



Michael C.
Feathers

Effektives Arbeiten mit **Legacy Code**

Refactoring und Testen bestehender Software

Stichwortverzeichnis

A

- Abdeckung 37
- Abfangpunkt 194, 197, 421
- Abhängigkeit Siehe Dependency
- Abhängigkeitsumkehrung 107
- Abschnürung 421
- Abstraktionsebene 267
- Adapt Parameter 164, 335, 338
- Aktivierungspunkt 60
- Alias-Parameter 154
- Änderung
 - Angst 109
 - Auswirkungen studieren 230
 - Dauer 101
 - Dependencies aufheben 103
 - Neukompilierung 107
 - Resignation 109
 - Verständlichkeit 101
 - Verzögerungszeit 102
 - von Legacy Code 227
- Änderungspunkt 43, 181, 421
- Angst
 - vor Änderungen 32, 109
- Anwendung
 - Struktur erkennen 233
- API-Aufrufe
 - in Legacy-Projekten 219
- Arbeit mit Legacy Code
 - Tools 69
- Arbeitsmoral 331
- Architektur 233
 - von JUnit 235
- Aufgabe
 - beim Design erkennen 265
 - einer Klasse 262
 - mit Heuristiken identifizieren 278
 - wie erkennen? 264
- Aufgaben
 - trennen 229
- Aufgabenbasierte Extraktion 225
- Aufruf
 - extrahieren 356

- Aufzählungsmethode 302
- Ausführungsdauer 37
- Automatisiertes Refactoring 70

B

- Beck, Kent 72, 238
- Bibliothek
 - Dependency 217
 - fremde Bibliothek einsetzen 219
 - Schnittstellen 217
- Break Out Method Object 315, 339
- Bug 29
- Build Dependencies 104
- Bulleted method 302

C

- C# 76
- C++
 - Test-Harnisch 149
- change point 421
- Charakterisierungs-Test 173, 179, 206
 - Heuristik 215
- Charakterisierungstest 421
- Class, Responsibility and Collaborations (CRC) 239
- Code 319
 - ändern 81
 - Änderungsaufwand 101
 - Aufgaben identifizieren 223
 - duplizierten Code entfernen III, II4
 - duplizierter Code 283
 - editieren 319
 - Funktionsbeschreibung 222
 - kleine Fragmente extrahieren 318
 - Open/Closed-Prinzip 300
 - prozeduraler Code 248
 - refaktorisieren 115
 - Seams 55
 - Skizzen anfertigen 228
 - ungenutzten Code löschen 231
 - unklarer 222

- Verhalten hinzufügen 252
- Verständlichkeit 101
- Code Siehe Software
- Command/Query Separation 169
- Compiler
 - als Stütze verwenden 325
- Construction Blob 138
- Conversation Scrutiny 241
- Coupling Count 313, 421
- Cover and Modify 33
- CppUnit 74
- CppUnitLite 72, 74
- CRC 239
 - Naked CRC 238
- C-Sprache 249
- Cunningham, Ward 238

- D**
- Decorator 96
- Decorator Pattern 96
 - Wrap Class 98
- Definition Completion 345
 - in C oder C++ 346
- Delegator 374
- Dependency 40, 217
 - aufheben 41, 43, 103, 335
 - Build Dependencies 104
 - einer Klasse aufheben 127
 - globale 140
 - in Legacy Code aufheben 42
 - in Unterklasse verschieben 394
 - Include-Dependencies 148
 - irritierende Parameter 127
 - Techniken zur Aufhebung 335
 - umkehren 107
 - und Packages 107
 - verborgene 134
 - verkettete Konstruktoren 138
 - von Schnittstellen 107
 - Warum? 45
 - zusammentragen 314
- Dependency-Inversion-Prinzip 107
- Design
 - Aufgaben erkennen 265
 - Einschnürpunkte 201
 - Namensvergabe 363
 - Refactoring 29
 - Testbarkeit 160
 - über Design reden 242
 - verbessern 29
- Duplizierter Code 283
 - entfernen 111, 114
 - Refactoring 283

