helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Sprzedaj swój software



Autor: Edward Hasted
Tłumaczenie: Jarosław Dobrzański
ISBN: 83-246-0338-7
Tytuł oryginału: Software That Sells: A Practical Guide to Developing and Marketing Your Software Project
Format: B5, stron: 368

Znajdź rynek i odbiorców dla swoich programów

- Zaplanuj projekt i zrealizuj go w odpowiedni sposób
- Poznaj skuteczne metody promocji oprogramowania
- Zorganizuj efektywny proces sprzedaży dla swojego przedsięwzięcia

Każdy, nawet najdoskonalszy i najbardziej rewolucyjny program, pozostanie wyłącznie dumą swoich twórców, jeśli nie trafi w odpowiednim momencie na rynek i nie zdobędzie grona użytkowników. Proces produkcji oprogramowania to dopiero pierwszy krok. Kolejnymi, wbrew pozorom o wiele bardziej skomplikowanymi etapami są promocja, sprzedaż i obsługa klientów. Niedocenianie bądź nawet lekceważenie tych kroków doprowadziło wiele firm, oferujących niezwykle ciekawe produkty, do upadku bądź „wchłonięcia" przez większe. Autorzy oprogramowania, którzy chcą, aby ich pomysły przyniosły nie tylko uznanie w środowisku programistów, ale również pieniądze, powinni poznać zasady budowania rynku dla swoich produktów.

Czytając książkę „Sprzedaj swój software", zdobędziesz wiedzę, która pomoże Ci stworzyć efektywnie działającą firmę programistyczną. Dowiesz się, w jaki sposób zaplanować projekt informatyczny, jak poznać oczekiwania klientów i przełożyć je na konkretny produkt. Nauczysz się budować zespół, zarządzać nim i eliminować potencjalne problemy. Poznasz metody kontroli jakości i usuwania błędów z oprogramowania. Zaplanujesz skuteczną kampanię promocyjną i działania sprzedażowe z wykorzystaniem różnych mediów i technik. Przeczytasz także o obsłudze klientów, rozwijaniu istniejących produktów i zarządzaniu firmą w warunkach szybkiego jej rozwoju, który na pewno będzie efektem wdrożenia wiadomości, jakie znajdziesz w tej książce.

- Badanie rynku pod kątem zapotrzebowania
- Wybór odpowiedniej formy prawnej przedsiębiorstwa
- Budowanie wizerunku firmy
- Zdobywanie kapitału na rozwój
- Planowanie projektu
- Rekrutacja i budowanie zespołu projektowego
- Kontrola jakości produktu
- Dokumentacja i instrukcja obsługi
- Polityka cenowa
- Kampania promocyjna
- Techniki sprzedaży
- Obsługa klientów
- Zarządzanie rozwijającą się firmą

Dołącz do grona znanych producentów oprogramowania

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

O autorze 13

Rozdział 1. Jak zwycięzcy rozpoznają zwycięzców? 19

Gdzie zwycięzcy znajdują swoje pomysły? 19
Jak utrwalić dobry pomysł? 20
Krok 1.Usiądź 20
Krok 2. Rozciągnij swój pomysł 21
Krok 3. Upewnij się, że pomysł jest dobrze zdefiniowany 22
Krok 4. Rozwiń swoją koncepcję 22
Krok 5. Skonfrontuj koncepcję z własnym doświadczeniem 23
Krok 6. Zbierz swoją paczkę 24
Krok 7. Jesteś tylko człowiekiem 24
Dlaczego opłaca się rozwijać pomysły? 25

Rozdział 2. O co pytają ludzie sukcesu? 27

Kogo pytać? 28
Ilu ludzi trzeba przebadać? 29
Etyka 30
Zadanie dla profesjonalisty? 30
Tworzenie kwestionariusza 31
Jaka może być cena? 31
Możliwości sprzedaży 32
Konkurencja 32
Informacje o firmie 33
Trudne pytania 33
Typy pytań 33
Pytania otwarte 34
Pytania zamknięte 34
Pytania porównawcze 34
Pytania nieporównawcze 34
Skala Likerta 35
Badanie pilotażowe 35

Metody kontaktu ... 36
Wywiady bezpośrednie ... 37
Z kim się kontaktować? ... 38
Jak interpretować wyniki? ... 39
Jakie cechy powinien mieć nowy program? ... 39
Gdzie taki produkt powinien być sprzedawany? ... 40
Czy znasz jakieś podobne produkty? ... 40
Jaka będzie wielkość sprzedaży? ... 41
Respondenci jako przyszli klienci ... 41
Podsumowanie ... 42
Przykładowy kwestionariusz ... 42

Rozdział 3. Plan sukcesu ... 45
Zwiększanie swoich szans ... 46
Planowanie zabiera czas, a jednocześnie go oszczędza ... 46
Myśl wstecz ... 47
Konsultacje z potencjalnymi użytkownikami ... 48
Przewidywanie cyklu sprzedaży ... 49
Tworzenie planu ... 50
Różnica między idealnym a skończonym ... 52
Ujednolicony plan realizacji ... 52
Jak realizować projekty w terminie? ... 55
Gromadzenie apostołów ... 56
Planowanie ... 62

Rozdział 4. Chcę tu pracować! ... 63
Ludzie ... 63
Podkreślanie etosu firmy ... 64
Wspólny język ... 65
Zebrania ... 65
Obieg informacji ... 66
Środowiska pracy ... 66
Relacje z pracownikami ... 68
Dzielenie się sukcesem ... 69
Stabilność firmy ... 70
Uwalnianie ludzkiego ducha ... 71

Rozdział 5. Gromadzenie środków ... 73
Porada ... 74
Pieniądze ... 74
Pieniądze, które będziesz musiał zwrócić osobiście ... 75
Pieniądze, które będzie musiała spłacić firma ... 77
Pieniądze zainwestowane w firmę ... 77
Biznesplan ... 79
Zespół zarządzający ... 80
Szansa ... 80
Okoliczności ... 81
Struktura ... 82
Streszczenie ... 83

Sprzedaj pomysł w windzie ... 83
Niepubliczna emisja pierwotna ... 83
Czy warto ubiegać się o kapitał wysokiego ryzyka? ... 84
Pomoc w naturze ... 84
Zrób to sam ... 84

Rozdział 6. Produkcja oprogramowania ... 87
Ludzie ... 88
Zdobywanie odpowiednich ludzi ... 88
Zabezpiecz się ... 93
Małe jest piękne ... 93
Proces ... 94
Negocjowanie zmian ... 95
Z dala od problemów ... 95
Zarządzanie ryzykiem ... 95
Zapas czasu i pieniędzy ... 96
Likwidowanie czynników rozpraszających ... 97
Projekty zlecone ... 104
Wiedzieć, kiedy skończyć ... 108
Lista kontrolna ... 108

Rozdział 7. Jak nie wyważać otwartych drzwi? ... 111
Cel ... 112
Myśl, dyskutuj, uzgadniaj ... 113
Główne obszary zastosowania komponentów ... 114
Szukanie pakietów komponentów ... 114
Im szersze zastosowanie, tym większe korzyści ... 116
Pamiętaj, że świat się zmienia ... 116
Programowanie wieloplatformowe ... 117

Rozdział 8. Programowanie bez zgryzoty ... 119
Czy masz doświadczenie? ... 119
Strategia ... 120
Budowanie zespołu ... 121
Wybór odpowiednich ludzi ... 121
Rekrutacja ... 121
Rozmowy kwalifikacyjne ... 123
Motywowanie pracowników ... 125
Struktura płac ... 125
Wyposażenie ... 126
Szkolenie ... 127
Planowanie i rozdzielanie pracy ... 127
Kontrola i komunikacja ... 127
Biurowe intrygi ... 128
Zwolnienia ... 128
Redukcje ... 129
Pracujący zdalnie ... 129
Awanse ... 130
Poznaj siebie i swoich ludzi ... 130
Utrzymywanie zespołu ... 130

Rozdział 9. Wczesna eliminacja błędów 133
Obsesja jakości 134
Jak podejść do kwestii jakości? 135
Jak klasyfikować błędy? 135
Jak mierzyć jakość? 136
Poprawność 137
Niezawodność 137
Wydajność 138
Integralność 138
Użyteczność 138
Utrzymywalność 139
Elastyczność 139
Testowalność 139
Przenośność 139
Możliwość wielokrotnego zastosowania 140
Interoperacyjność 141
Kwantyfikatory jakości to po prostu wskaźniki 141
Jak mierzyć jakość? 141
Charakterystyka funkcjonalna 142
Punkty funkcyjne 142
Metoda Mark II 143
Punkty charakterystyczne 143
Metryka wybuchowa 144
Punkty charakterystyczne 3D 144
COCOMO 144
Błędy 144
Kontrola wersji 145
Klasyfikacja 145
Długość 145
Liczba i rozmiar błędów 146
Metryka gęstości błędów 146
Dalsza klasyfikacja 146
Jak z góry minimalizować liczbę błędów? 147
Przeczesz swoje plany techniczne gęstym grzebieniem 147
Podziel duży projekt na etapy 148
Bazuj na codziennych konsolidacjach 148
Utwórz dobre środowisko dla komunikacji 148
Testowanie 148
Procedura 149
Plan testów 149
Testy autorskie 149
Sprzęt do testowania 150
Testy wewnętrzne (nieformalne) 150
Wspólne programowanie (nieformalny codzienny nadzór) 150
Kontrola zewnętrzna 150
Testy konsumenckie 151
Jak najwcześniejsze rozpoczęcie beta-testów 151
Monitorowanie funkcjonalności 151

Kryteria publikacji 152
Zakończenie testów 152
Ty, jakość i prawo 153

Rozdział 10. Opłacalne słowa 155
Wskazówki instalacyjne 157
Podręczniki i systemy pomocy 158
Zamiast drukowania 158
Podejście alfabetyczne 160
Podejście funkcjonalne 160
Podejście zadaniowe 160
Czy wszystko zostało opisane? 160
Nie zapomnij o udokumentowaniu samego programu 161
Kilka ogólnych wskazówek na temat pisania 161
Układ tekstu 163
Obraz służy za tysiące słów 164
Drobiazgi są ważne 164
Kiedy już napiszemy... 164

Rozdział 11. Zanim powiesz „Start!" — proces publikacji 165
Publikacja oprogramowania obejmuje wszystko 166
Odliczanie 167
Typy premier i stosowne działania 169
Nowe produkty 169
Nowe wersje 169
Aktualizacja pomiędzy nowymi wersjami 170
Nowe podwersje 170
Problemy związane z publikacją 170
Planowanie dat premier 170
Wybór dnia premiery 171
Czy złote płyty są tylko dla gwiazd muzyki? 172
Udostępnianie oprogramowania na stronie internetowej 172
Kto nad tym panuje? 172
Wstrzymywanie procesu publikacji 172
Co robić, jeżeli w czasie publikacji wystąpią poważne problemy? 173
Szukanie winnych w niczym nie pomaga 174

Rozdział 12. Zakładanie firmy 175
Uniwersalne rozwiązanie? 175
Dlaczego firma? 176
Plusy 176
Minusy 176
Zanim założymy firmę 177
Słowniczek firmowy 178
Ile kosztuje założenie spółki kapitałowej? 180
Profesjonalni doradcy 180
Podstawowe informacje o zakładaniu firmy 182
Kto może założyć firmę? 182
Kapitał zakładowy 182

Co dalej? ... 184
Ilu członków zarządu? ... 185
Udziały i kontrola nad spółką ... 185
Odpowiedzialność spółki ... 186
Odpowiedzialność członków zarządu ... 186
Odpowiedzialność administratora spółki lub sekretarza zarządu ... 187
Prowadzenie księgowości ... 188

