SŁOWO PRZEWODNIE ERIC RIES

ZAPYTAJ SWOJEGO PROGRAMISTĘ

JAK WYKORZYSTAĆ POTENCJAŁ PROGRAMISTÓW I PODBIĆ XXI WIEK

JEFF LAWSON

Helion

Tytuł oryginału: Ask Your Developer: How to Harness the Power of Software Developers and Win in the 21st Century

Tłumaczenie: Katarzyna Bogusławska

ISBN: 978-83-283-8225-1

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

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

Printed in Poland.

SPIS TREŚCI

ROZDZIAŁ 4.

KOD JEST KREATYWNY

Jeśli chcesz zbudować statek, nie zwołuj ludzi, by przynieśli Ci drzewo, przygotowali narzędzia, nie przydzielaj pracy i zadań, lecz wzbudź w nich tęsknotę za dalekim, bezkresnym morzem.

— Antoine de Saint-Exupéry

Jeśli jesteś prezesem, spędzasz pewnie sporo czasu ze swoim zespołem sprzedażowym i prawie na pewno wiesz, jak pracują i co ich motywuje. Prawdopodobnie zdajesz sobie sprawę, że sprzedawcy lubią wygrywać i będziesz chciał tak ułożyć swój proces sprzedażowy i system wynagrodzeń, by stanowił pewnego rodzaju grę i dawał im możliwość rywalizacji. Sprzedawcy, którzy „rozbijają bank", są ogłaszani bohaterami, a prezesi są z nimi po imieniu. Niestety, zdałem sobie sprawę, że bardzo niewielu prezesów tak samo dobrze wie, co kręci programistów. Podobnie jak większość ludzi, tak i programiści lubią wygrywać, ale w ich przypadku gra jest trochę inna. Jeśli zastanawiasz się, dlaczego tak trudno pozyskać i utrzymać świetnych programistów — takich, których zatrudnia Facebook, Google czy Amazon — zacznij od zrozumienia, co ich motywuje. Jeśli dziwi Cię, dlaczego „proste" zmiany w Twojej stronie czy aplikacji mobilnej trwają tak długo, zacznij od zrozumienia interakcji między programistami a managerami. Gdy zbudujesz środowisko, w którym programiści się spełniają, zaskoczy Cię, jak wiele mogą stworzyć. Wszystko zaczyna się od świadomości tego, co kręci programistów. Wbrew popularnemu przekonaniu więcej w tym kreatywności niż wyrachowania. A to dlatego, że kodowanie to działalność kreatywna.

Trzydzieści lat temu, jeśli chciałeś grać muzykę dla publiczności, musiałeś „zostać odkryty" — podpisać kontrakt z wytwórnią, by pokryć wydatki związane z wynajęciem studia nagraniowego, wydaniem płyt, zakupem czasu antenowego w radiu. Zaledwie kilku artystom przydarzało się to w ciągu roku, a szanse, że Twoja kariera muzyczna rozkwitnie, były naprawdę niewielkie, nawet jeśli byłeś nieziemsko utalentowanym muzykiem. Większość z nich nie rzucała pracy na etacie. Podobnie działo się w branży filmowej. Aspirujący reżyserzy przeprowadzali się do Hollywood i całe lata pracowali na zmywaku, czekając na przełom w karierze. W Hollywood produkowało się zaledwie około stu filmów rocznie, a konkurencja w procesie tworzenia kreatywnych ról była bardzo ostra. Nawet bardzo utalentowani filmowcy spotykali się z odmowami w tym systemie, większość z nich wracała do domów, a ich wielkie sny o karierze w Hollywood odchodziły w niepamięć.

Jednak w ciągu ostatnich kilku dekad komputery osobiste, tanie oprogramowanie i internet zmieniły praktyki w tej materii i otworzyły drzwi do tego przemysłu dla każdego, kto chce nagrywać i rozpowszechniać muzykę czy filmy. Tanie, wysokiej jakości narzędzia i niedrogie lub wręcz darmowe kanały dystrybucji obniżyły próg wejścia, dały szansę poszczególnym artystom i rozprawiły się ze strażnikami dawnego porządku.

Za 299 dolarów obiecujący filmowiec może kupić Final Cut Pro — to samo oprogramowanie, którego używają najważniejsze studia filmowe w Hollywood i dotrzeć do miliardów ludzi poprzez serwis YouTube. Za 199 dolarów muzycy mogą kupić to samo oprogramowanie Logic Pro X, którego do nagrywania albumów używa Beyoncé, a potem rozpowszechniać swoje nagrania za darmo na SoundCloud.

Takie zjawiska zachodzą cały czas. Montero Lamar Hill, znany jako Lil Nas X, dwudziestojednolatek, który kupił podkład muzyczny za trzydzieści dolarów, zarapował do niego tekst i w grudniu 2018 r. wydał piosenkę „Old Town Road" na SoundCloud. Utwór stał się hitem i ma teraz ponad miliard odtworzeń. Ustanowił też nowy rekord liczby tygodni utrzymywania się na pierwszym miejscu listy Bilboard i został piosenką roku podczas gali MTV Music Awards w 2019 r. Innymi przykładami są komik Joe Rogan, który w 2009 r. wyprodukował darmowy podcast, a w 2020 podpisał wart 100 milionów dolarów kontrakt ze Spotify, oraz dziewięcioletni Ryan Kaji, który zarabia 26 milionów dolarów rocznie (według „Forbesa") na swoim kanale na YouTube, gdzie testuje i ocenia zabawki.

To samo dotyczy innej grupy artystów kreatywnych — programistów. Infrastruktura związana z oprogramowaniem była kiedyś niesamowicie droga, ale w tej chwili jej ceny spadły, a niektóre elementy są wręcz początkowo darmowe. Nie musisz kupować gigantycznych serwerów i wynajmować przestrzeni w centrum danych. Istnieje cały przybornik narzędzi, który możesz zdjąć z półki i wykorzystać do zbudowania własnych aplikacji — Amazon lub Microsoft, jeśli chodzi o serwery i przechowywanie, Google w zakresie map, Twilio do komunikacji, Stripe do płatności. Każdy dzieciak ma do dyspozycji te same klocki, co największe korporacje na świecie.

Podobnie jest też w kwestii dystrybucji. Programiści nie muszą już dogadywać umów z wydawcami oprogramowania, ubiegać się o miejsce w witrynach CompUSA czy rezerwować miejsca pośród preinstalowanych aplikacji na telefonie. Każdy może umieścić swoją aplikację w sklepie mobilnym, tak jak każdy może udostępnić swój film na YouTube. Programista może też wykupić reklamy w Google za pomocą swojej karty kredytowej, płacąc za to grosze za kliknięcie.

Właśnie teraz jest najlepszy czas na bycie programistą. Ogranicza Cię wyłącznie Twoja wyobraźnia. Istnieje często pomijany powód, dla którego można zauważyć tyle podobieństw między programowaniem a przemysłem muzycznym czy filmowym.

Kodowanie jest kreatywne.

W kulturze popularnej programiści przedstawiani są jako zgredy naukowe czy matematyczne. Bohaterowie, tacy jak Steve Urkel w *Sprawach rodzinnych*, Sheldon Cooper w *Teorii wielkiego podrywu* czy Dennis Nedry w *Parku Jurajskim*, stanowią obraz społecznie niedostosowanych, dużych chłopców, którzy lepiej radzą sobie z suwakiem logarytmicznym niż rozmową. Media kochają takie stereotypy, ale są one bardzo zwodnicze.

Pisanie oprogramowania jest jednak bardziej podobne do komponowania muzyki czy pisania powieści niż do matematyki czy innych nauk ścisłych. Tak jak YouTube i SoundCloud otworzyły drzwi do kariery nowej grupie artystów, tak samo programiści wykorzystują swoją kreatywną duszę do tworzenia imponujących produktów i firm, często sięgających szerokiej publiczności, tak jak megagwiazdy muzyki, filmu czy treści online.

Dwóch inżynierów założyło Instagram i sprzedało go Facebookowi za miliard dolarów, gdy firma zatrudniała zaledwie trzynaście osób. Dwóch

programistów wymyśliło WhatsApp i zatrudniało zaledwie pięćdziesiąt osób, gdy sprzedawali go Facebookowi za 19 miliardów dolarów. Kilka lat temu dwóch programistów wybrało się na hackathon w Nowym Jorku z pomysłem na implementację aplikacji do rozmów grupowych. Przy użyciu Twilio stworzyli pierwszą wersję w jednym, osiemnastogodzinnym sprincie. Swoją aplikację nazwali GroupMe i piętnaście miesięcy później sprzedali firmę Microsoftowi za 80 milionów dolarów.

Takie historie inspirują programistów — nie tylko ze względu na duże kwoty w grze, ale także dlatego, że pokazują, jak fantastyczne rzeczy może stworzyć kilku inżynierów, gdy da się im swobodę pracy, pozwoli marzyć i rozwiązywać kreatywnie rzeczywiste problemy klientów.

Wielu programistów przyjmuje to na wiarę. „Zawsze mi się wydawało, że programowanie to jedno z najbardziej kreatywnych zajęć na świecie — mówi Werner Vogels, dyrektor ds. technologii Amazona. — Każdego dnia masz szansę zbudować coś nowego. Inżynieria to niesamowicie kreatywna profesja. Nie wszystkich inżynierów szkoli się w kierunku bycia artystą kodu, ale można nauczyć się tego z czasem".