- E**
- Edit and Pray 33
- Effect propagation 184
- Effect sketch 176, 421
- Effekt
 - erkennen 185
- Effektanalyse 173
 - Einkapselung 191
 - Lehren 188
 - Tools 186
 - und IDE-Unterstützung 174
 - Vorwärtsanalyse 179
- Effektfortpflanzung 184
- Effektskizze 175, 176, 421
 - Änderungspunkte 181
 - Funktionsskizze 269
 - Software-Qualität 177
 - vereinfachen 189
- Eingeschränktes-Überschreiben-Dilemma 218
- Einkapselung
 - Effektanalyse 191
 - Einschnürpunkte 201
 - und Testabdeckung 191
- Einmal-Dilemma 218
- Einschnürpunkt 194, 198, 200
 - Einkapselungen 201
 - Probleme 203
 - und Design 201
- Enabling Point 60
- Encapsulate Global References 347
- Expose Static Method 353
- Extract and Override Call 356
- Extract and Override Factory Method 138, 358
- Extract and Override Getter 360
- Extract Implementer 363
 - Beispiel 366
- Extract Interface 136, 153, 156, 368
 - nicht-virtuelle Funktionen 372
- Extract Method 415, 419
- Extract What You Know 312
- Extrahieren 312

- F**
- Factory-Methode
 - extrahieren 358

- Failing Test
 - Bestehen garantieren III, II4
 - kompilieren II0, II2, II3
 - schreiben II0, III, II2
 - Fake 249
 - Fake-Objekt 47, 42I
 - Ambivalenz 50
 - und Tests 49
 - Fassade 274
 - Feature sketch 267, 42I
 - Feedback 33, 103
 - Fehler 29
 - in Legacy Code finden 210
 - lokalisieren 37
 - Fehlerlokalisierung 36, 37
 - Fit 77
 - Fitness 78
 - Flow-Zustand 320
 - Fowler, Martin 69, 335, 415
 - Framework for Integrated Tests 77
 - free function 42I
 - Freie Funktion 42I
 - Funktion
 - durch Funktionszeiger ersetzen 397
 - hinzufügen 27
 - in alten Code integrieren II6
 - in Oberklasse verschieben 390
 - Methodenaufruf extrahieren 356
 - Funktionskizze 267, 42I
 - Effektskizze 269
 - Funktionszeiger 397
- G**
- Gamma, Erich 72
 - Gesprächsbeobachtung 24I
 - Getter
 - extrahieren 360
 - globale Referenz ersetzen 400
 - Gleaning Dependencies 314
 - Gleichnamige Methoden 387
 - Globale Dependency 140
 - Globale Referenz 347
 - durch Getter ersetzen 400
 - Globals 377
 - Green-Field-System 329
 - Große Klassen
 - Refactoring 262
- H**
- Haskell 189
 - Hauptzweck
 - einer Klasse 262
- I**
- Heuristik
 - um Aufgaben zu erkennen 265
 - zur Identifizierung von Aufgaben 278
 - Hill, Mike 76
 - HttpServletRequest 336
 - Hyperaware Editing 320
 - Hypercard 238
- J**
- IDE
 - Effektanalyse 174
 - Implementierungsebene 274
 - Implemeter
 - extrahieren 363
 - Include-Dependencies 148
 - Instanz-Delegator 374
 - Instanzvariable
 - ersetzen 405
 - Integrierte Tests 77
 - interception point 42I
 - Interface
 - extrahieren 368
 - Interface-Benennung 369
 - Interface-Segregation-Prinzip (ISP) 276
 - Introduce Instance Delegator 374
 - Introduce Sensing Variable 309
 - Introduce Static Setter 143, 147, 376
 - Irritierender Parameter 127
 - Isoliertes Testen 36
 - ISP 276
- K**
- Jeffries, Ron 77, 239
 - JUnit 73
 - Architektur 235
- K**
- Kernlogik
 - eines Systems 224
 - Klasse
 - Abstraktionsebenen 267
 - Aufgabe 262
 - benennen 243
 - charakterisieren 209
 - Dependencies aufheben 127
 - Größe als Testproblem 261
 - Hauptaufgabe suchen 274
 - Hauptzweck 262
 - in Test-Harnisch einfügen 127

