

Index

Symbols

- .dll 32
- .NET 21, 25, 56, 69, 100, 151
 - extension methods 129, 269
 - Moq framework 100, 126
 - NMock 100
 - Rhino Mocks 100
 - Typemock 100
- .NET mock object frameworks 129

A

- AAA *See* arrange-act-assert model
- AAA-style stubs 128
- abstract class 259
- abstract test class 159
- abstract test driver class pattern 159
- abstract test infrastructure class pattern 152
- abstractions 263
- acceptance testing 268, 283
- acceptance-style test 251
- accidental buging 9
- Act action 30
- action attributes 34
- action-driven testing 83
- actions *See* test actions
- Adapt Parameter 96
- Addison-Wesley 227, 231
- agent of change 220
- agile 142, 234
- Ajax 279
- analyzer object 188
- annotated method 79
- anonymous delegate 120, 124–125
- anonymous method 124, 249
- anti-patterns 192
 - constrained test order 192
 - external-shared-state corruption 198
 - hidden test call 194
 - shared-state corruption 196
- API 168
 - See also* interface
- API changes 174
- Arrange action 30
- arrange-act-assert model 103, 126
- arrange-act-assert style 269
- arrange-act-assert syntax 269
- art of unit testing 52
- ArtOfUnitTesting 268
- ASP.NET 274, 278
- Assert 31, 35–36, 38, 42–43
- Assert action 30
- Assert class 31
- assert failures 180
- assert messages 209, 212
 - best practices 212
 - repetition 213
- assert utility class 168
- assert utility method 168
- Assert.AreEqual 31
- Assert.AreSame 31
- Assert.IsTrue 31
- Assertion *See* test assertion
- assertion logic 188
- assumptions 205, 209
- attributes
 - Category 39
 - ExpectedException 38
 - Ignore 38
 - SetUp 34–35

- attributes (*continued*)
 - TearDown 34–35
 - Test 30
 - TestFixture 30
 - TestFixtureSetUp 36
 - TestFixtureTearDown 36
- See also* NUnit & unit-testing framework
- Auto Factory *See* Autofac
- Autofac 62, 261, 274–275
- automated build 142–143, 146–147, 169, 225
 - types 144
- automated test 6, 11, 28, 30, 141
- automocking container 274
- B**
- base class 75, 159, 163–165
- Beck, Kent 4
- Beizer, Boris 227
- best practices 172, 230
- blockers 220
- bottom-up change 223
- brittle test 94, 97, 107, 206
 - less brittle test 207
- broken test 38, 175, 182
- build break 142
- build process 142–143, 228, 234
- business logic 8
- C**
- C# 124, 236
- C++ 25, 100, 236, 239, 244, 265
 - mockpp 100
- call chains 96
- callback 120, 132
- calling test 196
- Castle 275
 - See also* Microsoft Castle
- Castle Windsor 62, 261, 274–275
- Category 39
- champions 220, 226
- CHESS 282
- class 5, 8, 148
 - methods 5, 148
- class extensibility 263
- class library 28
- class under test *See* system under test
- classic object-oriented design 81
- classic testing 6
- code adapter classes 252
- code churn 228
- “Code Churn Perspective” article 228
- Code Complete* 228
- code consistency 225
- code coverage 180, 273
- code design 208
- code duplication 254
 - refactoring 254
- code integration 142
- code integrity 224
- code library 24
- code quality 18, 231, 234
- code reuse 151, 156
- code smell 208
- code under test *See* system under test
- code with logic 23, 28
- collaborator 50, 208
- collections 209
- Common Language Runtime 247
- Common Service Locator 274–275
- Communications of the ACM* 235–237
- Complete Guide to Software Testing, The* 7
- components 5, 240–241
 - complex 243
 - easy 243
 - mapping 241
 - priority list 240
 - rating 240
- composite object hierarchy 62
- concrete class 259
- Conditional attribute 79
- conditional compilation 69, 79
 - #if and #endif 80
 - when to use 69
- configuration class 168
- configuration data 241
- Configuration property 95
- ConfigurationManager 152
- ConfigurationManagerTests 152
- conflicting test 176–177
- console application 5, 13
- constructor injection 58–59, 64–65, 260
 - parameters 63
 - when to use 63
- constructors 258
 - static constructors 258
- containers 62
 - Autofac 62
 - Castle Windsor 62

