

Spis treści

Słowo wstępne	17
Przedmowa	19
Część I. Teza	25
1. Czym jest inżynieria oprogramowania	27
Czas i zmiana	30
Prawo Hyruma	32
Przykład: uporządkowanie z mieszaniem	33
Dlaczego po prostu nie dążyć do sytuacji „nic się nie zmienia”?	34
Skala i efektywność	35
Zasady, które nie zapewniają skalowania	37
Zasady zapewniające dobre skalowanie	38
Przykład: aktualizacja kompilatora	38
Przesunięcie w lewą stronę	40
Kompromisy i koszty	42
Przykład: markery	43
Informacje zapewniane przy podejmowaniu decyzji	44
Przykład: kompilacje rozproszone	44
Przykład: wybór między czasem i skalą	46
Ponowne analizowanie decyzji i popełnianie błędów	47
Porównanie inżynierii oprogramowania i programowania	47
Podsumowanie	48
W skrócie	48

2. Jak dobrze pracować w zespołach?	53
Pomóż mi ukryć mój kod	53
Mit geniuszu	54
Ukrywanie uważane za coś szkodliwego	56
Wczesne wykrywanie	57
Wskaźnik autobusowy	57
Tempo postępów	58
Mówiąc wprost, nie ukrywaj się	60
Liczy się tylko zespół	60
Trzy filary interakcji społecznych	60
Dlaczego te filary są istotne?	61
Pokora, szacunek i zaufanie w praktyce	62
Kultura analizowania bez obwiniania	65
Bycie „googlowym”	67
Podsumowanie	68
W skrócie	68
3. Dzielenie się wiedzą	69
Wyzwania związane z uczeniem się	69
Filozofia	71
Przygotowanie: bezpieczeństwo psychologiczne	72
Doradztwo	72
Bezpieczeństwo psychologiczne w dużych grupach	73
Poszerzanie własnej wiedzy	74
Zadawaj pytania	74
Zrozum kontekst	75
Skalowanie pytań: zadaj je społeczności	76
Rozmowy grupowe	76
Listy wysyłkowe	77
YAQS — platforma pytań i odpowiedzi	78
Skalowanie posiadanej wiedzy — zawsze możesz się czegoś nauczyć	78
Spotkania w godzinach pracy	78
Konsultacje techniczne i zajęcia	79
Dokumentacja	80
Kod	82
Skalowanie wiedzy na poziomie organizacji	82
Doskonalenie kultury dzielenia się wiedzą	82
Ustanawianie kanonicznych źródeł informacji	85
Bycie na bieżąco	87

Czytelność — standaryzowane doradztwo w ramach inspekcji kodu	88
Czym jest proces oceny czytelności?	89
Dlaczego korzysta się z tego procesu?	90
Podsumowanie	92
W skrócie	93
4. Inżynieria zapewniająca równość	94
Upředzenie z założenia	95
Zrozumienie potrzeby uwzględniania różnorodności	97
Zapewnianie zdolności wspierania wielokulturowości	98
Ustanawianie różnorodności bodźcem do podejmowania działań	100
Odrzucanie osobliwych metod	100
Uporanie się z ugruntowanymi procesami	102
Wartości a wyniki	103
Zachowaj ciekawość i nie zatrzymuj się	104
Podsumowanie	104
W skrócie	105
5. Jak kierować zespołem?	106
Menedżerowie i kierownicy techniczni (i osoby pełniące obie role)	106
Menedżer inżynierów	107
Kierownik techniczny	107
Menedżer i kierownik techniczny w jednym	107
Przejście z roli pojedynczego uczestnika do roli przywódcy	109
Jedyna rzecz wywołująca obawy to... w zasadzie wszystko	109
Przywództwo służebne	110
Menedżer inżynierów	111
Menedżer to osoba identyfikowana przez 4-literowe słowo	111
Współczesny menedżer inżynierów	112
Antywzorce	114
Antywzorec: zatrudnianie naiwniaków	114
Antywzorec: ignorowanie osób o kiepskiej wydajności	114
Antywzorec: ignorowanie problemów ludzkich	116
Antywzorec: przyjażnienie się z każdym	116
Antywzorec: naruszanie poprzeczki zatrudnienia	117
Antywzorec: traktowanie własnego zespołu jak dzieci	118
Pozytywne wzorce	118
Zrezygnuj z ego	118
Bądź mistrzem zen	120
Bądź katalizatorem	121
Usuśaj przeszkody	122
Bądź nauczycielem i mentorem	122