Rozdział 13. Wyznaczanie cen ... 191
Teoria cen ... 191
Wyznaczanie odpowiedniej ceny ... 192
Jaki jest rzeczywisty koszt stworzenia programu? ... 192
Rozmiar rynku ... 193
Jaką cenę rynek może i chce zapłacić? ... 194
Bezpośrednia konkurencja ... 194
Ceny produktów podobnych ... 195
Korzyści, jakie odnosi użytkownik ... 195
Jakich oczekujesz zysków? ... 196
Pozytywny wpływ reklamy i działań PR ... 196
Jak produkty towarzyszące mogą zwiększyć zyski? ... 197
Przedziały cen ... 198
Nigdy nie sprzedawaj zbyt tanio ... 198
Inne sposoby osiągania przychodów ... 199
Masa krytyczna ... 200
Udział w rynku ma ścisły związek z ceną ... 200
Sedno ... 201

Rozdział 14. Promowanie produktu ... 203
Marketing ... 203
Pisanie planu marketingowego ... 205
Co takiego jest w nazwie? ... 206
Marka ... 206
Tworzenie wizerunku firmy ... 208
Komunikowanie się z opinią publiczną ... 217
Reklama ... 218
Drobna rada ... 218
Chciałbyś, ale nie możesz ... 219
Reklama jest bardzo uniwersalna ... 219
Wybór mediów ... 220
Operowanie małymi budżetami ... 220
Targi ... 226
Działania public relations ... 226
Imprezy ... 228
Testowe wersje oprogramowania dla dziennikarzy ... 228
Wybieranie rynku ... 229
Pozycjonowanie produktu w sposób zrozumiały ... 230
Wykorzystaj swoje zalety ... 231

Inne możliwości poza sprzedażą 231
Shareware 231
Adware 232
Upgrade Ware 232
Certyfikacja innego producenta 232
Darmowe oprogramowanie 232
Konkursy 233
Kilka wskazówek na zakończenie 233

Rozdział 15. Sprzedaż na dużą skalę 235
Poznajmy uczestników gry 236
Producent 236
Dystrybutorzy 237
Dystrybutorzy internetowi 238
Detaliści 238
Użytkownik 239
Międzynarodowi dystrybutorzy 239
Duże sieci detaliczne 239
Regionalne biura sprzedaży 239
Przedstawiciele handlowi 240
Marketing wielopoziomowy 240
Modele dystrybucji 240
Sprzedaż bezpośrednia 241
Dystrybucja dwuwęzłowa 241
Dystrybucja trójwęzłowa 242
Dystrybucja międzynarodowa 242
Dystrybucja wielopoziomowa 242
Sprzedaż poprzez inne firmy 244
Wybieranie dystrybutorów i detalistów 244
Marże 246
Warunki 246
Podział obowiązków 247
Procedury dystrybucyjne 247
Zarządzanie dystrybucją 250
Czy dystrybutorów i detalistów czeka śmierć? 252

Rozdział 16. Skuteczna sprzedaż 255
Istota sprzedaży 255
Czym tak naprawdę jest sprzedaż? 256
Koszty sprzedaży 256
Jak podejść do sprzedaży? 258
Tok rozumowania klientów 259
Dlaczego relacje z klientami są ważne 260
Podobieństwa skali 261
Marketing wirusowy 262
Toczenie krwi z kamienia 262
Zarządzanie zespołem handlowców 263
Bazowanie na danych 263

Analiza sprzedaży ... 264
Zespoły handlowców ... 265
Po czym poznać potencjalnie dobrego handlowca? ... 265
Nagrody i historie z przestrogą ... 266
Szkolenie ... 268
Akwizycja przez telefon ... 269
Programy mogą odciążać od rutynowych prac ... 270
Wizyty u klienta ... 270
Prezentacje ... 272
Kupowanie lub gromadzenie własnych kontaktów ... 272
Klienci, którzy ulegają ... 273
Poważni potencjalni klienci ... 273
Monitorowanie sprzedaży ... 274
Sortowanie klientów pod względem ważności ... 274
Sprzedawanie poprzez handlowców z innych firm ... 275
Relacje z klientami ... 276
Budowanie dynamiki klienta ... 277
Sprzedaż za pośrednictwem strony internetowej ... 277
Co musi znaleźć się na stronie? ... 278
Przetwarzanie zamówień elektronicznych ... 282
Obsługa opóźnień i zwrotów ... 284

Rozdział 17. Jak utrzymać klientów? ... 285
Co daje wsparcie? ... 285
Punkt widzenia klientów ... 286
Własne możliwości ... 286
Brak wsparcia ... 287
System pomocy ... 287
E-mail ... 287
Grupy dyskusyjne ... 288
Strony internetowe ... 289
Telefon i wideotelefon ... 289
Wizyty u klienta ... 289
Organizacja wsparcia ... 290
Koszty ... 290
Pracownicy wsparcia ... 291
Kiedy należy uruchomić wsparcie? ... 291
Na co zwracać uwagę, wybierając pracowników do zespołu wsparcia ... 291
Kiedy kupić wsparcie od innej firmy? ... 291
Szkolenie pracowników wsparcia ... 291
Angażowanie programistów ... 294
Mierzenie wsparcia ... 294
Pobieranie opłat za wsparcie ... 295
Dbanie o dobre samopoczucie klienta ... 296
Korzystanie z opinii klientów ... 296

Rozdział 18. Ratowanie tonącego statku 299
Ta druga złota reguła 299
Rachunki bankowe 301
Systemy finansowo-księgowe 301
Karty kredytowe 302
Oszustwa dokonywane za pomocą kart kredytowych 302
Korzystanie z usług pośredników 303
Fakturowanie i terminy płatności 303
Kontrola płatności 304
Kryzysy to rzecz normalna 305
Dostrzeganie wczesnych sygnałów ostrzegawczych 306
Przetrwają te firmy, które zareagują wcześniej 306
Prognozowanie 307
Płacenie dostawcom 310
Jak wydawać pieniądze? 310

Rozdział 19. Zarządzanie wzrostem 313
Sukces odsiewa kule u nogi od gwiazd 313
Wybieranie następnego celu 314
Wyciąganie wniosków z własnego sukcesu 315
Prawdziwym czynnikiem napędzającym wzrost są ludzie 316
Zarząd 317
Zmiana biura bez zmiany porządku 319
Rozkręcanie sprzedaży 320
Ekspansja zagraniczna 321
Finansowanie 322
Wnioski 322

Rozdział 20. Gotowość na dalsze sukcesy 323
Pouczająca opowieść 323
Zbyt piękne, by było prawdziwe? 326
Czy wysłali odpowiednią osobę? 326
Czy mają sensowny powód zakupu? 327
Jak rozpoznać konia trojańskiego? 328
Kiedy samemu wykonać pierwszy krok? 328
Jak prowadzić firmę i jednocześnie ją sprzedawać? 329
Zarys umowy 329
Kłopotliwość 330
Gotowość 330
Wycena firmy 332
Jak zmaksymalizować wartość firmy? 333
Umowa 333
Transakcja 333

Dodatek A Optymalizacja strony pod kątem wyszukiwarek (czyli zwycięzca zgarnia wszystko) 335

Krok 1. Poznaj potencjalnych użytkowników strony 336
Krok 2. Opanuj podstawy HTML-a 337
Krok 3. Wybór wyszukiwarek 337
Krok 4. Uczyń stronę przyjazną dla wyszukiwarek 339
Z ramkami czy bez? 340
Strony statyczne czy dynamiczne? 340
Krok 5. Przygotowywanie listy słów kluczowych 341
Kilka zasad stosowania słów kluczowych 343
Krok 6. Indeksowanie strony 343
Element Meta Name="Keywords" 344
Tytuły 344
Strony przygotowywane dla wyszukiwarek 345
Im więcej łączy, tym lepiej 345
Upraszczanie adresu 346
Nie zmieniajmy adresów URL 347
Inne przydatne metaparametry 347
Opis i tytuł strony 347
Zamieszczanie na stronie opinii o produkcie 348
Dodawanie grafiki i animacji 348
Pamiętaj o osobach z wadami wzroku i słuchu 348
Krok 7. Zgłaszamy stronę i monitorujemy efekty 349
Bądź realistą 350

Skorowidz 351

Rozdział 6.
Produkcja oprogramowania

Być może nie wiesz jeszcze, co przesądza o sukcesie w dziedzinie produkcji oprogramowania, ale wiesz już, co rodzi problemy:

- Niedokończone plany.
- Mgliste cele.
- Trywializacja procesu planowania.
- Nierealne terminy.
- Niewystarczające środki.
- Zbyt długie lub zbyt krótkie cykle produkcyjne.
- Nieodpowiedni ludzie w zespole.
- Brak prawdziwego wsparcia z góry.

Mimo tak wielkiej wagi planowania i gromadzenia funduszy to właśnie produkcja oprogramowania jest etapem, na którym firma programistyczna przekształca swoje marzenia w rzeczywistość. Produkcja oprogramowania to jego pisanie i zarządzanie tym pisaniem. Nie może działać, jeżeli harmonogramy, ludzie i techniki nie zostały ze sobą zgrane.

Przemyślany plan o odpowiedniej strukturze znacznie uprości proces produkcji. Po szczegółowym określeniu zasobów, kolejności i kluczowych etapów możliwe jest posuwanie się na przód z w miarę stałą prędkością. Nie trzeba podejmować krytycznych decyzji w połowie drogi.

Bywa jednak, że najprostszy fragment kodu może okazać się najbardziej skomplikowanym. Podobnie może być z produkcją. Jak nakazują reguły, musimy jednocześnie kontrolować cztery sprawy: ludzi, produkt, projekty i proces. Niniejszy rozdział to praktyczna szkółka żonglowania czterema piłkami. Na szczęście, jeżeli będziesz postępował zgodnie z zasadami opisanymi dotychczas w tej książce, większość pracy będziesz miał już za sobą.

Ludzie

Wszystkie opisane wcześniej stanowiska sprowadzają się do tylko dwóch fundamentalnych ról: zarządzania i produkowania. Pierwsza rola to administrowanie procesem, a druga to wykonywanie pracy. Nazwy konkretnych stanowisk różnią się w zależności od skali działalności. Stosowana terminologia też bywa różna w różnych firmach. Obowiązujące obecnie przykładowe nazwy stanowisk związanych z zarządzaniem to:

- menedżerowie projektu,
- dyrektorzy ds. technicznych,
- technolodzy,
- architekci,
- wiceprezes ds. technicznych.

Odpowiedniki po stronie programistów to:

- starsi programiści (*senior developer*),
- programiści,
- testerzy,
- twórcy dokumentacji.

Możliwe jest jednoczesne pełnienie funkcji zarządczej i programistycznej przez jedną osobą albo zajmowanie się więcej niż jednym obszarem zadaniowym. Jednak rzadko zdarza się informatyk biegły w zarządzaniu i odwrotnie — menedżerowie są najczęściej marnymi programistami, chyba że mają dość duże doświadczenie w tym zakresie. Podchodź więc realistycznie do swoich umiejętności. Nie porywaj się z motyką na słońce.

Zdobywanie odpowiednich ludzi

W każdym projekcie programistycznym czas jest na wagę złota. Ostatnią rzeczą, na jaką możesz sobie pozwolić, jest zatrudnienie ludzi, którzy będą go marnować. W pierwszej kolejności powinieneś szukać osób, które znają się na danym typie aplikacji i wszelkich kwestiach z nią związanych. Jeżeli więc tworzysz program do projektowania, to kandydaci, którzy pracowali przy programach graficznych albo byli szkoleni w tym kierunku, są zesłańcami niebios. Jeżeli tworzone przez Ciebie oprogramowanie wymaga szczególnych kompetencji, musisz zatrudnić kandydata z odpowiednim doświadczeniem.