containers (*continued*)
 container object 62
 factory methods 62
 Microsoft Unity 62
 Ninject 62
 Spring.NET 62
 StructureMap 62
 continuous integration 144, 225
 continuous integration build 144
 correctness 4
 CppUnit 25, 236
 CQL 253
 CRUD (create, retrieve, update and delete) 159
 CruiseControl.NET 143
 CSL *See* Common Service Locator
 Curtis, Bill 237
 custom header 159
 cyclomatic complexity 240

D

data access 8
 data helper 8
 data layer 277
 data objects 259
 database testing 268, 276
 data-holder class 241
 DBConfiguration 95
 debug switch 68
 debugging 19, 32, 146, 225
 bug-fixing time 228
 bugs 10, 18–19
 bugs per class 235
 reopening bugs 228
 decision-making code *See* logical code
 decoupled design 132
 defects 228
 delegates 247, 258
 dependencies 11, 40, 52–53, 62–63, 65, 80,
 132, 240–241, 243
 breaking 55
 direct dependency 51, 54, 260
 external dependency 87
 fake dependency 58, 69
 injecting fake dependencies 97
 injection with properties 64
 non-optional 61, 63
 production dependencies 75
 relevant 62
 replacing 73

dependency driven 242
 dependency injection 64, 245, 261, 268
 Dependency Inversion Principle 78
 dependency level 240
 Depender 246, 248
 derived class 71–72, 156, 158–159, 163–164,
 166
Design Patterns 68
 development stages 232
 DI container 275
 DI frameworks 275
 domain logic 61
 dos and don'ts of introducing unit
 testing 219
 driving force 229
 DRY (“don’t repeat yourself”)
 principle 151, 184
 duplicate code 246
 duplicate test 173, 177, 185
 duplication 182, 184, 186
 factory methods 187
 removing 186, 196
 setup method 188
See also duplicate test
 dynamic fake 102
 dynamic fake object
 definition 102
 dynamic languages 266
 dynamic mock object 102, 104, 137
 dynamic mocks 99
 dynamic stubs 99
 combining with mocks 112
 dynamically generated object 104

E

easy-first strategy 242
 pros and cons 242
 EasyMock 100
 Eclipse 165
 Eclipse for Java 22
 empirical evidence 235
 encapsulation 77
See also object-oriented
 end assert 208
 end functionality 174, 182
 end result 83, 90, 158, 208
 Endres, Albert 236
 entry points for test-driven
 development 222

