Cutler, legendarny inżynier, architekt i programista systemów, odpowiedzialny za powstanie Microsoft Windows NT i Azure (oraz wielu innych systemów). Rola Cutlera okazała się kluczowa dla sukcesu Microsoftu. Jak to ujął Steve Ballmer, były prezes Microsoftu: „Nie byłoby dzisiaj Microsoftu, gdyby nie Dave". Są to zaledwie dwa spośród licznych przykładów tego, jak inżynierowie i architekci systemów wnieśli decydujący wkład w budowę cieszących się sukcesami spółek technologicznych.

Programiści systemów

Większość inżynierów/architektów systemów zaczynała jako programiści systemów. Tacy programiści wiedzą, jak działają wszystkie komponenty systemu, w tym systemy operacyjne i komunikacyjne działające na front-endzie i back-endzie. Cytat z Boba Bartona w rozprawie doktorskiej Alana Kaya[2] streszcza pokrótce, jak programiści systemów postrzegani są przez swoje otoczenie:

Programiści systemów są arcykapłanami kultu tego, co u podstaw.

— BOB BARTON[3]

---

[2] Alan Kay jest wpływowym informatykiem, znanym najbardziej jako twórca języka Smalltalk i ze swego wkładu w programowanie obiektowe. Zob. *http://www.britannica.com/biography/Alan-Kay*.

[3] Bob Barton był naczelnym architektem komputera Burroughs 5000 i profesorem na University of Utah, gdzie Alan Kay napisał swoją rozprawę i uzyskał doktorat. Zob. *http://www.computer.org/profiles/robert-barton*.

Kup książkę Poleć książkę

Programiści systemów piszą sterowniki urządzeń, które łączą się ze sprzętem, tworzą systemy operacyjne, zapewniające środowiska uruchomieniowe, w których działają sterowniki i aplikacje, tworzą kompilatory i narzędzia do debugowania dla innych programistów i ogólnie zapewniają narzędzia oraz usługi, których inni programiści używają do tworzenia programów.

W przeszłości bycie nazwanym programistą systemów było traktowane przez ludzi o choćby minimalnych kompetencjach społecznych niemal jak zniewaga. Charakterystyczne koszulki i nastawienie wielu programistów systemów, których mieliśmy okazję poznać, były całkowicie zgodne z ikonicznym już, geekowym stylem, który z biegiem czasu stał się modny. Deklaracja „byłem geekiem, zanim to stało się modne" przychodzi nam na myśl, kiedy myślimy o tych ludziach. (Swoją drogą, obaj też jesteśmy programistami systemów).

Programiści aplikacji

Programiści aplikacji są programistami front-endowymi i back-endowymi i stanowią lwią część ogółu profesjonalnych programistów, studentów i hobbystów, którzy za programistów się uważają. Programiści aplikacji zazwyczaj tworzą programy bezpośrednio wykorzystywane przez użytkowników końcowych lub zwracające rezultaty, którymi użytkownicy się posługują. Przykłady takich aplikacji to edytory tekstu, arkusze kalkulacyjne, kalendarze, przeglądarki Chrome i Firefox, odtwarzacze w rodzaju iTunes, komunikatory typu Zoom i Slack, a także aplikacje i gry na iOS oraz Androida. Także programiści baz danych mogą tworzyć aplikacje do wykonywania konkretnych funkcji na danych pobranych z bazy lub w niej przechowywanych. Przykładami aplikacji bazodanowych są pakiety do analizy danych finansowych, systemy rezerwacji lotów i narzędzia do eksploracji danych takie jak Oracle Financials.

Niektórzy programiści aplikacji potrafią wznieść się ponad praktyczne szczegóły kodu, aby zrozumieć użytkowników aplikacji i docenić niuanse projektowania wizualnego oraz interaktywnego. Tacy programiści mają najlepsze predyspozycje, aby zająć się programowaniem interfejsów użytkownika. Kiedy połączy się w parę takiego utalentowanego programistę z projektantem UX, znającym się prawdopodobnie na projektowaniu graficznym, czynnikach ludzkich i psychologii poznawczej, uzyskane rezultaty zazwyczaj przerastają to, co każdy z nich mógłby osiągnąć samodzielnie.

Niektóre projekty, takie jak Mac Finder (interfejs użytkownika macOS), są do tego stopnia ukierunkowane na UX, że całe pracujące nad nimi zespoły muszą się składać z tego typu uzdolnionych programistów. Kiedy Ron zarządzał zespołem Mac Finder w Apple, poszukiwał kandydatów i odbywał z nimi dogłębne rozmowy próbne, aby znaleźć osoby, które poza znakomitymi umiejętnościami programistycznymi odznaczały się wysokim poziomem empatii w stosunku do użytkownika. „Same umiejętności związane z kodowaniem nie gwarantowały programistom powodzenia w tym zespole" — wspomina.

Kup książkę Poleć książkę

Nie całkiem programiści

Wielu pracowników technicznych zespołów programistycznych nazywa się wprawdzie programistami, ale w rzeczywistości nie do końca nimi są. Niektórzy określają logikę programową lub biznesową przy użyciu narzędzi graficznego interfejsu użytkownika (GUI), które generują aplikacje dostępne dla użytkowników. Inni piszą skrypty lub modyfikują pliki konfiguracyjne, co skutkuje spersonalizowanym widokiem treści. Głównym czynnikiem odróżniającym tych „programistów" od reszty jest to, że używają zaprogramowanych narzędzi i aplikacji, a nie zajmują się bezpośrednio pisaniem kodu.

Tacy „programiści" są ważni i cenni, lecz na ogół nie mogą się mierzyć pod względem umiejętności technicznych z pozostałymi kategoriami, które opisaliśmy. Na świecie wraz z rozwojem i wzrostem efektywności wspomnianych gotowych narzędzi pojawia się coraz więcej tego typu programistów, niemniej jednak w tej książce nie będziemy się zajmować tą grupą.

Wiele z opisanych tutaj technik zarządzania programistami można odnieść także do wskazanych tutaj „nie całkiem programistów", ale z doświadczenia wiemy, że wielu z nich czuje się w swojej pracy dość komfortowo. Większość z nich nie jest tak chętna do nauki jak „prawdziwi" programiści i charakteryzuje się niższym poziomem motywacji.

Specjalizacja dziedzinowa

Programiści różnią się od siebie także pod względem doświadczenia w dziedzinie lub branży, w jakiej działa Twoja organizacja.

Zauważyliśmy, że nacisk kładziony na doświadczenie tego typu w opisach i wymaganiach dotyczących stanowiska pracy różni się w zależności od stanu gospodarki. W prężnych gospodarkach organizacje poszukują pracowników technicznych (a także kierowników) o szerokiej specjalizacji dziedzinowej, licząc na to, że będzie to sprzyjać nieszablonowemu myśleniu, interdyscyplinarności, a także wymianie podejść i praktyk, które sprawdzają się w innych dziedzinach. Kiedy Ron znalazł zatrudnienie w branży usług IT w dziedzinie finansów, zaangażowano go jako specjalistę zajmującego się produktami konsumenckimi, jako że jego zespoły opracowywały w tej dziedzinie wysoce interaktywne doświadczenia (macOS) oraz produkty rozrywkowe (gry i narzędzia multimedialne). Ponieważ internet był w roku 1996 nowością, Schwab potrzebował specjalisty technologicznego z zewnątrz, który pokierowałby zespołem tworzącym wysoce interaktywne narzędzia webowe dla inwestorów.

W słabych warunkach gospodarczych organizacje z kolei ograniczają się, tną koszty, starają się napędzać wydajność i ograniczają prace programistyczne do głównej funkcjonalności dziedzinowej. Mają tym samym skłonność szukać przez lata ekspertów dziedzinowych, aby ograniczyć poziom ryzyka towarzyszącego zatrudnianiu nowych pracowników.