Osoby takie mogą wnieść głęboką znajomość aplikacji i rozwiązań programistycznych, na które ktoś nieobeznany nigdy sam by nie wpadł. Ich doświadczenie nie tylko tworzy fundament do rozwiązywania problemów technicznych, ale także pozwala pisać kod w taki sposób, jakiego oczekuje klient. W ten sposób można uniknąć znacznych poprawek w przyszłości. W innym bowiem razie wydawca oprogramowania będzie musiał przekonać klientów do pracy w programie odstającym od pewnych norm. Kiedy programiści znają dziedzinę, w której aplikacja będzie zastosowana, również o wiele łatwiej implementować usprawnienia dla klientów i odrzucać sugestie, które nie dają ostatecznym użytkownikom żadnej realnej korzyści.

Jeżeli z początku nie jesteś w stanie zebrać załogi takiego kalibru i z takim doświadczeniem, na jakie zasługuje projekt, bądź wytrwały. Nie ruszaj z realizacją, dopóki jej nie znajdziesz. Zawsze lepiej poczekać, aż dostępni będą lepsi, niż próbować zaczynać z osobami niekompetentnymi. Każdy zatrudniony powinien rozumieć, do czego służy produkt i dlaczego jest potrzebny. Jeżeli jakaś osoba nie widzi sensu istnienia takiego produktu albo nie akceptuje go ze strony etycznej lub emocjonalnej, zapraszanie jej do zespołu mogłoby być dla niej krzywdzące.



Zatrudniając poszczególne osoby, zwracaj uwagę, by inwestować pieniądze tam, gdzie wymagają tego Twoje potrzeby.

Menedżer projektu

Już od wczesnego etapu potrzebni są ludzie, którzy czuwaliby nad pracami zespołu i menedżer projektu jest oczywiście taką osobą. Menedżer projektu jest stałym orędownikiem produktu. Jego zadaniem jest nadzór nad pisaniem kodu i budowaniem programu, tak by zgadzał się ze specyfikacją co do natury i treści. W firmie będącej u progu powstania zwykle rolę tę bierze na siebie szef marketingu.

Dobry menedżer projektu to najlepsza gwarancja terminowości realizacji. Osoba ta będzie wiedziała, jak zarządzać projektem i tworzyć harmonogramy realizacji. Oprócz własnego wkładu w pisanie kodu w razie potrzeby pomoże i doradzi programistom w sprawie nowych podejść i technik. Co równie ważne, menedżer projektu sprawi, że wszyscy skupią się na tej samej wizji i powstanie mniej niepotrzebnego kodu.

Menedżer projektu nosi w sobie całą mądrość o tym, do czego służy produkt, jak trzeba go napisać i dlaczego jest taki ważny. Interpretuje firmową wizję produktu na potrzeby programistów w odniesieniu do kompozycji ekranów, komunikatów ekranowych oraz umieszczania logotypu firmy lub produktu. Poza tym pilnuje, by z pokoju programisty wychodziły tylko autoryzowane wersje kodu.

Niektóre firmy wierzą, nie bez powodu, że co bardziej delikatne kwestie tej roli powinny zostać przejęte przez doświadczoną osobę, która nie jest bezpośrednio zaangażowana w codzienny proces pisania kodu. Osoba ta może zajmować się również nadzorowaniem beta-testów.

To, jaką osobę możesz przyciągnąć, w dużej mierze zależy od rozmiaru Twojej organizacji i tego, co jesteś w stanie zaoferować. Poza tym zależy od nieograniczonych szans i nagród związanych z taką pracą.

Po czym poznać dobrego menedżera projektu?

Każdy wartościowy menedżer projektu powinien potrafić przeanalizować projekt, zaplanować jego realizację, dodać coś od siebie i dbać, by wszyscy byli skoncentrowani i zadowoleni. Menedżer projektu powinien być członkiem zespołu i nie stawiać się ponad nim.

Aby spełnić te wymagania, menedżerowie projektu muszą grać na obie strony. Muszą znać i rozumieć proces produkcji oprogramowania (w innym razie nie będą w stanie pomóc programistom), a jednocześnie muszą być dobrymi zarządcami. Co prawda, często

wspomina się o niewyczerpanej energii, umiejętności rozumienia bez słów i takiej umiejętności motywowania, która poprowadziłaby mrówki na Mount Everest, ale oto lista najważniejszych cech, których należy szukać:

- Zorganizowanie.
- Umiejętność rozwiązywania problemów.
- Pozostawanie w stałym kontakcie.
- Odpowiednie podejście do ludzi.

Każdy dobry menedżer techniczny, niezależnie od szczebla, musi być kimś więcej niż teoretykiem. Bardzo ważne jest duże praktyczne doświadczenie z technologiami, jakie musi stosować. Tylko dzięki pracy w roli programisty może znać wszystkie triki i pułapki. Zarządczy aspekt tej pracy wymaga wyuczonych umiejętności oraz naturalnych zdolności. Należy do nich doświadczenie w ocenianiu postępu robót zespołu i negocjacjach. Ci, którzy są dobrzy w koncepcyjnym i strategicznym myśleniu, są naturalnymi przywódcami, pod warunkiem że potrafią się sprawnie komunikować.



Być może zaskakujące jest to, że słabi menedżerowie projektów rzadko są dokładnym przeciwieństwem dobrych. Mało prawdopodobne, że będzie to osoba opryskliwa i niezorganizowana.

Słabi menedżerowie projektów często wypadają dobrze na rozmowach (inaczej nikt by ich nie zatrudnił w tej roli), ale kiedy zaczynają pracować, ich niedoskonałości szybko się ujawniają. Na początku, o ile plan został dobrze skonstruowany, projekt będzie toczył się w zasadzie sam. Menedżera projektu, który stracił grunt pod nogami, dostrzeżesz dopiero wtedy, gdy zaczną się problemy. Zwykle wtedy właśnie okaże się, że nie potrafi on radzić sobie dobrze z ludźmi. Że szuka winnych zamiast rozwiązywać problem. Organizacja zaczyna na to reagować i komunikacja staje się rozluźniona lub niewyraźna. Ludzie zaczynają ukrywać swoje problemy. Menedżer projektu, który jedynie reaguje, najczęściej ponosi porażkę.



Jeżeli okaże się, że masz słabego menedżera projektu i musisz bardzo szybko go zmienić, często najlepszym rozwiązaniem jest tymczasowe promowanie na to stanowisko starszego programisty, który będzie znał projekt i wiedział, co jest do zrobienia. Zapewnij mu jakieś zaplecze administracyjne, aby mógł wykonywać obydwa zadania. Rzadko jest to dobre rozwiązanie na dłuższą metę, ale daje czas na znalezienie zastępcy.

Dwadzieścia lat temu rzadko kto spośród zaangażowanych w projekty programistyczne miał doświadczenie w zarządzaniu projektami. Dziś tak wielu ludzi przeszło przez ten młyn, że nawet małe firmy często są profesjonalnie zarządzane przez ludzi, którzy zaczynali pracować jako zwykli programiści. Pytanie nie brzmi więc, czy możesz jednocześnie zarządzać i pisać kod, ale czy powinieneś robić obie te rzeczy na raz. Odpowiedź brzmi: raczej nie, o ile masz taką możliwość.

Programiści

Im bardziej wymagające jest zadanie do wykonania, tym ważniejsze jest posiadanie dobrych programistów. Gdybyś szukał dobrego architekta, inżyniera lub projektanta, zwracałbyś większą uwagę na to, co zrobili, niż na to, co mówią. Takie podejście sprawdza

się również w odniesieniu do programistów. Przyjrzyj się ostatnim projektom, jakie realizował kandydat na programistę, wybierz fragmenty kodu i poproś o ich objaśnienie. Słabi programiści mogą położyć cały projekt, jeżeli więc programista nie przejawia faktycznie tych umiejętności, jakie deklaruje, zdaj się na instynkt i odrzuć go.

Jeżeli zatrudnisz niekompetentnego programistę (a wielu z nas to się zdarzyło), otrzymasz bezużyteczny kod, atmosfera w zespole się pogorszy, stracisz czas, a budżet rozleci się w pył. Jedyne co możesz zrobić, to jak najszybciej pokazać takiej osobie drzwi, a kod, który napisała, przepuścić przez niszczarkę. Jeżeli trafisz na dobrego programistę, będziesz o tym wiedzieć. Podobnie reszta zespołu.



Jeżeli wybierając programistę, przy podejmowaniu decyzji bierzesz pod uwagę, jaki jest w obyciu, jesteś na dobrej drodze do klęski.

Oto sygnały świadczące o tym, że programista może nie być tak kompetentny, jak twierdzi:

- Twierdzi, że odpowiedzi zwykle pojawiają się same w czasie pisania kodu.
- Sprawiają mu problemy zadania, które wydają się proste.
- Jego praca jest słabo udokumentowana.
- Wolno się uczy.
- Błędami woli zająć się później.
- Nie potrafi podać zadowalających wyjaśnień.
- Elokwencją maskuje słabości swojej pracy.
- Nigdy nie przyznaje się do błędów.
- Pisze kod bardzo wolno.
- Jego kod regularnie nie przechodzi kontroli jakości.

Mimo panującego mitu, że dobrzy programiści to komputerowi maniacy, z którymi nie można się porozumieć, zwykle jest zupełnie odwrotnie. Dzięki wysoce rozwiniętym zdolnościom analitycznym posiadają oni umiejętność przekazywania skomplikowanych idei w prosty sposób oraz rozważania problemów z wielu różnych perspektyw. Dobrzy programiści mają następujące cechy:

- Mają niezwykłą umiejętność strukturalnego planowania.
- Pracują szybko.
- Systematycznie dokumentują postępy.
- Zanim zaczną pisać, mają gotowe rozwiązanie.
- Ich wizja jest oparta na wiedzy.
- Usuwają błędy na bieżąco.
- Rzadko muszą poprawiać swój kod.

- Troszczą się o jakość swojego kodu.
- Wiedzą z doświadczenia, kiedy można zaryzykować.
- Robią to, co do nich należy.

Tabela 6.1 przedstawia stanowiska, jakie powinieneś obsadzić w zależności od wielkości firmy.

Tabela 6.1. *Stanowiska do obsadzenia a wielkość firmy*

Rozmiar firmy	Programiści	Menedżerowie projektu	CTO/technolodzy	Kierownik produkcji	Podwykonawcy
jednoosobowa	tak (Ty sam)				
2 – 12 (mała)	tak	być może			być może
2 – 12	tak	być może	raczej tak		być może
18 – 30 (średnia)	tak	tak	tak (jeden albo drugi)	prawdopodobnie	
40 – 1000 (duża)	tak	tak	tak	tak	prawdopodobnie
1000 + (korporacja)	tak	tak	kilku	tak	prawdopodobnie

Stanowiska kierownicze

Branża IT zapożyczyła stanowisko *chief technical officer* (CTO) od branży badawczej pod koniec lat 70. W tym samym czasie obszar technologii informatycznych znacznie się rozszerzył w miarę pojawiania się nowych produktów, pomysłów i dyscyplin. To spowodowało, że osobie odpowiedzialnej za projekt coraz trudniej było ogarnąć wszystkie aspekty programowania. W konsekwencji rola CTO ograniczyła się do obserwacji. Jego podstawowym zadaniem jest zdobywanie wiedzy na temat produktów i technologii, jakie pojawiają się na horyzoncie, i określanie stawianych przez nie wymagań.