- event notification 26
- event recognition 26
- events 121
 - event source 124
 - event-related actions 121
 - event-related assertions 132
 - EventsVerifier 125
 - registering for an event 121
 - testing event triggering 124
 - triggering an event 123
- exceptions 14, 24, 38, 179, 199
 - ArgumentException 36
 - deliberate exceptions 33
 - expected exceptions 36
 - stack trace 24
 - unhandled exception 33
- Execute Tests button 252
- execution path 36
- expectations 104, 109, 135
 - assertion 105
 - best practices 136
 - expectations on mocks 105
 - expectations on stubs 105
 - mockEngine.Verify(mockObject) 105
 - mockEngine.VerifyAll() 105
 - MockRepository.GenerateStub 105
 - MockRepository.StrictMock 105
 - MockRepository.Stub 105
 - MockRepository.DynamicMock. 105
 - Moq framework 126
 - nonstrict mocks 106
 - order of method calls 109
 - overspecifying 136
 - recording stage 109
 - simulated object 104
 - strict mocks 106
 - stub object 111
 - Verify(mock) 105
 - verifying 111
- expected object 119
- ExpectedException 38, 44, 49
 - expected exception message 38
 - See also* exceptions
- extended reflection PIs 247
- extensions 240
- external dependency 50, 89, 258
 - definition 50
- external resource 44, 168, 196–197
- external-shared-state corruption 192
- Extract and Override 71, 73, 75, 81
 - when to use 74, 77
- F**
- factories 68
- factory class 168
 - fake factory 69
 - faking 70
 - implementation 68
- factory method 72, 151, 163–165, 167–168
 - virtualizing 76
- factory pattern 66
- failing test 18, 32–33, 36, 174, 178, 191, 200, 245
- fake class 249
- fake component 95
- fake dependency 70
 - See also* dependencies
- fake exception 129
- fake methods 134
- fake objects 44, 68, 84, 97, 108, 110, 189, 191
 - return values 108
- fake result 75
- fake values 247
- fakes 50, 90, 135, 191
 - when a mock object 90
 - when a stub 90
- faking 69–70
 - fake result 75
 - fake returning a fake 71
 - layers 69
- fast tests 39
- fast-running test 144
- Feathers, Michael 10, 55, 73, 96, 183, 246, 250–251, 253
- feedback 10
- FICC properties 257
- filesystem 50, 51, 53
 - configuration file 52
 - FileExtensionManager 53–54, 56
- final functionality 7
- final result *See* final functionality
- FinalBuilder 143
- FitNesse 246, 251, 283
 - download 253
 - usage 252
- FitNesse engine 252
 - wrapper classes 252

FitNesse table 252
 fixture 36, 252
 flow diagrams 224
 folder structure 145
 FUD (fear, uncertainty, and doubt) 220
 full objects 119, 203

G

Gallio 271–272
 generic implementation 15
 generics 152, 166, 247
 GlobalUtil 95
 goals 227
 good test 171
 good unit test 3
 See also unit test
 Gremillion, Lee L. 236
 gripped items 251
 groups
 test-code coverage report 226
 guerrilla-style implementation 223
 GUID (globally unique identifier) 116

H

Hanselman, Scott 62, 261
 hardcoded strings 209
 hard-first strategy 242, 244
 pros and cons 243
 hardware 236
 hardware simulator 236
 Haskell programming language 25
 helper classes 8, 117, 170
 Contains 117
 EndsWith 117
 Like 117
 StartsWith 117
 Text 117
 helper frameworks 63
 See also containers
 helper methods 15–16, 162, 186, 188,
 190–191, 198, 215, 258
 Hetzel, Bill 7
 hidden test call 192
 duplication 195
 flow testing 195
 laziness 195
 problems 195
 solutions 195
 HTML tables 252

HUnit 25
 Hunt, Andy 151

I

IDE *See* integrated development
 environment
IEEE Software 236
IEEE Transactions on Software Engineering 236
 IExtensionManager 58
 Ignore 38
 ignored test 38–39
 independent test 34
 indirect testing 40
 indirection 52–53, 71, 181
 layers 70
 infrastructure API 167
 inheritance 151, 157
 See also test class inheritance
 inheritance patterns 165
 initial state 196
 injecting 58
 injection 55, 89
 constructor injection 58
 factory class 66
 getting stub before method call 66
 local factory 71
 properties 64
 See also stub
 insiders 220
 integrated development environment 22
 integrating test-driven development 219
 integration process 144
 integration test 7–8, 45, 50–51, 169, 244
 failure points 8
 hidden 146
 problems 51
 slow-running tests 145
 integration testing 3, 7, 151, 168, 268, 272
 definition 8
 drawbacks 9
 See also integration test
 integration testing framework 280
 integration zone 147
 See also safe green zone
 integration-style test 244, 251, 277
 process 245
 intellectual property 264
 IntelliJ IDEA 165
 intellisense 210