Jednak większość firm tego nie rozumie i nie tworzy dla swoich programistów środowiska, w którym mogliby ćwiczyć swoją kreatywność, a tracą na tym wszyscy. Programiści nie realizują w pełni swoich możliwości i marzą o rzuceniu pracy oraz założeniu własnej firmy. Firmy tracą, bo ich najlepsi ludzie nie są w pełni wykorzystywani. Klienci tracą, bo produkty są smutnym efektem pozbawionej namiętności fabryki oprogramowania. Aby wyrwać się z tego marazmu, firmy muszą zrozumieć, że kodowanie to działalność kreatywna, a wielu programistów to osoby kreatywnie rozwiązujące problemy i powinny być tak traktowane.

NIE CHODZI O PING-PONG

Firmy spoza Doliny Krzemowej spędzają sporo czasu, analizując, jak działają firmy technologiczne. Wysyłają zespoły na „krzemowe safari", by odwiedziły start-upy i gigantów, takich jak Google czy Facebook, a potem zbudowały laboratoria innowacji. Spotkałem się z kilkoma członkami zarządów na takich wyjazdach i zawsze podkreślałem jedną rzecz — muszą pozwolić programistom torować drogę. Niestety, zbyt wiele razy widziałem, jak prezesi opuszczają Dolinę Krzemową z niewłaściwymi wnioskami.

Łatwo przywiązać się do powierzchownych benefitów, takich jak darmowe śniadania, chodzenie w kolorowych T-shirtach i bluzach czy przyprowadzanie psów do pracy. Jeśli odwiedzisz wystarczająco wiele biur w okolicy Doliny Krzemowej, łatwo będzie Ci dojść do wniosku, że jeśli postawisz wystarczająco wiele stołów do ping-ponga lub kolorowych rowerków, doskonałe oprogramowanie urodzi się samo. Jasne, powinieneś pozwolić swoim programistom nosić takie koszulki, jakie chcą, ale nie w tym tkwi sedno sukcesu.

Zwykli obserwatorzy innowacji często pomijają to, co najważniejsze, czyli fakt, że programistom daje się wolność i kładzie na ich barki odpowiedzialność. Chodzi nie tylko o swobodę godzin pracy czy stroju służbowego, ale wolność tworzenia.

Kiedy myślisz o programistach, zamiast widzieć w wyobraźni Urkela, Sheldona czy Dennisa Nedry'ego, patrz na Patricka McKenzie, Ryana Leslie, Leah Culver czy Chada Etzela.

Patrick McKenzie pracuje dla Stripe'a, a szerzej znany jest w internecie jako „Patio11", czyli pod pseudonimem, jakiego używa na stronie Hacker-News (chyba najbardziej popularnej wśród programistów stronie do rozmów zawodowych), gdzie od dawna jest jednym z najwyżej ocenianych komentatorów — i to z wielu dobrych powodów. Na swojej stronie Kalzumeus.com zamieścił jeden z najgłębszych i najbardziej zajmujących esejów na temat bycia programistą. Patrick mieszka w Japonii, gdzie kiedyś pracował jako korporacyjny programista, zanim wybił się na samodzielność; stworzył dwa proste biznesy online: generator kart do bingo dla nauczycieli i aplikację do przypomnień — które dały mu także finansową niezależność. To prawdziwy człowiek renesansu. Mówi po hiszpańsku i japońsku, potrafi godzinami opowiadać o szczegółach systemu podatkowego w Stanach Zjednoczonych, a kiedyś z pasją pisał o tym, jak dobrze działał japoński system zarządzania kryzysowego podczas trzęsienia ziemi w 2011 r. „Sposób, w jaki ten system działał — mówię to bez choćby odrobiny przesady — jest triumfem ludzkiej cywilizacji. Każdy inżynier w tym kraju powinien chodzić z głową nieco wyżej" — pisał.

Ryan Leslie jest nominowanym do nagrody Grammy raperem i producentem, przedsiębiorcą i — dokładnie — programistą. W wieku czternastu lat uzyskał idealne 1600 punktów podczas egzaminów SAT. Wcześnie skończył liceum, by odbyć studia na Harvardzie, gdzie zajmował się naukami

politycznymi i makroekonomią; uzyskał dyplom w wieku dziewiętnastu lat. Równolegle sam nauczył się produkcji muzyki, a po ukończeniu studiów podpisał kontrakt nagraniowy z Universal Motown. Wydany w 2009 r. utwór *Transition* dotarł do czwartego miejsca list przebojów R&B w Stanach. Kiedy poprosił wytwórnię o listę swoich fanów, jej przedstawiciele wzruszyli tylko ramionami — nie mieli pojęcia. Jaka firma nie ma żadnego kontaktu ze swoimi klientami? Na pewno nie taka, która zamierza przetrwać w cyfrowej gospodarce. Ryan znalazł branżę, która stała u progu zmian i to on miał do nich doprowadzić. Nauczył się programować i utworzył oprogramowanie pod nazwą SuperPhone, które umożliwia artystom bezpośrednią komunikację z fanami.

SuperPhone pozwala Ryanowi udostępniać swój numer telefonu podczas koncertów, na swojej stronie i w mediach społecznościowych, a gdy słuchacze dzwonią lub piszą wiadomości, dodaje ich do listy milionów swoich fanów. SuperPhone to w zasadzie aplikacja do zarządzania relacjami z klientami w oparciu o wiadomości tekstowe. Kiedy artysta wydaje nowy utwór lub ogłasza daty trasy koncertowej — może wysłać wiadomość bezpośrednio do fanów. To oprogramowanie pozwoliło Ryanowi wygenerować miliony dolarów ze sprzedaży albumów i produktów sygnowanych jego nazwiskiem. Co lepsze, SuperPhone staje się kwitnącym interesem sam w sobie dzięki środkom z kilku najlepszych funduszów inwestycyjnych w Dolinie Krzemowej, a wszystko to przy załodze nieprzekraczającej dwunastu osób. SuperPhone ma około dwóch tysięcy klientów: od gwiazd, takich jak Miley Cyrus czy 50 Cent, po wielkich sprzedawców elektroniki i luksusowych marek zegarków. Wszyscy korzystają z aplikacji do budowania osobistych, ludzkich kontaktów ze swoimi klientami. Ryan nie jest tylko raperem — jest także programistą, przedsiębiorcą, w którego inwestują najlepsze fundusze, i człowiekiem oprogramowania, który używa kodu, by rozwiązywać problemy. „To dopiero była przygoda! Coś, co naprawdę zmieniło moje życie. Jesteśmy naprawdę emocjonalnie związani z tym, co naszym zdaniem możemy dostarczyć".

Leah Culver skończyła studia informatyczne na Uniwersytecie w Minnesocie w 2006 r. i przeprowadziła się do Doliny Krzemowej. Do tej pory była założycielką lub współzałożycielką trzech firm. Dwie pierwsze zostały odkupione, a w tej chwili prowadzi trzecią — Breaker — siedmioosobową firmę sprzedającą oprogramowanie do podcastów. Między działaniem

w start-upach pracowała jako programistka w firmach Medium i Dropbox. Jak sama przyznaje, jako przedsiębiorca musi ćwiczyć obie półkule mózgu. Uczy się nowych umiejętności, takich jak tworzenie produktów, zarządzanie pracownikami czy prowadzenie biznesu. „W start-upach lubię to, że jest tam trudno. Wiele zwykłych stanowisk programistycznych jest zbyt prostych. Nie widzę wtedy przed sobą wyzwań, a powodem, dla którego ludzie angażują się w programowanie, jest właśnie to, że stoją przed wyzwaniami i sprawia im to frajdę. Lubię robić różne rzeczy każdego dnia. Chodzi o wyzwanie polegające na tym, że najpierw nie wie się, jak coś zrobić, potem się tego uczy, a wreszcie opanowuje do perfekcji". Wspomnienie nieustającego uczenia się i poszerzania horyzontów to coś, co słyszę cały czas od najlepszych programistów.

Chad Etzel to jeden z najbardziej kreatywnych programistów, jakich kiedykolwiek poznałem. Nie ma sobie równych w słuchaniu klientów i zamienianiu tego, co usłyszy, w interesujące oprogramowanie. Ostatnie pięć lat spędził w Apple na stanowisku programisty iOS, pierwszym miejscu, w którym czuł się jak w domu po przejściu przez kilku pracodawców (łącznie z Twilio) w ciągu dziewięciu lat od opuszczenia murów uniwersyteckich. Ma twórczy zarost na twarzy, lubi zawadiackie czapki i używa ksywki „Jazzy Chad", która była też jego nazwą użytkownika na AOL, gdy był dzieckiem (ponadto gra na saksofonie na tyle dobrze, że zagrał nawet kilka koncertów w klubach jazzowych w San Francisco, a czasem zabiera instrument ze sobą do pracy). Podobnie jak Patrick, Chad ma świetne poczucie humoru, swoje zdanie i niemal zerową tolerancję na ściemnianie — czy to korporacyjny bełkot czy inny. Jego ulubioną pracą była praca dla siebie w firmie, którą założył. „Miałem całkowitą autonomię — opowiada. — Tworzenie czegoś z niczego to sytuacja, gdy ujawnia się moja największa energia i wewnętrzny ogień. Moment, kiedy ktoś mówi mi, jak coś zrobić, albo sugeruje »Te trzy rzeczy musisz zrobić, a o szerszą perspektywę się nie martw«, to chwila, gdy tracę całą motywację". Chad tłumaczy, że powodem, dla którego przeszedł przez tyle różnych firm, zanim trafił do Apple, był fakt, że niezwykle trudno było mu „znaleźć firmę, która pozwala na taki poziom autonomii czy swobody, któremu mogę się całkowicie oddać".