Określ wyraźne cele	123
Bądź szczerzy	123
Monitoruj poziom zadowolenia	125
Nieoczekiwane pytanie	126
Inne wskazówki i rady	127
Ludzie są jak rośliny	129
Motywacja wewnętrzna i zewnętrzna	129
Podsumowanie	131
W skrócie	131
6. Przywództwo przy dużej skali	132
Zawsze bądź rozstrzygającym	133
Przypowieść o samolocie	133
Identyfikowanie osób „z kłapkami na oczach”	134
Ustalanie kluczowych kompromisów	135
Decydowanie, a następnie powtarzanie działań	135
Zawsze możesz odejść	137
Twoja misja: zbuduj niezależny zespół	138
Dzielenie obszaru problemu	138
Zawsze stawiaj na rozwój	141
Cykl sukcesu	141
Ważne kontra pilne	143
Uczenie się upuszczania piłek	144
Oszczędzanie własnej energii	146
Podsumowanie	147
W skrócie	148
7. Pomiar wydajności inżynierów	149
Dlaczego powinno się mierzyć wydajność inżynierów?	149
Segregacja: czy w ogóle jest to warte przeprowadzania pomiaru?	151
Wybór sensownych metryk uwzględniających cele i sygnały	154
Cele	156
Sygnały	157
Metryki	158
Użycie danych do weryfikowania poprawności metryk	159
Podejmowanie działania i monitorowanie wyników	163
Podsumowanie	163
W skrócie	163

Część III. Procesy	165
8. Wytyczne i reguły dotyczące stylu	167
Dlaczego stosujemy reguły?	168
Tworzenie reguł	169
Zasady ustalania wytycznych	169
Przewodnik stylów	177
Modyfikowanie reguł	180
Proces	182
Moderatorzy stylów	182
Wyjątki	183
Wytyczne	183
Stosowanie reguł	185
Narzędzia do sprawdzania błędów	187
Narzędzia formatujące kod	188
Podsumowanie	190
W skrócie	190
9. Inspekcja kodu	191
Przepływ procesu inspekcji kodu	192
Realizowanie inspekcji kodu w firmie Google	193
Zalety inspekcji kodu	196
Poprawność kodu	197
Zrozumiałość kodu	198
Spójność kodu	199
Korzyści natury psychologicznej i kulturowej	200
Dzielenie się wiedzą	201
Najlepsze praktyki dotyczące inspekcji kodu	202
Bądź miły i profesjonalny	202
Tworzenie drobnych zmian	203
Tworzenie dobrych opisów zmian	204
Ograniczanie liczby inspektorów do minimum	205
Automatyzowanie, gdy tylko jest to możliwe	205
Typy inspekcji kodu	206
Inspekcje zupełnie nowego kodu	206
Zmiany behawioralne, ulepszenia i optymalizacje	207
Poprawki błędów i wycofania	207
Refaktoryzacje i zmiany na dużą skalę	208
Podsumowanie	208
W skrócie	209

10. Dokumentacja	210
Co jest kwalifikowane jako dokumentacja?	210
Dlaczego dokumentacja jest niezbędna?	211
Dokumentacja jest jak kod	212
Poznaj swoich odbiorców	215
Typy odbiorców	216
Typy dokumentacji	217
Dokumentacja referencyjna	218
Dokumenty projektowe	220
Kursy	221
Dokumentacja pojęciowa	223
Strony docelowe	223
Inspekcje dokumentacji	224
Filozofia towarzysząca dokumentacji	226
KTO, CO, KIEDY, GDZIE i DLACZEGO	226
Początek, środek i zakończenie	227
Parametry dobrej dokumentacji	227
Wycofywanie dokumentów	228
Kiedy są potrzebni twórcy dokumentów technicznych?	229
Podsumowanie	229
W skrócie	230
11. Testowanie — przegląd	231
Dlaczego tworzymy testy?	232
Historia serwera Google Web Server	233
Testowanie z szybkością nowoczesnego procesu projektowania	234
Utwórz, uruchom i zareaguj	236
Korzyści wynikające z testowania kodu	237
Projektowanie zestawu testów	238
Wielkość testu	239
Zasięg testów	243
Reguła Beyoncé	246
Coś na temat pokrycia kodu	247
Testowanie w firmie tak dużej jak Google	247
Pułapki towarzyszące dużemu zestawowi testów	248
Historia testowania w firmie Google	250
Zajęcia wprowadzające	251
Program certyfikacji Test Certified	251
Testowanie w toalecie	252
Kultura testowania obecnie	253
Ograniczenia zautomatyzowanego testowania	254
Podsumowanie	255
W skrócie	255

