

A

Abbot 295, 297, 300, 418
 ComponentTester utility 297
 ComponentTestFixture
 class 297
abstract base class 174, 181,
 206, 237
 for DbUnit-based integration
 tests 239
Abstract Factory pattern 127
abstract user interface 356
abstraction 35, 212, 275
 between MVC controller and
 view 169
 keeping uniform level of
 87–88
 level of 419
 level, consistency in
 methods 69–70
acceptance 450
 exercising the internals 414
acceptance criteria 353
acceptance TDD *See* acceptance
 test-driven development
acceptance test 4, 33, 35, 324,
 335, 341, 404, 440
 accelerating execution
 426–431
 adding detail 337
 adding new 353
 as communication tool 421
 as indicator of completion 31
 as shared language 33
 as specification *See* tests as
 specification

automating 337, 340, 345, 391
automating through UI 417
avoiding turbulent
 interfaces 399
brittleness 403
categorization 426
challenge of automating 417
common fixtures
 between 431
complexity 425, 431
connecting with system under
 test 401
crawling under the skin 404
customer-friendly format 359
dependency between
 tests 429
design 403
detail in 332
difference between unit test
 and internals-exercising
 acceptance test 409
duplication 431
embedding into system under
 test 419
end-to-end strategy 401
example of 327, 330
executable, syntax of 339, 341
expressing with concrete
 examples 354
focus of 398
for FTP client 416
for programming library 413
for web application 431
generating data set for 433
getting access to
 transaction 430

heavyweight 409
hooking into service layer 422
identifying 342
implementation
 approaches 401–411
implementing 333, 337, 406
implementing against
 programming API 416
implementing for complex
 systems 425
introduction 327–333
list of 336
maintainability 429
mapping to user-accessible
 functionality 403
merging with unit test 413
organizing into test suites 428
owned by customer 329
passing 31
performance 356
properties 328–333
reducing complexity 431
refactoring 404, 425
running against stubbed
 system 410
running on simulator 358
side effects 411, 426
structuring 431
stubbing 357–358
subsets of 426
suite 403
targeted at system
 internals 408
technology-specific
 implications 411–425
timing 337

- acceptance test (*continued*)
 - tips 425–433
 - tools for implementing 338
 - translating to executable test 338–339
 - updating 399
 - validation of 331
 - what to test 397–401
 - when to write 345–346
 - writing 329, 337, 345, 350, 360
 - writing with jWebUnit 401
 - writing with wiki syntax 360
- acceptance test-driven
 - development 4, 7, 10, 31, 98, 324, 327, 341, 401
 - as team activity 348–351
 - automating tests 337–341
 - benefits 351
 - benefits of 351–355
 - continuous planning 344–346
 - cooperative work 353
 - customer properties 349–350
 - cycle 334–343
 - decreasing load 346
 - defining customer role 348–350
 - definition of “done” 351–353
 - frameworks for 37
 - homegrown tools 362
 - implementing
 - functionality 341–343
 - inside an iteration 343–348
 - mid-iteration sanity test 346
 - number of testers 350
 - picking user story 335–336
 - process 324
 - process, understanding 334–348
 - promoting 446, 463
 - properties of 324
 - scripting language-based frameworks 361
 - specification by example 354
 - splitting stories 347
 - table-based tools 359–361
 - text-based frameworks 361
 - tools 359–362
 - trust and commitment 354
 - what we’re testing 355–359
- whether to stub parts of system 357
- whether to test against UI 355–357
- whether to test business logic 358
- who writes tests with
 - customer 350
 - with Fit 365–369
- acceptance testing 37, 361
 - approach to 400
 - backdoor 411
 - crawling under the skin 412, 414–415, 419
 - end-to-end 406, 416–418, 421, 425
 - exercising the internals 407, 412, 414, 421
 - framework 37, 359, 365
 - implementation
 - approaches 401
 - strategy 412, 414, 416
 - strategy, choosing 411
 - strategy, cost 414
 - stubbing out the irrelevant 409
 - technological considerations 411
 - tools 359
- access point 212
- accessor method 211, 291
- accomplishment 438, 442
- account manager 12
- ActionFixture
 - 372, 379–382, 384
 - header row 379
- ActionListener 304
- actor 379
- Adobe Photoshop 417
- adoption 4, 10, 436
 - assigning roles 463–464
 - coaching and facilitation 461–463
 - destabilization 464
 - evangelizing for change 454–457
 - facilitating 454–465
 - lowering bar to change 457–458
 - of Java 200
 - rewards, delayed 465
 - sharing information 459–461
 - training and education 458
- Agile 2005 336
- Agile Finland 451
- agile methods 19, 454, 458
- agile software development 7, 31
- Agile Toolkit Podcast 336
- agility
 - key ingredient of 7
- Ajax 423
- alarm system 8
- Alexander, Christopher 364, 396
- algorithm 58, 77, 85, 101, 116, 223, 378
 - using to find connecting line 313
 - verbosity 313
- Alleman, Glen B. 323
- alternative implementation 90, 111, 116, 120, 124, 140, 160
- alternative implementation
 - See* test double
- Alur, Deepak 198
- ambiguity 21, 33–34
- Ambler, Scott 244
- analytic thinking 452
- annotation 50, 67, 186, 211
- anonymous class 291
- anonymous
 - implementation 139, 316
- anonymous inner class 141
- Ant *See* Apache Ant
- AntHill 39
- antidote 7
- anti-pattern 109
- Apache Ant 365, 392
 - build script 391
 - build script, adding target for running tests 478–479
 - build script, compiling source code 477–478
 - build script, directory structure of 476–477
 - build script, generating report 479–480
 - build script, running Fit with 392
 - FTP task 393

- Apache Maven 392
- Apache Shale 186
- Apache Tapestry 185–186, 193
- Apache Tomcat 174, 179
- Apache Velocity 174, 179
 - API 179
 - basics 179–180
 - template 179–180, 184
 - template plumbing 180–182
 - template, adding
 - assertions 182–184
 - templates, test-driving 179–184
- API 213, 225, 244–245
 - API-level access 400
 - awkwardness of 156
 - correct use of 221
 - designing with test 16
 - exposing logic through
 - public API 412
 - for controlling web
 - browser 424
 - for Hibernate 212–214
 - for regular expressions in
 - Java 84
 - for Velocity 179
 - independence of JDBC
 - API 212
 - internal, testing through 404
 - invoking public 144
 - native, end-to-end testing
 - through 418
 - obtaining system time
 - through standard API 255
 - of jWebUnit 402
 - operating system under test
 - through 406
 - Swing API 282, 284
 - third-party 103, 196
- Apple 455
 - application
 - console-based 416
 - deployment 422
 - desktop 417
 - domain 416
 - event-driven 417
 - graphical user interface 417
 - logic 419
 - logic, separating from
 - presentation 416
 - under test, launching 420
 - under test, state of 418
 - application domain 348–349
 - application logic 197, 223
 - application server 174, 220, 250, 422
 - restarting 430
 - architectural change 22
 - architecture 20, 22, 155–156
 - evolving 400
 - exposing internal API 405
 - for dependency injection 172
 - for web applications 154
 - system under test 333
 - verifying with end-to-end tests 414
 - Arendt, Hannah 325
 - artifact 186
 - artificial intelligence 414
 - ASCII 139
 - assertion 56, 67, 79, 138, 176–177, 181–182, 241–242, 268, 272–273, 393
 - Customer Assertion
 - pattern 130
 - Delta Assertion pattern 129
 - for color of a pixel 312
 - Guard Assertion pattern 128
 - hard-coding 313
 - HTML documents 183
 - Interaction Assertion
 - pattern 131
 - patterns of 128, 131, 137
 - performing to verify expected behavior 169
 - Resulting State Assertion
 - pattern 128
 - assertion pattern 128–132
 - assertRenderedPage
 - method 193
 - assumption 51, 221
 - about current behavior 73
 - about database schema or domain model 221
 - about fixture 128
 - about object-relational mappings 221
 - about query statements 221
 - about thread scheduling 272
 - flawed 4, 447
 - verifying with
 - characterization tests 147
 - with learning tests 77
 - asynchronous execution 261, 271, 273
 - verifying 271–274
 - ATDD *See* acceptance test-driven development
 - atomic operation 264
 - atomicity
 - of test 47
 - attack
 - responding with 448
 - Auckland University 384
 - authentication 160, 162–163
 - authority 441, 462
 - automated build 283
 - executing Fit tests 393
 - Automated Teardown 136
 - automated test
 - See* executable test
 - automation 26, 28, 34, 173, 224
 - availability 220
 - awareness 246, 454
 - of one's own behavior 462

B

- Bache, Geoffrey 361
- backdoor *See* testing backdoor
- background thread 269, 271, 274, 277, 282
- background code
 - for fixture table 371
- balance
 - between methods 69
- Bamboo 39
- barricade
 - to protect from change 448
- barrier 262, 264–265, 276–277
- barrier to entry 458
- baseline 14
- Bash 430
- batch 276
- batch processing 22
- battle
 - picking 453–454
- bean 170
- BeanInject 141
- Beck, Kent 36, 104