Kup książkę Poleć książkę

Niezależnie od stanu gospodarki każdy zespół może zyskać na zestawieniu znakomitych kompetencji programistycznych ze specjalizacją dziedzinową, analizą i komunikacją techniczną. Specjalizacja dziedzinowa jest ważnym czynnikiem, który warto uwzględnić przy zatrudnianiu zadowalającego zespołu programistów.

Wymagania i umiejętności

Udane zatrudnienie programisty i sprawne zarządzanie nim zaczyna się od zrozumienia, że każdy programista ma wyjątkowe umiejętności. To tak jak z płatkami śniegu — nie ma dwóch takich samych pracowników. My sami, podobnie jak wielu innych, często powtarzamy, że programiści pod względem umiejętności „tłuczenia kodu" potrafią się od siebie różnić o rząd wielkości. Z czego wynika taka rozbieżność? Odpowiada za to kilka czynników, w tym wykształcenie, doświadczenie, talent i intuicja, a także wiele innych nienamacalnych właściwości.

Większość programistów intuicyjnie pojmuje różnice dzielące ich kolegów po fachu; nie potrzebują jasno określonych rang ani tytułów roboczych. Niemniej jednak opracowanie sformalizowanego zestawu wytycznych dotyczących drabiny rozwoju zawodowego — typów i poziomów programistów (tabela 2.1) — z minimalnymi wymaganiami i umiejętnościami znacznie ułatwi Ci zarządzanie, pomoże zespołom i kierownikom projektów wskazywać najlepiej nadających się do wybranych zadań i projektów pracowników, a także zapewni starszemu kierownictwu lepszy wgląd w organizację i jej strukturę.

Tabela 2.1. *Wytyczne dotyczące poziomów front-endowych programistów*

Poziomy programistów	Programiści front-endowi
Początkujący	III programista
1 – 5 lat doświadczenia	II programista
Doświadczony (5 – 10 lat)	I programista
Doświadczony (10 – 20 lat)	II starszy programista
Zaawansowany (20+ lat)	I starszy programista/architekt

Z każdym z tych poziomów programistycznych[4] wiąże się zbiór kryteriów, które każdy programista musi spełnić, aby być na danym poziomie zatrudniony lub do niego awansowany. Rzecz jasna, przepracowanie danego stażu nie jest obowiązkowe, lecz służy jako ogólny wskaźnik doświadczenia oczekiwanego na poszczególnych poziomach. Umiejętności i doświadczenie każdego programisty są wyjątkowe, a celem wytycznych nie jest uniemożliwienie awansowania utalentowanym i doświadczonym pracownikom tylko

[4] Wymienione tutaj poziomy są zgodne z danymi uzyskanymi z większości serwisów oferujących informacje o zarobkach, takich jak Radford Surveys i Salary.com. Znajdują się w nich porównawcze i bezwzględne dane dotyczące zarobków, które okazują się nieocenione przy zarządzaniu zespołem. Harmonizując poziomy programistyczne z danymi o zarobkach używanymi w Twojej firmie, zyskujesz pewność, że będziesz posługiwać się najlepszymi dostępnymi informacjami na potrzeby zarządzania.

Kup książkę Poleć książkę

dlatego, że nie przepracowali określonej liczby lat. Ostatecznie programistów należy oceniać nie na podstawie tego, czym mogą się pochwalić, lecz co rzeczywiście wnoszą od siebie.

Ron i Mickey zetknęli się z sytuacjami, gdy najlepszym programistą wcale nie był najbardziej doświadczony ani ten o najwyższej pensji. Zachęcamy Cię, aby nie dostrzegać w takich sytuacjach problemu, lecz raczej okazję — okazję, by nagrodzić wybitnego programistę wyraźnie wyższą pensją lub bonusami. Korzystanie z poziomów programistów daje Ci możliwość sprawniejszego zawalczenia o to. Rzadko spotykaliśmy się ze sprzeciwem ze strony kadry zarządzającej, kiedy nawoływaliśmy do odpowiedniego wynagradzania zdecydowanie wyróżniających się programistów, choć Twoje działania mogą też mieć wpływ na to, co dzieje się z tymi, którzy nie przynoszą oczekiwanych rezultatów.

Tabela 2.2 przedstawia, w jaki sposób można wdrożyć stosowanie poziomów programowania w odniesieniu do opisanych wcześniej kategorii programistów.

Tabela 2.2. *Wytyczne ds. poziomów programistów*

Programiści front-endowi	Programiści back-endowi	Programiści baz danych
III programista front-endowy	III programista back-endowy	III programista baz danych
II programista front-endowy	II programista back-endowy	II programista baz danych
I programista front-endowy	I programista back-endowy	I programista baz danych
II starszy programista front-endowy	II starszy programista back-endowy	II starszy programista baz danych
I starszy programista/architekt front-endowy	I starszy programista/architekt back-endowy	I starszy programista/architekt baz danych

Istotne jest określenie zbioru kryteriów, rozwijającego się wraz z przechodzeniem programisty na kolejne poziomy. Tabela 2.3 pokazuje przykład dla programistów front-endowych.

Tabela 2.3. *Programiści front-endowi — kryteria poziomowe*

III programista (początkujący)	II starszy programista
Znajomość Windows 7/10, macOS lub Linuksa	Opracował dwie lub więcej komercyjnych aplikacji lub technologii
Umiejętność wytwarzania aplikacji na iOS lub Androida	Biegła umiejętność posługiwania się dwiema platformami (np. macOS i iOS)
Podstawowa znajomość dobrych praktyk kodowania	Znajomość zagadnień związanych z wieloplatformowością
Świadomość/zainteresowanie technologiami webowymi, m.in. HTML, CSS i JavaScript	Rozległa znajomość technologii webowych, m.in. HTML, CSS i JavaScript
Świadomość/zainteresowanie technologiami baz danych	Rozległa znajomość technologii baz danych
Znajomość C/C++, Objective C, Swift lub Javy	Rozległa znajomość C/C++, Objective C, Swift lub Javy
Umiejętność pracy w zespole i stosowania się do poleceń	Bardzo dobre umiejętności komunikacyjne

Kup książkę Poleć książkę

Tabela 2.3. *Programiści front-endowi — kryteria poziomowe (ciąg dalszy)*

III programista (początkujący)	**II starszy programista**
Umiejętność planowania zadań z przełożonym	Motywacja wewnętrzna, wymagany minimalny nadzór
	Znakomite umiejętności analizy, planowania projektu i oszacowywania terminów
	Wspiera programistów i wspomaga ich rozwój, zapewniając porady i instruktaż
	Zwraca uwagę na zmieniające się warunki i planuje działania adaptacyjne
II programista (podstawowe doświadczenie)	**I starszy programista**
Opracował jedną lub więcej komercyjnych aplikacji	Opracował dwie lub więcej złożonych, komercyjnych aplikacji lub technologii
Biegła umiejętność posługiwania się Windows 7/10, macOS lub Linuksem	Wnikliwa znajomość dwóch lub więcej platform (np. macOS, iOS i Android)
Umiejętność (biegła) wytwarzania aplikacji na iOS lub Androida	Znajomość zagadnień związanych z wieloplatformowością i umiejętność odniesienia się do nich
Stosowanie dobrych praktyk kodowania	Doskonała znajomość technologii webowych, m.in. HTML, CSS i JavaScript
Obeznanie w technologiach webowych, m.in. HTML, CSS i JavaScript	Rozległa znajomość technologii baz danych
Obeznanie w technologiach baz danych	Ekspercka znajomość C/C++, Objective C, Swift lub Javy
Solidna znajomość C/C++, Objective C, Swift lub Javy	Ekspercka znajomość praktyk projektowania oprogramowania
Motywacja wewnętrzna i umiejętność stosowania się do poleceń	Bardzo dobre umiejętności komunikacyjne, kontakty branżowe
Umiejętność samodzielnego planowania zadań	Motywacja wewnętrzna, pełna niezależność
	Znakomite umiejętności analizy, planowania projektu i oszacowywania terminów
	Generowanie, ulepszanie i promowanie nowych koncepcji
	Wspiera programistów i wspomaga ich rozwój, zapewniając porady, instruktaż i możliwości kształcenia
	Zwraca uwagę na zmieniające się warunki i planuje działania adaptacyjne

