

# Index

## A

**ABAP Object Unit**, 747

**ABAP Unit**, 747

**Abstract Setup Decorator**

defined, 449

example, 453

**acceptance tests**. *See also*

customer tests

defined, 785

why test?, 19

**accessor methods**, 785

**ACID**, 785

**acknowledgements**, xxvii–xxviii

**action components**, 280

**agile method**

defined, 785–786

property tests, 52

**AllTests Suite**

example, 594–595

introduction, 13

when to use, 593

**annotation**

defined, 786

Test Methods, 351

**Anonymous Creation Method**

defined, 417

example, 420

**Hard-Coded Test Data**

solution, 196

preface, xxi

**anonymous inner class**

defined, 786

Test Stub examples, 535–536

**Ant**, 753

**AntHill**, 753

**anti-pattern (AP)**

defined, 786

test smells, xxxv

**AOP (aspect-oriented programming)**

defined, 786

Dependency Injection, 681

retrofitting testability, 148

**API (application programming interface)**

Creation Methods, 416

database as SUT, 336

defined, 786

Test Utility Method, 600

**architecture, design for testability.**

*See design-for-testability*

**arguments**

messages describing, 371–372

as parameters (Dummy

Arguments), 729

role-describing, 725

- Arguments, Dummy**, 729
- Ariane 5 rocket**, 218
- aspect-oriented programming (AOP)**
  - defined, 786
  - Dependency Injection, 681
  - retrofitting testability, 148
- Assertion Message**
  - of Assertion Method, 364
  - pattern description, 370–372
- Assertion Method**
  - Assertion Messages, 364
  - calling built-in, 363–364
  - choosing right, 364–365
  - Equality Assertions, 365
  - examples, 368–369
  - Expected Exception
    - Assertions, 366
  - Fuzzy Equality Assertions, 365–366
  - implementation, 363
  - as macros, 364
  - motivating example, 367–368
  - overview, 362–363
  - refactoring, 368
  - Single-Outcome Assertions, 366–367
  - Stated Outcome Assertions, 366
- Assertion Roulette**
  - Eager Tests, 224–226
  - impact, 224
  - introduction, 14
  - Missing Assertion Message, 226–227
  - symptoms, 224
- assertions**
  - Built-in, 110–111
  - custom. *See* Custom Assertion
  - defined, 786
  - diagramming notation, xlii
  - Domain Assertions, 476, 481–482
  - improperly coded in Neverfail Tests, 274
  - introduction, 77
  - Missing Assertion Messages, 226–227
  - reducing Test Code Duplication, 114–119
  - refactoring, xlvii–xlix
  - Self-Checking Tests, 107–108
  - unit testing, 6
  - Verify One Condition per Test, 46–47
- assumptions**, xxxix–xl
- Astels, Dave**, 110
- asynchronous tests**
  - defined, 787
  - Hard-To-Test Code, 210–211
  - Humble Object, 696–697
  - Slow Tests, 255–256
  - testability, 70–71
- Attachment Method**
  - defined, 418
  - example, 421
- attributes**
  - defined, 787
  - dummy, 729
  - hiding unnecessary, 303–304
  - One Bad Attribute. *See* One Bad Attribute
  - parameters as, 608
  - Suite Fixture Setup, 442–443
  - Test Discovery using, 397
  - Test Selection, 403–405
- Automated Exercise Teardown**
  - defined, 505
  - example, 508
- Automated Fixture Teardown**, 504–505
- Automated Teardown**
  - ensuring Repeatable Tests, 27
  - examples, 507–508

- implementation, 504–505
  - Interacting Test Suites, 232
  - Interacting Tests solution, 231
  - motivating example, 505–506
  - overview, 503–504
  - of persistent fixtures, 99–100
  - refactoring, 506–507
  - resource leakage solution, 233
  - when to use, 504
  - automated unit testing**
    - author’s motivation, xxiv–xxv
    - fragile test problem, xxxi–xxxii
    - introduction, xxx–xxxii
- B**
- back door, defined, 787**
  - Back Door Manipulation**
    - control/observation points, 66–67
    - database as SUT API, 336
    - Expected State Specification, 464
    - fixture setup, 333–335
    - implementation, 330–332
    - motivating example, 332
    - overview, 327–328
    - refactoring, 333
    - setup, 329
    - teardown, 330
    - verification, 329–330
    - verification using Test Spy, 333
    - when to use, 328
  - Back Door Setup**
    - controlling indirect inputs, 128
    - fixture design, 59
    - Prebuilt Fixtures, 430–431
    - transient fixtures, 86
  - Back Door Verification, 130–133**
  - BDUF (big design upfront)**
    - defined, 787
    - design for testability, 65
    - test automation strategy, 49
  - Beck, Kent, xxii**
    - sniff test, xxxviii
    - Test Automation Frameworks, 301
    - test smells, 9
    - Testcase Class per Class, 618
    - xUnit, 57
  - Behavior Sensitivity**
    - cause of Fragile Tests, 242–243
    - caused by Overspecified Software, 246
    - defined, xxxi
    - smells, 14
  - behavior smells, 223–247**
    - Assertion Roulette. *See* Assertion Roulette
    - defined, 10–11, 788
    - Erratic Tests. *See* Erratic Test
    - Fragile Tests. *See* Fragile Test
    - Frequent Debugging. *See* Frequent Debugging
    - Manual Intervention. *See* Manual Intervention
    - overview, 13–15
    - Slow Tests. *See* Slow Tests
  - Behavior Verification**
    - approach to Self-Checking Tests, 108
    - examples, 472–473
    - implementation, 469–471
    - indirect outputs, 179–180
    - motivating example, 471–472
    - overview, 468–469
    - refactoring, 472
    - vs. state, 36
    - test results, 112–114
    - using Mock Objects. *See* Mock Object
    - using Test Spies. *See* Test Spy
    - using Use the Front Door First, 40

- verifying indirect outputs, 130–133
    - when to use, 469
  - behavior-driven development**
    - defined, 787–788
    - Testcase Class per Fixture usage, 632
  - Behavior-Exposing Subclass**
    - Test-Specific Subclass
      - example, 587
      - when to use, 580
  - Behavior-Modifying Subclass**
    - Defining Test-Specific Equality, 588–589
    - Substituted Singleton, 586–587
    - Test Stub, 584–585
    - when to use, 580
  - Bespoke Assertion. *See* Custom Assertion**
  - bimodal tests, 687**
  - binding, static**
    - defined, 809
    - Dependency Injection, 678–679
  - black box**
    - defined, 788
    - Remoted Stored Procedure Tests, 656
  - block closures**
    - defined, 788
    - Expected Exception Tests, 354–355
  - blocks**
    - cleaning up fixture teardown logic, l–liv
    - defined, 788
    - try/finally. *See* try/finally block
  - boundary values**
    - defined, 788
    - erratic tests, 238
    - Minimal Fixtures, 303
    - result verification patterns, 478
  - BPT (Business Process Testing)**
    - defined, 753
    - Recorded Tests, 280
    - Test Automation Frameworks, 301
  - Bug Repellent, 22**
  - Buggy Test**
    - introduction, 12–13
    - reducing risk, 181
    - symptoms, 260–262
  - Built-in Assertion**
    - calling, 363–364
    - introduction, 110–111
  - built-in self-tests**
    - defined, 788
    - test file organization, 164
  - built-in test recording**
    - defined, 281
    - example, 281–282
  - business logic**
    - defined, 789
    - developer testing, xxx
    - development process, 4–5
    - Layer Tests example, 344–345
    - testing without databases, 169–171
  - Business Process Testing (BPT).**
    - See* BPT (Business Process Testing)
- ## C
- Calculated Value. *See also* Derived Value**
    - Loop-Driven Tests, 615
    - Production Logic in Test solution, 205
  - Canoo WebTest**
    - defined, 753
    - Scripted Tests, 286
    - Test Automation Frameworks, 301
    - test automation tools, 53

- capacity tests, 52
- Capture/Playback Test.**
  - See* Recorded Test
- Chained Test**
  - customer testing, 6
  - examples, 459–460
  - implementation, 456–457
  - motivating example, 457–458
  - overview, 454–455
  - refactoring, 458
  - Shared Fixture strategies, 64–65
  - Shared Fixtures, 104–105, 322
  - when to use, 455–456
  - xUnit introduction, 57
- class attributes**
  - defined, 789
  - Test Discovery using, 397
  - Testcase Class Selection using, 404–405
- class methods**
  - defined, 789
  - with Test Helper, 645, 646
- class variables**
  - defined, 789
  - Suite Fixture Setup, 442
- classes**
  - diagramming notation, xlii
  - as fixtures, 59
  - Test Double, 569–570, 572–573
  - Testcase. *See* Testcase Class
- class-instance duality, 374**
- Cleanup Method, 602**
- closure, block**
  - defined, 788
  - Expected Exception Tests, 354–355
- Cockburn, Alistair**
  - pattern naming, 578
  - service layer tests, 339
- code**
  - inside-out development, 34–36
  - organization. *See* test organization
  - samples, xli–xlii
  - writing tests, 27–29
- code smells**
  - Conditional Test Logic. *See* Conditional Test Logic
  - defined, 10–11, 789
  - Hard-To-Test Code. *See* Hard-To-Test Code
  - obscure tests. *See* Obscure Test
  - Test Code Duplication. *See* Test Code Duplication
  - Test Logic in Production. *See* Test Logic in Production
  - types of, 16–17
- coding idioms**
  - defined, xxxv
  - design patterns, 792
- collisions**
  - Interacting Tests, 229–231
  - Shared Fixtures, 318
- Command object**
  - introduction, 82
  - Testcase Object as, 382
- Command-Line Test Runner**
  - Assertion Message, 371
  - defined, 379–380
  - introduction, 79
  - Missing Assertion Message, 226–227
- commercial recorded tests**
  - refactored, 283–284
  - tools, 282–283
- common location, Test Discovery, 397–398**
- Communicate Intent**
  - defined, 41
  - refactoring Recorded Tests to, 283–284
- compiler macro, Test Method Discovery, 395–396**
- Complex Teardown, 206–207**
- Complex Test. *See* Dependency Lookup**

- Component Broker.** *See* **Dependency Lookup**
- Component Registry, 688**
- component tests**
  - defined, 790
  - layer-crossing tests, 69
  - per-functionality, 52
  - test automation philosophies, 34–36
  - test strategy patterns, 340
- components**
  - defined, 790
  - depended-on component. *See* **DOC (depended-on component)**
- Composite object, defined, 82**
- Concerns, Separation of, 28–29**
- concrete classes, 581**
- Condition Verification Logic, 203–204**
- Conditional Test Logic**
  - vs. **Assertion Method**, 363
  - avoidance, 119–121
  - avoiding via **Custom Assertion**, 475
  - avoiding via **Guard Assertion**, 490–493
  - causes, 201–202
  - Complex Teardown**, 206–207
  - Condition Verification Logic**, 203–204
  - Flexible Tests**, 202–203
  - impact, 201
  - introduction, 16
  - Multiple Test Conditions**, 207–208
  - Production Logic in Test**, 204–205
  - symptoms, 200
  - Test Methods**, 155
- Configurable Mock Object, 546–547.**  
*See also* **Configurable Test Double**
- Configurable Registry, 691–692**
- Configurable Test Double**
  - examples, 564–567
  - implementation, 559–562
  - installing, 141–142
  - as kind of **Test Double**, 528
  - motivating example, 562–563
  - overview, 558
  - refactoring, 563
  - when to use, 559
- Configurable Test Stub.** *See also* **Configurable Test Double**
  - implementation, 532
  - indirect input control, 179
- Configuration Interface**
  - examples, 564–566
  - implementation, 560
- Configuration Mode**
  - example, 566–567
  - implementation, 560
- Constant Value.** *See* **Literal Value**
- constants in Derived Value, 718–722**
- constructing Mock Object, 546**
- Constructor Injection**
  - example, 683–684
  - implementation, 680–681
  - installing **Test Doubles**, 144
- Constructor Test**
  - defined, 351
  - example, 355–357
  - introduction, 77
- constructors**
  - defined, 790
  - problems with, 419
- containers, Humble Container Adapter, 698**
- Context Sensitivity**
  - avoiding via **Isolate the SUT**, 43–44
  - defined, 245–246
  - introduction, xxxii, 14
- continuous design, xxxiii**