Programiści, tacy jak Patrick, Ryan, Leah czy Chad, są niesamowicie kreatywni. Z dziecinną łatwością przychodzi im stosowanie rzemiosła programowania w celu służenia klientom i rozwiązywania ich problemów

oraz budowania biznesu. Jednak w większości firm to prezes czy kierownik produktu byliby odpowiedzialni za spotkania z klientem, projektowanie produktu i stworzenie dokumentu specyfikacji, na podstawie którego mieliby pracować programiści.

Wiosną 2020 r. Twilio zapytało około tysiąc programistów z całego świata o to, jak oni sami i ich przełożeni postrzegają rolę programisty w firmie. Rezultaty wiele nam powiedziały. Ponad 66% programistów odpowiedziało, że uważa, iż są ponadprzeciętnie kreatywni, ale tylko 50% przyznało, że potrzebuje ponadprzeciętnej kreatywności w wykonywaniu swojej pracy. Hmmm? To na co programiści pożytkują tę dodatkową kreatywność? Wielu z nich kieruje ją na aktywności pozazawodowe: 48% osób przyznało, że ich zainteresowania obejmują dziedziny, w których projektowanie jest kluczowe (np. architekturę, meble, sieć), a 32% odpowiedziało, że zajmuje się sztuką (np. malarstwem, rzeźbą i pracą w ceramice) w wolnym czasie (i jeszcze jeden pogromca stereotypów — programiści to sportowcy! 36% z nich to biegacze, 33% — kolarze, 28% gra w koszykówkę, a 25% wędruje).

Wręczenie takim programistom „Dokumentu wymagań produktowych" szczegółowo opisującego, co zbudować, marnuje większość ich potencjału. Właśnie dlatego moja najważniejsza rada dla wszystkich prezesów pielgrzymujących po Dolinie Krzemowej brzmi:

Ze swoimi programistami *dziel się problemami, nie rozwiązaniami*.

A potem patrz, jak dzieje się magia. Rośnie jakość oprogramowania, drastycznie maleje długość cykli wytwórczych, użytkownicy są szczęśliwsi, a programiści zostają w Twojej firmie na dłużej. Nigdy nie spotkałem szefa, który nie pragnąłby wszystkich tych elementów.

ASHTON KUTCHER I POTĘGA HACKATHONÓW

Co się dzieje, gdy pozwolisz programistom na kreatywność? Ashton Kutcher i Demi Moore zbudowali organizację non profit (pod nazwą Thorn) napędzaną technologią właśnie dzięki temu.

W 2012 r. Ashton i Demi obejrzeli film dokumentalny o seksualnym wykorzystywaniu dzieci i szok, jakiego doznali, zmusił ich do działania. Doprowadziło ich to do założenia organizacji Thorn — fundacji, która buduje technologię mającą chronić dzieci przed wykorzystaniem. Pochodzące stamtąd oprogramowanie jest stosowane przez rządowe agencje

na całym świecie do szybszego znajdowania ofiar przemytu dzieci i stręczycielstwa, a także eliminacji dziecięcej pornografii z internetu. Do dziś oprogramowanie fundacji Thorn pomogło zidentyfikować ponad 14 000 ofiar przemytu dzieci w celu prostytucji i zakończyć cierpienie około 2000 dzieci, których seksualne wykorzystywanie było nagrywane i dystrybuowane jako materiały pornograficzne.

Kilka lat temu Ashton zapytał nas w Twilio, czy nie zorganizowalibyśmy hackathonu, który pomógłby fundacji Thorn stworzyć funkcjonalności związane z komunikacją w ich oprogramowaniu. Byliśmy dumni, że możemy uczestniczyć w tak ważnej misji i dołożyć do niej swoją cegiełkę.

Ashton lepiej niż większość inwestorów rozumie to, jak pracują programiści i jak ich motywować (a zdecydowanie rozumie to lepiej niż większość aktorów). To po części dlatego fundusz inwestycyjny, którego jest współzałożycielem od 2010 r., A-Grade Investments, urósł z 30 milionów dolarów do 250 milionów (według „Forbesa"), a sam Kutcher zyskał reputację wyróżniającego się inwestora. Jego wyczucie w tej branży sprawiło, że zainwestował w firmy, które odniosły wielki sukces, np. Warby Parker, Spotify, Skype czy Airbnb. Jedną z jego najlepszych inwestycji jest 500 000 dolarów włożone w Ubera w 2011 r.

Będzie to znacznie mniejszą niespodzianką, gdy weźmiesz pod uwagę, że Kutcher studiował kiedyś inżynierię biochemiczną na Uniwersytecie w Iowa. „Kiedy byłem na studiach — opowiadał mi — jeden z profesorów inżynierii mawiał: »Naukowcy *znajdują* problemy, inżynierowie je *naprawiają*«. Zawsze w ten sposób patrzyłem na programistów. To specjaliści od rozwiązywania problemów. Siadają, przyglądają się problemowi i znajdują najbardziej wydajny sposób jego rozwiązania".

Początkowo Thorn zwrócił się do różnych firm technologicznych w okolicy San Francisco z prośbą o pomoc. „Wiedzieliśmy, czego nie wiemy — opowiada Kutcher. — Nie wiedzieliśmy, jak rozwiązać problem, ale odwiedziliśmy kilku świetnych programistów i powiedzieliśmy im: »Słuchajcie, te przestępstwa [handel dziećmi w celach prostytucji] przeniosły się do sieci. Musimy znaleźć sposób, by sprawić, że będzie to zły interes online. W tym celu będziemy musieli zbudować narzędzia. Ale potrzebujemy inspiracji do tego, jakie narzędzia zbudować«".

To spowodowało serię hackathonów. Co jakiś czas Thorn organizuje hackathony w różnych miastach i zaprasza programistów, by spędzili weekend na rozwiązywaniu problemów. Przybywają oni na te wydarzenia,

niewiele wiedząc o handlu dziećmi, ale zdają sobie sprawę, że to ważna kwestia, która domaga się rozwiązania. Ashton i inni liderzy z Thorn wyjaśniają, jak technologia pogarsza sytuację związaną z handlem dziećmi, i pytają, jak ta sama technologia może zostać wykorzystana, by powstrzymać przestępców. Programiści otrzymują pewne podsumowujące informacje dotyczące dostępnych narzędzi i danych, a potem mogą tworzyć i realizować własne pomysły. Ashton i inni członkowie Thorn pozostają do dyspozycji, jeśli chodzi o burze mózgów nad pomysłami, wnoszą też swoją wiedzę z tej dziedziny i odpowiadają na pytania.

Matką tego pomysłu była oczywiście potrzeba — podobnie jak większość organizacji non profit, Thorn nie ma worków z pieniędzmi do dyspozycji. Hackathony stały się częścią prac badawczo-rozwojowych w Thorn. „Zwykle przedstawiamy cztery czy pięć problemów i mówimy: »OK, do dzieła. Rozwiążcie te problemy«" — opowiada Kutcher. Thorn nadal wykorzystuje hackathony do poszerzania swojej pracy, ale zbudował także własny zespół inżynierów i specjalistów *data science*, w 100% skupiony na zaawansowanych narzędziach mających wyeliminować seksualne wykorzystywanie dzieci.

Co ciekawsze z mojej perspektywy, ta historia wiele mówi o samych programistach. Osoby, które uczestniczą w tych hackathonach, to często programiści, którzy w zatrudniających ich firmach są traktowani jak wyrobnicy. Thorn zaprasza ich, by spędzili weekend, próbując rozwiązać ważny i trudny problem — jak zakończyć handel dziećmi — i daje im całkowitą wolność. I wiesz co? Oni błyszczą! Ci marudni miłośnicy zamkniętych przestrzeni zamieniają się w superbohaterów! Wyobraź sobie, co mogłoby się stać, gdyby ich pracodawcy wiedzieli, ile dobra ci ludzie są w stanie stworzyć.

Zastanów się nad tym. Czy z chęcią wykonujesz swoją pracę w weekendy, za darmo, po prostu z pasji? Czy księgowi hobbystycznie rozliczają faktury poza tygodniem pracy? Być może są tacy, którzy to robią, ale pewnie nie jest ich wielu. Czy dentyści bawią się w kreatywną stomatologię w weekendy (mam nadzieję, że nie!)? Dla programistów kod to więcej niż praca — to ujście dla ich kreatywności. Kiedy nie mogą spożytkować jej w pracy, znajdują dla niej inne zastosowania. Wielu z nich angażuje się w poboczne projekty, a nawet zakłada własne start-upy.

Inżynieryjne wykształcenia Kutchera ułatwia mu zaufanie programistom i zdanie sobie sprawy, że to świetni ludzie do rozwiązywania problemów. Ale co zrobić, jeśli nie jesteś inżynierem? Cóż, to podejście sprawdza się także, jeśli jesteś prezydentem Stanów Zjednoczonych.