Kup książkę Poleć książkę

Tabela 2.3. *Programiści front-endowi — kryteria poziomowe (ciąg dalszy)*

I programista (doświadczony)
Opracował dwie lub więcej komercyjnych aplikacji
Biegła umiejętność posługiwania się dwiema platformami (np. macOS i iOS)
Posługiwanie się technologiami webowymi, m.in. HTML, CSS i JavaScript
Posługiwanie się technologiami baz danych
Sprawne posługiwanie się C/C++, Objective C, Swift lub Javą
Dobre umiejętności komunikacyjne
Motywacja wewnętrzna, wymagany minimalny nadzór
Dobre umiejętności planowania projektu i oszacowywania terminów
Rozpoznawanie problemów i pomaganie zespołom w adaptacji

Taki opis poziomów programistów oczywiście powinien być szczegółowym opisem każdego ze stanowisk pracy. Programiści słyną z bycia wolnymi ptakami i pogardy dla formalnych dokumentów. Z doświadczenia jednak wiemy, że zdecydowanie zależy im na charakterystyce pracy i jasnym określeniu warunków, które muszą spełnić, aby piąć się po drabinie organizacyjnej. Są od tego wyjątki, ale przeważająca większość programistów pracuje lepiej, kiedy taki system istnieje i rozumie, jak Ty lub inni kierownicy postrzegacie ich rolę w organizacji.

Sporządzanie szczegółowych opisów stanowisk pracy jest bardzo trudnym zadaniem. W ciągu ponad 15 lat Mickey wypracował ustrukturyzowany zestaw opisów odzwierciedlających omówione wcześniej poziomy programistów. Podstawowy format tych opisów możesz zobaczyć w przykładzie na rysunku 2.1.

Kup książkę Poleć książkę

STANOWISKO:	III programista
DZIAŁ:	programowanie front-endowe
PODLEGA:	kierownikowi ds. programowania front-endowego
WYMIAR ZATRUDNIENIA:	etat
LOKALIZACJA:	Szczecin

Opis stanowiska: Stanowisko podstawowe. Odpowiedzialny za zarządzanie, konwersję, weryfikację i konserwację kodu i/lub zasobów. Odpowiedzialny za pisanie jasno określonych części kodu źródłowego z przestrzeganiem ustanowionych standardów jakości dokumentacji i kodu. Pracuje dobrze w grupie i wykonuje polecenia kierownika oraz starszych członków zespołu. Oczekuje się pracy pod bezpośrednim nadzorem i informowania o pojawiających się problemach.

Wymagania

- Ukończone studia informatyczne I stopnia lub porównywalne doświadczenie
- Znajomość Windows, macOS lub Linuksa, w miarę możliwości więcej niż jednej platformy
- Znajomość C/C++, Objective C, Swift lub Javy i technik debugowania
- Podstawowa znajomość dobrych praktyk pisania kodu i podstawowych zagadnień informatycznych
- Zainteresowanie technologiami webowymi, m.in. HTML, CSS i JavaScript
- Zainteresowanie metodologią projektowania baz danych i systemami baz danych
- Umiejętność pracy w zespole i stosowania się do poleceń
- Motywacja wewnętrzna i umiejętność właściwego reagowania na wskazówki przełożonych; zadawanie celnych pytań
- Pozytywne nastawienie do działalności firmy i tworzenia firmowych produktów
- Umiejętność wspólnego planowania zadań z przełożonym i oszacowywania terminów ukończenia
- Umiejętność poznawania nowych technologii, języków programowania, platform, narzędzi i środowisk programistycznych
- Umiejętność dostosowywania się do zmiennych warunków

Rysunek 2.1. Przykładowy opis stanowiska pracy

Jak widać w powyższym przykładzie, opis stanowiska składa się z trzech części:

- podstawowych informacji (nazwa stanowiska, dział, przełożeni, wymiar zatrudnienia, lokalizacja);
- narracyjnego opisu, obejmującego obowiązki i oczekiwania;
- minimalnych wymagań wobec kandydata.

Łatwo przystosować ten format na potrzeby opisu niemal dowolnego stanowiska, ale stanowczo zachęcamy Cię do sporządzania nie pojedynczych opisów, lecz całych ich serii, określających różne poziomy kompetencji. Napisanie zestawu opisów zajmuje nieznacznie więcej czasu niż napisanie jednego, a pozwala to z łatwością odnieść się do pytań dotyczących rozwoju zawodowego i ścieżki awansu. Poświęcenie na to kilku dodatkowych minut z góry pozwala na oszczędzenie sobie wielu godzin pracy na późniejszym etapie.

Kup książkę Poleć książkę

Inni autorzy również zwracają uwagę na znaczenie hierarchii zawodowej. Ich dorobek i doświadczenie mogą posłużyć za dodatkowe źródła informacji o korzyściach, jakie organizacje odniosły z wprowadzenia takich hierarchii zespołów technicznych[5].

Zauważ, że takie hierarchie zawodowe z reguły nazywamy *wytycznymi*, ponieważ stosowanie się bez wyjątków do sztywnych struktur może być przyczyną problemów. Zdarzają się sytuacje, że wyraźnie utalentowanemu programiście należy się awans, nawet jeśli brak mu lat stażu lub konkretnych umiejętności technicznych wskazanych w opisie. Takie działania należy jednak podejmować w drodze wyjątku i jedynie wtedy, kiedy można je dostatecznie uzasadnić.

Tworząc opisy stanowisk określające ścieżkę rozwoju zawodowego w odniesieniu do podnoszenia kompetencji i zwiększania odpowiedzialności, będziesz mógł stać się bohaterem firmowego działu HR, a także zyskasz świetne narzędzie do zarządzania programistami.

Lokalizacja i relacje

Zarządzanie wytwarzaniem oprogramowania w ciągu minionych lat stało się zdecydowanie bardziej skomplikowane. Jeśli miałeś szczęście, Twoje życie wciąż jest w miarę proste, ale ten stan szybko przeminie. Jeżeli jeszcze nie odczuwasz skutków przemian, to prędzej czy później zaczniesz. W przeciwnym wypadku będzie to oznaczać, że Twoja firma nie jest konkurencyjna.

W czasach minionych zadawano tylko jedno pytanie: „Skąd wezmę programistę do projektu?". Należało następnie zdecydować, czy zatrudnić pełnoetatowego pracownika na miejscu, czy zlecić pracę podwykonawcy. Współcześnie o tym, gdzie i jak pracujesz, decyduje wiele wyborów, którym należy się uważnie przyjrzeć.

Są to zazwyczaj decyzje, których sam nie podejmujesz. Mogą wynikać z postanowień zarządu lub okoliczności, w jakich odbywa się praca nad projektem. Tak czy inaczej, musisz sprawnie zarządzać różnorodnością geograficzną i relacyjną, aby odnieść sukces.

Pod względem lokalizacji i relacji z firmą istnieje pięć rodzajów programistów, którymi będziesz musiał zarządzać:

- pracownicy wewnętrzni,
- pracownicy zdalni,
- podwykonawcy,
- zarządzane zespoły podwykonawców,
- firmy outsourcingowe.

Możesz potraktować to jako ranking siły więzi. Im więź jest bliższa (pracownicy wewnętrzni są najsilniej związani z firmą), tym jaśniejszy masz ogląd pracy programisty i większą

[5] Prateek Jain, *How We Designed an Effective Career Ladder for Engineers*, *https://www.axelerant.com/resources/articles/how-to-design-an-effective-career-ladder-for-engineers*; Andy Worsley, *A Software Engineering Career Path*, 16 listopada 2017, *http://www.andyworsley.com/2017/11/16/a-software-engineering-career-path*.