- continuous integration**
    - avoiding Lost Tests, 270
    - defined, 791
    - impact of Data-Driven Tests, 290
    - steps, 14
  - control points**
    - defined, 791
    - testability, 66–67
  - Coplien, Jim, 576**
  - CORBA standards, 744**
  - cost effectiveness, Self-Checking Tests, 107–108**
  - costs, test automation, 20–21**
  - Covey, Stephen, 121**
  - CppUnit**
    - defined, 748
    - Test Automation Frameworks, 300
    - Test Method enumeration, 401
  - Creation Method**
    - Delegated Setup, 89–91, 411–414
    - eliminating unnecessary objects/attributes, 303–304
    - examples, 420–423
      - as Hard-Coded Test Data solution, 196
    - hybrid setup, 93
    - implementation, 418–419
    - motivating example, 419
    - overview, 415–416
    - persistent fixtures
      - teardown, 100
    - preface, xxiii
    - refactoring, 420
      - as Test Utility Method, 600
      - when to use, 416–418
      - writing simple tests, 28
  - cross-functional tests, 52–53**
  - cross-thread failure assertion, 274**
  - Cruise Control, 754**
  - CsUnit, 748**
  - CSV files, xUnit Data-Driven Test, 296**
  - CUnit, 748**
  - Cunningham, Ward, xxv, 290**
  - Custom Assertion**
    - as Conditional Verification Logic solution, 204
    - examples, 480–484
    - implementation, 477–478
    - Indirect Testing solution, 198–199
    - Irrelevant Information solution, 193
    - motivating example, 478–480
    - overview, 474–475
    - reducing Test Code Duplication, 116–117
    - refactoring, 480
    - Test Utility Methods, 602
    - when to use, 475–477
    - writing simple tests, 28
  - Custom Assertion test**
    - example, 483–484
    - implementation, 477–478
  - Custom Equality Assertion, 476**
  - customer tests**
    - defined, 791
    - Eager Tests cause, 225
    - Missing Unit Test, 271
    - overview, 5–6
    - per-functionality, 51
    - as Scripted Test, 285–287
  - Cut and Paste code reuse, 214–215**
- D**
- data access layer**
    - database testing, 172–173
    - defined, 791
    - Slow Tests with Shared Fixtures, 319

- data leaks**
  - avoiding with Delta Assertions, 486–487
  - Complex Teardown, 206
- Data Loader, Back Door Manipulation, 330–331**
- data minimization, 738–739**
- data population script, 434**
- Data Retriever, 331**
- Data Sensitivity**
  - defined, 243–245
  - introduction, xxxii, 14
- Data Transfer Object (DTO)**
  - defined, 793
  - result verification, 116
- Database Extraction Script, 331**
- Database Partitioning Scheme**
  - Data Sensitivity solution, 244–245
  - developer independence, 173
  - example, 653
  - Global Fixtures, 430
  - implementation, 652
- database patterns, 649–675**
  - Database Sandbox, 650–653
  - Stored Procedure Test, 654–660
  - Table Truncation Teardown, 661–667
  - Transaction Rollback Teardown, 668–675
- Database Population Script, 330**
- Database Sandbox**
  - database testing, 168
  - design for testability, 7
  - pattern description, 650–653
  - as Test Run Wars solution, 236–237
  - Unrepeatable Tests cause, 235
  - when to use, 650
- database testing, 167–174**
  - overview, 167–169
  - persistent fixtures, 313
  - testing without databases, 169–171
  - types of, 171–174
- Database Transaction Rollback Teardown, 674–675**
- databases**
  - fake. *See* Fake Database as SUT API, 336
  - teardown, 100
- Data-Driven Test**
  - customer testing, 5
  - Fit framework example, 296–297
  - frameworks, 300
  - implementation, 290
  - implemented as Recorded Test, 281
  - introduction, 83
  - motivating example, 293–294
  - overview, 288–289
  - principles, 48
  - reducing Test Code Duplication, 118–119
  - refactoring notes, 294
  - Test Suite Object Simulator, 293
  - using Fit framework, 290–292
  - via Naive xUnit Test Interpreter, 292–293
  - via Test Suite Object Generator, 293
  - when to use, 289–290
  - xUnit with CSV input file, 296
  - xUnit with XML data file, 294–295
- DB Schema per Test Runner**
  - developer independence, 173
  - implementation, 651–652

**DbUnit**

- Back Door Manipulation, 335
- defined, 748
- Expected State Specification, 464

**DDSteps, 754****Decorated Lazy Setup, 449–450****Decorator**

- Abstract Setup Decorator, 449, 453
- Parameterized Setup Decorator, 452–453
- Pushdown Decorator, 450
- Setup. *See* Setup Decorator
- Test Hook as, 710

**Dedicated Database Sandbox, 651****Defect Localization**

- customer testing, 5
- defined, 22–23
- Frequent Debugging, 248
- Keep Tests Independent Tests, 43
- right-sizing Test Methods, 154
- test automation philosophies, 34
- unit testing, 6
- Verify One Condition per Test, 45

**defining tests**

- introduction, 76–78
- suites of, 78–79

**delays. *See* Slow Tests****Delegated Setup**

- example, 413–414
- introduction, 77
- matching with teardown code, 98–99
- overview, 411–414
- of transient fixtures, 89–91
- when to use, 412

**Delegated Teardown**

- example, 514–515
- overview, 511
- of persistent fixtures, 98–99
- Table Truncation Teardown, 665

**Delta Assertion**

- avoiding fixture collisions, 101
- as Data Sensitivity solution, 245
- detecting data leakage with, 487
- examples, 488–489
- introduction, 111
- pattern description, 485–486

**depended-on component (DOC). *See* DOC (depended-on component)****dependencies**

- Interacting Tests, 230–231
- replacement with Test Doubles, 739
- replacing using Test Hooks, 709–712
- retrofitting testability, 148
- test automation philosophies, 34
- Test Dependency in Production, 220–221
- test file organization, 165

**Dependency Initialization Test, 352****Dependency Injection**

- design for testability, 7
- examples, 683–685
- implementation, 679–681
- installing Test Doubles via, 143–144
- Isolate the SUT, 44
- motivating example, 682
- overview, 678
- Persistent Fresh Fixtures
- avoidance, 62–63
- refactoring, 682
- testability improvement, 70
- when database testing, 171
- when to use, 678–679

**Dependency Lookup**

- design for testability, 7
- examples, 691–693
- implementation, 688–689
- installing Test Doubles, 144–145

- Isolate the SUT, 44
- motivating example, 690
- names, 693–694
- overview, 686
- Persistent Fresh Fixtures, 62–63
- refactoring, 690–691
- when database testing, 171
- when to use, 687–688
- Derived Expectation**
  - example, 720
  - when to use, 719
- Derived Input, 719**
- Derived Value**
  - examples, 719–722
  - overview, 718
  - when to use, 718–719
- design patterns, xxxv, 792**
- design-for-testability**
  - control points and observation points, 66–67
  - defined, 792
  - divide and test, 71–72
  - ensuring testability, 65
  - interaction styles and testability patterns, 67–71
  - overview, 7
  - Separation of Concerns, 28–29
  - test automation philosophies. *See* test automation philosophies
  - test automation principles, 40
  - test-driven testability, 66
- design-for-testability patterns, 677–712**
  - Dependency Injection. *See* Dependency Injection
  - Dependency Lookup. *See* Dependency Lookup
  - Humble Object. *See* Humble Object
  - Test Hooks, 709–712
- deterministic values, 238
- developer independence, 173
- developer testing**
  - defined, 792
  - introduction, xxx
- Developers Not Writing Tests, 13**
- development**
  - agile, 239
  - behavior driven, 632, 787–788
  - document-driven, 793
  - EDD. *See* EDD (example-driven development)
  - incremental, 33–34, 799–800
  - inside-out, 463
  - inside-out vs. outside in, 34–36
  - need-driven. *See* need-driven development
  - outside-in, 469
  - process, 4–5
  - TDD. *See* TDD (test-driven development)
  - test-first. *See* test-first development
  - test-last. *See* test-last development
- Diagnostic Assertion, 476–477**
- diagramming notation, xlii**
- Dialog, Humble. *See* Humble Dialog**
- direct output**
  - defined, 792–793
  - verification, 178
- Direct Test Method Invocation, 401**
- disambiguation, test fixtures, 814**
- Discovery, Test. *See* Test Discovery**
- Distinct Generated Values**
  - Anonymous Creation Methods, 417
  - Delegated Setup, 90
  - example, 725–726
  - Hard-Coded Test Data solution, 196
  - implementation, 724
  - Unrepeatable Tests solution, 235

- Distinct Values, 717
  - Do No Harm, 24–25
  - DOC (depended-on component)
    - Behavior Verification, 469
    - control points and observation points, 66–67
    - defined, 791–792
    - outside-in development, 35
    - replacing with Test Double.  
*See* Test Double
    - retrieving. *See* Dependency Lookup
    - terminology, xl–xli
    - Test Hook in, 712
  - Documentation, Tests as.  
*See* Tests as Documentation
  - document-driven development, 793
  - Domain Assertion
    - defined, 476
    - example, 481–482
  - domain layer
    - defined, 793
    - test strategy patterns, 337
  - domain model, 793
  - Don't Modify the SUT, 41–42
  - drivers, test
    - defined, 813
    - lack of Assertion Messages, 370
  - DRY (don't repeat yourself), 28
  - DTO (Data Transfer Object)
    - defined, 793
    - result verification, 116
  - Dummy Argument, 729
  - Dummy Attribute, 729
  - Dummy Object
    - configuring, 141–142
    - defined, 133
    - as Test Double, 134–135, 526
    - as value pattern, 728–732
    - xUnit terminology, 741–744
  - dynamic binding
    - defined, 793
    - use in Dependency Injection, 679
  - Dynamically Generated Mock Object, 550
  - Dynamically Generated Test Double
    - implementation, 561–562
    - providing, 140–141
  - Dynamically Generated Test Stub, 534–535
- ## E
- Eager Test
    - Assertion Roulette, 224–226
    - Fragile Tests, 240
    - Obscure Tests, 187–188
    - right-sizing Test Methods, 154
  - EasyMock
    - defined, 754
    - Test Doubles, 140
  - eCATT
    - defined, 754
    - Test Automation Frameworks, 301
  - Eclipse
    - Debugger, 110
    - defined, 754
  - economics of test automation, 20–21
  - EDD (example-driven development)
    - defined, 794
    - tests as examples, 33
  - efficiency, 11
  - emergent design
    - vs. BDUE, 65
    - defined, xxxiii, 794
  - encapsulation
    - Creation Method. *See* Creation Method
    - Dependency Lookup
      - implementation, 688–689
    - indirect outputs and, 126

- Indirect Testing solution, 198
- SUT API. *See* SUT API
- Encapsulation
- using Test Utility Methods.
  - See* Test Utility Method
- endoscopic testing (ET)**
  - defined, 794
  - Mock Objects, 545
  - Test Doubles, 149
- Ensure Commensurate Effort and Responsibility, 47–48**
- Entity Chain Snipping**
  - example, 536–537
  - testing with doubles, 149
  - when to use, 531
- entity object, 794**
- enumeration**
  - customer testing, 5
  - Suite of Suites built using, 389–391
  - test conditions in Loop-Driven Tests, 614–615
  - Test Enumeration, 399–402
  - Test Suite Object built using, 388
  - xUnit organization
    - mechanisms, 153
- Equality, Sensitivity**
  - Fragile Tests, 246
  - test-first development, 32
- Equality Assertion**
  - Assertion Methods, 365
  - Custom, 476
  - example, 368
  - Guard Assertion as, 491
  - introduction, 110
  - reducing Test Code
    - Duplication, 115
  - unit testing, 6
- Equality Pollution, 221–222**
- equals method**
  - Equality Pollution, 221–222
  - Expected State Specification, 464
  - reducing Test Code Duplication, 115–116
- equivalence class**
  - Behavior Smells, 238
  - defined, 794
  - Untested Code, 272
- Erratic Test**
  - Automated Teardown and, 27
  - customer testing, 5
  - database testing, 168–169
  - impact, 228
  - Interacting Test Suites, 231–232
  - Interacting Tests, 229–231
  - introduction, 14–16
  - Lonely Tests, 232
  - Nondeterministic Tests, 237–238
  - Resource Leakage, 233
  - Resource Optimism, 233–234
  - symptoms, 228
  - Test Run Wars, 235–237
  - troubleshooting, 228–229
  - Unrepeatable Tests, 234–235
- essential but irrelevant fixture setup, 425**
- ET (endoscopic testing)**
  - defined, 794
  - Mock Object use for, 149, 545
- example-driven development (EDD)**
  - defined, 794
  - tests as examples, 33
- examples, tests as, 33**
- exclamation marks, xlii**
- Executable, Humble. *See* Humble Executable**
- Executable Specification, 51**
- execution optimization, 180–181**
- exercise SUT**
  - defined, 794
  - test phases, 359
- expectations**
  - defined, 795
  - Derived Expectations, 719, 720
  - messages describing, 371–372
  - naming conventions, 159