interaction testing 82–83, 137
 action-driven testing 83
 comparison with state-based testing 83
 definition 83
 interactive user dialogs 14
 interface 52
 custom implementation 58
 extraction 55
 IExtensionManager 58
 interactive 53
 original implementation 58
 receiving as a property 55, 58
 receiving at constructor level 55, 58
 underlying implementation 52–53
 See also stub
 Interface Segregation Principle 78
 interface-based design 260
 internal method 183
 InternalsVisibleTo 78, 80
 invalid test 173, 176
 inversion of control 62
 inversion of control containers *See* containers
 Inversion of Control principle 63
 IoC 275
 See also inversion of control
 IoC containers 63, 261, 268, 273
 See also containers
 IP *See* intellectual property
 Iscoe, N. 237
 isolated test 256
 isolation frameworks 268
 See also mock object frameworks
 Isolator *See* Typemock
 Ivonna 278
 IWebService 104
J
 Java 25, 100, 131, 239, 248, 258
 EasyMock 100
 jMock 100
 legacy code 250
 Vise 250
 Java JWT 281
 Java Spring Container 276
 jMock framework 100, 131, 270
 JMockit 237, 246, 248–250
 test sample 249
 Johnson, Mark 227
 Jones, Capers 231
 JUnit 25, 192

K
 Krasner, H. 237
L
 lambda syntax 275
 lambdas 128, 134, 269
 leftover state 34
 legacy code 9, 132, 236, 239, 246, 254
 definition 10
 problems 240
 legacy code tools
 when to use which 246
 legacy project 244
 legacy system 51
 license.txt 26
 Liskov Substitution Principle 78
 lists 209
 Load event 122
 local factory 71
 log files 25
 LogAn project 25, 29
 LogAnalyzer 35, 41, 50, 67, 87, 105, 152
 interaction with web service 89
 LogAnalyzerTests 152
 logger dependency 152
 LoggingFacility 152
 logical code 12
 logic-driven 242
 long-running test 144
M
 magic values 137
 maintainability 4, 44, 63, 81, 89, 94, 96, 141,
 150, 171, 176–177, 181, 185, 195, 216,
 231, 233
 maintainable test 11, 16, 18, 25, 33, 61, 97,
 175, 188, 203, 205, 215
 maintenance mode 237
 management 224
 manual mocks 94, 96, 98, 103, 113
 problems 96, 100
 manual stubs 94, 96
 problems 96
 Martin, Robert C. (“Bob”) 78, 257, 263, 266
 MbUnit 192, 236, 271–272, 277
 parameterized test 201
 RowSet attribute 201
 McConnell, Steve 228
 McGraw-Hill 231

- MEF *See* Microsoft Managed Extensibility Framework
 - Meszaros, Gerard 50, 88, 149, 205
 - method 28
 - method behavior 182
 - method constants 181
 - method contract 182
 - method logic 36
 - method parameters 181
 - method strings 133
 - Microsoft 274–275, 282
 - Microsoft Castle 275
 - Microsoft CHES 281
 - See also* CHES
 - Microsoft Managed Extensibility Framework 274, 276
 - Microsoft Press 228
 - Microsoft Unity 62, 261, 274
 - missing test 181
 - mock classes 132
 - mock email service 91
 - mock frameworks 74
 - mock object 44, 82, 85, 87, 89–90, 99, 105, 108, 135, 189, 257
 - asserts 85
 - combining with stubs 89, 97
 - definition 84
 - difference with stub 84, 97
 - dynamic mocks 99
 - expectations 105
 - how many per test 94
 - LastCall 108
 - mock object frameworks 85
 - nonstrict mocks 106
 - optimal mocks per test 136
 - return values 108
 - reusing 89
 - setters 94
 - strict mocks 106
 - using mocks too much 97
 - what to test 136
 - mock object frameworks 98–99, 121, 126
 - advantages 100, 134
 - arrange-act-assert 126
 - definition 100
 - parameter constraints 115
 - Rhino Mocks 99
 - smart stubs 110
 - when not to use 135
 - mock service 124
 - mockpp 100
 - MockRepository
 - StrictMock 103
 - mocks 50, 269
 - See also* mock object
 - model-based testing 272
 - module 24
 - Moq framework 100, 126, 130, 134, 269
 - MSBuild 143
 - MSDN 271, 277
 - MSTest 271
 - multiple aspects 202
 - multiple asserts 94, 179–180, 198, 202–203
 - multiple method calls 134
 - multiple tests 15, 177, 200, 202
 - multiple threads 254
 - multithreaded test 179, 282
 - Myre, Glenford 235
- N**
- namespace 32, 39, 145, 147
 - naming convention 210
 - NAnt 143
 - NCover 180, 234
 - NDepend 246, 253
 - download 253
 - query language 253
 - new test 180
 - Newkirk, James (“Jim”) 237, 272
 - nightly build 147, 225
 - Ninject 62, 261, 274, 276
 - NMock 100, 131, 269–270
 - NMock2 131
 - difference with Rhino Mocks 131
 - nonsealed classes 258, 260, 267
 - nonstrict mocks 106–107, 136
 - nontestable design 262, 266
 - NUnit 21, 24–27, 29–30, 32, 35, 49, 102, 156, 192, 199, 271–272, 277
 - actions 34
 - assembly 29, 32
 - Assert class 30
 - attribute scheme 29
 - attributes 26, 29, 34–36, 44
 - automated test 30
 - Categories tab 40
 - color scheme 33
 - getting started 26