Jak wspomniano w rozdziale 5., inwestorzy cenią wybitnych ekspertów technicznych w firmie, w którą planują zainwestować. Ich zdaniem firmy, którymi kieruje ekspert techniczny, demonstrują wizję i świadomość wagi najnowszej technologii.

Dobrzy CTO to rzadkość, ponieważ osoba taka musi rozumieć teraźniejszość i *przewidywać* przyszłość. Z drugiej strony, zatrudnianie kogoś takiego w pełnym wymiarze nie musi być konieczne. Szukaj osoby z dobrą praktyczną znajomością technologii, jakie mają być stosowane w programie, oraz konsekwencji ich stosowania. Ktoś taki powinien umieć poprowadzić Cię przez te obszary nowoczesnych technologii, w których sam masz ograniczoną wiedzę. Powinien również pomóc wykorzystać te techniki, które dopiero poznałeś.

CEO Engineering

W dużych projektach stanowisko to często obejmuje ważna osoba, która nadzoruje infrastrukturę techniczną komponentów dla zachowania wewnętrznej zgodności.

Kierownik produkcji

Kierowników produkcji można zwykle spotkać w organizacjach, które pracują jednocześnie nad kilkoma programami. Zwykle podlegają szefowi wydziału lub dyrektorowi generalnemu. Niestety, niewiele jest osób z doświadczeniem na tym stanowisku, więc istnieje tendencja do promowania każdego, kto zdaje się nadawać do tej roli. Wielu kierowników produkcji zaczyna pracę na tym stanowisku ze stosunkowo niewielkim doświadczeniem. Menedżera produkcji czasami nazywa się *CEO Engineering*, ale jego rola jest szersza.

Inżynierowie systemów, architekci lub technolodzy

Nazwy tych stanowisk są co prawda często stosowane zamiennie, ale z reguły inżynierowie systemów i architekci skupiają się na szkielecie rozwiązania, a technolodzy specjalizują się w funkcjach, które wypełniają poszczególne części.

Podwykonawcy

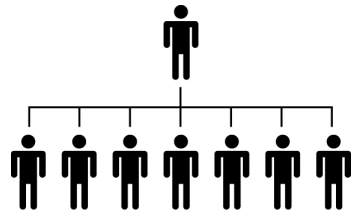
Nie ma sensu zatrudniać na stałe osoby, której umiejętności będzie można wykorzystać przez stosunkowo krótki okres. Do takich podprojektów lepiej wynająć podwykonawców. Warto zapamiętać szczególnie dobrych na potrzeby przyszłych projektów.

Zabezpiecz się

Zatrudnij świetnych ludzi, a otrzymasz realną szansę, by stać się gigantem. Ludzie rzadko odchodzą ze świetnych zespołów. Ale nie możesz być tego pewnym. Jeżeli Twój projekt jest krótkotrwały, a jego realizacja musi być szybka, całkiem uzasadnione jest poproszenie osoby, która ma dla Ciebie pracować, o to, by nie zmieniała pracy w trakcie realizacji projektu. W miarę możliwości wyjaśnij, że jest to w interesie wszystkich zaangażowanych. Jeżeli np. przewidywana jest premia w wysokości 5000 dolarów dla każdego członka zespołu, gdy projekt zostanie zrealizowany w terminie, i jedna osoba odejdzie do pracy, gdzie oferują o 10 000 dolarów rocznie więcej niż obecnie, zyskuje tylko 5000 dolarów i dożywotnią urazę ze strony byłych współpracowników.

Małe jest piękne

Duże zespoły nie pracują. One tyrają. Głównie dlatego, że tyle uwagi poświęca się temu, by wszyscy byli na bieżąco i wiedzieli, co się dzieje. Zarządzanie zajmuje im więcej czasu niż produkcja. O wiele lepiej stworzyć kilka mniejszych zespołów niż jeden duży moloch. Podziel więc zadanie, gdzie tylko się da, na odrębne zespoły składające się z od trzech do siedmiu osób. Praktyka wskazuje, że dziewięć osób w zespole to maksymalna liczba, jaką powinieneś brać pod uwagę. Każdy zespół w ramach projektu powinien mieć własnego menedżera, który współpracuje z innym grupami i kontaktuje się z „górą". Pożądaną strukturę zespołu przedstawiono na rysunku 6.1.



Rysunek 6.1. *Struktura zespołu odzwierciedlająca stosowną liczbę członków i menedżera*



Być może to szokujące, ale branża programistyczna ma już blisko pół wieku. Wielu z pierwszych innowatorów, takich jak dr Wang, który pomógł stworzyć pamięć RAM, już od nas odeszło. Budując zespoły, należy pamiętać, że młodość wnosi energię, a wiek doświadczenie. Starsi programiści wiedzą, jak uniknąć błędów, więc nie zapominaj o nich.

Proces

Proces to mechanizm przejścia od początku do końca pisania programu. Niemal każdy ma własną teorię procesu. Najważniejszy jest wybór metody postępowania, która daje największe szanse na powodzenie. Wiele składowych typowo kojarzonych z procesem, takich jak wykonalność projektu, ocena ryzyka, struktura zespołu i harmonogramy wykonania, zostało ustalonych już w planie projektu.

W praktyce proces to często następstwo kolejnych wydań, które składają się na życie produktu. Proces początkowy daje pierwszą wersję programu, a kolejne procesy dają wersje następne. Zaletą myślenia w kategoriach procesu jest to, że po jego zdefiniowaniu wszyscy rozumieją swoje obowiązki i wiedzą, czym zajmują się poszczególne osoby.

Złota reguła dotycząca rozpoczęcia produkcji brzmi: nie śpiesz się. Przejrzyj plan projektu i sprawdź uważnie, czy coś wymaga aktualizacji. Wiele rzeczy może wyniknąć od czasu powstania planu, do chwili gdy wszyscy są gotowi, by przystąpić do pracy. Warto je więc wyłowić. Pewne zmiany, które wymagają dokonania korekt, będą jasne jak słońce. Możemy mieć ośmiu ludzi do zadania przewidzianego dla dziesięciu. Czas na wykonanie może się skurczyć o kilka miesięcy. Może się okazać, że w kasie jest mniej pieniędzy, niż zakładałeś. Tego rodzaju rzeczy łatwo zauważyć, ale upewnij się też, czy nie nastąpiły zmiany w drobnych i z pozoru niegroźnych aspektach.



Przeczytaj uważnie wszystkie umowy, jakie podpisałeś. Jeden ze sponsorów pewnego projektu realizowanego dla administracji publicznej dodał na końcu bardzo długiego planu następujące zdanie: „Projekt musi zostać zrealizowany przy użyciu następującego oprogramowania…". Wstawienie tego zdania bez informowania kogokolwiek nie było mądrym posunięciem. Cała architektura rozwiązania musiała zostać przebudowana.

Standardowym środkiem ostrożności jest wysłanie przez menedżera projektu do wszystkich członków zespołu e-maila z zapytaniem, czy wiadomo im o jakichkolwiek różnicach pomiędzy pierwotnym planem projektu a bieżącą sytuacją, poza tymi, o których już wszyscy wiedzą (tutaj powinna zostać umieszczona lista znanych zmian). Nie zaczynaj realizacji, dopóki nie otrzymasz jednoznacznych odpowiedzi od wszystkich członków zespołu i nie uzgodnisz ostatecznie, co należy zrobić.

Negocjowanie zmian

Do zmian należy podchodzić z wyjątkową ostrożnością. Każdy projekt będzie podlegał zmianom. Ostateczna wersja będzie się różniła od pierwotnej specyfikacji, choć im lepszy jest plan projektu, tym mniej zmian zwykle jest wymaganych. Traktuj spóźnione dobre pomysły ostrożnie. Jeżeli będzie widoczne, że akceptujesz każdą zmianę, każdy będzie chciał dołożyć swoje trzy grosze. Jeżeli dopuścisz do swobodnego wprowadzania zmian, bardzo szybko przekonasz się, że kręcisz się w kółko.

Jeśli pomysł ma pierwszorzędne znaczenie, rozważ jego wpływ na architekturę, funkcjonalność, procedury, harmonogramy i budżet. Poproś każdego o ocenę skutków zmian, zanim zrobisz krok dalej. Najlepsza praktyka to odłożyć takie pomysły do czasu opracowania następnej wersji.



Zarząd często czuje, że ma prawo do wprowadzania zmian, i tak też robi. Jednak na tej samej podstawie wszyscy inni mają prawo wnieść sprzeciw.

Jeżeli okoliczności niezależne od firmy uległy znacznej zmianie od chwili zatwierdzenia planu, wprowadzanie przypadkowych zmian nie jest dobrym pomysłem. Najbardziej uzdrawiającą rzeczą, jaką możesz zrobić, jest zebranie wszystkich zainteresowanych stron, wyjaśnienie, co zaszło, i poproszenie o pomoc. W końcu wszyscy zaangażowani mają wspólne cele. W tym gronie powinno dać się wypracować najlepszy sposób reakcji na zmiany, które zaszły. Takie rozwiązanie jest o wiele lepsze niż ogłoszenie, że zatrudnionych zostanie mniej programistów, którzy nie będą mieli dodatkowego czasu na realizację. Menedżerowie działający pod naciskiem nie zawsze przywiązują wagę do ubocznych efektów swoich decyzji.

O wiele lepiej usiąść wspólnie i omówić wszystkie implikacje. Być może znajdzie się jakieś mniej bolesne rozwiązanie. Jeżeli nie, przystań tylko na to, co Twoim zdaniem jest rzeczywiście możliwe. Jeżeli Twoje uwagi nie są przyjmowane, zachowaj uprzejmość, spokój i notuj zastrzeżenia. Wyjaśnij, w jaki sposób mogą one zagrażać projektowi, a potem miej już tylko nadzieję, że byłeś w błędzie.

Z dala od problemów

W kolejnych punktach omówione zostaną szczegóły strategii zapobiegania problemom, które mogą pojawić się w trakcie realizacji projektu.

Zarządzanie ryzykiem

Nierealne jest oczekiwanie, że dobrze zaplanowany projekt nie pociąga za sobą żadnego ryzyka. Nie w informatyce. Zarządzanie ryzykiem ma wiele postaci, czego dowodzą kolejne punkty. Zanim zaangażujesz się w programowanie, warto uświadomić sobie, co jest niepewne, i wiedzieć, jak obserwować te rzeczy. Jedynym sposobem na radzenie sobie z ryzykiem jest jego akceptacja, pomiar i stawienie mu czoła.

Zapas czasu i pieniędzy

Opóźnienia, które można wyrazić w jednostkach czasu, równie dobrze można wyrazić w kategoriach pieniężnych i właśnie dlatego każdy członek zespołu musi pilnie obserwować piasek przesypujący się w złotej klepsydrze. Księgowi mierzą każdy wydatek. Szef firmy interesuje się efektem końcowym, także w kwestii finansów. Menedżer projektu pilnuje, by nie przekroczyć granic czasowych i kosztowych. Pracownicy pilnują, by ich wydatki nie przekroczyły dopuszczalnych limitów i by wykonać pracę na czas. Przy realizacji projektów programistycznych każdy musi kontrolować dostępne zasoby.

Oto pięć najczęstszych przyczyn przekraczania budżetu projektu:

- Zmiany w specyfikacji.
- Spóźnione uświadomienie sobie, czego tak naprawdę wymaga projekt.
- Ludzie się zwalniają i trzeba ich zastąpić.
- Wymagane są dodatkowe technologie w zakresie oprogramowania.
- Szacunki były błędne.