12. Testowanie jednostkowe	256
Doniosłość możliwości utrzymania	257
Unikanie niepewnych testów	258
Dążenie do uzyskania trwałych testów	258
Testowanie za pośrednictwem publicznych interfejsów API	259
Testuj stan, a nie interakcje	263
Tworzenie przejrzystych testów	264
Zapewnianie kompletności i zwięzłości testów	265
Testuj zachowania, a nie metody	266
Nie umieszczaj logiki w testach	270
Twórz przejrzyste komunikaty o niepowodzeniach	271
Testy i współużytkowanie kodu: zasada DAMP, a nie DRY	272
Wartości współużytkowane	274
Konfiguracja współużytkowana	276
Współużytkowane metody pomocnicze i sprawdzanie poprawności	277
Definiowanie infrastruktury testów	278
Podsumowanie	279
W skrócie	279
13. Dublery w testach	280
Wpływ dublerów w testach na projektowanie oprogramowania	281
Dublery w testach w firmie Google	281
Podstawowe pojęcia	282
Przykładowy dubler w teście	282
Łączy	283
Środowiska tworzące obiekty zastępcze	284
Techniki użycia dublerów w testach	285
Użycie fałszywego obiektu	285
Użycie obiektów stub	286
Testowanie interakcji	286
Rzeczywiste implementacje	287
Wybieraj realizm, a nie izolację	288
Decydowanie o tym, kiedy skorzystać z rzeczywistej implementacji	289
Zastosowanie fałszywego obiektu	291
Dlaczego fałszywe obiekty są ważne?	292
Kiedy powinno się tworzyć fałszywe obiekty?	292
Wierność fałszywych obiektów	293
Fałszywe obiekty powinny być testowane	294
Jak postąpić, jeśli fałszywy obiekt jest niedostępny?	294
Użycie obiektów stub	295
Zagrożenia związane z nadużywaniem obiektów stub	295
Kiedy użycie obiektów stub jest właściwe?	297

Testowanie interakcji	297
Zamiast testowania interakcji preferuj testowanie stanu	298
Kiedy testowanie interakcji jest właściwe?	299
Najlepsze praktyki związane z testowaniem interakcji	299
Podsumowanie	302
W skrócie	302
14. Testowanie na dużą skalę	303
Czym są większe testy?	303
Wierność	304
Typowe luki w testach jednostkowych	305
Dlaczego nie warto korzystać z większych testów?	307
Większe testy w firmie Google	308
Większe testy i czas	309
Większe testy przy skali działań firmy Google	310
Struktura dużego testu	311
Testowany system	312
Dane testu	317
Weryfikacja	318
Typy większych testów	319
Testowanie funkcjonalne jednego lub większej liczby plików binarnych prowadzących interakcję	320
Testowanie przeglądarki i urządzenia	320
Testowanie wydajności, obciążenia i przeciążenia	320
Testowanie konfiguracji wdrażania	321
Testowanie rozpoznawcze	321
Testowanie różnic A/B (regresji)	322
Testowanie akceptacyjne na poziomie użytkowników	324
Kontrolery i analiza kanarkowa	324
Przywracanie awaryjne i inżynieria chaosu	325
Ocena na poziomie użytkowników	326
Duże testy i przepływ informacji procesu projektowania	327
Tworzenie dużych testów	328
Uruchamianie dużych testów	329
Ustanowienie właścicieli dużych testów	332
Podsumowanie	333
W skrócie	333
15. Wycofywanie	334
Dlaczego warto wycofywać?	335
Dlaczego wycofywanie jest takie trudne?	336
Uwzględnienie wycofywania podczas projektowania	338

Typy procesu wycofywania	339
Wycofywanie wskazane	339
Wycofywanie obowiązkowe	340
Ostrzeżenia związane z wycofywaniem	341
Zarządzanie procesem wycofywania	342
Właściciele procesu	343
Kamienie milowe	343
Narzędzia do wycofywania	344
Podsumowanie	345
W skrócie	346