- Expected Behavior Specification**
    - defined, 470–471
    - example, 473
  - Expected Behavior Verification**
    - defined, 112
    - indirect outputs, 131–132
  - Expected Exception Assertion**
    - defined as Assertion Method, 365–366
    - example, 369
  - Expected Exception Test**
    - Conditional Verification Logic solution, 204
    - introduction, 77
    - as Test Method, 350–351
    - using block closure, 354–355
    - using method attributes, 354
    - using try/catch, 353–354
  - Expected Object**
    - reducing Test Code Duplication, 115–116
    - refactoring tests, xlv–xlviii
    - State Verifications, 109, 466–467
    - unit testing, 6
  - expected outcome, 795**
  - Expected State Specification, 464–465**
  - expected values, 546–547**
  - exploratory testing**
    - cross-functionality, 53
    - defined, 795
    - Scripted Tests, 287
  - Expression Builders, 564–566**
  - expressiveness gaps, 27–28**
  - external resource setup, 740**
  - external result verification, 111–112**
  - external test recording, 280**
  - Extract Method**
    - Creation Methods, 418
    - Custom Assertions, 117
    - Delegated Setup, 89
    - as Eager Tests solution, 225
    - example, xlvii
      - in persistent fixture teardown, 98
      - refactoring Recorded Tests, 283
  - Extract Testable Component, 197, 735–736**
  - eXtreme Programming**
    - defined, 795
    - projects affected by Slow Tests, 319–321
  - eXtreme Programming Explained* (Beck), xxii**
- ## F
- factories**
    - defined, 795
    - Factory Method, 592–593
    - Object Factories, 145, 688
  - failed tests**
    - due to Unfinished Test Assertions, 494–497
    - implementation, 80
  - “Fail-Pass-Pass”, 234–235**
  - failure messages**
    - Assertion Messages, 370–372
    - Built-in Assertions, 110–111
    - removing “if” statements, 120
    - Single-Outcome Assertions, 366–367
  - Fake Database**
    - avoiding persistence, 101
    - database testing, 170
    - example, 556–557
    - Slow Component Usage solution, 254
    - Slow Tests with Shared Fixtures, 319
    - when to use, 553
  - Fake Object**
    - configuring, 141–142
    - customer testing, 6
    - defined, 134
    - examples, 556–557
    - implementation, 553–554

- motivating example, 554–555
  - optimizing test execution, 180
  - overview, 551–552
  - refactoring, 555–556
  - as Test Double, 139, 525
  - when to use, 552–553
  - xUnit terminology, 741–744
- Fake Service Layer**, 553
- Fake Web Services**, 553
- false negative**, 795
- false positive**, 795–796
- fault insertion tests**
  - defined, 796
  - per-functionality, 52
- Feathers, Michael**, 40
  - Highly Coupled Code solution, 210
  - Humble Object, 708
  - pattern naming, 576
  - retrofitting testability, 148
  - Self Shunt, 578
  - test automation roadmap, 176
  - Unit Test Rulz*, 307
- features**
  - defined, 796
  - right-sizing Test Methods, 156–157
  - Testcase Class per. *See* Testcase Class per Feature
  - visibility/granularity in Test-Specific Subclass, 581–582
- feedback in test automation**, xxix
- file contention**. *See* Test Run War
- File System Test Runner**, 380
- Finder Method**
  - accessing Shared Fixtures, 103–104
  - Mystery Guests solution, 190
  - when to use, 600–601
- fine-grained testing**, 33–34
- Fit**
  - Data-Driven Test example, 296–297
  - Data-Driven Test implementation, 290–292
  - defined, 754–755, 796
  - Expected State Specification, 464
  - fixture definition, 59, 86
  - fixture vs. Testcase Class, 376
  - Scripted Tests implementation, 286
  - Test Automation Framework, 301
  - test automation tools, 54
  - tests as examples, 33
  - vs. xUnit, 57
- Fitnessse**
  - Data-Driven Test implementation, 290
  - defined, 755
  - Scripted Test implementation, 286
- “Five Whys”, 11**
- fixture design**
  - upfront or test-by-test, 36
  - Verify One Condition per Test, 46
  - xUnit sweet spot, 58
- fixture holding class variables**, 797
- fixture holding instance variables**, 797
- fixture setup**
  - Back Door Manipulation, 329, 333–335
  - cleaning up, liv–lvii
  - defined, 797
  - Delegated Setup, 89–91
  - external resources, 740
  - Four-Phase Test, 358–361
  - Fresh Fixtures, 313–314
  - hybrid setup, 93

- Implicit Setup, 91–93
- In-Line Setup, 88–89
- introduction, 77
- matching with teardown code, 98–99
- Shared Fixtures, 104–105
- speeding up with doubles, 149–150
- strategies, 60
- fixture setup patterns, 407–459**
  - Chained Test. *See* Chained Test Creation Method. *See* Creation Method
  - Delegated Setup, 411–414
  - Implicit Setup, 424–428. *See also* Implicit Setup
  - In-line Setup, 408–410. *See also* In-line Setup
  - Lazy Setup. *See* Lazy Setup
  - Prebuilt Fixture. *See* Prebuilt Fixture
  - Setup Decorator. *See* Setup Decorator
  - Suite Fixture Setup. *See* Suite Fixture Setup
- Fixture Setup Testcase, 456**
- fixture strategies**
  - overview, 58–61
  - persistent fresh fixtures, 62–63
  - shared fixture strategies, 63–65
- fixture teardown**
  - avoiding in persistent fixtures, 100–101
  - Back Door Manipulation, 330
  - cleaning up, l–liv
  - Complex Teardown, 206–207
  - data access layer testing, 173
  - defined, 797
  - fixture strategies, 60
  - Four-Phase Test, 358–361
  - Implicit Setup, 426
  - introduction, 77
  - Lazy Setup problems, 439
  - persistent fixtures, 97–100
  - Persistent Fresh Fixtures, 314
  - refactoring, l–liv
  - Shared Fixtures, 105
  - transient fixtures, 93–94
  - Verify One Condition per Test, 46
- fixture teardown patterns, 499–519**
  - Automated Teardown, 503–508
  - Garbage-Collected Teardown, 500–502
  - Implicit Teardown, 516–519. *See also* Implicit Teardown
  - In-line Teardown, 509–515. *See also* In-line Teardown
  - Table Truncation Teardown, 661–667
  - Transaction Rollback Teardown. *See* Transaction Rollback Teardown
- fixtures**
  - collisions, 100–101
  - database testing, 168–169
  - defined, 796, 814
  - Four-Phase Test, 358–361
  - fresh. *See* Fresh Fixture
  - introduction, 78
  - Minimal. *See* Minimal Fixture
  - right-sizing Test Methods, 156–157
  - Shared. *See* Shared Fixture
  - speeding up setup with doubles, 149–150
  - Standard. *See* Standard Fixture
  - Testcase Class as, 376
  - Testcase Class per Fixture. *See* Testcase Class per Fixture
  - transient. *See* transient fixtures

- Flexible Test, 202–203
- fluent interface, 797
- For Tests Only, 219–220
- foreign-key constraints, 663
- forms, pattern, xxxiv–xxxv
- Four-Phase Test
  - Custom Assertions, 478
  - fixture design, 59
  - introduction, 76–78
  - Mock Object patterns, 546
  - pattern description, 358–361
  - unit testing, 6
  - Verify One Condition per Test, 46
- Fowler, Martin, xxvi
  - code smells, 16
  - Creation Methods, 418
  - Custom Assertions, 117
  - Cut and Paste code reuse, 215
  - Delegated Setup, 89, 413
  - Eager Tests solution, 225
  - Multiple Test Conditions
    - solution, 208
  - pattern forms, xxxvi
  - refactoring, xxxix
  - refactoring Recorded Tests, 283
  - reusable test logic, 123
  - self-testing code, xxi
  - Standard Fixtures, 306
  - state vs. behavior
    - verification, 36
  - test smells, 9
  - Testcase Object exception, 385
- Fragile Fixture
  - defined, 246–247
  - introduction, 14, 16
  - setUp method misuse, 93
- Fragile Test
  - Behavior Sensitivity, 242–243
  - Buggy Tests, 260
  - causes, 240–241
  - Context Sensitivity, 245–246
  - Data Sensitivity, 243–245
  - Fragile Fixture, 246–247
  - High Test Maintenance
    - Cost, 266
  - impact, 239
  - Interface Sensitivity, 241–242
  - introduction, xxiii, xxxi–xxxii, 13–14
  - Overspecified Software, 246
  - Sensitivity Equality, 246
  - symptoms, 239
  - troubleshooting, 239–240
- frameworks
  - Fit. *See* Fit
  - Test Automation Framework, 75, 298–301
- Frequent Debugging
  - avoidance with Custom
    - Assertion, 475
  - causes, 248–249
  - impact, 249
  - introduction, 15
  - solution patterns, 249
  - symptoms, 248
- Fresh Fixture
  - Creation Method. *See* Creation Method
  - Data Sensitivity solution, 244–245
  - Delegated Setup, 411–414
  - example, 316
  - fixture strategies, 60–61
  - implementation, 312
  - Implicit Setup, 424–428
  - Interacting Tests solution, 231
  - motivating example, 315
  - Mystery Guests solution, 190
  - overview, 311
  - persistent, 62–63, 313–314. *See also* persistent fixtures
  - refactoring, 315

- setup, 313–314
  - test automation philosophies, 36
  - Test Run Wars solution, 236–237
  - transient, 61–62. *See also*
    - transient fixtures
  - Transient Fresh Fixture, 314
  - when to use, 312
- front door, 797**
- Front Door First**
  - defined, 40–41
  - Overspecified Software
    - avoidance, 246
- Fully Automated Test**
  - behavior smells and, 15
  - Communicate Intent and, 41
  - Manual Fixture Setup
    - solution, 251
  - minimizing untested code, 44–45
  - running, 25–26
  - unit testing, 6
- functional tests**
  - defined, 798
  - per-functionality, 50–52
- Fuzzy Equality Assertion**
  - defined, 365–366
  - example, 368–369
  - external result verification,
    - 111–112
  - introduction, 110
- G**
- Gamma, Erich, 57**
- garbage collection, 798**
- Garbage-Collected Teardown**
  - design-for-testability, 7
  - pattern description, 500–502
  - persistent fixtures, 97
  - transient fixtures, 87–88
- General Fixture**
  - database testing, 169
  - defined, 187
  - misuse of setUp method,
    - 92–93
  - Obscure Tests, 190–192
  - Slow Tests, 255
- Generated Value, 723–727**
- Geras, Adam, 280**
- Global Fixture, 430**
- global variables**
  - defined, 798
  - instance variables as, 92
- goals, test automation.**  
*See test automation goals*
- Gorts, Sven, 537**
- granularity**
  - test automation tools and,
    - 53–54
  - Test-Specific Subclass,
    - 581–582
- Graphical Test Runner**
  - clicking through to test code,
    - 226–227
  - defined, 378–379
  - green bar, 26
  - introduction, 79, 300
- graphical user interface (GUI).**  
*See GUI (graphical user interface)*
- green bar, defined, 798**
- Guaranteed In-Line Teardown, 233**
- Guard Assertion**
  - Conditional Verification Logic
    - solution, 203–204
  - introduction, 80
  - pattern description, 490–493
  - removing “if” statements in
    - Test Method, 120
- GUI (graphical user interface)**
  - defined, 799
  - design for testability, 7
  - Interface Sensitivity, xxxii
  - testing with Humble
    - Dialogs, 696

## H

**Hand-Built Test Double.** *See also*

**Hard-Coded Test Double**

Configurable Test Double,  
560–561

providing, 140–141

**Hand-Coded Mock Object, 548–550**

**hand-coded teardown, 97–98**

**Hand-Coded Test Stub, 533–534**

**Hand-Scripted Test.** *See also*

**Scripted Test**

introduction, 75

tools for automating, 53–54

**Hand-Written Test.** *See* **Scripted Test**

**happy path**

defined, 799

Responder use, 530

Simple Success Tests, 349–350

test automation roadmap,  
177–178

**Hard-Coded Mock Object.** *See* **Hard-Coded Test Double**

**Hard-Coded Setup Decorator**

defined, 449

example, 451–452

**Hard-Coded Test Data**

causing Obscure Tests, 194–196

defined, 187

introduction, lv–lvii, 16

**Hard-Coded Test Double**

configuring, 141–142

implementation, 527, 569–571

motivating example, 571

naming patterns, 576–578

overview, 568

refactoring, 572

Self Shunt/Loopback, 573

Subclassed Inner Test Double,  
573–575, 578

Test Double Class, 572–573

testing with, 140–142

when to use, 569

**Hard-Coded Test Spy.** *See*

**Hard-Coded Test Double**

**Hard-Coded Test Stub.** *See also*

**Hard-Coded Test Double**

implementation, 531–532

indirect input control, 179

**Hard-Coded Value, 103**

**Hard-To-Test Code**

Asynchronous Code, 210–211

Buggy Tests, 261

code smells, 16

Developers Not Writing  
Tests, 264

divide and test, 71–72

High Test Maintenance Cost,  
266–267

Highly Coupled Code, 210

impact, 209

solution patterns, 209

symptoms, 209

Untestable Test Code, 211–212

**hierarchy of test automation needs,  
176–177**

**High Test Maintenance Cost**

Conditional Test Logic, 200

In-Line Setup, 89

introduction, 12–13

smell description, 265–267

**Higher Level Language**

Custom Assertion, 117

Interface Sensitivity solution, 241

xUnit sweet spot, 58

**Highly Coupled Code, 210**

**historical patterns and smells, xxxviii**

**Hollywood principle**

defined, 56, 799

test results, 79

**Hook, Test.** *See* **Test Hook**

**HTML user interface sensitivity, xxxii**

- HttpUnit**, 755
- Humble Container Adapter**, 698
- Humble Dialog**
  - design-for-testability, 7
  - example, 706–708
  - Hard-To-Test Code, 72
  - minimizing untested code, 45
  - when to use, 696–697
- Humble Executable**
  - asynchronous tests, 70–71
  - minimizing untested code, 44
  - motivating example, 700–702
  - Neverfail Test solution, 274
  - when to use, 697
- Humble Object**
  - Asynchronous Code solution, 211
  - Humble Dialog, 706–708
  - Humble Transaction
    - Controller, 708
  - implementation, 698–700
  - motivating example, 700–702
  - overview, 695–696
  - Poor Man's Humble Executable, 703
  - refactoring, 702
  - True Humble Executable, 703–706
  - when to use, 696–698
- Humble Transaction Controller**
  - data access layer testing, 173
  - example, 708
  - when to use, 697–698
- Hurst, John**, 670–671
- hybrid setup**, 93
- I**
- IDE (integrated development environment)**
  - defined, 799
  - introduction, 78
  - refactoring, xxxix
- Idea**, 755
- IeUnit**
  - defined, 748
  - Graphical Test Runner, 378
- “if” statements**
  - Conditional Test Logic, 201
  - Guard Assertions, 490–491
  - removing, 120
- IFixtureFrame**, 442
- ignoring tests**, 270
- Immutable Shared Fixture**
  - defined, 323
  - example, 326
  - Interacting Tests solution, 231
  - introduction, 61, 65
  - vs. Irrelevant Information, 192
  - Test Run Wars solution, 237
- impact**
  - Assertion Roulette, 224
  - Asynchronous Code, 211
  - Buggy Tests, 260
  - Conditional Test Logic, 201
  - Developers Not Writing Tests, 263
  - Equality Pollution, 221
  - Erratic Tests, 228
  - Flexible Tests, 203
  - Fragile Tests, 239
  - Frequent Debugging, 249
  - General Fixtures, 191–192
  - Hard-Coded Test Data, 195
  - Hard-To-Test Code, 209
  - High Test Maintenance Cost, 265
  - Highly Coupled Code, 210
  - Indirect Testing, 197
  - Irrelevant Information, 193
  - Manual Intervention, 250
  - Mystery Guests, 189
  - Neverfail Tests, 274
  - Nondeterministic Tests, 237