- interne Beziehungen suchen 267
- konsistente Namen 297
- nach dem Extrahieren 281
- Name 243
- parallel bearbeiten 261
- Produktionsklasse 244
- Sprout Class 87
- Strategie der Zerlegung 278
- Taktik der Zerlegung 279
- Testklasse 244
- Utility-Klasse 374
- verborgene Entscheidungen suchen 266
- verborgene Klasse finden 202
- verkleinern 261
- Wrap Class 95
- Kollaborateur
 - simulieren 47
 - testen 47
- Konstruktor
 - parametrisieren 383
 - verketteter 138
- Kopplung
 - zeitliche 91
- Kopplungszahl 313, 421
- L**
- Lag time 102
- Lazy Getter 361
- Lean on the Compiler 165, 280, 325
- Legacy Code
 - ändern 42, 227
 - Arbeitsmoral 331
 - Dependency aufheben 42
 - Erfolgsfaktoren 330
 - Fehlen von Tests 109
 - Fehler finden 210
 - Schwierigkeiten 329
 - Test-Dilemma 247
 - und TDD 115
 - vs. Green-Field-Systeme 329
- Legacy-Code-Dilemma 40
- Legacy-Projekt
 - API-Aufrufe 219
 - schleichendes Wachstum 219
- Link Seam 422
- Link Substitution 382
- Linker 60
- Link-Seam 60
- Liskov Substitution Principle (LSP) 123
- Vererbung 124
- Listing Markup 229
- LSP 123
 - Programming by Difference 123
- M**
- Makro-Präprozessor 57, 249
- Manuelles Refactoring 308
- Martin, Robert C. 108
- Methode
 - ableiten 402
 - Aufruf extrahieren 356
 - einhüllen 91
 - extrahieren 230, 415
 - Factory-Methode 358
 - gleichnamige Methoden 387
 - in einem Test-Harnisch ausführen 159
 - konsistente Namen 297
 - Methoden gruppieren 265
 - Monster-Methode 301
 - Parameter 336
 - parametrisieren 386
 - private 159
 - Sequenzen suchen 316
 - skelettieren 316
 - Sprout Method 83
 - statische 353, 374
 - Struktur verstehen 229
 - testen 159, 209
 - überschreiben 402
 - verborgene Methoden suchen 266
 - Wrap Method 91
 - zwecks Test auswählen 173
- Methodengruppierung 265
- Methodenobjekt
 - herauslösen 315
- Meyer, Bertrand 169, 300
- Mock-Objekt 51, 71, 422
- Monster-Methode 301
 - Arten 301
 - Aufzählungsmethode 302
 - Refactoring 306
 - tief verschachtelte Methode 303
- Motivation
 - als Erfolgsfaktor 330
- N**
- Naked CRC 238, 239
- Namensvergabe 363
- Nebeneffekte 166
- .NET-Sprache 76