Konsekwencje przekroczenia budżetu mogą budzić grozę: szanse na sprzedaż zostają zaprzepaszczone, kluczowe funkcje muszą zostać porzucone albo projektu trzeba zaniechać. W najgorszym scenariuszu wiąże się to również z bankructwem firmy. Każdy członek zespołu, od najmłodszego do najstarszego, musi uświadomić sobie, jak ważne są pieniądze, jak poważne są konsekwencje nadmiernych wydatków i że nawet najbogatsze firmy nie mają nieograniczonych środków.

Starsi rangą menedżerowie boją się ujawniać rezerwy budżetowe, ponieważ sądzą, że menedżerowie projektu i kierownicy zespołów uwzględnią je w swoim myśleniu. Menedżerowie powinni jednak być równie ostrożni, jeżeli mają poprowadzić projekt o absolutnie sztywnym budżecie. Ledwo jeden na sześć projektów jest realizowany na czas i mieści się w budżecie. Dlatego rozpoczynanie projektu bez pozostawienia sobie pola manewru jest – jak pokazuje statystyka – samobójstwem.

Niezależnie od tego, czy Twoją silną stroną jest zarządzanie czy programowanie, musisz umieć zarządzać swoim czasem. Nieraz jest to trudniejsze niż zarządzanie innymi. Sztuczka polega na tym, by stworzyć krótką listę rzeczy, jakie zamierzasz dziś zrobić, zanim jeszcze odbierzesz pocztę. Przygotuj grunt dla możliwie wydajnej pracy i poświęć jej jak najwięcej czasu w ciągu dnia.

Stwórz krótką listę podobną do następującej:

- Dziś zebranie — czy pozostały jakieś niezałatwione sprawy?
- Co musisz zrobić dzisiaj i czy pozostały jakieś sprawy z poprzedniego dnia?
- Czy trzeba z kimś formalnie porozmawiać? Jeżeli tak, umów spotkanie.
- Czy są jakieś prace administracyjne do wykonania?
- Czy praca idzie zgodnie z harmonogramem?

Likwidowanie czynników rozpraszających

Kiedy wszyscy uświadomią sobie, że każde przerwanie pracy programiście sprawia, że traci on pół godziny, szybko przestaną przeszkadzać. Zapewniając programistom odpowiednią przestrzeń do pracy (omawianą w rozdziale 4.), zapobiegamy spontanicznym pogawędkom. Jednak takie starania są zupełnie bezużyteczne, jeżeli ludzie i tak będą je pośrednio atakować. Oto początkowy zbiór reguł dla programistów:

- Zanim zaczniesz pracować, wyłącz telefon komórkowy.
- Wyłącz wszelkie komunikatory internetowe.
- Sprawdzaj pocztę tylko w czasie przerw.
- Każdy, kto musi się z Tobą pilnie skontaktować w czasie godzin pracy, powinien zadzwonić pod numer stacjonarny.
- Inni członkowie zespołu, którzy chcą się z Tobą skontaktować, powinni to robić tylko za pośrednictwem menedżera projektu.

Co robić, gdy programiści się opóźniają?

Często jest to pierwsza oznaka kłopotów. Czasami programista dokładnie wie, na czym polega problem, ale nie wie, jak z niego wybrnąć. Przeszkody przy pisaniu kodu to coś, co trudno zdefiniować. Nie są to jakieś widoczne obiekty z nalepioną dużą etykietą „kłopoty". Zwykle jest to fragment, który po prostu nie chce zadziałać tak, jak trzeba. Często upływa trochę czasu, zanim programista zda sobie sprawę, że może nie być w stanie rozwiązać problemu własnymi siłami.

Dobra zasada mówi, że jeżeli programista poświęcił więcej niż pół godziny, poruszając się drogą donikąd, powinien zastosować plan B, nawet jeżeli zakłóci tym pracę kolegów. Najpierw powinien omówić problem z menedżerem projektu — od tego właśnie jest menedżer. Jeżeli menedżer projektu nie potrafi rozwiązać problemu, powinni we dwoje ustalić, kto najprawdopodobniej będzie znał odpowiedź (osoba z zespołu lub spoza organizacji). Jeżeli to nie da efektów, powinni wysłać zapytanie na grupy dyskusyjne. Odpowiedź może pojawić się zaskakująco szybko, ale często trzeba poczekać nawet do czterech godzin. Jeżeli programista zdecyduje się na taki krok, oczywiście w międzyczasie zajmie się czymś innym, przerywając co godzinę, by sprawdzić, jak przebiega internetowa burza mózgów.



Poczucie wstydu i zażenowania samo w sobie nie przełamie blokady. Pomoże w tym za to rozmowa z kimś innym. Zawsze kiedy ktoś utknie, należy pamiętać, że co dwie głowy to nie jedna.

Niech menedżer projektu będzie pasterzem

Nie ma nic gorszego niż programista, który utknął i w związku z tym znalazł w Internecie fragment kodu pozwalający obejść problem, wstawił go do programu i pisał dalej. Nie sposób wymienić wszystkich implikacji związanych z jakością, kodem źródłowym, wsparciem, aktualizacją i prawami autorskimi, nie mówiąc już o opłatach licencyjnych.

Nie można oczekiwać od nowych programistów, że będą zdawali sobie z tego sprawę. Szukanie rozwiązań w Internecie to czarna magia, która kosztuje wiele czasu. O wiele lepiej, jeżeli programista powie menedżerowi projektu, na czym polega problem, a ten znajdzie, sprawdzi i oceni wszelkie potencjalne rozwiązania. Taki problem to doskonały temat do poruszenia na jednym z cotygodniowych zebrań. Bardzo możliwe, że wśród tylu zgromadzonych znajdzie się osoba, która po prostu zna już stosowane rozwiązanie. Może się przy tym okazać, że Twoja organizacja jest w posiadaniu praw autorskich do jego wykorzystania.

Prowadzenie za rękę

Mimo że niewielu menedżerów projektu to robi, jedno z ich najważniejszych zadań polega na prowadzeniu programistów za rękę wtedy, gdy utkną. Skuteczny menedżer musi znać się na językach i technikach programowania, jakie są stosowane. Wówczas może pracować wspólnie z programistą, pozwolić mu omówić problem, możliwe rozwiązania i wspólnie znaleźć odpowiedź.

Nie obawiaj się podzielić wówczas pracą i napisać część kodu samemu. Poza tym, że pogłębiasz tym sposobem więź z zespołem, pracujesz nad zwiększeniem wzajemnego zrozumienia i szacunku, unikając zagrożenia terminu realizacji trudnego fragmentu projektu.

Reagowanie na złe wieści

Największym wyzwaniem dla Twojej samodyscypliny jest sposób, w jaki reagujesz na złe wieści. Jeżeli za każdym razem, gdy ktoś wyznaje, że opóźnił cały zespół o jednej dzień, zamieniasz się w wulkan grożący wybuchem, doprowadzisz jedynie do tego, że każdy będzie instynktownie zwlekał ze zgłoszeniem problemu. Będąc dalecy od zrozumienia dla Ciebie, mogą czuć, że Twoje reakcje są bardziej przygnębiające niż same złe wieści.

Uzyskasz o wiele większe wsparcie, jeżeli weźmiesz głęboki oddech i zastanowisz się nad konsekwencjami, bo właśnie o tym myśli cała reszta. Nie sugeruję, że nie powinieneś później dać upustu swojemu zdenerwowaniu, ale lepiej, szczególnie jeżeli jesteś przywódcą zespołu, pomóc innym pogodzić się z niepowodzeniem. W ten sposób zyskasz duży szacunek.

Bicie na alarm

Programowanie, najkrócej mówiąc, jest jak jednotorowa linia kolejowa. Jeżeli ktoś utknie w jakimś punkcie, często reszta zespołu ma problemy z ruszeniem do przodu. Dlatego ważne jest, by przeszkody usuwane były w najkrótszym możliwym czasie. Nie myśl, że jeżeli osoba, która natrafiła na problem, będzie udawać, że nie jest to problem, to ten sam zniknie albo odpowiedź sama się znajdzie. Próby prowizorycznego załatwienia problemu zwykle pogarszają tylko sprawę. Gdy tylko ktoś natrafi na problem, powinien bić na alarm i prosić o pomoc kolegów, prowadzącego projekt lub kogokolwiek, kto może służyć radą. Jeżeli nikt nie zna odpowiedzi, trzeba jak najszybciej wysłać zapytanie na grupy dyskusyjne. Nie obawiaj się szukać pomocy poza własną organizacją. Jeżeli odpowiedź istnieje (a przeważnie istnieje), to ktoś na pewno ją zna.



Jeżeli spojrzysz na harmonogram, powinieneś widzieć, czym masz zająć się następnego dnia. Jeżeli jest to coś, z czym nie jesteś obeznany, wyślij wieczorem na grupy dyskusyjne pytanie o ogólne rady i najlepsze rozwiązania problemów, które Cię trapią. Następnego ranka zwykle znajdziesz szereg rad i gotowych rozwiązań. W ten sposób nie stracisz całego dnia.

Szacowanie czasu

Nie wszystko dzieje się zgodnie z harmonogramem. Stosowanie analogii do innych dziedzin, takich jak drukarstwo, hydraulika czy ręczne kalkulacje, nie pozwoli nam ustalić, ile zajmie napisanie kodu, który komputeryzuje jakąś czynność. Coś, co jest łatwe do opisania, może być trudne do zaprogramowania, a coś, co wydaje się ogromnym wyzwaniem, może okazać się błahostką. Na przykład program zliczający wszystkie słowa, jakich użył Szekspir w swoich dziełach, można sprowadzić do trzech wierszy kodu. Szekspir użył 29 066 słów. Z kolei napisanie i zapełnienie danymi formularza WWW, który używa wariantowych list rozwijanych pozwalających wprowadzić nazwę kraju, miasta i ulicy, zajmie doświadczonemu programiście osiem godzin pracy.

Programiści często potrafią podać przybliżony czas wykonania różnych rzeczy przy założeniu, że kod będzie działał zgodnie z planem bez żadnych komplikacji. Jednak kod, który napisali, może mieć wpływ na inną procedurę programu. Dotarcie do przyczyny może być bardzo trudne. Tak więc, o ile czasem możliwe jest wykonanie całodniowej pracy w minutę, innym razem minuta pracy rozciąga się do całego dnia. Monitorowanie i zarządzanie działalnością jest jedynym możliwym mechanizmem jej kontrolowania.

Nadawanie tempa

Największa różnica pomiędzy zespołami programistów, które odnoszą sukcesy, a tymi, którym się nie udaje, polega na tym, że zwycięski zespół ma odpowiednie tempo pracy. Zespół programistów, grupowo i indywidualnie, wyznacza cele do osiągnięcia na każdy dzień. Każda osoba zobowiązuje się wykonać określoną ilość pracy do końca dnia. Pięć minut przed końcem dniówki kierownik produkcji pyta każdą osobę w obecności innych, jak jej poszło. Odpowiedzi są odnotowywane, ale nie padają żadne komentarze, nawet pochwały. Sytuacja ma mówić sama za siebie. Kierownik produkcji prosi wówczas wszystkich o zastanowienie się do następnego ranka, co zamierzają zrobić jutro. Następnego dnia w pierwszej kolejności prosi każdego członka zespołu o podanie celów w obecności innych. Cele te nie mają być ambitne. Nie chodzi tu o bicie rekordów tylko o sukces całego zespołu. Najważniejsza jest cicha determinacja.