- behavior
 - changes in 443
 - close to desired 19
 - correct, testing 417
 - depending on system
 - time 250
 - description of 49
 - expected 49
 - expected, testing for 147
 - exposing or modifying 144
 - externally visible 27
 - nailing down 147
 - of system 11
 - of test double 110
 - of view components 282
 - preserving 27
 - reflecting on 462
 - test-driving into JavaServer
 - Page 177
 - testing for 193
 - triggering with event
 - queue 405
 - value of 102
 - verifying by examining log
 - file 418
 - verifying expected 110
 - Belshee, Arlo 336
 - benefit
 - of adopting TDD 14
 - of delivering often 33
 - of having adopted TDD 11
 - of having proper fixtures 109
 - of incremental DDL
 - scripts 233
 - of using tests as
 - specification 35
 - bias 462
 - big picture 102, 456
 - bitmap
 - capturing 312
 - bitmap raster
 - extracting from buffer 310
 - black market 267
 - black-box testing
 - 401–403, 418–419
 - blanket statement 439
 - blocking 261, 265–268, 270, 276–277
 - blocking operation
 - test-driving 266–268
 - boilerplate 205, 211
 - bootstrapping
 - dependencies 127
 - bounds
 - comparing to size 309
 - Bourrillion, Kevin 272
 - breadth first 45
 - breadth-first 58, 80, 101
 - brittle test 112
 - brittleness 173
 - Brooks, Frederick P. 148
 - brown bag session 460
 - browser
 - simulating 360
 - browser *See* web browser
 - brute force 262
 - buffer 277
 - rendering component on 310
 - BufferedImage 311
 - bug count *See* defect rate
 - bug report 32
 - bugginess 144
 - build
 - automated, testing
 - during 392–394
 - building the right thing 7, 11
 - building the thing right 7, 11
 - build cycle
 - integrating Fit 391
 - build script 224
 - Ant, adding target for running
 - tests 478–479
 - Ant, compiling source
 - code 477–478
 - Ant, directory structure
 - of 476–477
 - generating report 479–480
 - build server 29–30, 38–39
 - build.xml 477
 - burn rate 346
 - business domain 132
 - language of 409
 - business logic 48, 155–156, 196–197, 356–358
 - testing directly 359, 408
 - verifying 390
 - business rule 374
 - isolating 409
 - verifying 426
 - verifying with acceptance
 - test 408
 - button
 - wiring to trigger events 304
 - buy vs. build 362
 - byte code 121, 261
-
- ## C
- C# 360, 366
 - C++ 123
 - caching 213, 430
 - caching context object 431
 - callback 272, 277, 315
 - camel case
 - in DoFixture cells 385
 - canvas component 281, 310
 - capacity
 - of change agents 461
 - of working memory 22
 - card *See* story card
 - card, conversation,
 - confirmation 332
 - catalyst 135
 - centralized testing function 437
 - certainty 102, 271, 464
 - challenge
 - of being able to deliver 8
 - champion 441
 - change 440, 447
 - ability to lead 436
 - ability to lead, as related to
 - role 441–443
 - absorbing flux of 460
 - accepting 449
 - adopting facilitation 454–465
 - anticipated change 23
 - assigning roles 463–464
 - benefits of 437, 442, 448, 452, 456
 - boundary 440
 - changing code 24
 - changing interface 22
 - coaching and
 - facilitation 461–463
 - cost of 400
 - daring to make 14
 - encouraging by
 - destabilization 464
 - evangelizing for 454–457
 - getting others aboard
 - 440–444
 - in behavior 445

- change (*continued*)
 - knowing what to change 44
 - lowering bar 457–458
 - making big change in small steps 76
 - making too big 80
 - piggybacking on other change 440
 - reflecting on 457
 - resistance against 445
 - resistance to, fighting 444–454
 - resistance to,
 - overcoming 449–453
 - resistance to, picking battles 453–454
 - resistance to,
 - recognizing 444–448
 - resistance to, three standard responses 448–449
 - reverting after too big step 107
 - rewards, delayed 465
 - running tests after 96
 - sharing information 459–461
 - significance of 53
 - software evolving through 29
 - stimulating 464
 - stream of small changes 38
 - sustainability of 451
 - sustainable 436, 438, 440
 - testing changes 13
 - time for 439
 - training and education 458
 - uncertainty of 23
 - understanding 440, 443
 - world changing around system 7
- change agent 442–443, 446, 449–450, 454–455, 464–466
- change boundary 440
- change champion *See* champion
- change point 146–147
- characterization test 147
- check action 380
- checkComponent method 295
- child fixture
 - delegating to 385
- choice 451
- class
 - anonymous 291
 - class hierarchy 105, 119
 - causing ripple effect through 119
 - class loader 240
 - class loading 108
 - class path 213, 227, 229, 240
 - loading XML file from 238
 - class under test 111–112, 115, 142, 258
 - classification
 - of test doubles 113, 115
 - CLI *See* command-line interface
 - clickLink method 193
 - close proximity 146–147
 - Clover 40
 - cluster 220
 - coach 461, 465
 - coaching 452, 456, 461, 463, 466
 - Cobertura 40
 - Cockburn, Alistair 459
 - code
 - dead, deleting 97
 - failing to meet needs 6–7
 - just enough code 18
 - maintainability 324
 - poorly written 5–6
 - producing ugly code 19
 - code base 142, 145–146
 - broken 81
 - cleaning up 10
 - growing 8
 - internal quality of 281
 - leaving duplication into the 106
 - refactoring toward system-time abstraction 252
 - code coverage *See* test coverage
 - code smell 64, 89, 106
 - code without tests
 - See* legacy code
 - coding dojo 451, 460
 - cohesion 109
 - Cohn, Mike 327
 - coinciding change 444–445
 - collaboration 10, 31–33, 37, 45, 112, 274, 350, 447, 463
 - between developers, testers, and customers 359
 - between Hibernate interfaces 212
 - expected 117
 - for common goal 464
 - ineffective, risk of 421
 - of fixture objects 209
 - supporting through
 - acceptance tests 425
 - collaborator 110–111, 117, 139
 - interacting with 131
 - testing interactions with 111
 - collective code ownership 37
 - ColumnFixture 372–376, 389
 - example of 374
 - extending 383
 - COM component model 424
 - comfort 448
 - command and control 453
 - command object 405
 - command-line interface 415
 - commitment 354, 443, 451, 458, 463
 - commodity 4
 - common good
 - appealing to 449
 - communication 32, 34, 110, 439, 442
 - efficiency of 33
 - making tests communicate better 130
 - sustaining 349
 - communication overhead 428
 - community of practice 461
 - comparison
 - between data sets 241–242, 379
 - compatibility 91, 223
 - compilation 47, 55
 - compiler 36, 161, 182, 221, 261
 - satisfying 91
 - completeness 456
 - complexity 17, 21, 26, 110, 223
 - architectural 223
 - gaining more and more 81
 - keeping an eye on 86
 - of business rules 409
 - of desktop applications 417
 - of interaction-based tests 112
 - semantic, of desktop applications 417
 - tackling 58
 - component
 - custom 304
 - laying out with Swing API 281

- component (*continued*)
 - layout with template language 185
 - plot map, adding and operating standard widgets 300–304
 - plot map, associating gestures with coordinates 314–319
 - plot map, drawing custom graphics 304–314
 - plot map, example 297–319
 - plot map, layout design 298–300
 - setting size 308
 - UI 185
 - view, adding and operating standard widgets 300–304
 - view, associating gestures with coordinates 314–319
 - view, drawing custom graphics 304–314
 - view, layout design 298–300
 - view, test-driving 297–319
 - view, tools for testing 290–297
- component chooser 294
- component hierarchy 291–292
 - of Java Swing application 419
- component operator 294
- component tree 191–192, 419
- component-based framework
 - 174, 185–186, 193
 - test-driving 184–193
 - typical 185–186
- component-based web framework 154
- ComponentChooser
 - interface 294–295
- ComponentTester utility 297
- ComponentTestFixture
 - class 297
- composition 119
- compromise 454
- concept
 - testing 108
- concurrency 260, 262, 264–265
- concurrent behavior 259, 261
- concurrent programming
 - 250, 259–260, 271, 275
 - literature on 261
- conditional “yes” 446
- conditional construct 179
- confidence 10, 12–14, 44–45, 262
 - false 13–14
- configurability 254
- configuration 127, 186–187
 - for Hibernate 213
 - for Spring Framework 170
 - managing database configuration 222
 - of Spring Framework 168
 - problem with non-default configurations 224
 - programmatic configuration of Hibernate 213
 - test configuration for Hibernate 227
- Configuration class 213, 229
- configuration management
 - 224, 391
- conflict 462
- confrontation 446
- confusion 452
- congruency 439
- Connection interface 203
- consensus 460, 462
- consistency 439
 - of abstraction level 69
- console application 415
- constant 167, 258
- constraint 352
 - checking 390
- construct 39, 60, 119
 - try-catch 66–67
- constructor injection 164
- consulting 456
- context object 431
- context-driven testing 463
- continuous design 44
- continuous integration 29, 36, 38–39, 353, 359
 - philosophy behind 39
- continuous integration server 394
- continuous planning 344
- Continuum 39
- control 262, 266, 464
 - yielding over junior colleague 441
- controller
 - 154, 156–172, 283–284
 - in model-view-controller 155, 163
 - in Spring Framework 168
 - Spring, managing dependencies 170–172
 - Spring, test-driving 168–172
- controller logic 283
- conversation 458
- conversion 91, 98, 206, 208, 240
 - of column contents 374
 - of static methods into instance methods 121
 - of user input 281
 - to executable test 367
- Cooper, Alan 135
- cooperation
 - resulting from acceptance TDD 353
- coordinate
 - associating with gesture in view component 314, 319
- coordinates
 - associating with gestures 314
- correctness 10, 111, 271, 275
- cost
 - of acceptance test maintenance 406
 - of additional code 120
 - of asking question 403
 - of building and maintaining homegrown framework 362
 - of building things in specific order 33
 - of defects 5, 10
 - of delaying feature 33
 - of dependency injection 126
 - of development 9
 - of evolutionary design 23
 - of fixing defects 5, 9
 - of maintenance 9
 - of over-engineering 18
 - of tests accessing network 357
 - operational 5
 - perceived 20
- CountDownLatch class 270, 276
- counting semaphore 275
- coupling 11, 122, 124, 197, 283, 310
 - between controller and view 287
 - trade-off with performance 287
- CPU 259, 261

- Crane, David 423
 - createCanvas method 306
 - creation method 134–135, 137
 - credibility 439
 - criteria
 - for completion 50
 - CRM *See* customer relationship management
 - cross-functional team 336
 - CRUD operation 212
 - CruiseControl 39, 394
 - Crupi, John 198
 - Crystal Clear 459
 - CSS 187
 - Cunningham, Ward 360, 365, 384, 392, 394
 - Custom Assertion 130
 - custom thread class 271
 - customer 6, 10, 31–32, 464
 - asking for details 332, 335, 342, 347
 - authority 349
 - availability of 339, 345
 - collaborating with 329–330, 333, 343, 350, 359
 - defining role in acceptance TDD 348–350
 - functionality valued by 341
 - involvement 331, 349
 - language 349
 - needs of 5, 7, 11
 - on-site 334
 - role 33, 348–350
 - satisfied 32
 - selecting 348
 - customer relationship management 23
 - customer team 349
 - customer test-driven development *See* acceptance test-driven development
 - cycle
 - acceptance TDD cycle 334, 341
 - component life cycle 185
 - integration test cycle 231
 - of implementing user story 352
 - TDD cycle 8, 11, 15–16, 19, 38, 44, 106–107, 173, 184, 243, 341
 - TDD cycle with database integration tests 244
 - CyclicBarrier class 264–265, 276
- D**
-
- DAO pattern
 - separating layers with 198–199
 - DAO *See* data access object
 - Dashorst, Martijn 193
 - data 155
 - as part of systems 196
 - comparing datasets with RowFixture 379
 - creating test data 113
 - displaying 283
 - driving access with unit tests 199–219
 - exchanging at synchronization point 276
 - extending data set 432
 - for acceptance tests 432
 - generating test data 432–433
 - inheriting 25
 - managing test data 432–433
 - operating on 196
 - parameterized
 - See* Parameterized Test
 - partitioning test data 432
 - plotting from database 281
 - populating 383, 388
 - pulling from text fields 304
 - setting up for unit test 86
 - static definition of test data 433
 - test data 359
 - data binding 284
 - data definition language 230–231
 - data set
 - comparing with DbUnit 241–242
 - defining 240–241
 - definition of 433
 - data-access code 196, 198, 205
 - crossing boundaries 197
 - simplifying 199, 205
 - data-access object 198–201, 206, 219, 221, 225–226, 236–237
 - implementing with
 - Hibernate 213–214, 216, 218
 - implementing with JdbcDaoSupport 210
 - implementing with JdbcTemplate 208, 210
 - implementing with plain JDBC 200, 202
 - test-driving with
 - JdbcTemplate 208–209
 - database 197–198, 222–223
 - accessing and controlling 222
 - as external resource 29
 - browsing contents of 225
 - cleaning up objects from 136
 - closing connections to 218
 - code, testing with DbUnit 237
 - connecting directly to 411
 - connecting to 115, 200, 229
 - connection 218
 - connection, providing to DbUnit 237
 - creating from scratch 230
 - engine 221
 - exporting contents of 237, 241–242
 - initializing system from 430
 - live connection to 110, 113
 - maintaining in known state 226
 - mapping objects into 211
 - populating 235, 237, 239, 430
 - populating in test setup 429
 - proprietary features of 212, 223
 - refactoring 199, 244
 - schema, creating 233
 - selecting for integration test 222–225
 - sequence 222–223
 - server 196, 220, 224
 - database administration 447
 - database administrator 230
 - database schema 219, 221–222, 230–231, 243–244
 - automatically exporting with Hibernate 230
 - creating 230