- Obscure Tests, 186
- Production Bugs, 268
- Slow Tests, 253
- Test Code Duplication, 214
- Test Dependency in
  - Production, 221
- Test Hooks, 218–219
- Test Logic in Production, 217
- Test Run Wars, 236
- For Tests Only, 220
- Untestable Test Code, 211
- Untested Requirements, 273
- Implicit Setup**
  - vs. Four-Phase Test, 360–361
  - introduction, 7, 77
  - matching with teardown code, 98–99
  - pattern description, 424–428
  - pattern naming, 577
  - reusing test code with, 162
  - transient fixtures, 91–93
- Implicit Teardown**
  - Complex Teardown solution, 206–207
  - database, 100
  - vs. Four-Phase Test, 360–361
  - pattern description, 516–519
  - persistent fixtures, 98–99
  - Self-Checking Tests with, 108
- Imposter.** *See* **Test Double**
- incremental delivery**
  - agile development, 239
  - defined, 799
- incremental development**
  - defined, 799–800
  - test automation philosophies, 33–34
- Incremental Tabular Test**
  - implementation, 609–610
  - Parameterized Test patterns, 613–614
- incremental tests,** 322
- In-Database Stored Procedure Test**
  - database testing, 172
  - example, 658–659
  - implementation, 655–656
- Independent Tabular Test,** 612–613
- independent testing.** *See* **Keep Tests**
- Independent**
- indirect input**
  - alternative path verification, 179
  - controlling, 128–129
  - controlling in Layer Tests, 341
  - defined, 800
  - importance of, 126
  - Test Doubles, 125–126
- indirect output**
  - Behavior Verification. *See* Behavior Verification
  - defined, 800
  - importance of, 126–127
  - registries, 541
  - Test Doubles, 125–126
  - verification, 130–133, 178–180
  - verifying in Layer Tests, 341
- Indirect Testing**
  - defined, 187
  - Fragile Tests cause, 240
  - Obscure Tests cause, 196–199
  - testability, 70–71
- Infrequently Run Test**
  - Frequent Debugging cause, 248–249
  - Production Bugs cause, 268–269
- inheritance**
  - reusing test code, 164
  - reusing test fixtures, 62
- injected values, Test Stub.** *See* **Test Stub**
- Injection, Parameter.** *See* **Parameter Injection**
- in-line Four Phase Test,** 360

- in-line resources, 736–737**
- In-line Setup**
  - introduction, 77
  - matching with teardown code, 98–99
  - Mystery Guest solution, 190
  - pattern description, 408–410
  - transient fixtures, 88–89
- In-line Teardown**
  - examples, 512–515
  - implementation, 510–511
  - motivating example, 511
  - Naive In-Line Teardown, 512
  - overview, 509
  - of persistent fixtures, 98–99
  - refactoring, 512
  - when to use, 510
- In-Memory Database, 553**
- inner class**
  - anonymous, 535–536, 786
  - defined, 800
- Inner Test Double**
  - example, 573–574
  - Hard-Coded Test Double
    - implementation, 570–571
  - Subclassed from Pseudo-Class, 574–575, 578
  - Test Spy implementation, 541
- input**
  - derived, 719
  - indirect. *See* indirect input
  - naming conventions, 158–159
- inside-out development**
  - vs. outside-in development, 34–36
  - State Verification, 463
- installing Test Doubles, 528**
  - Dependency Injection, 143–144, 679–680
  - Dependency Lookup, 144–145
  - Fake Object, 554
  - introduction, 143
  - Mock Object, 547
  - retrofitting testability, 146–148
- instance methods**
  - defined, 800–801
  - with Test Helper, 645, 647
- instance variables**
  - converting for Implicit Setup, 427
  - Data-Driven Tests using Fit Framework, 297
  - defined, 801
  - Fresh Fixtures, 313
  - as global variables, 92
  - Reuse Tests for Fixture Setup, 418–419
  - with Test Specific Subclass, 558
  - Testcase Class per Fixture, 632
- instances**
  - reusing, 63
  - Testcase Object exception, 384–385
- integrated development environment (IDE). *See* IDE (integrated development environment)**
- Integration Build, 4**
- Intent-Revealing Name**
  - Custom Assertion, 474–475
  - Implicit Setup, 92
  - Parameterized Test, 608
  - Test Utility Method, 602–603
- Interacting Test Suites, 231–232**
- Interacting Tests**
  - avoiding with Database Sandbox, 650–653
  - avoiding with Delta Assertion, 111, 486
  - caused by Shared Fixture, 63
  - Chained Tests, 455
  - customer testing, 5–6
  - database testing, 169

- Erratic Test cause, 229–231
- introduction, 15
- Keep Tests Independent, 43
- interaction point, 801**
- interaction styles, 67–71**
- Interaction Testing. *See* Behavior Verification**
- Interface Sensitivity**
  - defined, 241–242
  - introduction, xxxii, 13
- interfaces**
  - Configuration Interface, 560
  - defined, 801
  - GUI. *See* GUI (graphical user interface)
  - outgoing interface, 804–805
  - standard test, 378
  - Test Runner. *See* Test Runner
  - Use the Front Door First, 40–41
- internal recording tools, 56**
- interpreters in Data-Driven Tests.**  
*See* Data-Driven Test
- Intervention, Manual. *See* Manual Intervention**
- Introduce Explaining Variable refactoring, lvii–lviii**
- IoC (inversion of control) framework**
  - defined, 801
  - for Dependency Injection, 680
- irrelevant information**
  - defined, 187
  - Obscure Test, 192–194
- Isolate the SUT, 43–44**
- iterative development, 802**

## J

- Java**
  - language-specific xUnit terminology, xl
  - test code packaging, 165

## JBehave

- defined, 748
- tests as examples, 33

## JFCUnit, 755

## JMock

- Configuration Interface, 560
- defined, 755
- Test Double implementation, 140

## Johnson, Rod, 670

## JUnit

- defined, 748
- Expected Exception Test expression, 351
- fixture design, 59
- language-specific terminology, xl
- Suite Fixture Setup support, 442–443
- Test Automation Framework, 300
- test automation tools, 55
- Testcase Object exception, 384–385
- testing stored procedures, 657

## K

### Keep Test Logic Out of Production Code

- minimizing risk, 24
- principle, 45
- test code organization, 164–165

### Keep Tests Independent

- running, 26
- test automation principles, 42–43
- using Fake Object. *See* Fake Object

## Kerievsky, Joshua, xxxix

## keys, Literal Values as, 714

## King, Joseph, 319–321

**L****languages**

- terminology, xl–xli
- variations in Built-in Assertions, 110–111
- xUnit implementations, 76

**language-specific xUnit terminology, xl–xli****“Law of Raspberry Jam”, xxv****Layer Test**

- Business Layer Tests, 344–345
- database testing, 169–171
- implementation, 340–341
- motivating example, 341–342
- overview, 337–338
- Presentation Layer Tests, 343
- refactoring, 342
- Subcutaneous Tests, 343–344
- when to use, 338–340

**layer-crossing tests**

- defined, 802
- testability, 67–69

**Layered Architecture**

- design-for-testability, 7
- layer-crossing tests, 67–69

**Lazy Initialization, 435****Lazy Setup**

- Decorated, 449–450
- examples, 439–440
- implementation, 436–437
- Interacting Tests solution, 231
- motivating example, 437–438
- overview, 435
- vs. Prebuilt Fixtures, 431–432
- refactoring, 439
- Shared Fixture, 64, 105
- when to use, 436

**Lazy Teardown**

- example, 665–666
- implementation, 663–664

**leakage, resource**

- Erratic Tests, 233
- persistent fixtures, 99

**learning styles, xxxix–xl****legacy software**

- Buggy Tests, 261–262
- defined, 802
- tests as safety net, 24

**lenient Mock Object**

- defined, 138
- when to use, 545

**lightweight implementation using****Fake Object. *See* Fake Object****Literal Value**

- Hard-Coded Test Data, 195
- pattern description, 714–717

**local variables**

- converting in Implicit Setup, 427
- defined, 802
- Fresh Fixtures, 313

**Lonely Test**

- caused by Chained Test. *See* Chained Test
- Erratic Tests, 232
- Interacting Tests. *See* Interacting Tests

**Long Tests. *See* Obscure Test****Loopback. *See* Self Shunt****Loop-Driven Test**

- implementation, 610
- Parameterized Test, 614–615

**loops**

- as Conditional Test Logic, 201
- eliminating, 121
- Production Logic in Test cause, 204–205

**Lost Tests**

- avoiding, 597
- Production Bugs cause, 269–271

## M

**Mackinnon, Tim**, 149

**macros, Assertion Methods as**, 364

**maintenance**

High Test Maintenance Cost. *See*

High Test Maintenance Cost

optimizing, 180–181

test automation goals, 27–29

**Manual Event Injection**, 251–252

**Manual Fixture Setup**, 250–251

**Manual Intervention**

impact, 250

introduction, 15

Manual Event Injection,

251–252

Manual Fixture Setup, 250–251

Manual Result Verification, 251

symptoms, 250

**Manual Result Verification**, 251

**manual testing**

defined, 802

right-sizing Test Methods, 154

**Marrick, Brian**

purpose of tests, 51

right-sizing Test Methods, 155

tests as examples, 33

**Maslow**, 176

**MbUnit**

defined, 749

Parameterized Test

implementation, 608–609

Tabular Test with framework

support, 614

**Message, Assertion.** *See* Assertion

Message

**messages, failure.** *See* failure

messages

**meta objects**

Data-Driven Tests, 290

defined, 803

**metatests**, 803

**method attributes**

defined, 803

Expected Exception Tests, 354

Test Discovery using, 397

Test Method Selection

using, 405

**method names**

language-specific xUnit

terminology, xl–xli

Test Method Discovery, 395–396

**methods**

diagramming notation, xlii

instance. *See* instance methods

setUp. *See* setUp method

static, 809

suite, 399

tearDown. *See* tearDown method

Template Method, 164

test commands, 82

verification. *See* result

verification

**Miller, Jeremy**, 687

**Minimal Fixture**

external result verification, 112

General Fixtures solution, 192

minimizing data, 738–739

misuse of setUp method, 93

pattern description, 302–304

strategy, 62–63

test automation philosophies, 36

**Minimize Test Overlap**, 44

**Minimize Untestable Code**, 44–45

**Missing Assertion Message**, 226–227

**Missing Unit Test**

Defect Localization, 23

Production Bugs, 271

**mixins**

defined, 803

Test Helper Mixins, 639,

641–642

**Mock Object**

- Configurable. *See* Configurable Test Double
- configuring, 141–142
- defined, 133
- examples, 548–550
- Expected Behavior Specification, 470–471
- implementation, 546–548
- motivating example, 548
- Overspecified Software
  - cause, 246
- overview, 544–545
- refactoring, 548
- Test Double patterns, 525
- Test Doubles, 137–139
- unit testing, 6
- vs. Use the Front Door First, 40
- verifying indirect output, 131–133
- when to use, 545
- xUnit terminology, 741–744

**MockMaker, 560****modules, 803–804****Move Method, 413****MSTest, 749****Mugridge, Rick, xxiv****multimodal tests, 687****multiple-condition tests**