Co prawda każdy członek zespołu będzie chciał pokazać swoim kolegom, że przykłada się do pracy, ale cel, jaki sobie wyznacza, musi być osiągalny. Kierownik produkcji na tym etapie okaże pomoc każdemu, kto opóźnia się w stosunku do własnego harmonogramu. W uzasadnionych przypadkach może poprosić innych o pomoc.

Inteligentne zespoły zdają sobie sprawę, że zahamowanie tempa jest tylko kwestią czasu. Dlatego powinny od początku wypracować zapas czasu, dając z siebie więcej i starając się upchać dodatkowe 15 minut pisania kodu w siedmiogodzinnym dniu pracy i tym sposobem uzyskiwać odpowiednią przewagę.

Kluczowe znaczenie dla utrzymywania tempa ma przygotowanie etapów realizacji projektu. Bez tego, niezależnie od stopnia trudności, każdy program będzie pisany dłużej jako całość niż jego poszczególne etapy z osobna. Istnieje tak zwany czynnik SNZ (strach, niepewność, zwątpienie), który trapi szczególnie nowych i niedoświadczonych członków zespołu. Dlatego menedżerowie projektów dzielą projekt na kilka niezależnych etapów. Pierwszy podział opiera się na funkcjach programu. Następnie główne etapy dzieli się na rozsądne części możliwe do wykonania przez tydzień. Dalszy podział pozostawia się już samym programistom. Na przykład wykonanie strony witryny internetowej może zostać podzielone na następujące etapy:

1. Uzgodnienie zawartości.
2. Stworzenie modelu proponowanego interfejsu.
3. Wybór technologii.
4. Zaprojektowanie szablonu.
5. Zatwierdzenie szaty graficznej.
6. Napisanie kodu HTML.
7. Testowanie.

Następnie możemy monitorować postęp realizacji kolejnych etapów, w miarę jak zespół nabiera pewności, radzi sobie z wszystkimi wyzwaniami i kończy realizację poszczególnych kroków.

W efekcie, dzięki zwiększonemu wysiłkowi podjętemu wcześniej, szybciej otrzymasz gotowy produkt.

Monitorowanie postępu

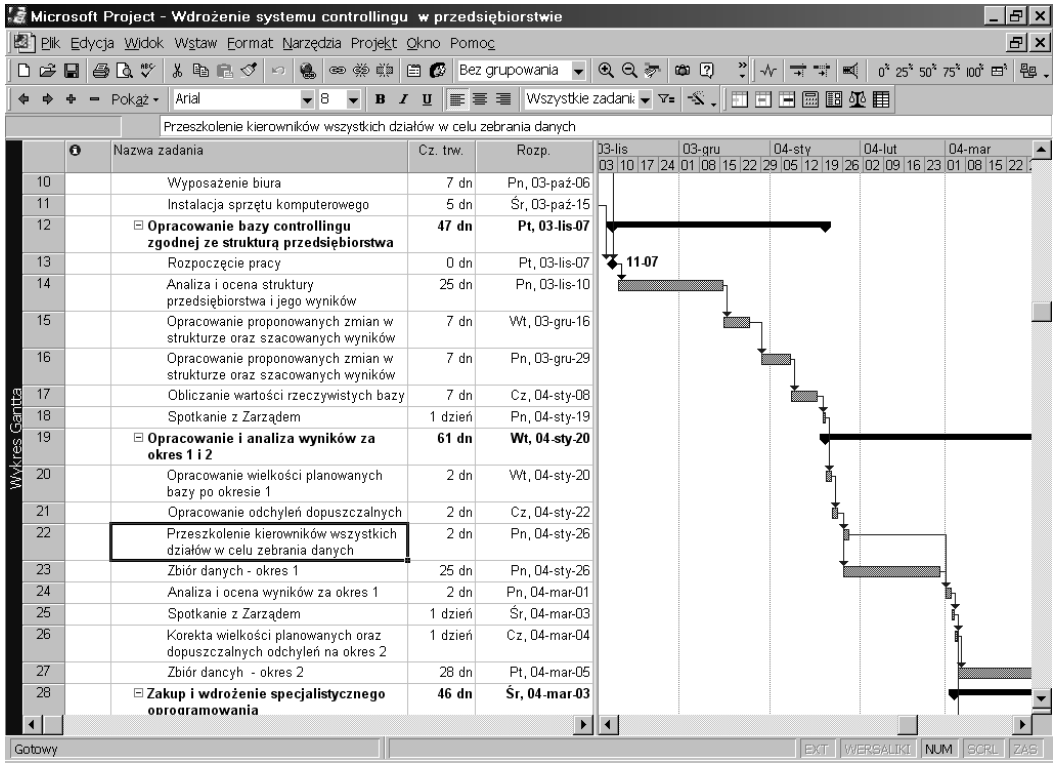
Jak kapitan statku masz w rękach mapę nawigacyjną. Mapa ta zatytułowana jest „Plan produkcji". Wiesz, skąd wyruszasz, dokąd chcesz dotrzeć, i znasz drogę, którą trzeba podążać. Aby śledzić postęp, musisz każdego dnia zaznaczać bieżącą pozycję na mapie. Nawet jeżeli pracujesz sam, pamiętaj o oznaczaniu kluczowych punktów, kiedy tylko uda się je minąć. Zwiększa to komfort psychiczny i przypomina Ci o postępach, jakie czynisz.

Najlepiej stworzyć dwie mapy postępu — ogólnie dostępną, która prezentuje postęp w realizacji głównych etapów, i szczegółową, dostępną tylko dla członków zespołu. Mapę tę można aktualizować na koniec każdego tygodnia i posługiwać się nią jako wizualną pomocą na poniedziałkowych porannych zebraniach (patrz rysunek 6.2).



Jeżeli nie masz pewności, w jaki sposób najlepiej monitorować postęp, polecam standardowe wykresy Gantta. Większość ludzi zna ten format, nawet jeżeli nie wie, jak się nazywa, i jest on bardzo łatwy do zrozumienia.

Panuje przekonanie, że programiści nie cierpią raportów o postępach, ponieważ pokazują one, jak bardzo są spóźnieni. Dobrzy programiści nigdy nie są tacy krótkowzroczni. Dzięki tym raportom każdy może mieć świadomość bieżącego postępu w realizacji projektu. Praca bez raportów przypomina rejs do Ameryki bez codziennego zaznaczania pozycji statku.



	Nazwa zadania	Cz. trw.	Rozp.
10	Wyposażenie biura	7 dn	Pn, 03-paź-06
11	Instalacja sprzętu komputerowego	5 dn	Śr, 03-paź-15
12	**Opracowanie bazy controllingu zgodnej ze strukturą przedsiębiorstwa**	**47 dn**	**Pt, 03-lis-07**
13	Rozpoczęcie pracy	0 dn	Pt, 03-lis-07
14	Analiza i ocena struktury przedsiębiorstwa i jego wyników	25 dn	Pn, 03-lis-10
15	Opracowanie proponowanych zmian w strukturze oraz szacowanych wyników	7 dn	Wt, 03-gru-16
16	Opracowanie proponowanych zmian w strukturze oraz szacowanych wyników	7 dn	Pn, 03-gru-29
17	Obliczanie wartości rzeczywistych bazy	7 dn	Cz, 04-sty-08
18	Spotkanie z Zarządem	1 dzień	Pn, 04-sty-19
19	**Opracowanie i analiza wyników za okres 1 i 2**	**61 dn**	**Wt, 04-sty-20**
20	Opracowanie wielkości planowanych bazy po okresie 1	2 dn	Wt, 04-sty-20
21	Opracowanie odchyleń dopuszczalnych	2 dn	Cz, 04-sty-22
22	Przeszkolenie kierowników wszystkich działów w celu zebrania danych	2 dn	Pn, 04-sty-26
23	Zbiór danych - okres 1	25 dn	Pn, 04-sty-26
24	Analiza i ocena wyników za okres 1	2 dn	Pn, 04-mar-01
25	Spotkanie z Zarządem	1 dzień	Śr, 04-mar-03
26	Korekta wielkości planowanych oraz dopuszczalnych odchyleń na okres 2	1 dzień	Cz, 04-mar-04
27	Zbiór dancyh - okres 2	28 dn	Pt, 04-mar-05
28	**Zakup i wdrożenie specjalistycznego oprogramowania**	**46 dn**	**Śr, 04-mar-03**

Rysunek 6.2. *Przykład harmonogramu realizacji projektu*

Jak kontrolować

Pamiętaj, że nigdy nie otrzymujesz tego, czego oczekujesz, tylko to, co sam sprawdzisz. Przegląd postępów i rezultatów powinien następować pod koniec każdego dnia pracy. Jeżeli z jakiegoś powodu nie udało się tego zorganizować, przyjrzyj się dokładnie następnego ranka, co zostało osiągnięte dzień wcześniej. Nigdy nie pomiń przeglądu dwa dni z rzędu.

Programiści będą bardziej szczerzy, jeżeli okażesz im wsparcie. Nie zawsze dadzą Ci powody do zadowolenia, choć będą chcieli. Dlatego zawsze zwracaj się do nich w swobodny sposób. Jeżeli wiesz, że pracują akurat nad czymś skomplikowanym, uzgodnij wygodny dla nich czas spotkania. W pozostałych przypadkach po prostu dostaw sobie krzesło i zacznij codzienną pogawędkę. Interesują Cię trzy rzeczy:

- Czy w jakiś sposób możesz im pomóc?
- Czy mają jakieś problemy?
- Czy wykonali pracę przewidzianą na dzisiaj?

Jeżeli dana osoba pracuje w domu, zadzwoń do niej i porozmawiaj, poproś o wysłanie e-maila ze zmianami, które wymagają sprawdzenia, albo połącz się zdalnie z jej komputerem, by zobaczyć, nad czym pracuje. Niezależnie od tego, czy Twoi ludzie pracują

zdalnie, czy na miejscu, nie ograniczaj się do rozmowy. Przeglądaj kod i proś o zademonstrowanie, jak działa. Komentuj. Udzielaj zasłużonych pochwał. Zanim się pożegnasz, omów zadania na jutro.

Dzięki codziennym przeglądom Twoi programiści nigdy nie będą musieli wyznawać, że mają miesiąc opóźnienia. Najdłuższe opóźnienie, o jakim możesz usłyszeć, to jeden dzień.

Cotygodniowe zebrania

Produkcja oprogramowania to praca zbiorowa, ale sam akt programowania to samotne działanie. Regularne cotygodniowe spotkania to najlepszy sposób na podtrzymanie ducha zespołu, dbanie o to, by wszyscy byli poinformowani, omówienie bieżących problemów i przegląd postępów. Jest to również okazja do dyskusji na bardziej ogólne tematy. Cotygodniowe spotkania to dla programistów idealna okazja do zadawania pytań, a dla menedżerów do poinformowania wszystkich, jakie wystąpiły problemy.

Zebrania najlepiej odbywać na początku pracy w poniedziałek rano. Zespół będzie czuł się świeżo po weekendzie, który daje też czas na przemyślenie tego, co osiągnęło się w zeszłym tygodniu.

Burze mózgów

Jeżeli w zespole nie ma żadnych wybitnych myślicieli, rozwiązaniem może być burza mózgów. Najlepiej gdyby udało się zebrać cały zespół i obecny był ktoś z marketingu, aby był świadomy postępu i wyjaśnił kwestie, które mają wpływ na poziom sprzedaży. Warto zaprosić również blisko mieszkających pracujących zdalnie. Jeżeli mieszkają dalej, zorganizuj telekonferencję lub zarezerwuj im bilety lotnicze. Wszystko w miarę zdrowego rozsądku.