- database schema (*continued*)
 - creating as DDL scripts 231–233
 - exporting into database 230
- database sequence 223
- database server 224
- database transaction
 - See* transaction
- DatabaseTestCase class 237–238
- data-driven testing 137–138
- DataSource 200–201, 203, 212, 220
- DB2 222
- dbdeploy 231–232
- DbUnit 235, 237–239, 242–243
 - data set, defining 240–241
 - data sets, comparing 241
 - DatabaseTestCase class 237
 - file format 240
 - populating data 237
- DDL script
 - incremental 231, 233
- DDL *See* data definition
 - language
- de Tocqueville, Alexis 108
- dead code 97
 - deleting 97
- deadlock 261
- decision
 - delegating to someone else 163
 - implications of 349
 - making 349
 - of which test to write next 100
- decision making 462
- decomposition
 - example of 46
 - of requirements 45
 - of requirements to tasks 46
 - of requirements to tests 46
- defect 5, 12, 24, 28, 34, 437, 457
 - consequences of 6
 - defect-free 13
 - fixing 9
 - introducing because of duplication 27
- defect rate 5, 24, 44
- defensive behavior 448
- defensive functionality 103
- definition
 - of “done” 46
 - of good test 47
 - of data set 433
 - of unit test 199, 245
- delayed reward 465
- delegation 25, 119, 142, 170, 252–253, 427
 - of user gestures 284
- Delta Assertion 129–130
- delta script 231–232
- DeMarco, Tom 347, 457
- Deming, W. Edwards 437
- dependency 81, 92, 110, 122, 170, 175, 282, 284
 - acquisition of 122, 125
 - between acceptance tests 429
 - between user stories 335
 - breaking out 145, 147
 - holding in instance variable 125
 - injecting 124
 - injection,
 - constructor-based 126
 - injection, field-based 126
 - injection, setter-based 126
 - isolating 118, 122–124, 142
 - on system time 251
 - substituting 142
 - to APIs that are difficult to mock 156
 - to Java EE 179
 - to JDBC interface 206
 - using constructor-injection for 164
- dependency injection 118, 124, 126, 142, 200, 213, 216
 - architecture for 172
 - example of 125
 - in Spring Framework 170
- depth-first 45, 58, 80, 101
- design 44
 - adjusting 20
 - bad 24
 - based on dependency injection 216
 - based on
 - Model-View-Presenter 298
 - based on need to know 407
 - based on Passive View 288
 - based on Supervising Controller 285
 - changing in controlled manner 80–89
 - decision 20, 54, 118, 162, 197
 - designing last 8, 15
 - driving design with tests 17
 - driving with executable tests 4
 - encouraging good 4
 - evaluating by trying to use 17
 - for plot map component 297
 - for testability 196
 - guidelines for 118
 - improving 9
 - improving with refactoring 25
 - incremental 8, 15
 - keeping clean 77
 - making decision 16
 - of fixtures 109
 - opportunity to make decisions 76
 - pattern 100, 154–155, 196, 198–199, 205, 255, 280, 284, 356
 - predicting problems with 25
 - principles of good 246
 - simplest we can think of 107
 - tips for simplifying file access 246
 - up-front 21
 - working around bad 143
- design principle 156
- design-code-test 44
- destabilization 464
- detail 237, 268, 456
 - diving into 101
 - implementing detail first
 - See* depth-first
 - in user stories 325
- deterministic execution
 - See* predictability
- developer
 - burnout 346
 - collaborating with test engineers 353
 - dedicating tester to 350
 - gap between customer and 360
 - splitting user story 341
- developer test 199
- development
 - incremental 19, 24, 33
- development environment 457
- DI *See* dependency injection
- DICEE 455–456
- diff 246
- directory service 29

directory structure 188, 190, 245–246
 discipline 24, 60, 436–437
 discussion 462
 dispatchEvent method 315
 distributed system 412–413, 421
 automating 415
 distributed test farm 427
 complexity 428
 many-to-many pattern for scaling 427
 many-to-many-more pattern for scaling 427
 one-to-many pattern for scaling 427
 scaling 428
 divide and conquer 167, 275, 356
 Document Object Model 182
 documentation 34, 78, 457
 DoFixture 384–388, 390
 delegating to child fixtures 385–388
 ensure action 388
 example of 385–386
 mapping actions to method calls 387
 splitting tables 385
 writing tests in natural language 384–385
 DOM *See* Document Object Model
 domain 17, 275, 451, 454
 domain class 433
 domain language 331
 domain logic 155, 283
 exposing to test interface 407
 not mixing with presentation 399
 domain model 221
 domain object 135, 198, 211, 223, 244, 288
 persisting 433
 domain of influence 442
 domain-specific language 341, 361
 creating 407
 done
 definition of 351
 double rendering
 testing for 73, 76

drawing
 lines 312
 dump file 433
 duplication 6, 8, 27, 60, 64, 106, 166–167, 243
 avoiding with fixtures 109
 between production and test code 258
 creating with copy-paste 144
 in Fit fixtures 392
 in test document 368
 lack of 109, 134
 of configuration data 227
 of data 202
 of test data 240, 432
 reducing with fuzzy matching and named queries 219
 removing from tests 82–84
 spotting 106
 dynamic mock object
 See mock object
 dynamics 441
 of change 436

E

early success 458
 EasyMock 112, 116–117, 132, 160, 200–201, 203, 209, 215–216
 and custom matchers 209–210
 custom argument matching with 209–210
 overriding default behavior of equals() 210
 Eclipse 54, 295
 effort 102, 110
 of creating pixel-precise tests 173
 effort estimate 326
 EJB 197
 embedded database 197, 220, 224–225, 229–230
 embedded database
 See also database
 embedded software 358
 emerging design 451
 Emery, Dale 452
 EMMA 40
 enabling point 123
 enabling point *See also* seam
 enabling value 326
 encapsulation 158, 206, 283, 291
 of complex verification logic 130
 of teardown logic 136
 of test logic 137
 end method 79
 end-to-end 220
 end-to-end test 356
 ensure action 388
 enter action 380
 Enterprise Java *See* Java EE
 entity object 132
 EntityManager 197
 environment 200, 220–221, 245, 438–439, 441, 444, 451, 453, 459, 461
 setting up 224, 447
 environment variable 123
 equals method
 and hashCode 93
 overriding default behavior with EasyMock 210
 EqualsTester class 93
 equipment
 availability of 358
 ERP 414
 error handling
 See exception handling
 error situation 275
 maximizing chances of 262
 testing for 103
 error situation space 260
 evaluate method 94
 evangelism 461
 evangelist 455–456, 458–459
 evangelization
 455, 457–458, 465
 event 266, 271, 440, 458
 handling 284
 listening for 292
 triggering with button 304
 event object 284
 event queue 292, 295, 405
 evolutionary design
 20, 22–24, 44
 Exactor 37, 361

example

- expressing requirement
 - through 35
 - of desired behavior
 - See specification by example
- exception 96, 200, 204, 219, 244
 - catching from event dispatch thread 297
 - causing for testing
 - purposes 114
 - testing for 66, 70
 - throwing 67
 - exception handling 45, 50, 217–218, 416
 - adding to test 66–71
 - exception message 70
 - exception path 103
 - Exchanger class 276
 - executable acceptance test 366
 - executable specification 50, 421
 - executable test 10–11, 34, 37–38, 156, 331, 335, 337, 345, 365, 398
 - execute method 383
 - execution
 - asynchronous, verifying 271–274
 - execution path 9
 - execution sequence 266
 - Executors class 271
 - Exoftware 361
 - expectation 112–113, 201, 219
 - for test double 110
 - on mock object 117
 - experience
 - shared 451
 - experiment 451, 461
 - exploration
 - exploring the uncertain 102
 - exploratory testing 463
 - expression
 - of acceptance test 373
 - extension point 382
 - external interface 355
 - external pressure 436, 440, 465
 - external resource 29
 - external threat 462
 - Extra Constructor pattern 142
 - extracting a method
 - 68–69, 88–89, 252
 - Extreme Programming 7, 31, 76, 360

F

- façade 212, 247
- factory method
 - overriding 306
- Factory Method pattern 127
- failure 29, 439, 442, 458, 463
- faith
 - lack of 173
- fake 100–101, 113, 115–116, 118, 120, 140, 148
 - faking data in tests 196
 - faking JDBC connection 200
 - faking method 121
 - faking persistence layer 200
- fake implementation
 - 160, 163–164, 167, 186, 253
 - for java.sql.ResultSet 201
- faking it 104–105, 148
- false agreement 445
- false negative 266
- familiarity 451
- fear 447–448, 452
 - overcoming 451
- feasibility 452
- Feathers, Michael 122–123, 139, 144, 199, 245, 287
- feature creep 57
- feedback 29, 34, 179, 437, 441, 460
 - from tests 29
 - from writing test 17
 - reducing risks and costs with 32
 - speed of 357
- field
 - transient 211
- file access *See* file system
- file system 29, 136, 200, 245
 - abstracting away 410
 - access 245–247
 - in-memory 410
 - persisting domain objects on 198
 - practices for testable access 246–247
 - reading data set from 242
 - reading Velocity template from 180 180
 - virtual 247
- FileRunner 391
- filter-mapping 187
- find method 79
- finder method 213, 216, 236
- first test 243, 259
 - for Hibernate-based data access object 226
 - for JavaServer Page 175, 177
 - for Wicket application 188
- Fit 37, 360, 362, 365, 381, 387, 413
 - ActionFixture 379–382
 - add-ons 384
 - built-in fixture 365
 - built-in fixture classes 372
 - ColumnFixture 373–376
 - executing tests 390–391
 - fixture lifecycle 383
 - fixture table 369–371
 - fixtures 371–372
 - fixtures, built-in 372–384
 - fixtures, built-in, extending 382–384
 - for acceptance TDD 366–369
 - introduction 365–372
 - resulting of running test document 375
 - RowFixture 376–379
 - rules for translating tables 369
 - running with Ant build script 393
 - template for test documents 371
 - test document 369–371
 - test document, location of 391
 - test runner 391
 - testing during automated build 392–394
 - tests, executing 390–394
 - tests, executing with single test document 391
 - tests, in folder structure 391
- fit.FileRunner class 391
- FitLibrary 365, 384, 388, 390
 - child fixtures, delegating to 385–388
 - DoFixture 384–388
 - FolderRunner class 392
 - SetUpFixture 388–390
 - tests, writing in natural language 384–385

