

# S P I S   T R E Ś C I

<b>Przedmowa wydawcy serii</b>	<b>19</b>
<b>Wstęp</b>	<b>23</b>
<b>0 autorze</b>	<b>31</b>
<b>CZĘŚĆ 1    Przyspieszenie</b>	<b>33</b>
<b>Rozdział 1    Sztuka czy nauka?</b>	<b>35</b>
1.1. Budowanie domu	36
1.1.1. Problem związany z projektami	36
1.1.2. Problem etapów	37
1.1.3. Zależności	38
1.2. Pielęgnacja ogrodu	39
1.2.1. Dzięki czemu ogród rośnie?	40
1.3. W kierunku inżynierii	40
1.3.1. Oprogramowanie jako rzemiosło	41
1.3.2. Heurystyki	42

## SPIS TREŚCI

1.3.3. Wcześniejsze poglądy na inżynierię oprogramowania	43
1.3.4. Ku inżynierii oprogramowania	45
1.4. Wniosek	46
<b>Rozdział 2 Listy kontrolne</b>	<b>48</b>
2.1. Pomaganie pamięci	48
2.2. Lista kontrolna dla nowego kodu źródłowego	50
2.2.1. Użyj Gita	51
2.2.2. Zautomatyzuj proces budowania	53
2.2.3. Włącz wszystkie komunikaty o błędzie	57
2.3. Włączanie narzędzi kontrolnych dla istniejącego kodu	63
2.3.1. Stopniowe ulepszenia	64
2.3.2. Zhakuj swoją organizację	65
2.4. Wniosek	66
<b>Rozdział 3 Radzenie sobie ze złożonością</b>	<b>68</b>
3.1. Cel	69
3.1.1. Zrównoważony rozwój	69
3.1.2. Wartość	70
3.2. Dlaczego programowanie jest trudne	72
3.2.1. Metafora mózgu	73
3.2.2. Więcej kodu się czyta, niż pisze	74
3.2.3. Czytelność	75
3.2.4. Praca intelektualna	76
3.3. W stronę inżynierii oprogramowania	79
3.3.1. Relacja z informatyką	79
3.3.2. Ludzki kod	80
3.4. Wniosek	81

<b>Rozdział 4</b>	<b>Pionowy wycinek</b>	<b>83</b>
4.1.	Zacznij od działającego oprogramowania	84
4.1.1.	Od otrzymania danych po ich utrwalenie	84
4.1.2.	Minimalny wycinek pionowy	85
4.2.	Chodzący szkielec	87
4.2.1.	Test charakterystyczny	88
4.2.2.	Zasada Arrange-Act-Assert	90
4.2.3.	Moderowanie analizy statycznej	92
4.3.	Podejście outside-in	95
4.3.1.	Przyjmowanie danych w formacie JSON	96
4.3.2.	Przesyłanie rezerwacji	99
4.3.3.	Test jednostkowy	103
4.3.4.	DTO i model domeny	106
4.3.5.	Fałszywy obiekt	109
4.3.6.	Interfejs repozytorium	110
4.3.7.	Tworzenie w repozytorium	110
4.3.8.	Konfiguracja zależności	112
4.4.	Kończenie wycinka	113
4.4.1.	Schemat	114
4.4.2.	Repozytorium SQL	116
4.4.3.	Konfiguracja uwzględniająca bazę danych	118
4.4.4.	Wykonanie testu dymnego	119
4.4.5.	Test graniczny z fałszywą bazą danych	120
4.5.	Wniosek	122
<b>Rozdział 5</b>	<b>Enkapsulacja</b>	<b>123</b>
5.1.	Zapisywanie danych	124
5.1.1.	Zasada Transformation Priority Premise	124
5.1.2.	Test parametryzowany	126
5.1.3.	Kopiowanie obiektu DTO do modelu domeny	127