PREZYDENT OBAMA PYTA PROGRAMISTÓW

Po odejściu z Twilio w 2014 r. Evan Cooke, mój przyjaciel i współzałożyciel Twilio, odebrał telefon od Todda Parka, ustępującego szefa gabinetu ds. technologii przy prezydencie Stanów Zjednoczonych. Todd poprosił Evana o spotkanie, ale nie podał żadnych szczegółów. Evan był zaciekawiony, zwłaszcza domeną *@whitehouse.gov*, z której przyszedł e-mail, więc podał prywatne informacje (przez archaiczny i niezabezpieczony system e-mail) w celu dokonania weryfikacji, a w oznaczonym dniu pojawił się w Hotelu Fairmont w San Francisco, ubrany w sprane jeansy i tanią marynarkę, czyli zestaw, który w jego garderobie najbardziej przypominał prawdziwy garnitur.

On i kilka innych osób (później dowiedzieli się, że wszyscy byli programistami z seniorskim stażem w Amazonie, Apple czy Facebooku) zostali zaprowadzeni do penthouse'u z zapierającym dech w piersiach widokiem na Zatokę San Francisco. Przywitali ich Todd oraz Megan Smith, kiedyś wiceprezes w Google, w tamtym czasie następczyni Parka na stanowisku szefa gabinetu ds. technologii przy prezydencie Stanów Zjednoczonych. Wyjaśnili, że budują nową organizację — United States Digital Service. Chcieli zrekrutować małą liczbę najlepszych inżynierów z Doliny Krzemowej i przenieść ich do Waszyngtonu, by wykonali gruntowny przegląd najważniejszych elementów cyfrowej infrastruktury rządowej. Mieli oni utworzyć technologiczną jednostkę SWAT.

Potem, w promieniach słońca zachodzącego nad mostem Bay Bridge zobaczyli nadlatujące nad miasto śmigłowce Marine One i V22 Osprey, które wkrótce wylądowały na Crissy Field.

Był luty 2015 r., a poprzednie osiemnaście miesięcy nie należało do łatwych dla Białego Domu. Pod koniec 2013 r. rząd opublikował stronę HealthCare.gov w blasku fleszy, by wkrótce potem obserwować niewydolność tego systemu. Była to wielka, niezabliźniona rana dla całej administracji i dla prezydenta Obamy szczególnie, ponieważ uczynił on z reformy

systemu zdrowia kluczowy punkt swojej prezydentury. Todd i jego współpracownicy zatrudnili kilku inżynierów z Doliny Krzemowej do wsparcia naprawy systemu i stabilizacji strony.

To doświadczenie pokazało jednak, jak kluczowa w realizacji najważniejszych zadań administracji jest rządowa — często stara — infrastruktura technologiczna. Problemy były wszędzie — objęły Pentagon, Small Business Administration, Departament Edukacji, Zdrowia i Opieki Społecznej, Bezpieczeństwa Krajowego oraz General Services Administration. Systemy były liczone w tysiącach, a linie kodu — w miliardach, istna plątanina łat, obejść i prowizorek, a większość z nich tak stara, że nikt nie pamiętał, skąd się wzięła i czemu służyła.

Rząd stał przed tymi samymi problemami, które spotykają większość firm — coraz większa część działalności rządu zależała od oprogramowania, a za sprawą firm, takich jak Spotify, Uber czy Facebook, obywatele oczekiwali takiego standardu cyfrowej obsługi ze strony administracji, jaki zapewniają komercyjni dostawcy. Ponadto pojawiały się problemy z wydajnością i jakością nowo kupowanego oraz implementowanego oprogramowania. Umowy opiewające na miliardy dolarów były podpisywane niemal bezrefleksyjnie z tymi samymi firmami, którym całe lata zajmowało wdrożenie projektów, jakie okazywały się i tak czymś innym niż początkowo ustalono, mimo że szersza branża technologiczna już dawno znalazła sposób szybkiego i niedrogiego dostarczania oprogramowania o wysokiej jakości. Najnowszym bublem była strona internetowa za miliardy dolarów, ale ten projekt przynajmniej został dostarczony!

I to właśnie z jego powodu Evana i pozostałych zaproszono na to spotkanie.

Todd i Megan musieli mieć ich po swojej stronie, ale nie mieli zbyt wielu asów w rękawie. Ci programiści mogli pracować wszędzie, gdzie tylko chcieli. Nie dało się skusić ich pieniędzmi, bo agencje rządowe nie były w stanie dorównać pensjom gigantów Doliny Krzemowej.

Megan obrała więc inną strategię. Podeszła do okna, wskazała stocznie Richmond i zaczęła opowiadać, że dokładnie tam Stany Zjednoczone zbudowały najlepsze statki towarowe, cuda inżynierii, które potrafiły oszukać niemieckie U-boty. Powiedziała też, że Evan i pozostali programiści mają szansę być jak ci inżynierowie, którzy zbudowali tamte zwycięskie okręty, a ich kraj mógł triumfować w drugiej wojnie światowej.

Wtedy otworzyły się drzwi i do pokoju wszedł prezydent Obama z prostym przekazem: „Wasz kraj Was potrzebuje". Obszedł cały stół, z każdym przywitał się osobiście. „Podaj mi jeden powód, dla którego nie możesz pojechać do Waszyngtonu i służyć swojemu krajowi — powiedział. — Czy chodzi o pracę? Mam do kogoś zadzwonić? Zadzwonię do każdego, kto by to nie był!". Nikt z piątki gości nie poprosił prezydenta o wykonanie telefonu. Nikt nie wymyślił dobrego powodu, by nie podjąć się tej pracy. Potem szybko zrobili zdjęcie grupowe i Obama wraz ze swoją świtą zniknęli za drzwiami.

Dwa miesiące później Evan pojawił się w Waszyngtonie w pierwszym dniu swojej pracy dla U.S. Digital Services, którego biuro mieściło się dwie przecznice od Białego Domu. Spędził tam następne trzy lata i do teraz twierdzi, że było to jedno z najlepszych doświadczeń w jego karierze.

Zastanów się, co Obama zrobił w tej sytuacji i czego nie zrobił. Nie kazał Evanowi i pozostałym programistom przyjechać do stolicy i wypocić z siebie kod. Podzielił się z nimi problemem i to dużym. *Rząd Stanów Zjednoczonych trzeba naprawić i chcę, żebyś to Ty się tym zajął*. To dopiero podzielenie się problemem! Prezydent odwołał się bezpośrednio do kreatywnej duszy programistów. Jak podeszliby do wbudowania technologii w rząd? Jakie problemy mogliby rozwiązać, gdyby mieli poparcie prezydenta? Pozwolił im zobaczyć, że interesuje go ich umysł i wyobraźnia, a nie tylko umiejętność programowania. Kryło się w tym tak wiele szans — „środowisko bogate w cele" — jak nazywa to Evan.

BASECAMP — PRZYPISUJ PROBLEMY, NIE ZADANIA

Firma Ashtona — Thorn — zwróciła się z prośbą o pomoc do programistów z konieczności. Obama sięgnął po ich pomoc, bo był w kryzysie. Jednak najlepsze firmy przyjmują podejście „zapytaj swojego programistę" jako część swojej codziennej praktyki, uwalniając swoich programistów i ich kreatywność w każdej sytuacji. Świetnym tego przykładem jest Basecamp — mała, ale kwitnąca firma z Chicago, która zatrudnia około sześćdziesiąt osób.

Jason Fried i David Heinemier Hansson prowadzą Basecamp niemal jak laboratorium poświęcone badaniu nowych sposobów pracy, które uszczęśliwią pracowników i pozwolą im realizować w pracy pełnię swoich

możliwości. David, znany jako DHH, to programista, który stał się popularny po zbudowaniu Ruby on Rails, szeroko używanego frameworka do aplikacji sieciowych. Jednak on i Jason wiele myślą i piszą na temat samej pracy. Opublikowali dwie napisane razem książki poświęcone wytwarzaniu oprogramowania i trzy inne mówiące o współczesnych miejscach pracy pod tytułami: *Remote. Pracuj zdalnie, biuro jest zbędne*, *Rework* i *Mam kocioł w pracy. Odrzuć chaos i niepokój, osiągnij sukces*. Obaj uwielbiają dzielić się pomysłami. Oferują nawet jednodniowe seminaria poświęcone wdrażaniu nieco nieortodoksyjnego podejścia do prowadzenia biznesu, jaki praktykowany jest w Basecampie.

Fried mówi, że ich sposób polega na dzieleniu się problemami. „Po prostu mówimy zespołowi: »Mamy taki pomysł, mniej więcej w tym kierunku chcemy iść czy też coś w tym stylu chcemy zbudować. To Waszą odpowiedzialnością jest skorzystać z tych informacji i wymyśleć całe rozwiązanie«. Jest w tym wiele sprawczości i autonomii. Oni decydują, jak ugryźć dany problem. Możemy dyskutować, szachować się argumentami, ale projekt jest ich. Moglibyśmy stworzyć szczegółową specyfikację, dojść do tego, że potrzebne są dokładnie czterdzieści dwa kroki, a potem... rozpisać czterdzieści dwa zadania..., ale wtedy mówisz swojemu zespołowi: »Nie używajcie mózgu! Róbcie, co Wam każemy«". Fried i Heinemier Hansson opisują, skąd wziął się ich pomysł i dlaczego jest ważny. Być może przedstawią zespołowi jakiś ogólny diagram narysowany mazakiem na tablicy, który zobrazuje początkową ideę, ale nic więcej nie narzucają.