Kup książkę Poleć książkę

kontrolę nad nią. Im więź jest luźniejsza (najsłabiej związane są firmy outsourcingowe), tym większa jest konieczność zarządzania pośredniego, często z wykorzystaniem specyfikacji i sprawozdań, bez możliwości bezpośredniego zarządzania ludźmi.

Pracownicy wewnętrzni

Większość opisanych w tym rozdziale narzędzi służy do zarządzania pracownikami wewnętrznymi. Zatrudnienie takiego pracownika jest jednoznacznym zobowiązaniem do zapewnienia mu, poza samym wynagrodzeniem, określonych korzyści oraz środowiska pracy w zamian za pracę utrzymaną na zadowalającym poziomie. To ostatecznie umowa.

W owej umowie zawarte są jednak pewne milczące założenia, przynajmniej w naszym doświadczeniu. Pracownikom firmowym zależy nie tylko na zatrudnieniu, lecz i na karierze. Musisz zatem pamiętać o zapewnieniu im:

- możliwości rozwoju zawodowego i okazji temu sprzyjających;
- regularnych informacji zwrotnych;
- informacji o wydarzeniach i sytuacji w firmie;
- niemal nieskończonej liczby innych korzyści, o których dowiesz się z biegiem czasu.

Zyskujesz jednak okazję do nawiązania bliskiej relacji z takim pracownikiem, co minimalizuje ilość komunikacji potrzebnej do sprawnego zarządzania.

Pracownicy zdalni

Pracownikami zdalnymi z gruntu zarządza się tak samo jak wewnętrznymi. Oczekiwania są niezmiennie w mocy, a pracownicy zdalni są zobowiązani do dostarczania rezultatów. Niemniej jednak zarządzanie pracownikiem, którego nie widuje się często lub z którymi nie ma się regularnej styczności na żywo, wiąże się z większym obciążeniem związanym z komunikowaniem się z nim.

Jeden z założycieli Evans & Sutherland (E&S) powiedział kiedyś Mickeyowi, jakie może przyjąć podejście do sprawnej komunikacji interpersonalnej:

Komunikacja jest w skali kraju funkcją odwrotności kwadratu odległości,
a w skali międzynarodowej funkcją odwrotności sześcianu odległości.

— DAVID C. EVANS

Z doświadczenia wiemy, że ta reguła sprawdza się, i to także po wprowadzeniu takich narzędzi jak IRC lub obecnie Slack. Widzieliśmy aż za dużo przypadków załamania komunikacji przez to, że ktoś nie siedział obok. Im bardziej ludzie są od siebie oddaleni (na drugim końcu korytarza, w budynku obok, w sąsiedniej strefie czasowej, na drugim końcu świata), tym bardziej problem ten się pogłębia. Komunikacja często jest dodatkowo utrudniona przez rozbieżności czasowe, odmienne języki i akcenty oraz różnice kulturowe.

Kup książkę Poleć książkę

Zanim więc postanowisz zatrudnić oddalonego geograficznie pracownika — lub utrzymać współpracę z pracownikiem przeprowadzającym się do innej miejscowości lub za granicę — musisz uświadomić sobie milczące założenie, że jesteś zobowiązany do zapewnienia przekazu informacji na adekwatnym poziomie. To trudniejsze, niż się spodziewasz, nawet jeśli czujesz, że nie będzie łatwo.

Innym problemem wiążącym się z pracownikami zdalnymi jest to, że niekiedy mogą oni podejmować pracę jedynie nad ograniczonym zakresem typów projektów. Twoja firma lub organizacja może wprawdzie mieć nieograniczony przegląd projektów, nad którymi programista może pracować w samotności, ale w rzeczywistości rzadko tak bywa. Jedynie szczególnego rodzaju programiści potrafią wznieść się ponad ograniczenia czasowe i przestrzenne, aby skutecznie dostarczać rezultaty w ścisłej współpracy z zespołem oddalonych fizycznie współpracowników. Upewnij się, czy dany człowiek jest właśnie taką osobą lub czy masz odpowiedni zapas projektów możliwych do zrealizowania w pojedynkę, zanim zdecydujesz się na umożliwienie mu pracy zdalnej lub zatrudnisz go w roli pracownika zdalnego.

Podwykonawcy

Decyzji o zatrudnieniu podwykonawcy zamiast pełnoetatowego pracownika nie należy traktować lekko. Często to sytuacja stanowi o tej decyzji. Jeżeli dane zadanie należy do kategorii tych, które można ukończyć, lub brakuje pracownika, który mógłby się tym zająć, najprostszym rozwiązaniem jest zatrudnienie podwykonawcy.

Podwykonawcy są z definicji „najemnikami", którym zleca się wykonanie jakiegoś zadania w zamian za wynagrodzenie. Podwykonawca nie powinien mieć milczących oczekiwań. Wszelkie oczekiwania *koniecznie* trzeba wymienić w umowie zawartej z podwykonawcą przed rozpoczęciem prac.

Taka relacja między firmą a podwykonawcą należy zazwyczaj do najprostszych, jakie tutaj omawiamy. Jest tak, ponieważ umowę można rozwiązać w dowolnej chwili bez podania powodu i uprzedzenia[6]. Nie oznacza to jednak, że ta relacja nie może się skomplikować. Jeśli tak się stanie, to bez wątpienia nastąpi to z Twojej winy. To Ty ustalasz zasady.

W jaki sposób taka prosta relacja może się skomplikować? Często wynika to z tego, że w rzeczywistości nie starasz się znaleźć podwykonawcy do wykonania konkretnego zadania, co jest klasyczną definicją zlecenia. Nagminne są sytuacje, w których pracodawca szuka pełnoetatowego pracownika i albo nie może znaleźć kogoś odpowiedniego, albo nie jest pewien, czy trafił na właściwą osobę, wobec czego zleca pracę podwykonawcy, lecz zaczyna go traktować jak pracownika. Prowadzi to do pojawienia się niewypowiedzianych oczekiwań właściwych dla pełnoetatowego pracownika, których spełnienie bywało nakazywane na drodze sądowej. Co za tym idzie, w zarządzanych przez nas firmach działy HR i prawne jasno dawały do zrozumienia menedżerom, by nie zapewniali podwykonawcom dodatkowych

[6] Zakładamy tutaj, że w umowie z podwykonawcą zawarto klauzulę uprawniającą do rozwiązania jej na warunkach firmy, bez podania powodu i powiadomienia z wyprzedzeniem. W każdej dobrej umowie z podwykonawcą takie zastrzeżenie powinno się znaleźć.

Kup książkę Poleć książkę

świadczeń — czy to koszulek, czy wynagrodzenia za czas na służbie spędzony poza miejscem pracy.

Bądź zatem ostrożny w określaniu, czego podwykonawca może od Ciebie oczekiwać, i w tym, jak go traktujesz. Traktuj go jak najemnika i nie czuj się źle z tym, że ma inny status niż pracownicy — on nie jest pracownikiem.

Zarządzane zespoły podwykonawców i firmy outsourcingowe

Uważamy, że liniowi kierownicy zespołów programistycznych nie powinni być obarczeni obowiązkami związanymi z outsourcingiem, więc pomijamy w książce te kwestie. Outsourcing wymaga szczególnej uwagi i umiejętności. Jeżeli mierzysz się z koniecznością zlecenia obcej firmie pracy nad częścią projektu, podejmij się tego, *wyłącznie jeśli* możesz uzyskać pomoc i wsparcie ze strony kogoś, kto ma doświadczenie w zarządzaniu tego typu personelem i zespołami.

Zarządzanie zasobami z outsourcingu jest samo w sobie pełnoetatowym zajęciem, ale w rozdziale 5. omówimy wyzwania towarzyszące uzyskiwaniu wartości dodanej od zagranicznych podwykonawców.