- Conditional Test Logic, 207–208
- defined, 45–47

**Multiresource In-line Teardown, 513–514****MySQL, 651****Mystery Guest**

- defined, 187
- Obscure Test cause, 188–190

**N****Naive In-line Teardown**

- defined, 511
- example, 512
- of persistent fixtures, 97

**Naive xUnit Test Interpreter, 292–293****Named State Reaching Method, 417–418****Named Test Suite**

- examples, 594–598
- implementation, 594
- introduction, 160–161
- overview, 592–593
- refactoring, 594
- Test Enumeration, 400
- when to use, 593–594

**names**

- Dependency Lookup, 693–694
- intent-revealing. *See* Intent-Revealing Name
- referring to patterns and smells, xxxviii
- Scripted Test, 287
- Suite Fixture Setup, 446

**naming conventions**

- assertion-identifying messages, 371
- making resources unique, 737–738
- patterns, 576–578
- vs. test code organization, 158–159
- Test Method Discovery, 395–396
- Testcase Class per Class, 618
- Testcase Class per Feature, 626
- Testcase Class per Fixture, 632
- For Tests Only solution, 220

**need-driven development**

- Behavior Verification, 469
- defined, 804
- testing with doubles, 149
- using Mock Objects, 545

**Neverfail Test, 274****New River Gorge bridge, xxvi****Newkirk, James, 384–385****NMock, 756****No Test Risk, 24–25****Nondeterministic Test**

- dangers of, 26–27
- Erratic Test, 237–238
- Generated Values cause, 723–724

**notation, diagramming, xlii****Null Object vs. Dummy Object, 730****null values in Dummy Objects, 729–732****NUnit**

- defined, 749
- Expected Exception Test expression, 351
- fixture design, 59
- Interacting Test Suites, 232
- Suite Fixture Setup support, 442–443
- Test Automation Frameworks, 300
- test automation ways and means, 55
- test fixtures, 814
- Testcase Classes, 376
- Testcase Object exception, 384–385

**O****Object Attribute Equality Assertion, 476****Object Factory**

- Dependency Lookup, 688
- installing Test Double, 145

**Object Mother**

- in Delegated Setup, 90–91
- when to use, 644–645

**object technology, xxxix–xl****Object Transaction Rollback****Teardown, 673–674****object-oriented programming****language (OOPL), 76****object-relational mapping (ORM).**

- See* ORM (object-relational mapping)

**objects**

- Creation Method. *See* Creation Method
- determining necessary, 303–304
- diagramming notation, xlii
- fake. *See* Fake Object
- Test Suite Objects. *See* Test Suite Object
- Testcase. *See* Testcase Object

**Obscure Test**

- avoiding with Custom Assertion, 475
- avoiding with Separation of Concerns, 28–29
- Buggy Test, 261
- causes, 186–187
- vs. Communicate Intent, 41
- customer testing, 5
- database testing, 169
- Eager Test, 187–188
- General Fixture, 190–192
- Hard-Coded Test Data, 194–196
- High Test Maintenance Cost, 266
- impact, 186
- Indirect Testing, 196–199
- introduction, xlvi, 12–13, 16
- Irrelevant Information, 192–194
- Mystery Guests, 188–190

- optimizing test execution/
      - maintenance, 180
      - smells, 10
      - solution patterns, 199
      - symptoms, 186
  - observation points**
    - defined, 804
    - test automation strategy, 66–67
  - O’Grady, Ted, 319–321**
  - One Bad Attribute**
    - example, 721–722
    - introduction, xxiii, 90
    - Minimal Fixtures, 304
    - when to use, 719
  - OOPL (object-oriented programming language), 76**
  - optimism, resource, 189, 233–234**
  - order of tests, 456**
  - organization, test. *See* test organization; test organization patterns**
  - ORM (object-relational mapping)**
    - defined, 804
    - Table Truncation Teardown, 663
    - Table Truncation Teardown using, 667
    - Transaction Rollback Teardown, 671
  - Outcome Assertions, Stated. *See* Stated Outcome Assertion**
  - outcome verification patterns. *See* result verification patterns**
  - outcome-describing Verification Method, 117**
  - outgoing interface, 804–805**
  - out-of-order calls, 138**
  - output, indirect. *See* indirect output**
  - outside-in development**
    - Behavior Verification, 469
    - vs. inside-out development, 34–36
  - Overcoupled Software, 40**
  - overlapping tests**
    - minimizing, 44
    - Too Many Tests, 256–257
  - Overspecified Software**
    - avoiding with Fake Objects, 552
    - Fragile Tests, 246
    - testing with doubles, 150
    - Use the Front Door First, 40
- P**
- Parameter Injection**
    - example, 683
    - implementation, 680
    - installing Test Doubles, 144
  - Parameterized Anonymous Creation Method, 417**
  - Parameterized Creation Method**
    - defined, 417
    - Delegated Setup, 90
    - example, xxiii, 420–421
    - Irrelevant Information solution, 193
  - Parameterized Setup Decorator**
    - defined, 449
    - example, 452–453
  - Parameterized Test**
    - example, 611–612
    - extracting. *See* Data-Driven Test
    - further reading, 615–616
    - implementation, 608–610
    - Incremental Tabular Test, 613–614
    - Independent Tabular Test, 612–613
    - Loop-Driven Tests, 614–615
    - motivating example, 610–611
    - overview, 607–608
    - reducing Test Code Duplication, 118–119
    - refactoring, 611

- Tabular Test with framework support, 614
- Test Utility Method, 602
- when to use, 608
- parameters, arguments as, 729**
- “Pass-Fail-Fail”, 234–235**
- pattern language**
  - defined, xxxv–xxxvi, 805
  - pattern naming, 577
- Pattern Languages of Programming (PLoP), 576**
- patterns**
  - aliases and variations, 767–784
  - database. *See* database patterns
  - defined, 805
  - design-for-testability. *See* design-for-testability patterns
  - design-for-testability patterns
  - fixture setup. *See* fixture setup patterns
  - result verification. *See* result verification patterns
  - test automation introduction, xxxiv–xxxviii
  - Test Double. *See* Test Double
  - test organization. *See* test organization patterns
  - test strategy. *See* test strategy patterns
  - testability, 67–71
  - value. *See* value patterns
  - xUnit basics. *See* xUnit basics patterns
- peeling the onion, 11**
- per-functionality test, 50–52**
- Perrotta, Paolo, 537**
- Per-Run Fixtures, 323**
- persistence layer, 339–340**
- persistence resources, 504**
- persistent fixtures, 95–106**
  - database testing, 168–169
  - issues caused by, 96
  - managing, 103–105
  - overview, 95–96
  - Slow Tests cause, 102
  - Table Truncation Teardown. *See* Table Truncation Teardown
  - teardown avoidance, 100–101
  - tearing down, 97–100
  - test strategy patterns, 313–314
  - what’s next, 106
- Persistent Fresh Fixture**
  - building, 88
  - defined, 60–61
  - strategies, 62–63
- Personal Oracle, 651**
- philosophy, test automation. *See* test automation philosophies**
- PHPUnit, 749**
- PLoP (Pattern Languages of Programming), 576**
- Pluggable Behavior**
  - in Named Test Suites, 597
  - Testcase Object
  - implementation, 383
- pollution**
  - Equality Pollution, 221–222
  - Shared Fixture, 326
- polymorphism, 805**
- Poor Man’s Humble Executable, 703**
- Poor Man’s Humble Object**
  - implementation, 699
  - Transaction Rollback
  - Teardown, 671
- Poppendieck, Mary, 51**
- Pragmatic Unit Testing*, 743**
- Prebuilt Fixture**
  - examples, 432–434
  - implementation, 430–431
  - motivating example, 431–432
  - overview, 429–430

- refactoring, 432
- Shared Fixture strategies, 64
- Shared Fixtures, 104–105
- Unrepeatable Tests cause, 235
- presentation layer**
  - defined, 805
  - Layer Tests example, 343
  - testing, 338–339
- presentation logic, 805**
- Preserve Whole Object refactoring, xlviii–xlix**
- principles**
  - list of, 757–759
  - patterns vs., xxxv–xxxvi
  - test automation. *See* test automation principles
- Private Fixture. *See* Fresh Fixture**
- private methods, 586**
- problem statements, xxxvi–xxxvii**
- Procedural Behavior Verification**
  - defined, 470
  - example, 472–473
  - indirect outputs, 131
  - introduction, 112–113
  - Test Spy usage, 137
- Procedural State Verification**
  - defined, 463–464
  - example, 466
  - introduction, 109
- Procedural Test Stub**
  - defined, 526
  - introduction, 135–136
  - when to use, 531
- Procedure Test, Stored. *See* Stored Procedure Test**
- procedure variables, 805–806**
- production, 806**
- Production Bugs**
  - Infrequently Run Tests, 268–269
  - introduction, 12–13
  - Lost Tests, 269–271
  - Missing Unit Tests, 271
  - Neverfail Tests, 274
  - overview, 268
  - reducing risk, 181
  - Untested Code, 271–272
  - Untested Requirements, 272–274
- production code**
  - defined, 806
  - keeping test logic out of, 45
- Production Logic in Test, 204–205**
- profiling tools, 254**
- Programmatic Test. *See* Scripted Test**
- programmer tests, 806**
- project smells, 259–274**
  - Buggy Tests, 260–262
  - defined, 806
  - Developers Not Writing Tests, 263–264
  - High Test Maintenance Cost, 265–267
  - overview, 12–13
  - Production Bugs. *See* Production Bugs
- property tests, 52**
- Pseudo-Object**
  - Hard-Coded Test Double implementation, 570–571
  - Inner Test Double Subclassed from Pseudo-Class, 574–575, 578
  - testing with doubles, 140–141
- pull system, 806–807**
- Pull-Up Method refactoring**
  - Delegated Setup, 413
  - moving reusable test logic, 123
  - Testcase Superclass, 640
- Pushdown Decorator, 450**
- PyUnit**
  - defined, 749
  - Test Automation Framework, 300

## Q

QA (quality assurance), 22–23

QaRun, 244

QTP (QuickTest Professional)

Data-Driven Tests, 290

defined, 756

record and playback tools, 282

Test Automation

Framework, 301

quality assurance (QA), 22–23

QuickTest Professional (QTP).

*See* QTP (QuickTest Professional)

## R

random values

Nondeterministic Tests, 238

Random Generated Values, 724

Record and Playback Test, 13

record and playback tools

introduction, xxxi

Recorded Tests, 282–283

xUnit sweet spot, 58

Recorded Test

built-in test recording,  
281–282

commercial record and  
playback tool, 282–283

customer testing, 5

Data-Driven Tests and, 289

implementation, 280–281

Interface Sensitivity, 241

overview, 278–279

refactored commercial recorded  
tests, 283–284

vs. Scripted Tests, 286

smells, 10

tools, 56

tools for automating, 53–54

when to use, 279–280

Recording Test Stub. *See* Test Spy

red bar, 807

Refactored Recorded Tests

commercial, 283–284

overview, 280

refactoring. *See also* test refactorings

Assertion Message, 372

Assertion Method, 368

Automated Teardown,  
506–507

Back Door Manipulation, 333

Chained Test, 458

Configurable Test Double, 463

Creation Method, 420

Custom Assertion, 480

Database Sandbox, 653

Data-Driven Test, 294

defined, 807

Delegated Setup, 413

Delta Assertion, 488

Dependency Injection, 682

Dependency Lookup, 690–691

Derived Value, 720

Dummy Object, 731

Fake Object, 555–556

Fresh Fixture, 315–316

Garbage-Collected  
Teardown, 502

Generated Value, 725

Guard Assertion, 492

Hard-Coded Test Double, 572

Humble Object, 702

Implicit Setup, 427

Implicit Teardown, 518–519

In-line Setup, 410

In-line Teardown, 512

Layer Test, 342

Lazy Setup, 439

Literal Value, 716

Mock Object, 548

Named Test Suite, 594

Parameterized Test, 611

- Prebuilt Fixture, 432
- Setup Decorator, 451
- Shared Fixture, 324
- Standard Fixture, 309–310
- State Verification, 465–466
- Stored Procedure Test, 658
- Suite Fixture Setup, 444
- Table Truncation Teardown, 664–665
- Test Discovery, 395
- Test Helper, 646
- Test Spy, 541–542
- Test Stub, 533
- Test Utility Method, 605
- Testcase Class per Feature, 627–628
- Testcase Class per Fixture, 634–635
- Testcase Superclass, 640
- Test-Specific Subclass, 584
- Transaction Rollback Teardown, 672
- Unfinished Test Assertion, 496
- Refactoring: Improving the Design of Existing Code* (Fowler), 9, 16
- references, 819–832
- reflection
  - defined, 807
  - Test Discovery, 393
  - Testcase Object implementation, 383
- Registry
  - configurable, 691–692
  - in Dependency Lookup, 688–689
  - Interacting Tests, 230
  - Test Fixture, 644
- regression tests
  - defined, 807
  - Recorded Tests. *See* Recorded Test
  - Scripted Tests, 285–287
- Related Generated Values
  - example, 726–727
  - implementation, 725
- Remoted Stored Procedure Test
  - example, 659–660
  - implementation, 656–658
  - introduction, 172
- Repeatable Test
  - defined, 26–27
  - indirect inputs control, 179
- Replace Dependency with Test
  - Double refactoring
    - Behavior Verification, 472
    - defined, 739
- Repository
  - Data-Driven Test files, 290
  - persistent objects, 90
  - source code, 24, 79, 234, 561, 656
  - test code, 164, 561
- Requirement, Untested.
  - See* Untested Requirement
- ReSharper, 756
- Resource Leakage
  - Erratic Tests, 233
  - persistent fixtures, 99
- Resource Optimism, 189, 233–234
- resources
  - external, 740
  - in-line, 736–737
  - unique, 737–738
- Responder
  - defined, 524
  - examples, 533–535
  - indirect input control, 179
  - introduction, 135
  - when to use, 530
- response time tests, 52
- result verification, 107–123
  - Behavior Verification, 112–114
  - Conditional Test Logic avoidance, 119–121