Część IV. Narzędzia **347**

16. Kontrola wersji i zarządzanie gałęziami	349
Czym jest kontrola wersji?	349
Dlaczego kontrola wersji jest ważna?	351
Porównanie scentralizowanego i rozproszonego systemu VCS	353
„Źródło prawdy”	356
Kontrola wersji i zarządzanie zależnościami	358
Zarządzanie gałęziami	359
Praca w trakcie jest równoznaczna istnieniu gałęzi	359
Gałęzie rozwojowe	360
Gałęzie wersji do opublikowania	362
Kontrola wersji w firmie Google	362
Jedna wersja	363
Scenariusz: wiele dostępnych wersji	364
Reguła „jednej wersji”	365
(Prawie) żadnych długo istniejących gałęzi	366
A co z gałęziami publikowanych wersji?	368
Repozytoria monolityczne	368
Przyszłość kontroli wersji	370
Podsumowanie	372
W skrócie	373
17. Narzędzie Code Search	374
Interfejs użytkownika narzędzia Code Search	375
W jaki sposób pracownicy firmy Google korzystają z narzędzia Code Search?	376
Gdzie?	376
Co?	377
Jak?	377
Dlaczego?	378
Kto i kiedy?	378

Dlaczego zdecydowano się na osobne narzędzie internetowe?	378
Skala	378
Globalny widok kodu bez wymogu konfiguracji	379
Specjalizacja	380
Integracja z innymi narzędziami do projektowania	380
Udoszczelnianie interfejsów API	382
Wpływ skali na projekt	382
Opóźnienie zapytania wyszukiwania	383
Opóźnienie indeksowania	384
Implementacja firmy Google	385
Indeks wyszukiwania	385
Klasyfikowanie	386
Wybrane kompromisy	390
Kompletność — liczy się przede wszystkim repozytorium	390
Kompletność — wszystkie wyniki i te najbardziej trafne	390
Kompletność — porównanie bieżącej gałęzi kodu z innymi gałęziami kodu oraz całej historii i obszarów roboczych	391
Wyrazistość — porównanie tokenów, podłańcuchów i wyrażeń regularnych	392
Podsumowanie	393
W skrócie	394
18. Systemy do kompilacji i filozofia procesu kompilacji	395
Przeznaczenie systemu do kompilacji	395
Co się dzieje, gdy nie ma systemu do kompilacji?	397
Wszystko, czego potrzebuję, to kompilator	397
Czy skrypty powłoki nas uratują?	398
Nowoczesne systemy do kompilacji	399
Wszystko sprowadza się do zależności	399
Systemy do kompilacji oparte na zadaniach	400
Systemy do kompilacji oparte na artefaktach	404
Kompilacje rozproszone	411
Czas, skala i kompromisy	414
Radzenie sobie z modułami i zależnościami	415
Użycie szczegółowych modułów i reguła 1:1:1	415
Minimalizowanie widoczności modułów	417
Zarządzanie zależnościami	417
Podsumowanie	422
W skrócie	423
19. Critique — narzędzie firmy Google do inspekcji kodu	424
Zasady dotyczące narzędzi do inspekcji kodu	424
Przebieg inspekcji kodu	425
Powiadomienia	427

Etap 1. Tworzenie zmiany	427
Ujawnianie różnic	428
Wyniki analizy	429
Ścisła integracja narzędzi	431
Etap 2. Żądanie inspekcji	432
Etapy 3. i 4. Zaznajomienie się ze zmianą i skomentowanie jej	433
Komentowanie	433
Stan zmiany	434
Etap 5. Zatwierdzenie zmiany	437
Etap 6. Wprowadzanie zmiany	438
Po wprowadzeniu zmiany — historia śledzenia	438
Podsumowanie	439
W skrócie	440
20. Analiza statyczna	441
Właściwości efektywnej analizy statycznej	442
Skalowalność	442
Użyteczność	442
Kluczowe wnioski związane z wdrażaniem analizy statycznej	443
Skoncentrowanie się na zadowoleniu projektanta	443
Ustanowienie analizy statycznej częścią głównego przepływu informacji	
procesu projektowania	444
Pozwól użytkownikom zaangażować się	444
Tricorder — platforma do analizy statycznej firmy Google	445
Zintegrowane narzędzia	446
Zintegrowane kanały informacji zwrotnych	447
Poprawki sugerowane	448
Dostosowywanie przed rozpoczęciem projektu	449
Sprawdzenia przed wysłaniem	450
Integracja z kompilatorem	450
Analiza w trakcie edytowania i przeglądania kodu	451
Podsumowanie	452
W skrócie	452
21. Zarządzanie zależnościami	453
Dlaczego zarządzanie zależnościami jest tak trudne?	455
Wymagania powodujące konflikt i zależności diamentowe	455
Importowanie zależności	457
Możliwości zapewnienia zgodności	457
Kwestie uwzględniane podczas importowania	460
Jak w firmie Google radzimy sobie z importowaniem zależności?	461