Najważniejsze w burzy mózgów jest to, że żaden pomysł nie jest zły. Prowadzący może zaakceptować najdziwniejsze idee. Ewentualnie może poprosić o wyjaśnienie. Wszelkie dalsze nagabywanie zwykle nie sprzyja efektywności.

Każdy pomysł należy zapisać w zwięzły i jasny sposób bez opatrywania go komentarzami. Argumenty przeciwko pomysłom zwykle mówią same za siebie, a w luźnej atmosferze jedna myśl prowadzi do drugiej. Menedżer projektu powinien przejrzeć protokoły, podpisać je i rozdać wszystkim zainteresowanym.

W przypadku większych projektów korzystny jest krótki raport na koniec każdego tygodnia, odnotowujący postępy i problemy. Takie raporty, zwykle przygotowywane przez menedżerów projektu, nie muszą być zbyt formalne i mogą być albo rozdawane, albo rozsyłane e-mailem.

Walka z bałaganem

Bardzo często programiści natrafiają na mur. Z jakiegoś powodu program nie akceptuje kolejnej inkrementacji i nikt nie jest w stanie stwierdzić dlaczego. Wówczas konieczny jest powrót do poprzedniej wersji. W trakcie pisania programu coś takiego może się zdarzyć wiele razy.

Aby uniknąć niepotrzebnego zamieszania, warto dysponować dobrym programem do kontroli wersji oprogramowania. Taki program zachowuje stare kopie, indeksuje zmiany i śledzi, kto co zrobił i kiedy. Ten elektroniczny archiwista pozwala analizować wcześniejsze wersje programu i wyszukiwać różnice między poszczególnymi wersjami. Jest szczególnie przydatny w sytuacji, kiedy nie potrafisz dotrzeć do tego, co sprawiło, że program zwariował.

Najprostsze programy do kontroli wersji przechowują jedynie wskazane pliki i zarządzają nimi. Bardziej wyrafinowane umożliwiają jednoczesne wprowadzenie zmian, automatyzują konsolidację, wymuszają testy regresyjne, a czasem nawet tworzą i kompilują gotowy produkt. Niezależnie od wymaganego stopnia złożoności, nie zaczynaj pracy, nie instalując wcześniej oprogramowania do kontroli wersji. Zadbaj też, by Twoi podwykonawcy i pracownicy zdalni dysponowali kompatybilnymi wersjami.

Regularne konsolidacje

Im szybciej zaczniesz testować produkt, tym szybciej wykryjesz i usuniesz usterki. Im szybciej programiści dowiedzą się, że ich kod integruje się i działa, tym szybciej możliwe stanie się wprowadzenie produktu na rynek. Sprytni inżynierowie tworzą projekt w taki sposób, by dopuszczał on montaż całości z gotowych prefabrykatów. Dzięki temu mogą uruchomić produkt w fazie zarodkowej tak szybko, jak to tylko możliwe. Kolejną zaletą tej techniki jest możliwość zademonstrowania klientom i zarządowi pierwszych działających wersji. 94% firm, które odniosły sukces, wykonuje codzienne lub cotygodniowe konsolidacje. Firmy, które nie odniosły sukcesu, konsolidują kod raz na miesiąc albo i rzadziej (*Secrets to Software Success*, Detlev Hoch et al).

Kopie zapasowe kodu

Przed rozpoczęciem pracy nad jakimkolwiek projektem musisz zadbać o mechanizm, który automatycznie tworzy kopie zapasowe kodu. Kopie powinny być przechowywane poza terenem firmy, aby zapobiec ich utracie w przypadku włamania lub pożaru. Można albo fizycznie przechowywać je w innym miejscu, albo przesyłać je pocztą elektroniczną lub protokołem FTP. Pamiętaj, by uwzględnić kod pisany przez podwykonawców i pracowników zdalnych. Za bezpieczne tworzenie kopii zapasowych odpowiada menedżer projektu.

Konsolidacje demonstracyjne

O komercyjnym sukcesie programu nie przesądzi Twoje własne przekonanie o tym, jaki jest dobry, tylko to, jak łatwy dla użytkowników jest w instalacji i użyciu. Warto więc dać małej grupie potencjalnych klientów szansę na wypróbowanie go, w miarę jak zbliżasz się do końca projektu. Słuchaj uważnie ich komentarzy. Jeżeli np. użytkownicy zrobią coś, co jest niedozwolone, a oprogramowanie na to nie zareaguje, będziesz wiedział, że nie mieli pojęcia, jak trzeba zrobić daną rzecz, i że trzeba wyposażyć program w jakąś widoczną albo słyszalną wskazówkę. Im pewniej czują się użytkownicy, tym częściej będą polecać program innym, a to sposób na dużą sprzedaż. Oprogramowanie przyjazne w użyciu jest zawsze łatwiejsze do sprzedania.

Nic nie zastąpi możliwości zaprezentowania potencjalnemu użytkownikowi roboczej wersji programu. Opisując program, możemy wzbudzić entuzjazm. Wdając się w szczegóły, możemy wywołać zainteresowanie. Pisząc inspirujący plan projektu, możemy skłonić inwestora, by sięgnął do portfela. Pokazywanie próbnych ekranów może nawet wywołać pewne sugestie, ale nic nie zastąpi możliwości pokazania programu w akcji. Tylko wówczas użytkownik będzie mógł wyobrazić sobie przyszłą pracę z programem i zadawać te podstępne i druzgocące pytania, po których będziesz zachodzić w głowę, dlaczego sam o tym nie pomyślałeś. Jedyny sposób na przekształcenie zastrzeżeń w sprzedaż to zaprogramować dobre rozwiązania tych problemów.

Projekty zlecone

Jeżeli projekt jest realizowany na zlecenie, to im dłuższy jest czas od rozpoczęcia projektu do dostarczenia gotowego produktu, tym bardziej narażony jesteś na problemy z klientem. Jeżeli klient jest niedoświadczony, poświęć mu więcej czasu i zadbaj, by zrozumiał cały proces (możesz np. namówić go na przeczytanie niniejszej książki), a także poświęć szczególną uwagę temu, by priorytety były zgodne z kolejnością potrzeb klienta, a nie Twoją wygodą.

Po dopracowaniu specyfikacji i zasileniu Cię pierwszym zastrzykiem gotówki klient będzie miał dużo czasu na refleksję na temat tego, o co tak naprawdę mu chodziło. Często będzie próbował wprowadzić późniejsze zmiany bez ponoszenia kosztów. Zmiany w prawie, gospodarce lub wewnętrznej sytuacji firmy klienta również mogą wpłynąć na wymagania. Jednak nawet bez tego typu oddziaływań niewielu klientów potrafi naprawdę wyobrazić sobie to, co zlecili. W końcu dlatego właśnie Ty piszesz program, a klient jest klientem.

Klienci wiedzą za to dokładnie, co im się nie podoba, kiedy już to zobaczą. Wówczas nie omieszkają o tym powiedzieć. Czasami przesadzają w obawie, że jeżeli nie będą tego robić, zostaną zignorowani. Czasami nie do końca potrafią wyrazić, o co im chodzi. Słuchaj uważnie i na bieżąco notuj wszystkie kluczowe uwagi. Słuchaj nie tylko tego, co mówią, ale tego, co próbują powiedzieć. Później przemyśl wszystko z ich punktu widzenia i ustal, co jest dla nich najbardziej korzystne.

Jeżeli klient wydelegował do współpracy z nami menedżera średniego szczebla, najlepiej interpretować wszelkie jego zalecenia dosłownie. Jeżeli później okaże się, że program nie działa jak trzeba, okaże się również, że został napisany idealnie według specyfikacji menedżera. Menedżerowie mają świadomość, że ktoś znajdzie nieprawidłowości i obwini właśnie ich. Krytyka może być w tym przypadku drobna albo potężna. W takiej sytuacji najlepiej nie robić zamieszania. Zaoferuj, że sam postarasz się wszystko poprawić. Następnie dopisz do rachunku słuszną sumę, ale dyskretnie. Osoba, z którą się kontaktujesz, doceni, że wyratowałeś ją z opresji i że Twoja firma stworzyła dokładnie to, czego potrzebuje jej firma. Najczęściej będzie wdzięczna.

Pamiętaj jednak, że za pewne błędy obwiniony możesz zostać Ty. W końcu też nie jesteś nieomylny. Jeżeli okaże się, że rzeczywiście popełniłeś błąd, weź za niego odpowiedzialność i postaraj się jak najszybciej go poprawić. Jeżeli jednak to, o co prosi klient, jest poważną zmianą, przypomnij taktownie, że przy okazji zatwierdzania specyfikacji utworzone zostały rezerwy na dodatkowe koszty. Polegało to na dokładnym,

wspólnym przeczytaniu każdego elementu specyfikacji, by zminimalizować konieczność wprowadzenia dużej liczby poprawek. Przede wszystkim rób wszystko, by do takiej sytuacji nie doszło. Jeżeli już dojdzie, staraj się załagodzić problem na wczesnym etapie, pokazując klientowi kolejne konsolidacje programu.

Pokazywanie klientowi konsolidacji w trakcie realizacji projektu przypomina nieco pokazywanie przyszłym rodzicom zdjęć płodu. Gdy potem przychodzi na świat syn lub córka, nie jest to już taka niespodzianka. Zawsze, gdy klient prosi o coś nowego lub nowatorskiego, sprytne firmy programistyczne zdobywają jego poparcie, demonstrując kolejne konsolidacje i prototypy. Klient szybko dostrzeże wówczas, że nie da się od razu trafić w dziesiątkę i zawsze potrzebne są wyczerpujące dyspozycje w każdym zakresie. Zapoznając klienta z postępem prac sprawiasz, że zaczyna on uczestniczyć w procesie tworzenia, i gdy już jest po Twojej stronie, będziesz zaskoczony jego pomysłowością i wsparciem.

Jeżeli rozwiązanie nie ma zaawansowanego technologicznie charakteru, najlepszą osobą do prezentacji konsolidacji jest zwykle menedżer projektu. Najczęściej to on najlepiej zna potrzeby klienta oraz tworzony program.

Zawsze, kiedy klient prosi o nową funkcję, pamiętaj, że dodając ją, być może rozwiązujesz problemy, z którymi borykają się inni klienci. Zastanów się, czy w Twoim najlepszym interesie leży obciążenie klienta całym kosztem jej stworzenia, czy podzielić koszt pomiędzy przyszłych klientów. Pomyśl, ile czasu zajmie stworzenie tej funkcji i w jaki inny sposób firma mogłaby ten czas wykorzystać. Pomyśl o tym wszystkim również z perspektywy rynku.

Prototypy danych

Podobne podejście często przydaje się, kiedy tworzymy system operujący dużą ilością danych. Jeżeli nie jesteś pewien możliwości manipulowania wymaganą ilością danych, stwórz prototyp i wykorzystaj go w testach. Dla mnie okazało się to nieocenione, kiedy poproszono mnie o sprawdzenie, czy niektóre z pierwszych sieci opartych na komputerach osobistych działają na tyle szybko, by móc zastąpić dominujące wówczas systemy na bazie minikomputerów przy obsłudze transakcji giełdowych. Sądziliśmy, że to możliwe, ale niektórzy członkowie zespołu klienta byli sceptyczni, szczególnie dyrektor zarządzający. Rozwiązaniem była budowa prototypu z serwerem, który generował setki tysięcy różnych liczb na minutę (symulując rynek papierów wartościowych o stałym, dużym obciążeniu transakcjami), a stacja robocza wyświetlała je na monitorze w standardowym formacie stosowanym przez maklerów. Zademonstrowaliśmy wówczas, że sieć stworzona z komputerów osobistych radzi sobie z największym nawet przepływem danych czternaście razy szybciej niż starsze, bardziej tajemniczo wyglądające minikomputery.