- NUnit (*continued*)
 - GUI 26, 32–33, 39
 - initial state 34
 - installing 26
 - open source license 26
 - parameterized test 201
 - Row attribute 201
 - RowSet attribute 201
 - Run button 32, 40
 - Selected Categories 40
 - SetUp attribute 34
 - special attributes 34
 - TearDown attribute 34
 - test actions 30
 - Test attribute 30
 - test class 30
 - test method 30
 - Test Not Run tab 39
 - TestFixture attribute 30
 - typename 32
 - unit test runner 26
 - version 26
- NUnit.Extensions.dll 201
- NUnit.Framework namespace 31
- NUnit.Mocks 130, 133, 269–270
 - limitations 131
- NUnitAsp 279
- NUnitForms 280–281

- O**
- object calls 82
- object configuration methods 167
- object model 167
- object-oriented 52, 66, 77–78, 151, 182, 257, 263, 266
 - encapsulation 77
 - object-oriented design 78
- Open Closed Principle 78, 257
- ordered mocks 109
- organizational change 223
 - bottom-up 223
 - top-down 223–224
- organizational culture 219
- organizational structure 228
- Osherove.ThreadTester 281–282
- outside consultant 224
 - advantages 224
- overriding 203
- overspecification 205
- overspecified test 206–207
 - features 205

- P**
- parameter constraints 115–116, 132
 - combining constraints 118
 - helper classes 116
 - parameter object properties 118
 - string constraint 116
- parameter injection 58
- parameter object refactoring 62
- parameter verification 120, 134
- parameterized test 199, 202
- parser class 162
- partial code test 23
- patent issues 265
- Peer Reviews in Software: A Practical Guide* 227
- Pex 271, 273
- phishing 95
- pilot project 223, 232
 - statistics 232
- pilot test 222
- political support 229
- polymorphism 258
- Poole, Charlie 131
- posters 225
- Pragmatic Programmer, The* 151
- presenter class 122
- “Principles of OOD” article 257
- priority 240, 244
- private method 182
- problem output 14
- Proceedings of the 8th International Symposium on Software Metrics* 236
- production bug 173
 - fixing 173
- production class 13
- production code 18, 29, 33, 66, 70, 72, 75, 79, 123, 136, 141, 148, 173, 180, 264
 - See also* production class
- production server 143
- productivity 231
- profiler APIs 247
- program exploration *See* Pex
- Programming Productivity* 231
- progress metrics 228
- project 148
 - See also* testing project
- project failure
 - causes 229

- project feasibility 222–223
- Project White 281
- properties 12, 64
- property injection 65
 - when to use 65
- protected method 182
- public functionality 206
- public properties 94