- FitNesse 37, 360, 391
 - fixture 64, 100, 108–112, 234, 371–372
 - allowing for focused tests 109
 - built-in 372, 379, 382, 384, 389
 - built-in, extending 382–384
 - child, delegating to 385–388
 - clean slate fixture 109
 - creating objects in 132
 - custom 372, 382, 387
 - flow-style 384
 - getting rid of 132
 - in unit testing 371
 - making assumptions
 - about 128
 - not being in control of 129
 - patterns for creating 136
 - patterns of 132
 - removing duplication
 - with 109
 - setting up and tearing down 149
 - transactional 234–235
 - fixture class 365–366, 368, 371
 - built-in 372
 - example of 404, 408
 - for jWebUnit 402
 - implementing 368
 - setup 389
 - fixture library 372
 - fixture object 126
 - created by setup method 136
 - exercising functionality
 - on 128
 - returning from
 - DoFixture 385
 - fixture table 365, 368–371, 380
 - backing code 371
 - fixure
 - custom, example 383
 - flexibility 154, 190, 222
 - of design 119–120
 - flow-style test 337
 - focused test 48, 109, 245
 - FolderRunner class 392–393
 - setup/teardown support 392
 - formal role 441
 - Fowler, Martin 24, 29, 39, 89, 106, 110, 283–284
 - frame of reference 450–452
 - framework 101
 - component-based,
 - test-driving 184–193
 - component-based,
 - typical 185–186
 - Framework for Integrated Tests
 - See Fit
 - FTP 415
 - client 416
 - task 393
 - functional requirement 358
 - functional testing 186
 - functionality 16, 46, 55, 221, 243, 261
 - accessing 352
 - adding 81
 - breaking existing 28
 - completing throughout
 - iteration 343
 - defining 10
 - implementing 345
 - implementing with Spring
 - MVC Controller 168
 - inheriting 25
 - inheriting from
 - superclass 119
 - islands of 119
 - missing 16, 341, 343
 - on domain-level 401
 - permission to add 189
 - reusing through
 - inheritance 25
 - test-driving into web page 190
 - testing through real
 - interface 401
 - time-dependent 114
 - unpredictable
 - functionality 250
 - valued by customer 341
 - wiring behind widget 302
 - Future interface 277
 - FutureTask class 277
 - fuzzy matching 130
- G**
-
- Gamma, Erich 205
 - generalization 104–105, 447
 - gesture
 - associating with coordinates in
 - view component 314–319
 - getDataSet(String) method 241
 - getDescription method 295
 - getJdbcTemplate method 208
 - getWebRoot method 174
 - Glassfish 174
 - goal 437–439
 - conflicting 353
 - of acceptance test-driven
 - development 32
 - of coaching 462
 - of making test pass 104
 - shared 460
 - supporting change with 449
 - gold plating 353, 456
 - good test
 - definition of 47
 - granularity
 - of tests 17
 - graphical output
 - testing for 310
 - graphical user interface
 - 155, 412
 - evaluating quality of 407
 - testing 356
 - graphics
 - custom 304
 - custom, drawing in view
 - component 304–314
 - testing for graphical
 - output 310
 - Graphics class 310, 312, 314
 - green bar 61, 107, 285
 - unexpected 53
 - green field project 145
 - group 440, 459, 464
 - developing solution as 461
 - in Java regular expression 78
 - group programming 451
 - groupCount method 78
 - GSSBase 93
 - JUnit extensions 93
 - Guard Assertion 128, 137
 - GUI *See* graphical user interface
 - guideline
 - for good tests 47
 - for test-driving 106

H

-
- happy path 45, 101, 103
 - hard coding 56–57, 60, 104–105, 219–220
 - returning hard-coded data from DAO 200
 - hashCode method 93
 - Hatcher, Erik 476
 - header row
 - of ColumnFixture 373–374
 - of fixture tables 368
 - of SetUpFixture 389
 - Heisenbug 260
 - helper class
 - moving creation methods into 134
 - heuristic 101
 - for good tests 47
 - Hibernate 197, 199, 205, 211–212, 214, 216–219, 221–223, 225–227, 229–230, 234–236, 433
 - API, essentials 212–213
 - automatically exporting database schema 230
 - configuring for testing 227–229
 - DAO, implementing 216–217
 - DAO, testing 213–216
 - DAO, testing for exceptions 217
 - populating data 236–237
 - Query interface 212
 - Session interface 212
 - SessionFactory instance, obtaining 213
 - SessionFactory interface 212
 - Hibernate Query Language 212, 214
 - hierarchy
 - of Java Swing application 290
 - higher priority syndrome 447
 - Hillenius, Eelco 193
 - hook 212, 262
 - exposing from application 418
 - for exposing system internals 418–419
 - HSQLDB 223–225, 229
 - in-memory database, configuring 229–230
 - HTML 37, 181, 184–185, 187, 189–192, 360–361
 - parsing and interpreting 422
 - rendering JavaServer Pages into 174
 - HtmlTestCase 174
 - HTTP 156, 174–175, 251, 413, 421
 - faking traffic 422
 - session 421
 - writing acceptance tests 333
 - HTTP header 158–159
 - HTTP request 157–158
 - HTTP session 162
 - HttpServletRequest
 - interface 158, 160, 250–251
 - HttpServletResponse
 - interface 158, 160
 - HttpUnit 178, 184
 - Hugo, Victor 115
 - Humble Dialog Box 284, 287
 - Hunt, Andy 156
-
- I**
 - I/O component 276
 - IDataSet 241
 - IDE 50, 53, 107
 - identifier 223, 227
 - identity 222
 - identity column 223
 - identity generator 223
 - impedance mismatch 406
 - implementation
 - alternative 409, 425
 - anonymous 316
 - fake 357
 - postponing 80
 - pushing with tests 56
 - replacing 120
 - implementation detail 431
 - in-container testing 179, 186
 - increment *See* incremental development
 - incremental DDL script 231
 - incremental design 20, 44, 451
 - independent test 197
 - indicator
 - for recognizing resistance 465
 - of completion 31
 - of having reservations 445
 - indirection 156, 269
 - individual 448, 457, 462, 464
 - addressing about resistance 449
 - inflection point 146–147
 - influence 441–442, 453, 459
 - domain of 441
 - on ability to change 436
 - to GUI development 283
 - information 33, 39, 459
 - flow of 459
 - from spike 80
 - loss of 33
 - revealing new 102
 - to help in problem solving 70
 - informed decision 220
 - infrastructure 174, 178, 220
 - for integration testing 225
 - inheritance 25–26, 119, 243, 431
 - inheritance hierarchy 119
 - inherited setup 431
 - initializer block 108
 - in-memory database
 - See* embedded database
 - in-memory file system *See* Jakarta Commons VFS
 - input 119, 137
 - external input to user interface 282
 - handling malformed 103
 - responding to 113
 - inquiry 462
 - instantiation 121, 138, 142
 - insulation 274
 - integrated development environment 440
 - integration 23, 451, 454
 - continuous 353
 - of developers' work 38
 - integration test 136–137, 179, 213, 220–226, 229–230, 233–235, 243–244
 - differences between production and testing 220
 - example 225–235
 - Hibernate 226–230

- integration test (*continued*)
 - in TDD cycle 243–244
 - introduction 220–222
 - populating data for 235–242
 - populating data with DbUnit 237
 - populating data with Hibernate 236–237
 - production code 233–234
 - selecting database for 222–225
 - test-driving with, vs. unit tests 243
 - what may be missed 221
 - writing before code 219–225
 - writing with Java 243
 - integration testing 220, 225, 242
 - separate configuration for 227
 - integrity
 - exhibiting 440, 465
 - intent
 - deducing from test 128
 - documenting with tests 93, 146
 - expressing with code 83, 175
 - of tests 110
 - preserving in acceptance test 332
 - interaction
 - between threads in concurrent programs 260
 - simulating 283
 - Swing GUI 280
 - testing for 111, 282
 - testing for
 - See* interaction-based testing
 - verifying 305
 - when to refactor 112
 - with application 282
 - Interaction Assertion 131–132
 - interaction sequence 358
 - interaction-based test 355
 - interaction-based testing 100, 111, 113, 116, 118, 132, 149
 - example of 112
 - interaction-oriented test 373
 - interface 247, 252, 269
 - alternative 411
 - avoiding turbulence 399
 - between components 22
 - closed 414
 - default implementation 308
 - designing behavior of 90
 - encapsulating file access behind 246
 - example of DAO interface 198
 - faking with test doubles 184
 - machine-to-machine, automating 414
 - mocking hierarchy of 156
 - of Java Servlet API 158
 - refactoring code behind public 27
 - too big 112
 - toward JspTest 174
 - turbulence 399
 - internal state
 - of fixture objects 109
 - internals
 - side effects of exposing 407
 - Internet Explorer 360, 424
 - interpretation 34–35, 79, 450
 - interprocess
 - communication 418
 - interruption 457, 460
 - inter-thread
 - communication 275
 - intervention 454, 457, 459, 461, 464–465
 - Intimate Inner Class pattern 140–141
 - intimate test 330
 - introspection 462
 - intuition 28, 34–35, 101, 103, 163
 - invariant 275
 - inversion of control
 - container 127
 - IoC *See* inversion of control
 - isolation 47, 118, 237, 246–247, 271
 - of dependencies 122, 124
 - of sequential logic from multi-threading logic 259
 - iteration 19, 24, 339, 440
 - acceptance TDD in 343–348
 - adjusting scope 347
 - goal for 343–344
 - length 346
 - planning for 344
 - within iteration 341
 - iteration plan 19
- ## J
-
- Jakarta Cactus 179
 - Jakarta Commons VFS 247
 - Jakarta Velocity 154
 - Jasper 174, 179
 - Java 4, 83, 119, 137–138, 160, 172, 182, 185, 187, 199, 205, 210–211, 225, 236, 239, 360–361, 366
 - concurrent programming in 259, 262, 268, 271
 - dynamic mock object libraries for 112
 - executing SQL scripts from 230
 - launching threads in 268
 - obtaining current time in 252
 - regular expression API of 76
 - writing integration tests with 243
 - Java 5
 - variable-length argument lists 83
 - Java Database Connectivity 110
 - Java EE 155–156, 160, 168, 178, 184, 196, 198, 205, 250, 422, 430
 - headaches of 154
 - Java Servlet 17, 154, 156, 163, 167, 169, 174, 178, 184, 214
 - example of 157
 - faking request and response for 158–159
 - first test for 160
 - Java Swing 280–281, 283, 308, 319, 417, 419–420, 459
 - and MVC 283
 - component 418
 - component size 308
 - component, locating 292, 297
 - custom component 281, 319
 - interaction, what to test 282

