

Spis treści

Przedmowa 11

CZĘŚĆ PIERWSZA. ZASADY DLA PROGRAMISTÓW

Rozdział 1. Zanim zaczniesz... 17

Będziesz to robić, więc rób to dobrze 18

Rozdział 2. Postawa inżyniera 21

Rozdział 3. Niezwykła tajemnica programisty gwiazdora 25

Rozdział 4. Projekt oprogramowania w dwóch sentencjach 29

CZĘŚĆ DRUGA. ZŁOŻONOŚĆ OPROGRAMOWANIA I JEJ PRZYCZYNY 31

Rozdział 5. Wskazówki dotyczące nadmiernej złożoności 33

Rozdział 6. Drogi do stworzenia złożoności. Zepsuj swoje API 35

Rozdział 7. Kiedy wsteczna kompatybilność nie jest warta swojej ceny? 39

Rozdział 8. Złożoność to więzienie 43

CZĘŚĆ TRZECIA. PROSTOTA I PROJEKTOWANIE OPROGRAMOWANIA 45

Rozdział 9. Projektuj od początku 47

Ruszając dobrą drogą 48

Rozdział 10. Dokładność przyszłych przewidywań 49

Rozdział 11. Prostota i precyzja 53

Rozdział 12. Dwa to za dużo 57

Refaktoryzacja 58

Rozdział 13. Rozsądny projekt oprogramowania 61

Zła droga 62

Analiza złej drogi 64

Odnosząc to do grupy 65

Dobra droga 66

Przestrzegaliśmy praw tworzenia oprogramowania 69

CZĘŚĆ CZWARTA. DEBUGOWANIE 71

Rozdział 14. Czym jest bug? 73

Sprzęt 74

Rozdział 15. Źródło błędów 75

Spotęgowana złożoność 76

Rozdział 16. Spraw, by to nie powróciło 79

Spraw, by to nigdy nie powróciło - przykład 80

W głąb króliczej nory 84

Rozdział 17. Fundamentalna filozofia debugowania 85

Wyjaśnij błąd 87

Patrz na system 88

Znajdź prawdziwą przyczynę 89

Cztery kroki 90

CZĘŚĆ PIĄTA. INŻYNIERIA W ZESPOŁACH 93

Rozdział 18. Efektywna produktywność inżynierii 95

Co powinieneś zrobić? 97

Rozwiązanie 98

Wiarygodność i rozwiązywanie problemów 100

Blocker 101

Zmierzając w stronę podstawowego problemu 103

Rozdział 19. Mierząc produktywność dewelopera 107

Definicja "produktywności" 108

Czemu nie "linie kodu"? 108

Określając prawidłowy wskaźnik 110

A co, jeśli Twoim produktem jest kod? 111

A co z ludźmi, którzy pracują nad produktywnością deweloperów? 111

Wniosek 113

Rozdział 20. Jak radzić sobie ze złożonością kodu w firmie programistycznej 115

Krok pierwszy - lista problemów 117

Krok drugi - spotkanie 117

Krok trzeci - raport błędów 118

Krok czwarty - priorytetyzacja 119

Krok piąty - zadanie 120

Krok szósty - planowanie 121

Rozdział 21. W refaktoryzacji chodzi o funkcjonalności 123

Być efektywnym 124

Ustalając granice refaktoryzacji 127

Refaktoryzacja nie marnuje czasu, ona go oszczędza 128

Refaktoryzacja aż do jasności 128

Podsumowanie 130

Rozdział 22. Życzliwość i kodowanie 131

Oprogramowanie to ludzie 131

Przykład uprzejmości 132

Bądź miły i twórz lepsze oprogramowanie 134

Rozdział 23. Społeczność open source, w uproszczeniu 135

Utrzymanie współtwórców 136

Usuwać bariery 142

Zainteresować ludzi 145

Miej superpopularny produkt 146

Miej produkt napisany w popularnym języku programowania 146

Podsumowanie 147

CZĘŚĆ SZÓSTA. ROZUMIEĆ OPROGRAMOWANIE 149

Rozdział 24. Czym jest komputer? 151

Rozdział 25. Komponenty oprogramowania: struktura, akcja i wynik 155

Rozdział 26. Oprogramowanie na nowo: (I)SAR wyjaśnione 157

Struktura 158

Akcja 159

Wyniki 159

ISAR w pojedynczej linii kodu 160

Podsumowując SAR 161

Rozdział 27. Oprogramowanie jako wiedza 163

Rozdział 28. Cel technologii 167

Czy są jakieś kontrprzykłady tej zasady? 168

Czy postęp technologiczny jest "dobry"? 168

Rozdział 29. Prywatność, w uproszczeniu 171

Prywatność przestrzeni 171

Prywatność informacji 173

Podsumowanie prywatności 177

Rozdział 30. Prostota i bezpieczeństwo 179

Rozdział 31. Test-Driven Development i cykl obserwacji 183

Przykłady ODA 184

Proces wytwarzania i produktywność 185

Pierwsza ODA 187

Rozdział 32. Filozofia testowania 189

Wartość testu 190

Asercje testu 190

Granice testu 191

Założenia testu 191

Projekt testu 192

Testowanie end to end 192

Testy integracyjne 194

Testy jednostkowe 195

Rzeczywistość 196

Podróbki 197

Determinizm 199

Prędkość 200

Pokrycie 202

Wniosek - ogólny cel testowania 202

CZĘŚĆ SIÓDMA. MNIEJ DAWAĆ CIAŁA 203

Rozdział 33. Tajemnica sukcesu: mniej dawać ciała 205

Dlaczego to zadziało? 206

Rozdział 34. Jak odkryliśmy, co dawało ciała 209

Rozdział 35. Potęga "nie" 213

Rozpoznawanie złych pomysłów 215

Nie mając lepszego pomysłu 215

Wyjaśnienie, akceptacja i uprzejmość 217

Rozdział 36. Dlaczego programiści dają ciała 219

Czego się uczyć 222

Rozdział 37. Sekret szybkiego programowania: przestań myśleć 225

Zrozumienie 226

Rysowanie 227

Rozpoczynanie 228

Pomijanie kroku 229

Problemy fizyczne 229

To, co rozprasza 230

Zwątpienie w siebie 230

Fałszywe pomysły 231

Zastrzeżenie 231

Rozdział 38. Pycha dewelopera 233

Rozdział 39. Spójność nie oznacza jednolitości 235

Rozdział 40. Użytkownicy mają problemy, deweloperzy mają rozwiązania 237

Zaufanie i informacja 238

Problemy pochodzą od użytkowników 238

Rozdział 41. Natychmiastowa gratyfikacja = natychmiastowa porażka 241

Rozwiązania na dłuższą metę 242

Jak zniszczyć firmę tworząc oprogramowanie 243

Rozdział 42. Sukces bierze się z wykonania, nie z innowacji 245

Rozdział 43. Oprogramowanie doskonałe 247

1. Robi dokładnie to, co użytkownik mu polecił do wykonania 248

2. Zachowuje się dokładnie tak, jak użytkownik oczekuje, że się zachowa 249

3. Nie blokuje użytkownika przed komunikowaniem jego intencji 250

Doskonałość jest istotniejsza (ale nie w konflikcie) od prostoty kodu 252

Skorowidz 253