Charakter pokoleniowy

Różnice pokoleniowe zawsze były obecne wśród inżynierów i programistów, lecz współcześnie mają one szczególny wpływ na atmosferę w miejscu pracy. Istnieją teraz cztery odmienne pokolenia, które muszą sprawnie ze sobą współpracować, aby inżynierowie i programiści tworzyli sprawne zespoły. Mają one jednak odmienne wartości i poglądy, a ponadto różne czynniki motywujące.

Przydatne jest określenie nie tylko Twojego własnego stylu pracy, ale także osób, które zatrudniasz i którymi zarządzasz. Jednym ze sposobów jest spojrzenie na to pokoleniowo. Sęk w tym, że nie ma szerokiego konsensusu co do zakreślania granic między pokoleniami. Tradycyjnie dzielono pokolenia według roczników:

- boomerzy (urodzeni w latach 1946 – 1963), czyli pokolenie powojennego wyżu demograficznego;
- pokolenie X (urodzeni w latach 1964 – 1985);
- milenialsi (urodzeni w latach 1986 – 2000), nazywani też pokoleniem Y;
- pokolenie Z (urodzeni w okresie od 2001 roku) jest grupą demograficzną następującą po milenialsach; nie można wyraźnie wskazać pierwszego rocznika, ale 2001 jest mniej więcej odpowiednią datą, przy czym jednocześnie brakuje jednoznacznej zgody co do tego, który rocznik jest ostatni.

W tabeli 2.4 znajduje się krótka charakterystyka tych pokoleń.

Kup książkę Poleć książkę

Skupienie się na datach nie oddaje w pełni podziału pokoleniowego; o „prawdziwym" wieku nie decyduje data urodzenia, ale także sposób myślenia. Wszyscy zetknęliśmy się z ludźmi, którzy zachowują się inaczej, niż wskazywałby ich wiek. Jedni wydają się dojrzali, a inni niedojrzali jak na swoje lata, więc kategoryzowanie ludzi według samych roczników jest ryzykowne. Ponadto poszczególne pokolenia przyswajają sobie różne aspekty tych pokoleń, które je poprzedzają lub po nich następują, co dodatkowo zaciera granice.

To, z czym się wychowujesz — muzyka, media, technologie, mentalność Twoich rodziców i ich pokolenia — często nieodwracalnie wpływa na cechy i fundamentalne wartości determinujące to, kim jesteś i jakim torem biegną Twoje myśli. W każdym pokoleniu jednak można dostrzec pewne cechy i wartości wspólne dla wszystkich jego przedstawicieli, niezależnie od ich rzeczywistego wieku.

Najważniejszym wnioskiem wynikającym z tej tabeli jest to, że w każdym pokoleniu występują swoiste czynniki, które kształtują jego przedstawicieli w toku ich wychowania i, co ważniejsze, odmienne fundamentalne wartości, za sprawą których ludzie różnie funkcjonują w pracy.

- Boomerzy skłaniają się do optymizmu i lojalności, pragnąc stabilnej kariery. Wielu z nich jest pracoholikami skłonnymi robić więcej, niż jest to wymagane, aby pomóc swojej firmie i rozwinąć się zawodowo.
- Przedstawiciele pokolenia X mają tendencję do cynizmu i cechują się niezależnością, skłaniającą ich do cenienia sobie postawy „samotnego wilka" bardziej od lojalności wobec firmy. Postrzegają czas jako dobro, którego nie chcą się pozbywać, ani się nim dzielić.
- Milenialsi mają skłonność do bycia zorientowanymi grupowo indywidualistami. Ambitni i pewni siebie, odznaczają się wspólnym poczuciem optymizmu, śmiałością i duchem heroizmu, czyli cechami pogłębionymi za sprawą atmosfery jedności w obliczu tragedii 11 września. Podobnie jak przedstawiciele pokolenia X, milenialsi postrzegają czas jako dobro, z którym nie chcą się rozstawać ani którym nie chcą się dzielić — praca jest dla nich tym, co robią w przerwach między kolejnymi weekendami.
- Przedstawiciele pokolenia Z dopiero wkraczają na rynek pracy, ale mają już duży wkład jako pierwsi prawdziwi „cyfrowi obywatele świata", których idee i wartości zwiastują przyszłe trendy i kształt światowej gospodarki. Odznaczają się większym tradycjonalizmem niż milenialsi.

Od razu widać, jak rozbieżności fundamentalnych wartości mogą prowadzić do chaosu w biurze, gdzie praca musi być rozdzielana równo. Choć różnice pokoleniowe istnieją, można też dostrzec różnice dzielące przedstawicieli poszczególnych pokoleń. Umiejętność określenia i dostosowania swojego podejścia do tych skrajnie odmiennych pokoleń i jednostek jest ważna dla Twojego powodzenia w roli kierownika.

Kup książkę Poleć książkę

Kup książkę

Tabela 2.4. *Różnice pokoleniowe*

Pokolenie*	Lata urodzenia	Muzyka	Media	Technologie†	Cechy†	Fundamentalne wartości§
Starsi boomerzy	1946 – 1955	Płyty winylowe	Radio AM, telewizja, filmy, gazety	Analogowe (np. gitary elektryczne), telefony, samochody, poczta, aparaty analogowe, planszówki	Chętnie korzystają z technologii, ale często jedynie do kontaktowania się z krewnymi i znajomymi.	Bunt przeciwko konformizmowi i życiowy perfekcjonizm oparty na wartościach osobistych i rozwoju duchowym.
Młodsi boomerzy	1956 – 1963	Kasety	Radio FM, telewizja, kablówka, filmy, gazety	Komputery, faks, e-mail, aparaty analogowe, gry komputerowe	Korzystają bez skrępowania z internetu, mediów społecznościowych i urządzeń mobilnych; przyswajają sobie nowe technologie, ale rzadko z zapałem.	Otwartość na pracę zespołową, stabilne kariery cechujące się lojalnością w stosunku do pracodawcy.
Pokolenie X	1964 – 1985	Płyty CD	Kablówka, filmy, witryny internetowe, filmy online	Komputery, internet, e-mail, komunikatory, aparaty cyfrowe, czaty, gry FPS i MMORPG	Uwielbiają technologie, które zwiększają ich niezależność, oraz cyfrowe zasoby, które wpływają korzystnie na ich życie.	Mentalność gospodarczych i psychicznych „robinsonów”; są sceptyczni wobec autorytetów i ostrożnie nastawieni do przyjmowania zobowiązań. Ambitni i niezależni, obecnie dążą do znalezienia równowagi między konkurującymi sferami życia zawodowego, rodzinnego i osobistego.

Poleć książkę

Kup książkę

Tabela 2.4. *Różnice pokoleniowe (ciąg dalszy)*

Pokolenie*	Lata urodzenia	Muzyka	Media	Technologie†	Cechy†	Fundamentalne wartości§
Milenialsi, pokolenie Y	1986 – 2000	iPod, Pandora	Telewizja over-the-top, witryny internetowe, filmy, filmy online, YouTube, Facebook Twitter	Telefony komórkowe, komunikatory, Facebook, Twitter, aparaty w telefonach, aplikacje do gier, carpooling	Ich główną cechą jest korzystanie z urządzeń mobilnych: pisanie wiadomości tekstowych, planowanie imprez na bieżąco, przechowywanie danych tożsamościowych na telefonach. Można oczekiwać od nich szybkich, zwięzłych odpowiedzi.	Dojrzawszy w okresie cechującym się zwrotem ku wartościom, dryfują ku organizacjom, których deklaracje misji odnoszą się do spraw donioślejszych niż wyniki finansowe i preferują pracę w środowiskach nastawionych na współdziałanie. Oczekują ciągłych informacji zwrotnych, a mają skłonność do skupiania się na sobie oraz zależności od innych. Są obeznani technicznie i mają pozytywne, ambitne nastawienie, mówiące „Przybyłem tutaj, aby wpłynąć na otaczający mnie świat".
Pokolenie Z	2001 – 2014	Smartfony, serwisy streamingowe	Telewizja over-the-top, YouTube, Instagram, Snapchat, TikTok	Telefony komórkowe, komunikatory, Facetime, aparaty w telefonach, smartwatche, Siri, Alexa, aplikacje do wszystkiego	Ich główną cechą jest stałe łączenie się ze światem cyfrowym przez smartfon: Instagram, aplikacje randkowe.	Pokolenie Z składa się z cyfrowych tubylców, którzy dorastali w czasach niepewności. Dzieci pokolenia X są też pierwszym „prawdziwie globalnym pokoleniem" i niemal powszechnie czują się komfortowo, korzystając z technologii i mediów społecznościowych. Mają skłonność do większej samoświadomości i samodzielności.