## SPIS TREŚCI

5.2. Walidacja	129
5.2.1. Błędne daty	130
5.2.2. Procedura czerwone, zielone, refaktoryzacja	133
5.2.3. Liczby naturalne	136
5.2.4. Prawo Postela	139
5.3. Ochrona niezmienników	142
5.3.1. Zawsze poprawny	143
5.4. Wniosek	146
<b>Rozdział 6 Triangulacja</b>	<b>147</b>
6.1. Pamięć krótkoterminowa kontra długoterminowa	148
6.1.1. Zastany kod i pamięć	149
6.2. Wydajność	150
6.2.1. Zbyt wiele rezerwacji	151
6.2.2. Adwokat diabła	155
6.2.3. Istniejące rezerwacje	158
6.2.4. Adwokat diabła kontra czerwone, zielone, refaktoryzacja	160
6.2.5. Kiedy jest wystarczająco wiele testów?	163
6.3. Wniosek	164
<b>Rozdział 7 Dekompozycja</b>	<b>166</b>
7.1. Psucie się kodu	166
7.1.1. Wartości progowe	167
7.1.2. Złożoność cyklopatyczna	169
7.1.3. Reguła 80/24	171
7.2. Kod, który mieści się w mózgu	173
7.2.1. Kwiat sześciokątów	173
7.2.2. Spójność	176
7.2.3. Zazdrość o kod	180
7.2.4. Między wierszami	181
7.2.5. Parsuj, nie waliduj	182

## SPIS TREŚCI

7.2.6. Architektura fraktalna	186
7.2.7. Liczba zmiennych	190
7.3. Wniosek	190
<b>Rozdział 8 Projektowanie API</b>	<b>193</b>
8.1. Zasady projektowania API	194
8.1.1. Afordancja	194
8.1.2. Poka-Yoke	196
8.1.3. Pisz dla czytelników	198
8.1.4. Przedkładaj dobrze napisany kod nad komentarze	198
8.1.5. Zastąpienie nazw znakami x	199
8.1.6. Rozdzielenie poleceń i zapytań	202
8.1.7. Hierarchia komunikacji	205
8.2. Przykładowy projekt API	207
8.2.1. MaitreD'	208
8.2.2. Interakcja z opakowanym obiektem	210
8.2.3. Szczegóły implementacyjne	213
8.3. Wniosek	215
<b>Rozdział 9 Praca zespołowa</b>	<b>217</b>
9.1. Git	218
9.1.1. Komunikaty rewizji	219
9.1.2. Ciągła integracja	222
9.1.3. Małe rewizje	225
9.2. Zbiorowa własność kodu	228
9.2.1. Programowanie w parach	230
9.2.2. Mob Programming	231
9.2.3. Opóźnienia w inspekcji kodu	233
9.2.4. Odrzucenie zmian	235
9.2.5. Recenzje kodu	237
9.2.6. Żądania aktualizacji	239
9.3. Wniosek	240

<b>CZĘŚĆ II</b>	<b>Zrównoważony rozwój</b>	<b>243</b>
<b>Rozdział 10</b>	<b>Rozbudowywanie kodu</b>	<b>245</b>
10.1.	Flagi funkcji	246
10.1.1.	Flaga kalendarza	247
10.2.	Wzorzec dusiciela	252
10.2.1.	Dusiciel na poziomie metody	254
10.2.2.	Dusiciel na poziomie klasy	258
10.3.	Wersjonowanie	262
10.3.1.	Wcześniejsze ostrzeżenie	263
10.4.	Wniosek	264
<b>Rozdział 11</b>	<b>Edycja testów jednostkowych</b>	<b>265</b>
11.1.	Refaktoryzacja testów jednostkowych	265
11.1.1.	Zmiana sieci bezpieczeństwa	266
11.1.2.	Dodawanie nowego kodu testowego	267
11.1.3.	Osobna refaktoryzacja testów i kodu produkcyjnego	270
11.2.	Testy kończące się niepowodzeniem	276
11.3.	Wniosek	277
<b>Rozdział 12</b>	<b>Rozwiązywanie problemów</b>	<b>278</b>
12.1.	Zrozumienie	278
12.1.1.	Metoda naukowa	279
12.1.2.	Upraszczenie	280
12.1.3.	Gumowa kaczuszka	281
12.2.	Defekty	283
12.2.1.	Odtwórz błędy za pomocą testów	284
12.2.2.	Wolne testy	288
12.2.3.	Defekty niedeterministyczne	290
12.3.	Bisekcja	295
12.3.1.	Bisekcja za pomocą Gita	296
12.4.	Wniosek	300