- Data Sensitivity, 243–245
    - defined, 807
  - Four-Phase Test, 358–361
  - Mock Object, 547–548
  - other techniques, 121–122
  - reducing Test Code Duplication, 114–119
  - reusable test logic, 123
  - Self-Checking Tests, 107–108
  - State Verification, 109–112
  - result verification patterns, 461–497**
    - Behavior Verification. *See* Behavior Verification
    - Behavior Verification
    - Custom Assertion. *See* Custom Assertion
    - Custom Assertion
    - Delta Assertion, 485–489
    - Guard Assertion, 490–493
    - State Verification. *See* State Verification
    - State Verification
    - Unfinished Test Assertion, 494–497
  - results, test**
    - defined, 815
    - introduction, 79–80
  - Retrieval Interface, 137, 540**
  - retrospective, 807–808**
  - reusable test logic**
    - Creation Method, 418–419
    - fixture setup patterns, 422–423
    - organization, 162–164
    - result verification, 123
    - Test Code Duplication, 214–215
    - Test Utility Method. *See* Test Utility Method
    - Utility Method
  - Reuse Tests for Fixture Setup, 90**
  - Robot User Test. *See* Recorded Test**
  - robot user tools**
    - defined, 55–56
    - introduction, xxxi
    - Test Automation Framework, 299
  - Robust Tests**
    - defined, 29
    - indirect inputs control, 179
  - role-describing arguments, 725**
  - root cause analysis**
    - defined, 808
    - smells, 11
  - round-trip tests**
    - defined, 808
    - introduction, 67–69
    - Layer Tests, 340–341
  - row tests. *See* Tabular Test**
  - RSpec**
    - defined, 750
    - fixture design, 59
    - tests as examples, 33
  - runit**
    - defined, 750
    - Test Automation Frameworks, 300
  - running tests**
    - introduction, 79
    - structure, 81
    - test automation goals, 25–27
  - runtime reflection, 393**
- ## S
- Saboteur**
    - defined, 135
    - example, 535–536
    - inside-out development, 35
    - Test Double patterns, 524
    - when to use, 530
  - Safety Net**
    - Buggy Tests, 260
    - tests as, 24
  - sample code, xli–xlii**
  - screen scraping, 241**
  - Scripted Test**
    - Communicate Intent, 41
    - customer testing, 5

- Data-Driven Tests and, 289
  - introduction, 75
  - pattern description, 285–287
  - vs. Recorded Tests, 279
  - smells, 10
  - UI, 55
  - Verify One Condition per Test, 46
- Self Shunt**
- Behavior Verifications, 113
  - example, 573
  - Hard-Coded Test Double implementation, 570
  - pattern naming, 576
  - Test Spy implementation, 540–541
- Self-Call, 582**
- Self-Checking Test**
- Assertion Method usage, 362
  - Conditional Test Logic solution, 201
  - defined, 80
  - happy path code, 178
  - introduction, 107–108
  - running, 26
- Self-Describing Value**
- example, 717
  - Literal Value patterns, 715
- self-testing code, xxi**
- self-tests, built-in**
- defined, 788
  - test file organization, 164
- Sensitive Equality**
- Fragile Tests, 246
  - test-first development, 32
- sensitivities**
- automated unit testing, xxxi–xxxii
  - behavior. *See* Behavior Sensitivity
  - Buggy Tests cause, 260
  - context. *See* Context Sensitivity
  - data. *See* Data Sensitivity
  - interface. *See* Interface Sensitivity
- Separation of Concerns, 28–29**
- Service Facade, 71–72**
- service layers**
- fake, 553
  - tests, 7, 339
- Service Locator**
- in Dependency Lookup. *See* Dependency Lookup
  - installing Test Doubles, 145
- service objects, 808**
- Setter Injection**
- Configuration Interface using, 564
  - example, 684–685
  - implementation, 681
  - installing Test Doubles, 143
- setters, 808**
- setup, fixtures. *See* fixture setup**
- Setup Decorator**
- examples, 451–453
  - implementation, 448–450
  - Implicit Setup, 426
  - motivating example, 450–451
  - overview, 447–448
  - refactoring, 451
  - Shared Fixture strategies, 64, 104–105
  - when to use, 448
- setUp method**
- Implicit Setup, 91–92, 424–428
  - misuse of, 92–93
  - pattern naming, 577
  - Setup Decorator. *See* Setup Decorator
  - Suite Fixture Setup. *See* Suite Fixture Setup
- shadows, diagramming notation, xlii**
- Shank, Clint, 457–458, 613, 616**

**Shared Fixture.** *See also* **Standard Fixture**

- Behavior Verification, 108
- Chained Test. *See* Chained Test
- customer testing, 5
- Data Sensitivity cause, 243
- database testing, 169
- defined, 60–61
- Delta Assertions, 111
- example, 324–325
- Immutable. *See* Immutable
  - Shared Fixture
- Immutable Shared Fixtures, 326
- implementation, 322–323
- incremental tests, 322
- Interacting Tests cause, 229–231
- introduction, 15, 63–65
- Lazy Setup. *See* Lazy Setup
- managing, 103–105
- motivating example, 323–324
- in Nondeterministic Tests, 27
- overview, 317
- Prebuilt Fixture. *See* Prebuilt
  - Fixture
- refactoring, 324
- Setup Decorator. *See* Setup
  - Decorator
- Slow Tests cause, 318–321
- Suite Fixture Setup. *See* Suite
  - Fixture Setup
- Table Truncation Teardown.
  - See* Table Truncation Teardown
- Test Run Wars cause, 236
- Unrepeatable Tests cause, 235
- using Finder Methods, 600–601
- when to use, 318

**Shared Fixture Guard Assertion,**  
492–493

**Shared Fixture State Assertion,** 491

**Simple Success Test**

- example, 352–353
- happy path code, 177

- introduction, 77
- pattern description, 349–350

**The simplest thing that could possibly work (STTCPW),** 810

**Single Glance Readable.**

- See* Communicate Intent

**Single Layer Test.** *See* Layer Test

**Single Test Suite**

- example, 596–597
- Lost Tests solution, 270
- when to use, 593–594

**single tests,** 161–162

**Single-Condition Test**

- Eager Tests solution, 225–226
- Obscure Tests solution, 188
- principles. *See* Verify One
  - Condition per Test
  - unit testing, 6

**Single-Outcome Assertion**

- Assertion Method, 366–367
- defined, 365
- example, 369

**Singleton**

- in Dependency Lookup, 688–689
- Interacting Tests, 230
- retrofitting testability, 146–147

**Singleton, Substituted**

- example, 586–587
- when to use, 581

**skeletons,** 744

**Slow Component Usage,** 254

**Slow Tests**

- Asynchronous Tests, 255–256
- avoiding with Shared Fixture, 318–321
- database testing, 168
- design for testability, 7
- due to Transaction Rollback
  - Teardown, 669
- General Fixtures, 255
- impact, 253

- introduction, 15
- optimizing execution, 180
- persistent fixtures, 102
- preventing with Fake Object.
  - See* Fake Object
- preventing with Test
  - Double, 523
- Slow Component Usage, 254
- symptoms, 253
- Too Many Tests, 256–257
- troubleshooting, 253–254
- smells, test.** *See* test smells
- Smith, Shaun, 39**
- Smoke Test**
  - development process, 4
  - suites, 597–598
  - Test Discovery, 394
- sniff test**
  - defined, xxxviii
  - test smells, 10
- solution patterns, behavior smells**
  - Asynchronous Tests, 256
  - Behavior Sensitivity, 242–243
  - Context Sensitivity, 246
  - Data Sensitivity, 243–245
  - Eager Tests, 225–226
  - Frequent Debugging, 249
  - General Fixture, 255
  - Interacting Test Suites, 232
  - Interacting Tests, 231
  - Interface Sensitivity, 241–242
  - Manual Intervention, 250–252
  - Missing Assertion Messages, 226–227
  - Resource Leakage, 233
  - Resource Optimism, 234
  - Slow Component Usage, 254
  - Test Run War, 236–237
  - Too Many Tests, 257
  - Unrepeatable Tests, 235
- solution patterns, code smells**
  - Asynchronous Code, 211
  - Conditional Verification Logic, 203–204
  - Cut and Paste code reuse, 215
  - Eager Test, 188
  - Equality Pollution, 222
  - Flexible Test, 203
  - General Fixture, 192
  - Hard-Coded Test Data, 196
  - Hard-To-Test Code, 209
  - Highly Coupled Code, 210
  - Indirect Testing, 197–199
  - Irrelevant Information, 193
  - Multiple Test Conditions, 207–208
  - Mystery Guests, 190
  - Obscure Tests, 199
  - Production Logic in Test, 205
  - Test Code Duplication, 115–216
  - Test Dependency in
    - Production, 221
  - Test Hook, 219
  - For Tests Only, 220
  - Untestable Test Code, 212
- solution patterns, project smells**
  - Buggy Test, 261–262
  - Infrequently Run Test, 269
  - Lost Test, 270–271
  - Missing Unit Test, 271
  - Neverfail Test, 274
  - Untested Code, 272
  - Untested Requirements, 274
- Special-Purpose Suite, 595–596**
- specification**
  - Expected Behavior, 470–471
  - Expected Behavior
    - example, 473
  - Expected Object example, 466
  - Expected State, 464–465
  - tests as, xxxiii, 22

- spikes, 809
- Spy, Test. *See* Test Spy
- SQL, Table Truncation Teardown using, 666–667
- Standard Fixture
  - implementation, 307–308
  - motivating example, 308
  - overview, 305–306
  - refactoring, 309–310
  - when to use, 306–307
- standard test interface, 378
- starbursts, diagramming notation, xlii
- state, initializing via
  - Back Door Manipulation. *See* Back Door Manipulation
  - Named State Reaching Method, 417–418
- State Verification
  - vs. behavior, 36
  - examples, 466–467
  - implementation, 463–465
  - indirect outputs, 179–180
  - introduction, 109–112
  - motivating example, 465
  - overview, 462–463
  - refactoring, 465–466
  - Self-Checking Tests, 108
  - Use the Front Door First, 41
  - when to use, 463
- Stated Outcome Assertion
  - Assertion Methods, 366
  - defined, 365
  - example, 369
  - Guard Assertions as, 491
  - introduction, 110–111
- State-Exposing Subclass
  - Test-Specific Subclass, 289–590
  - when to use, 580
- stateless, 809
- statements, “if”. *See* “if” statements
- static binding
  - defined, 809
  - Dependency Injection, 678–679
- static methods, 809
- static variables, 809
- Statically Generated Test
  - Doubles, 561
- STDD (storytest-driven development), 4, 810
- stop on first failure
  - Naive xUnit Test Interpreter, 292–293
  - xUnit introduction, 57
- Stored Procedure Test
  - database testing, 172
  - examples, 658–660
  - implementation, 655–658
  - motivating example, 658
  - overview, 654
  - refactoring, 658
  - when to use, 654–655
- storytest, 810
- storytest-driven development (STDD), 4, 810
- strategies, test automation. *See* test automation strategies
- stress tests, cross-functionality, 52
- strict Mock Object
  - defined, 138
  - when to use, 545
- STTCPW (The simplest thing that could possibly work), 810
- Stub, Test. *See* Test Stub
- Subclass, Test-Specific. *See* Test-Specific Subclass
- Subclassed Humble Object, 700
- Subclassed Inner Test Double, 573–574
- Subclassed Singleton, 7
- Subclassed Test Double, 146–147

**Subcutaneous Test**

- customer testing, 5
- database testing, 174
- design for testability, 7
- Layer Tests, 343–344

**Subset Suite**

- example, 594–598
- implementation, 594
- introduction, 160–161
- overview, 592
- Too Many Tests solution, 257
- when to use, 593

**substitutable dependencies**

- defined, 810
- Dependency Initialization Test, 352
- using Test Spy, 540

**Substitutable Singleton**

- in Dependency Lookup, 689
- example, 586–587, 692–693
- retrofitting testability, 146–147
- when to use, 581

**substitution mechanisms, 688–689****Suite Fixture Setup**

- example, 444–446
- implementation, 442–443
- implicit, 426
- motivating example, 443–444
- overview, 441–442
- refactoring, 444
- Shared Fixture strategies, 64
- Shared Fixtures, 104–105
- when to use, 442

**suite method, 399****suites**

- Named Test Suite. *See* Named Test Suite
- test organization, 160–162
- Test Suite Object. *See* Test Suite Object

**Suites of Suites**

- building with Test enumeration, 400
- defined, 388
- example, 389–391
- Interacting Test Suites, 231–232
- introduction, 7, 15, 78

