

Spis treści

Przedmowa	15
Wstęp	19
Podziękowania	23
O autorze	24
Część I. Czym jest inżynieria oprogramowania?	25
Rozdział 1. Wprowadzenie	27
Inżynieria — praktyczne zastosowanie nauki	27
Czym jest inżynieria oprogramowania?	28
Przywracanie „inżynierii” w „inżynierii oprogramowania”	29
Jak robić postępy?	30
Narodziny inżynierii oprogramowania	30
Zmiana paradygmatu	32
Podsumowanie	33
Rozdział 2. Czym jest inżynieria?	34
To nie produkcja jest naszym problemem	34
Inżynieria projektowa zamiast inżynierii produkcyjnej	35
Robocza definicja inżynierii	39
Inżynieria != kod	40
Dlaczego inżynieria jest ważna?	41
Ograniczenia rzemiosła	42
Precyzja i skalowalność	43
Radzenie sobie ze złożonością	43
Powtarzalność i precyzja pomiarów	45

Inżynieria, kreatywność i rzemiosło	46
Dlaczego to, co robimy, nie jest inżynierią oprogramowania	48
Kompromisy	48
Iluzja postępu	49
Droga od rzemiosła do inżynierii	50
Rzemiosło to za mało	51
Czas na zmianę perspektywy?	51
Podsumowanie	52
Rozdział 3. Podstawy podejścia inżynierskiego	53
Branża zmian?	53
Znaczenie pomiarów	54
Wprowadzanie stabilności i wydajności	56
Podstawy inżynierii oprogramowania	57
Eksperti od uczenia się	58
Eksperti od radzenia sobie ze złożonością	59
Podsumowanie	60
Część II. Optymalizacja z myślą o uczeniu się	61
Rozdział 4. Praca w modelu iteracyjnym	63
Praktyczne zalety podejścia iteracyjnego	65
Podejście iteracyjne jako strategia projektowania defensywnego	66
Pokusa tworzenia planu	67
Praktyczne aspekty podejścia iteracyjnego	73
Podsumowanie	74
Rozdział 5. Informacje zwrotne	75
Praktyczny przykład ilustrujący znaczenie informacji zwrotnych	75
Informacje zwrotne w czasie pisania kodu	78
Informacje zwrotne na etapie integracji	79
Informacje zwrotne na etapie projektowania	81
Informacje zwrotne w architekturze	83
Preferuj szybkie informacje zwrotne	84
Informacje zwrotne w kontekście projektu produktu	85
Informacje zwrotne w organizacji i kulturze	86
Podsumowanie	88

Rozdział 6. Podejście przyrostowe	89
Znaczenie modułowości	90
Podejście przyrostowe w organizacjach	91
Narzędzia ułatwiające przyrostową pracę	93
Ograniczanie zakresu wpływu zmian	94
Projektowanie przyrostowe	95
Podsumowanie	97
Rozdział 7. Podejście empiryczne	98
Zakorzenie w rzeczywistości	99
Oddzielenie podejścia empirycznego od eksperymentów	99
„Znam ten błąd!”	100
Unikanie oszukiwania samego siebie	101
Wymyślanie rzeczywistości pasującej do argumentów	102
Kierowanie się rzeczywistością	105
Podsumowanie	105
Rozdział 8. Nastawienie na eksperymentowanie	106
Czym jest „nastawienie na eksperymentowanie”?	107
Informacje zwrotne	108
Hipotezy	109
Pomiary	110
Kontrolowanie zmiennych	111
Zautomatyzowane testy jako eksperymenty	112
Zapewnianie kontekstu dla wyników testów przeprowadzanych w ramach eksperymentów	113
Zakres eksperymentów	115
Podsumowanie	115
Część III. Optymalizowanie z myślą o radzeniu sobie ze złożonością	117
Rozdział 9. Modułowość	119
Cechy charakterystyczne modułowości	120
Niedoceniać znaczenia dobrego projektu	121
Znaczenie testowalności	122
Projektowanie z myślą o łatwości testowania poprawia modułowość	123
Usługi i modułowość	129
Łatwość wdrażania a modułowość	130

Modułowość w różnych skalach	132
Modułowość w systemach ludzkich	133
Podsumowanie	134
Rozdział 10. Spójność	135
Modułowość i spójność — podstawy projektowania	135
Prosty przykład niskiej spójności	136
Kontekst ma znaczenie	139
Wysoco wydajne oprogramowanie	141
Związki z powiązaniem	142
Zapewnianie wysokiej spójności za pomocą programowania sterowanego testami	142
Jak uzyskać spójne oprogramowanie?	143
Koszty niskiej spójności	145
Spójność w systemach ludzkich	146
Podsumowanie	146
Rozdział 11. Podział zadań	147
Wstrzykiwanie zależności	150
Oddzielanie złożoności zasadniczej od złożoności przypadkowej	151
Znaczenie podejścia DDD	154
Testowalność	156
Porty i adaptery	156
Kiedy stosować wzorzec porty i adaptery?	158
Czym jest API?	159
Stosowanie programowania sterowanego testami do wprowadzania podziału zadań	160
Podsumowanie	161
Rozdział 12. Ukrywanie informacji i abstrakcja	162
Abstrakcja lub ukrywanie informacji	162
Co jest powodem powstawania „wielkiej błotnej bryły”?	163
Problemy organizacyjne i kulturowe	163
Problemy techniczne i problemy projektowe	165
Obawy przed „nadinżynierią”	168
Tworzenie bardziej abstrakcyjnego kodu za pomocą testów	170
Wartość abstrakcji	171
„Dziurawe” abstrakcje	172

Wybór odpowiednich abstrakcji	174
Abstrakcje z dziedziny problemu	175
Wyodrębnianie złożoności przypadkowej za pomocą abstrakcji	176
Izolowanie zewnętrznych systemów i zewnętrznego kodu	179
Zawsze preferuj ukrywanie informacji	180
Podsumowanie	181

Rozdział 13. Radzenie sobie z powiązaniami 182

Koszty powiązań	182
Skalowanie	183
Mikrouslugi	184
Wylimowanie powiązań może prowadzić do większej ilości kodu	185
Luźne powiązanie nie jest jedynym, które ma znaczenie	187
Preferuj luźne powiązania	187
W czym powiązania różnią się od podziału zadań?	189
Zasada DRY jest zbyt uproszczona	189
Asynchroniczność jako narzędzie do uzyskiwania luźnych powiązań	191
Projektowanie z myślą o luźnych powiązaniach	192
Luźne powiązania w systemach ludzkich	193
Podsumowanie	195

Część IV. Narzędzia ułatwiające inżynierię w branży oprogramowania197

Rozdział 14. Narzędzia w dziedzinie inżynierii 199

Czym jest rozwój oprogramowania?	199
Testowalność jako narzędzie	201
Punkty pomiaru	204
Problemy z osiągnięciem testowalności	204
Jak zwiększyć testowalność?	208
Łatwość wdrażania	209
Szybkość	210
Kontrolowanie zmiennych	211
Ciągłe dostarczanie	212
Ogólne narzędzia wspomagające inżynierię	213
Podsumowanie	214

Rozdział 15. Współczesny inżynier oprogramowania	215
Inżynieria jako proces ludzki	216
Organizacje dokonujące przełomu w świecie cyfrowym	217
Skutki a mechanizmy	219
Trwałe i uniwersalne	221
Podstawy inżynierii	224
Podsumowanie	224