- Java Swing (*continued*)
 - internal plumbing and utilities, what to test 281
 - relative positions of components 292
 - rendering and layout, what to test 281–282
 - TDD-friendly tools 292–297
 - test-driving view
 - components 297–319
 - tools for testing view
 - components 290–297
 - UI, what to test 280–282
 - view component, adding and operating standard widgets 300–304
 - view component, associating gestures with
 - coordinates 314–319
 - view component, drawing
 - custom graphics 304–314
 - view component, layout out design 298–300
 - why testing tools are needed 290–291
 - Java Swing API 281, 312
 - layout of GUI
 - components 281
 - rendering GUI
 - components 281
 - Java Virtual Machine 108, 199, 229, 268
 - Java WebStart 420
 - java.awt.Canvas class 309
 - java.awt.Component class 308–309, 315
 - java.awt.image.BufferedImage class 311
 - java.util.Calendar class 252
 - java.util.concurrent
 - package 268, 275–276
 - java.util.Date class 252
 - java.util.regex API 77
 - java.util.regex.Matcher class 78
 - Javadoc 76, 78
 - JavaScript 187, 360, 421–423
 - JavaServer Faces
 - 156, 185–186, 193
 - interfaces, fake implementations of 186
 - JavaServer Pages
 - 154, 174, 176–179
 - test-driving with JspTest 174–179
 - @javax.persistence.Transient 211
 - javax.servlet.http.HttpServlet 157
 - JButtonOperator 294
 - JDBC 197, 206, 211, 220, 222, 229, 238
 - code, writing code to pass test 204–205
 - code, writing test for 202–203
 - exposing access to 212
 - tyranny of 200
 - JDBC API
 - tyranny of 200–205
 - JDBC driver 197, 220, 224, 229
 - not testing 221
 - JdbcDaoSupport class 206, 208
 - implementing DAO 210
 - JdbcTemplate class 199, 205, 208, 210–211
 - test-driving DAO 208–209
 - Jeffries, Ron 36
 - JellyTools 292
 - Jemmy 292, 294–295, 319, 418
 - ComponentChooser interface 294
 - Jemmy API 293
 - Jetty 174, 178
 - JFCUnit 418
 - JFrame 294, 297
 - JFrameOperator 294
 - jMock 112
 - JSF *See* JavaServer Faces
 - JSP engine *See* JavaServer Pages
 - JSP *See* JavaServer Pages
 - JspTest 174, 176, 178–179
 - alternatives to 178
 - example of 174
 - test-driving JSPs 174–179
 - JUnit 36, 41, 50, 64, 67, 93, 136–138, 141, 213, 272, 280, 413
 - extension, EqualsTester method 93
 - <junit/> task 478
 - JUnit Yahoo! Group 272
 - JUnit-Addons 141
 - <junitreport/> task 479
 - jWatir 424
 - jWebUnit 402, 422–423
-
- K**
-
- Kawasaki, Guy 455
 - Kerievsky, Joshua 26
 - keystroke
 - simulating 292
 - keyword 379, 388
 - transient 211
 - known state
 - bringing system into 430
 - moving target system into 431
 - Kuchana, Partha 26
-
- L**
-
- language
 - of unambiguous tests 47
 - shared 31, 33, 108
 - understanding 443
 - Larman, Craig 7
 - latch 262, 270–271, 276–277
 - Laughing Panda 141
 - Law of Demeter 156
 - layer
 - separating with DAO pattern 198–199
 - layout
 - Swing GUI 280
 - testing for 281
 - LDAP 163
 - leading change 441–442, 466
 - learning test 77–80, 221
 - example of 78–79
 - least qualified person 336
 - legacy code 100, 118, 127, 246, 439, 459
 - analyzing change 146–147
 - definition for 144
 - preparing for change 147
 - process for working with 145
 - test-driving change 148
 - test-driving development 146
 - tweaking through Reflection API 141
 - working with 144–148

- level of abstraction 69, 87, 112, 185
 - library
 - dynamic mock object
 - libraries 112
 - license to fail 458
 - lifecycle 135
 - of Fit fixtures 383
 - lightweight database 223–224
 - line of sight 459
 - link seam 123
 - list 18
 - from RowFixture 379
 - of acceptance tests 342
 - of principles for test-driving 106
 - of tests 48–50, 60, 62, 71, 86, 102
 - of Very Important Practices 101
 - listener
 - fake 316
 - notifying 285
 - listening 449
 - Lister, Timothy 347
 - literal string 57, 167
 - load
 - decreasing 346
 - load-balancing 220
 - log
 - test-driving 250–252
 - testing output with faked system time 256–259
 - logging 180, 250
 - logic 58–59
 - domain 398
 - duplication of 27
 - encapsulation of 130
 - enforcing implementation of 162
 - isolated blocks of 10
 - replacing 76, 86
 - responding to resistance with 448
 - separating 23
 - testing directly 412
 - looping construct 179
 - Loughran, Steve 476
 - lowering the bar 457–458, 466
 - low-hanging fruit 81, 101, 103, 280, 309, 456
- M**
-
- Mac OS X 424
 - machine-to-machine 356
 - macro 123
 - mailing list 141
 - main method 199
 - maintainability 6–7, 24, 44, 118–120, 211, 219, 224, 230, 240, 245
 - isolating dependencies for 123
 - lack of 5
 - of acceptance tests 356
 - maintenance 58
 - nightmare 6
 - Malks, Dan 198
 - management 457
 - managing upward 442
 - Mann, Kito 193
 - Manns, Mary Lynn 453
 - manual analysis 260
 - manual intervention 359
 - manual testing 28, 196, 352
 - mapping 169
 - acceptance tests to user-accessible functionality 403
 - DoFixture actions to method calls 387
 - fixture tables to method calls 373
 - objects to database tables 211
 - table data to fixture class 374–376
 - table rows to objects 208
 - markup language
 - 174, 191–192, 360
 - matcher 297
 - custom, and EasyMock 209–210
 - matching
 - fuzzy, reducing duplication with 219
 - Maven *See* Apache Maven
 - McCabe
 - complexity metric 68
 - McKoi database 224
 - memory
 - long-term memory 22
 - short-term 84
 - working memory 21
 - mentoring 441
 - Meszaros, Gerard 115, 128
 - metadata 247
 - method
 - consistency of abstraction level 69–70
 - extracting 88–89
 - keeping in balance 69–70
 - size 313
 - static, avoiding 120–122
 - methodology 455
 - Microsoft Excel 360, 362
 - Microsoft Office
 - using with Fit 366
 - Microsoft Outlook 417
 - Microsoft Windows 424
 - Microsoft Word 360
 - migration 86
 - of database schemas 231–232
 - mismatch
 - between change and incentives 442
 - mock object 100, 112, 115, 118, 137, 148, 161, 201, 203, 207, 209–210, 214, 218–219, 221
 - creating for Hibernate interfaces 213
 - example 116–118
 - mock object framework 160
 - mock object library 132
 - MockMultiResultSet 201–202
 - MockObjects library 201
 - MockObjects.com 203
 - model 283
 - in Model-View-Controller 155
 - model/view separation
 - See* Model-View-Controller
 - ModelAndView 168–169
 - Model-View-Controller 154, 156, 163, 168, 172, 184, 283
 - controller 156–172
 - creating view test-first 173–184
 - in web applications 154–156
 - Model-View-Presenter 284, 286–287, 289, 298, 319
 - modular design 163
 - mouse click
 - capturing 314
 - simulating 316
 - simulating and responding to 315–317

mouse gesture
 simulating 292
 MouseAdapter 316
 Mozilla Firefox 360, 417, 424
 Mugridge, Rick 365, 384,
 392, 394
 multithreaded code
 test-driving 259–275
 what to test for 260–261
 multithreaded programming *See*
 concurrent programming
 multithreading 86, 250, 260,
 262, 268, 275, 277, 417
 Murphys Law 12
 mutable
 counter object 307
 MVC *See* Model-View-Controller
 MVP *See* Model-View-Presenter
 MySQL 222–223

N

naming convention 221
 NanoContainer 127
 natural language 384
 need
 anticipating 22
 changing 6
 failing to meet 6, 31
 nested inner class 141
 .NET 422
 NetBeans 292
 network
 tests communicating over 200
 network socket 414
 networked system
See distributed system
 networking 22, 116, 225,
 245, 269
 networking library 414
 nondeterministic 72, 114
 nonfunctional requirement 358

O

object
 creation 133
 transient 225, 227
 under test 111, 115, 139
 under test, configuring 126
 object database 198

object factory 135
 object graph 108, 119, 216
 providing access to 135
 Object Mother 134–135, 137
 object seam 123
 object seam *See also* seam
 objection 455
 ObjectMentor 360
 object-oriented
 programming 119, 123
 object-relational mapping 205,
 211, 221–222, 236–237, 433
 assumptions about 221
 Observer pattern 283–285
 obvious implementation
 104–105, 148, 218
 openness 442, 448, 465
 optimistic locking 30
 optimization 212, 272
 Oracle 220, 222
 org.apache.wicket.markup.html.
 WebPage 187
 org.apache.wicket.protocol.http.
 WebApplication 187
 organizational change 465
 ORM *See* object-relational
 mapping
 osmotic communication 459
 output
 actual, in Fit 376
 expected, in Fit 376
 over-engineering 4, 18, 57
 overhead
 of thread startup
 262, 264–265
 semantic 416
 overriding 121–122,
 124, 126, 252
 default behavior of EasyMock
 argument matching 210
 factory method 306
 getter method 167
 ownership 144, 441,
 444, 451, 460