Q

- QA engineer 220, 233, 235
- QA process 233
- QA team 233

R

- readability 4, 61, 79, 89, 94, 96, 98, 128, 130, 135, 149–150, 170, 177–178, 209, 216
 - assert messages 209, 212
 - definition 171
 - importance 171
 - mock object initialization 215
 - separation of asserts and actions 214
 - setup and teardown method 214
 - test naming 210
 - variable naming 211
- readable test 11, 18, 25, 33, 67, 156, 188, 190, 203
 - best practices 209
- real object 90
- real test 159
- real-world scenarios 45
- record-and-replay model 102, 104, 126, 269
 - difference with arrange-act-assert model 128
- recording stage 122
- redefine class 249
- Red-Green-Refactor 33
- Refactor from DevExpress 165
- refactored test 176
- refactoring 17–18, 55, 87, 95, 132, 150, 152, 154, 165, 168, 173, 177, 180, 186, 190, 200, 208, 239, 244, 250
 - automated tools 134
 - definition 19, 55
 - maintainable state 186
 - over-refactoring 190
 - Vise 250–251
- refactoring pattern
 - extracting an interface 55

- refactoring phase 244
- regression 9
- reinvention 230
- renaming 173, 177
- Repeat 272
- repeatable test 6, 22
- repetition 137
- replaceability 260
- ReSharper 165, 246, 253, 264
 - download 254
 - navigations 253
- ReSharper for .NET 134
- result-driven testing 83
 - end result 83
- return values 212
- Rhino Mocks 99–100, 107, 110, 114, 116, 126, 132, 137, 269
 - arrange-act-assert model 103
 - comparison with other .NET mock object frameworks 129
 - introduction 102
 - Mock Repository 103
 - ordered mocks 109
 - parameter constraints types 117
 - record-and-replay model 102
 - Rhino.Mocks.Dll 102
 - single parameter method 121
 - smart stubs 110
 - stub object properties 111
- roles 259
- Rollback attribute 277
- Ruby 265, 279
- runtime 98–99, 102, 249

S

- safe green zone 147, 169, 180
- Scrum 226
- sealed classes 78
- seam 58, 67–68, 70–71, 74, 80
 - target layer 70
 - See also* seams
- seams 55, 68, 74, 240, 257, 260, 263
 - creating 58
 - definition 55
 - hiding 69
 - multiple 259
 - seam 67–68, 70
 - seam statements 69
- security 260, 264

- selection strategy 242
- Selenium 278, 280–281
- semantics 174, 264
 - changes 175, 185
 - See also* test semantics
- SetUp 35, 45, 49, 187
 - See also* NUnit
- setup action 34
- setup code 151
- setup method 152, 154, 156–157, 162, 170, 186, 189, 196, 198, 273
 - best practices 190
 - initializing objects 189–190
 - maintainable 190
 - wrong use 188
 - See also* setup action
- shared class 188
- shared factory method 186
- shared resources 196, 198
- shared-state corruption 192
 - causes 197
 - maintainability 197
 - manifestation 196
 - problems 197
 - solutions 198
 - test subsets 197
- Simian 246, 254
 - download 254
- similar tests 177
- simple objects 49
- simple unit test 12, 27
- single asserts 200
- single responsibility principle 262, 264
- single unit 9
- singletons 198, 258, 261
- slow tests 39
- slow-running test 144
- Smalltalk 4, 265
- smart factories *See* inversion of control
- smart stubs 110
- software 236, 256
- Software Assessments, Benchmarks, and Best Practices* 231
- software development 4, 23
- software module 8
- Software QA Quarterly, The* 227
- SOLID 78, 237, 267
- Solution Explorer 128
- source control 142, 144, 147
- Spring.NET 62, 261, 274, 276
- SRP *See* single responsibility principle
- state sharing 36
- state verification *See* state-based testing
- state-based testing 40, 82–83, 137
 - comparison with interaction testing 83
 - definition 40
 - result-driven testing 83
 - when to use 84
- static languages 266
- static method 183, 258, 260
- static state 198
- statics 67, 258
- StoryTeller 283
- strict mocks 106
 - failure 106
 - StrictMock method 107
- structured test 16, 22, 24
- StructureMap 62, 261, 274–275
- stub 54, 58, 66
 - asserts 85
 - combining with mock object 89
 - definition 50
 - difference with mock object 84
 - getting before method call 55, 66
 - injecting stub implementation 55, 58
 - injection 80
 - receiving before method call 58
 - setters 94
 - stub analyzer 60
 - stub extension manager 57
 - stub injection 58
 - stub manager 57
 - stub method 71
 - stub object 61
- stub chains 95
- stub class 57, 61, 72
 - configurability 57
 - StubExtensionManager 53
 - See also* stub
- stub loggers 167
- stub method 71
- stub object 71
 - See also* stub
- stubbing *See* faking
- StubLogger 152
- stubs 49, 82, 87, 102, 108, 124, 136, 269
 - combining with mocks 97
 - difference with mocks objects 97