Taka była ich polityka od momentu założenia firmy w 1999 r., wtedy jeszcze pod nazwą 37signals. „Nigdy nie uważałem, że należy ludziom rozpisywać zadania w pracy. Jeśli coś choćby ociera się o kreatywność, to musisz po prostu pozwolić im działać. Po to ich zatrudniasz. Jeśli chcesz kierować każdą najdrobniejszą rzeczą, kończy się na tym, że zatrudniasz ludzi bezmyślnych. Czy możesz wtedy oczekiwać, że zrobią świetną robotę? Ja wolę zatrudniać utalentowanych ludzi i pozwolić im rozwiązywać problemy".

Fried i DHH nigdy nie ujawnili danych biznesowych na temat Basecampu, ale mają około pół miliona obserwatorów na Twitterze, właśnie napisali książkę o tym, jak mało pracują, a DHH słynie ze swojej kolekcji aut wyścigowych — więc sądzę, że nieźle im się wiedzie. Jeśli powierzenie programistom największych problemów firmy sprawdziło się w ich przypadku, zgaduję, że sprawdzi się i w Twoim. Na pewno zadziałało to u mnie.

DZIELENIE SIĘ PROBLEMEM W OSTATECZNOŚCI

Na początku funkcjonowania Twilio odebrałem lekcję tego, jak kreatywni mogą być programiści i jak szybko mogą uwinąć się ze swoją pracą, jeśli tylko zejdziesz im z drogi i pozwolisz działać. W tym przypadku jeden kierownik produktu i jeden inżynier zdołali w dwa tygodnie wyprodukować coś, co w innych okolicznościach mogło zająć dziewięć miesięcy.

Gdy założyliśmy Twilio, mieliśmy dwa produkty — Twilio Voice i Twilio Phone Numbers. Programiści musieli sprawić, żeby telefon zadzwonił i stąd wziął się nasz produkt głosowy, ale potrzebowali także numerów telefonów, by odbierać i wykonywać połączenia, więc przygotowaliśmy interfejs API do kupowania numerów telefonów, początkowo w wyznaczonych okręgach w kraju, a w końcu w przeszło stu krajach na całym świecie. Oczywiście niektórzy użytkownicy chcieli korzystać z numerów, które już mieli, więc pozwoliliśmy klientom przenosić dotychczasowe numery od ich obecnych dostawców do Twilio. Prawdopodobnie przenosiłeś już kiedyś numer, gdy zmieniałeś operatora sieci telefonicznej.

To, co w przypadku przenoszenia numeru dzieje się za kulisami, jest w Stanach Zjednoczonych strasznie pogmatwane. System ten powstał naprędce w 1997 r. w odpowiedzi na ustawę Telecommunications Act z 1996 r. i nigdy nie był specjalnie poprawiany. Mówiąc ogólnie, jest to w dużej mierze ręczny proces, w którym konsultanci u operatora telefonii przerzucają się papierami. A ponieważ jeden z operatorów w tym procesie traci klienta, ma wszystkie powody, by całość utrudniać i się ociągać.

Na samym początku operacyjnie wymagająca praca, jaką była obsługa migracji numerów, przypadła pierwszej osobie, którą zatrudniliśmy na stanowisku wsparcia operacyjnego — Lisie Weitekamp. Lisa była naszym pracownikiem od wszystkiego. Trafiła do nas z obsługi Wells Fargo FOREX, gdzie koordynowała handel walutą. Jakimkolwiek problemem bym jej nie obarczył, potrafiła rozwiązać go po mistrzowsku. A na początku działania firmy takich problemów są tysiące. W naszym przypadku migracja numerów była jednym z nich.

W końcu Lisa zatrudniła młodego człowieka — nazwijmy go Tim — by przejął od niej obowiązki związane z przenoszeniem numerów. Pokazała mu, jak to wszystko działa i zapoznała z arkuszem, w którym śledziła status wszystkich migracji. Przenieśliśmy naszą uwagę gdzie indziej i zaufaliśmy, że Tim będzie monitorował stan migracji. Co robił przez pewien czas.

Wiosną 2012 r. zaczęliśmy otrzymywać od klientów reklamacje, w których skarżyli się, że przeniesienie numeru zajmuje całe wieki. Początkowo był to strumyk wiadomości e-mailowych, potem coraz więcej wiadomości o takiej treści zaczęło napływać na Twitterze, a potem osobiste wiadomości do mnie czy nawet członków zarządu. Najpierw jedna skarga, druga, a potem jakby rozerwał się worek. Któregoś dnia zdaliśmy sobie sprawę, że 90% zażaleń naszych klientów dotyczy migracji numerów. Więc zaczęliśmy to badać.

Jak się okazało, liczba wniosków o przeniesienie numeru do Twilio była gigantyczna. W miarę, jak nasz biznes nabierał rozpędu, rosła też liczba numerów telefonu do przeniesienia. Tim robił, co mógł, ale migracji było znacznie więcej niż jedna osoba była w stanie objąć pamięcią. Cały czas dodawał je do arkusza, ale napływały one szybciej niż był w stanie je przeprocesować. Ponieważ był młody i nie miał doświadczenia zawodowego, było mu wstyd powiedzieć o tym komukolwiek. Więc wnioski o migrację się nawarstwiały.

Przypominało to odcinek starego serialu *Kocham Lucy*, a którym bohaterka pracuje w fabryce cukierków. Im szybciej porusza się taśma produkcyjna, tym szybciej Lucy zjada cukierki, a potem upycha je po kieszeniach. Było to przezabawne, gdy dotyczyło Lucy Ball, ale śmieszyło mnie znacznie mniej w mojej własnej firmie.

Stało się jasne, że cały proces przenoszenia numerów nie może polegać na ręcznej pracy i monitorowaniu arkusza. Trzeba stworzyć na jego podstawie oprogramowanie i zautomatyzować tyle, ile się da. A musiało to być gotowe na wczoraj.

Ponieważ Lisa wiedziała, jak działa ten proces, miała to, czego potrzeba, by go zautomatyzować. Chris Corcoran był nowym programistą w zespole, ale wykazał się niesamowitą zdolnością rozpracowania problemów do końca. Mimo że zdobył dyplom na UMass Lowell zaledwie dwa lata wcześniej, miał już za sobą staż w NASA, pracę dla Google jako student, a programowaniu poświęci wszystkie swoje licealne i studenckie lata. Zaczęto mówić na niego Ozon, ponieważ jego inicjały tworzyły skrót CFC, który jest też nazwą substancji niszczących warstwę ozonową, która otacza i chroni Ziemię.

Zabrałem Ozona i Lisę, i zarezerwowałem jedną z naszych cennych salek konferencyjnych na całe dwa tygodnie. Spotkałem się tam z nimi,

opowiedziałem o problemie z migracją i postawiłem przed nimi wyzwanie zbudowania oprogramowania automatyzującego ten przepływ pracy... w dwa tygodnie. Lisa miała pełną wiedzę o tym, jak robi się migracje, a Ozon znał nasz kod od podszewki. Lisa miała podzielić się z Ozonem wszystkim, co wiedziała, i pozwolić mu wymyślić rozwiązanie. Potem wyszedłem.

W pierwszej chwili byli przytłoczeni. A potem zabrali się do pracy. Lisa przeprowadziła Ozona przez kilka wniosków o przeniesienie numeru, wykonując wszystkie związane z tym czynności z programistą uczepionym jej ramienia. Potem przekazała mu klawiaturę i patrzyła, jak sam je wykonuje — był to przykład tego, co nazywamy zajęciem miejsca klienta. Dopiero gdy Ozon sam poczuł, w czym leży problem, zapytała go: „No dobra, jak to zrobić za pomocą oprogramowania?".

Ozon zaczął modelować problem przy użyciu struktur danych i cały czas pytał Lisę: „Czy to wygląda dobrze?". Na koniec pierwszego dnia udało im się przygotować podstawy modelu danych. Rozumiejąc to, Ozon mógł zbudować formularz, z którego klienci korzystaliby w celu złożenia wniosku o przeniesienie swoich danych. Lisie i Ozonowi rzuciło się jednak w oczy, jak często użytkownicy podawali niepoprawne lub niepełne informacje, co wymagało potem wielu tur doprecyzowywania i wyjaśniania. Ten problem z łatwością można było rozwiązać z wykorzystaniem oprogramowania. Na koniec drugiego dnia pracy gotowy był formularz, który zbierał kompletne, niezbędne informacje.

Potem Lisa zwróciła uwagę, że gdyby dało się pogrupować wnioski o przeniesienie numeru na różnych etapach, znacznie przyspieszyłoby to pracę operatora. Ozon stwierdził, że w bardziej typowym procesie wytwarzania oprogramowania dodanie tej funkcjonalności potrwałoby miesiące. A w naszym przypadku był w stanie utworzyć działającą implementację w ciągu godziny.

Oczywiście, byłoby lepiej, gdybyśmy nie nawarzyli tego piwa wcześniej. Ale tak już bywa w start-upach, które szybko rosną. Wyraźnie podkreślam, że nie jestem zwolennikiem podejścia, w którym na porządku dziennym zamyka się kierownika produktu i programistę w sali konferencyjnej na dwa tygodnie i wsuwa im się przysłowiową pizzę przez przysłowiową szparę pod drzwiami. To nie jest najlepsza praktyka zarządzania. Ta historia pokazuje tylko, jak wiele programiści mogą osiągnąć, gdy są zmotywowani. A teraz zwracam się do Lisy i Ozona. Przepraszam. I dziękuję!