P

pair programming 336, 343,
 441, 444–445, 460
 Pamie 424
 Parameterized Creation
 Method 132–134, 136

parameterized query 219
 Parameterized Test
 pattern 137–138
 Parkinsons Law 351
 parse method 85
 parseSegments method 91–92
 Pascarello, Eric 423
 Passive View pattern
 284, 287–289, 297, 319
 pattern 26, 35–36, 38, 66, 71,
 105, 142, 441
 assertion 128–132
 Automated Teardown
 136–137
 Extra Constructor 142–143
 fixture 132–137
 for generating test data 433
 for testable UI code 283–290
 Intimate Inner Class 140
 Model-View-Presenter 284
 Object Mother 134–135
 occurrences of in data 78
 of component-based
 frameworks 185
 of writing unit tests 127
 Parameterized Fixture
 Method 132–134
 Parameterized Test 137–139
 Passive View 287
 Privileged Access 141–142
 Self-Shunt 139
 Supervising Controller
 284–287
 test 137–144
 testing for 313
 Test-Specific Subclass
 143–144
 unit-testing 127
 pattern recognition 102, 414
 peer pressure 438, 446
 peer review 441
 peer-to-peer application 276
 peopleware 347
 perceived threat 448
 performance 22, 71–72, 103
 testing for 72–73
 trade-off with coupling 287
 Perl 424, 430
 permission 275
 permit 275
 Perrotta, Paolo 252

- persistence 196, 433, 453, 466
 - cheating on 427
 - persistence framework
 - 197, 199–200, 205, 220, 222–226, 244
 - downsides to relying on 237
 - faking results from 235
 - persistence layer 197, 223
 - persistence logic 197, 199, 220, 222
 - persistence technology 198–199
 - persona 135
 - personal adoption 440–441
 - persuasion 448, 453
 - phase
 - of changing legacy code 146
 - PicoContainer 127
 - pilot 448
 - pixel-precise test 311
 - planning 344
 - continuous *See* continuous planning
 - for iteration 344
 - pre-iteration planning
 - See* continuous planning
 - planning meeting
 - during acceptance TDD 335
 - planning tool 327
 - plot map component
 - adding and operating
 - standard widgets 300–304
 - associating gestures with
 - coordinates 314–319
 - drawing custom
 - graphics 304–314
 - example 297–319
 - layout out design 298–300
 - plot point
 - adding 300
 - drawing 300
 - removing 300
 - plumbing 156, 174, 199–200, 208, 294
 - for Abbot 300
 - for integration testing 243
 - for Java Swing
 - applications 280
 - for Jemmy library 295
 - for testing blocking
 - methods 268
 - for testing Hibernate-based
 - data access object 213–214
 - for testing JavaServer
 - Pages 177, 179
 - for testing Velocity
 - templates 180–181
 - PMD *See also* static analysis
 - point solution 459
 - polling 272
 - polymorphism 89, 94
 - pooling 213
 - portability 235
 - positive functionality 103
 - Potel, Mike 284
 - potential 447
 - power of imagination 48
 - Pragmatic Programmers 89, 156
 - “Tell, Don’t Ask” principle 89
 - precision
 - of testing exactly what
 - we want 409
 - premature optimization 57
 - PreparedStatement 203
 - preprocessing seam 123
 - presentation 281, 283, 460
 - in web applications 421
 - not mixing with domain
 - logic 399
 - of view components 282
 - separating from application
 - logic 416
 - presentation layer 280
 - presenter 284
 - responsibility 286–287, 289
 - simulating 291
 - presenter class 285
 - Presenter pattern 356
 - presenter *See also*
 - Model-View-Presenter
 - press action 380
 - primary key 114
 - primary key generator 223
 - Primitive Obsession code
 - smell 89
 - principle
 - of good design 109, 118
 - of test-driving 101
 - Tell, Don’t Ask 89
 - PrintWriter 158
 - PrivateAccessor 141
 - Privileged Access 141
 - PrivilegedAccessor class 141
 - problem 7
 - quality problem 5
 - simplifying complex problem
 - to smaller problems 21
 - solving wrong problem 5
 - source of 9
 - problem space 196
 - production code 44, 47, 86, 218, 236, 243, 257–258
 - adding just enough 55
 - controlling execution of 262
 - injecting dependencies
 - into 124
 - invoking from test 132
 - obtaining system time 255
 - sharing code between test
 - code 167
 - skeleton of 53
 - squeezing out hard-coding
 - from 104
 - production database 229, 231
 - production environment
 - 12, 221–222
 - productivity 11, 19, 32, 44, 107, 224, 439, 448, 455
 - hampering 433
 - initial drop in 457
 - negative effect on 458
 - professionalism 449
 - programming by intention
 - 48, 50–51, 53, 87, 90, 167, 175, 226–227
 - programming language 36, 119, 123, 221, 371
 - expressiveness 406
 - programming library
 - strategy for testing 412
 - progress
 - delaying 102
 - in terms of produced
 - software 46
 - tests as measurable
 - progress 22
 - three ways to make 104
 - with least effort 47
 - project management 351
 - project manager 441, 443, 458
 - promise
 - of future conversation 325

prompt
 in command-line interface 415
 propaganda 439
 property file 127
 prose 35
 prototyping 76–77
 proximity 459
 psychology 448
 Punke, Stephanie 135
 purpose
 of tests 28
 Python 360–361, 366, 424

Q

quality 5, 443, 450, 454, 458
 external 7
 improving 10
 in terms of defects 9
 internal 7, 9, 14, 281
 of code 8
 of design 22
 of personal software process 16
 perceived quality *See* external quality
 sacrificing 110, 346
 technical 10
 quality assurance 8, 32, 397, 463
 query
 named, reducing duplication with 219
 Query interface 212–213, 215–216, 218
 query language 212, 221
 query method
 of RowFixture 376, 379

R

Rainsberger, J. B. 40, 66, 112–113, 178, 410
 raster
 capturing bitmap into 312
 rationale 451
 behind TDD 16
 for change 437
 read operation 226, 234–235
 readability 14, 109, 167

read-only 282
 reality TV show
 implementing with CyclicBarriers 276
 record-and-replay 37
 recording
 expected interactions 117
 recording mode
 of EasyMock-generated mock objects 117
 recursion 291
 red bar 53, 55
 red bar *See also* green bar
 red-green-refactor 16
 redirection 189
 of HTTP request 160–161, 165
 redundancy 64–65
 refactoring 8, 15, 18, 24, 28, 38, 44, 60, 62–66, 68–69, 81–82, 94, 98, 148, 243, 331, 439
 ability to perform 355
 as transformation 25
 automated 107
 do not skip 106
 extract method 69
 lack of 106
 potential, in test code 64
 proceeding with 90
 removing duplication from acceptance tests 431
 test document 368
 tests to be more expressive 413
 toward cohesive creation methods 135
 toward creation methods 134
 toward smaller methods 68–69
 toward system time abstraction 252
 when tests become complex 112
 reflection 462
 in Fit 376
 Reflection API 141, 291
 registerParseDelegate method 389
 regression 8, 28, 355
 regression test 29–30

regular expression 60, 73, 76–77, 80, 259, 412
 enumerating matches for 85
 finding matches in 80
 groups in 78, 80
 relational database 163, 196, 198–200, 220, 222
 relational database
See also database
 relationship
 between TDD and acceptance TDD 31
 remote interface 419
 Remote Method Invocation 419
 remoting 419, 422
 rendering 45, 48, 72
 component on buffer 311
 JavaServer Pages 175
 Swing GUI 280
 Velocity templates 179, 181–182
 rendezvous point 262, 265
 replay mode
 of EasyMock-generated mock objects 117, 209–210, 215
 request parameter 158–161, 167, 177, 185
 request-response 154
 requirement 33, 45, 449
 being oblivious to 18
 decomposing 45–46
 expressing 18
 expressing with user stories 325
 for customer involvement 331
 for template engine 49
 format for expressing 31
 fulfilling 50, 398
 misunderstanding 14
 satisfying 45
 splitting 48
 translating into test 10, 46, 49
 requirement bug 331
 requirement format 329
 requirement specification 32
 requirements document 10
 requirements management 325, 327, 334
 reserved word 223
 reset method 383

- resistance 443, 445, 456, 464
 - addressing 448
 - defense against 448
 - feeding 447–448
 - fighting 444–454
 - forms of 447
 - labeling 450
 - overcoming 449–453
 - picking battles 453–454
 - recognizing 444–448
 - responding to 444, 448, 452, 463, 465
 - source of 447–448, 451–452, 454, 464
 - strategies for overcoming 449
 - three standard responses to 448–449
 - resource
 - shared 261, 276
 - respect 442
 - responding defensively 448
 - responsibility 26, 45, 113, 185, 212, 237–238, 446, 463
 - division of 6, 98, 119, 206
 - duplication of 27
 - for generating unique identifiers 223
 - for triggering teardown logic 136
 - in testing software 350
 - of individual 441
 - result
 - expected, in Fit 376
 - result column
 - expected 374
 - Resulting State Assertion 128–130, 132
 - ResultSet 203
 - RetriedAssert class 272
 - return type
 - of fixture class methods 375
 - reuse 184–185, 198, 268
 - reverse engineering 414
 - reward
 - delayed 465
 - rewrite 27
 - rich client application 283
 - Rising, Linda 453
 - risk 221, 437, 440, 452, 458
 - of brittle tests 403
 - of bypassing UI 407
 - of finding configuration issues during deployment 227
 - of misunderstanding 35
 - of misuse 40
 - of not integration-testing data-access code 221
 - of stakeholder estrangement 403
 - of unmaintainable test code 404
 - reducing through test selection 102
 - tackling by starting implementation from details 102
 - risk management 410
 - RMI *See* Remote Method Invocation
 - rMock 112
 - robustness 45, 103, 129, 218, 257
 - of UI tests 282
 - role 441, 462–464
 - and ability to lead change 441–443
 - roles involved in acceptance test-driven development 324
 - rollback 235
 - of database change 231–232, 234
 - rot
 - keeping code from rotting 24
 - rotting 63
 - RowFixture 372, 376, 379
 - comparing datasets 379
 - example of 378
 - RowMapper interface 206–208, 211
 - implementing 207–208
 - Ruby 340, 361, 366, 424
 - Ruby on Rails 231
 - rule 44, 80
 - of no duplication 167
 - of TDD 15
 - of thumb 103, 109
 - Runnable
 - typed 271
 - runtime environment 109
- ## S
-
- Sadalage, Pramodkumar 244
 - Safari 424
 - SafariWatir 424
 - safety 448, 451, 462, 464
 - Samie 424
 - sanity check
 - mid-iteration 346
 - Satir Change Model 451
 - scalability 220
 - Schuh, Peter 135
 - scope creep 353
 - screencast 457
 - scripting language 37, 187, 333, 340
 - flexibility 361
 - Scrum 351, 458
 - Scrum Master 458
 - seam 122–123, 146, 149
 - link 123
 - manipulating in test 147
 - object 123
 - preprocessing 123
 - section 508 173
 - Segment interface 90–98
 - Selenium 360, 425
 - Self Shunt pattern 139–140, 301
 - self-imposed constraint 447
 - semaphore 275–277
 - counting 275
 - single-permit 276
 - seniority 441
 - sense of achievement 438, 440, 457, 465
 - sense of urgency 437, 440
 - separation
 - of test data from test logic 236
 - of view and controller 283
 - sequential execution 261–262, 264
 - sequential program 259–260
 - service layer 356, 400
 - service method 158
 - Servlet
 - test-driving 167
 - testing 158–160
 - testing for redirections 160–162
 - working with parameters and session 162