* Nie ma większej zgody co do cech definiujących charakter pokoleń. Opisy w tej tabeli odzwierciedlają te właściwości, które wydają się najistotniejsze z perspektywy tematyki tego rozdziału.

† *State of Consumers and Technology: Benchmark*, Forrester Research, 2009 i 2017.

§ Dan King, *Defining a Generation: Tips for Uniting Our Multi-Generational Workforce*, *www.meaningfulcareers.com/defining-a-generation*; i Peakon, *Working Better Together — Understanding the Experiences and Needs of a Multigenerational Workforce*, *heartbeat.peakon.com/reports/working-better-together*.

Poleć książkę

Typy osobowości

Poza różnymi kategoriami programistów mamy też do czynienia z typami osobowości, jednostkowymi właściwościami i przyzwyczajeniami, które wydają się występować u programistów, przy czym każde wiąże się z własnym szeregiem wyzwań.

Literatury traktującej o teoriach osobowości, ich klasyfikacji i radzeniu sobie z nimi jest pełno. Z tego olbrzymiego korpusu najważniejsze wydaje się wskazanie dorobku Myers i Briggs, które w latach 1942 – 1962 wyłożyły fundamenty leżące u podstaw testów osobowości i opracowały ramy kategoryzacji osobowości. Celem kwestionariusza Myers Briggs Type Indicator (MBTI) było sprawienie, aby teoria typów psychologicznych C. G. Junga stała się zrozumiała i przydatna w życiu codziennym[7]. Ich dorobek został spopularyzowany w 1984 w książce *Please Understand Me*[8], w której zaprezentowano metodologię Myers-Briggs w bardzo przystępnej formie.

Typy opisane pierwotnie przez Junga to ekstrawersja (E) i introwersja (I), poznanie (S) i intuicja (N), myślenie (T) i odczuwanie (F) oraz obserwacja (P) i osądzanie (J). Co ciekawe, badania wykazały[9], że programiści niewspółmiernie często należą do typu INTJ (introwersja, intuicja, myślenie i osądzanie), a Mickey i Ron rzeczywiście skłaniają się ku niemu.

W *Please Understand Me* można zapoznać się z wieloma wartymi zgłębienia zagadnieniami, zwłaszcza w odniesieniu do własnych relacji. Niemniej jednak komunikowanie się z ludźmi o zróżnicowanych osobowościach w tak schematyczny sposób niesie pewne zagrożenia. Zalecamy, abyś zapoznał się ze sformalizowanymi klasyfikacjami osobowości, ale skoncentrował się na zarządzaniu tymi typami osobowości, które sam u ludzi dostrzegasz.

Poniżej przedstawiamy szereg typów osobowości, z którymi, jak wnioskujemy z własnego doświadczenia, prawdopodobnie spotkasz się w pracy. Choć nie jest to wyczerpująca lista, jest ona dobrą ilustracją różnych typów, którymi będziesz musiał zarządzać, a ponadto podajemy w niej wskazówki co do ich rozpoznawania.

Lewopółkulowcy i prawopółkulowcy

Teoria półkul mózgowych wyrosła na gruncie dorobku Rogera W. Sperry'ego[10], którego badania wykazały, że lewa i prawa półkula mózgu są wyspecjalizowane w wykonywaniu różnych zadań. Lewa półkula mózgu standardowo wykonuje zadania analityczne i werbalne. Lewa półkula lepiej posługuje się mową niż prawa, podczas gdy prawa odpowiada za postrzeganie przestrzenne i muzykalność, między innymi.

---

[7] The Myers & Briggs Foundation, *MBTI Basics*, *www.myersbriggs.org/my-mbti-personalitytype/mbti-basics*.

[8] David Keirsey i Marilyn Bates, *Please Understand Me: Character & Temperament Types*, B & D Books, 1984.

[9] C. Bishop-Clark i D. Wheeler, *The Myers-Briggs Personality Type and Its Relationship to Computer Programming*, „Journal of Research on Computing in Education" 26, nr 3 (wiosna 1994), s. 358 – 70.

[10] Nagroda Nobla w dziedzinie fizjologii lub medycyny w roku 1981 została podzielona między Rogerem W. Sperrym „za odkrycia dotyczące specjalizacji funkcjonalnej półkul mózgu" a Davidem H. Hubelem i Torstenem N. Wieselem „za odkrycia dotyczące przetwarzania informacji w układzie wzrokowym". Zob. *www.nobelprize.org/prizes/medicine/1981/summary*.

Kup książkę Poleć książkę

Jeżeli jesteś programistą lub zajmujesz się techniką, prawdopodobnie byłeś już uznawany za lewopółkulowca, czyli osobę werbalną, logiczną, analityczną i obiektywną. Poprawniej byłoby jednak mówić o *dominacji* lewej półkuli, ponieważ obie półkule są czynne przez cały czas. Możesz zatem być lewopółkulowcem, ale wykazywać się właściwymi dla prawej półkuli skłonnościami do myślenia niewerbalnego, wykorzystywania intuicji, niesztampowego myślenia i subiektywizmu — cechami odpowiadającymi raczej twórcom, takim jak muzycy, pisarze i artyści.

Choć właściwe dla lewej półkuli umiejętności analityczne są nieodzowne dla dobrego programisty, czynności związane z prawą półkulą często okazują się równie ważne, jako że programowanie jest bardzo twórczym działaniem (przypominającym raczej pisanie powieści, o czym wspomnieliśmy w poprzednim rozdziale). W istocie podobnie jak inni odkryliśmy, że niektórzy z najlepszych programistów są również muzykami. Mickey sam był zaangażowanym muzykiem, kiedy odkrył świat programowania na pierwszym roku studiów. „Kiedy zacząłem programować, od razu poczułem się komfortowo z tym medium. Byłem już wtedy muzykiem. Teoria muzyki jest w całości oparta na matematyce, a godziny spędzone na ćwiczeniach wymagają poświęcenia i dyscypliny. Odkryłem, że kreatywny aspekt grania i komponowania jest bardzo podobny do twórczej części projektowania i opracowywania programu. O dziwo, nawet debugowanie programu pod pewnymi względami przypomina naukę grania piosenki — trzeba grać jedną z jej części w kółko, aż wreszcie uda się to zrobić poprawnie, tak samo jak program trzeba uruchamiać w kółko, aż zacznie działać. Zauważyłem nawet, że podczas programowania tracę poczucie czasu, podobnie jak przy ćwiczeniu gry na gitarze. Programowanie stało się głównym ujściem dla mojej energii twórczej, choć cały czas gram na instrumencie i komponuję piosenki".

Ta historia nie jest niczym wyjątkowym wśród licznych programistów, których znamy. W istocie temat ten stał się standardowym wątkiem podejmowanym przy pierwszej rozmowie z potencjalnym pracownikiem. Jeżeli kandydat jest muzykiem, rozmowa o gatunkach muzyki, jakie lubi i jakie gra, teorii muzyki i roli muzyki w jego życiu daje nie tylko cenny wgląd w charakter kandydata, ale pomaga mu się też rozluźnić. Choć bycie muzykiem nie jest warunkiem bycia świetnym programistą, jeszcze nie zetknęliśmy się z przypadkiem, w którym byłoby to utrudnieniem!