- stubs (*continued*)
 - dynamic stubs 99
 - fake values 110
 - how many per test 94
 - preference over mock objects 209
 - subclass 164
 - subteam 222
 - superclass 165
 - swap approach 249
 - swap class 249
 - system initialization methods 167
 - system state 167
 - system test 144
 - system testing 168
 - system under test *See* unit testing
- T**
- T generic type 167
 - target server 143
 - tasks 144
 - TDD 17–18, 227, 235, 257
 - See also* test-driven development
 - team formation 222
 - team support 230
 - Team System tools 281
 - Team System UI Tests 280–281
 - Team System Web Test 278
 - TeamCity 143
 - TearDown 35, 45, 49
 - See also* NUnit
 - teardown action 34
 - teardown code 151
 - teardown method 196, 198, 273
 - See also* teardown action
 - technical definition 5
 - techniques 268
 - template test class pattern 158
 - test actions 30
 - test API 150
 - test assembly 146
 - test assertion 30, 38, 43
 - assert class 24
 - assert message 24, 31
 - assert method 24
 - test blockage 191
 - test bug 173, 178
 - debugging 174
 - fixing 174
 - test categories 39
 - test class 27–28, 30, 36, 43–44, 103, 148–149, 154, 164
 - test class hierarchy 170
 - See also* class
 - test class inheritance 151
 - abstract test driver class 151
 - abstract test infrastructure class 151
 - template test class 151
 - test class patterns 148–149
 - one-test-class-per-class pattern 149
 - one-test-class-per-feature pattern 150
 - test code 30, 210
 - semantics 74
 - test code coverage 227
 - code coverage tools 227
 - sample report 234
 - test code coverage report 225
 - test complexity 178
 - test conditions 29
 - test constructors 36
 - test design 16, 18, 237
 - guidelines 257
 - interface-based design 258
 - object-oriented 266
 - test destructors 36
 - test development
 - time consumed 242
 - test feasibility table 240
 - test fragility 178, 205
 - test frameworks 268, 271
 - test hierarchies 141
 - test isolation 182, 191, 198
 - flow testing 193
 - importance 193
 - laziness 193
 - multiple asserts 200
 - test design 193
 - test layers
 - advantages 70
 - disadvantages 70
 - test loading 29
 - test logic 41, 178
 - test mapping 148
 - test method 13, 27–28, 30, 40, 44, 57, 159, 164–165
 - expected behavior 29
 - method name 29
 - multiple 150
 - test name 210