Nigdy nie zapominaj o krzywej uczenia się

Pamiętasz, jak długo uczyłeś się pierwszego języka programowania, tworzyłeś pierwszą bazę danych albo interaktywną stronę WWW? Poznanie każdej nowej technologii informatycznej wymaga czasu, by dowiedzieć się, jak działa, odnaleźć i zrozumieć opcje i wreszcie osiągnąć niemal automatyczną biegłość. Nawet wówczas potrzebujesz na tyle dużego doświadczenia, by wiedzieć, kiedy najlepiej skorzystać z danego narzędzia.

Nie sposób oczekiwać od programistów, że w pełni poznają każdą nową technologię w czasie krótszym niż miesiąc. Na naukę pisania prostego kodu z większą od ślimaczej szybkością powinieneś dać im trzy miesiące, a dopiero po sześciu miesiącach będą w stanie podejmować słuszne, oparte na wiedzy decyzje. Być może sam uczysz się szybciej albo wolniej — tak czy inaczej, szacując wydajność zespołu, musisz to uwzględnić. Nikt nie potrafi od razu pisać doskonałego kodu.

Sprawdzanie wzorów matematycznych

Wzory matematyczne stosowane przez programistów bywają skomplikowane, a do pierwszej kompilacji programu — i tym samym możliwości jego sprawdzenia — musi upłynąć trochę czasu. Można z tego wybrnąć, tworząc próbne wzory i testując je w arkuszu kalkulacyjnym, gdzie możesz je dopracowywać, aż zadziałają idealnie, zanim użyjesz ich w kodzie programu. Dodatkowo sprawdź, czy wyniki po skompilowaniu są takie same, jakie dawał arkusz.

Nowi pracownicy

Pamiętasz swój pierwszy dzień w nowej szkole? Któryś z uczniów wziął Cię pod swoje skrzydła i pokazywał Ci wszystko. Najszybszy sposób na wdrożenie nowego członka do zespołu to wyznaczenie kogoś ze starszych pracowników, by go wprowadził. Kilka dni patronowania zwykle wystarczy, by nowa osoba poznała ludzi i dowiedziała się, u kogo szukać pomocy.

Najlepsza metoda przedstawiania

Kiedy ludzie zaczynają pracę, trzeba ich przedstawić. Powinna przy tej okazji nastąpić wymiana nazwisk, funkcji i łatwych do zapamiętania szczegółów. Na przykład: „To jest Sadie Brown. Opiekuje się bazami danych i jest maniaczką latania lotnią za łodzią motorową", „A to jest Chris White, nasz spec od marketingu. Zrobił świetną robotę przy projekcie Mars". Z każdą osobą skojarz jakiś wart zapamiętania fakt.

W czasie oprowadzania nowego pracownika po wszystkich działach (lub po pokoju) zdążysz z grubsza opowiedzieć mu o tempie pracy, o tym, kto komu podlega, co powinien zrobić, gdy napotka poważny problem, jakiego wsparcia może oczekiwać itd.

Staraj się poznać osobiście swoich ludzi. Podwoź ich do domu albo zabieraj na obiady. Dowiedz się, czego dokonali, czy lubili szkołę, jak im się układa w domu, z czego są dumni i jakie mają aspiracje. Powiedz też coś o sobie. Czasem możecie w ogóle nie rozmawiać o sprawach zawodowych, a dzięki temu środowisko pracy stanie się bardziej komfortowe.

Zanim rozdzielisz pracę lub opublikujesz harmonogramy, zorganizuj zebranie, by przedyskutować najdrobniejsze szczegóły. Wyjaśnij, w jaki sposób projekt ma tworzyć całość, jaki jest Twój styl zarządzania, czego oczekujesz od każdego pracownika i czego pragniesz za wszelką cenę unikać.

Kiedy zegar zaczyna tykać

Takie zasoby, jak biuro, komputery i ludzie, są proste w tym sensie, że można je mieć albo ich nie mieć. Jeżeli zaczynasz od zera, masz zatwierdzony projekt i wyasygnowane zasoby, największym problemem jest czas, jaki upływa od przyjęcia odpowiednich ludzi do chwili, gdy dotrze zamówiony dla nich sprzęt lub przygotowane zostaną pomieszczenia biurowe. Wyjaśnij wszystkim nowym pracownikom, że z chwilą gdy wszystko będzie gotowe do pracy, zegar zaczyna tykać i że budżety zostaną przekroczone, jeżeli opóźnienia w dostarczeniu różnych rzeczy będą wstrzymywać realizację projektu. Najlepszym lekarstwem jest jednak zapobieganie. Zadbaj o to, by podział zasobów i zaopatrzenie zostały uwzględnione w harmonogramie i żeby harmonogram uwzględniał realistyczny zapas czasu na niespodziewane opóźnienia.

Czasami trzeba zrobić krok wstecz

Mimo że to bolesne, czasami bardziej efektywne jest zarzucenie fragmentu złego kodu i rozpoczęcie od nowa. Typowe powody i sposoby reagowania wymieniono w tabeli 6.2.

Tabela 6.2. *Postępowanie ze złym kodem*

Powód	Reakcja
Programista napisał beznadziejny kod.	Zlecić to komuś innemu.
Programista robi postępy zbyt wolno.	Zlecić to komuś innemu.
Kod działa, ale jest bardzo mało wydajny.	Rozważyć napisanie od nowa.
Kod działa, ale programista jest niezadowolony z tego, jak działa.	Napisać od nowa, o ile czas na to pozwala.
Program częściowo działa.	Zobaczyć, czy da się go poprawić.

Jeżeli znajdziesz się w której z powyższych sytuacji, nie obawiaj się zlecić innemu programiście poprawienia lub napisania od nowa wadliwego fragmentu kodu. Jeżeli nową wersję kodu pisze ten sam człowiek, zwykle trwa to trzy razy szybciej, niż napisanie pierwotnej wersji. Jeżeli programista notorycznie traci grunt pod nogami, zastanów się poważnie, czy w ogóle się do niej nadaje.

Jeżeli na uczelni będziesz badać coś przez trzy lata, po czym napiszesz pracę magisterską, w której stwierdzisz, że to coś nie działa, tytuł magistra masz prawie zapewniony. W świecie biznesu niepowodzenie to ostatnia rzecz, jaka jest nagradzana. Pamiętaj jednak, że niektórzy z największych ludzi sukcesu doświadczyli porażki, zanim osiągnęli sukces. Ich zdaniem porażka nie zasługuje na społeczne potępienie, z którym zwykle się spotyka. Jeżeli zdasz sobie sprawę, że utknąłeś w martwym punkcie, jak najszybciej zwołaj zebranie. Jeżeli nikt nie znajduje ratunku, spróbuj podejść do problemu z innej strony. Mimo że jest to trudna decyzja, czasami więcej sensu ma wstrzymanie projektu niż parcie dalej.

Oto typowe okoliczności, w których najlepiej zwołać wszystkich menedżerów i ustalić, czy warto kontynuować projekt:

- Dotrzymanie budżetu jest niemożliwe.
- Harmonogram nie pozwala wykorzystać nadarzających się okazji.

- Kluczowi ludzie nie są już dostępni, a dalsza praca bez nich nie jest możliwa.
- Pojawił się znaczący konkurent.
- Technologia uległa zmianie.
- Powstają usterki, których w żaden sposób nie da się wyeliminować.
- Inwestor traci wiarę w projekt.
- Sytuacja rynkowa uległa radykalnej zmianie.

Ponieważ z kadrą zarządzającą komunikujesz się na bieżąco, cel spotkania raczej nie będzie niespodzianką. Jeżeli projekt ma być wstrzymany, warto go łagodnie wyhamować, a nie nagle przerywać. Trzeba ustalić, jakie komponenty projektu można wykorzystać w przyszłości, i udokumentować wnioski płynące z niego. Porzucony projekt może wciąż się opłacić, ułatwiając realizację innych.

Unikaj zbytniego przeciążenia

Bywa tak, że realizacja projektu idzie całkiem dobrze do czasu, gdy 90% pracy zostanie wykonane. Wówczas pojawia się przeszkoda, która zatrzymuje wszystko na dobre. Czasami wynika to z faktu, że zespoły były bardzo przeciążone pracą i, kiedy najgorsze mają już za sobą, uchodzi z nich para. Takiego nacisku nie da się długo podtrzymywać. To jak jeżdżenie samochodem cały czas na najwyższych obrotach — prędzej czy później silnik odmówi posłuszeństwa. Z ludźmi jest tak samo.

Skuteczniejsza technika zarządzania polega na złagodzeniu nacisku, zanim zespół się wypali. Kiedy programowanie dociera do etapu tworzenia dokumentacji i prowadzenia beta-testów, zwolnij tempo. Wyjaśnij, dlaczego to robisz, i pozwól, by ludzie odzyskali siły na batalię ostatniego etapu produkcji.

Wiedzieć, kiedy skończyć

Programiści, jak wszyscy rzemieślnicy, potrafią dopracowywać i udoskonalać swoje dzieło w nieskończoność. Krytyczne możliwości funkcjonalne wymagane z punktu widzenia marketingu zawarte zostały w planie projektu. Pozytywne wyniki testów potwierdzą eliminację usterek. Po spełnieniu tych kryteriów oprogramowanie jest gotowe do premiery. Nawet jeżeli siedzisz w pokoju i samotnie przesz do przodu, powinieneś pamiętać, że Twoim zadaniem jest coś więcej niż tylko pisanie kodu.

Lista kontrolna

Nie myśl nawet o rozpoczęciu programowania, jeżeli nie zakończyłeś tworzenia planu. Oto lista kontrola pozwalająca dokończyć plan:

1. Zatrudnij najlepszych możliwych ludzi.
2. Wybierz osoby, które znają branżę, w jakiej działa użytkownik końcowy.
3. Znajdź menedżerów, którzy zajmowali się tym w przeszłości.

4. Nie kontynuuj, dopóki nie znajdziesz odpowiednich ludzi.
5. Zachowaj małe rozmiary zespołów.
6. Podziel zadanie na krótkie możliwe do ogarnięcia etapy.
7. Zadbaj o to, by każda osoba wiedziała, co ma robić.
8. Dbaj o zapał i bieżące rozeznanie członków zespołu.
9. Wtajemnicz zespół w swoje sprawy.
10. Chroń programistów przed przerywaniem im pracy.
11. Wetuj nieistotne zmiany.
12. Bądź dstępny dla swoich ludzi.
13. Codziennie wyznaczaj rytm pracy programistom.
14. Kontroluj czas.
15. Zachęcaj programistów, by pomagali sobie nawzajem.
16. Daj dobry przykład.
17. Szybko pozbywaj się czarnych owiec.
18. Prowadź potencjalnego klienta przez poszczególne etapy produkcji.
19. Regularnie testuj oprogramowanie.
20. Twórz kopie bezpieczeństwa kodu.



Słowo przestrogi przed przesadnym dopracowywaniem kodu. Joe Tucci, który przez sześć lat prowadził firmę Wang i uchronił ją przed bankructwem, zawsze mawiał na spotkaniach z nami programistami, że dobry program powstaje przeważnie dopiero za trzecim razem. Dlatego nie obawiaj się, że wydasz nie całkiem idealny produkt. Pierwsza wersja ma mieć tyle funkcji, by zadowolić użytkowników. W miarę sukcesu i przychodów przekształcisz ją w to, czym chciałeś, żeby była (co najpewniej stanie się w okolicach trzeciej wersji).