Sowy i skowronki

W przeciwieństwie do przeważającej części pracowników większości organizacji, którzy są skowronkami, programiści są przeważnie sowami. Zazwyczaj zjawiają się w ciągu dnia roboczego i pracują do późna, niekiedy przez noc, kiedy zajmują się ważnym lub interesującym projektem. Ogólnie rzecz biorąc, jeżeli jesteś nastawiony na wyniki, nie powinieneś mieć problemu z tym, że tacy ludzie dostarczają wszystko to, czego oczekiwałeś, lub nawet więcej. Niemniej jednak wiąże się to niekiedy z problemami komunikacyjnymi, jako że tacy pracownicy muszą być wystarczająco dyspozycyjni, aby zjawiać się na zebraniach i uczestniczyć w wymianie informacji.

Kup książkę Poleć książkę

Geoffrey James przytacza w *Tao programowania*[11] następującą sytuację:

> *Menedżer poszedł do swoich programistów i stwierdził: „Jeśli idzie o wasze godziny pracy: musicie przychodzić o dziewiątej rano i wychodzić o piątej po południu". Usłyszawszy to, programiści rozwścieczyli się i kilku na miejscu odeszło z pracy.*
>
> *Menedżer rzekł: „Dobrze więc, w takim wypadku możecie sobie sami ustalić godziny pracy, pod warunkiem że będziecie kończyć wasze projekty zgodnie z planami". Wówczas programiści, zadowoleni, zaczęli przychodzić do pracy po południu i pracować do wczesnych godzin porannych.*

Aby uniknąć takich problemów, stanowczo zalecamy, aby nie zmuszać sów do zjawiania się w pracy o 9 rano. Polecamy natomiast wskazanie „godzin urzędowania", w których wszyscy mają zapewniać choćby minimalny, rozsądny poziom komunikacji zespołowej[12]. Pozwoli to z łatwością oszczędzić ludziom dużo czasu i męki. Miej na uwadze, że okresowo należy rewidować godziny urzędowania, ponieważ z biegiem czasu harmonogramy niektórych programistów zaczną się rozjeżdżać, przez co będą coraz mniej ściśle ich przestrzegać.

Drugim problemem jest wizerunek Twojego zespołu w oczach reszty organizacji. „Twoi programiści zjawiają się w pracy dopiero w południe" to krytyczne spostrzeżenie, z którym zetknęliśmy się we wszystkich organizacjach. Takie uwagi należy kontrować aktywnym zwracaniem uwagi na to, ile godzin pracy zespół wkłada w projekt i że programiści siedzą do późna, pisząc i sprawdzając kod. Wskaż kilka najbardziej charakterystycznych sów i zwróć uwagę na skupienie, jakie są w stanie osiągnąć za sprawą swoich przyzwyczajeń. Działaj z wyprzedzeniem — nie czekaj z tym, aż pierwsze takie uwagi się pojawią.

Kowboje i farmerzy

Większość programistów ma skłonność do bycia kowbojami, a nie farmerami. Innymi słowy, kiedy pojawia się jakiś problem, ich pierwszą reakcją jest „dosiąść konia i odjechać", by samodzielnie go rozwiązać. Tacy programiści często pomijają planowanie, co skutkuje doraźnymi rozwiązaniami, które mogłyby jedynie zyskać na lepszym wykorzystaniu standardów, praktyk i możliwości zespołu.

Proces wytwarzania oprogramowania powinien jednak bardziej przypominać uprawę roli. Farmerzy metodycznie orientują się w stanie rzeczy, przyglądają się warunkom glebowym, sadzą, podlewają, pielą i zbierają plony. Niezawodne, rozszerzalne i łatwe w konserwacji oprogramowanie tworzy się w równie metodyczny sposób.

Ważne jest zatem wyłonienie kowbojów i przypilnowanie, aby zanadto nie szarżowali przy opracowywaniu rozwiązań, które ostatecznie staną się źródłem kolejnych problemów.

---

[11] Geoffrey James, *Tao programowania*, *http://www.mteg.jaszczur.org/trash/tao/* [dostęp: 6 maja 2020].

[12] To samo zalecamy w stosunku do tych członków zespołu, którzy znajdują się za granicą lub w innych strefach czasowych.

Kup książkę Poleć książkę

Ponieważ wielu programistów ma skłonności kowbojskie, tym ważniejsze jest wypracowanie takiej kultury, w której tego typu zachowania nie są tolerowane, a przy wszystkich większych projektach przestrzega się metodycznego cyklu rozwoju.

Czasami jednak rzeczywiście potrzeba kowboja zamiast farmera, przeważnie przy małych projektach prototypów, którymi może zająć się jedna osoba. Przekonaliśmy się, że warto mieć pod ręką „rewolwerowca" do takich projektów, ponieważ korzysta na tym i kierownik, i on sam. Dopasowanie potrzeb i osobowości programisty przynosi wszystkim zainteresowanym satysfakcję i zwiększa szansę na powodzenie.

Wielu kowbojów jest świetnymi programistami, ale trzeba zarządzać nimi ostrożnie, aby uzyskać od nich pożądane wyniki. Mają skłonność do gwiazdorzenia i tworzenia podziałów w organizacji, więc uważnie ich obserwuj i działaj szybko, kiedy pojawiają się problemy.

Programiści, którzy nie potrafią przestać być kowbojami, zwykle nie wytrzymują długo w organizacji. Wynika to z tego, że ostatecznie albo kierownik ma już dość ich ciągłego szarżowania w pojedynkę i ich zwalnia albo że sami są zmęczeni ciągłym trzymaniem ich w ryzach i odchodzą z pracy.

Bohaterzy

Bohater jest kimś, kto podejmuje się pracy wymagającej nadludzkiego wysiłku — i dostarcza rezultaty. Bohaterzy przypominają pod tym względem kowbojów, lecz potrafią sprawnie pracować w zespole i w ramach procesu produkcyjnego. O bohaterów należy dbać, a sami często stają się gwiazdami organizacji.

Największe wyzwanie związane z zarządzaniem bohaterami polega na tym, że można łatwo wywołać u nich poczucie wypalenia, kiedy stale oczekuje się od nich nadludzkiego wysiłku. Czasami można tego wymagać, ale nie przez cały czas. Dbaj o ich dobrostan i korzystaj z ich zdolności, aby napędzać ważne inicjatywy i kluczowe projekty.

Zarówno Mickey, jak i Ron od czasu do czasu podejmowali się wykonywania trudnych projektów, dostarczając wyniki przekraczające najśmielsze oczekiwania (często po wielu zarwanych nocach i maratonach roboczych). Przyniosło to korzyści firmom, dla których pracowaliśmy. W Kenway i E&S dostarczyliśmy kluczowe produkty, ukończyliśmy przełomową demonstrację produktu Apple Computer, która okazała się decydująca dla dobrej sprzedaży najnowszej wówczas linii komputerów, zapewniliśmy naszym firmom patenty, a także przesuwaliśmy zarówno swoje własne granice, jak i granice tego, co można osiągnąć przy użyciu naszych technologii i narzędzi z korzyścią dla klientów i pracodawców. To doświadczenie okazało się dla nas nieocenione, kiedy już zaczęliśmy pracę w roli menedżerów i byliśmy dzięki temu w stanie określić granicę dzielącą selektywne, nadludzkie starania od wypalenia.

Kup książkę Poleć książkę

Introwertycy

Niektórzy członkowie załogi są tak cisi i powściągliwi, że aż niewidoczni. Mogą dostarczać bardzo dobre wyniki, ale nie wnoszą wielkiego wkładu w dynamikę zespołu i przebieg zebrań. Można ich wprawdzie zachęcić do otworzenia się w rozmowie w cztery oczy, ale w interakcjach grupowych ponownie nikną w tłumie.