**SUnit**

- defined, 750
- Test Automation Frameworks, 300

**Superclass, Testcase. *See* Testcase****Superclass****SUT (system under test)**

- control points and observation points, 66–67
- dangers of modifying, 41–42
- defined, 810–811
- Four-Phase Test, 358–361
- interface sensitivity, xxxii
- isolation principle, 43–44
- minimizing risk, 24–25
- preface, xxii–xxiii
- replacing in Parameterized Test, 609
- result verification. *See* result verification
- state vs. behavior verification, 36
- terminology, xl–xli
- test automation tools, 53–54
- Test Hook in, 711–712
- understanding with test automation, 23

**SUT API Encapsulation**

- Chained Tests as, 455
- Indirect Testing solution, 198
- Interface Sensitivity solution, 241

**SUT Encapsulation Method, 601–602**

**Symbolic Constants**

- example, 716
- Literal Value, 715

**symptoms, behavior smells**

- Assertion Roulette, 224
- Asynchronous Tests, 255
- Behavior Sensitivity, 242
- Context Sensitivity, 245
- Data Sensitivity, 243
- Eager Tests, 224–225
- Erratic Tests, 228
- Fragile Tests, 239
- Frequent Debugging, 248
- General Fixtures, 255
- Interacting Test Suites, 231
- Interacting Tests, 229
- Interface Sensitivity, 241
- Manual Intervention, 250–252
- Missing Assertion Messages, 226
- Nondeterministic Tests, 237
- Resource Leakage, 233
- Resource Optimism, 233
- Slow Tests, 253
- Test Run Wars, 236
- Too Many Tests, 256
- Unrepeatable Tests, 234–235

**symptoms, code smells**

- Asynchronous Code, 210
- Complex Teardown, 206
- Conditional Test Logic, 200
- Eager Tests, 187–188
- Equality Pollution, 221
- Flexible Tests, 202
- General Fixtures, 190–191
- Hard-Coded Test Data, 194–195
- Hard-To-Test Code, 209
- Highly Coupled Code, 210
- Indirect Testing, 196–197
- Irrelevant Information, 192–193
- Multiple Test Conditions, 207

Mystery Guests, 188–189

Obscure Tests, 186

Production Logic in Test,  
204–205

Test Code Duplication, 213–214

Test Dependency in  
Production, 220

Test Logic in Production, 217

test smells, 10

For Tests Only, 219

Untestable Test Code, 211

**symptoms, project smells**

Buggy Tests, 260

Developers Not Writing Tests,  
263

High Test Maintenance  
Cost, 265

Infrequently Run Tests, 268–269

Lost Tests, 269

Missing Unit Tests, 271

Neverfail Tests, 274

Production Bugs, 268

Untested Code, 271–272

Untested Requirements, 272–273

**symptoms, test smells, 10****synchronous tests**

avoiding with Humble Object,  
696–697

defined, 810

**system under test (SUT).** *See* SUT  
(system under test)

**T****Table Truncation Teardown**

data access layer testing, 173

defined, 100

examples, 665–667

implementation, 662–664

motivating example, 664

overview, 661–662

- refactoring, 664–665
- when to use, 662
- tabular data, 291**
- Tabular Test**
  - Chained Tests, 457–458
  - with framework support, 614
  - implementation, 609–610
  - Incremental, 613–614
  - Independent, 612–613
- tasks, 811**
- TDD (test-driven development)**
  - defined, 813
  - implementing utility methods, 122
  - introduction, xxxiii–xxxiv
  - Missing Unit Tests, 271
  - need-driven development, 149
  - process, 4–5
  - Test Automation
    - Frameworks, 301
  - test automation principles, 40
- teardown, fixture. *See* fixture**
- teardown**
- Teardown Guard Clause**
  - example, 513
  - Implicit Teardown, 517–518
  - In-line Teardown, 511
- tearDown method**
  - Implicit Teardown, 516–519
  - persistent fixtures, 98
  - Setup Decorator. *See* Setup Decorator
- Template Method, 164**
- Temporary Test Stub**
  - when to use, 530–531
  - xUnit terminology, 741–744
- terminology**
  - test automation introduction, xl–xli
  - transient fixtures, 86–88
  - xUnit. *See* xUnit basics
- test automater, 811**
- test automation, xxix–xliii**
  - assumptions, xxxix–xl
  - automated unit testing, xxx–xxxii
  - brief tour, 3–8
  - code samples, xli–xlii
  - developer testing, xxx
  - diagramming notation, xlii
  - feedback, xxix
  - fragile test problem, xxxi–xxxii
  - limitations, xliii
  - overview, xxix
  - patterns, xxxiv–xxxviii
  - refactoring, xxxviii–xxxix
  - terminology, xl–xli
  - testing, xxx
  - uses of, xxxiii–xxxiv
- Test Automation Framework**
  - introduction, 75
  - pattern description, 298–301
- test automation goals, 19–29**
  - ease of running, 25–27
  - improving quality, 22–23
  - list of, 757–759
  - objectives, 21–22
  - reducing risk, 23–25
  - system evolution, 29
  - understanding SUT, 23
  - why test?, 19–21
  - writing and maintaining, 27–29
- Test Automation Manifesto, 39**
- test automation philosophies, 31–37**
  - author’s, 37
  - differences, 32–36
  - importance of, 31–32
- test automation principles, 39–48**
  - Communicate Intent, 41
  - Design for Testability, 40
  - Don’t Modify the SUT, 41–42
  - Ensure Commensurate Effort and Responsibility, 47–48

- Isolate the SUT, 43–44
- Keep Test Logic Out of
  - Production Code, 45
- Keep Tests Independent,
  - 42–43
- Minimize Test Overlap, 44
- Minimize Untestable Code,
  - 44–45
- overview, 39–40
- Test Concerns Separately, 47
- Use the Front Door First,
  - 40–41
- Verify One Condition per Test,
  - 45–47
- Write the Tests First, 40
- test automation roadmap, 175–181**
  - alternative path verification,
    - 178–179
  - difficulties, 175–176
  - direct output verification, 178
  - execution and maintenance
    - optimization, 180–181
  - happy path code, 177–178
  - indirect outputs verification,
    - 178–180
  - maintainability, 176–177
- test automation strategies, 49–73**
  - brief tour, 3–8
  - control points and observation
    - points, 66–67
  - cross-functional tests, 52–53
  - divide and test, 71–72
  - ensuring testability, 65
  - fixture strategies overview, 58–61
  - interaction styles and testability
    - patterns, 67–71
  - overview, 49–50
  - per-functionality tests, 50–52
  - persistent fresh fixtures, 62–63
  - shared fixture strategies, 63–65
  - test-driven testability, 66
    - tools for, 53–58
    - transient fresh fixtures, 61–62
    - what’s next, 73
    - wrong, 264
- Test Bed. *See* Prebuilt Fixture**
- test cases, 811**
- test code, 811**
- Test Code Duplication**
  - causes, 214–215
  - Custom Assertions, 475
  - Delegated Setup, 412
  - High Test Maintenance
    - Cost, 266
  - impact, 214
  - In-Line Setup, 89
  - introduction, 16
  - possible solution, 216
  - reducing, 114–119
    - reducing with Configurable
      - Test Doubles. *See* Configurable
        - Test Double
      - reducing with Parameterized
        - Tests. *See* Parameterized Test
    - reducing with Test Utility
      - Methods. *See* Test Utility
        - Method
  - removing with Testcase Class per
    - Fixture. *See* Testcase Class per
      - Fixture
  - reusing test code, 162
  - symptoms, 213–214
- Test Commands, 82**
- Test Concerns Separately, 47**
- test conditions, 154, 811–812**
- test database, 812**
- test debt, 812**
- Test Dependency in Production, 220–221**
- Test Discovery**
  - introduction, 78
  - Lost Tests solution, 271

- pattern description, 393–398
- Test Suite Object Generator, 293
- Test Suite Objects, 388
- Test Double, 125–151, 521–590**
  - Back Door Manipulation, 332
  - Behavior Verification, 112
  - Configurable Test Double.
    - See* Configurable Test Double
  - configuring, 141–142
  - considerations, 150
  - customer testing, 5
  - database testing, 169–171
  - Dependency Injection.
    - See* Dependency Injection
  - Dependency Lookup, 144–145
  - dependency replacement, 739
  - design for testability, 7
  - Don't Modify the SUT, 41–42
  - Dummy Object, 134–135
  - example, 526–528
  - Fake Object. *See* Fake Object
  - Fragile Test, 240
  - Hard-Coded Test Double.
    - See* Hard-Coded Test Double
  - Highly Coupled Code
    - solution, 210
  - indirect input and output, 125–126
  - indirect input control, 128–129
  - indirect input, importance
    - of, 126
  - indirect output, importance of, 126–127
  - indirect output verification, 130–133
  - installing, 143
  - minimizing risk, 25
  - Mock Object. *See* Mock Object
  - other uses, 148–150
  - outside-in development, 35–36
  - overview, 522–523
  - providing, 140–141
  - retrofitting testability, 146–148
  - reusing test code, 162
  - terminology, 741–744
  - vs. Test Hook, 709–712
  - Test Spy, 137, 538–543
  - Test Stub. *See* Test Stub
  - Test-Specific Subclass.
    - See* Test-Specific Subclass
  - types of, 133–134
  - when to use, 523–526
- Test Double Class**
  - example, 572–573
  - implementation, 569–570
- Test Double Subclass**
  - implementation, 570
  - when to use, 580–581
- test drivers**
  - Assertion Messages, 370
  - defined, 813
- test driving, 813**
- Test Enumeration**
  - introduction, 153
  - pattern description, 399–402
- test errors, 80, 813**
- test failure, 80, 813**
- test first development**
  - defined, 813–814
  - process, 4–5
  - test automation philosophy, 32–33
  - vs. test-last development, xxxiv
- Test Fixture Registry**
  - accessing Shared Fixtures, 104
  - Test Helper use, 644
- test fixtures. *See* fixtures**
- Test Helper**
  - Automated Teardown, 505
  - introduction, xxiii
  - pattern description, 643–647

- Test Helper Mixin**
  - example, 641–642
  - vs. Testcase Superclass, 639
- Test Hook**
  - pattern description, 709–712
  - in Procedural Test Stub, 135–136
  - retrofitting testability, 148
  - Test Logic in Production, 217–219
  - testability, 70
- Test Logic, Conditional.**
  - See Conditional Test Logic*
- Test Logic in Production**
  - Equality Pollution, 221–222
  - impact, 217
  - introduction, 17
  - symptoms, 217
  - Test Dependency in Production, 220–221
  - Test Hooks, 148, 217–219
  - For Tests Only, 219–220
- test maintainer, 815**
- Test Method**
  - calling Assertion. *See Assertion Method*
  - Constructor Test example, 355–357
  - Constructor Tests, 351
  - Dependency Initialization Tests, 352
  - enumeration, 401
  - Expected Exception Test, 350–351
  - Expected Exception Test using block closure, 354–355
  - Expected Exception Test using method attributes, 354
  - Expected Exception Test using try/catch, 353–354
  - fixture design, 59
  - implementation, 349
  - invocation, 402
  - Lost Tests, 269–270
  - minimizing untested code, 44–45
  - organization, 7, 155–158. *See also* test organization patterns
  - overview, 348–349
  - persistent fixtures. *See* persistent fixtures
  - right-sizing, 154–155
  - running, 81
  - selection, 404–405
  - Simple Success Test, 349–350
  - Simple Success Test example, 352–353
  - test automation philosophies, 34
  - Test Commands, 82
  - Test Concerns Separately, 47
  - Test Suite Objects, 82
  - Testcase Object implementation, 384–385
  - transient fixture management. *See* transient fixtures
  - unit testing, 6
  - Verify One Condition per Test, 46–47
  - writing simple tests, 28
- Test Method Discovery**
  - defined, 394–395
  - examples, 395–397
- Test Object Registry. *See* Automated Teardown**
- test organization, 153–165**
  - code reuse, 162–164
  - introduction, 153
  - naming conventions, 158–159
  - overview, 7
  - right-sizing Test Methods, 154–155
  - test files, 164–165

- Test Methods and Testcase
  - Classes, 155–158
- test suites, 160–162
- test organization patterns, 591–647**
  - Named Test Suite. *See* Named Test Suite
  - Parameterized Test.
    - See* Parameterized Test
  - Test Helper, 643–647
  - Test Utility Method. *See* Test Utility Method
  - Testcase Class per Class.
    - See* Testcase Class per Class
  - Testcase Class per Feature.
    - See* Testcase Class per Feature
  - Testcase Class per Fixture.
    - See* Testcase Class per Fixture
  - Testcase Superclass, 638–642
- test packages**
  - defined, 815
  - test file organization, 164–165
- test readers, 815**
- test refactorings. *See also* refactoring**
  - Extractable Test Component, 735–736
  - In-line Resource, 736–737
  - Make Resources Unique, 737–738
  - Minimize Data, 738–739
  - Replace Dependency with Test Double, 739
  - Set Up External Resource, 740
- test results**
  - defined, 815
  - introduction, 79–80
  - verification. *See* result verification
- Test Run War**
  - database testing, 169
  - Erratic Tests cause, 235–237
  - introduction, 15
  - vs. Shared Fixture strategy, 64
- Test Runner**
  - Graphical. *See* Graphical Test Runner
  - implementation, 378–381
  - introduction, 79
  - Missing Assertion Messages, 226–227
  - overview, 377–378
  - Test Automation Frameworks, 300
- test runs, 815**
- Test Selection**
  - pattern description, 403–405
  - Test Suite Object, 388
- test smells, 9–17**
  - aliases and causes, 761–765
  - behavior. *See* behavior smells
  - catalog of, 12–17
  - code smells. *See* code smells
  - database testing. *See* database testing
  - defined, 808, 816
  - introduction, xxxvii
  - overview, 9–11
  - patterns and principles vs., xxxv–xxxvi
  - project smells. *See* project smells
  - reducing Test Code Duplication, 114–119
- Test Spy**
  - Back Door Verification, 333
  - Behavior Verification, 113
  - Configurable. *See* Configurable Test Double
  - examples, 542–543
  - implementation, 540–541
  - indirect outputs verification, 179–180
  - introduction, 131–133, 137, 525
  - motivating example, 541