Projekt oprogramowania do przenoszenia numerów jest świetnym przykładem dzielenia się problemem w celu rozwiązania go nie tylko właściwie, ale też efektywnie. Umożliwienie programiście całkowitego zrozumienia potrzeb klienta i ich poczucia na własnej skórze jest właśnie tym, co stanowi sedno dzielenia się problemem. Lisa pomogła Chrisowi zrozumieć, dlaczego ludzie potrzebują kodu, który pisał i jak im on pomoże. Po zbudowaniu tego kapitału empatii sama czynność napisania aplikacji do przepływu procesu pracy była trywialna.

EMPATIA W STOSUNKU DO KLIENTA = LEPSZE PRODUKTY SZYBCIEJ

Prawda jest taka, że większość oprogramowania to aplikacje naprawdę proste. To rozwiązania CRUD: *Create* (stwórz), *Read* (odczytaj), *Update* (zmień), *Delete* (usuń). Większość aplikacji online to formularze, które pozwalają użytkownikom wprowadzać dane, modyfikować je, zgłaszać i usuwać. Niemal każda strona internetowa czy aplikacja mobilna, których kiedyś używałeś, w 95% opierała się na operacjach CRUD. To nie jest wyższa matematyka.

Oznacza to, że prawdziwa różnica leżąca w tym, ile zajmuje rozwiązanie danego problemu i na ile jest ono właściwe, pochodzi stąd, że programista rozumie, na czym polega problem, zupełnie tak, jak rozumiał to Ozon. Kiedy programistom naprawdę *zależy* na pracy, włącza się ich motywacja wewnętrzna i uruchamia nowe, coraz bardziej kreatywne pomysły. Gdy natomiast programista czyta jedynie specyfikację, buduje się dystans między nim a ludźmi, którzy będą rzeczywiście używać jego oprogramowania. Kod staje się lichy i podatny na błędy, ponieważ programista nie jest w stanie wczuć się w to, jak ludzie będą go używać. Co więcej, samo napisanie takiego kodu zajmuje mnóstwo czasu, ponieważ nie budzi ani emocji programisty, ani jego intuicji, jak wykonać postawione przed nim zadanie.

Jeśli kiedyś programiści czy całe zespoły budowali dla Ciebie oprogramowanie i trudno było osiągnąć coś szybko, taki właśnie był tego powód. Zawsze wydaje się, że proces ten trwa dłużej niż biznes oczekuje. Chad Etzel — Jazzy Chad — uważa to za naturalny wynik wadliwego procesu, w którym managerowie nakazują programistom zbudowanie konkretnego rozwiązania, zamiast aktywować ich na wcześniejszych etapach, kiedy definiowany był problem wymagający rozwiązania.

„Na managerach wywierana jest presja ze strony biznesowej i finansowej. Wymyślają więc pewne rozwiązanie i zobowiązują się do jego dostarczenia w określonym czasie, ale w zakresie realizacji tej obietnicy zależą od programistów. Jednak dla zespołu inżynierów może to być trudne przeżycie, gdy ktoś przychodzi do nich i mówi: »Zróbcie to, szybko, do tego czasu«, a potem ucieka".

Włączanie programistów we wcześniejsze etapy dyskusji nie jest tylko uprzejmym gestem i nie wynika z obawy przed urażeniem uczuć. Daje to także wymierne korzyści. Jak kierownicy mogą obiecywać wykonanie zadania w danym czasie, skoro nie rozumieją prawdziwej pracy, jaka musi zostać wykonana? A jeśli wskazanego rozwiązania nie da się zbudować w oznaczonym czasie? Albo funkcjonalność zostanie okrojona, albo implementacja będzie pospieszna, albo programiści wypalą się i odejdą. Nic z tego nie jest korzystnym rezultatem.

Zamiast przedstawiać programistom rozwiązania, które już wcześniej zdefiniowano, kierownicy produktów mogą podzielić się z programistami problemem i zapytać ich o najszybszy sposób rozwiązania z uwzględnieniem tego, jak działa istniejący system — jego struktury danych i ścieżki kodu.

„Za każdym razem, gdy słyszę uwagi typu: »To powinna być mała rzecz«, »Nie zajmie Ci to więcej niż dzień« krew się we mnie gotuje — mówi Chad. — O ile nie rozumiesz, jak poszczególne elementy współpracują, nie wiesz, jak zbudowana jest infrastruktura — nie masz pojęcia, ile to zadanie potrwa. Wydaje mi się, że czasem jest to powodem napięć między managerami a inżynierami".

Chad zauważa, że wielu programistów „ma głębszy wgląd w integrację lub możliwość dostarczenia pewnych produktów czy funkcjonalności i podczas gdy kierownicy produktu mówią tylko »Potrzebujemy funkcjonalności X«, mówią »OK, super. Potrzebujemy na to pół roku, ponieważ sposób, w jaki zbudowana jest nasza infrastruktura, uniemożliwia łatwe dodanie takiego kodu«. W takiej sytuacji manager prawdopodobnie nie rozumie w pełni, czemu zgłoszone przez niego zadanie jest tak trudne".

Znajdowanie najkrótszej ścieżki technicznej w danym kontekście jest tym, co programiści robią za pieniądze. Do tego się ich szkoli podczas zajęć na uczelni. Na tym w zasadzie polega algorytm Dijkstry — na wyznaczaniu najkrótszej ścieżki między wieloma wierzchołkami i wszyscy się go

uczyliśmy. Mimo to, zamiast korzystać z tych umiejętności poszukiwawczych, większość firm nakazuje swoim programistom *wyłączyć* tę część mózgu. To praktycznie zbrodnia!

Jazzy Chad jest tak utalentowanym programistą, iż nadal żałuję, że za czasów jego pracy w Twilio nie potrafiliśmy znaleźć optymalnego sposobu spożytkowania jego kreatywnej wyobraźni. Znalazł swoje miejsce w Apple, gdzie zadomowił się na ponad cztery lata, pracując nad iOS — systemem operacyjnym w iPhone'ach i iPadach.

Czym go urzekli? Tym, że poprosili o rozwiązanie problemów i usunęli się z jego drogi. Kiedy dołączył do firmy, umieścili go w zespole z kilkoma utalentowanymi inżynierami sztucznej inteligencji, ale bez jakiegokolwiek programisty aplikacji mobilnych. Następnie postawili przed nim takie wyzwanie: znajdź najlepszy sposób, by Siri robiła coś więcej niż tylko sterowała głosowo urządzeniem. To Chad stoi za całym systemem poleceń Siri w Twoim iPhonie. Chad uwielbia taką wolność. Zamiast mówić mu: „Ten piksel powinien być tutaj", jego manager powiedział tylko: „Wymyśl, jak Siri może dostarczać więcej wartości użytkownikom iPhone'a".

Jedną z kwestii, na które Chad zwraca uwagę, jest to, że przełożeni potrafią połączyć programistów z potrzebami klientów i pomóc im w wymyśleniu rozwiązania. Świetni kierownicy produktu nie są poziomem pomiędzy potrzebami klientów a programistami, wręcz przeciwnie — usuwają oni poziomy złożoności, eliminują rozwiązania pełne bezpodstawnych założeń, korygują błędne przesłanki i ułatwiają komunikację. Świetni kierownicy produktu nie izolują programistów od potrzeb klientów — pomagają im zrozumieć problemy użytkowników. Im więcej jest warstw czy poziomów między ludźmi, którzy używają produktu, a tymi, którzy go tworzą — tym gorzej. Wszystko to zmienia się wtedy w wielką grę w głuchy telefon, w której wiadomość jest przekazywana tyle razy między tyloma osobami, że programistom trudno zrozumieć, kto będzie używał oprogramowania, jakie budują.

DLACZEGO ISTNIEJE OPROGRAMOWANIE, KTÓRE NISZCZY DUSZĘ

O jednym z najbardziej ekstremalnych przypadków „zbyt wielu warstw" opowiedział mi Patrick McKenzie (znany jako Patio11) pracujący dla japońskiej firmy integrującej różne systemy, która zajmowała się outsourcingiem usług programistycznych dla różnych japońskich przedsiębiorstw, szczególnie instytucji edukacyjnych. Jej model biznesowy wyostrzał wszystkie problemy związane z „dzieleniem się rozwiązaniami" do niemal astronomicznych proporcji. Jeden szczególnie nieudany projekt obejmował tyle kłopotów komunikacyjnych i tur rozmów, że skłonił Patricka nie tylko do porzucenia tej firmy, ale całego korporacyjnego świata na przeszło piętnaście lat.

Pewien japoński uniwersytet posługiwał się manualnym systemem fakturowania, który należało zautomatyzować. W tym systemie pracownik siedział przy biurku, po jego lewej stronie leżał stos faktur, po prawej stos wyciągów bankowych. Zadaniem pracownika było dopasowanie faktur do wyciągów i opieczętowanie opłaconych faktur. Zrozumiałe, że uniwersytet chciał, by działo się to za pomocą oprogramowania. W związku z tym właściciel biznesowy na uniwersytecie przekazał informację o takim zapotrzebowaniu kanclerzowi uczelni, który kontaktował się z przedstawicielami handlowymi różnych firm, którzy z kolei przekazywali specyfikację kierownikowi produktu, który przekazywał ją innemu kierownikowi produktu, a ten wreszcie programistom.