Wywołując ich na zebraniach i reagując pozytywnie w sytuacjach, w których dzielą się swoimi komentarzami i spostrzeżeniami, możesz wzmocnić w nich poczucie, że mogą wnieść coś od siebie w życie zespołu. Zwracaj uwagę na okazje do włączenia ich do rozmów i docenienia ich wkładu. Rozmawiaj z nimi w cztery oczy. Nawiązuj z nimi szczególne relacje na poziomie drobnostek, takich jak podzielenie się jakimś doświadczeniem lub pożyczenie książki. Innymi słowy, znajdź sposób, aby tę więź pogłębić.

Mickey wspomina pewnego pracującego w Brøderbund introwertycznego autora tekstów technicznych, z którym nawiązał relację dzięki grom RPG. Za sprawą zachęt Mickeya ów autor zainteresował się projektowaniem gier, po czym został głównym projektantem Brøderbund, a wreszcie opublikował wiele swoich tekstów oraz cieszących się popularnością gier w innych firmach.

Ta i inne tego typu relacje były dla nas przez lata niezwykle cenne, ponieważ mogliśmy być świadkami tego, jak niepozorni ludzie za sprawą naszych zachęt rozwijali się i demonstrowali ukryty w sobie talent.

Cynicy

Za wszelką cenę należy unikać zatrudniania ludzi cynicznych do szpiku kości. Potrafią oni zatruć atmosferę w zespole i wywołać spustoszenie w organizacji, siejąc niezgodę i niezadowolenie tam, gdzie takie uczucia nigdy by się w innym wypadku nie pojawiły.

Problemem cynizmu jest to, że jest on zakorzeniony w rzeczywistości, ale rozdmuchany pod względem swojego wpływu na życie organizacji. Stwierdzenie „zarządu nie obchodzą programiści" może być prawdziwe, lecz cynicy wykorzystają każdą okazję, aby to udowodnić, nawet jeśli w rzeczywistości wcale tak nie jest. Potrafią wytykać każdą niezamierzoną zniewagę, taką jak zmiana rodzaju napojów w biurowej lodówce, jako przykład „próby upokorzenia załogi programistów". Takie komentarze są w oczywisty sposób nieprawdziwe, a w najlepszym wypadku demoralizujące. Jeśli zatrudnisz takiego cynika, nie będziesz nawet mógł wrócić z urlopu, żeby nie zastać pogrążonego w chaosie, wykolejonego, a być może nawet zbuntowanego zespołu.

Buce

Niektórzy ludzie są zwyczajnie bucami. Są opryskliwymi, pełnymi jadu, toksycznymi istotami. Oczywiście mogą też mieć swoje jasne punkty, takie jak błyskotliwość, talent techniczny i znakomite umiejętności programistyczne. Taka błyskotliwość jednak nie jest warta ceny, jaką jest obecność tego typu człowieka w organizacji. Problemem jest to, że znajdują się w pobliżu.

Kup książkę Poleć książkę

Przyjmij zasadę „zero tolerancji dla buców". Z własnego doświadczenia możemy Cię zapewnić, że pożałujesz, jeśli postąpisz inaczej. Rozciągnij tę zasadę także na cyników i palantów. Twoi pracownicy to docenią, a Twoja praca stanie się o wiele łatwiejsza. Więcej o radzeniu sobie z cynikami, palantami i bucami przeczytasz w rozdziałach 5. i 8.

Podsumowanie

Ten rozdział miał na celu pomóc Ci przekonać się, że zrozumienie programistów wcale nie jest proste, nawet jeśli sam jesteś jednym z nich. Różne perspektywy, które tutaj przedstawiliśmy, mają Ci jedynie pomóc w odkryciu takich sposobów na radzenie sobie z zatrudnianymi i zarządzanymi programistami, jakie odpowiadają Ci najbardziej. Zarządzanie ludźmi jest trudne, a niektórzy z najbardziej utalentowanych programistów są też najtrudniejsi w obyciu. To miecz obosieczny.

Zaprezentowaliśmy listę różnych typów osobowości, abyś mógł zwrócić na nie uwagę, ale stanowczo odradzamy przypisywania ludzi do takich prostych kategorii. Traktując każdego pracownika jako wyjątkową jednostkę, będziesz odnosić większe sukcesy w roli kierownika zespołu programistycznego.

Kup książkę Poleć książkę

Kup książkę Poleć książkę


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

Lichty i Mantle napisali przewodnik, który pomoże Ci w zatrudnianiu, motywowaniu i prowadzeniu działających na najwyższych obrotach zespołów programistycznych. Przedstawione w nim praktyczne zasady i porady szkoleniowe składają się na świetny plan postępowania zarówno dla świeżo upieczonych, jak i doświadczonych kierowników projektów programistycznych.

— **Tom Conrad, dyrektor ds. technologii, Pandora**

Zawarte w tej książce perełki sprawiają wrażenie porad i wskazówek, które otrzymałbym od zaufanego mentora — takiego, któremu nie tylko ja sam wierzę, ale który także ufa, że jestem zdolny do przyswojenia sobie tej mądrości, dostrzeżenia jej granic i właściwego jej zastosowania.

— **Mike Fauzy, dyrektor ds. technologii, FauzyLogic**

Efektywne zarządzanie: znakomity produkt, zadowolony zespół!

O ile techniki wytwarzania oprogramowania rozwinęły się w imponujący sposób, o tyle metody zarządzania tym procesem wciąż są dalekie od doskonałości. Absurdalne przekroczenia budżetów lub terminów zbyt często były dowodem na to, że uzyskanie kontroli nad zespołem programistów czy skuteczne zarządzanie całym projektem programistycznym jest niezwykle trudne, o ile w ogóle możliwe. Niemniej osoby interesujące się tą branżą potrafią wskazać projekty, które zespół ukończył zgodnie z założeniami: terminowo, uzyskując znakomitą jakość produktu. Jak zatem powinien pracować kierownik projektu programistycznego, aby odnieść sukces?

Książka jest bardziej źródłem inspiracji niż typowym podręcznikiem przywództwa w świecie programistów. Znalazło się tu mnóstwo praktycznych zasad, opisów przydatnych technik i narzędzi, obficie okraszonych anegdotami i przykładami z życia wyjadaczy w dziedzinie zarządzania zespołami IT. Omówiono kwestie związane z programowaniem zwinnym, rekrutacją i adaptacją nowych pracowników, jak również metody zarządzania tzw. problemowymi pracownikami i motywowania zespołów do osiągania trudnych celów. Wyjaśniono, skąd się biorą trudności w zarządzaniu programistami, i wskazano, z jakich perspektyw te trudności są łatwiejsze do rozwiązania. Sporo miejsca poświęcono problematyce przywództwa i roli kierownika w realizacji projektu, a także kluczowej roli menedżera w tworzeniu kultury zespołowej — czyli czemuś, co warunkuje sukces.

W książce między innymi:

- tworzenie najlepszego zespołu do danego projektu
- zarządzanie z poszanowaniem indywidualności programistów
- skuteczne motywowanie i pielęgnowanie efektywności zespołu
- zapewnianie funkcjonowania zespołu w warunkach korporacji
- techniki zarządcze odpowiednie dla kierowników zespołów IT
- kierowanie a samoorganizowanie się zwinnych zespołów

MICKEY W. MANTLE zajmował się chyba wszystkimi dziedzinami IT: od oprogramowania sterującego robotami po zaawansowaną animację 3D. Kierował też zespołami programistycznymi i badawczo-rozwojowymi z całego świata, między innymi w Indiach, Rosji, Kanadzie, Japonii i Korei.

RON LICHTY od 35 lat zarządza wytwarzaniem oprogramowania. Pracował nad UX/UI Apple, a następnie nadzorował wdrażanie dyscypliny inżynieryjnej na Uniwersytecie Stanforda, w Check Point, Razorfish i Fujitsu. Obecnie jest konsultantem.

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

Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
HELIONSZKOLENIA.PL

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-6977-1
9 788328 369771
Cena: 89,00 zł
INFORMATYKA W NAJLEPSZYM WYDANIU
Pearson
Addison-Wesley