- overview, 538–539
- Procedural Behavior
  - Verification, 470
- refactoring, 541–542
- when to use, 539–540
- xUnit terminology, 741–744
- test strategy patterns, 277–345**
  - Data-Driven Test. *See* Data-Driven Test
  - Fresh Fixture. *See* Fresh Fixture
  - Layer Test. *See* Layer Test
  - Minimal Fixture, 302–304
  - Recorded Test. *See* Recorded Test
  - Scripted Test, 285–287
  - Shared Fixture. *See* Shared Fixture
  - Standard Fixture. *See* Standard Fixture
  - Test Automation Framework, 298–301
- test strippers, 816**
- Test Stub**
  - Behavior-Modifying Subclass, 584–585
  - Configurable. *See* Configurable Test Double
  - configuring, 141–142
  - Context Sensitivity solution, 246
  - controlling indirect inputs, 129
  - creating in-line resources, 737
  - examples, 533–537
  - implementation, 531–532
  - indirect inputs control, 179
  - inside-out development, 34–35
  - introduction, 133, 135–136, 524
  - motivating example, 532–533
  - overview, 529–530
  - refactoring, 533
  - unit testing, 6
    - when to use, 530–531
    - xUnit terminology, 741–744
- test success, 816**
- Test Suite Enumeration**
  - defined, 400
  - example, 402
- Test Suite Factory, 232**
- Test Suite Object**
  - enumeration, 400
  - Interacting Test Suites, 231–232
  - introduction, 7, 82
  - pattern description, 387–392
- Test Suite Object Generator, 293**
- Test Suite Object Simulator, 293**
- Test Suite Procedure**
  - defined, 388–389
  - example, 391–392
- test suites**
  - defined, 816
  - Lost Tests, 269–270
  - Named Test Suites. *See* Named Test Suite
- Test Tree Explorer, 161–162, 380–381**
- Test Utility Method**
  - Communicate Intent, 41
  - eliminating loops, 121
  - example, 605–606
  - implementation, 602–603
  - introduction, xxiii, 16–17, 23, 162–163
  - motivating example, 603–604
  - Obscure Tests solution, 199
  - overview, 599
  - reducing risk of bugs, 181
  - refactoring, 605
  - reusing, lviii–lix
  - reusing via Test Helper, 643–647
  - reusing via Testcase Superclass, 638–642

- using TDD to write, 122
- when to use, 600–602
- Test Utility Test**, 603
- testability, design for.** *See* **design-for-testability**
- Testcase Class**
  - introduction, 78
  - organization, 7, 155–158
  - pattern description, 373–376
  - reusable test logic, 123
  - selection, 404–405
- Testcase Class Discovery**
  - defined, 394
  - example, 397–398
- Testcase Class per Class**
  - example, 618–623
  - implementation, 618
  - overview, 617
  - when to use, 618
- Testcase Class per Feature**
  - example, 628–630
  - implementation, 626
  - motivating example, 626–627
  - overview, 624
  - refactoring, 627–628
  - when to use, 625
- Testcase Class per Fixture**
  - example, 635–637
  - implementation, 632–633
  - motivating example, 633–634
  - overview, 631
  - refactoring, 634–635
  - Verify One Condition per Test, 46–47
  - when to use, 632
- Testcase Class per Method**, 625
- Testcase Class per User Story**, 625
- Testcase Object**
  - introduction, 81
  - pattern description, 382–386
- Testcase Superclass**
  - pattern description, 638–642
  - reusing test code, 163–164
  - Test Discovery using, 397–398
- test-driven bug fixing**, 812
- test-driven development (TDD).**
  - See* **TDD (test-driven development)**
- Test-Driven Development: By Example** (Beck), 301
- test-driven testability**, 66
- Testing by Layers.** *See* **Layer Test**
- testing terminology.** *See* **terminology**
- test-last development**
  - defined, 815
  - strategy, 65
  - test automation philosophy, 32–33
  - vs. test-first development, xxxiv
- TestNG**
  - defined, 750
  - Interacting Tests, 231
  - Testcase Object exception, 384–385
  - vs. xUnit, 57
- Tests as Documentation**
  - Communicate Intent, 41
  - customer testing, 5
  - defined, 23
  - reusing test code, 162
  - unit testing, 6
- Tests as Safety Net**, 24, 260
- Tests as Specification**, xxxiii, 22
- test-specific equality**, 588–589, 816
- Test-Specific Extension.**
  - See* **Test-Specific Subclass**
- Test-Specific Subclass**
  - Behavior-Exposing Subclass, 587
  - Behavior-Modifying Subclass (Substituted Singleton), 586–587

- Behavior-Modifying Subclass (Test Stub), 584–585
- defining Test-Specific Equality, 588–589
- Don't Modify the SUT, 42
- implementation, 581–582
- Isolate the SUT, 44
- motivating example, 582–584
- overview, 579–580
- refactoring, 584
- retrofitting testability, 146–147
- State-Exposing Subclass, 289–590
- For Tests Only solution, 220
- when to use, 580–581
- Test::Unit**, 750
- Thread-Specific Storage**, 688–689
- Too Many Tests**, 256–257
- tools**
  - automated unit testing, xxx–xxx1
  - commercial record and playback, 282–283
  - QTP. *See* QTP (QuickTest Professional)
  - robot user. *See* robot user tools
  - for test automation strategy, 53–58
  - types of, 753–756
- Transaction Controller, Humble.**  
*See* Humble Transaction Controller
- Transaction Rollback Teardown**
  - data access layer testing, 173
  - defined, 100
  - examples, 673–675
  - implementation, 671
  - motivating example, 672
  - overview, 668–669
  - refactoring, 672
  - when to use, 669–671
- transient fixtures**, 85–94
  - Delegated Setup, 89–91
  - hybrid setup, 93
  - Implicit Setup, 91–93
  - In-Line Setup, 88–89
  - overview, 85–86
  - vs. persistent fixtures, 96
  - tearing down, 93–94
  - terminology, 86–88
  - what's next, 94
- Transient Fresh Fixture**
  - database testing, 170
  - defined, 60–61, 314
  - vs. Shared Fixture, 61–62
- troubleshooting**
  - Buggy Tests, 261
  - Developers Not Writing Tests, 264
  - Erratic Tests, 228–229
  - Fragile Tests, 239–240
  - High Test Maintenance Cost, 267
  - Slow Tests, 253–254
- True Humble Executable**, 703–706
- True Humble Objects**, 699–700
- TRUNCATE command.** *See* Table Truncation Teardown
- try/catch**
  - Expected Exception Tests, 353–354
  - Single-Outcome Assertions, 367
- try/finally block**
  - cleaning up fixture teardown logic, l–liv
  - Implicit Teardown, 519
  - In-line Teardown, 512–513
- type compatibility**, 679
- type visibility**
  - Test Helper use, 644
  - Test Utility Methods, 603
  - Testcase Superclass use, 639

**U****UAT (user acceptance tests)**

- defined, 817
- principles, 42

**UI (User Interface) tests**

- asynchronous tests, 70–71
- Hard-To-Test Code, 71–72
- tools, 55

**UML (Unified Modeling Language), 816****Unconfigurable Test Doubles, 527****unexpected exceptions, 352****Unfinished Test Assertion, 494–497****Unfinished Test Method from Template, 496–497****Unified Modeling Language (UML), 816****unique resources, 737–738*****Unit Testing with Java (Link), 743*****unit tests**

- defined, 817
- introduction, 6
- per-functionality, 51
- rules, 307
- Scripted Tests, 285–287
- xUnit vs. Fit, 290–292

**unnecessary object elimination, 303–304****Unrepeatable Test**

- database testing, 169
- Erratic Test cause, 234–235
- introduction, 15, 64
- persistent fresh fixtures, 96
- vs. Repeatable Test, 26–27

**Untestable Test Code**

- avoiding Conditional Logic, 119–121
- Hard-To-Test Code, 211–212

**Untested Code**

- alternative path verification, 178–179
- indirect inputs and, 126
- Isolate the SUT, 43
- minimizing, 44–45
- preventing with Test Doubles, 523
- Production Bugs, 271–272
- unit testing, 6

**Untested Requirement**

- Frequent Debugging cause, 249
- indirect output testing, 127
- preventing with Test Doubles, 523
- Production Bugs cause, 272–274
- reducing via Isolate the SUT, 43

**usability tests, 53****use cases, 817****Use the Front Door First**

- defined, 40–41
- Overspecified Software avoidance, 246

**user acceptance tests (UAT)**

- defined, 817
- principles, 42

**User Interface (UI) tests**

- asynchronous tests, 70–71
- Hard-To-Test Code, 71–72
- tools, 55

**user story**

- defined, 817
- Testcase Class per, 625

**utility methods. *See* Test Utility Method****utPLSQL, 750**

## V

- value patterns, 713–732**
  - Derived Values, 718–722
  - Dummy Objects, 728–732
  - Generated Values, 723–727
  - Literal Values, 714–717
- variables**
  - in Derived Values, 718–722
  - global, 92, 798
  - instance. *See* instance variables
  - local. *See* local variables
  - procedure variables, 805–806
  - static, 809
- VB Lite Unit, 751**
- VbUnit**
  - defined, 751
  - Suite Fixture Setup support, 442
  - Testcase Class terminology, 376
  - xUnit terminology, 300
- Verbose Tests. *See* Obscure Test**
- verification**
  - alternative path, 178–179
  - Back Door Manipulation, 329–330
  - Back Door using Test Spy, 333
  - cleaning up logic, xlvi–l
  - direct output, 178
  - indirect outputs, 130–133, 178–180
  - state vs. behavior, 36
  - test results. *See* result verification
  - Verify One Condition per Test, 45–47
- Verification Method**
  - defined, 477, 602
  - example, 482–483
- Verify One Condition per Test**
  - defined, 40, 45–47
  - right-sizing Test Methods, 154–155
- verify outcome, 817**
- Virtual Clock, 246**

- visibility**
  - of SUT features from Test-Specific Subclass, 581–582
  - test file organization, 165
  - type. *See* type visibility
- visual objects, Humble Dialog use, 706**
- Visual Studio, 756**

## W

- waterfall design, 65**
- Watir**
  - defined, 756
  - Test Automation Frameworks, 301
  - test automation tools, 53
- Weinberg, Gerry, xxiv–xxv, 61–62**
- widgets**
  - Humble Dialog use, 706
  - recognizers, 299
- Wikipedia, 729**
- Working Effectively with Legacy Code (Feathers), 210***
- Write the Tests First, 40**
- writing tests**
  - Developers Not Writing Tests
    - project smells, 263–264
  - development process, 4–5
  - goals, 27–29
  - philosophies. *See* test automation philosophies
  - principles. *See* test automation principles

## X

- XML data files, Data-Driven Tests, 294–295**
- xUnit**
  - Data-Driven Tests with CSV
    - input file, 296
  - Data-Driven Tests with XML
    - data file, 294–295

- defined, 751
- family members, 747–751
- vs. Fit, 291–292
- fixture definitions, 86
- Interacting Test Suites, 232
- introduction, 56–57
- language-specific terminology, xl–xli
- modern, 55
- Naive xUnit Test Interpreter, 292–293
- profiling tools, 254
- Suite Fixture Setup support, 442–443
- sweet spot, 58
- terminology, 741–746
- Test Automation Frameworks, 300
- test fixtures, 814
- test organization mechanisms, 153
- xUnit basics, 75–83**
  - defining suites of tests, 78–79
  - defining tests, 76–78
  - fixtures, 78
  - overview, 75–76
  - procedural world, 82–83
  - running Test Methods, 81
  - running tests, 79
  - Test Commands, 82
  - test results, 79–80
  - Test Suite Object, 82
- xUnit basics patterns, 347–405**
  - Assertion Message, 370–372
  - Assertion Method.
    - See* Assertion Method
  - Four-Phase Test, 358–361
  - Test Discovery, 393–398
  - Test Enumeration, 399–402
  - Test Method.
    - See* Test Method
  - Test Runner.
    - See* Test Runner
  - Test Selection, 403–405
  - Test Suite Object, 82, 387–392
  - Testcase Class, 373–376
  - Testcase Object, 382–386