Zgłoszone zapotrzebowanie dotyczyło zaprojektowania programu komputerowego, który przedstawia wirtualny stos faktur po lewej stronie ekranu i wirtualny stos wyciągów po prawej stronie ekranu. Użytkownik klika jeden z elementów po lewej stronie, a następnie odpowiadający mu element po prawej stronie, a następnie klika przycisk *Enter*. Jedna faktura po drugiej. Dosłownie naśladowało to manualny, papierowy proces, ale na ekranie komputera.

„Nasi sprzedawcy potwierdzili: »Tak, możemy to zbudować«. Potem trafiło to do programistów i zareagowaliśmy: »To jest — żeby nie użyć mocniejszego słowa — szaleństwo!«".

Oczywiście, komputer jest w stanie dopasować faktury i wyciągi bankowe w ciągu milisekund, bez angażowania człowieka. Jednak programiści nie

mieli szans na przekazanie tej informacji przez wszystkie te poziomy do osób, które używałyby tego systemu.

„Stanęło na tym, że zbudowaliśmy system, który nie powinien istnieć, który będzie w dodatku trudniej zbudować niż ten, w którym cała aktywność byłaby po stronie technologii i to za większe pieniądze. Nie wspominając już o tym biedaku, który spędzi całe miesiące swojego życia, robiąc ciągle klik, klik, klik, klik, klik, klik... — wspomina Patrick. — Wszystko, co opisano w tym zadaniu, mógłby robić komputer — szybciej, taniej, lepiej i bez pożerania dusz programistów i użytkowników".

Gdy Patrick był świadkiem tej dysfunkcji któryś raz z rzędu, zdecydował, że miarka się przebrała i założył własną firmę. Zdając sobie sprawę z tego, że nauczyciele to grupa, która nadal ma zbyt mały dostęp do niedrogiego oprogramowania pozwalającego na przeprowadzanie lekcji, zaprojektował Bingo Card Creator, stronę, na której nauczyciele mogą się rejestrować i za kilka dolarów miesięcznie są w stanie generować dopasowane do swoich potrzeb — tak, zgadłeś — karty do bingo. Okazuje się, że to potężna pomoc dydaktyczna. Nauczyciele szkół podstawowych tworzą takie karty na podstawie materiału z danej lekcji, np. przedstawiając głoski czy rysunki ułatwiające czytanie słów. A do tej pory robili to wszystko ręcznie. Jeśli uczysz w trzydziestoosobowej klasie, potrzebujesz trzydziestu niepowtarzalnych kart bingo, a to mnóstwo pracy. Strona Patricka pozwala Ci wskazywać słowa, liczby i obrazy — i już — możesz pobrać tyle kart bingo, ile potrzebujesz, a wszystko to za 30 dolarów. Patrick doszedł od liczby ośmiu tysięcy klientów i w czasach swojej największej świetności strona generująca karty do bingo przynosiła 80 000 dolarów przychodu, z czego czysty zysk wynosił 48 000 dolarów. Oczywiście nie stał się dzięki temu krezusem, ale gdy podliczył, ile czasu wymagało doprowadzenie produktu do tego stanu, okazało się, że jego stawka wynosiła 1000 dolarów za godzinę pracy. Całkiem nieźle jak na grupę docelową i potrzebę, która prawdopodobnie nie jest pierwszą, o jakiej się biznesowo myśli.

W kolejnym przedsięwzięciu zwrócił się do innej grupy klientów: przedsiębiorców prowadzących niezależne biznesy. Wizyty, które nie odbywają się z powodu niepojawienia się klienta kosztują lekarzy, dentystów czy fryzjerów mnóstwo pieniędzy każdego roku. W przypadku małych, często jednoosobowych działalności stare powiedzenie, że czas to pieniądz, jest aktualne jak nigdy. Patrick stworzył narzędzie AppointmentReminder.com, koleją stronę typu SaaS, umożliwiającą automatyczne, niedrogie

rozsyłanie powiadomień SMS o nadchodzącej wizycie, by zmniejszyć prawdopodobieństwo, że klienci zapomną o spotkaniu. W momencie, gdy sprzedawał swój biznes, generował on sześciocyfrowy przychód i zysk. Rozwiązanie tego problemu było czymś, co sprawiło mu szczególną satysfakcję, ponieważ wiedział, że ułatwia życie klientom.

Magia internetu jest o tyle inna od specyfiki minionej ery oprogramowania w pudełkach, że Patrick był w stanie nie tylko sam zbudować swoje strony, ale też samodzielnie zająć się marketingiem i sprzedażą za pomocą reklam Google i Facebook. Zapłacił Google trochę za zdobycie klientów, kod napisał sam, zajmował się obsługą klientów — był jednoosobową armią. Zarabiał wystarczająco dużo, by opłacić swoje rachunki i regularnie dzielił się na blogu tym, czego nauczył się po drodze. Całkowicie wyeliminował wszystkich pośredników oddzielających produkt od klientów i pokazywał, jak wiele mu to daje. Co ważniejsze, był żywym dowodem, że programista to więcej niż producent kodu. Sprawiło to, że wśród przedsiębiorców stał się kimś w rodzaju celebryty — może lekceważyć wielkie korporacje i jest w stanie utrzymać siebie i swoją wymagającą rodzinę, a utworzył dwa biznesy.

Na początku działania Twilio Patio11 był jednym z naszych największych fanów. Zbudował AppointmentReminder w całości na naszym API w celu wykonywania połączeń telefonicznych i wysyłania przypomnień przez SMS.

A potem stało się coś zabawnego. Patio11, jeden z najbardziej sławnych niezależnych programistów na świecie — a może wręcz *najsłynniejszy* — odebrał telefon od Patricka Collisona, współzałożyciela Stripe'a. Stripe to interfejs API, zupełnie jak Twilio, ale ich domeną są płatności, a nie telekomunikacja. Dzięki kilku liniom kodu programista może pobrać płatność w oprogramowaniu, które buduje. Co ciekawsze, Patio11 korzystał ze Stripe'a, by pobierać opłaty za korzystanie z AppointmentReminder. Mimo że Patio11 zarzekał się prywatnie i publicznie, że nie będzie już nigdy dla nikogo pracował, Collison złożył mu propozycję, której nie mógł odrzucić.

Collison i zespół w Stripe pracowali nad nową inicjatywą pod nazwą Atlas, której celem było rozwiązanie problemu, z jakim borykało się wielu jego klientów: założenie firmy jest zbyt trudne dla większości osób. Stripe był zainteresowany zajęciem się tą kwestią, ponieważ jego misją jest zwiększenie procentu PKB pochodzącego z internetu, zwłaszcza że każda kolejna

firma online oznacza, że dojdzie do większej liczby transakcji w sieci, a Stripe pobiera przecież malutką część wielu z nich. Collison poprosił, by Patio11, człowiek, którego pasją było zakładanie i prowadzenie własnej działalności, dołączył do zespołu ramię w ramię z Taylorem Francisem, który kierował inicjatywą Atlas na początku. Celem było takie ułatwienie procesu zakładania firmy, by mógł to zrobić *każdy* za pomocą wypełnienia formularza online. Dzięki temu Stripe usidlił najgorętszego kawalera biznesu online (jeśli się zastanawiasz — tak, jestem zazdrosny o to, że nie wiedziałem, czym skusić Patio11).

Atlas to łatwa w użyciu aplikacja, która znacząco ułatwia proces założenia firmy, „usuwając papierkową robotę, wizyty w bankach, złożoność prawną i liczne opłaty". W dwadzieścia minut przedsiębiorca może założyć odrębnie opodatkowaną firmę w Daleware, otworzyć konto firmowe, uzyskać konfigurację niezbędną do otrzymywania płatności online, a nawet uzyskać konto start-upowe w usługach, takich jak Amazon Web Services czy Google Cloud.

Czy Patio11, wyborny programista niezależny, zatrudniłby się w dużej firmie, gdyby został poproszony o produkowanie kodu? Mało prawdopodobne, ale Collison przedstawił mu duży, olbrzymi problem i poprosił o rozwiązanie, udostępniając mu zasoby stabilnego start-upu z Doliny Krzemowej i wsparcie jego inwestorów. Dla programisty takiego jak Patio11 to ekscytujące wyzwanie.

WEZWANIE DO START-UPÓW

Paul Graham jest jednym z współzałożycieli Y Combinator — jednego najbardziej udanych inkubatorów i początkowych inwestorów w Dolinie Krzemowej. Od początku swojej działalności w 2005 r. Y Combinator inwestował w ponad dwa tysiące start-upów, m.in. Airbnb, Stripe, DoorDash i Dropbox. Łączna wartość startupów finansowanych przez YC przekroczyła 150 miliardów dolarów w październiku 2019 r. Paul to programista, przedsiębiorca i specjalista technologii komputerowych. To logicznie myślący człowiek, który opiera się na zasadach i uczy początkujących przedsiębiorców, jak poruszać się w tym nowym świecie, a potem daje im wolność działania.

Jednym ze sposobów, w jaki Y Combinator znajduje, a nawet pomaga założyć start-upy, jest przedstawianie listy problemów, które wymagają rozwiązania. Y Combinator nazywa to „Request for Startups" („zapotrzebowanie na start-upy") i przedstawia następująco: „Wiele najlepszych pomysłów, jakie finansowaliśmy, zaskoczyło nas. Nie były to idee, których wyglądaliśmy. Niemniej jednak istnieją pewne typy start-upów, których aplikacje bardzo chcielibyśmy zobaczyć. Poniżej zamieszczamy zaktualizowaną listę »Request for Startups« (RFS), która bardzo ogólnie opisuje te obszary".