- state (*continued*)
 - testing for changes in
 - See state-based testing
 - verifying with fixture class 376
 - state machine 103
 - state of practice 4, 451
 - state-based testing
 - 100, 111–112, 118, 132, 149
 - example of 111
 - static analysis 39
 - static method 118, 120–121, 138, 149
 - avoiding 120–122
 - obtaining dependency with 124
 - wrapping access to 122
 - static reference 258
 - static variable 108, 121
 - stderr 391
 - stench *See* code smell
 - stored procedure 223
 - story card 335
 - story test-driven development
 - See acceptance test-driven development
 - storytelling 325, 452
 - strategy
 - for making tests pass 100, 104
 - for managing test data 196
 - for overcoming resistance 453
 - for selecting next test *See* test, selecting next
 - Strategy pattern 255
 - stream object 246
 - structure 17, 27, 119, 124
 - altering code's 25
 - Struts 156
 - stub 100, 113, 115, 118, 208, 409, 413, 421
 - partially stubbed
 - implementation 144
 - stubbing out JDBC API 197
 - stubbing out persistence logic 197
 - stubbing 81, 305, 357–358
 - subclassing 121
 - subjective evaluation 352
 - SUnit 36
 - Supervising Controller
 - pattern 284–289, 297, 319
 - Swan, Brian 361
 - Swing *See* Java Swing
 - SWT *See* Standard Widget Toolkit
 - synchronization 259, 261–262, 264, 275, 277
 - between threads 274–275
 - of GUI widgets 287
 - view-model 284
 - synchronization lock 276
 - synchronization object 250, 262, 264–265, 270, 275–277
 - barrier 276
 - future 277
 - latch 276
 - semaphore 275–276
 - standard 275–277
 - synchronization point 275–276
 - synchronizer *See* synchronization object
 - syntax 221, 223–224
 - annotation-based 67
 - for expressing acceptance tests 333
 - of executable acceptance tests 339, 341
 - system
 - under test 333, 338, 368, 371, 379
 - under test, connecting to 390, 402, 404, 411, 434
 - under test, exposing
 - alternative interfaces 400
 - under test, frequency of changes to 416
 - under test, interacting with 374
 - under test, restarting 430
 - under test, setting up 426
 - under test, setting up test data for 387
 - system command 199
 - system property 109
 - system testing 451
 - system time 251–252
 - abstract, adopting API 257–258
 - abstracting 252–256
 - faked, testing log output with 256–259
 - System#currentTimeMillis method 252
 - systemic requirement 358
 - SystemTime class 257
 - Systir 362

T

- table-based framework 359, 361
- Taligent, Inc. 284
- Tapestry 156
- target system
 - under test *See* system
- task 45, 53
 - vs. test 46
- taxonomy
 - of test doubles 149
 - of test doubles *See* classification of test doubles
- TDD *See* test-driven development
- team 21, 440, 446, 461–462
 - building trust within 33
 - cross-functional 336
 - integrated project team 32
 - relationships between members of 350
 - size of 343
 - structure 348
- teamwork 460
- teardown 234, 247
- teardown document 392
- teardown method 136, 234, 254, 294
- tech lead 441
- technical architect 441
- technical correctness 334
- technical quality 324, 456
- technical quality *See also* internal quality
- technical user story 326
- technique
 - adopting new 436
- technology
 - annoyances caused by 412
 - bias 400
 - catching up with 10
 - effect on acceptance testing productivity 406
 - familiarizing with 77
 - limitations 400
 - limitations of 400, 410
- Tell, Don't Ask 89, 156
- Telnet 424

- template 457
 - for Fit test documents 371
 - for user stories 325
 - Velocity, test-driving 179–184
- template engine 45, 48, 72, 76, 80, 179
 - basic functionality for 66
 - changing behavior of 97
 - performance of 71
- Template Method pattern 205
- template technology 174
- TemplateParse class 81–89
- temporary directory 247
- temporary file 247
- temporary solution 448
- test
 - adding error handling 66–71
 - alternative strategies for automation 397
 - ambiguity 331, 333
 - as safety net 18, 84, 95
 - atomic 47
 - automated 8
 - automated, problems with 399
 - automation 18, 37, 77, 280, 339, 341–342, 346, 420, 425, 463
 - automation,
 - coordinate-based 418
 - breadth-first 58–63
 - breaking 166
 - brittle 403, 417, 422
 - characterization 147
 - choosing first 48–58
 - collecting result after execution 420
 - complexity 356
 - conciseness 331–332
 - coverage 410
 - dependent 429
 - depth-first 58–63, 80–89
 - design guideline, avoid static and Singleton 120–122
 - design guideline, choose composition over inheritance 119–120
 - design guideline, inject dependencies 124
 - design guideline, isolate dependencies 122–124
 - design guidelines 118–127
 - designing from requirements 45–48
 - duration 425
 - end-to-end scenario 399
 - executable 331, 337
 - execution as part of
 - automated build 420
 - execution speed 357, 403, 417, 422, 425
 - execution speed vs. complexity 428
 - execution through remoting module 420
 - execution, distributing 427
 - execution, speeding up 426
 - expressing requirement 18
 - failing 15, 18, 44–45, 91, 104, 302, 314–315, 318, 341, 368
 - failing acceptance test 341
 - failing intentionally 62
 - failing, as progress 53
 - failing, commenting out 92
 - failing, figuring out cause 236
 - failing, making pass 54–56
 - failing, writing 50–53
 - failing, writing first 50–53
 - failure 224, 297
 - Fit, executing 390–394
 - Fit, executing with single test document 391
 - Fit, placing in folder structure 391
 - flow-style 337, 385
 - focused 12, 109, 130, 134
 - for pixel-precise layout 281–282
 - functional 4
 - granularity 17
 - guidelines for writing 47, 100–107
 - how to make pass 100–107
 - implementation strategies 104–106
 - implementation strategy, faking it 104
 - implementation strategy, obvious implementation 105
 - implementation strategy, triangulation 104
 - implementation, splitting 419
 - integration, differences between production and testing 220
 - integration, example 225–235
 - integration, Hibernate 226–230
 - integration,
 - introduction 220–222
 - integration, populating data for 235–242
 - integration, populating data with DbUnit 237
 - integration, populating data with Hibernate 236–237
 - integration, production code 233–234
 - integration, selecting database for 222–225
 - integration, what may be missed 221
 - integration, writing before code 219–225
 - interaction-based 355
 - intimate 330
 - isolated 47
 - learning 77–80
 - list, creating 49–50
 - list, working from 47
 - maintainability 418
 - naïve 251
 - organizing 100
 - passing 45, 47, 50, 53, 55–58, 81, 104–105, 161, 167, 233, 266
 - passing after refactoring 84
 - passing as quickly as possible 55
 - performance 425
 - properties of good 47
 - redundant, removing 65
 - refactoring 63–66
 - refactoring safely with passing tests 87
 - removing duplication 82–84
 - robustness 406
 - selecting next 100–103, 106
 - selection strategies 101–103
 - selection strategy, details vs. big picture 101
 - selection strategy, happy path vs. error situations 103

- test (*continued*)
 - selection strategy, high value
 - vs. low-hanging fruit 102
 - selection strategy, uncertain
 - vs. familiar 102
 - self-checking 129
 - simplicity 331
 - skipping 34
 - slow 29, 113, 403
 - spotting errors in 13
 - verbosity 356
 - vs. task 46
 - writing 16
 - writing for sequential
 - behavior 260
 - writing in natural
 - language 384
- test case
 - adding to Parameterized
 - Test 138
 - deriving 9
 - documenting 34
 - hierarchy 431
 - reducing complexity 431
- test case class *See* test class
- test class 51, 60, 64, 81, 96, 108, 134, 166, 189, 199, 238–239, 294, 301
 - becoming more complex 139
 - for testing Velocity
 - templates 181
 - helper methods in 86
 - instantiating 138
 - splitting due to lack of
 - cohesion 109
 - turning into Parameterized
 - Test 137
- test configuration
 - for Hibernate 229
- test coverage 9, 34, 39, 148
- test data 135, 225, 236–237
 - defining in external file
 - 235, 237, 240
 - guidelines for managing 432
 - setting up 244
- test database 219, 225, 229, 231, 235, 240
- test definition language 384
- test document 365, 369–371
 - Fit 391
 - output 366
- test double 100, 108, 110, 113, 116, 120, 169, 184, 186, 200–201, 209, 216, 243–245, 275, 284, 301, 306
 - dealing with difficult
 - collaborators by using 100
 - example 113–115
 - for HTTP request and
 - response objects 159
 - for JDBC interfaces 201, 203
 - generating with dynamic
 - mock object library 112
 - implementing state-based 112
 - mock object, example
 - 116–118
 - passing to Spring MVC
 - Controller 170
 - standing in for
 - dependency 110
 - state-based and interaction-based testing 110
 - state-based testing 111
 - stubs, fakes, and mocks
 - 115–116
 - substituting 141
 - substituting dependency
 - with 142
 - testing for interactions 111
 - types of 139
- test engineer
 - as customer *See also* tester
 - ratio of testers to
 - developers 351
 - role of 345
- test execution 29, 47, 53, 62, 109, 234–235, 260, 262, 381
 - cleaning up after 136
 - distributed, bottleneck of 428
 - duration of 72
 - having side effects 257
 - of Fit tests 390, 393
 - parallel 428
 - replacing code for duration
 - of 122
 - results of 428
 - running right test 161
 - running subset of tests 29, 38
 - speed of 110, 113, 115, 118, 197, 200, 220, 235, 243–245, 283, 406
 - speeding up 429–431
 - tearing down after 120
- test fixture *See* fixture
- test harness 29, 244, 300, 355, 438
 - instantiating objects in 119
- test method 51–52, 64, 108–109, 137–138, 181, 214, 234, 240–241, 265, 267, 297, 301, 313
 - cleaning up after 136
 - sharing state between 140
 - size of 203
- test object registry *See* Automated Teardown
- test plan 32
- test point 146
- test report 428
- test result 36, 53, 392
 - collecting 428
 - color-coding of 366
- test runner 36, 430
 - Fit 391–392
 - multithreaded 427
- test setup 246, 431
 - pushing data into
 - database 432
- test suite 29, 38, 44, 97, 224–225, 355
 - organizing 426
 - slow 426
 - speeding up 427, 431
 - splitting 426
- testability 17, 100, 114, 118–121, 167, 170, 189, 196, 200, 246, 454
 - designing for 141
 - enforcing by writing tests
 - before code 118
 - improving by making use of
 - composition 149
 - improving by stubbing out
 - components 409
 - improving through dependency injection 126
 - improving through
 - isolation 124
 - isolating dependencies
 - for 123
 - issue with Singleton 118
 - of file access code 245
 - supporting 283
- testable code *See* testability