- Neukompilierung
 - bei Änderungen 107
- Nicht-virtuelle Methode
 - vs. virtuelle Methode 218
- Null Object Pattern 133
- NUnit 76
- O**
- Object Seam 255
- Objektorientierung 116
 - Für und Wider 258
 - von Projekten 247
- Objekt-Seam 64, 422
- Once-Dilemma 218
- OO 116
 - und Test-Driven Development 116
- Opdyke, Bill 69
- Open/Closed-Prinzip 300
- Optimierung
 - und Refactoring 30
- Orthogonalität 299
- P**
- Package
 - und Dependencies 107
- Pair Programming 320, 327
- Parallelarbeit 261
- Parameter 336
 - Alias-Parameter 154
 - Zwiebel-Parameter 152
- Parameterize Constructor 136, 148, 383
- Parameterize Method 148, 386
- Parametertyp
 - vereinfachen 388
- Pass Null 132, 153
- Pinch point 198, 422
- Platzhalter-Objekt 422
- Pointer 397
- Präprozessor 57
- Präprozessor-Direktive 250
- Präprozessor-Seam 57
- Preserve Signatures 322
- Primitivize Parameter 388
- Produktionsklasse 244
- Programm
 - als Text 53
- Programmieren
 - als Beruf 329
- Programming by Difference 116, 422
 - Beispiel 116
 - Klasse umbenennen 122
 - LSP 123
 - Vererbung 119
- Projekt
 - Objektorientierung 247
- Prozeduraler Code 248
- Pull Up Feature 390
- Push Down Dependency 394
- Q**
- Qualität
 - von Software 177
- R**
- Reasoning Forward 179
- Refactoring 120, 415
 - Anfang finden 286
 - automatisiertes 70
 - Definition 29, 69
 - duplizierter Code 283
 - Extract Method 415
 - große Klassen 262
 - Katalog der Methoden 335
 - Klasse umbenennen 122
 - manuelles 308
 - Scratch Refactoring 230, 277
 - Strategie 316
 - Tools 69
 - und Optimierung 30
 - von Monster-Methoden 306
 - Vorgehensweise 415
- Refactoring-Tools
 - Mängel 215
- Refaktorisieren 44
- Referenz
 - durch Getter ersetzen 400
 - einkapseln 347, 352
- Regressionstest 34
 - Probleme 34
- Replace Function with Function Pointer 397
- Replace Global Reference with Getter 400
- Resignation
 - bei Änderungen 109
- Responsibility-Based Extraction 225, 226
- Restricted-Override-Dilemma 218
- Risiko
 - Software-Änderung 31
- Risikomanagement 32
- Ruby 412

S

Scheinobjekt 51
 Scheme 189
 Schnittstelle
 benennen 369
 extrahieren 104, 368
 und Dependencies 107
 Schnittstellenebene 274
 Scratch Refactoring 230, 277, 278
 Seam 54, 422
 Aktivierungspunkt 60
 Arten 57
 Definition 55
 Enabling Point 60
 Seam-Modell 53
 sensing 45
 separation 45
 Sequenzen suchen 316
 Setter 408
 Statischer Setter 376
 Shadowing 280
 Signatur
 bewahren 322
 Single-Goal Editing 321
 Single-Responsibility Prinzip (SRP) 262, 274
 Singleton 218
 Singleton Design Pattern 141, 377
 Skelettierung 316
 Skin and Wrap the API 165, 224, 226
 Skizze
 von Code 228
 von Effekte 175
 Snarled method 303
 Software
 ändern 27
 Änderungen im Überblick 30
 Angst vor Änderungen 32
 Dependency-Probleme 40
 Design verbessern 29
 Fehler 29
 Fehler beseitigen 28
 Funktion hinzufügen 27
 Optimierung 30
 Qualität 177
 Refactoring 29, 69
 Risiken beim Ändern 31
 Risikomanagement 32
 Verhalten 28
 Software-Entwicklung
 Dependencies 217
 Zeit sparen 219