- test naming 18, 28, 33, 179
 - naming conventions 28
 - test order 192
 - test output 205
 - test pass 18
 - test path 242
 - test pattern names 50
 - test project 44, 145, 149
 - test quality 177
 - test result 40
 - test review 172, 191, 221
 - test run 32, 34, 36, 67, 252
 - test runner 14, 24, 199
 - test results 24
 - test semantics 173, 184
 - Test Spy 88
 - test state 36, 196–197
 - test stubs 44
 - test subsets 193, 195
 - test suite 252
 - testability 45, 53, 55–56, 78, 150, 244, 247, 256–257, 263, 270, 273
 - alternatives 265
 - complexity 264
 - designing code 238
 - problems 248
 - testable design 78, 81, 256–257, 262–263
 - properties 256
 - testable object-oriented design (TOOD) 78, 81
 - testable system 257
 - test-code coverage report 226
 - test-driven development 3, 16–18, 33, 182, 219, 225, 236, 257
 - incremental 17
 - style choice 237
 - technique 18
 - Test-Driven Development in Microsoft .NET* 237
 - test-driven development techniques 239
 - test-first 16
 - TestFixtureSetUp 36
 - TestFixtureTearDown 36
 - testing guidance 151, 159
 - testing project 29
 - test-inhibiting design 51
 - third-party tools 25
 - Thomas, Dave 151
 - threading problems 254
 - thread-related testing 268, 281
 - threads 50
 - three pillars 172
 - threshold of logic 241
 - Timeout 272
 - TOOD *See* testable object-oriented design
 - tooling 239
 - tools 268
 - top-down change 223–224
 - ToString() 203
 - traditional coding *See* traditional development
 - traditional development 18
 - TransactionScope 277–278
 - triggers 144
 - trustworthiness 171, 215
 - trustworthy test 172, 180
 - definition 171
 - guidelines 172
 - try-catch 199, 202
 - Typemock 100, 126, 128–129, 137, 237, 244, 246, 248, 260, 266, 279
 - expectations 132
 - Typemock code
 - download 254
 - Typemock Isolator 97, 269
 - See also* Typemock
 - Typemock Racer 246, 254, 281–282
- ## U
- UI testing 268, 280–281
 - UIAutomation API 281
 - UIs 280
 - unit test 3, 6, 11, 23, 45, 126, 141, 169
 - actions 30
 - automation 10, 24
 - core techniques 138
 - ease of development 11
 - ease of running 180
 - fast tests 39
 - fast-running tests 145
 - final definition 11
 - frameworks 22
 - good unit test 5–6, 9–11
 - human factor 146
 - initial state 34
 - lifecycle 34
 - multiple tests 57
 - properties of a good unit test 6
 - quickness 10

- unit test (*continued*)
 - removing 173
 - running a test 24
 - setup methods 34
 - slow tests 39
 - status 24
 - test placement 141
 - tools 44
 - unit 4
- unit testing 4–5, 83
 - classic definition 4
 - coding 233
 - designing for testability 266
 - effect on release date 232
 - GUI 6
 - implementation 230, 232–233, 238
 - metrics 234–235
 - overall time reduction 233
 - system under test (SUT) 4
 - time added 231
 - time consumed 232
 - tough questions 231
- unit-testing framework 11, 22–24, 145, 268
 - attributes 24
 - base classes 24
 - benefits 22
 - interfaces 24
 - list 24
 - xUnit framework 25
- unit-testing technique 18, 243–244
- Unity 274–275
 - See also* Microsoft Unity
- Unix 22
- unreadable test 177
- user interface (UI) 5
- utility class 150, 167, 241
- utility method 141, 150, 167, 176, 183

V

- valid test 175
- validation check 41
- value to the project 241
- variable naming
 - importance 211
- VB.NET 124, 236, 260
- verbosity 201
 - MbUnit 201

- NUnit 201
- versioning info 159
- view class 122
- virtual by default 258
- virtual method 71–72, 75, 96, 134, 258–260, 267
- virtualizing 76
- Vise 246, 250–251
- Visual Build Pro 143
- Visual Studio 79, 128, 134, 149, 165, 253, 271
- Visual Studio .NET 22, 272
- Visual Studio 2008 26
- Visual Studio Team Foundation Server 143
- Visual Studio Team System 271, 281–282
- Visual Studio Team System Test Edition 180
- Visual Studio Team Test 279
- VS.NET 253

W

- WatiN 278–279, 281
- Watir 278–279
- web service 49, 61, 74, 83, 87, 89–90, 113, 129, 257
 - MockWebService 87
 - stub 90
- web testing 268, 278
- whiteboards 225
- Wieggers, Karl 227
- wiki page 252
- Win32 281
- WinForm 281
- Working Effectively with Legacy Code* 10, 55, 73, 96, 183, 253
- WPF 281
- wrapper classes 96

X

- xUnit 25, 271, 273
- xUnit framework 25
- xUnit Test Patterns 50, 88, 149, 205

Y

- YUnit 271