Lista nie wskazuje dokładnie, jak rozwiązać dane problemy. To jest rolą przedsiębiorcy, zwykle osób z zapleczem technicznym. Nie jest rzadkością, że Y Combinator finansuje kilka firm, które pracują nad tym samym zagadnieniem, ale traktują je inaczej.

Oto klika przykładów z listy RFS.

SKLEPY FIZYCZNE 2.0

Interesują nas start-upy, które wykorzystują fizyczną przestrzeń komercyjną lub sprzedażową w ciekawy i efektywny sposób. Amazon eliminuje z rynku kolejne hipermarkety. Zamiast podejmować skazaną na porażkę walkę z Amazonem, firmy muszą przemyśleć sposób wykorzystania przestrzeni handlowej w taki sposób, by podkreślał on ich zalety. Tesla, Warby Parker i Peloton wykorzystują fizyczne placówki jako ekspozycje typu showroom, co uzupełnia ich internetowe kanały sprzedażowe. Bez potrzeby przechowywania zasobów magazynowych przestrzeni handlowej można używać znacznie wydajniej.

TECHNOLOGIE USUWANIA WĘGLA

Porozumienie Paryskie wyznacza cel ograniczenia wzrostu temperatury na świecie o 1.5°C w tym stuleciu. Samo przejście na technologie wykorzystujące surowce odnawialne nie wystarczy, by osiągnąć ten cel. Musimy też usunąć węgiel z atmosfery.

ROLNICTWO KOMÓRKOWE I CZYSTE MIĘSO

Ostatnie postępy naukowe każą nam zmienić sposób, w jaki myślimy o produkowaniu białka. Po raz pierwszy jesteśmy w stanie produkować jedzenie, które jest naukowo nieodróżnialne od produktów zwierzęcych, takich jak mięso czy nabiał, wyłącznie przy użyciu komórek i bez

krzywdzenia jakichkolwiek zwierząt. Hodowanie prawdziwego zwierzęcego mięsa tylko przy użyciu komórek to naukowa rewolucja. Bardzo chcielibyśmy finansować więcej start-upów wdrażających tę technologię na rynku. Pragniemy też wesprzeć pieniężnie start-upy specjalizujące się w skalowaniu rolnictwa komórkowego. Świat niesamowicie skorzysta na bardziej zrównoważonej, tańszej i zdrowszej produkcji mięsa.

OCHRONA PRZED PODRABIANYMI TREŚCIAMI WIDEO

Ilość fałszywych treści wideo rośnie. Istnieje już technologia pozwalająca tworzyć zmontowane filmy, których nie da się odróżnić od rzeczywistości, a wkrótce będzie ona dostępna dla każdego posiadacza smartfona. Interesuje nas finansowanie technologii, która wyposaży społeczeństwo w narzędzia niezbędne do wykrycia fałszywych treści wideo i audio.

Paul i jego zespół inwestorów, podobnie jak wielu innych liderów biznesowych, znajdują się pod presją ze strony ich własnych inwestorów — partnerów w spółkach — by generować zyski poprzez dostarczanie innowacji i sukcesu komercyjnego. Realizują te cele przez zwrócenie się do programistów i przedsiębiorców. To znakomite podejście.

Wyobraź sobie, co mogłoby się stać, gdyby taką samą taktykę zastosować wewnątrz firmy. Zdefiniować najtrudniejszy, najbardziej przerażający problem, przed jakim stoi biznes, i ogłosić „Zapotrzebowanie na rozwiązania" wewnątrz firmy. Nie każde rozwiązanie będzie poprawne czy warte rozwijania, ale poprzez nazywanie dużych problemów dajesz programistom szansę pomyśleć nad tymi samymi kwestiami, które nurtują też Ciebie.

Czy to dobrze, czy źle — wszystkim nam najbardziej podobają się nasze własne pomysły. Jak lepiej sprawić, by ludzie poczuli się odpowiedzialni za efekt, niż dać im możliwość wymyślenia własnego rozwiązania? Kiedy to zrobimy, ludzie będą w stanie wyjść ze skóry, by nie zawieść. Do tego zjawiska odwołują się Paul, Ashton i Barack Obama.

DZIEL SIĘ PROBLEMAMI, NIE ROZWIĄZANIAMI

Krąży stara historia o tym, jak NASA chciała zbudować długopis, którego astronauci mogliby używać w kosmosie. Trudno było sprawić, żeby tusz płynął pod prąd, więc prototypy długopisów były rozczarowujące. Wydano

miliony dolarów, próbując stworzyć kosmiczny długopis, zanim ktoś zorientował się, jak radzą sobie z tym Rosjanie — używali ołówków. Niestety, ta historia to legenda, ale powtarza się ją często w świecie programistów. Jak wszystkie dobre bajki, przedstawia ona częsty błąd — błąd polegający na tym, że ludzie biorą się za rozwiązywanie niewłaściwego problemu. Problemem, jaki NASA powinna była rozwiązać, nie było „Jak stworzyć długopis, który będzie pisał do góry nogami bez grawitacji?". Rzeczywistym problemem było „Jak możemy pisać w kosmosie?".

Zasadniczo tematem tej książki jest *empowerment* — upełnomocnianie, przydanie władzy. Ludzie w każdej branży mobilizują się, by sprostać stawianym im wymaganiom. Książka *Zapytaj swojego programistę* poświęcona jest stawianiu wysokich oczekiwań swoim programistom. Nie chodzi jednak o liczbę linii kodu, które wyprodukują, ale o to, jak dobrze są w stanie używać swojej pomysłowości i kreatywności do rozwiązywania największych problemów świata. Są w stanie zrobić to tylko wtedy, gdy mają wystarczająco dużo sprawczości i mogą oddychać pełną piersią. Najważniejsze jest, by programistom *dawać problemy, a nie rozwiązania*.

Chociaż stoły do ping-ponga i rowerki są fajne, jestem przekonany, że kluczowe w budowaniu inżynieryjnej kultury najwyższych lotów jest włączanie programistów w problemy, które starasz się rozwiązać, i korzystanie z ich rozsądku oraz wiedzy. Nietrudno powiedzieć, czy tak się dzieje w Twojej firmie. Kiedy spotkasz programistę, spytaj go, nad czym pracuje i który problem klienta stara się rozwiązać. Będzie wiedział? Zapytaj, kiedy ostatnio rozmawiał z jakimś klientem, jak się wtedy czuł. Czy to go motywowało? Zapytaj, co z rzeczy, o których się dowiedział, zaskoczyło go. Na podstawie odpowiedzi będziesz w stanie z grubsza powiedzieć, czy programista rzeczywiście jest angażowany w problemy klientów, czy ma tylko zaimplementować rozwiązania.

ZAPYTAJ SWOJEGO PROGRAMISTĘ

JAK WYKORZYSTAĆ POTENCJAŁ PROGRAMISTÓW I PODBIĆ XXI WIEK

Kilka lat temu giełda wyceniła wartość firmy Twilio na kilka miliardów dolarów. Jej sukces nie był dziełem asów marketingu, a nieco dziwnie sformułowanej filozofii: jeśli potraktujesz swoich programistów jak partnerów i stworzysz im środowisko, które sprzyja kreatywności, przyciągniesz do siebie najlepszych. A dzięki odblokowaniu technicznych talentów i dopuszczeniu ich do współdecydowania, klienci otrzymają rewelacyjny produkt, który przyniesie sukces finansowy. To ważne: żyjemy w czasach, gdy sprawna transformacja cyfrowa oznacza przetrwanie na rynku.

To książka przeznaczona dla liderów, którzy stoją przed niełatwym zadaniem zacieśnienia współpracy między zespołami w swojej firmie i usiłują usunąć organizacyjne i mentalne bariery utrudniające porozumienie. Dowiesz się z niej, jak uwolnić potencjał swoich inżynierów i sprawić, by zaczęły powstawać innowacyjne, wspaniałe produkty. Przekonasz się, że można skłonić programistów i specjalistów w zakresie sprzedaży do mówienia jednym językiem. Znajdziesz tu mnóstwo metod, wskazówek i podpowiedzi, dzięki którym uwolnisz w zespole pasję, kreatywność i empatię — niezbędne do wytwarzania najlepszego oprogramowania na rynku. Rozprawisz się też z nieporozumieniami, które blokują innowacyjność i utrudniają skuteczną i szybką transformację cyfrową.

Dowiedz się:

- o co pytać programistę przed podjęciem decyzji biznesowej
- jak wygląda nowoczesny łańcuch dostaw oprogramowania
- dlaczego eksperyment prowadzi do sukcesu
- jak tworzyć środowisko stymulujące kreatywność zespołu
- dlaczego agile i dlaczego nie agile
- na czym polega zdrowe podejście do platform

Twój zespół. Potrzebujesz talentu ich wszystkich!

Jeff Lawson jest współzałożycielem i dyrektorem generalnym Twilio, słynnej platformy komunikacyjnej działającej w chmurze. Wcześniej pracował w Amazonie, gdzie zajmował się wdrażaniem technologii związanej z Amazon Web Services. Był także założycielem kilku startupów i autorem pomysłowych wdrożeń związanych z platformami komunikacyjnymi dla firm.

Helion

helion.pl

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

ISBN 978-83-283-8225-1

Cena: 59,90 zł