Software-Zwinge 34
 Sprout Class 87, 262
 Nachteile 91
 Schritte 90
 Vorteile 91
 Sprout Method 83, 262
 Nachteile 86
 Schritte 85
 Vorteile 86
 SRP 262, 274
 Implementierungsebene 274
 Schnittstellenebene 274
 Static Cling 375
 Statische Methode 353, 374
 Statischer Setter 376
 Strategie
 Refactoring 316
 Subclass and Override Method 145, 157, 170, 402
 Supersede Instance Variable 138, 405
 System
 Architektur 233
 Architektur vereinfachen 235
 Geschichte erzählen 234
 im Team analysieren 235
 Kernlogik 224
 mit API-Aufrufen verbessern 219

T

TDD 44, 110
 und Legacy Code 115
 TDD (Test-Driven Development) 422
 TDD-Algorithmus 116
 Teamarbeit
 Systeme analysieren 235
 Technik der kleinen Schritte 44
 Template
 umdefinieren 409
 Template Redefinition 409
 in C++ 411
 Test
 Abdeckung 37
 Ausführungsdauer 37
 Ausführungsebene 39
 automatisiertes Refactoring 70
 Charakterisierungs-Test 206
 Failing Test kompilieren 112, 113
 Failing Test schreiben 111, 112
 Fehlen in Legacy Code 109
 Fehlerlokalisierung 36, 37
 für eine Klasse schreiben 193

- für Methoden 209
 - Harnische 77
 - integrierte Tests 77
 - Isoliertes Testen 36
 - Methode auswählen 173
 - mögliche Probleme 36
 - schreiben 44
 - Sinn und Zweck 34
 - Testklassen 243
 - und Fake-Objekte 49
 - Unit-Test 36
 - Vorgehen mit Unit-Tests 35
 - welche schreiben? 205
 - wo speichern? 244
 - test coverage 422
 - test harness 36, 422
 - Testabdeckung 34, 39, 422
 - und Einkapselung 191
 - Test-Code 243
 - Test-Dilemma
 - bei prozeduralem Legacy Code 247
 - Test-Driven Development (TDD) 44, 110, 422
 - Algorithmus 110
 - und OO 116
 - Testfall
 - schreiben 110
 - Test-Framework
 - NUnit 76
 - Testgesteuerte Entwicklung 110
 - Testgetriebene Entwicklung 110
 - Test-Harnisch 36
 - C++ 149
 - Klasse einfügen 127
 - Methode ausführen 159
 - Testharnisch 422
 - testing subclass 422
 - TestKit 76
 - Testklasse 244
 - Testpunkt 43
 - Test-Tool 72
 - Test-Unterklasse 422
 - Text
 - umdefinieren 412
 - Text Redefinition 412
 - in C und C++ 413
 - in Ruby 413
 - Tool
 - für Effektanalysen 186
 - Tools
 - für Refactoring 69
 - für Tests 72
 - Legacy Code 69
 - Trennung 45
- U**
- Überwachung 45
 - Überwachungsvariable 309
 - UML 239
 - Unit-Test 36, 422
 - Ausführungsebene 39
 - gute Eigenschaften 37
 - maximale Dauer 38
 - Unit-Test-Harnisch 72
 - Utility-Klasse 374
- V**
- Variable
 - Überwachungsvariable 309
 - Variablen-Management 263
 - VB.NET 76
 - Verborgene Dependency 134
 - Verborgene Klasse 202
 - Vererbung 326
 - LSP 124
 - Programming by Difference 119
 - Verhalten
 - hinzufügen 252
 - von Software 28
 - Verkettete Konstruktoren 138
 - Verständlichkeit
 - von Code 101
 - Verzögerungszeit 102
 - Virtuelle Methode
 - vs. nicht-virtuelle Methode 218
 - Vorwärtsanalyse 179
- W**
- Wiederverwendung 217
 - Wrap Class 95
 - Decorator Pattern 98
 - Schritte 98
 - Wrap Method 91
 - Nachteile 94
 - Schritte 94
 - Vorteile 94
- X**
- xUnit-Test-Framework 72
- Z**
- Zeitliche Kopplung 91
 - Zugriffsschutz 163
 - Zwiebel-Parameter 152