- testable design 107, 118, 149, 196
- testable design *See* testability
- testable *See* testability
- test-authoring problem 382
- test-code-refactor 15, 19, 24, 38, 44, 184
- test-code-refactor *See* TDD cycle
- Test-Driven Design *See* test-driven development
- test-driven development 4, 14, 110, 141, 156, 175, 179, 185, 196, 222, 243, 283, 350, 355
 - adopting 40, 436–437, 440, 443, 451
 - and ATDD 341
 - cycle 15
 - cycle, with integration
 - tests 243–244
 - elegance of 456
 - getting others aboard 440–444
 - implementing spikes with 77
 - implementing template engine with 48
 - of concurrent programs 250, 259
 - of Java Swing
 - components 280, 298
 - promoting 439, 443, 451, 455
 - requirements to adopt 436
 - resistance to 453
 - simple rules of 44
 - tools for 36
 - training on 106
- test-driving
 - guideline, don't skip refactoring 106
 - guideline, get to green fast 107
 - guidelines 106–107
 - slow down after mistake 107
 - unit vs. integration tests 243
- tester 28, 31–32
 - as customer 334
 - dedicated 350
 - qualities of professional 398
- Test-First Programming *See* test-driven development
- testing 5, 28
 - backdoor 419
 - black-box 401–403, 418–419
 - context-driven 407
 - cost vs. value 397
 - crawling under the skin 404, 412, 414–415, 419
 - end-to-end 401–402, 416–418, 421, 425
 - essential concepts 108–113
 - exercising the internals 407, 412, 421
 - exploratory 407
 - for interactions 111
 - for patterns 313
 - for special case 62
 - framework 420
 - framework, replacing 413
 - in isolation 354
 - intrusive 400
 - library 420
 - manual 407
 - programming libraries 412
 - programming library, example 413
 - state-based 111
 - state-based and interaction-based 110–113
 - stubbing out irrelevant 409
 - through UI 404
 - white-box 400, 419
- testing algorithm 398
- testing backdoor 411
- testing framework 50, 421
- testing language 356
- testing library 425
- testing phase 5
- testing tool
 - homegrown 362
- tests as specification 34
- Test-Specific Subclass pattern 144
- text field
 - pulling data from 304
- TextTest 361
- thinking out of the box 9
- third-party software 362
- Thomas, Dave 156
- Thread 264, 268–269
- thread 259–261, 264–265, 267, 269, 274–277
 - interrupting 267
 - overhead of starting 262, 264–265
 - starting 268–271
 - starting in factory 268
 - stopping 268–271
 - Swing event dispatch thread 297
 - synchronization
 - between 274–275
 - tests launching multiple 262
- Thread API 268
- thread class
 - custom 269, 271
- thread scheduler 259, 261, 266
- Thread state
 - don't rely on 268
- Thread#getState method 268
- Thread#sleep method 264
- ThreadFactory 271
- ThreadFactory interface 268–269, 271
 - using with custom threads 269–271
- thread-safety 86, 261–262, 264, 266
 - example of testing for 262–264
 - limitations on testing 262
 - maximizing concurrency while testing 264–266
- throughput 5
- time
 - abstraction of system time 252, 254, 258
 - between question and answer 403
 - configuring alternative source for 254
 - current 252, 259
 - depending on 72
 - faking system 252
 - for change 440
 - problem with 250
 - querying for current 251
 - spent in maintaining tests 110
 - spent in writing tests 110
 - stubbing out system clock 413
 - time-dependent
 - functionality 250
 - to correct mistake 30
 - to develop 12
 - to fix defect 9
- time line
 - of an iteration 343

- time-based functionality
 - test-driving 250–259
 - timestamp 251, 257–259
 - format, testing for 258–259
 - test-driving 250–252
 - time-to-market 5
 - trade-off 23, 30, 38
 - training 106, 456, 458–459, 461, 466
 - Transaction 235
 - transaction 218, 227, 233, 235, 266
 - rolling back 430
 - transactional behavior 227, 235
 - transactional fixture 225, 234–235
 - transformation 8, 15, 25, 50, 451, 454, 463
 - applying to code 27
 - applying to system's internal structure 26
 - behavior-preserving 25
 - of requirements into executable tests 34
 - of requirements into tests 35
 - transition 35, 459
 - translation
 - of acceptance test to executable test 339
 - of requirements 46
 - traversal 58
 - tree
 - traversing 101
 - trial 448
 - triangulation 56–57, 104–105, 148
 - turbulence
 - avoiding 399
 - Twill 422
 - typed Runnable 271
 - typo 221
- U**
-
- UI
 - code, patterns for testable 283–290
 - UI component 185
 - UML *See* Unified Modeling Language
 - uncertainty
 - reducing 102
 - understanding 32, 446
 - existing code 26
 - increasing with learning tests 77
 - lack of 443
 - of TDD 44
 - shared 460
 - transferring 34
 - verifying with learning tests 78
 - Unified Modeling Language 128
 - Uniform Resource Identifier 159
 - Uniform Resource Locator 168
 - unit test 35, 47, 108, 110, 114, 127, 137, 146, 156, 158, 174, 213, 221, 243–244, 251, 253, 260, 352
 - accessing network 143
 - controlling time from 256
 - definition of 199, 245
 - expressing expectations with 334
 - first steps in programming 290
 - merging with acceptance test 413
 - running 62
 - test-driving data access code with 199
 - test-driving with, vs. integration tests 243
 - what may be missed 221
 - writing 100, 104
 - writing with Hibernate 212
 - unit testing 36, 39, 185–186, 242, 350, 441
 - framework 36, 41, 108, 285
 - patterns 127
 - support for 295
 - unpredictability
 - See* predictability
 - URI *See* Uniform Resource Identifier
 - URL *See* Uniform Resource Locator
 - usability 5
 - use case 31
 - user action sequence
 - See* interaction sequence
 - user gesture 285–286
 - delegating 289
 - handling 287, 289
 - reacting to 298
 - responding to 315
 - simulating 291, 297
 - user input
 - capturing 283
 - user interface 13
 - abstract 404–405
 - abstract, testing through 404
 - acceptance testing 352
 - alternative 404
 - building with off-the-shelf components 304
 - bypassing 416
 - concept 404
 - console-based 412
 - depending on 403
 - designing for user 17
 - test-driving piece of 177
 - testing below UI 404
 - testing through 355, 412, 421
 - text-based 415
 - user story 31, 324, 331–332, 366, 398, 416
 - acceptance criteria for 328
 - and acceptance tests 327
 - benefits of 351
 - business value 335
 - conditions of satisfaction for 328
 - conveying what and why 329
 - dropping 347
 - estimating 345
 - estimating size 333
 - example of 326–327, 398
 - for programming libraries 412
 - format 325
 - implementing 343
 - introduction 325–327
 - meaning of 354
 - obsolete 344
 - power of storytelling 325
 - prioritizing 335, 345
 - running acceptance tests while implementing 403
 - scheduling 344, 346

user story (*continued*)
 selecting 335–336, 343
 size of 343
 splitting 345, 347
 splitting by detail 347
 splitting by function 347
 splitting into tasks 341
 splitting, strategy 348
 swapping 347
 technical risk 335
 writing 344–345
 writing tests for 336, 343

V

validation 105, 143, 441
 of user input 281
 validation rule
 verifying 426
 value 11, 102
 expected 374
 of exceptional scenarios 103
 of software 5
 of test 173
 source of 329
 varargs 83
 variability 60
 Variable class 96
 variable-length argument list 83
 Velocity *See* Apache Velocity
 Velocity template 180, 184
 VelocityContext 179–180
 verbosity 112, 120, 218, 268
 verification 29, 38, 108, 111,
 173, 346
 against internal model 282
 of conformance to
 specification 31
 of correctness 4
 of desired interaction 305
 of expectations 130
 version control 26, 224,
 230, 432–433
 VFS *See* Jakarta Commons VFS
 view 154, 283
 creating test-first 173–184
 implementation 288
 in model-view-controller
 155, 168, 173

logic 187, 283
 making thin 287–288, 290,
 297, 319
 synchronizing 299
 telling what to display 291
 view component 298
 adding and operating
 standard widgets 300–304
 associating gestures with
 coordinates 314–319
 behavior 298
 drawing custom
 graphics 304–314
 layout out design 298–300
 test-driving 297–319
 tools for testing 290–297
 view implementation 290
 view logic *See* view
 view object
See Model-View-Presenter
 view state 284
 view template 179, 184, 187,
 190, 192
 visual inspection 196, 281
 visual inspection *See also* manual
 testing
 visual object 283
 visual verification 173
 vocabulary 108, 402
 custom 361
 Vodde, Bas 106
 volume
 of data-access code 199
 of image data 310
 of test code 201, 211

W

waiting
 for thread 271
 Watir 424
 web application 37, 155–156,
 168, 173, 185–187, 406,
 413, 421
 acceptance test implementa-
 tion strategy 425
 implementing acceptance
 tests for 421–422
 testing 360
 writing acceptance
 tests for 333
 web browser 158–159, 173, 185,
 188, 192, 421
 controlling 421–422, 424–425
 simulating 421–422
 web container 154, 178, 189
 web container
See Servlet container
 web infrastructure 421
 web server 29, 420
 web service 23, 413
 web standard 421
 WebFixture 402
 WebWork 156
 Weinberg, Gerald M. 459
 white-box testing 400, 419
 Wicket 154, 156,
 185–188, 191–193
 adding components to
 application 190–193
 applications, unit-testing
 186, 193
 component 190
 interacting with
 components 193
 treating pages as classes 188
 WicketFilter class 187
 WicketTester class 189–190
 widget 185, 404
 capturing user input
 from 283–284
 child, creating 304
 color of 309
 exposing access to 419
 giving access to 297
 locating 417
 native 418
 sketch 298
 standard, adding to view
 component 300–304
 variety of 417
 wiki 360
 Wilson, Woodrow 444
 windowing system 418
 working software 7, 14, 19, 28,
 31–32, 98
 workspace 38, 55, 228
 write operation 226, 234, 282

X

- XDoclet 123
- XLS 360
- XML 127, 186, 198, 213, 237–239, 412
 - as source for data-driven testing 139
 - defining test data in 240
- XP *See* extreme programming
- XPath 183, 412
- XPCOM 424
- xUnit 36
- xunitpatterns.com 115, 128