Zarządzanie zależnościami w teorii	463
Nic się nie zmienia (czyli statyczny model zależności)	464
Wersjonowanie semantyczne	464
Pakietowe modele dystrybucji	466
Model Live at Head	466
Ograniczenia wersjonowania semantycznego	468
Wersjonowanie semantyczne może nadmiernie ograniczać	469
Wersjonowanie semantyczne może zapewniać zbyt wygórowaną obietnicę	470
Motywacje	471
Wybór wersji minimalnej	472
Czy zatem wersjonowanie semantyczne się sprawdza?	473
Zarządzanie zależnościami przy nieograniczonych zasobach	474
Eksportowanie zależności	477
Podsumowanie	480
W skrócie	481
22. Zmiany na dużą skalę	483
Czym jest zmiana na dużą skalę?	484
Kto zajmuje się zmianami na dużą skalę?	485
Bariery w przypadku zmian atomowych	487
Ograniczenia techniczne	487
Konflikty związane ze scalaniem	487
Nie ma „nawiedzonych cmentarzy”	488
Niejednoznaczność	488
Testowanie	489
Inspekcja kodu	491
Infrastruktura zmian na dużą skalę	493
Zasady i kultura pracy	493
Analiza bazy kodu	494
Zarządzanie zmianami	495
Testowanie	495
Obsługa zmian przez języki	495
Proces zmian na dużą skalę	497
Autoryzowanie	497
Tworzenie zmiany	498
Podział na zmiany składowe i wprowadzanie ich do bazy kodu	498
Oczyszczanie	502
Podsumowanie	502
W skrócie	502

23. Integracja ciągła	503
Pojęcia związane z integracją ciągłą	505
Szybkie pętle informacji zwrotnej	505
Automatyzacja	507
Testowanie ciągłe	509
Wyzwania towarzyszące integracji ciągłej	515
Testowanie hermetyczne	516
Integracja ciągła w firmie Google	519
Analiza przypadku procesu integracji ciągłej — aplikacja Google Takeout	522
Nie mogę jednak pozwolić sobie na zastosowanie integracji ciągłej	528
Podsumowanie	529
W skrócie	529
24. Wdrażanie ciągłe	530
Idiomy wdrażania ciągłego w firmie Google	531
Szybkość to sport zespołowy — sposób podziału procesu wdrażania na możliwe do zarządzania części	532
Ocena wyizolowanych zmian — opcje z flagą ochraniającą	533
Dążenie do zapewnienia zwinności — przygotowanie łańcucha wersji	534
Żadne binaria nie są idealne	535
Dotrzymuj ustalonego ostatecznego terminu publikacji wersji	535
Skupienie się na jakości i użytkowniku — udostępniaj tylko to, co jest przydatne	536
Przesunięcie w lewo — wcześniejsze podejmowanie decyzji opartych na danych	537
Zmiana kultury zespołowej — uwzględnienie dyscypliny w procesie wdrażania	539
Podsumowanie	540
W skrócie	540
25. Model CaaS	541
Ujarzmianie środowiska przetwarzania	542
Automatyzacja mozolnej pracy	542
Konteneryzacja i wielodostępność	544
Podsumowanie	547
Tworzenie oprogramowania dla zarządzanego środowiska przetwarzania	547
Tworzenie architektury pod kątem awarii	547
Zadania wsadowe i obsługujące	549
Zarządzanie stanem	551
Nawiązywanie połączenia z usługą	552
Kod jednorazowy	554
Wpływ czasu i skali na model CaaS	555
Kontenery jako abstrakcja	555
Jedna usługa, by wszystkimi rządzić	557
Wprowadzana konfiguracja	560

Wybór usługi przetwarzania	560
Centralizacja kontra dostosowywanie	562
Poziom abstrakcji: wariant bezserwerowy	564
Publiczne i prywatne	568
Podsumowanie	570
W skrócie	571

Część V. Podsumowanie	573
 Połowie	573