<b>Rozdział 13</b>	<b>Separacja pojęć</b>	<b>302</b>
13.1.	Kompozycja	303
13.1.1.	Kompozycja a zagnieżdżona	303
13.1.2.	Kompozycja sekwencyjna	307
13.1.3.	Przezroczystość referencyjna	309
13.2.	Kwestie przekrojowe	312
13.2.1.	Zapisywanie zdarzeń w dziennikach	313
13.2.2.	Dekorator	314
13.2.3.	Co rejestrować w dziennikach	318
13.3.	Wniosek	320
<b>Rozdział 14</b>	<b>Rytm</b>	<b>322</b>
14.1.	Osobisty rytm	323
14.1.1.	Dzielenie czasu na bloki	323
14.1.2.	Rób przerwy	325
14.1.3.	Wykorzystuj czas celowo	326
14.1.4.	Pisanie bezwzrokowe	328
14.2.	Rytm zespołowy	329
14.2.1.	Regularne aktualizowanie zależności	329
14.2.2.	Zaplanuj inne rzeczy	331
14.2.3.	Prawo Conwaya	332
14.3.	Wniosek	333
<b>Rozdział 15</b>	<b>Typowi podejrzani</b>	<b>334</b>
15.1.	Wydajność	335
15.1.1.	Spuścizna z przeszłości	336
15.1.2.	Czytelność	337
15.2.	Bezpieczeństwo	340
15.2.1.	Metoda STRIDE	340
15.2.2.	Spoofing	342
15.2.3.	Tampering	342
15.2.4.	Repudiation	343

## SPIS TREŚCI

15.2.5. Information Disclosure	344
15.2.6. Denial of service	346
15.2.7. Elevation of privilege	347
15.3. Inne techniki	348
15.3.1. Testowanie oparte na właściwościach	348
15.3.2. Behawioralna analiza kodu	354
15.4. Wniosek	357
<b>Rozdział 16 Wycieczka</b>	<b>358</b>
16.1. Nawigacja	358
16.1.1. Ogólne spojrzenie	360
16.1.2. Organizacja pliku	363
16.1.3. Znajdowanie szczegółów	365
16.2. Architektura	367
16.2.1. Monolit	368
16.2.2. Cykle	369
16.3. Użycie	373
16.3.1. Uczenie się na podstawie testów	373
16.3.2. Słuchaj swoich testów	375
16.4. Wniosek	377
<b>Dodatek A Lista praktyk</b>	<b>379</b>
A.1. Adwokat diabła	379
A.2. Arrange-act-assert	379
A.3. Bisekcja	380
A.4. Czerwone, zielone, refaktoryzacja	380
A.5. Dekoratory dla aspektów przekrojowych	381
A.6. Dusiciel	381
A.7. Flaga funkcji	382
A.8. Funkcyjny rdzeń, imperatywna powłoka	382
A.9. Hierarchia komunikacji	382



## SPIS TREŚCI

A. 10. Inspekcja kodu	383
A. 11. Liczenie zmiennych	383
A. 12. Lista kontrolna dla nowego kodu źródłowego	383
A. 13. Model zagrożeń	384
A. 14. Odtwarzaj defekty w testach	384
A.15. Parsuj, nie waliduj	385
A. 16. Prawo Postela	385
A. 17. Programowanie sterowane X	385
A. 18. Regularnie aktualizuj zależności	385
A. 19. Reguła 50/72	386
A.20. Reguła 80/24	386
A.21. Rozdzielenie poleceń i zapytań	387
A.22. Rozdzielenie refaktoryzacji testów i refaktoryzacji kodu produkcyjnego	387
A.23. Transformation Priority Premise	387
A.24. Usprawiedliwaj wyjątki od reguły	388
A.25. Wersjonowanie semantyczne	388
A.26. Wycinek	388
A.27. Zastąp nazwy znakami X	388
A.28. Złożoność cykliczna	389
<b>Bibliografia</b>	<b>390</b>